АЛЕКСЕЙ ВАСИЛЬЕВ

Mathcad 13

НА ПРИМЕРАХ

ПРОСТЫЕ И ПАРАМЕТРИЧЕСКИЕ ПРОИЗВОДНЫЕ

КРАТНЫЕ И КРИВОЛИНЕЙНЫЕ ИНТЕГРАЛЫ

РЯДЫ И ИНТЕГРАЛЬНЫЕ ПРЕОБРАЗОВАНИЯ

ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ И СИСТЕМЫ

ПРОГРАММИРОВАНИЕ И СТАТИСТИКА

ГРАФИКА И АНИМАЦИЯ

+CD

bhv®

Алексей Васильев

Mathcad 13
НА ПРИМЕРАХ

Санкт-Петербург

«БХВ-Петербург»

2006

УДК 681.3.06
ББК 32.973.26-018.2
В19

Васильев А. Н.

В19 Mathcad 13 на примерах. — СПб.: БХВ-Петербург, 2006. — 528 с.: ил.

ISBN 5-94157-880-6

На конкретных примерах показаны возможности популярного математического пакета Mathcad 13. Базовые примеры подобраны с таким расчетом, чтобы они охватывали все основные направления в рамках классического университетского курса высшей математики, в том числе вычисление всевозможных производных, расчет двойных, тройных и криволинейных интегралов первого и второго рода, разложение функций в ряды, интегральные преобразования, решение дифференциальных уравнений, систем и многое другое. Рассматриваются и более сложные примеры, как из области высшей математики, так и комплексные физические задачи, подразумевающие привлечение всего арсенала утилит Mathcad. Описан графический интерфейс, методы работы с изображениями, создание анимации, построение пространственных кривых, поверхностей и диаграмм, методы программирования в среде Mathcad.

Прилагаемый компакт-диск содержит более 150 готовых к использованию примеров из книги.

Для студентов, преподавателей и инженеров

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Леонид Кочин*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 27.02.06.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 42,57.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-880-6

© Васильев А. Н., 2006
© Оформление, издательство "БХВ-Петербург", 2006

Оглавление

Введение **1**

Для кого эта книга 2
Структура книги 2
О версии пакета 3
Компакт-диск 3
О списке литературы 3
Благодарности 4
Обратная связь 4

Глава 1. Графический интерфейс пользователя **5**

Структура рабочего окна 5
Панель меню 7
  Меню *File* 8
  Меню *Edit* 11
  Меню *View* 12
  Меню *Insert* 14
  Меню *Format* 17
  Меню *Tools* 19
  Меню *Symbolics* 20
  Меню *Window* 22
  Меню *Help* 23
Панели инструментов 25
  Стандартная панель инструментов 25
  Панель форматирования 27
  Математическая панель 28
  Панели ресурсов и элементов управления 30
Настройка интерфейса пользователя 31
  Настройка панелей инструментов 31
  Линейка, масштаб и фон 35

Настройка колонтитулов ... 37
Разбивка на страницы ... 40
Использование групп ... 41
Основные настройки ... 43
Параметры документа ... 43
Настройки приложения ... 52

Глава 2. Основы работы с документами ... 59

Математические выражения и команды пользователя ... 59
Математические выражения ... 59
Пример 2.1. Сложение чисел ... 64
Пример 2.2. Вычисление сложного выражения ... 64
Пример 2.3. Инициализация переменной ... 65
Пример 2.4. Знак равенства ... 66
Способы инициализации переменных ... 66
Пример 2.5. Глобальное определение переменной ... 67
Работа с числами ... 70
Особенности ввода числовых значений ... 70
Пример 2.6. Научная нотация ... 71
Настройки форматирования численных данных ... 71
Пример 2.7. Числовые форматы ... 72
Пример 2.8. Инженерный формат ... 73
Пример 2.9. Различные форматы ... 75
Пример 2.10. Выделение целой части ... 77
Пример 2.11. Использование дробей ... 77
Использование комплексных чисел ... 78
Пример 2.12. Комплексные числа ... 79
Различные системы счисления ... 81
Пример 2.13. Двоичные числа ... 82
Пример 2.14. Двоичная система ... 82
Пример 2.15. Шестнадцатеричные и восьмеричные числа ... 83
Константы ... 85
Пример 2.16. Использование констант ... 85
Пример 2.17. Запись данных в файл ... 87
Работа с функциями ... 88
Вставка встроенных функций ... 88
Стандартные математические функции ... 91
Пример 2.18. Математические функции ... 93
Пример 2.19. Работа с комплексными числами ... 94
Специальные функции ... 94

Функции округления численных значений 98
Пример 2.20. Преобразование значений 98
Пример 2.21. Функции округления 99
Функции пользователя 100
Пример 2.22. Функции пользователя 100
Пример 2.23. Символьный результат 101
Вычисление выражений 102
Вычисление по диапазону значений 102
Пример 2.24. Диапазон значений 104
Пример 2.25. Функция двух переменных 105
Вычисление сумм 107
Пример 2.26. Вычисление суммы 107
Вычисление произведений 109
Пример 2.27. Вычисление произведений 109
Использование размерных единиц 110
Пример 2.28. Размерные величины 112
Пример 2.29. Преобразование размерностей 125
Пример 2.30. Определение размерностей 128
Пример 2.31. Переопределение размерностей 129
Матрицы и векторы 130
Создание массивов 131
Пример 2.32. Векторы и матрицы 132
Пример 2.33. Начальный индекс массива 134
Пример 2.34. Определение массива 135
Вложенные массивы 139
Пример 2.35. Вложенные массивы 139
Операции с массивами 142
Пример 2.36. Операции с массивами 143
Пример 2.37. Транспонирование матриц 144
Пример 2.38. Обращение матриц 145
Пример 2.39. Определитель матрицы 146
Пример 2.40. Векторное произведение 147
Пример 2.41. Сумма элементов 148
Пример 2.42. Процедура векторизации 149
Преобразование массивов 150
Пример 2.43. Выделение строк и столбцов 150
Функции для работы с массивами 153
Пример 2.44. Функция *matrix()* 153
Пример 2.45. Функции *stack()* и *augment()* 154
Пример 2.46. Функция *submatrix()* 155
Пример 2.47. Характеристики матриц 157

Решение уравнений и систем 157
Пример 2.48. Решение уравнений и систем 158
Пример 2.49. Решение нелинейной системы 159
Вычисление производных 160
Пример 2.50. Вычисление производных 161
Вычисление интегралов 162
Пример 2.51. Вычисление интеграла 163
Полезные советы и возможности 163
Редактирование выражений 164
Использование нестандартных названий 164
Пример 2.52. Операторы в названиях 165
Пример 2.53. Символы-операторы 165
Определение операторов пользователя 166
Пример 2.54. Операторы пользователя 166

Глава 3. Графика и визуализация результатов в Mathcad 169

Создание двумерных графиков 169
Пример 3.1. График функции 173
Пример 3.2. Графики нескольких функций 174
Настройка параметров графика 177
Пример 3.3. Способы отображения графиков 187
Пример 3.4. График с заголовком 190
Графики в полярных координатах 192
Пример 3.5. Спираль Архимеда 192
Основные методы работы с графиками 197
Отображение массивов данных 205
Пример 3.6. Отображение значений массива 205
Создание трехмерных графиков 207
Создание поверхностей 208
Пример 3.7. График функции двух переменных 208
Основные настройки трехмерного графика 211
Пример 3.8. Графики разных типов 212
Пример 3.9. График с заголовком 216
Пример 3.10. График в тумане 221
Пример 3.11. Закраска цветом 223
Отображение параметрических поверхностей 224
Пример 3.12. Эллипсоид 225
Пример 3.13. Геликоид 226
Пример 3.14. Винтовая линия 227
Пример 3.15. Конус 228
Пример 3.16. Спираль 229

Особенности построения трехмерных графиков 230
Пример 3.17. Контурные графики 230
Пример 3.18. Векторное поле.....232
Создание диаграмм..... 233
Пример 3.19. Диаграмма.....234
Отображение нескольких графиков..... 235
Пример 3.20. Поверхности 235
Пример 3.21. Поверхность и диаграмма 236
Пример 3.22. Конус и сфера 237
Пример 3.23. Тень от поверхности 238
Пример 3.24. Пространственные кривые 239
Работа с утилитой создания графиков 240
Работа с изображениями 242
Пример 3.25. Вставка изображения 242
Пример 3.26. Считывание изображения.....244
Пример 3.27. Создание изображений 248
Анимация..... 252
Пример 3.28. Создание анимации.....252

Глава 4. Программирование в Mathcad 263

Основы программирования в Mathcad 263
Пример 4.1. Дважды два 266
Логические операторы 267
Пример 4.2. Логические операторы.....269
Условный оператор..... 269
Пример 4.3. Условный оператор.....269
Пример 4.4. Функция с условным оператором.....270
Пример 4.5. Несколько условных операторов.....271
Пример 4.6. Использование условного оператора 271
Пример 4.7. Кусочно-непрерывная функция 273
Пример 4.8. Вложенные операторы.....274
Пример 4.9. Арифметическое выражение в условии.....276
Операторы циклов 277
Пример 4.10. Сумма квадратов 277
Пример 4.11. Сумма квадратов с ограничением 278
Пример 4.12. Еще один оператор цикла.....279
Пример 4.13. Использование инструкции *break*.....280
Рекурсия 281
Пример 4.14. Двойной факториал.....281
Пример 4.15. Вычисление суммы через рекурсию 282
Элементы управления 283
Внешние параметры элемента управления 286

Пример 4.16. Кнопка 289
Язык сценариев VBScript 289
Переменные и константы 290
Массивы 295
Функции VBScript 296
Операторы VBScript 302
Условные операторы VBScript 305
Операторы циклов VBScript 307
Процедуры и функции пользователя 308
Отображение диалоговых окон 308
Объекты, методы и свойства 310
Работа с программным кодом 315
Пример 4.17. Подсчет количества щелчков на кнопке 316
Пример 4.18. Изменение атрибутов кнопки 317
Свойства и методы элементов управления 319
Пример 4.19. Совместное использование элементов управления 323

Глава 5. Символьные вычисления 335

Методы дифференцирования и интегрирования 336
Вычисление производных 336
Пример 5.1. Производная от выражения 336
Пример 5.2. Символьная производная 338
Пример 5.3. Производная от функции 341
Пример 5.4. Производные высоких порядков 342
Интегрирование 344
Пример 5.5. Вычисление интегралов 344
Использование команд Maple 348
Работа с выражениями 350
Пример 5.6. Упрощение выражений 351
Пример 5.7. Упрощение при дополнительных условиях 353
Пример 5.8. Преобразование выражений 355
Пример 5.9. Коэффициенты полинома 356
Пример 5.10. Замена переменных 356
Пример 5.11. Разложение на простые дроби 356
Пример 5.12. Вычисление выражений 357
Символьные операции с матрицами 358
Пример 5.13. Операции с матрицами 358
Интегральные преобразования и разложение в ряды 360
Пример 5.14. Интегральные преобразования 361
Пример 5.15. Z-преобразование 365
Пример 5.16. Ряд Тейлора 366

Решение уравнений в символьном виде 367
Пример 5.17. Решение уравнений и систем 368
Суммирование рядов, вычисление границ и произведений 370
Пример 5.18. Вычисление бесконечных сумм и произведений 370
Пример 5.19. Вычисление пределов 371
Использование команд меню 372
Пример 5.20. Использование команд меню 373

Глава 6. Задачи высшей математики 377

Вычисление производных 377
Производная явной функции 377
Производная от неявно заданной функции 379
Параметрическая производная 382
Производная функции, заданной в полярных координатах 384
Производные высоких порядков 388
Вычисление интегралов 390
Пример 6.1. Определенные и неопределенные интегралы 390
Криволинейные интегралы 392
Двойные интегралы 401
Тройные интегралы 407
Ряды Фурье 412
Пример 6.2. Ряд Фурье 412
Дифференциальные уравнения 416

Глава 7. Численные и статистические методы 435

Решение алгебраических уравнений и систем 435
Алгебраические уравнения 435
Пример 7.1. Решение уравнений 436
Пример 7.2. Корни полиномов 445
Системы уравнений 446
Пример 7.3. Решение систем уравнений 446
Пример 7.4. Система линейных уравнений 447
Интерполяция функций 452
Интерполяция сплайнами 453
Интерполяционный полином Лагранжа 457
Численное решение дифференциальных уравнений 460
Пример 7.5. Решение дифференциального уравнения 460
Пример 7.6. Решение системы дифференциальных уравнений 462
Пример 7.7. Сведение уравнения к системе 463
Статистический анализ 469

Основные понятия теории вероятностей 469
Пример 7.8. Основные распределения 470
Пример 7.9. Числовые характеристики случайных величин 479
Элементарные задачи математической статистики..... 484
Пример 7.10. Выборочные статистические характеристики 485
Пример 7.11. Корреляционные характеристики 489
Регрессионный анализ 490
Пример 7.12. Линейная регрессионная модель 491
Пример 7.13. Медианный метод 492
Пример 7.14. Полиномиальная регрессия.....493
Пример 7.15. Регрессия и интерполяция.....494
Пример 7.16. Регрессия полиномами второй степени 496
Пример 7.17. Тригонометрическая регрессия 498
Пример 7.18. Регрессия на основе комбинации функций 500
Пример 7.19. Регрессия общего вида 501
Пример 7.20. Сглаживание данных 503

Заключение 505

Приложение. Описание компакт-диска 507

Литература..... 509

Предметный указатель..... 511

Введение

Первое знакомство с математикой (и ее материальными атрибутами) у каждого человека происходило по-своему. Для многих это были счетные палочки, с помощью которых выполнялись элементарные математические операции сложения и вычитания. Далее на сцену выходили более замысловатые механизмы, вроде логарифмической линейки. На фоне калькуляторов все это кажется детской игрушкой особенно сейчас, когда практически любое рабочее место, причем не только ученого-исследователя, но и простого клерка, оснащено персональным компьютером. На заре же развития вычислительной техники те возможности, которыми в настоящее время обладает человечество, выглядели просто фантастикой. И если в области численных расчетов исключительную роль компьютеров предугадать было несложно, возможность их использования для анализа в символьном виде не представлялась столь очевидной. Реальность, тем не менее, превзошла все ожидания.

На сегодня существует несколько математических пакетов, с помощью которых быстро, просто и эффективно выполняются сложные математические расчеты, причем даже в символьном виде. Здесь следует обратить внимание читателя на такие средства вычислений, как Maple, Mathematica и Mathcad. Пакеты эти выделены именно потому, что в них на достаточно серьезном уровне поддерживаются символьные вычисления. Что касается пакета Mathcad, которому посвящена данная книга, то в нем символьные вычисления являются скорее данью моде и ответом на современные вызовы рынка. Однако это ни в коей мере не умаляет возможностей Mathcad, скорее наоборот. Он неизменно популярен, особенно среди студентов и инженеров (по-видимому, в силу простоты взаимодействия пользователя с приложением и прозрачности выполняемых операций). Возможности приложения постоянно расширяются, в силу чего Mathcad все чаще используется не только в физико-инженерных расчетах, но и при решении задач экономики, в финансовых расчетах, в социологии и биологии (список можно продолжить).

Для кого эта книга

Читателю предлагается книга, в которой на простых и более сложных примерах и задачах описываются возможности Mathcad и методы работы с ним. Представленные задачи в основном математические или физические. Однако они могут служить основой для решения задач из других областей.

Примеры разного уровня сложности рассматриваются не случайно. С одной стороны, простые примеры будут полезны тем, кто пока что плохо знаком с высшей математикой. Простота примеров позволит им сосредоточиться на особенностях работы приложения, а не математической сути решаемой задачи. С другой стороны, читатель, обладающий достаточными познаниями в области математического анализа и дифференциальных уравнений, сможет воочию убедиться, что Mathcad исключительно полезен при решении сложных задач. Тут, однако, сразу отметим, что если решение той или иной задачи требует специальных познаний в области высшей математики, в книге приводится вся необходимая теоретическая информация по этому вопросу. Так что читателю не придется обращаться к специальной литературе (хотя это полезно делать).

Таким образом, книга может быть полезной специалистам, имеющим дело с прикладными математическими расчетами. В основном это научные сотрудники, инженеры, студенты физико-математических, инженерных и экономических вузов, а также все, кто интересуется современными вычислительными средствами.

Структура книги

Книга состоит из семи глав. Условно ее можно разделить на две части. Первые три главы посвящены описанию графического интерфейса пользователя (первая глава), основным элементарным операциям, которые могут выполняться с помощью приложения (вторая глава), методам работы с графикой (третья глава). Эти три главы содержат базовые сведения, необходимые для успешной работы с приложением. Четвертая глава посвящена методам программирования с помощью Mathcad. В некотором смысле эта глава переходная. Пятая, шестая и седьмая главы целиком посвящены решению практических задач (в основном из курса высшей математики). В пятой (относительно небольшой) главе рассматриваются элементарные методы выполнения символьных вычислений. Более сложные задачи решаются в шестой главе. Седьмая глава посвящена численным методам и статистическому анализу.

Каждая глава (кроме первой) включает в себя примеры и задачи (возможности приложения иллюстрируются в процессе их решения). Разделение прак-

тических материалов на примеры и задачи достаточно условное. Как правило, к задачам отнесены комплексные проблемы, решение которых подразумевает использование нескольких методологических подходов (хотя это и не всегда так). На прилагаемом к книге компакт-диске содержатся файлы с примерами и задачами из книги (см. *приложение 1*). Каждый пример имеет название, в точности совпадающее с названием соответствующего файла на компакт-диске. Названия задач несколько отличаются от тех, что в книге, однако соответствие можно установить по номерам. Нумерация примеров и задач сквозная в пределах главы. Примеры и задачи нумеруются независимо друг от друга.

О версии пакета

Основная версия пакета, рассмотренная здесь — Mathcad 13 (на момент написания книги это последняя версия). Тем не менее, все примеры подбирались с таким расчетом, чтобы они решались теми же методами и в более ранних версиях приложения. Кроме того, *глава 5* содержит раздел (использование команд Maple), описывающий особенности предыдущей версии (Mathcad 11). Изложенный в книге материал в минимальной степени ориентирован на особенности, присущие только 13-й версии Mathcad (за исключением, пожалуй, некоторых элементов графического интерфейса пользователя).

Все же на одно обстоятельство следует обратить внимание читателя. Дело в том, что записанные на прилагаемом к книге компакт-диске файлы сохранялись как документы Mathcad 13, поэтому при попытке открыть их с помощью более ранней версии приложения могут возникнуть проблемы.

Компакт-диск

Как отмечалось, к книге прилагается компакт-диск с примерами из книги. Это должно облегчить читателю процесс усвоения материала. Для удобства все примеры разбиты по главам, названия файлов однозначно указывают, о каком примере идет речь. Подробное описание компакт-диска приведено в *приложении 1*.

О списке литературы

В конце книги приведен краткий список литературы, которая может быть полезной читателю в процессе изучения Mathcad или в дальнейшем при работе с приложением. Кроме книг непосредственно по приложению, там представлены также некоторые специальные издания по физике и высшей

математике. Следует понимать, что список этот, в силу понятных и объективных причин, не является исчерпывающим (в него, например, не включены англоязычные издания, а книги по разным версиям пакета Mathcad указаны, начиная с Mathcad 11). Читателю предоставляется полная свобода выбора источников информации. Отметим лишь, что в последнее время книг и статей по системам компьютерной математики выходит все больше и больше, уже можно выделить целые направления и школы в этой области.

Благодарности

Мои студенческие годы совпали с тем временем, когда рушилась большая страна, вокруг был жуткий хаос и неопределенность. Наука и тогда, и сейчас переживает не самые лучшие времена. Тем не менее, я никогда не испытывал недостатка ни в книгах, ни во всем остальном, что так или иначе относится к жизни студента (и не только). В результате это позволило получить образование и заниматься в жизни тем, что мне нравится. В этом всецело заслуга моих родителей, которым я хочу выразить свою искреннюю и огромную благодарность этой скромной книгой.

Обратная связь

Автор будет благодарен читателям за критические замечания по поводу книги. Адрес электронной почты: `vasilev@univ.kiev.ua`. Можно также выйти на сайт `http://www.phys.univ.kiev.ua/theory/vasilev.php`. К сожалению, лично ответить на все письма в силу понятных причин автор не обещает, однако мнения читателей по поводу того, что в книге можно было бы усовершенствовать и на какие темы обратить внимание, весьма важны. Хочется заранее выразить благодарность всем, кто откликнется на эту просьбу.

Глава 1

Графический интерфейс пользователя

В этой главе речь пойдет о графическом интерфейсе пользователя. Будут описаны основные элементы интерфейса и принципы работы с приложением в целом. Помимо этого, в главе обсуждаются вопросы, связанные с настройкой интерфейса Mathcad и методами форматирования данных в рабочих документах.

Структура рабочего окна

При запуске Mathcad открывается окно приложения, вид которого показан на рис. 1.1.

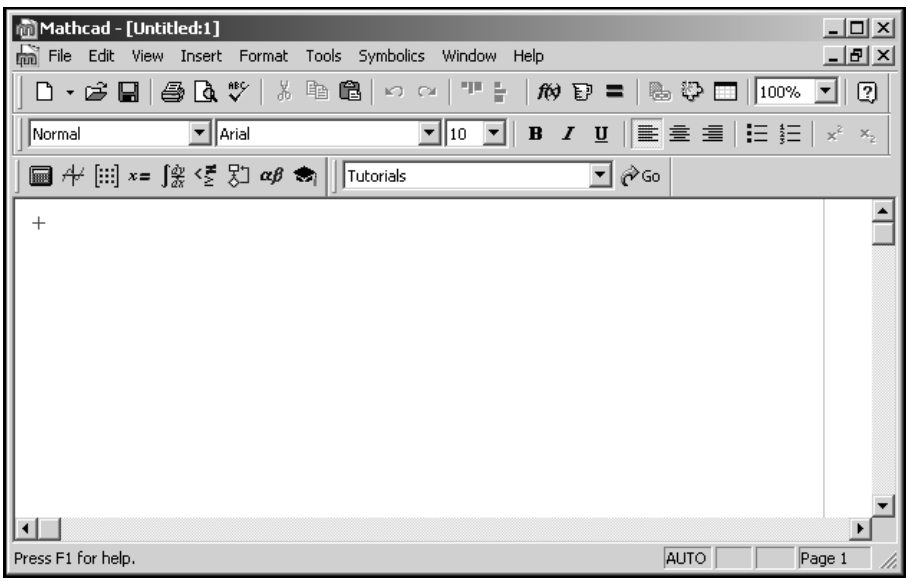


Рис. 1.1. Рабочее окно приложения Mathcad

В строке заголовка, как и во всех приложениях, работающих с операционной системой Windows, за названием приложения (Mathcad) следует имя рабочего документа. В данном случае это присваиваемое по умолчанию название для нового рабочего документа (**Untitled: 1**). После строки заголовка, внизу, располагается панель меню (рис. 1.2).

Рис. 1.2. Панель меню

Панель содержит девять пунктов (**File**, **Edit**, **View**, **Insert**, **Format**, **Tools**, **Symbolics**, **Window** и **Help**). Несмотря на то, что заголовки меню практически полностью совпадают с их аналогами в прочих популярных приложениях (например, Excel или Word), команды, собранные в них, во многом специфичны для Mathcad, хотя, безусловно, существует и много общих свойств, особенно для редактирования рабочих документов и настройки графического интерфейса пользователя. В правой части панели меню можно видеть три стандартных для Windows элемента управления, позволяющих сворачивать, разворачивать и закрывать рабочее окно документа. Пиктограмма слева служит тем же целям — однако в этом случае открывается список с соответствующими командами.

Стандартная панель инструментов (рис. 1.3) располагается под панелью меню.

Рис. 1.3. Стандартная панель инструментов

На этой панели собраны кнопки для выполнения основных команд, наиболее часто используемых при работе с Mathcad. Эти команды доступны также через основное меню. Однако панель инструментов существенно упрощает работу с приложением в целом.

Помимо стандартной панели инструментов по умолчанию отображается и панель форматирования (рис. 1.4). Команды, представленные здесь, имеют принципиально иное назначение. Если на общей панели собраны в основном команды, связанные с манипулированием рабочим документом в целом или базовыми инструментами, необходимыми в работе, то панель форматирования может быть полезна в процессе оформления данных (в основном текстовых) в документе.

Рис. 1.4. Панель форматирования

Если панель форматирования и общая панель инструментов (в несколько меньшей степени) имеют (по крайней мере, внешне) для большинства пользователей Windows достаточно знакомый вид, то математическая панель (рис. 1.5) является всецело "детищем" приложения Mathcad.

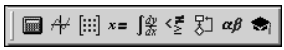


Рис. 1.5. Математическая панель

Щелчок на одной из кнопок этой панели приводит к тому, что будет отображена соответствующая палитра. Каждая палитра имеет свое назначение, а их использование существенно упрощает процесс взаимодействия пользователя с системой. В основном палитры полезны при вычислениях и преобразовании выражений. Справа (по умолчанию) от математической панели размещается панель ресурсов (рис. 1.6).

Рис. 1.6. Панель ресурсов

С помощью элементов этой панели (их, собственно, два — раскрывающийся список и кнопка выполнения перехода) можно активизировать дополнительные ресурсы Mathcad. Большая белая область внизу под перечисленными панелями (см. рис. 1.1) является непосредственно рабочей областью, где пользователем вводятся команды. Там же отображается результат их выполнения. В нижней части рабочего окна, под рабочей областью, можно видеть строку состояния (рис. 1.7).

Рис. 1.7. Строка состояния

На этом основные элементы рабочего окна (если не считать полосы прокрутки) исчерпываются. Не следует, однако, забывать, что такой вид рабочее окно имеет по умолчанию. Пользователь может самостоятельно настраивать внешний вид рабочего окна. В зависимости от настроек окно может иметь свои особенности. Далее панели инструментов и меню описываются подробнее.

Панель меню

Как отмечалось, панель содержит девять меню. В каждом из них собраны команды, посредством которых, по большому счету, и реализуется весь спектр функциональных возможностей Mathcad. Команды сгруппированы

в зависимости от их назначения. Таким образом, команды каждого меню объединены между собой тематически, хотя такая их общность в некоторых случаях выглядит несколько условной.

Меню *File*

В меню **File** представлены команды, полезные при работе с документом в целом. При щелчке мышью на этом меню открывается список доступных команд. Рабочее окно при этом выглядит так, как показано на рис. 1.8.

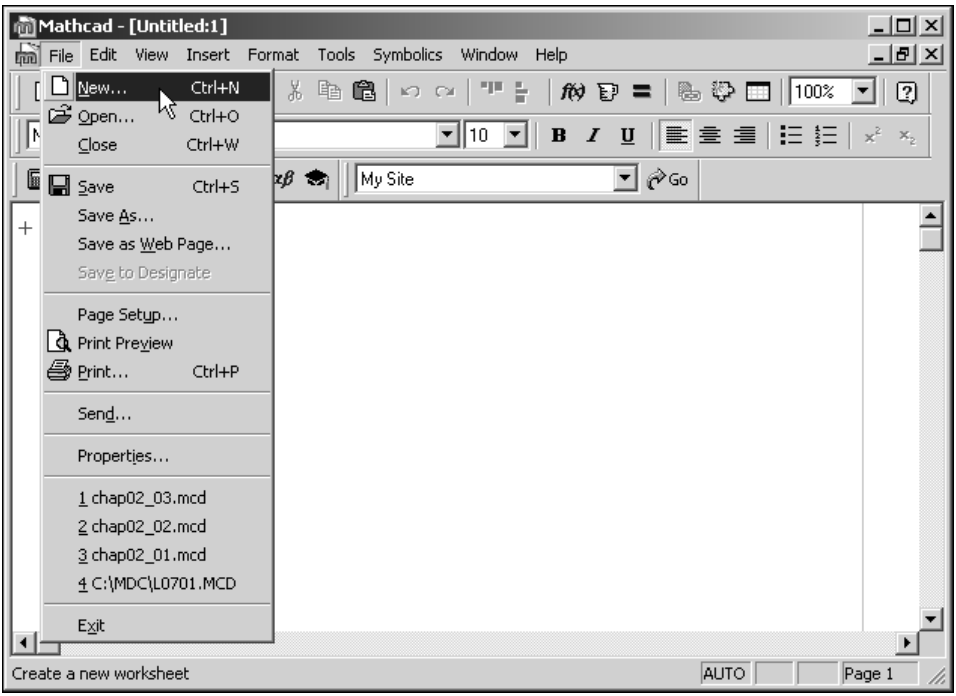


Рис. 1.8. Содержимое меню **File**

Помимо непосредственно названия команд, в раскрывающемся списке для некоторых их них справа указаны комбинации клавиш, нажатие на которые приводит к выполнению команды (так называемые горячие клавиши). Это удобно, поскольку позволяет выполнить команду без непосредственного обращения к панели меню. Кроме того, для некоторых команд слева указаны пиктограммы, позволяющие безошибочно найти кнопку на панели инструментов, с помощью которой команда может быть выполнена. Назначение команд меню **File** описано в табл. 1.1. Многие из них имеют достаточно точные аналоги в других приложениях и поэтому в особых комментариях не нуждаются.

Таблица 1.1. *Команды меню* ***File***

Команда	Описание
New	Команда создания нового рабочего документа
Open	С помощью этой команды можно открыть ранее созданный документ. В результате ее выполнения открывается диалоговое окно, с помощью которого непосредственно и осуществляется выбор нужного файла
Close	Выполнение этой команды приводит к закрытию текущего рабочего документа
Save	Команда сохранения рабочего документа. Если документ сохраняется впервые, будет выведено диалоговое окно, в котором следует указать название файла документа и место его сохранения. Если документ ранее сохранялся, то выполнение этой команды приведет к сохранению изменений после последнего сохранения — и никаких диалоговых окон
Save As	Команда позволяет сохранить копию текущего рабочего документа. При этом исходный документ не изменяется. При выборе этой команды открывается такое же диалоговое окно, как и при первом сохранении документа. Помимо этого, можно указать тип формата для сохраняемого файла (впрочем, как и при первом сохранении документа)
Save as Web Page	Сохранение рабочего документа в качестве Web-страницы
Save to Designate	Сохранение рабочего документа в репозиторий
Page setup	Настройки параметров страницы. Они выполняются обычно при подготовке документа к печати
Print Preview	Предварительный просмотр документа перед выводом его на печать
Print	Вывод на печать рабочего документа
Send	Отправка документа в качестве вложения по электронной почте
Properties	В результате выбора команды открывается диалоговое окно **File Properties**. В этом диалоговом окне на трех вкладках **Summary**, **Custom** и **XML Options** собрана вся основная информация о рабочем документе
Список файлов	В этой вкладке содержится список документов, которые открывались последними. Чтобы открыть один из них, достаточно щелкнуть на нем мышью
Exit	Завершение работы с приложением Mathcad

Использование команды **New** создания нового рабочего документа требует некоторых комментариев. Дело в том, что по сравнению с прочими приложениями, процесс создания нового документа в последних версиях Mathcad имеет свои особенности. В частности, при выполнении указанной команды открывается диалоговое окно **New** (рис. 1.9), в котором представлен список доступных в Mathcad шаблонов для создания на их основе документов. По умолчанию установлен шаблон **Normal**. В этом случае создается стандартный пустой документ Mathcad.

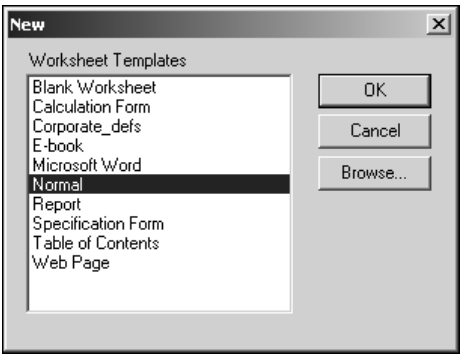


Рис. 1.9. Окно создания нового документа

В шаблоне документа определяются параметры текста и математические стили, типы заголовков и колонтитулы, значения встроенных переменных и прочие аналогичные параметры. Предлагаемые по умолчанию шаблоны Mathcad позволяют, помимо стандартного рабочего документа, создавать документы специального типа (например, отчеты). В этом случае пользователю, при создании документа, сразу предлагаются (в рамках текущего документа) готовые структурные элементы, которые можно редактировать по своему усмотрению. Следует, однако, отметить, что обычно подобного рода документы приходится создавать в соответствии с теми требованиями, которые предъявляются пользователем при решении конкретной прикладной задачи. Достаточно редко данный вопрос может быть решен с помощью предлагаемых шаблонов, и приходится создавать собственные. Впрочем, это делается весьма просто: документ сохраняется в качестве шаблона командой **File | Save As** (файл сохраняется с расширением xmct). Если сохранить шаблон в каталоге **Template** (это подкаталог того каталога, где размещена система Mathcad), то данный шаблон будет предлагаться в списке доступных шаблонов при выборе команды **File | New**.

Меню *Edit*

В меню **Edit** (рис. 1.10) собраны команды, позволяющие эффективно редактировать данные в рабочих документах Mathcad.

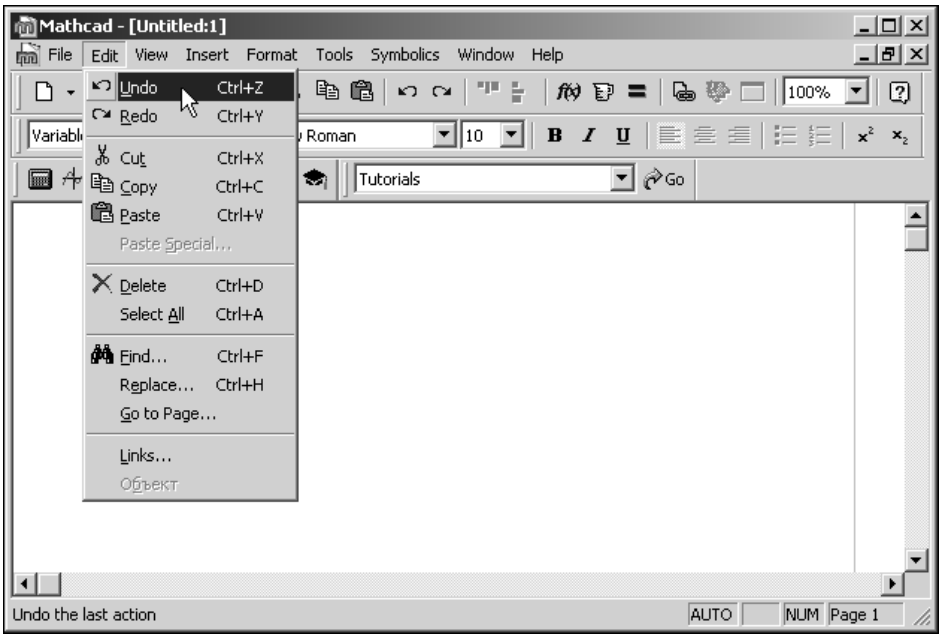


Рис. 1.10. Содержимое меню **Edit**

Команды меню **Edit** описаны в табл. 1.2.

Таблица 1.2. *Команды меню* ***Edit***

Команда	Описание
Undo	Команда отмены последнего действия. По умолчанию можно отменить до 100 действий. Количество запоминаемых действий (т. е. доступных для отмены) может меняться пользователем (команда **Tools | Preferences**)
Redo	Команда повторения отмененного действия
Cut	Выделенный фрагмент удаляется и копируется в буфер обмена
Copy	Копирование выделенного фрагмента в буфер обмена
Paste	Вставка содержимого буфера обмена в место размещения курсора

Таблица 1.2 *(окончание)*

Команда	Описание
Paste Special	Вставка содержимого буфера обмена в место размещения курсора в специальном формате. При выборе команды открывается диалоговое окно, которое позволяет выбрать формат, в котором будет вставлен записанный в буфере обмена объект
Delete	Удаление выделенного объекта
Select All	Выделение всех объектов в рабочем документе
Find	В результате выполнения команды открывается диалоговое окно, позволяющее выполнять поиск в рабочем документе указанных текстовых фрагментов, символов или величин
Replace	Команда поиска и замены найденных элементов (на тот, что указан пользователем). При выполнении команды открывается диалоговое окно, посредством которого и осуществляется взаимодействие пользователя и системы в процессе поиска и замены элементов в рабочем документе
Go to Page	Переход к странице, которую указывает пользователь в открывающемся диалоговом окне
Links	Команда позволяет редактировать ссылки в рабочем документе. Вставка ссылки может быть выполнена с помощью команды **Edit	Paste Special**
Object	Команда редактирования вставленных в документ объектов. Если таких объектов в документе нет, команда недоступна

Более подробно некоторые из перечисленных команд описываются далее в контексте рассматриваемых задач.

Меню *View*

В меню **View** (рис. 1.11) представлены в основном те команды, которые позволяют выполнять настройку внешнего вида рабочего окна приложения Mathcad.

Команды этого меню кратко описаны в табл. 1.3.

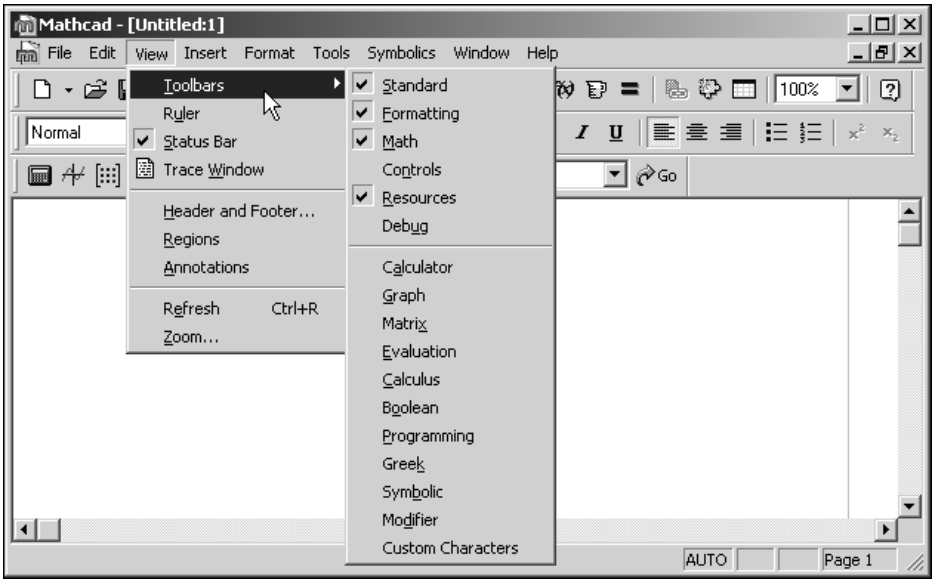


Рис. 1.11. Команды меню **View**

Таблица 1.3. *Команды меню* ***View***

Команда	Описание
Toolbars	Подменю позволяет отображать всевозможные панели и палитры в рабочем листе. Список доступных панелей можно увидеть, если навести курсор мыши на это подменю. Чтобы отобразить ту или иную панель, возле ее названия следует поставить флажок. Если флажок убрать, панель отображаться не будет
Ruler	Команда отображения линейки. Если возле этого пункта меню поставить флажок, в верхней части рабочей области (сразу под панелями) будет отображена линейка
Status Bar	Команда отображения строки состояния. Чтобы строка состояния была отображена в рабочем листе, у этого пункта меню следует поставить флажок
Trace Window	Команда отображения окна отладки программ
Header and Footer	С помощью этой команды можно задавать верхние и нижние колонтитулы рабочих листов. Эти настройки играют роль только при выводе документов на печать. При вызове команды открывается диалоговое окно, в котором и задаются колонтитулы

Таблица 1.3 *(окончание)*

Команда	Описание
Regions	Команда перехода в режим отображения фона. В этом режиме фон документа отображается серым цветом, за исключением тех областей, где размещены команды пользователя. В областях выделения команд отображается специальная точка — она определяет то место, где непосредственно размещена команда в рабочем листе (ее *якорь*). Чтобы отменить режим, следует повторно вызвать команду
Annotations	Команда перехода в режим выделения выражений с комментариями
Refresh	Команда обновления экрана
Zoom	Команда выбора масштаба отображения. Открывается диалоговое окно, в котором можно выбрать или установить масштаб отображения

Поскольку в Mathcad в процессе работы с документом как таковой разбивки рабочей области на листы нет (она имеет место только при выводе на печать), это обусловливает некоторую специфику в работе с приложением (в плане настройки внешнего вида документа). Особенности настройки вида рабочего документа описываются далее в этой главе.

Меню *Insert*

Следующее после **View** — меню **Insert** (рис. 1.12). Здесь собраны команды, полезные при работе с различными объектами Mathcad.

Как несложно заметить, не все команды этого меню доступны при создании нового документа. Данное замечание относится к команде вставки математического поля (**Math Region**) и гиперссылки (**Hyperlink**). Тем не менее, они будут доступны в процессе работы (в зависимости от выполняемых пользователем действий, когда вставка соответствующих объектов будет возможна).

В табл. 1.4 представлены и кратко описаны команды меню **Insert**.

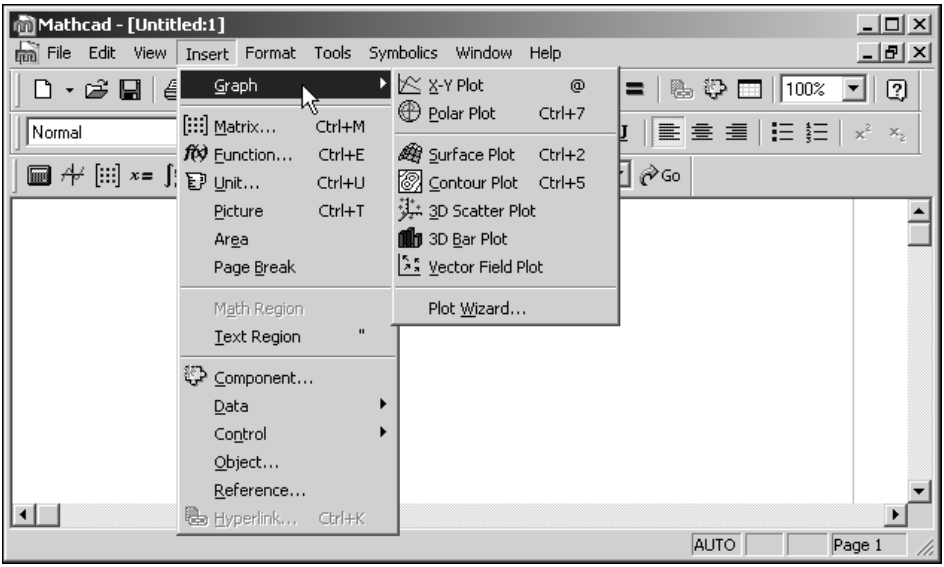


Рис. 1.12. Команды меню **Insert**

Таблица 1.4. *Команды меню* ***Insert***

Команда	Описание
Graph	Подменю содержит команды вставки всевозможных графиков, диаграмм, поверхностей. Подробнее эти команды описываются в *главе 3*, которая посвящена целиком и полностью работе с графикой в Mathcad
Matrix	Команда вставки матрицы. После ее вызова открывается диалоговое окно, в котором следует указать число строк и столбцов матрицы. Методы работы с матрицами описываются в *главе 2*
Function	Вставка функции. Открывается диалоговое окно, в котором следует выбрать соответствующую функцию. Если синтаксис функции пользователю известен, можно набрать название функции с помощью клавиатуры. Методы работы с функциями рассмотрены в *главах 2*, *4 — 7*
Unit	Вставка единиц размерности для физических величин. Обычно размерные величины используются при решении физических и инженерных задач
Picture	Команда вставки рисунка
Area	Вставка группы в рабочий лист. В этом случае в рабочем документе выделяется горизонтальная область. В этой области можно вводить команды, как и в обычном рабочем документе. Удобство состоит в том, что такую группу можно свернуть (тогда ее содержимое не отображается), а потом развернуть. Режим удобен при работе с большими документами

Таблица 1.4 *(окончание)*

Команда	Описание
Page Break	Команда перехода на новую страницу. Эффект имеет место при выводе документа на печать
Math Region	Вставка математического поля в тексте
Text Region	Вставка текстового поля
Component	Вставка компонента в рабочий документ. Запускается специальная утилита, с помощью которой в диалоговом режиме можно выбрать тип вставляемого объекта
Data	Подменю позволяет экспортировать и импортировать данные
Control	Подменю содержит команды вставки в рабочий документ элементов управления (кнопок, переключателей, текстовых полей и т. д.), которые позволяют выполнять всевозможные действия с элементами рабочего документа. С этой целью используется программный код VBScript или JScript. Подробнее вопросы работы с элементами управления описываются в последующих главах
Object	После вызова команды открывается диалоговое окно вставки объекта OLE в рабочий документ
Reference	С помощью этой команды в рабочем документе можно выполнить ссылку на файл Mathcad. Впоследствии этот файл (по размещенной в рабочем документе ссылке) можно будет открыть из текущего документа. При выборе команды открывается диалоговое окно выбора файла, с помощью которого можно найти нужный файл в каталогах системы
Hyperlink	Команда выполнения гиперссылки. Команда активна при работе с текстовым полем. После ее вызова открывается диалоговое окно, позволяющее, во-первых, определить файл (на локальном компьютере или в сети), который будет открываться при двойном щелчке на гиперссылке (текстовом поле), а во-вторых, выполнить ряд дополнительных настроек (например, задать текстовое сообщение, которое будет появляться в строке состояния при наведении курсора мыши на гиперссылку)

В данном меню представлены, несомненно, многие базовые команды, определяющие функциональность всего приложения. Описаны они были очень кратко с той лишь целью, чтобы можно было составить общее представление о них. Детальное описание некоторых команд приводится в книге в тех случаях, если это актуально в контексте рассматриваемых примеров и задач.

Меню *Format*

В меню **Format** (рис. 1.13) собраны утилиты форматирования объектов и элементов в рабочих документах.

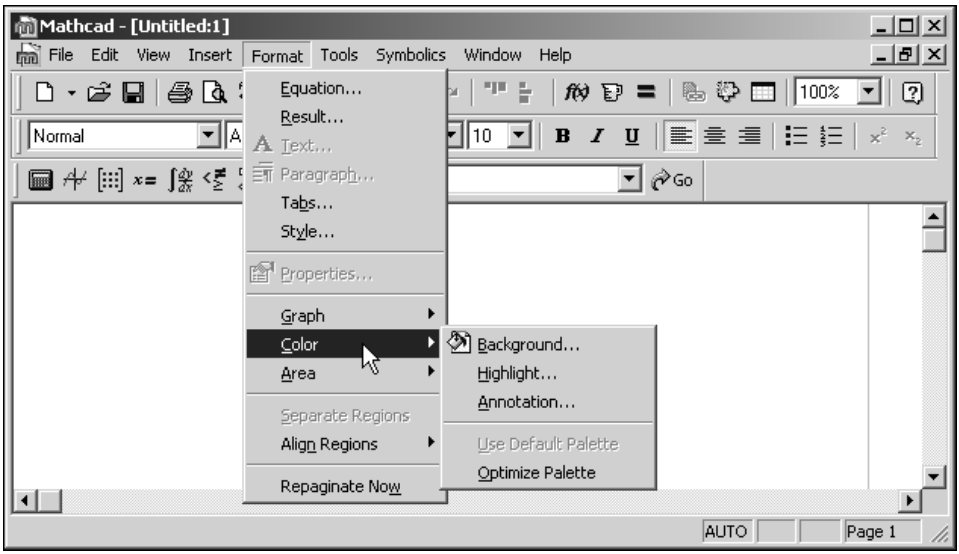


Рис. 1.13. Команды меню **Format**

Краткое описание команд, представленных в этом меню, можно найти в табл. 1.5.

Таблица 1.5. *Команды меню* ***Format***

Команда	Описание
Equation	Команда позволяет выполнять настройки для математических стилей (размер и тип шрифта, цвет и т. д.). Эти стили служат для отображения переменных и констант в рабочих документах. Изменения в стиле применяются ко всем элементам, использующим данный стиль. Помимо этого, можно создавать собственные стили пользователя
Result	Команда выполнения настроек для отображения численных значений в рабочем документе. Настройки выполняются в диалоговом окне, которое открывается при выборе команды
Text	Команда определения формата текста в текстовых полях. Определяется тип шрифта, размер, цвет, а также ряд других параметров и эффектов выделения текста

Таблица 1.5 *(окончание)*

Команда	Описание
Paragraph	Команда определения формата параграфа. Задаются отступ, выравнивание, нумерация и т. п.
Tabs	Команда определения точек переходов маркера в рабочем документе при нажатии клавиши <Tab>. При использовании таких меток разумно отображать линейку (команда **View** ▏ **Ruler**)
Style	В результате выполнения команды открывается диалоговое окно **Text Style**, с помощью которого можно либо применить к фрагменту стиль, либо создать новый
Properties	В результате выполнения команды открывается диалоговое окно. В этом диалоговом окне задается ряд весьма полезных параметров и настроек, как, например, выделение диапазона цветом, отображение рамки вокруг рисунка, можно также выбрать ряд режимов отображения данных и т. п.
Graph	Подменю содержит ряд команд, которые позволяют выполнить настройку основных, установленных по умолчанию, параметров графиков (двумерных, полярных, трехмерных) таких, как толщина, тип и цвет линий, координатные оси, координатные линии и т. п.
Color	Подменю содержит команды, позволяющие выполнить цветовые настройки рабочих листов. В частности, здесь можно задать цвет фона рабочей области документа (команда **Format** ▏ **Color** ▏ **Background**)
Area	Подменю содержит команды блокировки и управления областями. Такие области можно сворачивать, разворачивать или блокировать (ограничивать доступ к ним, например, с помощью пароля) и, разумеется, отменять ранее установленные блокировки
Separate Regions	Команда разделения диапазонов. Необходима в тех случаях, когда в рабочем документе формулы или текст накладываются друг на друга, что вызывает очевидные неудобства
Align Regions	Команда выравнивания диапазонов. Диапазоны могут выравниваться по горизонтали и вертикали
Repaginate Now	Команда корректировки разбивки на страницы. Такая корректировка необходима для того, чтобы диапазоны с данными в рабочем документе не попадали на линию раздела страниц. Эта команда выполняется автоматически при предварительном просмотре документа перед печатью и при выводе документа на печать

Команды меню **Format** обычно применяются при оформлении документов Mathcad.

Меню *Tools*

В меню **Tools** содержатся команды вызова утилит общего характера, настройки параметров документа, опции выполнения вычислений и т. п. Развернутое меню представлено на рис. 1.14.

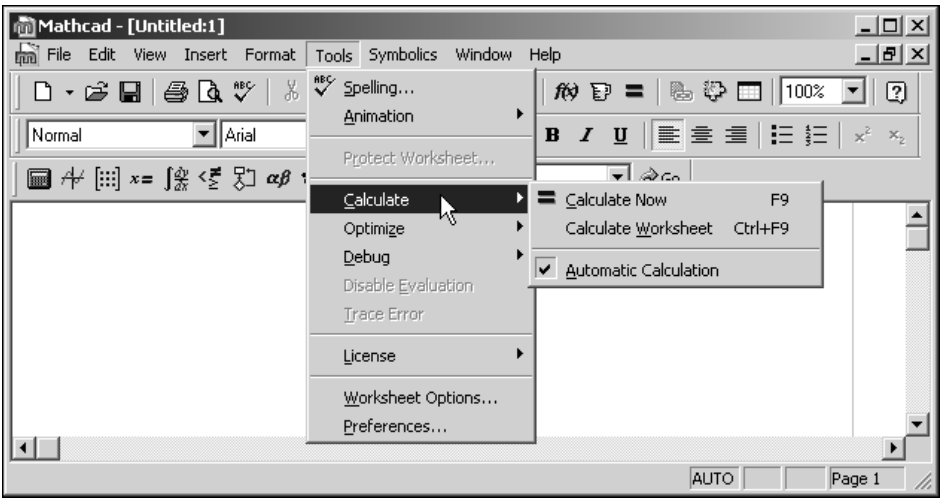


Рис. 1.14. Команды меню **Tools**

В табл. 1.6 можно найти описание команд и подменю меню **Tools**.

Таблица 1.6. *Команды меню* ***Tools***

Команда	Описание
Spelling	Команда проверки орфографии
Animation	Подменю для создания и просмотра анимации. Работа с анимацией подробнее объясняется в *главе 3*
Protect Worksheet	Команда защиты рабочего документа от несанкционированного доступа
Calculate	В подменю представлены команды, активизирующие вычисления в рабочем документе. Эти команды разумно использовать в том случае, если отключен режим автоматического вычисления данных в документе (этот режим включен по умолчанию — установлен флажок у команды **Automatic Calculation** из данного подменю)

Таблица 1.6 *(окончание)*

Команда	Описание
Optimize	Подменю с командами оптимизации. Здесь включается и выключается режим оптимизации символьных вычислений. Этот режим может быть применен как ко всему документу, так и к отдельному выражению
Debug	Подменю с командами отладки программного кода
Disable Evaluation	Команда деактивации (активизации) вычисления выражения. Если выражение деактивировано (то есть при вычислениях в рабочем листе оно игнорируется), команда заменяется на **Enable Evaluation**
Trace Error	Команда отслеживания ошибок
Worksheet Options	Запуск одноименного диалогового окна, в котором можно задать общие параметры системы
Preferences	Настройка дополнительных параметров системы. Открывается одноименное с командой диалоговое окно

Перечисленные в табл. 1.6 команды частично описываются в разделах, посвященных выполнению общих настроек системы, а также в *главах 2* и *5*, где речь в основном идет о вычислениях, в том числе и символьных.

Меню *Symbolics*

В меню **Symbolics** представлены команды, которые крайне важны при выполнении символьных вычислений. Команды этого меню показаны на рис. 1.15 и кратко описаны в табл. 1.7.

Таблица 1.7. *Команды меню* ***Symbolics***

Команда	Описание
Evaluate	Подменю позволяет преобразовывать всевозможные математические выражения, включая интегралы и производные. Это подменю содержит три команды: **Symbolically** (для выполнения преобразований в символьном виде), **Floating Point** (в этом случае при символьных преобразованиях возвращается, если это возможно, численный результат) и **Complex** (преобразования на множестве комплексных чисел)
Simplify	Команда упрощения выражений
Expand	Команда раскрытия скобок в суммах, произведениях и т. п.

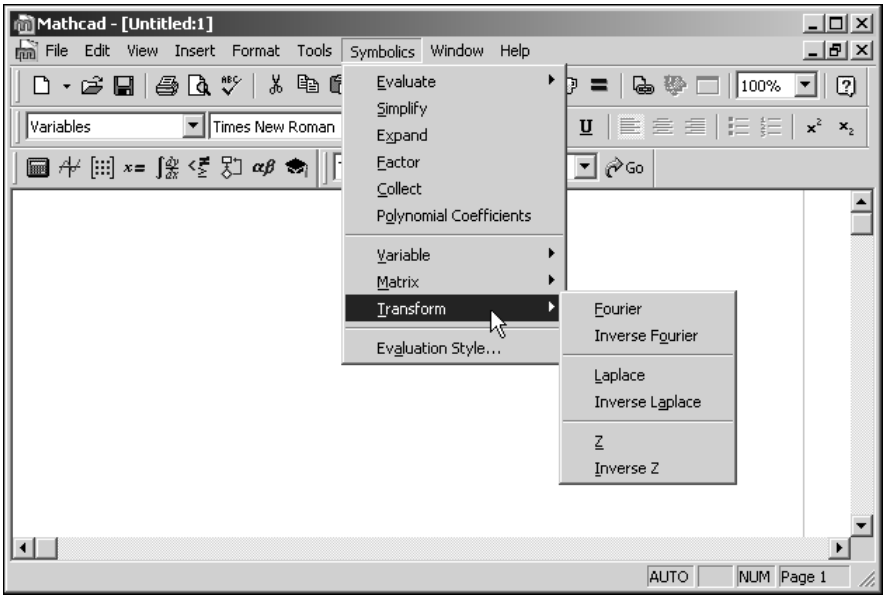


Рис. 1.15. Команды меню **Symbolics**

Таблица 1.7 *(продолжение)*

Команда	Описание
Factor	Команда во многом обратная команде **Expand**. В результате ее выполнения сумма (разность) дробей приводится к общему знаменателю, предпринимается попытка представить выражение в виде произведения и т. п.
Collect	Команда группировки степенных слагаемых в выражении
Polynomial Coefficients	Команда вычисления коэффициентов полинома. В результате выполнения команды возвращается вектор с коэффициентами полинома
Variable	Подменю содержит ряд команд. В частности, выбрав команду **Solve**, можно решить уравнение, получающееся в результате приравнивания нулю выражения, содержащего выделенную (на момент выполнения команды) переменную относительно этой переменной. Команда **Substitute** удобна, если нужно заменить выделенную переменную в выражении содержимым буфера обмена. Если выбрать команду **Differentiate**, можно вычислить производную по выделенной переменной от выражения, содержащего эту переменную. Чтобы вычислить интеграл по переменной, следует выбрать команду **Integrate**. Кроме этого выражение можно разложить в степенной ряд по переменной (команда **Expand to Series**) и раскладывать выражения на элементарные дроби (команда **Convert to Partial Fraction**)

Таблица 1.7 *(окончание)*

Команда	Описание
Matrix	Подменю содержит три команды, весьма полезные при работе с матрицами. Это **Transpose** — транспонирование матрицы, **Invert** — вычисление обратной матрицы (матрица, как известно, должна быть квадратной) и **Determinant** — вычисление определителя (детерминанта) матрицы
Transform	Подменю содержит команды выполнения интегральных преобразований (прямых и обратных). В частности можно выполнять прямое и обратное преобразование Фурье (соответственно команды **Fourier** и **Inverse Fourier**), прямое и обратное преобразование Лапласа (команды **Laplace** и **Inverse Laplace**), а также прямое и обратное Z-преобразование (команды **Z** и **Inverse Z**)
Evaluation Style	Команда для определения области вывода результата символьных операций. Можно выводить результат под преобразуемым выражением или справа от него. При этом имеется возможность выводить комментарий. Кроме того, можно задать режим, при котором результат замещает исходное выражение

Перечисленные команды более детально описаны в *главе 5*, посвященной методам символьных вычислений в Mathcad.

Меню *Window*

В меню **Window** (рис. 1.16) собраны команды управления окнами открытых на момент сеанса работы документов.

Собственно, команд в этом меню совсем немного (табл. 1.8).

Следует заметить, что разница между командами **Tile Horizontal** и **Tile Vertical** прослеживается, если открыто не больше трех окон. В противном случае практический результат их выполнения один и тот же. Окна рабочих документов выравниваются как по горизонтали, так и по вертикали. В целом же назначение команд этого меню должно быть хорошо знакомо всем, кто работал хотя бы с одним популярным офисным приложением.

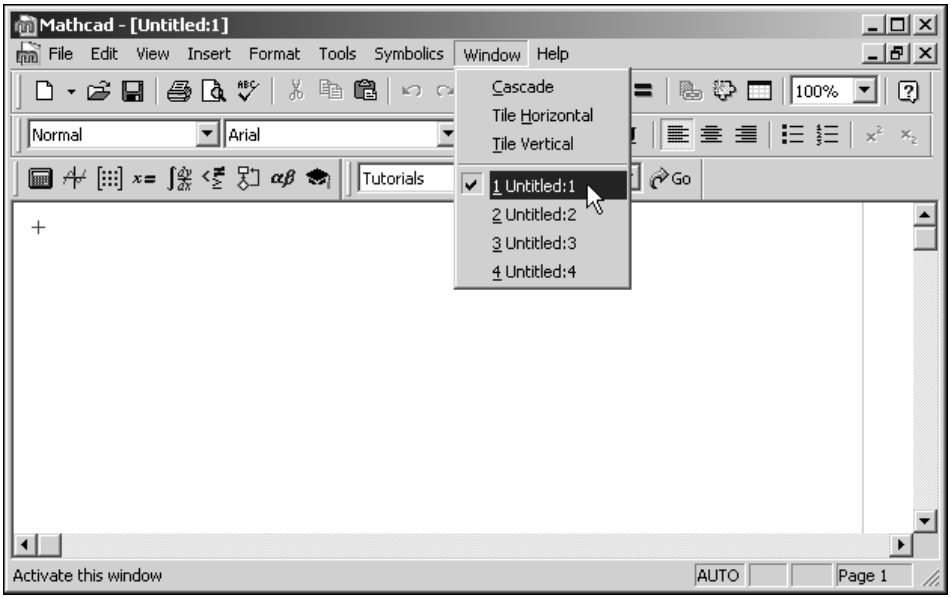


Рис. 1.16. Меню **Window**

Таблица 1.8. *Команды меню* ***Window***

Команда	Описание
Cascade	Окна располагаются каскадом одно за другим, но так, чтобы были видны их панели названий
Tile Horizontal	Окна размещаются одно под одним так, что полностью закрывается рабочая область приложения
Tile Vertical	Окна выстраиваются одно возле другого. При этом рабочая область приложения полностью закрывается окнами
Список окон	Список с названиями открытых на текущий момент рабочих документов. Если открыто более 10 рабочих документов, они отображаются не все. Но в этом случае в меню появляется команда **More Windows**, выбрав которую можно получить доступ к полному списку открытых документов. Чтобы сделать документ активным, возле него следует поставить флажок

Меню *Help*

Самое полезное меню, причем практически в любом приложении, — **Help** (рис. 1.17).

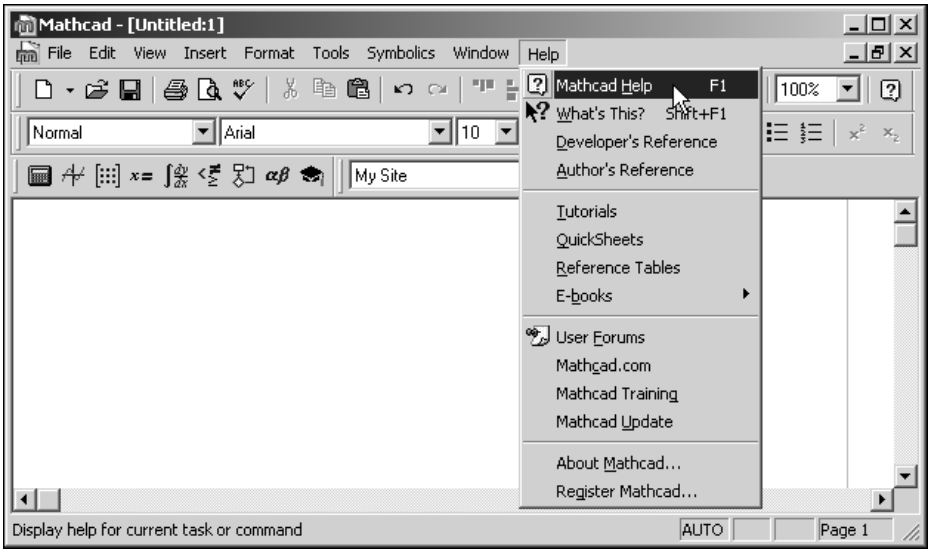


Рис. 1.17. Меню **Help**

Команды меню **Help** позволяют получать справку по самым различным аспектам работы с приложением Mathcad (табл. 1.9).

Таблица 1.9. *Команды меню* ***Help***

Команда	Описание
Mathcad Help	Вызов полномасштабной справки по Mathcad
What's This?	Вызов контекстной справки, позволяющей получить оперативную помощь по конкретному элементу интерфейса
Developer's Reference	Вызов справочного ресурса разработчика
Author's Reference	Вызов справочного ресурса с данными о способах создания электронных книг и конвертируемых Web-документов
Tutorials	Справка с обучающими утилитами, позволяющими быстро и легко усвоить методы работы с приложением
QuickSheets	База данных с описанием полезных шаблонов и освещением некоторых специфических методов работы с приложением
Reference Tables	Справочная база данных по ключевым вопросам, имеющим отношение к научным исследованиям

Таблица 1.9 *(окончание)*

Команда	Описание
E-books	Это подменю содержит команды для работы с электронными книгами и пакетами расширений
User Forums	Форум пользователей Mathcad
Mathcad.com	Переход на Web-сайт поддержки Mathcad
Mathcad Training	Переход на страницу обучающего центра Mathcad
Mathcad Update	Проверка возможности обновления приложения на сайте поддержки Mathcad **http://www.mathcad.com/services/training/**
About Mathcad	Регистрационные данные версии приложения
Register Mathcad	Регистрация программного продукта

На этом краткий обзор основных команд панели меню можно считать завершенным. Большинство из упомянутых команд еще будут описываться в книге по мере изложения материала, относящегося к конкретным функциональным возможностям приложения. Однако использование команд панели меню — далеко не единственный (хотя и достаточно эффективный) метод взаимодействия пользователя и приложения Mathcad.

Панели инструментов

С функциональной точки зрения не меньший интерес, чем панель меню, представляют панели инструментов Mathcad. При редактировании документов полезными будут стандартная панель инструментов и панель форматирования. Математическая панель инструментов содержит кнопки, позволяющие быстро и эффективно оперировать с математическими объектами в рабочем листе. Далее последовательно описываются эти три панели инструментов, которые, напомним, отображаются по умолчанию при запуске приложения.

Стандартная панель инструментов

На этой панели расположены кнопки, дублирующие в основном команды из меню **File**, **Edit** и частично **Format** и **Tools**. В табл. 1.10 можно найти описание кнопок (то есть тех команд, которые выполняются при щелчке на соответствующих кнопках).

Таблица 1.10. *Кнопки стандартной панели инструментов*

Кнопка	Описание
	При щелчке на кнопке создается новый рабочий документ. Если щелкнуть на стрелке справа от кнопки, откроется список с указанием базовых шаблонов, на основе которых можно создать новый документ. Для этого достаточно выбрать соответствующий элемент из списка. Кнопка во многом дублирует команду **File** ▏ **New**
	Кнопка для команды открытия документа. Эквивалент команды **File** ▏ **Open**
	При щелчке на кнопке выполняется сохранение рабочего документа. Эквивалент команды **File** ▏ **Save**
	Вывод документа на печать. Вывод на печать осуществляется также через меню командой **File** ▏ **Print**
	Кнопка для команды предварительного просмотра перед выводом документа на печать. Того же результата можно добиться, выбрав команду **File** ▏ **Print Preview**
	При щелчке на этой кнопке активизируется утилита проверки орфографии. То же имеет место при выборе команды **Tools** ▏ **Spelling**
	Удаление выделенного фрагмента в буфер обмена. Эквивалент команды **Edit** ▏ **Cut**
	Копирование выделенного фрагмента в буфер обмена. Эквивалент команды **Edit** ▏ **Copy**
	Вставка содержимого буфера обмена в место размещения курсора. Щелчок на этой кнопке эквивалентен выбору команды **Edit** ▏ **Paste**
	Щелчок на кнопке приводит к отмене последнего действия. Того же эффекта можно добиться выбором команды **Edit** ▏ **Undo**
	Щелчок на кнопке приводит к повторению отмененного действия. Эквивалент команды **Edit** ▏ **Redo**
	Выравнивание объектов рабочего документа в выделенной области в поперечном направлении. Эквивалент команды **Format** ▏ **Align Regions** ▏ **Across**
	Выравнивание объектов рабочего документа в выделенной области в продольном направлении. Эквивалент команды **Format** ▏ **Align Regions** ▏ **Down**
	При щелчке на кнопке открывается диалоговое окно вставки функции **Insert Function**, в котором можно выбрать функцию для вставки в рабочий документ. Щелчок на этой кнопке эквивалентен выбору команды **Insert** ▏ **Function**
	Щелчок на кнопке приводит к открытию диалогового окна **Insert Unit**, посредством которого осуществляется вставка единиц размерности. Эквивалентная команда меню — **Insert** ▏ **Unit**

Таблица 1.10 *(окончание)*

Кнопка	Описание
	Щелчок на кнопке равнозначен выбору команды **Tools	Calculate	Calculate Now**, в результате чего выполняется вычисление рабочего документа
	После щелчка на этой кнопке, как и после выбора команды **Insert	Hyperlink**, можно создать гиперссылку
	Кнопка дублирует команду вставки объекта **Insert	Component**
	Кнопка полезна при вставке таблицы. Щелчок на ней эквивалентен выбору команды **Insert	Data	Table**
100%	Поле с раскрывающимся списком, позволяющим задать масштаб отображения рабочего документа. Альтернативой может быть команда **View	Zoom**
	Щелчок на кнопке приводит к тому же эффекту, что и выбор команды **Help	Mathcad Help**, т. е. к вызову справки по приложению Mathcad

Важно понимать, что описанные в табл. 1.10 кнопки отображаются на стандартной панели инструментов по умолчанию. Однако в рамках приложения Mathcad пользователю предоставляется возможность самостоятельно настраивать существующие панели. В частности, туда можно добавлять или удалять кнопки. Поэтому если подобные настройки по отношению к приложению были пользователем выполнены, панель, соответственно, изменит свой вид. Это же замечание относится и к прочим панелям инструментов.

Панель форматирования

Панель форматирования в первозданном своем виде содержит кнопки, дублирующие те команды, которые имеют непосредственное отношение к форматированию текстовых полей. Кнопки этой панели инструментов представлены и описаны в табл. 1.11.

Таблица 1.11. *Кнопки панели форматирования*

Кнопка	Описание
Normal	Раскрывающийся список выбора стиля. Результат после щелчка на кнопке такой же, как и при выборе команды **Format	Style**
Arial	Поле выбора типа шрифта. Соответствующая настройка через панель меню может быть выполнена после выбора команды **Format	Text**

Таблица 1.11 *(окончание)*

Кнопка	Описание
10	Поле выбора размера шрифта. Размер можно также установить посредством команды **Format	Text**
B	Применение полужирного стиля. Устанавливается также в диалоговом окне, открывающемся после выбора команды **Format	Text**
I	Выделение курсивом. Может быть применено также после выбора команды **Format	Text**
U	Выделение подчеркиванием. Полезна может быть все та же команда **Format	Text**
	Выравнивание текста по левому краю. Можно также установить этот режим, если с помощью команды **Format	Paragraph**, в результате выбора которой открывается диалоговое окно **Paragraph Format**, задать настройки параметров стиля параграфа
	Выравнивание текста по центру. Альтернатива — воспользоваться командой **Format	Paragraph**
	Выравнивание текста по правому краю. Полезной может быть команда **Format	Paragraph**
	Переход в режим маркированного списка. Можно также воспользоваться командой **Format	Paragraph**
	Переход в режим нумерованного списка. Альтернатива — вызов команды **Format	Paragraph**
x^2	Переход в режим верхних индексов. Режим также может быть установлен в диалоговом окне **Text Format**, которое открывается после выбора команды **Format	Text**
x_2	Переход в режим нижних индексов. Режим устанавливается в том же диалоговом окне **Text Format** (команда **Format | Text**)

Поскольку в Mathcad имеется панель, предназначенная для работы с текстом, то вполне логично, что есть и панель, полезная при работе с математическими объектами, которая описывается в следующем подразделе.

Математическая панель

На математической панели представлено по умолчанию (и изменить эти настройки не удастся) девять кнопок. Данные обо всех этих кнопках сведены в табл. 1.12. Щелчок на любой из них приводит к тому, что отображается

одна из математических палитр, которые, в свою очередь, служат для быстрого ввода математических выражений и вставки всевозможных объектов в рабочий документ. Сами математические палитры описываются несколько позже.

Таблица 1.12. *Кнопки математической панели*

Кнопка	Описание
	Отображение палитры **Calculator**, позволяющей вводить щелчком мыши базовые функции, числа, константы и символы арифметических операций. Эквивалент команды **View	Toolbars	Calculator**
	Отображение палитры **Graph**, позволяющей выполнять вставку всевозможных графиков в рабочий документ. Эквивалент команды **View	Toolbars	Graph**
	Отображение палитры **Matrix**. Палитра используется для вставки в удобном режиме векторов и матриц в рабочий документ. Эквивалент команды **View	Toolbars	Matrix**
	Отображение палитры **Evaluation**, полезной при проведении преобразований, в том числе символьных. Палитра отображается после выполнения команды (установки флажка у соответствующего элемента) **View	Toolbars	Evaluation**
	Отображение палитры **Calculus**, которая непосредственно связана с выполнением таких операций, как вычисление производных, интегралов, определение пределов, вычисление сумм и произведений. Палитра отображается после выбора команды **View	Toolbars	Calculus**
	Отображение палитры **Boolean** для выполнения логических операций. Эквивалент вызова команды **View	Toolbars	Boolean**
	Отображение палитры **Programming**, которая удобна для составления программных кодов в среде приложения Mathcad. Эквивалент вызова команды **View	Toolbars	Programming**
	Отображение палитры ввода греческих символов **Greek**. Для отображения этой палитры можно также воспользоваться командой **View	Toolbars	Greek**
	Отображение палитры с командами символьных преобразований выражений **Symbolic**. Для отображения этой палитры можно также выбрать команду **View	Toolbars	Symbolic**

После щелчка на кнопке математической панели, эта кнопка остается в нажатом состоянии (отображается на панели специальным образом в виде вдавленной области). Если на кнопке щелкнуть еще раз, она вернется в свое исходное состояние. При этом закроется и соответствующая палитра. Помимо

описанных трех панелей инструментов, при работе с приложением могут использоваться и другие. О них — далее.

Панели ресурсов и элементов управления

Кроме рассмотренных основных панелей, в окне приложения могут отображаться вспомогательные, необходимые не так часто, но, вместе с тем, весьма полезные. В первую очередь это относится к панели ресурсов. Она обычно отображается при запуске приложения (хотя этот режим можно и отменить) и содержит всего два элемента (табл. 1.13).

Таблица 1.13. *Кнопки панели ресурсов*

Кнопка	Описание
Tutorials	Список выбора ресурса. Практически эти же элементы могут быть выбраны, если обратиться к содержимому меню **Help**
Go	Переход к тому ресурсу, который отображается в поле списка выбора ресурсов

Кроме этого, интерес представляет еще одна панель инструментов (рис. 1.18). Для ее отображения следует воспользоваться командой **View | Toolbars | Controls**.

Рис. 1.18. Панель элементов управления

Описание кнопок, размещенных на этой панели, можно найти в табл. 1.14.

Таблица 1.14. *Кнопки панели ресурсов*

Кнопка	Описание
	Вставка в рабочий документ индикатора. Это же можно сделать с помощью команды **Insert	Control	Check Box**
	Вставка в рабочий документ переключателя. В этом случае также можно вызвать команду **Insert	Control	Radio Button**
	Вставка в рабочий документ кнопки. Полезной также может быть команда **Insert	Control	Push Button**

Таблица 1.14 *(окончание)*

Кнопка	Описание
	Вставка в рабочий документ полосы прокрутки. Это же можно сделать с помощью команды **Insert	Control	Slider**
	Вставка в рабочий документ текстового поля. Используется также команда **Insert	Control	Text Box**
	Вставка в рабочий документ списка. Можно вызвать команду **Insert	Control	List Box**

Добавление элементов управления в рабочий документ может быть весьма эффективным. Особенно это справедливо в случае, когда в документе реализуются сложные логические схемы вычислений, существенно зависящие от выбора пользователя. Помимо этого, элементы управления упрощают процедуру взаимодействия пользователя с приложением. Подробнее об этом рассказывается в следующих главах книги и, в частности, в *главе 4*, посвященной методам программирования в Mathcad.

Настройка интерфейса пользователя

Даже если предлагаемый по умолчанию вид интерфейса Mathcad устраивает очень многих пользователей, наверняка найдутся те, кто пожелал бы внести в него некоторые, хотя бы незначительные, изменения. Часто для этого есть объективные причины. Ведь, как правило, одни команды необходимы чаще, чем другие. Причем у каждого пользователя предпочтения в области этих любимых команд свои. На этот случай в Mathcad предусмотрена возможность вносить изменения во внешний вид рабочего документа, отображаемого приложением при запуске. Начнем с наиболее простых методов, имеющих в основном "косметический" характер.

Настройка панелей инструментов

Основной метод настройки панелей инструментов состоит в добавлении и удалении расположенных на ней кнопок. Сразу следует отметить, что выполнять такие манипуляции можно далеко не со всеми панелями. Если быть абсолютно точным, то настраивают стандартную панель инструментов и панель форматирования. Проверить, настраивается панель или нет, достаточно легко. Для этого можно навести курсор мыши на соответствующую панель и щелкнуть на ней правой кнопкой. В результате откроется список, состоящий из двух элементов: **Hide** и **Customize** (рис. 1.19).

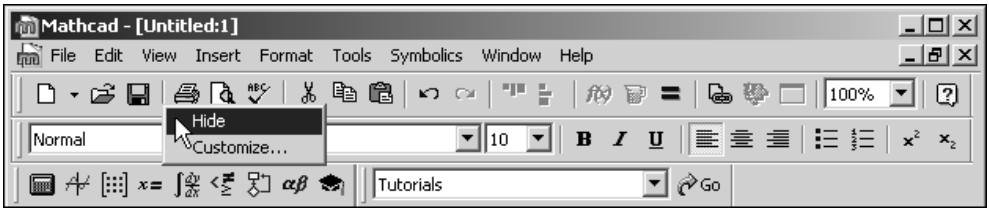


Рис. 1.19. Раскрывающееся меню с командами скрытия и редактирования панели инструментов

Если выбрать первый элемент списка (**Hide**), соответствующая панель будет скрыта. Второй элемент списка (**Customize**) выбирают как раз в том случае, если есть желание несколько видоизменить данную панель. Проблема в том, что этот элемент не всегда доступен. Если он выделен серым цветом, это означает, что панель настроить не удастся (если выбрать этот элемент для такой панели, ровным счетом ничего не произойдет).

Если же щелкнуть правой кнопкой мыши на панели форматирования или стандартной панели, а потом выбрать элемент **Customize**, то в результате откроется диалоговое окно **Настройка панели инструментов** (рис. 1.20).

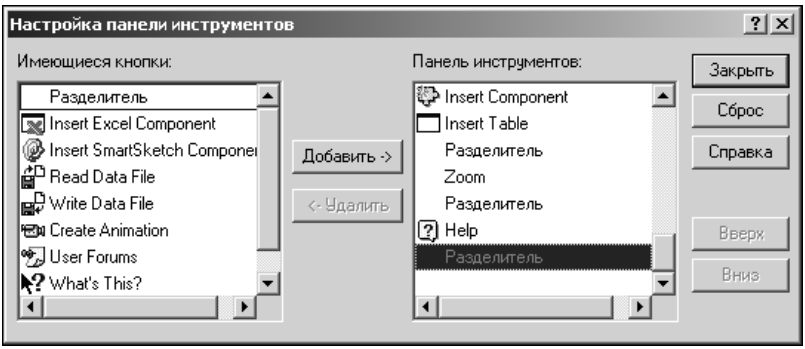


Рис. 1.20. Диалоговое окно **Настройка панели инструментов**

В этом диалоговом окне в левой части, в списке **Имеющиеся кнопки**, содержатся те кнопки (представлены как названием, так и соответствующей пиктограммой), которые можно разместить на панели. В правой части диалогового окна, в списке **Панель инструментов**, перечислены кнопки, размещенные на текущий момент на панели инструментов. В зависимости от того, для какой панели выполняется настройка, разными будут и упомянутые списки. На рис. 1.20 показано окно, которое открывается при настройке стандартной панели инструментов.

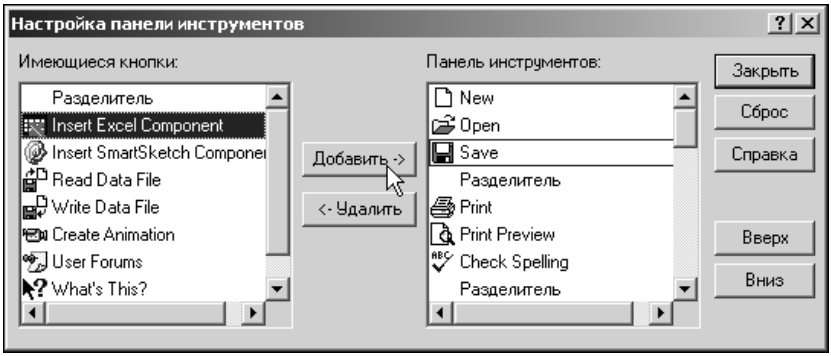


Рис. 1.21. Процесс добавления кнопки на панель

Для того чтобы добавить кнопку на панель инструментов, эту кнопку следует выбрать в левом списке диалогового окна **Настройка панели инструментов**. После этого нужно щелкнуть на кнопке **Добавить** (рис. 1.21).

Кнопка добавляется перед той кнопкой панели инструментов, которая была выделена в списке **Панель инструментов**, до начала процедуры добавления новой кнопки (то есть до того, как была выделена добавляемая кнопка). В данном случае кнопка вставки компонента Excel (**Insert Excel Component**) будет добавлена между кнопками сохранения и открытия документа (**Save** и **Open**). После щелчка на кнопке **Добавить** соответствующая кнопка перемещается из списка **Имеющиеся кнопки** в список **Панель инструментов** и сразу отображается и на панели инструментов.

После размещения кнопки можно менять ее позицию на панели инструментов. Для того чтобы переместить кнопку вниз по списку **Панель инструментов**, следует выделить эту кнопку (уже в списке **Панель инструментов**) и щелкнуть на кнопке **Вниз**. Щелчок на кнопке **Вверх** приводит к противоположному эффекту. На рис. 1.22 показано, как после щелчка на кнопке **Вверх**, кнопка **Insert Excel Component** оказывается между кнопками создания нового документа **New** и открытия документа **Open**.

Чтобы удалить кнопку с панели инструментов, следует кнопку выделить (в списке **Панель инструментов**, разумеется) и щелкнуть на кнопке **Удалить** (рис. 1.23).

Если в процессе перемещения и удаления кнопок на панели логическая цепочка внесения изменений утеряна, можно щелкнуть на кнопке **Сброс**, в результате чего все станет так, как было до начала внесения изменений. Следует также обратить внимание на наличие в обоих списках такого элемента, как **Разделитель**. Это не кнопка, а именно разделитель. Его удобно размещать на панели инструментов для обособления отдельных групп ячеек. Интересно и то, что перемещать кнопки из списка **Имеющиеся кнопки**

в список **Панель инструментов** и обратно можно обычным перетаскиванием пиктограммы соответствующей кнопки с помощью мыши. Для этого достаточно пиктограмму выделить и, удерживая нажатой левую кнопку мыши, перетащить кнопку в нужное место другого списка (рис. 1.24).

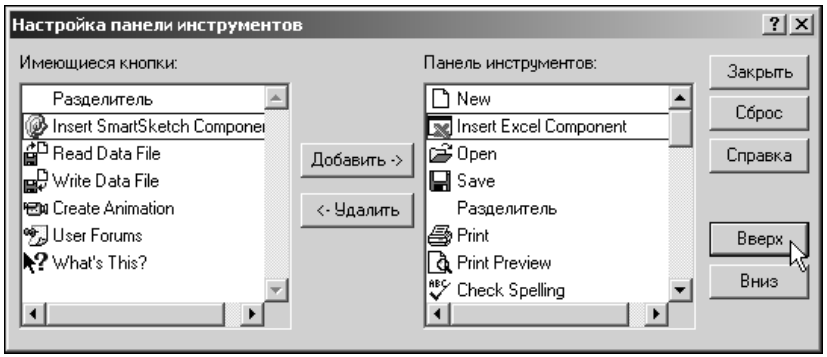


Рис. 1.22. Перемещение кнопки на панели

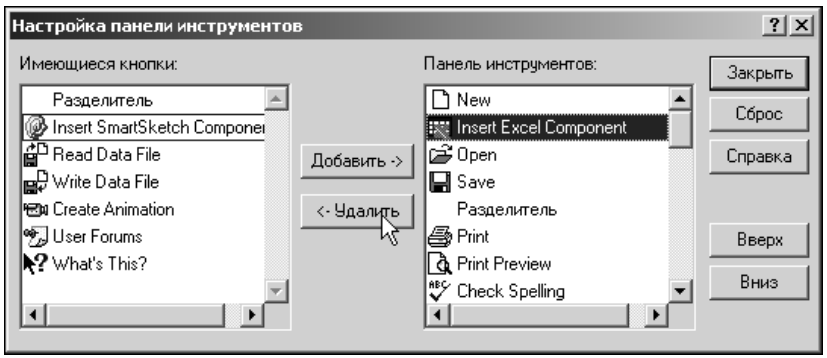


Рис. 1.23. Удаление кнопки с панели инструментов

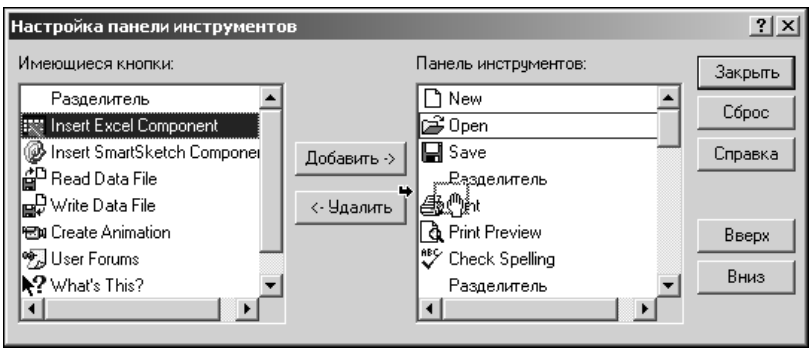


Рис. 1.24. Перемещение кнопки с помощью мыши

После внесения всех необходимых изменений щелкают на кнопке **Закрыть**.

На этом возможности по настройке панелей инструментов, по большому счету, заканчиваются. Сами панели можно перемещать по рабочему листу. Для этого на ней достаточно щелкнуть левой кнопкой мыши и, удерживая кнопку мыши нажатой, переместить панель в нужное место рабочего листа. Кроме того, как уже отмечалось, панели можно скрывать с помощью команды **Hide** из меню, раскрывающегося после щелчка правой кнопкой мыши на панели, или, отменив флажок у названия скрываемой панели в подменю **Tools** из меню **View**. Чтобы отобразить панель снова, флажок следует вернуть на место.

Линейка, масштаб и фон

Помимо панелей инструментов, существуют и другие способы выполнения настроек, которые непосредственно влияют на вид рабочего документа. Например, можно добавить такой элемент графического интерфейса, как линейка. Для этого следует выполнить команду **View | Ruler**. Рабочее окно с отображенной в нем сразу под панелями инструментов линейкой показано на рис. 1.25.

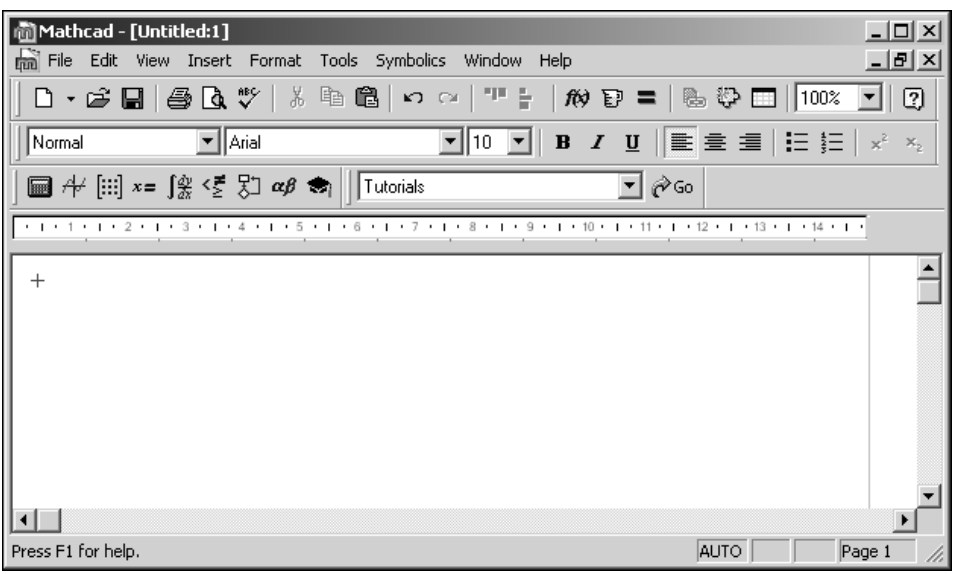


Рис. 1.25. Окно с линейкой

Хотя с помощью линейки и легче контролировать расположение объектов в рабочей области документа, она имеет в основном декоративный характер.

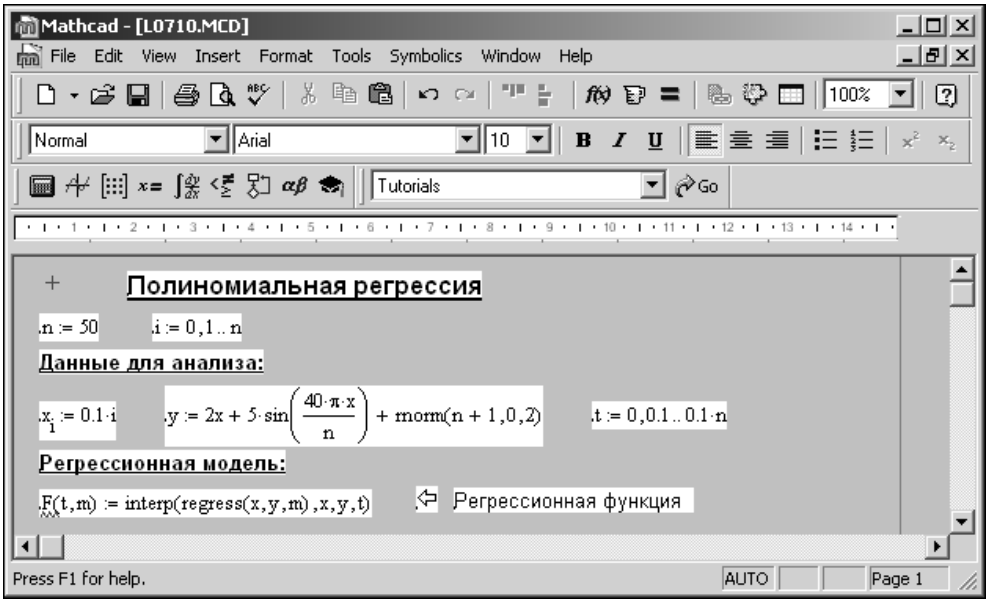


Рис. 1.26. Рабочий документ в режиме выделения выражений

Если выполнить команду **View | Regions**, то в результате рабочая часть документа будет закрашена серым цветом, за исключением тех областей, где в документе размещены объекты. Обычно это всевозможные математические выражения. На рис. 1.26 показано, как документ может выглядеть в этом случае.

Такой режим бывает весьма удобен, особенно в тех случаях, когда документ содержит большое число элементов, которые перекрываются на рабочем листе. Выделение их цветом позволяет, как минимум, контролировать ситуацию. Хотя справедливости ради следует отметить, что это не единственный способ борьбы с данной проблемой. Так, можно варьировать масштаб отображения элементов в рабочей области. Для изменения масштаба выполняют команду **View | Zoom**. После этого открывается диалоговое окно выбора масштаба **Zoom** (рис. 1.27).

В этом диалоговом окне можно либо выбрать один из предопределенных масштабов, либо указать собственный в поле **Custom**. Например, при выборе масштаба 200 % рабочее окно будет выглядеть так, как показано на рис. 1.28.

Для установки масштаба можно воспользоваться раскрывающимся списком на стандартной панели инструментов. Однако в этом случае доступны только те значения, что предлагаются в раскрывающемся списке. Это не всегда удобно, хотя соответствующий список и более полный, чем в диалоговом

окне **Zoom**. Есть, однако, ряд настроек, которые непосредственно на внешнем виде рабочего окна Mathcad не сказываются, зато существенно влияют на вид документа при выводе его на печать.

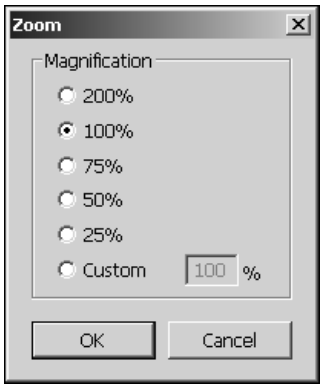


Рис. 1.27. Диалоговое окно **Zoom**

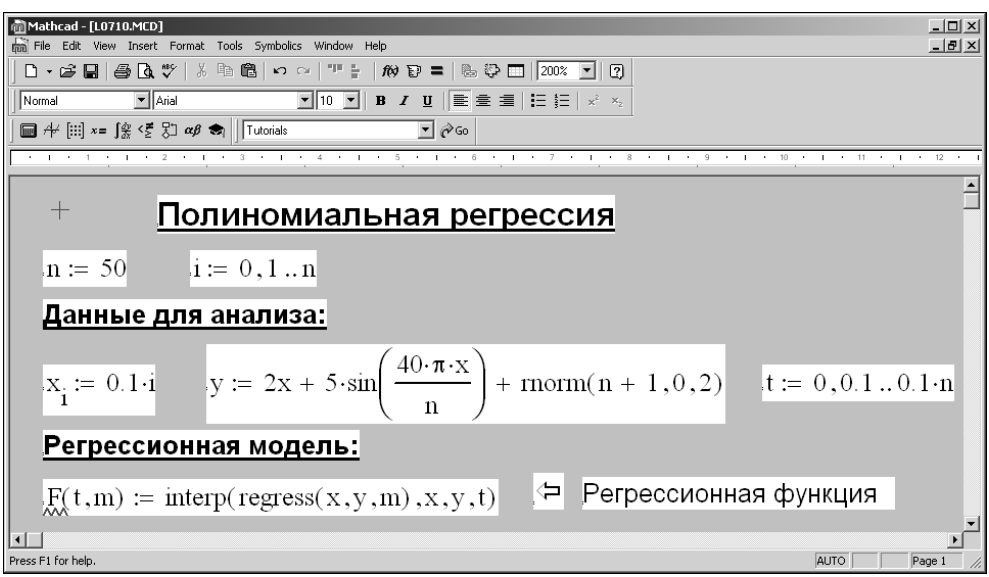


Рис. 1.28. Масштаб отображения 200 %

Настройка колонтитулов

В отличие от текстового редактора Word, в Mathcad, как уже отмечалось, рабочая область на листы не разбита. Другими словами, при вводе команд в рабочую область окна Mathcad иногда крайне проблематично определить,

как его содержимое будет выглядеть при выводе на печать. Самый простой способ проверить это — выбрать команду **File | Preview**. В результате можно узнать, что будет представлять собой документ в распечатанном виде. При этом есть возможность выполнить ряд настроек, которые сказываются при распечатке документа. Основные настройки, помимо тех, что относятся непосредственно к процессу печати документов, выполняются в диалоговом окне **Header and Footer** (рис. 1.29), которое открывается с помощью команды **View | Header and Footer**. В нем задается вид колонтитулов, принцип нумерации страниц, а также выполняется ряд смежных настроек.

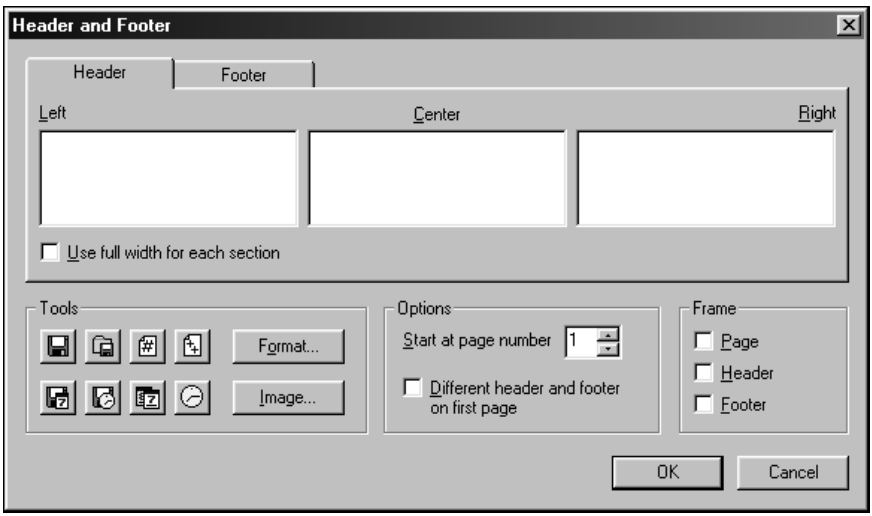


Рис. 1.29. Диалоговое окно **Header and Footer**

В исходном варианте окно содержит две вкладки: **Header** и **Footer**, где определяются соответственно верхние и нижние колонтитулы, принцип заполнения полей и выбора опций для которых идентичен. Далее для определенности речь будет идти о верхних колонтитулах. Часть настроек, имеющих отношение как к верхним, так и нижним колонтитулам, выполняется в нижней части диалогового окна, под вкладками.

В полях **Left**, **Center** и **Right** вводится текст, отображаемый в колонтитулах соответственно слева, по центру и справа. Флажок у опции **Use full width for each section** имеет смысл устанавливать только при наличии в колонтитулах длинных текстовых выражений. Если опция не выбрана, то такие значения размещаются в той части колонтитула, что отведена для них (то есть слева, справа или по центру). Если флажок у опции поставить, то текст будет выравниваться по ширине страницы. При этом возможны накладки частей

колонтитула друг на друга, так что пользоваться данным режимом следует крайне осторожно.

Кроме упомянутых вкладок диалоговое окно содержит три раздела. В одном из них, под названием **Tools**, представлены кнопки, которые позволяют вставлять в колонтитулы всякого рода полезную информацию, в том числе и номера страниц. Назначение кнопок описано в табл. 1.15.

Таблица 1.15. *Кнопки раздела* ***Tools*** *диалогового окна* ***Header and Footer***

Кнопка	Описание
	Вставка в колонтитул названия файла (инструкция `{f}`)
	Вставка в колонтитул полного пути к файлу (инструкция `{p}`)
	Вставка в колонтитул номера страницы (инструкция `{n}`)
	Вставка в колонтитул количества страниц в документе (инструкция `{nn}`)
	Вставка в колонтитул даты последнего сохранения файла (инструкция `{fd}`)
	Вставка в колонтитул времени последнего сохранения файла (инструкция `{ft}`)
	Вставка в колонтитул даты (инструкция `{d}`)
	Вставка в колонтитул времени (инструкция `{t}`)
Format...	Щелчок на кнопке приводит к раскрытию диалогового окна, в котором можно определить формат используемого в колонтитулах текста
Image...	В результате щелчка на кнопке открывается системное диалоговое окно выбора графического файла, который затем будет отображаться в колонтитулах

При вставке данных в колонтитулы с помощью перечисленных в табл. 1.15 кнопок в полях **Left**, **Center** и **Right** (в зависимости от того, куда данные вводятся), отображается специальный символ, заключенный в фигурные скобки (эти инструкции приведены в таблице). Если соответствующую инструкцию ввести с помощью клавиатуры, то эффект будет таким же. Удаляются введенные в поля колонтитулов инструкции обычным образом.

Помимо **Tools**, в диалоговом окне **Header and Footer** существуют еще два раздела: **Options** и **Frame**. В первом из них есть поле **Start at page number**, где указывается начальное значение для нумерации страниц, а также там представлена опция **Different header and footer on first page**. Если у нее уста-

новить флажок, то в диалоговом окне появится две дополнительных вкладки, в которых отдельно задаются колонтитулы для первой страницы (рис. 1.30). К этой услуге удобно прибегать в тех случаях, когда первая страница является, например, титульным листом.

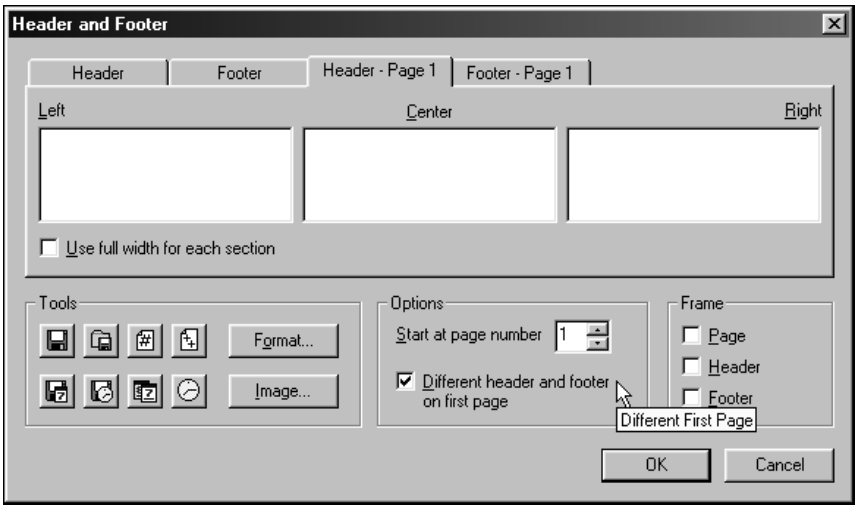


Рис. 1.30. Окно **Header and Footer** с дополнительными вкладками

Справа в диалоговом окне **Header and Footer** находится раздел **Frame** с тремя опциями: **Page**, **Header** и **Footer**. Если у этих опций установить флажки, то область верхнего колонтитула, рабочая область страницы и нижний колонтитул при выводе документа на печать будут заключены в рамки. Эти рамки можно видеть, если выполнить команду предварительного просмотра документа **File | Preview**.

Разбивка на страницы

Схожая с настройкой колонтитулов ситуация имеет место и при разбивке документа на страницы. Хотя при выводе документа на печать разбивка на страницы осуществляется автоматически, иногда бывает необходимо структурировать материал по своему усмотрению. Другими словами, желательно уметь размещать выбранный фрагмент данных на отдельной странице. В таких случаях в рабочий документ вставляют инструкцию перехода на новую страницу, для чего вызывают команду **Insert | Page Break**. В результате в рабочем документе, там, где был маркер (крестик в виде знака плюс) появится горизонтальная полоса. Все, что расположено ниже нее, будет при распечатке документа находиться на следующей странице.

Использование групп

В рабочих документах Mathcad данные можно размещать в специальных структурных блоках или группах. Преимущество такого подхода состоит в том, что группы можно сворачивать и разворачивать, скрывая и отображая тем самым фрагменты данных в рабочем документе. Когда документ не очень большой, это не нужно. Однако если в документе много команд, рисунков и прочих элементов, использование групп может оказаться весьма продуктивным, особенно если документ содержит много промежуточных вычислений. Вспомогательные или второстепенные данные помещаются в группы, и, если нужно проиллюстрировать только основные моменты решения, группы сворачиваются. В случае необходимости можно показать и всю "подноготную" — достаточно развернуть группы в рабочем документе.

Для вставки группы в рабочий документ следует вызвать команду **Insert | Area**. После этого в документе под маркером появляется область, выделенная двумя горизонтальными линиями с маленькими стрелками слева (рис. 1.31).

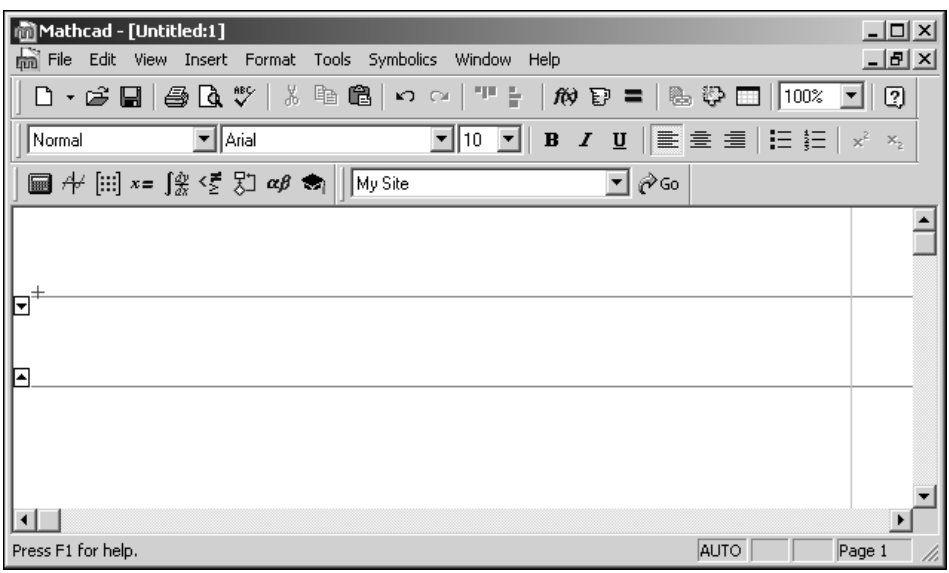


Рис. 1.31. Рабочий документ, содержащий группу

Границы группы можно перемещать. Для этого нужно щелкнуть на одной из границ группы левой кнопкой мыши. В результате соответствующая граница будет выделена. Если после этого навести на данную границу курсор мыши, он примет вид развернутой ладони. Удерживая нажатой левую кнопку мыши, можно переместить границу в требуемое место рабочего листа (рис. 1.32).

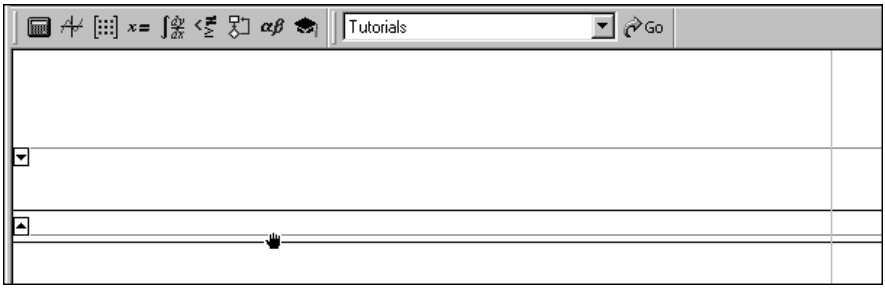


Рис. 1.32. Перемещение границ группы

Чтобы переместить группу в рабочем документе, ее нужно выделить (удерживая нажатой левую кнопку мыши с захватом обеих границ группы) и после этого перемещать по документу. Если на границе группы выполнить двойной щелчок мышью, она будет свернута. В результате обе границы группы совмещаются, а все, что было между ними, скрывается. В документе отображается одна горизонтальная линия со стрелкой в левой части. На рис. 1.33 показана развернутая группа с данными, а на рис. 1.34 — эта же группа, но в свернутом виде.

Двойной щелчок на свернутой группе (той горизонтальной линии, что от нее осталась) приводит к ее разворачиванию.

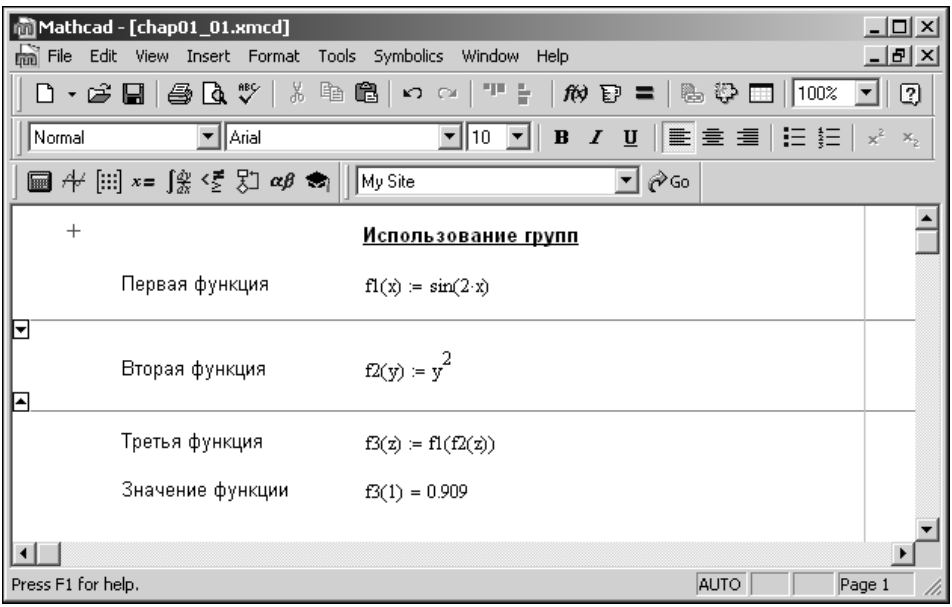


Рис. 1.33. Группа с данными развернута

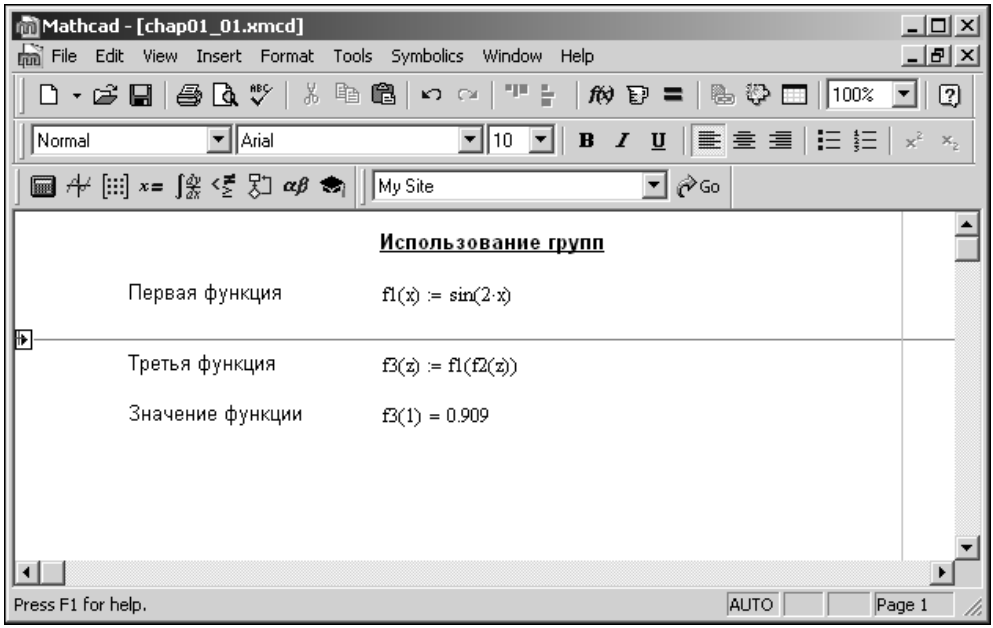


Рис. 1.34. Группа с данными свернута

Основные настройки

Хотя графический интерфейс является достаточно важной характеристикой приложения, не только он определяет его функциональность. Существует много параметров, которые существенным образом влияют на работу приложения. Некоторые из них могут настраиваться пользователем.

Параметры документа

Часть настроек выполняется в диалоговом окне **Worksheet Options**, которое открывается после выполнения команды **Tools | Worksheet Options**. Окно содержит несколько вкладок. На рис. 1.35 показано данное окно, открытое на вкладке **Built-In Variables**.

На вкладке размещено шесть полей и одна кнопка. Это не считая трех стандартных кнопок в нижней части диалогового окна: **OK** для подтверждения изменений, **Cancel** (Отмена) для отмены изменений и **Help** (Справка) для получения справки.

Кнопка **Restore Defaults** на вкладке **Built-In Variables** используется в том случае, если нужно восстановить настройки системы по умолчанию. Что это

за настройки, можно определить по числам, которые указаны справа от соответствующих полей в скобках (см. рис. 1.35). Назначение полей вкладки **Built-In Variables** описано в табл. 1.16.

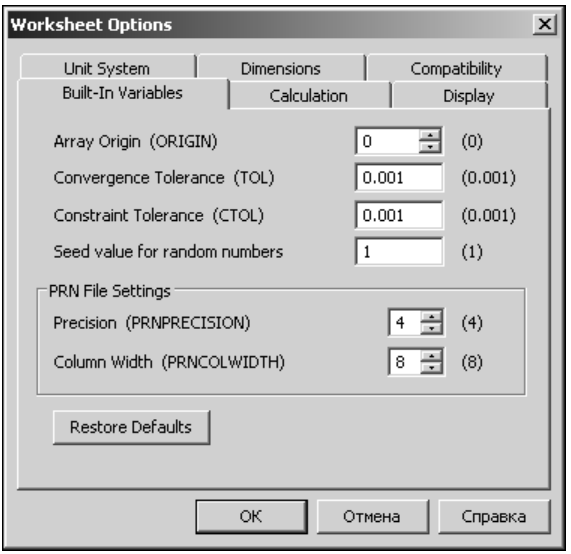


Рис. 1.35. Диалоговое окно **Worksheet Options** открыто на вкладке **Built-In Variables**

Таблица 1.16. *Назначение полей вкладки* ***Built-In Variables***

Поле	Описание
Array Origin (ORIGIN)	Значение переменной `ORIGIN`, которая определяет начальный индекс элементов массивов, используемых в рабочих документах. Значение по умолчанию — 0
Convergence Tolerance (TOL)	Значение переменной `TOL`, которая задает точность вычисления интегралов и производных. Кроме этого переменная влияет на точность получения решений уравнений. Значение по умолчанию — 0.001
Constraint Tolerance (CTOL)	Значение переменной `CTOL`, определяющей точность выполнения дополнительных условий в задачах поиска решений и оптимизации. Значение по умолчанию — 0.001
Seed value for random numbers	Параметр инициализации для генератора случайных чисел. Значение по умолчанию — 1

Таблица 1.16 *(окончание)*

Поле	Описание
Precision (PRNPRECISION)	Значение переменной `PRNPRECISION`, в которой задается число значащих цифр при записи файлов данных в формате ASCII. Значение по умолчанию — 4
Column Width (PRNCOLWIDTH)	Значение переменной `PRNCOLWIDTH`, определяющей ширину столбцов в создаваемом файле данных ASCII. Значение по умолчанию — 8

На вкладке **Calculation** задаются параметры, относящиеся к процессу вычислений в рабочем документе. В частности, здесь имеется четыре опции (рис. 1.36).

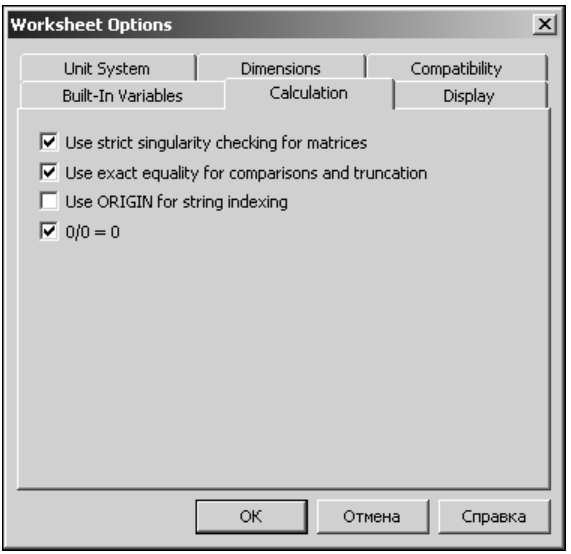


Рис. 1.36. Диалоговое окно **Worksheet Options** открыто на вкладке **Calculation**

Наличие флажка у опции **Use strict singularity checking for matrices** позволяет выполнять проверку матриц на сингулярность. Эта опция важна при вычислении обратных матриц, когда детерминант обращаемой матрицы должен быть отличен от нуля. Если опция установлена, проверка осуществляется более строго, что может существенно сказаться на результатах работы.

Опция **Use exact equality for comparisons and truncation** позволяет контролировать точность совпадения чисел при сравнении и округлении. Так, если

опция установлена, то два числа считаются одинаковыми, когда разница между ними не превышает точности математического сопроцессора. При отключенной опции для того, чтобы числа интерпретировались как одинаковые, отношение разности этих чисел к их среднему значению не должно превышать 10^{-12}.

Если установлен флажок опции **Use ORIGIN for string indexing**, то при работе с текстовыми функциями значение переменной `ORIGIN` будет использоваться в качестве индекса, ассоциирующегося с первым символом текстовой строки.

Опция **0/0 = 0** предназначена для контроля и обработки ситуаций, когда имеет место деление нуля на нуль. Если флажок опции установлен, то в результате выполнения такой операции будет возвращаться нулевое значение. В противном случае — сообщение об ошибке.

На вкладке **Display** представлено семь раскрывающихся списков (рис. 1.37), с помощью которых определяется способ отображения основных математических операторов в рабочем документе.

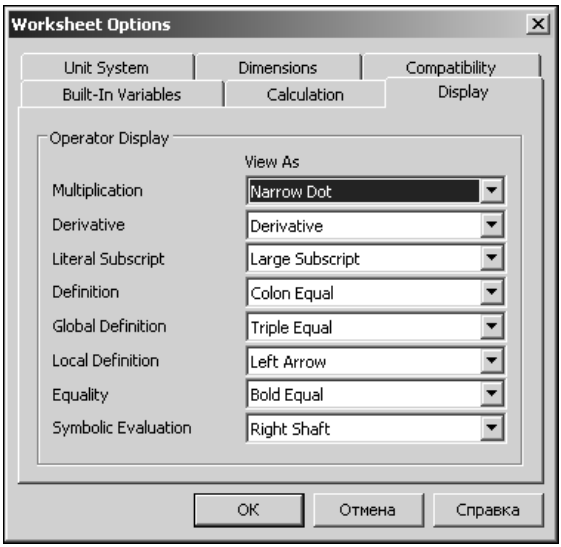


Рис. 1.37. Диалоговое окно **Worksheet Options** открыто на вкладке **Display**

Некоторые сведения по этому поводу приведены в табл. 1.17. Те, кого эта проблема интересует более серьезно, могут заняться детальным изучением вопроса самостоятельно, лучше всего методом "прямого эксперимента" или прибегнув к помощи справочной системы Mathcad.

Таблица 1.17. *Настройки вкладки* ***Display***

Раскрывающийся список	Описание
Multiplication	Определение вида оператора умножения. Возможные значения: **AutoSelect** — вид оператора умножения автоматически выбирается в контексте выражения, **Dot** — оператор умножения имеет вид точки, **Narrow Dot** — оператор умножения в виде небольшой точки (значение по умолчанию), **Large Dot** — оператор умножения в виде крупной точки, **X** — оператор умножения в виде косого креста, **Thin Space** — вместо оператора умножения оставляется узкий пробел между операндами, **No Space** — оператор умножения не отображается и пробел между операндами не оставляется
Derivative	Определение способа отображения производных. Возможные значения: **Derivative** — обычная производная (значение по умолчанию) и **Partial Derivative** — частная производная
Literal Subscript	Определение вида нижних индексных надписей. Возможные значения: **Large Subscript** — для отображения крупным шрифтом и **Small Subscript** — для отображения мелким шрифтом (значение по умолчанию)
Definition	Определение вида оператора декларирования переменной. По сути своей — это оператор присваивания. Возможные значения: **Colon Equal** (значение по умолчанию) — оператор присваивания имеет вид знака равенства с двоеточием (**:**=) и **Equal** — оператор присваивания совпадает со знаком равенства (=)
Global Definition	Определение оператора глобального декларирования переменной. Возможные значения: **Triple Equal** (значение по умолчанию) — в этом случае оператор глобального присваивания формально совпадает со знаком тождественного равенства (≡) и **Equal** — оператор задается знаком равенства (=)
Local Definition	Определение оператора локального декларирования переменной. Речь идет об операторе присваивания, используемом в программах. Возможны два значения: **Left Arrow** (значение по умолчанию) — оператор имеет вид стрелки влево, и **Equal** — оператор имеет вид знака равенства
Equality	Определение вида оператора равенства. Возможны два значения: **Bold Equal** (значение по умолчанию) — оператор имеет вид знака равенства, выделенного полужирным стилем, и **Equal** — оператор имеет вид обычного знака равенства
Symbolic Evaluation	Определение вида оператора символьного равенства. Возможные значения: **Right Shaft** (стрелка вправо, установлено по умолчанию) и **Equal** (знак равенства)

Принципы работы с операторами и их назначение более детально описываются в последующих главах при решении задач. Здесь же хочется обратить внимание на два обстоятельства. Во-первых, на вкладке **Display** задается способ отображения операторов в рабочем документе. Если, например, установить в списке **Definition** значение **Equal**, то операции присваивания в рабочем документе будут отображаться с помощью знака равенства. Однако при выделении выражения с оператором присваивания соответствующий оператор отображается в виде равенства с двоеточием.

Во-вторых, на вкладке **Display** задаются настройки для всего документа. Если необходимо изменить формат отображения оператора в отдельном выражении, следует щелкнуть на нем правой кнопкой мыши и из раскрывающегося списка выбрать нужную команду. В частности, на рис. 1.38 проиллюстрирован процесс изменения вида оператора присваивания.

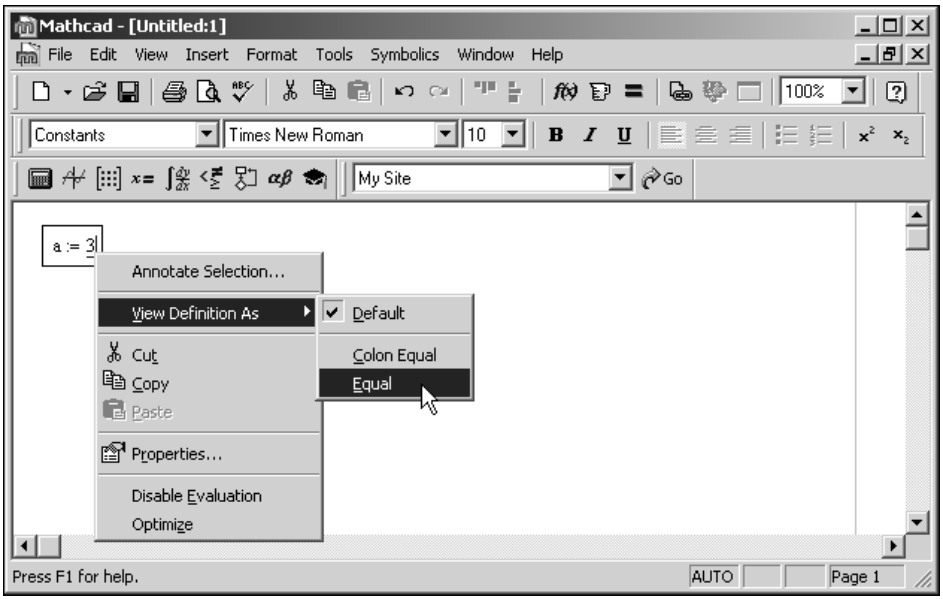


Рис. 1.38. Изменение вида оператора присваивания

В общем же случае, в подменю **View** ***оператор*** **As** следует выбрать одно из возможных значений для вида оператора (значение по умолчанию, помимо отображения его среди прочих возможных значений, еще и явно вынесено в отдельный пункт).

На вкладке **Unit System** выбирается система единиц. Настройка осуществляется с помощью переключателей группы **Default Units** (рис. 1.39).

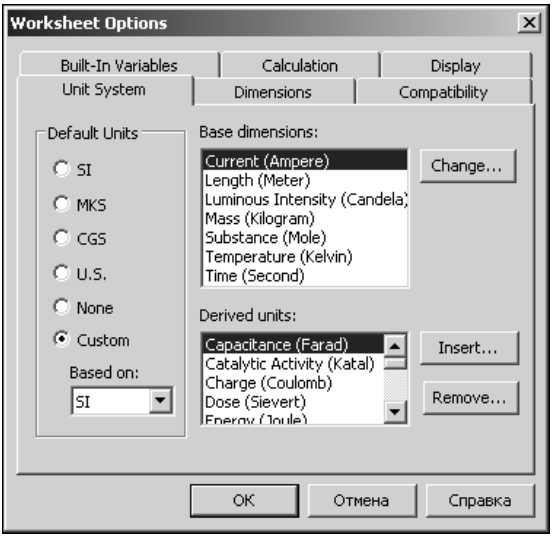


Рис. 1.39. Диалоговое окно **Worksheet Options** открыто на вкладке **Unit System**

В левой части вкладки размещена группа переключателей **Default Units**. С помощью этих переключателей можно выбрать одну из предопределенных систем единиц или определить собственную. Назначение переключателей описано в табл. 1.18.

Таблица 1.18. *Переключатели вкладки* ***Unit System***

Переключатель	Описание
SI (International)	Международная система единиц СИ. В этой системе расстояние измеряется в метрах, время — в секундах, масса — в килограммах и т. д.
MKS	Система МКС (аббревиатура от метр-килограмм-секунда). Как несложно догадаться, в этой системе расстояние, время и масса определяются в тех же единицах, что и в системе СИ, сама система является системой механических единиц и входит в состав системы СИ
CGS	Система СГС (аббревиатура от сантиметр-грамм-секунда). Единица длины — сантиметр, массы — грамм, времени — секунда
US	Американская система единиц. Единица длины — фут, массы — фунт, времени — секунда
None	Не используется ни одна из систем единиц
Custom	Определение системы единиц пользователем. Если переключатель установлен в это положение, доступным становится раскрывающийся список **Based on** в нижней части вкладки. С помощью этого списка выбирается базовая система единиц, на основе которой пользователь определяет новую

В текстовых полях **Base dimensions** и **Derived units** для выбранной системы отображаются основные единицы и производные от них. Возможность выбора размерных единиц существенно расширяет функциональность приложения.

На вкладке **Dimensions** задают способ отображения размерностей при выполнении операций с размерными величинами (рис. 1.40).

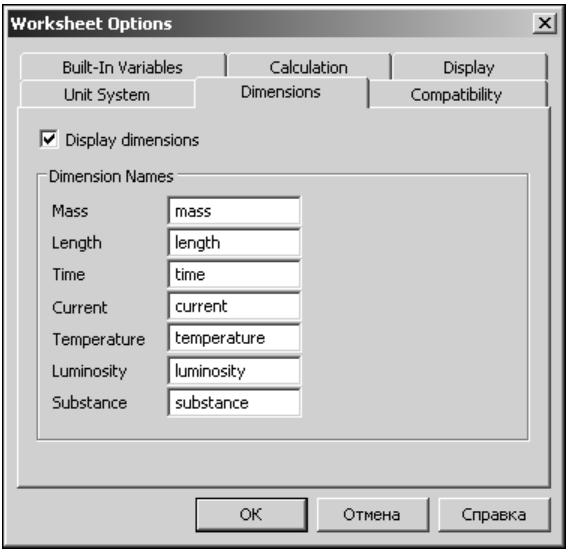


Рис. 1.40. Диалоговое окно **Worksheet Options** открыто на вкладке **Dimensions**

Если флажок у опции **Display dimension** не установлен, то для размерного результата отображается та размерная единица, что соответствует выбранной системе единиц. Если флажок у опции **Display dimension** имеется, то активизируется вкладка **Dimension Names**, содержащая семь полей (см. рис. 1.40), в каждом из которых можно задать название, отображаемое в качестве размерности величины, полученной в результате манипуляций с исходными данными. Назначение полей вкладки **Dimensions** описано в табл. 1.19.

Таблица 1.19. *Поля вкладки* ***Dimensions***

Поле	Размерность
Mass	Масса
Length	Длина
Time	Время
Current	Ток (электрический)

Таблица 1.19 *(окончание)*

Поле	Размерность
Temperature	Температура
Luminosity	Яркость
Substance	Вещество (имеются в виду величины, характеризующие количество вещества, в молях, например)

В данных настройках необходимость возникает редко, но все же иногда прибегать к ним бывает очень даже полезно.

На вкладке **Compatibility** имеются две группы переключателей, каждый на два положения (рис. 1.41).

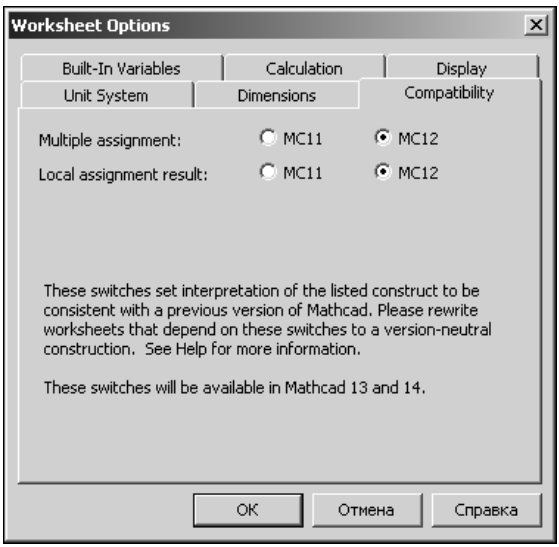


Рис. 1.41. Вкладка **Compatibility** диалогового окна **Worksheet Options**

Переключатели могут находиться в положении **MC11** и **MC12**, что означает режим совместимости с версиями программы Mathcad 11 и Mathcad 12, соответственно. Совместимость можно устанавливать для процедуры множественного и локального присваивания (группы **Multiple assignment** и **Local assignment result**). Дело в том, что при определении элементов массива в версии Mathcad 11 значения элементам присваиваются по очереди, так что уже заданные значения можно использовать для определения новых элементов. В Mathcad 12 и Mathcad 13 значения элементам присваиваются одновременно. При локальном присваивании в Mathcad 11 и Mathcad 13 возвращается

значение справа от операнда, а в Mathcad 12 — слева. Эти особенности совместимости разных версий приложений следует иметь в виду.

Настройки приложения

Перечисленные настройки применяются по отношению к рабочему документу. Помимо этого, существует возможность вносить такие изменения в настройки, что они автоматически применяются ко всему приложению. Делается это в окне **Preferences**, которое открывается после выбора команды **Tools | Preferences** (рис. 1.42).

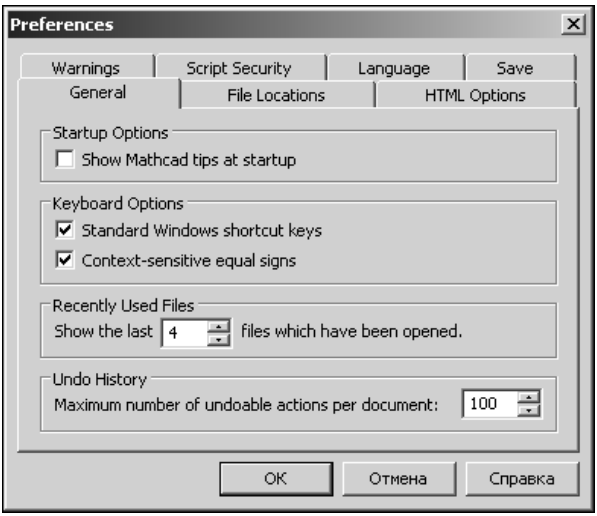


Рис. 1.42. Диалоговое окно **Preferences** открыто на вкладке **General**

Диалоговое окно, как несложно заметить, содержит семь вкладок. На вкладке **General** выполняются общие настройки приложения. В частности, там представлены три опции и два поля. Опция **Show Mathcad tips at startup** (единственная в разделе **Startup Options**) выбирается в том случае, если при запуске приложения есть желание наблюдать диалоговое окно **Tip of the Day**, приветствующее пользователя порцией полезных советов. Понятно, что если от советов голова и так идет кругом, флажок у опции ставить не нужно. В разделе **Keyboard Options** две опции: **Standard Windows shortcut keys** и **Context-sensitive equal signs**. Если выставлен флажок у первой из них, то для команд Mathcad в качестве горячих используются общепринятые в Windows комбинации клавиш. Если флажка у опции нет, устанавливаются те комбинации, что свойственны именно для Mathcad. В любом случае при вызове команд через панель меню комбинации клавиш, действующие на данный момент

в соответствии с выполненной настройкой, отображаются справа от команд. С одной стороны, выбор стандартных для Windows комбинаций клавиш кажется вполне разумным. Это удобно, и, кроме того, стандартный набор Windows несколько шире предлагаемого в Mathcad. Однако не исключено, что многие пользователи, кто уже давно работает с Mathcad, привыкли к традиционным настройкам Mathcad. В этом смысле наличие опции **Standard Windows shortcut keys** вполне оправдывает себя.

Описание опции **Context-sensitive equal signs** требует некоторых пояснений. В Mathcad для того, чтобы узнать значение переменной, после ее названия в рабочем документе указывается знак равенства. Чтобы переменной значение присвоить, имеется оператор присваивания (это знак равенства с двоеточием и вводится он нажатием комбинации клавиш <Shift>+<:>). Если переменной значение не присвоено, то попытка узнать ее значение (указав знак равенства после ее названия) приведет к ошибке. Однако существует возможность перейти в режим, при котором такая ситуация интерпретируется не как попытка определить значение переменной, а как выполнение присваивания. Другими словами, если переменной значение не присвоено, то знак равенства после имени переменной интерпретируется как оператор присваивания. Данный режим установлен по умолчанию, о чем свидетельствует флажок у опции **Context-sensitive equal signs**. Чтобы выйти из этого режима, флажок у опции следует убрать.

Во вкладке **Recently Used Files** устанавливается число файлов, которые указываются списком в меню **File**. Это те файлы, что открывались последними. Соответствующее число находится в поле **Show the last ... file which have been opened**. По умолчанию отображается список из четырех файлов. В разделе **Undo History** выполняется достаточно важная настройка, определяющая количество действий, которые можно отменять командой **Undo**. Это может быть число в диапазоне от 20 до 200, которое указывается в поле **Maximum number of undoable actions per document**. Значение по умолчанию — 100.

На вкладке **HTML Options** контролируются настройки, определяющие характер сохранения документа в формате Web-страницы. Открытое на вкладке **HTML Options** диалоговое окно **Preferences** показано на рис. 1.43.

Формат изображений задается переключателем группы **Save images as**. Там, в частности, можно установить формат PNG (переключатель в положении **PNG**) или JPEG (переключатель в положении **JPEG**). В последнем случае в поле **Quality** указывается степень сжатия.

Сохранять документ в виде Web-страницы можно с фиксированной или относительной разметкой. Для фиксированной разметки в группе **Save layout**

as переключатель устанавливают в положение **Fixed**, положение **Relative** соответствует относительной разметке. В последнем случае в поле **Web page template** указывается шаблон для сохранения документа в виде Web-страницы. При этом полезной может быть кнопка **Browse**, с помощью которой файл шаблона намного легче искать по системе.

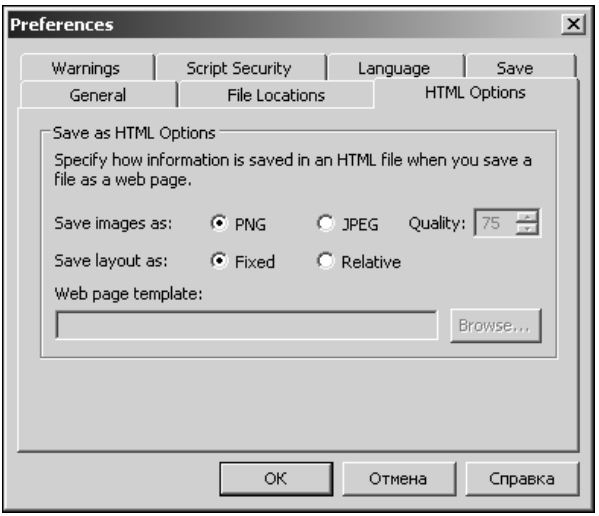


Рис. 1.43. Диалоговое окно **Preferences** открыто на вкладке **HTML Options**

Часто в рабочих документах необходимы элементы управления, которые, в свою очередь, используют какие-то сценарии. Такие элементы могут представлять потенциальную угрозу, например, содержать вирусы. На вкладке **Script Security** можно определить уровень защиты для подобных случаев. В частности, в единственном разделе **Scriptable Components** можно, установив переключатель в одно из трех положений (**High Security**, **Medium Security** и **Low Security**), определить высокий, средний и низкий уровень защиты, соответственно (рис. 1.44).

Если переключатель установлен в положение **High Security**, все элементы, содержащие сценарии, будут автоматически деактивизированы. Чтобы сделать их активными, придется выделять каждый такой элемент и активизировать их, что называется, вручную. Для этого элемент выделяется, после чего на нем следует щелкнуть правой кнопкой мыши и из раскрывающегося списка выбрать команду **Enable Evaluation**. При среднем уровне безопасности (переключатель в положении **Medium Security**) при открытии документа пользователю предоставляется возможность сделать выбор: активизировать или нет элементы со сценариями. Низкий уровень безопасности (положение **Low Security**) подразумевает отсутствие всякой защиты.

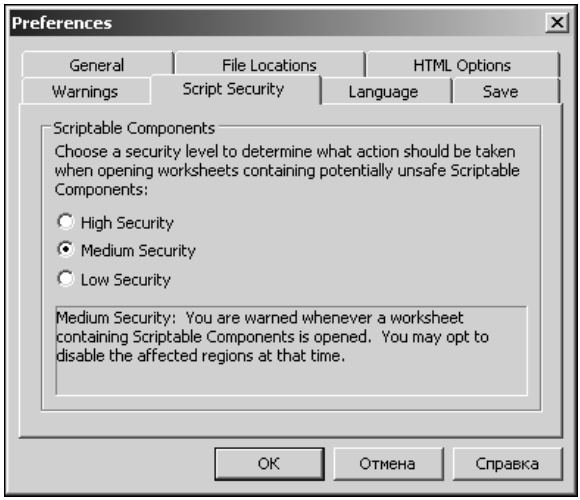


Рис. 1.44. Диалоговое окно **Preferences** открыто на вкладке **Script Security**

На вкладке **Language** в разделе **Language Options** выбирается язык для меню и диалогов (список **Menus and dialogs**) и язык математических выражений (список **Math language**) (рис. 1.45).

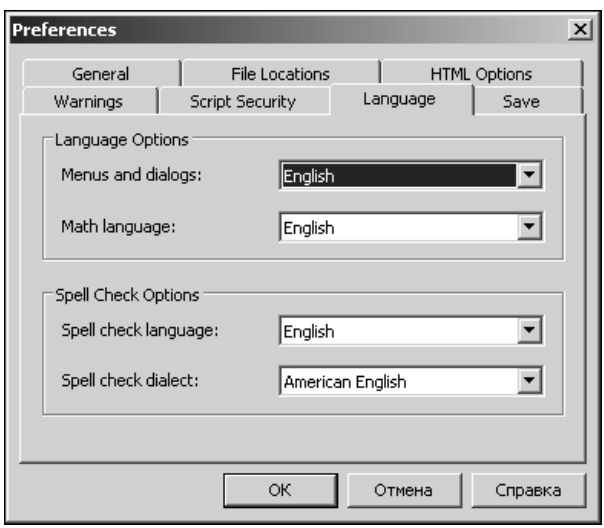


Рис. 1.45. Диалоговое окно **Preferences** открыто на вкладке **Language**

В разделе **Spell Check Options** задаются настройки языка для проверки орфографии: в списке **Spell check language** указывают язык, а в списке **Spell check dialect** можно выбрать диалект языка при работе с документами.

На вкладке **File Locations** устанавливают используемые по умолчанию папки для работы с приложением (рис. 1.46).

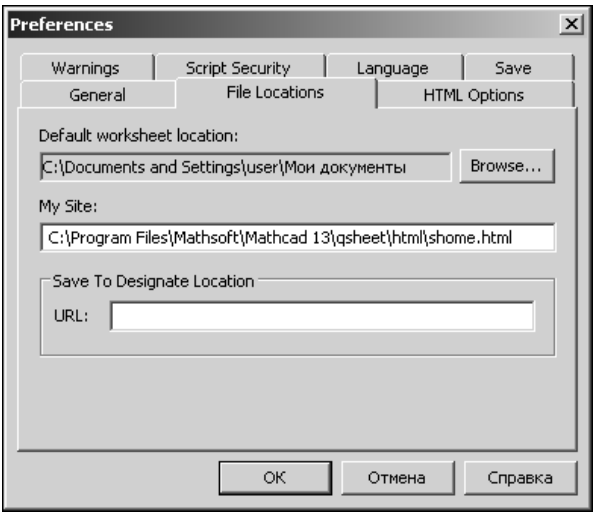


Рис. 1.46. Диалоговое окно **Preferences** открыто на вкладке **File Locations**

В поле **Default worksheet location** задается рабочая папка, которая будет предлагаться пользователю при сохранении документов и открытии уже существующих. Для того чтобы определить эту папку, следует щелкнуть на кнопке **Browse**. В результате открывается диалоговое окно **Browse for Folder**, с помощью которого и выбирается нужная папка. Понятно, что в качестве рабочей папки удобно выбирать ту, что наиболее часто необходима в работе. Папка с персональной страницей указывается в поле **My Site**.

Пользователь может управлять системой сообщений и предупреждений, выводимых системой при работе. Опции управления этим процессом собраны на вкладке **Warnings** (рис. 1.47).

Более точно, эти опции позволяют определять ситуации, в которых выводится сообщение о переименовании объектов рабочего листа Mathcad. Чтобы такие сообщения появлялись, следует установить флажок опции **Show warnings on redefinition of**. Если опция активизирована, доступными будут и прочие опции вкладки, которые разбиты на два раздела. В одном разделе речь идет о встроенных объектах (**Built In**), во втором — об определенных пользователем (**User Defined**). Здесь можно переопределить следующие объекты:

- функции (опции **Functions**);
- единицы размерности (опция **Units**);
- константы (опция **Constants**);

- переменные (опция **Variables**);
- скалярные величины (опция **Scalar Variables**);
- векторы и матрицы (опция **Vectors and Matrices**).

Наконец, на вкладке **Save** задают формат сохранения рабочих документов Mathcad (рис. 1.48).

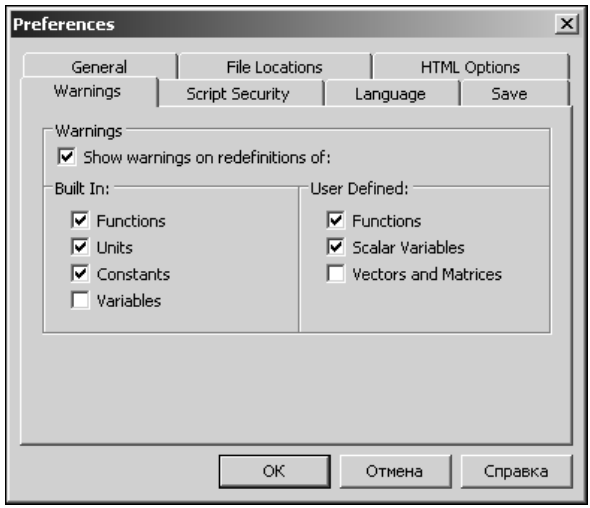


Рис. 1.47. Диалоговое окно **Preferences** открыто на вкладке **Warnings**

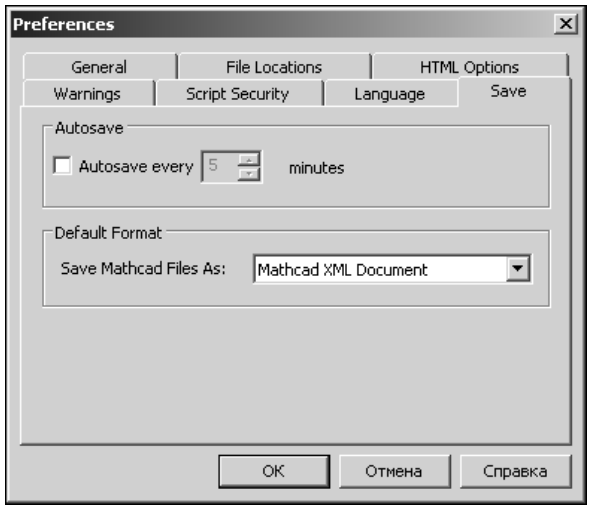


Рис. 1.48. Диалоговое окно **Preferences** открыто на вкладке **Save**

Вкладка содержит всего один раскрывающийся список **Save Mathcad Files As**, в котором указывают определенный по умолчанию формат для сохранения документов Mathcad. Доступны такие форматы: **Mathcad XML Document** (документ XML) и **Mathcad Compressed XML Document** (документ XML со сжатием), а в Mathcad 12 также формат **Mathcad Worksheet** (рабочий документ Mathcad). По умолчанию установлен формат **Mathcad XML Document**.

Таким образом, в этой главе были описаны основные элементы и методы настройки графического интерфейса пользователя. В Mathcad существуют достаточно широкие возможности по управлению рабочими документами. Во многом они существенно зависят от используемых объектов. В связи с этим соответствующие темы рассматриваются в следующих главах, в контексте решаемых прикладных задач. Кроме того, следует отметить, что иногда бывает довольно сложно отделить непосредственно настройки графического интерфейса от методов работы с документом в целом или отдельными его объектами. Поэтому в дальнейшем по ходу книги, если возникает такая необходимость, приводятся сведения о методах настройки системы или более детально уточняются те, что были изложены в этой главе.

Глава 2

Основы работы с документами

В этой главе рассказывается о том, как в Mathcad выполняются базовые, наиболее простые, математические операции. Методы решения различных математических задач, арифметические вычисления, решение уравнений, нахождение интегралов и производных описываются достаточно кратко, на таком уровне, чтобы можно было составить представление о возможностях системы. Более детально эти вопросы обсуждаются в последующих главах книги. Здесь же основное внимание уделяется, в первую очередь, взаимодействию пользователя с приложением в процессе ввода команд. Кроме того, излагаются некоторые сведения, полезные при форматировании данных в рабочих документах.

Математические выражения и команды пользователя

Как уже отмечалось в *главе 1*, после запуска приложения открывается пустое рабочее окно. Все команды пользователя выполняются именно в нем. Команды вводятся в месте размещения указателя в рабочей области. Последний по умолчанию имеет вид красного крестика. Изменить его положение на рабочем листе можно либо щелчком мыши, либо с помощью клавиш со стрелками. Что касается непосредственно команд пользователя, то они также могут вводиться с клавиатуры или выбором элементов на панелях меню. Само собой разумеется, что для эффективной работы с приложением необходимо знать хотя бы основные правила ввода команд.

Математические выражения

Самыми простыми, пожалуй, являются арифметические операции. В Mathcad для выполнения сложения, вычитания, умножения и деления используются стандартные математические операторы, а именно: оператор сложения (+),

вычитания (–), умножения (*) и деления (/). Эти операторы обычно вводятся непосредственно с помощью клавиатуры, однако их же можно ввести, щелкая на кнопках математической палитры **Calculator**. Кроме упомянутых основных операторов, в рабочих листах Mathcad можно вводить и более сложные. В табл. 2.1 перечислены эти операторы с описанием их назначения и методов ввода в рабочий документ.

Таблица 2.1. *Математические операторы*

Оператор	**Описание**	**Символ или комбинация клавиш**
Сложение	Бинарный оператор (+), используется для сложения двух чисел	<Shift>+<=>
Сложение с переходом к новой строке	Оператор сложения (бинарный) — такой же, как в предыдущем случае, только операнды вводятся в разных сроках	<Ctrl>+<Enter>
Комплексное сопряжение	Оператор вычисления числа, комплексно сопряженного к данному (в рабочем документе выглядит как “)	<Shift>+<’>
Деление	Оператор (бинарный) деления двух чисел. Дробь отображается в стандартной, общепринятой нотации, т. е. с числителем и знаменателем	</>
Деление в одну строку	Бинарный оператор деления. От предыдущего случая отличие состоит в том, что оператор деления и соответствующие операнды отображаются в одной строке	<Ctrl>+</>
Факториал	Унарный оператор вычисления факториала (!). Указывается после операнда	<Shift>+<1>
Возведение в степень	Бинарный оператор возведения в степень (^). Первый операнд задает число, возводимое в степень, второй — степень этого числа	<Shift>+<6>
Модуль	Модуль числа. Число заключается в прямые скобки	<Shift>+<\>
Умножение	Бинарный оператор умножения чисел	<Shift>+<8>
Отрицание	Унарный оператор отрицания (знак "минус")	<–>
Корень из числа	Вычисление корня заданной степени из числа	<Ctrl>+<\>
Произведение	Вычисление произведения набора чисел	<Ctrl>+<Shift>+<3>

Таблица 2.1 *(окончание)*

Оператор	**Описание**	**Символ или комбинация клавиш**
Произведение по диапазону	Вычисление произведения набора чисел. Индексная переменная задается в виде диапазона	<Shift>+<3>
Сумма по диапазону	Вычисление суммы нескольких слагаемых. Переменная суммирования задается в виде диапазона	<Shift>+<4>
Квадратный корень	Корень квадратный из числа	<\>
Вычитание	Бинарный оператор вычисления разности двух чисел	<–>
Сумма	Вычисление суммы нескольких слагаемых	<Ctrl>+<Shift>+<4>

Ввод выражений в рабочем листе Mathcad имеет ряд особенностей. Они кратко описываются далее. В частности, при вводе чисел или символов последние выделяются линиями редактирования. Это одна горизонтальная и одна вертикальная линии. Некоторые (не все) возможные способы выделения линиями редактирования подвыражений в сложном выражении приведены в табл. 2.2. Следует отметить, что такую же последовательность выделений можно получить, если в рабочем документе поочередно нажимать клавишу пробела (т. е. <Space>).

Таблица 2.2. *Линии редактирования*

Пример выделения	**Комментарий**
$\dfrac{a\cdot\exp(x)+\sum_{i=1}^{n} i^{2}}{\sin(z)+\prod_{j=1}^{10} j}$	В числителе выражения в символьной сумме выделен показатель степени слагаемых
$\dfrac{a\cdot\exp(x)+\sum_{i=1}^{n} i^{2}}{\sin(z)+\prod_{j=1}^{10} j}$	В числителе выражения выделены слагаемые в символьной сумме

Таблица 2.2 *(окончание)*

Пример выделения	Комментарий
$\dfrac{a\cdot\exp(x)+\sum_{i=1}^{n} i^2}{\sin(z)+\prod_{j=1}^{10} j}$	Выделена символьная сумма в числителе
$\dfrac{a\cdot\exp(x)+\sum_{i=1}^{n} i^2}{\sin(z)+\prod_{j=1}^{10} j}$	В числителе выражения выделена символьная сумма вместе со вторым множителем первого слагаемого числителя
$\dfrac{a\cdot\exp(x)+\sum_{i=1}^{n} i^2}{\sin(z)+\prod_{j=1}^{10} j}$	Выделен числитель
$\dfrac{a\cdot\exp(x)+\sum_{i=1}^{n} i^2}{\sin(z)+\prod_{j=1}^{10} j}$	Выделено все выражение

Как уже отмечалось, выделять разные части выражения можно, нажимая клавишу <Space>. Существенным при этом является то, какая часть выражения выделена первоначально. В зависимости от этого последовательно выделяются разные части выражения. Поскольку перебираются не все возможные способы выделения подвыражений, на это следует обращать внимание. Общую закономерность таких переходов отследить достаточно сложно. Однако на практике можно пользоваться таким правилом: при нажатии клавиши <Space> выделяется минимальное выражение, для которого выделенный на момент нажатия клавиши фрагмент является подвыражением. Расширение выражения, если так можно выразиться, осуществляется в направлении горизонтальной линии редактирования по отношению к вертикальной. После выделения всего выражения следующим выделяется тот фрагмент, с которого все начиналось. При этом различные арифметические операторы имеют равный приоритет.

Кроме клавиши <Space> выделять различные части выражения можно клавишами со стрелками или с помощью мыши. Более того, вертикальная линия редактирования может находиться не только справа или слева от горизонтальной, но и посредине. Примеры приведены в табл. 2.3. Они же иллюстрируют последовательность выделения разных фрагментов выражения при нажатии клавиши <Space>.

Таблица 2.3. *Положение линий редактирования*

Пример выделения	**Комментарий**
$\frac{a\cdot \underline{\exp}(x) + \sum_{i=1}^{n} i^2}{\sin(z) + \prod_{j=1}^{10} j}$	В числителе выражения выделено название экспоненциальной функции. При этом горизонтальная линия редактирования находится посредине
$\frac{a\cdot \underline{\exp}(x) + \sum_{i=1}^{n} i^2}{\sin(z) + \prod_{j=1}^{10} j}$	После нажатия клавиши <Space> горизонтальная линия редактирования смещается к ближайшей границе горизонтальной линии редактирования. В данном случае это левая граница
$\frac{a\cdot \underline{\exp(x)} + \sum_{i=1}^{n} i^2}{\sin(z) + \prod_{j=1}^{10} j}$	Последующее нажатие на клавише <Space> приводит к выделению линиями редактирования всей экспоненциальной функции, вместе с ее аргументом
$\frac{a\cdot \exp(x) + \sum_{i=1}^{n} i^2}{\sin(z) + \prod_{j=1}^{10} j}$	Выделена экспоненциальная функция и символьная сумма в числителе
$\frac{a\cdot \exp(x) + \sum_{i=1}^{n} i^2}{\sin(z) + \prod_{j=1}^{10} j}$	Выделено все выражение. Если нажать на клавише <Space>, выделяется экспоненциальная функция (без аргумента). Горизонтальная линия редактирования будет расположена на левой границе, т. е. результат будет соответствовать второй, а не первой строке данной таблицы

При вводе операторов автоматически, в зависимости от их типа (т. е. бинарные, унарные и т. п.) в рабочем документе, в области ввода выражения появляются структурные заполнители, вместо которых и вводятся числа или символы. На рис. 2.1 представлена ситуация, когда в рабочий документ введен

оператор сложения (+). В результате автоматически добавляются два структурных заполнителя (справа и слева от знака плюс).

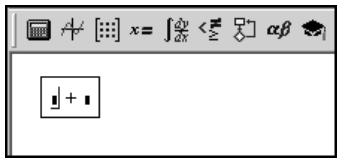


Рис. 2.1. Структурные заполнители

Очень часто в рабочих документах объявляются переменные или вычисляется их значение. Для этих целей используют операторы присваивания (:=) и равенства (=) соответственно.

Пример 2.1. Сложение чисел

Например, чтобы сложить два числа 3 и 4, достаточно ввести в рабочий лист команду `3+4=`. Сразу после ввода знака равенства справа от него появится результат, т. е. число 7 (рис. 2.2).

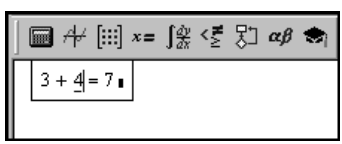


Рис. 2.2. Вычисление суммы чисел

Точно так же вычисляются и прочие выражения, в том числе и те, что содержат переменные. При этом допускаются скобки, которые необходимы для определения правильного порядка выполнения арифметических операций. Зачастую, даже если скобки и не несут прямой функциональной нагрузки, они придают выражению более простой и понятный вид. Поэтому можно рекомендовать их при вводе больших выражений.

Пример 2.2. Вычисление сложного выражения

В качестве примера на рис. 2.3 показан результат вычисления выражения, содержащего операторы сложения, умножения, деления и вычитания.

На рисунке сохранено первое выражение. Такой вид оно будет иметь после того, как с выражения снято выделение. Чтобы снять выделение, достаточно после его вычисления (т. е. ввода знака равенства), нажать клавишу <Enter> или щелкнуть мышью за пределами области выражения.

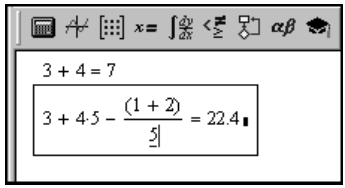


Рис. 2.3. Вычисление сложного выражения

Пример 2.3. Инициализация переменной

Как уже отмечалось, важным моментом при работе с документами Mathcad является инициализация переменных. Самая простая ситуация — когда переменной присваивается численное значение. Например, чтобы переменной x присвоить значение $\sqrt{2}\sin(\pi/6)$, следует ввести в рабочий документ команду $x := \sqrt{2} \cdot \sin\left(\frac{\pi}{6}\right)$ (рис. 2.4).

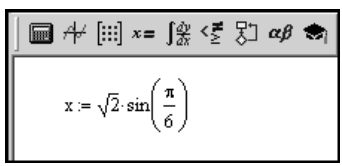


Рис. 2.4. Инициализация переменной

Чтобы в дальнейшем узнать значение переменной x, в рабочем документе вводится название переменной и знак равенства после него (рис. 2.5).

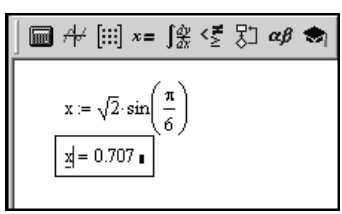


Рис. 2.5. Определение значения переменной

Следует отметить, что по умолчанию это значение в общем случае является приближенным, поскольку возвращается в виде числа с плавающей точкой. В символьном виде значение переменной вычисляется с помощью команд меню **Symbolics**. Об этом речь пойдет в *главе 5*.

Пример 2.4. Знак равенства

В некоторых случаях знак равенства может служить не только для определения значения переменной, но и для ее инициализации. Дело в том, что в Mathcad попытка узнать (указав после переменной знак равенства) значение переменной, которая предварительно не инициализирована, интерпретируется как ее инициализация. Так, если переменной a значение еще не присваивалось, то ввод знака равенства после ее названия будет эквивалентен вводу оператора присваивания (рис. 2.6).

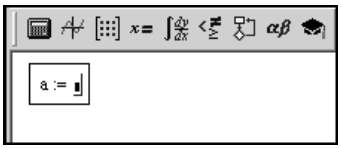


Рис. 2.6. Инициализация переменной с помощью знака равенства

Однако если значение переменной уже присвоено, то ввод знака равенства будет способом вычисления значения переменной (рис. 2.7).

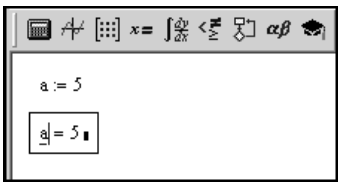


Рис. 2.7. После инициализации переменной знак равенства используется для вычисления значения

Помимо чисел, в качестве значения переменным можно присваивать выражения или строки. Об этом речь идет далее, а здесь остановимся детальнее на способах объявления переменных в рабочем документе.

Способы инициализации переменных

Все команды в рабочем документе Mathcad выполняются последовательно, т. е. в той очередности, как они введены в рабочий лист. Поэтому, например, чтобы использовать в выражениях какую-то именованную константу (т. е. переменную с фиксированным значением), эту константу следует инициализировать до ее первого использования. Это разумно, но не всегда удобно, особенно если документ Mathcad является своеобразным черновиком для проведения оценочных вычислений, результат которых сложно спрогнозировать. Обычно в этих случаях приходится часто менять подгоночные параметры,

уточнять значения переменных и т. п. Если документ достаточно большой и переменных в нем много, все такие переменные удобно собирать в отдельные группы. Тогда их легче найти в документе. При этом, как правило, применяется *глобальная инициализация переменных*.

Дело в том, что способ инициализации с помощью оператора присваивания, который упоминался ранее, подразумевает, что переменная сначала объявляется в документе (то есть ей присваивается значение), и только после этого используется. Такой способ инициализации называется *локальным* (не следует путать с локальным определением переменных в программных кодах). При *глобальной инициализации* первоначальное значение переменной можно присваивать в любом месте рабочего документа. Выполняется глобальная инициализация с помощью оператора тождества (три горизонтальные линии ≡). В рабочем документе этот оператор можно ввести, если нажать комбинацию клавиш <Shift>+<`> (т. е. ввести с клавиатуры символ ~, который в рабочем документе будет отображен как оператор тождества) или воспользоваться соответствующей пиктограммой на палитре **Evaluation** (рис. 2.8).

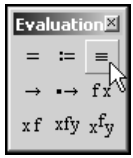


Рис. 2.8. Вставка оператора тождества с помощью палитры **Evaluation**

Пример 2.5. Глобальное определение переменной

На рис. 2.9 приведен пример глобального определения переменной.

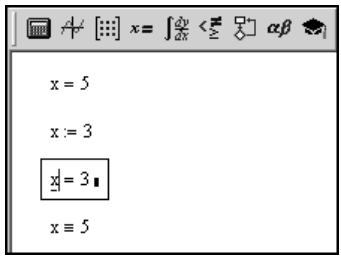


Рис. 2.9. Глобальное определение переменной

В данном случае переменная инициализирована последней командой `x ≡ 5` (четвертая строка сверху) с помощью оператора тождества (ей присвоено значение 5). Таким образом, это глобальное определение переменной, и поэтому

соответствующее значение доступно в рамках всего документа, независимо от места вызова значения переменной в рабочем документе. В первой командной строке на рис. 2.9 после того, как за названием переменой указан знак равенства, возвращается именно то значение, что было присвоено при глобальной инициализации переменной (т. е. значение 5). Если в дальнейшем значение переменной переопределить (вторая по счету команда `x := 3`), то именно такое значение и будет возвращаться для данной переменной (третья команда).

Заданные по умолчанию операторы присваивания (локального и глобального) можно менять. Например, можно выбрать режим, при котором оператор присваивания в рабочем документе будет отображаться в виде обычного знака равенства. Для этого нужно выбрать команду **Tools | Worksheet Options**, в результате чего открывается одноименное диалоговое окно **Worksheet Options**. В этом окне следует перейти к вкладке **Display** (рис. 2.10).

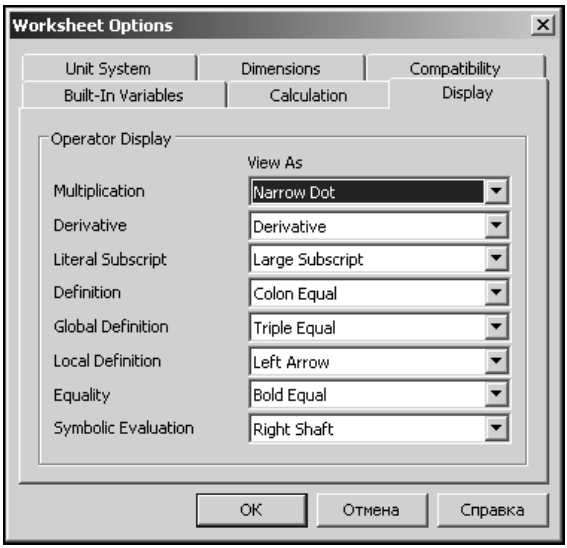


Рис. 2.10. Диалоговое окно **Worksheet Options** открыто на вкладке **Display**

На этой вкладке справа от поля **Definition** в раскрывающемся списке можно выбрать один из двух элементов: **Colon Equal** (установлен по умолчанию и соответствует оператору присваивания, т. е. двоеточие и знак равенства) и **Equal** (в качестве оператора присваивания отображается знак равенства). Если выбрать элемент **Equal**, все операторы присваивания в рабочем документе будут заменены на знаки равенства, как показано на рис. 2.11.

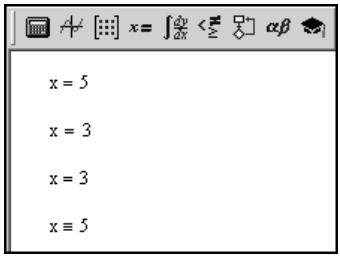


Рис. 2.11. В качестве оператора присваивания используется знак равенства

К сожалению, в такой ситуации бывает крайне проблематично определить, какой знак равенства соответствует вычислению значения переменной, а какой — присваиванию ей значения. В этом легко убедиться из того же рис. 2.11. Чтобы узнать, какой именно оператор в команде, эту команду следует выделить (рис. 2.12).

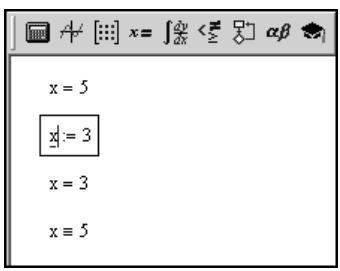


Рис. 2.12. Выделение команды с оператором присваивания

Подобно тому, как устанавливается способ отображения оператора присваивания, можно изменить и способ отображения оператора глобального присваивания. В частности, для этого следует все в том же окне **Worksheet Options** на вкладке **Display** в списке поля **Global Definition** выбрать элемент **Equal**, что соответствует отображению оператора глобального присваивания в виде знака равенства (по умолчанию установлено значение **Triple Equal** для отображения этого оператора в виде знака тождества). На рис. 2.13 представлена ситуация, когда операторы локального и глобального присваиваний отображаются в виде знака равенства.

Совершенно очевидно, что теперь определить, какой оператор встречается в той или иной команде, практически невозможно. Из ситуации выходят, как и ранее, выделяя нужную команду. На рис. 2.14 выделена команда с оператором глобального присваивания.

Хотя возможность переопределения способа отображения операторов иногда весьма кстати, злоупотреблять ею все же не стоит. Разумно ограничиваться настройками по умолчанию.

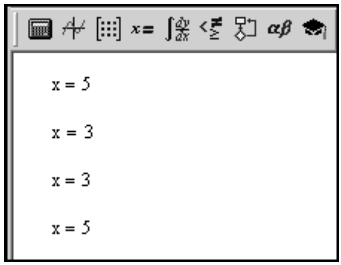


Рис. 2.13. Отображение операторов локального и глобального присваивания с помощью знака равенства

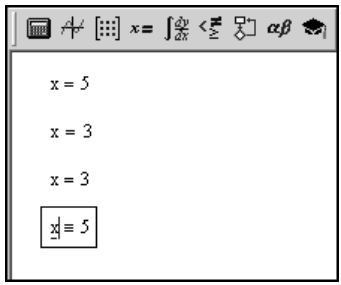


Рис. 2.14. Выделена команда с оператором глобального присваивания

Хочется также обратить внимание на то, что на вкладке **Display** диалогового окна **Worksheet Options** имеется поле **Local Definition**. Несмотря на характерное название, это поле никакого отношения к обсуждавшемуся оператору локального присваивания не имеет. Там устанавливается способ (вид оператора) декларирования локальных переменных в программных модулях.

Работа с числами

При инициализации переменных с численными значениями или просто при вводе чисел в рабочем документе Mathcad следует иметь в виду несколько принципиальных моментов. Это позволит существенно облегчить работу с приложением.

Особенности ввода числовых значений

Понятно, что ввод целых чисел особых проблем вызвать не должен. Несложен ввод и небольших (по абсолютному значению) рациональных чисел. Целая и десятичная части числа разделяются точкой. Неудобства возникают при вводе больших чисел. В этом случае разумно прибегнуть к научной нотации.

Пример 2.6. Научная нотация

Число представляется в виде произведения мантиссы и десяти в целочисленной степени. Например, число 2000 000 можно ввести с помощью команды `2*10^6` (в рабочем документе это может выглядеть так, как показано на рис. 2.15).

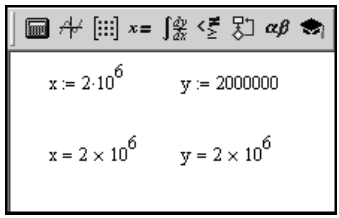


Рис. 2.15. Ввод больших чисел

На рис. 2.15 проиллюстрирована ситуация, когда двум переменным `x` и `y` присваиваются одинаковые значения, но в разной нотации. Присваиваемое первой переменной значение представлено в виде произведения, а значение второй — в обычном для целого числа виде. В следующей строке проверяются значения переменных. Как видно из рис. 2.15, вне зависимости от способа присваивания значения, результат отображается одинаково. Однако способ отображения численных результатов можно переопределять.

Настройки форматирования численных данных

Существует два принципиальных способа форматирования данных: задать форматирование для всего рабочего документа или только для отдельных выражений в документе. Это относится не только к форматированию действительных чисел, но и данных прочих типов. Выполнить настройки форматирования данных (и не только численных) можно с помощью диалогового окна **Result Format**, которое открывается после выбора команды **Format | Result** (рис. 2.16).

Окно содержит четыре вкладки: **Number Format**, **Display Options**, **Unit Display** и **Tolerance**. Интерес в данном случае представляет вкладка **Number Format**, которая и открыта на рис. 2.16. В левой части вкладки имеется список **Format**, содержащий пять элементов. Выбрав тот или иной элемент, можно задать настройки форматирования численных данных в рабочем документе или отдельном выражении. Если перед выполнением команды **Format | Result** было выделено какое-то выражение (команда) в рабочем документе Mathcad, то настройки будут применены только к этому выражению. Если никакое выражение на момент вызова диалогового окна выделено не было

(для этого достаточно щелкнуть мышью в рабочем документе вне области команд и выражений), то настройки применяются ко всему документу.

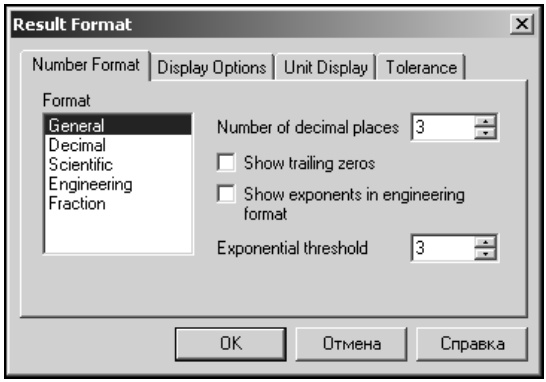


Рис. 2.16. Диалоговое окно **Result Format**

Выбрав элемент **General**, можно задать ряд параметров, определяющих способ отображения численных данных. В частности, при выбранном элементе **General** в правой части диалогового окна **Format Result** представлены две опции **Show trailing zeros** и **Show exponents in engineering format** и два поля **Number of decimal places** и **Exponential threshold**. Поставив флажок опции **Show trailing zeros**, переходят в режим, при котором отображаются незначащие нули в десятичной части чисел. Эта опция имеет отношение к полю **Number of decimal places**, где указывается количество цифр, отображаемых после десятичной точки. Таким образом, если в десятичной части числа цифр меньше, чем указано в поле **Number of decimal places**, а опция **Show trailing zeros** активизирована, то десятичная запись числа дополняется нулями.

Пример 2.7. Числовые форматы

Если оставить предлагаемое по умолчанию значение 3 и установить флажок опции **Show trailing zeros**, то число `123.45` (оно присваивается в качестве значения переменной) будет отображено в виде `123.450`, как это показано на рис. 2.17.

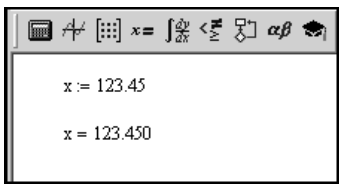


Рис. 2.17. Отображение числа с незначащими нулями в десятичной части

В поле **Exponential threshold** указывается число цифр в целой части числа, при превышении которого число отображается в экспоненциальной форме, т. е. с мантиссой, умноженной на десять в некоторой степени. В качестве иллюстрации проделаем следующую процедуру. Установим значение поля **Exponential threshold** равным 4 и выставим флажок опции **Show trailing zeros** (если он до этого не был установлен). Далее переменной `x` присвоим в качестве значения целое число `1230`, после чего значение этой переменной вычисляется (рис. 2.18).

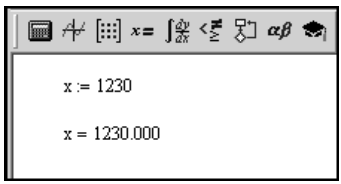


Рис. 2.18. Инициализация значения переменной и отображение ее значения в обычной нотации

Поскольку число цифр во введенном числе равно 4 и не превышает порогового значения в поле **Exponential threshold**, то значение соответствующей переменной будет отображено в обычном виде с тремя нулями после десятичной точки (при значении 3 в поле **Number of decimal places**). Если же к значению переменной в команде инициализации добавить еще одну цифру (ноль), т. е. ввести число `12300`, получим несколько иной результат (рис. 2.19).

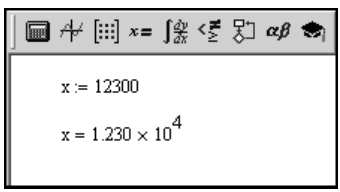


Рис. 2.19. Инициализация значения переменной и отображение ее значения в экспоненциальной нотации

В этом случае значение переменной отображается в экспоненциальной нотации, причем в мантиссе дробная часть указана, в соответствии с настройками поля **Number of decimal places**, тремя цифрами с незначащим нулем.

Пример 2.8. Инженерный формат

Флажок опции **Show exponents in engineering format** устанавливается в том случае, если необходимо отображать значения в экспоненциальной нотации в инженерном формате. Этот формат подразумевает такое же представление,

как и в обычной экспоненциальной нотации, но только показатель степени является числом, кратным трем. Например, при значении полей **Exponential threshold** и **Number of decimal places** равным 3 и установленных флажках опций **Show trailing zeros** и **Show exponents in engineering format** для переменных, которые инициализируются со значениями `12345` и `1200000`, в последующем значения возвращаются в таком виде, как показано на рис. 2.20.

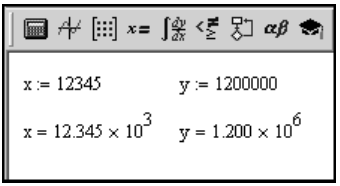


Рис. 2.20. Использование инженерного формата

Общий формат **General**, обсуждавшийся ранее, хотя и удобен, но не всегда приемлем. Иногда бывает необходимо обеспечить однотипность отображения численных результатов. На этот случай предусмотрены прочие элементы списка **Format** диалогового окна **Result Format**. Так, если выбрать элемент **Decimal** из упомянутого списка, вкладка **Number Format** примет вид, как показано на рис. 2.21.

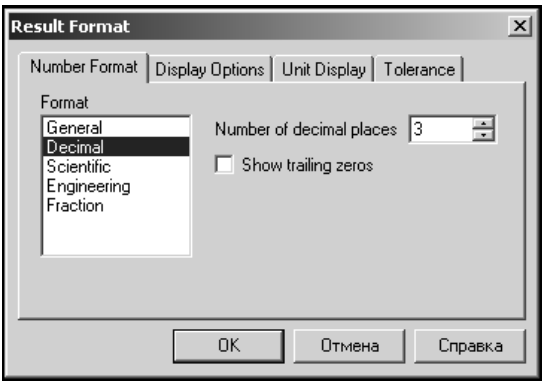


Рис. 2.21. Выбор элемента **Decimal** из списка **Format**

Вкладка содержит одно поле и одну опцию. В поле **Number of decimal places** указывается количество цифр, отображаемых в десятичной части числа. Если установлен флажок опции **Show trailing zeros**, в десятичной части числа будут, в случае потребности, отображаться незначащие нули. Применение формата **Decimal** означает, что числа представляются только в десятичном виде. Научная нотация, другими словами, при таком формате не применяется.

Для отображения чисел в научной нотации (в виде мантиссы, умноженной на десять в целочисленной степени), в списке **Format** диалогового окна **Result Format** выбирают элемент **Scientific**. При этом диалоговое окно **Result Format** будет выглядеть так, как показано на рис. 2.22.

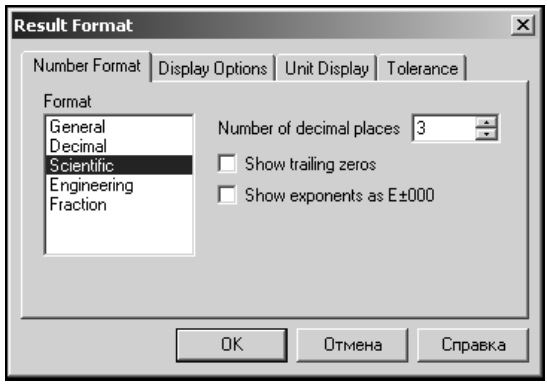


Рис. 2.22. Выбор элемента **Scientific** из списка **Format**

В этом случае, помимо уже знакомого читателю поля **Number of decimal places** и опции **Show trailing zeros** (назначение у них такое же, как и ранее), присутствует еще и опция **Show exponents as E±000**. Установив флажок этой опции, переходят в режим, при котором вместо множителя-десятки в соответствующей степени указывается символ E и степень, в которую возводится десять. При выборе формата **Scientific** все численные результаты отображаются только в научной нотации. Для формата **Engineering** настройки точно такие же, как и для **Scientific**. Разница состоит только в том, что в данном случае используется инженерный формат.

Пример 2.9. Различные форматы

В документе на рис. 2.23 показан результат применения различных форматов: десятичного (**Decimal**), научного (**Scientific**) и инженерного (**Engineering**) для одних и тех же численных данных.

Следует отметить, что в данном случае форматы применялись не ко всему документу, а только к отдельным результатам. Для этого выполняется двойной щелчок мышью на числе в выражении. В результате открывается диалоговое окно **Result Format**, в котором и делаются нужные настройки. Они, как отмечалось, применяются не ко всему документу, а только к тому численному результату, на котором был выполнен двойной щелчок мышью. Что касается непосредственно настроек, то во всех трех случаях применялся режим отображения незначащих нулей (количество цифр после десятичной точки — 3).

Кроме того, в научном и инженерном форматах для переменной z активизировалась опция **Show exponents as E±000**.

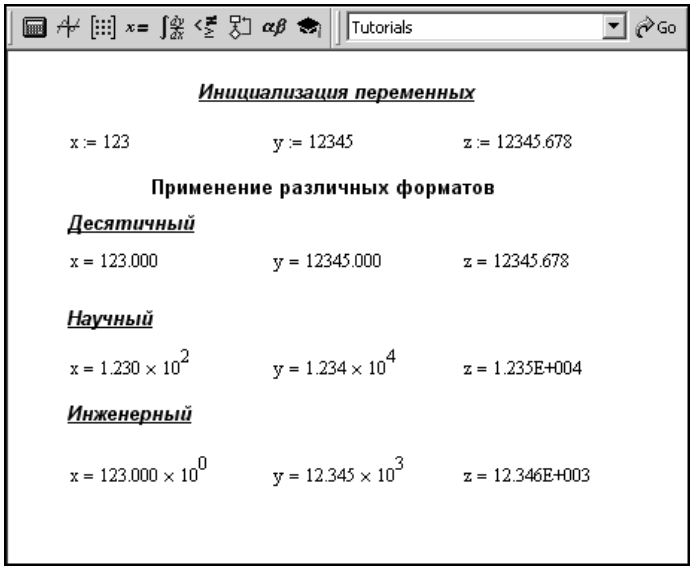


Рис. 2.23. Применение различных форматов

Наконец, можно установить формат отображения численных данных в виде дробей. Для этого из списка **Number Format** выбирают элемент **Fraction** (рис. 2.24).

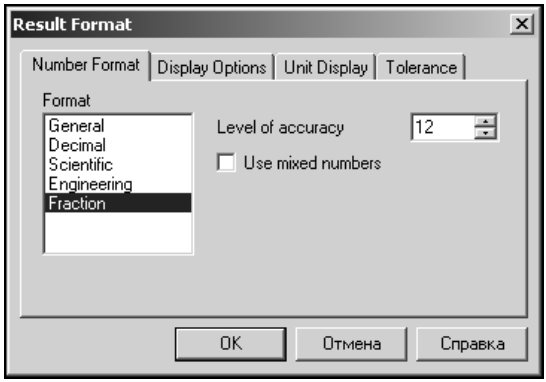


Рис. 2.24. Установка формата отображения чисел в виде дроби

После применения данного формата все числа в рабочем документе (если, разумеется, формат применялся к документу, а не к отдельному выражению)

отображаются в виде обычной дроби — с числителем и знаменателем. Степень точности задается в поле **Level of accuracy**. Она влияет на близость дробного числа к его выражению в формате с десятичной точкой. Чем больше значение в поле **Level of accuracy**, тем выше точность.

Пример 2.10. Выделение целой части

Кроме этого, можно установить флажок опции **Use mixed numbers**. При установленном флажке в дробных выражениях выделяется целая часть, в противном случае — нет. Особенности отображения дробного числа в режиме **Use mixed numbers** иллюстрирует рис. 2.25.

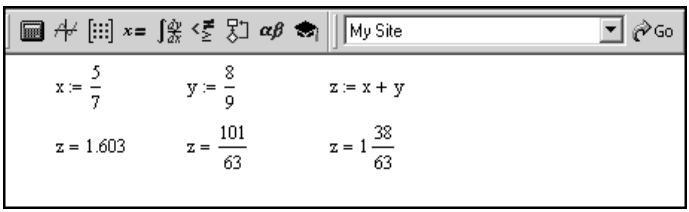


Рис. 2.25. Выделение целой части в дробном выражении

Пример 2.11. Использование дробей

В случае применения дробного формата все численные результаты в рабочем документе представляются в виде дробей. На рис. 2.26 инициализированы три переменные, причем каждой из них присваивается численное значение в разном формате.

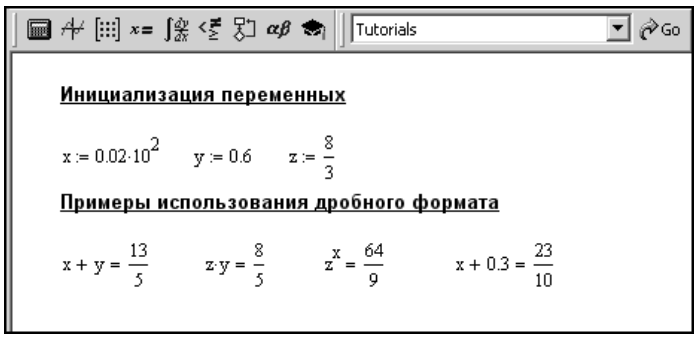


Рис. 2.26. Использование дробного формата

Тем не менее, результат операций с этими переменными представляется в виде дробных выражений (в пределах установленной в поле **Level of accuracy** точ-

ности). По умолчанию в дробных выражениях целая часть не выделяется. Для того чтобы изменить ситуацию, как уже отмечалось, следует установить флажок опции **Use mixed numbers**. На рис. 2.27 представлен тот же документ Mathcad, что и в предыдущем случае, но только теперь опция **Use mixed numbers** активизирована.

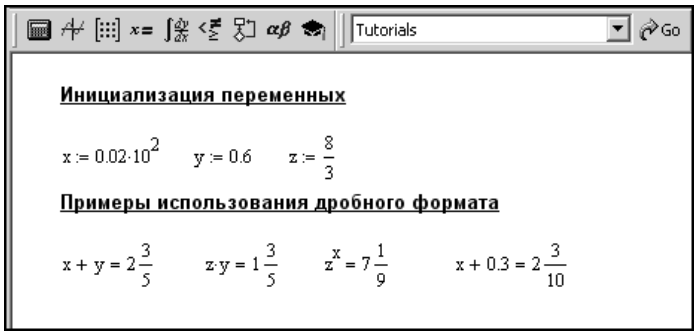


Рис. 2.27. Выделение целой части в дробях

Как следствие активизирования опции **Use mixed numbers**, в численных дробных выражениях выделяется целая часть, но результат при этом не меняется.

Использование комплексных чисел

Помимо действительных чисел, в Mathcad можно работать и с комплексными. Символами мнимости числа может быть либо `i`, либо `j`. Причем для ввода мнимой единицы один ее символ указать недостаточно. В этом случае `i` или `j` будут интерпретироваться как обычные переменные. Чтобы Mathcad воспринимал эти переменные именно как мнимую единицу, перед ними указывается (без знака умножения — это важно!) число, задающее мнимую часть. Например, комплексная единица может быть введена как `1i` или `1j`, по выбору пользователя. При этом единица в мнимой части комплексного числа не отображается. Она видна только при выделении соответствующего выражения. Если число содержит не только мнимую часть, но и действительную, оно вводится как сумма действительной и мнимой частей. В одном рабочем документе для обозначения мнимой части чисел допустимы одновременно оба символа (`i` и `j`), однако по умолчанию независимо от того, какая литера выбрана, в процессе выполнения математических операций над комплексными числами она меняется на `i`. Следует также отметить, что ввести мнимую единицу можно с помощью палитры **Calculator**, выбрав на ней пиктограмму с литерой **i**.

Пример 2.12. Комплексные числа

С комплексными числами в Mathcad можно выполнять практически те же операции, что и с действительными. В частности, их можно складывать, вычитать, умножать и делить. Комплексные числа также могут указываться аргументами функций. Пример приведен на рис. 2.28.

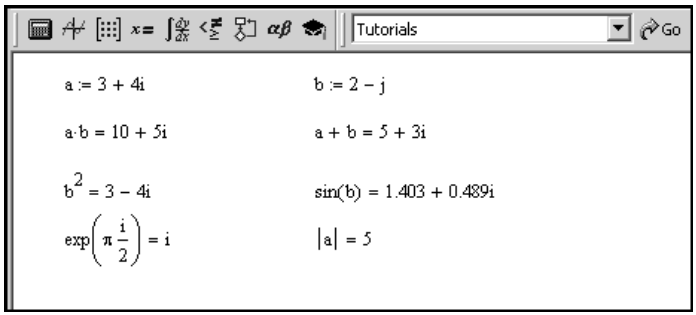


Рис. 2.28. Операции с комплексными числами

Еще раз хочется обратить внимание на то, что мнимая часть комплексного числа вводится без знака умножения. Нагляднее это продемонстрировано на рис. 2.29. Там выделена команда, которой переменной присваивается в качестве значения комплексное число с мнимой частью, равной `-1`. Эта единица обязательно указывается перед символом комплексной единицы (т. е. `i` или `j`). Знак умножения при этом не нужен.

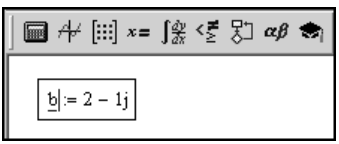


Рис. 2.29. Выделено выражение с комплексным числом

Более того, если знак умножения все же поставить, то число как комплексное восприниматься системой Mathcad не будет. Поскольку, как уже отмечалось, единица в мнимой части комплексного числа не отображается, без предварительного выделения выражения бывает сложно определить, как именно оно вводилось. В качестве примера можно привести ситуацию с вычислением экспоненты от комплексного числа, мнимая часть которого равна $\pi/2$. Значение выражения вычислено на рис. 2.28, а на рис. 2.30 эта же команда приведена выделенной.

Число π можно ввести либо с помощью палитры **Calculator** (рис. 2.31), либо нажав комбинацию клавиш <Ctrl>+<Shift>+<P>.

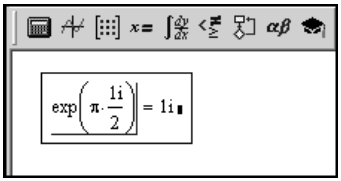


Рис. 2.30. Выделено выражение с функцией, у которой мнимый аргумент

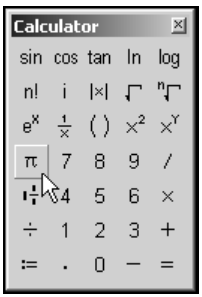


Рис. 2.31. Ввод числа π с помощью палитры **Calculator**

Сложность состоит в том, что если после числа π ввести символ мнимой единицы (i или j), то автоматически будет добавлен и знак умножения. В результате число в аргументе как комплексное восприниматься не будет. Проблема может быть решена так: вводится число π, после чего это число умножается на мнимую единицу, т. е. на `1i` (или `1j`).

Символ мнимой части комплексных чисел (`i` или `j`) можно переопределить, воспользовавшись командой **Format | Result**. В результате открывается диалоговое окно **Result Format**, в котором следует перейти к вкладке **Display Options** (рис. 2.32).

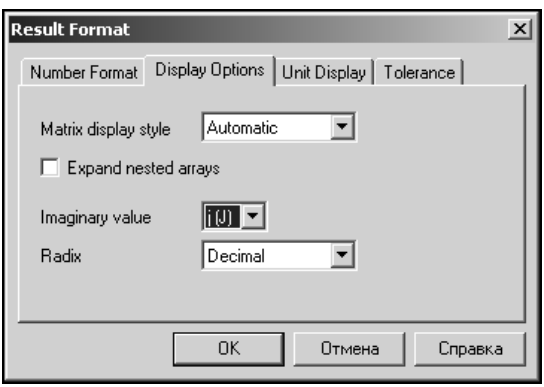


Рис. 2.32. Выбор символа для отображения мнимой части

Вкладка **Display Options**, помимо прочего, содержит раскрывающийся список **Imaginary Value**, в котором два элемента. Выбрав один из них, устанавливают символ отображения мнимой части числа. В частности, на рис. 2.33 представлен рассмотренный ранее документ, в котором изменены настройки: теперь мнимая часть отображается с символом `j`.

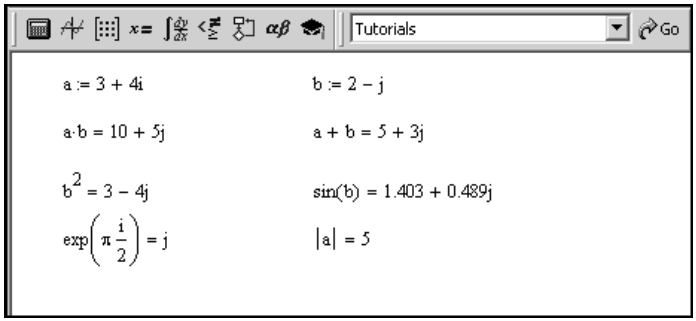


Рис. 2.33. Использование символа `j` для отображения мнимой части числа

Символ `j` обычно присутствует при решении задач, связанных с расчетом электрических цепей, поскольку по сложившейся традиции символом `i` обозначается сила тока. Хотя, как известно, традиции именно для того и существуют, чтобы иногда их менять.

Различные системы счисления

Достаточно часто возникает необходимость в числах, представленных в системах счисления, отличных от десятичной. Наиболее популярными являются двоичная и шестнадцатеричная системы счисления. В двоичной системе, как известно, число представляется только с помощью нулей и единиц. Если позиционная запись числа в двоичной системе счисления имеет вид $a_n a_{n-1} ... a_2 a_1 a_0$ (где $a_i = 0,1$ для всех $i = 0,1,...,n$), то в десятичной системе счисления это число равняется $a_0 2^0 + a_1 2^1 + a_2 2^2 + ... + a_{n-1} 2^{n-1} + a_n 2^n$. Например, двоичное число 101 в десятичной системе равно 5 ($1 \cdot 2^0 + 0 \cdot 2 + 1 \cdot 2^2 = 5$). Аналогичная ситуация с представлением чисел в шестнадцатеричной системе. Для числа с позиционной записью вида $a_n a_{n-1} ... a_2 a_1 a_0$ в шестнадцатеричной системе значение вычисляется как $a_0 16^0 + a_1 16^1 + a_2 16^2 + ... + a_{n-1} 16^{n-1} + a_n 16^n$. Но только в этом случае a_i могут принимать значения от 0 до 15. Поскольку отдельными цифрами в десятичной системе задаются числа от 0 до 9, то для чисел от 10 до 15 используют латинские буквы: A (число 10), B (число 11),

C (число 12), D (число 13), E (число 14), F (число 15). Например, десятичное число 2589 в шестнадцатеричной системе может иметь вид A1D ($a1d = 13 \cdot 16^0 + 1 \cdot 16 + 10 \cdot 16^2$). В Mathcad допустимы как двоичные, так и шестнадцатеричные числа. Возможна также работа с числами, записанными в восьмеричной системе счисления, в которой число с позиционной записью $a_n a_{n-1} ... a_2 a_1 a_0$ представимо в виде $a_0 8^0 + a_1 8^1 + a_2 8^2 + ... + a_{n-1} 8^{n-1} + a_n 8^n$, а числа a_i могут принимать значения от 0 до 7.

Пример 2.13. Двоичные числа

Для ввода числа в системе, отличной от десятичной, сначала задается его позиционная запись, после чего указывается буква-индикатор системы счисления. Чтобы число интерпретировалось как двоичное, необходимо сразу после числа, без пробела, указать литеру `b` или `B`. Простой пример приведен на рис. 2.34.

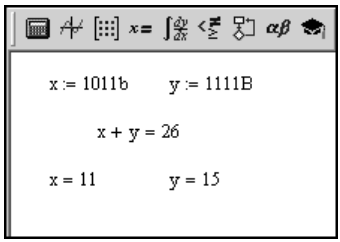


Рис. 2.34. Использование двоичных чисел

Двум переменным, `x` и `y`, присваиваются в качестве значений двоичные числа. Однако если попытаться вычислить значение, например, суммы таких двоичных чисел, результат будет представлен как число в десятичной системе счисления. В таком же виде вычисляется и значение самих переменных `x` и `y` (см. рис. 2.34).

Пример 2.14. Двоичная система

Чтобы изменить систему счисления, в которой отображается результат, следует открыть вкладку **Display Options** диалогового окна **Result Format** (см. рис. 2.32). Система счисления задается выбором одного из элементов раскрывающегося списка **Radix**. В этом списке присутствуют такие элементы: **Decimal** (десятичная система счисления), **Binary** (двоичная), **Octal** (восьмеричная), **Hexadecimal** (шестнадцатеричная). На рис. 2.35 представлен документ, в котором установлен режим отображения результата в двоичной системе счисления.

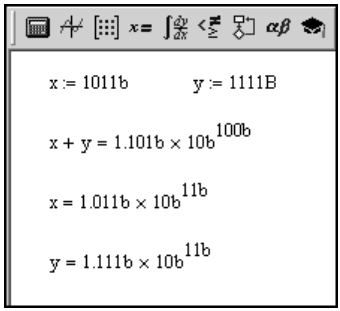


Рис. 2.35. Отображение результата в двоичной системе счисления

При этом имеет место достаточно интересная ситуация. По умолчанию, как известно, к численным значениям применяется общий формат (элемент **General** в списке **Format** вкладки **Number Format** диалогового окна **Result Format**). При тех настройках, что используются, целое число отображается в научной нотации, если его длина превышает три цифры. Это правило остается в силе, когда число представлено и не в десятичной системе счисления. В двоичной системе, очевидно, даже небольшие по величине числа содержат достаточно много цифр. Поэтому при переходе к двоичной системе такие числа представляются в виде мантиссы, умноженной на два в некоторой степени. Причем и мантисса, и степень отображаются в двоичной системе счисления.

Чтобы увидеть числа в обычном представлении, без множителей, можно на вкладке **Number Format** в списке **Format** выбрать элемент **Decimal**. Результат применения этого формата показан на рис. 2.36.

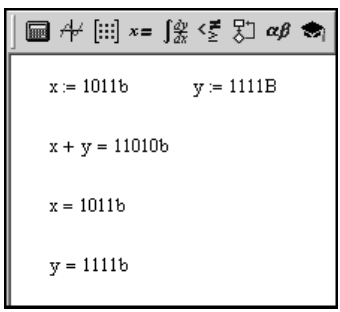


Рис. 2.36. Применение формата **Decimal**
к рабочему документу с двоичными числами

Пример 2.15. Шестнадцатеричные и восьмеричные числа

Индикатором числа в восьмеричной системе счисления является буква `o` или `O`. Наконец, шестнадцатеричные числа отображаются с литерой `h` (или `H`) в конце. Причем шестнадцатеричное число обязательно должно начинаться

с цифры (иначе оно как число интерпретироваться не будет даже при наличии литеры h в конце). Поэтому если в представлении шестнадцатеричного числа первым символом является буква (от a до f), перед ней следует поставить 0. Пример реализации численного результата в различных системах счисления приведен на рис. 2.37.

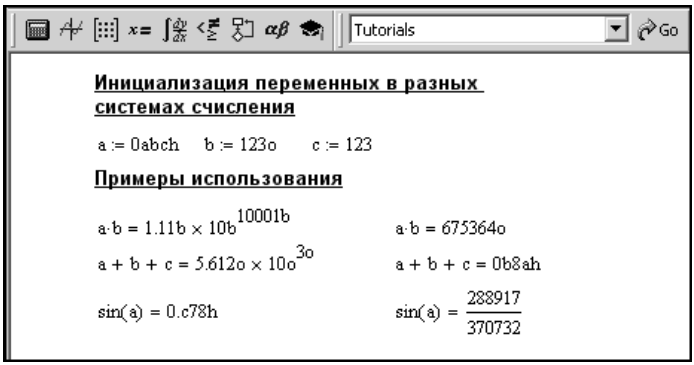


Рис. 2.37. Численные результаты в разных системах счисления

Для каждого выражения применялся свой формат. Стоит напомнить, что применение настроек к отдельному числу осуществляется его выделением и двойным щелчком на нем мышью. В результате открывается диалоговое окно **Result Format** и все настройки выполняются так же, как и при определении формата для целого документа. Что касается непосредственно документа, то в нем инициализируются три переменные (a, b и c). Этим переменным присваиваются численные значения (шестнадцатеричное, восьмеричное и десятичное соответственно). Далее вычисляется произведение первых двух переменных, сумма трех переменных и синус от первой переменной. Все эти выражения вычисляются дважды, и результат представляется по-разному. В левом столбце на рис. 2.37 результат представлен в двоичной, восьмеричной и шестнадцатеричной системах счисления, соответственно. Никакие дополнительные настройки не применялись. Правый столбец содержит данные в восьмеричной системе счисления (с применением формата **Decimal**), шестнадцатеричной системе счисления и численное значение в дробном формате. Причем в последнем случае допускается только десятичная система счисления. Стоит обратить внимание на следующее обстоятельство. При вводе букв-индикаторов системы счисления Mathcad автоматически пытается вставить знак умножения. Его придется удалить вручную, чтобы число интерпретировалось правильно.

Константы

В Mathcad, как и в любом другом аналогичном приложении, есть целый набор предопределенных констант. Это такие фиксированные значения, к которым можно обращаться через имя, например число π. Наиболее часто встречаются математические константы. Помимо упоминавшейся ранее мнимой единицы (`1i` или `1j`), в Mathcad есть константа Эйлера (символ `e` вводится непосредственно с клавиатуры), число π (комбинация клавиш <Ctrl>+<Shift>+<p>), символ бесконечности ∞ (комбинация клавиш <Ctrl>+<Shift>+<z>) и процент, обозначающийся символом `%` (значение этой константы равно 0,01).

Пример 2.16. Использование констант

В документе на рис. 2.38 приведены примеры использования предопределенных констант в рабочем документе Mathcad.

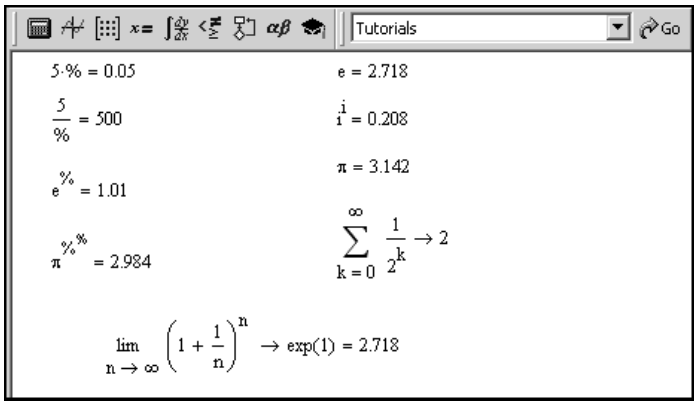


Рис. 2.38. Примеры использования предопределенных констант

Как несложно заметить, численные значения, в силу настроек системы, отображаются с тремя цифрами в десятичной части, хотя точность их вычисления существенно выше (до 15 знаков). Случается, что нужно знать более точное значение в сравнении с тем, которое указано в рабочем листе. Для этого можно изменить глобальные настройки системы (увеличить число отображаемых после запятой знаков). Если достаточно просмотреть точное значение отдельного результата, его следует выделить и нажать комбинацию клавиш <Ctrl>+<Shift>+<N>. При этом в строке состояния будет отображено точное (в пределах возможностей системы) значение численного результата. На рис. 2.39 в левом нижнем углу в строке состояния представлена константа Эйлера, которая была предварительно вычислена как значение предела $\lim\limits_{n\to\infty}(1+1/n)^n$.

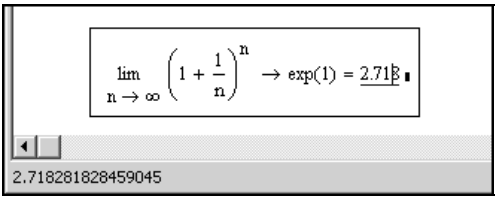


Рис. 2.39. Отображение в строке состояния точного значения

Пользователи Mathcad могут не только определять собственные, но и переопределять перечисленные ранее константы. Переопределение констант осуществляется достаточно просто — как и с обычными переменными с помощью оператора присваивания. Это очень сильно расширяет возможности системы при решении прикладных задач.

Помимо математических констант, в Mathcad есть ряд системных. Они отвечают за основные настройки системы и могут изменяться в процессе работы. Эти константы перечислены в табл. 2.4.

Таблица 2.4. *Системные константы Mathcad*

Системная константа	Описание	Значение по умолчанию
TOL	Задает точность вычисления интегралов и производных. Ее значение также определяет продолжительность итерационного процесса при решении уравнений	0,001
CTOL	Задает точность, с которой выполняются ограничения при решении задач оптимизации	0,001
ORIGIN	Значение этой константы определяет начальный индекс нумерации элементов всех массивов в рабочем документе	0
PRNPRECISION	Определяет количество значащих цифр для численных данных при записи их в файл с помощью функции WRITEPRN	4
PRNCOLWIDTH	Определяет ширину столбцов в файле, в который записываются данные с помощью функции WRITEPRN	8
CWD	Значением константы является текстовая строка с названием рабочей папки (точнее, полный путь к ней)	

Переопределять значения перечисленных констант можно прямо в рабочем документе. Достаточно константе присвоить новое значение. Еще один способ

определения значений системных констант упоминался в *главе 1*. Там речь шла о настройках, выполняемых в диалоговом окне **Worksheet Options** на вкладке **Built-In Variables**. Диалоговое окно открывается выбором команды **Tools | Worksheet Options**.

Несмотря на наличие специального диалогового окна, с помощью которого можно задавать значения системных констант, переопределение их значений непосредственно в рабочем документе представляется более перспективным. В этом случае, фактически, можно для разных данных установить различные настройки.

Пример 2.17. Запись данных в файл

На рис. 2.40 приведен ряд команд, с помощью которых в конечном итоге в файл записываются элементы численной матрицы.

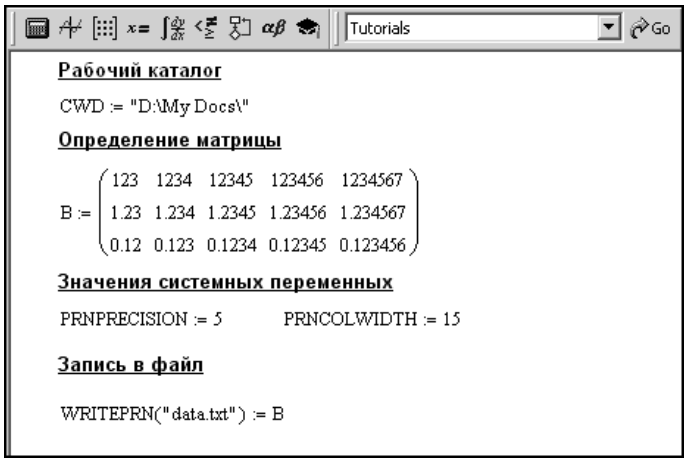


Рис. 2.40. Запись данных в файл

Первой командой `CWD:="D:\My Docs\"` устанавливается текущий рабочий каталог. Далее задается численная матрица `B` (методы работы с матрицами обсуждаются несколько позже). Количество значащих цифр для записи данных в файл определяется командой `PRNPRECISION:=5`, а ширина столбцов — командой `PRNCOLWIDTH:=15`. Наконец, запись в текстовый файл с названием data.txt осуществляется командой `WRITEPRN("data.txt"):=B`. В результате в рабочем каталоге (т. е. D:\My Docs\) элементы матрицы `B` будут записаны в файл data.txt. Результат (содержимое файла data.txt) представлен на рис. 2.41.

Как несложно заметить, все числа отображаются не более чем с пятью значащими цифрами, в соответствии с выполненными настройками. В том случае,

если предварительно файл в рабочем каталоге создан не был, он создается при выполнении команды `WRITEPRN`.

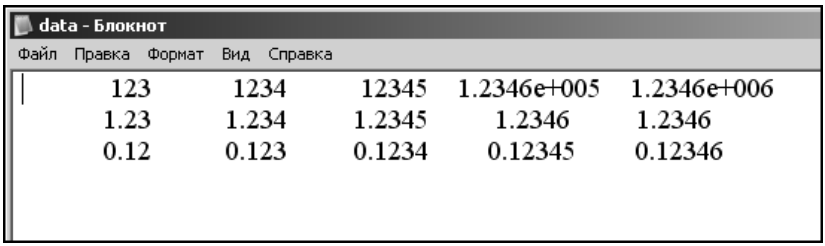


Рис. 2.41. Содержимое файла data.txt

Работа с функциями

Практические возможности любого математического приложения определяются во многом тем набором встроенных функций, которые доступны пользователю в работе. Не является исключением и Mathcad. Сразу оговоримся, что в Mathcad встроенных функций достаточно много. Практически все они математические. Кроме этого, на основе встроенных функций пользователь может определять собственные. Делается это достаточно просто (так же просто, как и большинство других операций в Mathcad). Соответствующие методы описываются сразу после обзора встроенных функций Mathcad.

Вставка встроенных функций

Существует два способа вставки встроенной функции в рабочий документ Mathcad. Можно просто ввести название функции непосредственно с клавиатуры. Однако для этого следует знать и помнить синтаксис вызова функции. Для многих функций их имена в Mathcad совпадают или близки к общепринятым математическим названиям, поэтому с ними проблем не возникает. Все же совершенно очевидно, что запомнить синтаксис абсолютно всех функций практически невозможно, тем более что в Mathcad их много. Второй способ вставки функции в документ подразумевает вызов специального диалогового окна, с помощью которого и осуществляется выбор нужной функции. В этом случае нет необходимости запоминать подробности того, как именно называется функция, сколько у нее аргументов, и в каком порядке они указываются. Кроме того, в вызываемом диалоговом окне можно увидеть также и краткую справку по выбранной функции.

Чтобы открылось диалоговое окно вставки функции **Insert Function** (рис. 2.42), следует выбрать команду **Insert | Function** <Ctrl>+<E>. Кроме этого, можно

воспользоваться специальной кнопкой на стандартной панели инструментов (на кнопке изображен символ функции).

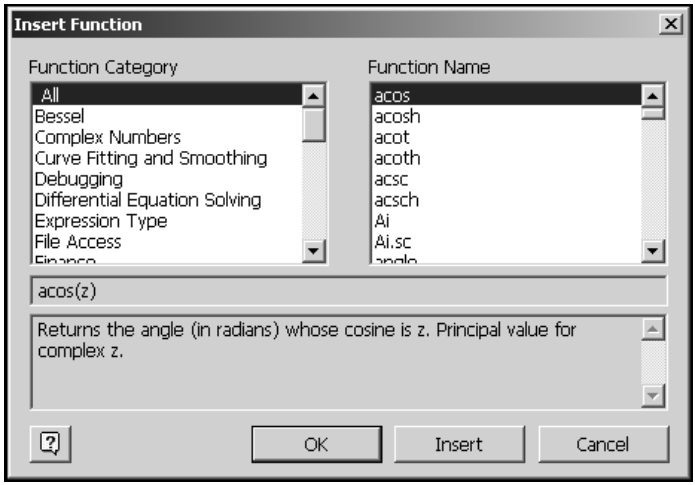


Рис. 2.42. Диалоговое окно **Insert Function**

Диалоговое окно **Insert Function** содержит два списка: **Function Category** и **Function Name**. В первом отображены те категории, на которые условно разбиты все встроенные функции Mathcad. В частности, в списке **Function Category** представлены следующие элементы:

- **All** — при выборе этого элемента в правой части диалогового окна **Insert Function** в списке **Function Name** отображается список всех встроенных функций Mathcad;
- **Bessel** — выбор элемента приводит к тому, что в правой части отображается список функций Бесселя (их довольно много). Элемент выбирают в тех случаях, если вставляемая в рабочий документ функция относится к одному из типов специальных функций Бесселя;
- **Complex Numbers** — в эту категорию входят функции, предназначенные для работы с комплексными числами;
- **Curve Fitting and Smoothing** — категория функций для выполнения аппроксимации и сглаживания;
- **Debugging** — функции отладки кодов;
- **Differential Equation Solving** — функции для решения дифференциальных уравнений и смежных задач;
- **Expression Type** — категория функций для определения типов выражений;

- **File Access** — функции для работы с внешними файлами;
- **Finance** — финансовые функции;
- **Fourier Transform** — категория функций, имеющих отношение к преобразованию Фурье;
- **Graphing** — функции для работы с графическими объектами;
- **Hyperbolic** — гиперболические функции;
- **Image Processing** — категория функций для обработки изображений;
- **Interpolation and Prediction** — группа функций для выполнения интерполяции и экстраполяции;
- **Log and Exponential** — логарифмические и показательные функции;
- **Lookup** — группа функций для выполнения поиска данных;
- **Number Theory/Combinatorics** — группа функций для работы с численными данными;
- **Piecewise Continuous** — кусочно-непрерывные функции;
- **Probability Density** — группа функций, относящихся к теории вероятностей, а именно к определению вероятностей для различных распределений;
- **Probability Distribution** — функции всевозможных распределений;
- **Random Numbers** — категория функций, относящихся к генерированию случайных чисел;
- **Signal Processing** — функции для обработки сигналов;
- **Solving** — выбрав этот элемент, в правой части диалогового окна **Insert Function** можно увидеть список функций, используемых при решении уравнений, поиске экстремумов и т. п.;
- **Sorting** — функции сортировки и упорядочивания данных;
- **Special** — набор некоторых специальных функций и таких, которые по разным причинам не попали в другие категории;
- **Statistics** — статистические функции;
- **String** — функции для работы с текстовыми строками;
- **Trigonometric** — тригонометрические функции;
- **Truncation and Round Off** — функции округления и преобразования значений;
- **User Defined** — определяемые пользователем функции;
- **Vector and Matrix** — функции для работы с векторами и матрицами;
- **Wavelet Transform** — функции для выполнения вейвлет-преобразования.

Выбирать функции из полного списка — занятие не самое простое. С этой точки зрения разумнее предварительно выбрать ту категорию, к которой искомая функция относится. Тем не менее, следует отметить, что принцип разбиения функций по группам или категориям в некоторых случаях представляется достаточно спорным. Поэтому полезно обращать внимание на два поля, следующих сразу за списками категорий функций и самих функций. В первом поле отображается название выбранной функции (с указанием ее аргументов), в следующем за ним — краткая, но, как правило, исчерпывающая справка по данной функции. Она позволяет сделать вывод относительно возможности (или невозможности) использования функции в проводимом исследовании. Однако увлекаться справочными возможностями диалогового окна **Insert Function** не следует. Если есть хотя бы малейшее сомнение в адекватности выбранной функции решаемой задаче, то разумнее будет затратить дополнительное время на ознакомление с полной справочной информацией по функции. Дело в том, что очень часто встроенные функции Mathcad, по сравнению с их математическими двойниками, имеют некоторые особенности определения (например, ограниченная область определения, обработка комплексных аргументов и т. п.), незнание которых может привести к весьма печальным последствиям. Это замечание, кстати, относится не только к Mathcad, но и к большинству аналогичных математических пакетов. Далее предлагается краткий обзор основных функций Mathcad. Для удобства эти функции разбиты на группы, исходя из возможной области их применения, поэтому разбиение не всегда совпадает с тем, что использовано в Mathcad в диалоговом окне **Insert Function**.

Стандартные математические функции

Основная нагрузка, если можно так выразиться, в процессе решения практически любой задачи ложится на ядро из стандартных математических функций, которые приведены и кратко описаны в табл. 2.5.

Таблица 2.5. *Основные математические функции*

Функция	Описание
`acos(z)`	Арккосинус от `z`
`acosh(z)`	Гиперболический арккосинус от `z`
`acot(z)`	Арккотангенс от `z`
`acoth(z)`	Гиперболический арккотангенс от `z`
`acsc(z)`	Арккосеканс от `z`
`acsch(z)`	Гиперболический арккосеканс от `z`

Таблица 2.5 *(окончание)*

Функция	Описание
`asec(z)`	Арксеканс от `z`
`asech(z)`	Гиперболический арксеканс от `z`
`asin(z)`	Арксинус от `z`
`asinh(z)`	Гиперболический арксинус от `z`
`atan(z)`	Арктангенс от `z`
`atan2(x,y)`	Функцией в качестве результата возвращается угол (в радианах) между осью `Ox` и прямой, проходящей через начало координат и точку с координатами `x` и `y`. Другими словами, $\mathrm{atan2}(x, y) = \mathrm{atan}(y/x)$
`atanh(z)`	Гиперболический арктангенс от `z`
`cos(z)`	Косинус от `z`
`cosh(z)`	Гиперболический косинус от `z`
`cot(z)`	Котангенс от `z`
`coth(z)`	Гиперболический котангенс от `z`
`csc(z)`	Косеканс от `z`
`csch(z)`	Гиперболический косеканс от `z`
`exp(z)`	Показательная функция ($e \approx 2.72$ в степени `z`)
`Γ(z)`	Гамма-функция Эйлера от `z`. Вычисляется как $\Gamma(z) = \int_0^\infty t^{z-1} \exp(-t) dt$
`ln(z)`	Логарифм натуральный от `z`
`lnΓ(z)`	Логарифм натуральный от гамма-функции Эйлера с аргументом `z`. Символ Γ можно ввести так: сначала ввести большую литеру G (комбинация клавиш <Shift>+<G>), после чего нажать комбинацию клавиш <Ctrl>+<G>
`log(z,b)`	Логарифм от `z` по основанию `b`. Если второй аргумент функции не указан, логарифм вычисляется по основанию `10`
`sec(z)`	Секанс от `z`
`sech(z)`	Секанс от `z`
`sin(z)`	Синус от `z`
`sinc(z)`	Функция, значение которой вычисляется по формуле $\mathrm{sinc}(z) = \sin(z)/z$
`sinh(z)`	Синус гиперболический от `z`
`tan(z)`	Тангенс от `z`
`tanh(z)`	Тангенс гиперболический от `z`

Как видим, приведенный перечень содержит в основном прямые и обратные тригонометрические и гиперболические функции.

Пример 2.18. Математические функции

В документе на рис. 2.43 представлены примеры вызова некоторых функций.

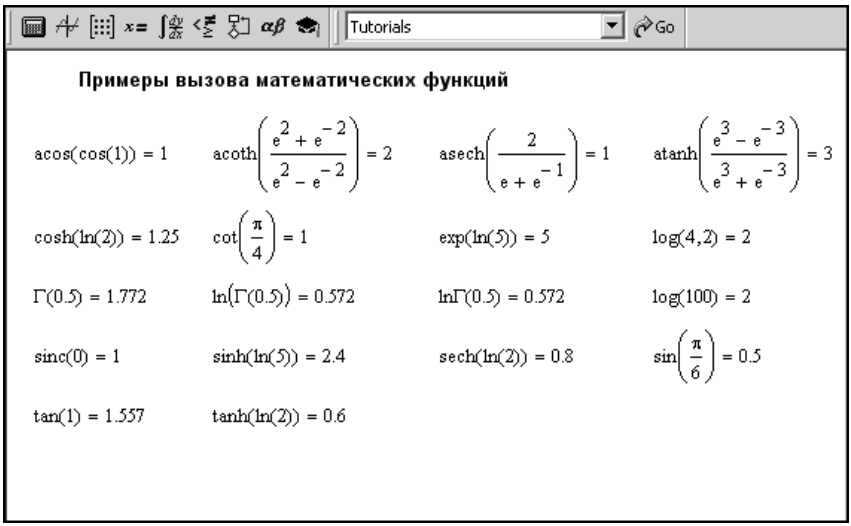


Рис. 2.43. Примеры вызова математических функций

В рабочих документах Mathcad допустимы не только действительные числа, но и комплексные. Они могут быть, в частности, аргументом (или аргументами) для большинства стандартных математических функций. Кроме того, в Mathcad есть функции, предназначенные для работы именно с комплексными числами, причем в некоторых случаях возвращаемое функцией значение, в силу ее определения в Mathcad, несколько отличается от того, которое можно было бы ожидать исходя из общематематических представлений. Например, функцией `signum()` в качестве значения возвращается в общем случае комплексное число, равное отношению числа-аргумента функции к модулю этого аргумента. Если аргумент у функции нулевой, функцией возвращается значение `1`. Таким образом, для действительного аргумента функция возвращает его знак. В некотором родстве к ней (если так можно сказать о функции) находится функция `csgn()`. Однако возвращаемое этой функцией значение всегда действительно. Алгоритм его вычисления довольно замысловат. Функцией возвращается знак действительной части аргумента. Если число мнимое (т. е. действительная часть равна нулю), функция возвращает знак мнимой части аргумента. Для нулевого аргумента значение функции равно нулю.

Поэтому для действительных аргументов различие функций `signum()` и `csgn()` состоит лишь в том, что первая для нуля возвращает значение 1, а вторая — 0.

Функции вычисления действительной и мнимой частей комплексного числа в Mathcad имеют общепринятые математические названия: `Re()` для вычисления действительной части и `Im()` — для мнимой. Наконец, аргумент комплексного числа (это угол между действительной осью и вектором, направленным из начала системы координат к точке на комплексной плоскости) можно определить с помощью функции `arg()`. Здесь уместно напомнить, что для задания комплексного числа достаточно знать два параметра. Это могут быть действительная и мнимая части комплексного числа либо его модуль и аргумент. Модуль комплексного числа можно вычислить, заключив число в прямые скобки.

Пример 2.19. Работа с комплексными числами

Пример функций с комплексными аргументами приведен в документе на рис. 2.44.

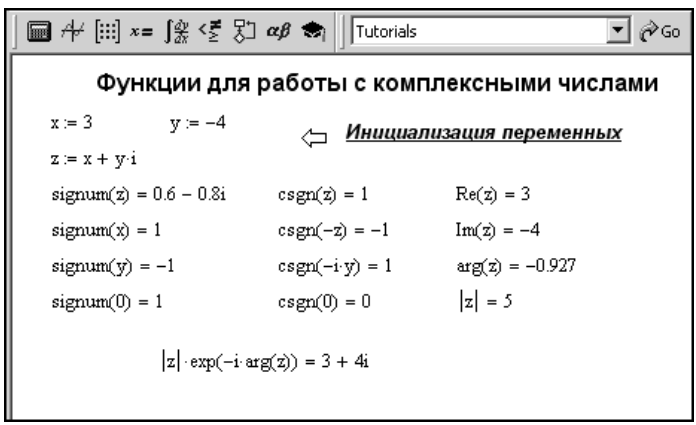


Рис. 2.44. Пример вызова функций для работы с комплексными числами

Описанные функции встречаются достаточно часто. К сожалению, приходится прибегать к помощи и более серьезного математического аппарата. Важным является класс так называемых специальных функций. В данном случае к этому классу отнесем те математические функции Mathcad, которые не принято считать элементарными.

Специальные функции

В первую очередь следует обратить внимание на существование в Mathcad функций Бесселя. Для того чтобы увидеть список доступных функций Бесселя и синтаксис их вызова, необходимо в списке **Function Category** диалогового окна **Insert Function** выбрать пункт **Bessel**. В результате появляется доста-

точно длинный список встроенных функций, названия которых далеко не всегда указывают на принадлежность к данному классу функций. Причем многие из них специфичны именно для Mathcad. Это масштабированные функции, которые получаются из базовых умножением на экспоненциальную функцию. В табл. 2.6 приведены краткие сведения об этих функциях.

Таблица 2.6. *Функции Бесселя в Mathcad*

Функция	**Описание**
`Ai(x)`	Функция Эйри первого рода от аргумента `x`
`Ai.sc(x)`	Масштабированная на $\exp(\lvert \mathrm{Re}(2/3x^{3/2}) \rvert)$ функция Эйри первого рода от аргумента `x`
`bei(m,x)`	Мнимая функция Кельвина порядка `m` от аргумента `x`
`ber(m,x)`	Действительная функция Кельвина порядка `m` от аргумента `x`
`Bi(x)`	Функция Эйри второго рода от аргумента `x`
`Bi.sc(x)`	Масштабированная на $\exp(\lvert \mathrm{Re}(2/3x^{3/2}) \rvert)$ функция Эйри второго рода от аргумента `x`
`H1(m,x)`	Функция Ханкеля первого рода (альтернативное название — функция Бесселя третьего рода)
`H1.sc(m,x)`	Масштабированная функция Ханкеля порядка `m` от аргумента `x`. Масштабный множитель перехода от масштабированной функции к обычной — $\exp(-(3-2m)xi)$
`H2(m,x)`	Функция Ханкеля второго рода порядка `m` от аргумента `x`
`H2.sc(m,x)`	Масштабированная функция Ханкеля второго рода, получается из обычной функции умножением на $\exp(-(3-2m)xi)$
`I0(x)`	Модифицированная функция Бесселя первого рода нулевого порядка от аргумента `x`
`I0.sc(x)`	Масштабированная модифицированная функция Бесселя первого рода нулевого порядка от аргумента `x`. Может быть получена из обычной функции умножением на $\exp(-\lvert \mathrm{Re}(x) \rvert)$
`I1(x)`	Модифицированная функция Бесселя первого рода первого порядка от аргумента `x`
`I1.sc(x)`	Масштабированная на $\exp(-\lvert \mathrm{Re}(x) \rvert)$ модифицированная функция Бесселя первого рода первого порядка от аргумента `x`
`In(m,x)`	Модифицированная функция Бесселя первого рода порядка `m` от аргумента `x`
`In.sc(m,x)`	Масштабированная на $\exp(-\lvert \mathrm{Re}(x) \rvert)$ модифицированная функция Бесселя первого рода порядка `m` от аргумента `x`

Таблица 2.6 *(окончание)*

Функция	Описание
`J0(x)`	Функция Бесселя первого рода нулевого порядка от аргумента `x`
`J0.sc(x)`	Масштабированная на $\exp(-\lvert \mathrm{Im}(x) \rvert)$ функция Бесселя первого рода нулевого порядка от аргумента `x`
`J1(x)`	Функция Бесселя первого рода первого порядка от аргумента `x`
`J1.sc(x)`	Масштабированная на $\exp(-\lvert \mathrm{Im}(x) \rvert)$ функция Бесселя первого рода первого порядка от аргумента x
`Jn(m,x)`	Функция Бесселя первого рода порядка `m` от аргумента `x`
`Jn.sc(m,x)`	Масштабированная на $\exp(-\lvert \mathrm{Im}(x) \rvert)$ функция Бесселя первого рода порядка m от аргумента x
`js(m,x)`	Сферическая функция Бесселя первого рода порядка `m` от аргумента `x`
`K0(x)`	Модифицированная функция Бесселя второго рода нулевого порядка от аргумента `x`
`K0.sc(x)`	Масштабированная модифицированная функция Бесселя второго рода нулевого порядка от аргумента `x`. Может быть получена из обычной функции умножением на $\exp(x)$
`K1(x)`	Модифицированная функция Бесселя второго рода первого порядка от аргумента `x`
`K1.sc(x)`	Масштабированная на $\exp(x)$ модифицированная функция Бесселя второго рода первого порядка от аргумента `x`
`Kn(m,x)`	Модифицированная функция Бесселя второго рода порядка `m` от аргумента `x`
`Kn.sc(m,x)`	Масштабированная на $\exp(x)$ модифицированная функция Бесселя второго рода порядка `m` от аргумента `x`
`Y0(x)`	Функция Бесселя второго рода нулевого порядка от аргумента `x`
`Y0.sc(x)`	Масштабированная на $\exp(-\lvert \mathrm{Im}(x) \rvert)$ функция Бесселя второго рода нулевого порядка от аргумента `x`
`Y1(x)`	Функция Бесселя второго рода первого порядка от аргумента `x`
`Y1.sc(x)`	Масштабированная на $\exp(-\lvert \mathrm{Im}(x) \rvert)$ функция Бесселя второго рода первого порядка от аргумента `x`
`Yn(m,x)`	Функция Бесселя первого рода порядка `m` от аргумента `x`
`Yn.sc(m,x)`	Масштабированная на $\exp(-\lvert \mathrm{Im}(x) \rvert)$ функция Бесселя второго рода порядка `m` от аргумента `x`
`ys(m,x)`	Сферическая функция Бесселя второго рода порядка `m` от аргумента `x`

Названия масштабированных функций отличаются от названий обычных наличием нижнего индекса `sc`. В табл. 2.6 этот индекс отделен от названия функции точкой. Если вводить название масштабированной функции в рабочем документе с помощью клавиатуры, то после названия функции можно ввести точку (точнее, нажать клавишу <.>), в результате чего курсор ввода переходит в положение нижнего индекса у названия функции. В результате можно корректно ввести индекс масштабирования. Именно этим и объясняется способ указания названий масштабированных функций в таблице.

Помимо функций Бесселя, в практически важных задачах часто встречаются ортогональные полиномы, гипергеометрические и ряд других функций, которые также имеются в Mathcad. Большая их часть собрана в категории **Special** в диалоговом окне **Insert Function**. Доступные в Mathcad математические функции из этой категории приведены в табл. 2.7.

Таблица 2.7. *Специальные функции Mathcad*

Функция	**Описание**
`DMS(x)`	Значение угла в радианах
`erf(x)`	Функция ошибок
`erfc(x)`	Дополнительная функция ошибок
`fhyper(a,b,c,x)`	Гипергеометрическая функция Гаусса с параметрами `a,b,c` от аргумента `x`
`Her(n,x)`	Полином Эрмита степени `n` от аргумента `x`
`ibeta(a,x,y)`	Неполная бета-функция от аргументов `x` и `y` с параметром `a`
`Jac(n,a,b,x)`	Полином Якоби степени `n` от аргумента `x` с параметрами `a` и `b`
`Lag(n,x)`	Полином Лагерра степени `n` от аргумента `x`
`Leg(n,x)`	Полином Лежандра степени `n` от аргумента `x`
`mhyper(a,b,x)`	Вырожденная гипергеометрическая функция от аргумента `x` с параметрами `a` и `b`
`Tcheb(n,x)`	Полином Чебышева первого рода степени `n` от аргумента `x`
`Ucheb(n,x)`	Полином Чебышева второго рода степени `n` от аргумента `x`

Как правило, представленного набора функций достаточно для решения любой самой сложной задачи.

Функции округления численных значений

Группа функций Mathcad, предназначенная для преобразования и округления численных значений, приведена в табл. 2.8. Эти функции собраны в категории **Truncation and Round Off** диалогового окна вставки функций **Insert Function**.

Таблица 2.8. *Функции преобразования значений*

Функция	Описание
`Ceil(x,y)`	Функцией в качестве значения возвращается наименьшее кратное числа `y`, не меньшее числа `x`
`ceil(x)`	Функцией возвращается наименьшее целое число, которое больше либо равно `x`
`Floor(x,y)`	Возвращается наибольшее кратное `y`, не превышающее `x`
`floor(x)`	Значением функции является наибольшее возможное целое число, не превышающее `x`
`Round(x,y)`	Значение `x` округляется до ближайшего числа, кратного второму аргументу функции `y`. Это значение и возвращается функцией, при этом значение переменной `x` не меняется
`round(x,n)`	Число `x` округляется до `n` знаков после десятичной точки. Если второй аргумент функции не указан, то округление выполняется до целочисленного значения. При отрицательном значении `n` абсолютное значение второго аргумента определяет количество разрядов округления в целой части числа
`Trunc(x,y)`	Функцией в качестве значения возвращается результат умножения второго аргумента функции (т. е. `y`) на целую часть отношения первого аргумента (т. е. `x`) ко второму
`trunc(x)`	Функцией возвращается целая часть числа `x`

Часто такие функции необходимы в процессе решения задач смешанного типа, в которых, например, из непрерывного диапазона значений выбираются, в силу специфики задачи, только целочисленные.

Пример 2.20. Преобразование значений

В документе на рис. 2.45 представлены простые примеры вызова функций округления значений.

Между функциями Mathcad для округления численных значений существуют очевидные соотношения, а сам набор этих функций хотя и не совсем полон, но, тем не менее, позволяет решать любые возникающие на практике задачи.

Например, чтобы вычислить дробную часть от числа `x`, можно воспользоваться командой `x-round(x)`. Кроме этого, представленные в табл. 2.8 функции имеют ряд интересных особенностей, непосредственно вытекающих из их определения. В частности, стоит обратить внимание, что если первый аргумент функции `Trunc()` меньше второго по абсолютной величине, то ее значение будет равно нулю.

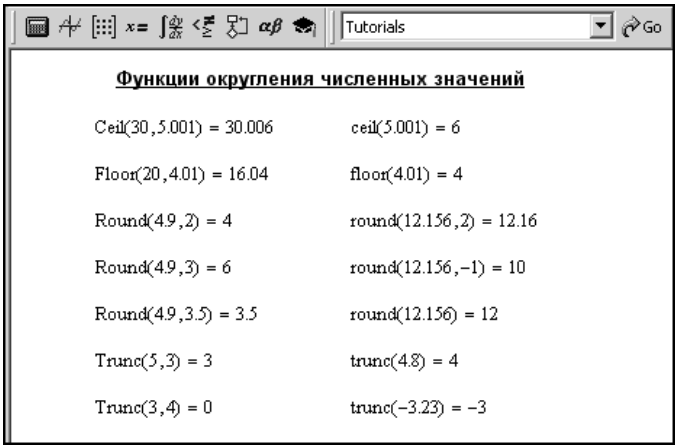


Рис. 2.45. Функции округления численных значений

Пример 2.21. Функции округления

Различия между функциями на первый вгляд могут быть достаточно незначительными. В качестве примера можно привести функции `trunc()` и `floor()`. Для положительных аргументов результат вызова обеих функций одинаков, для отрицательных — значения могут отличаться на единицу (рис. 2.46).

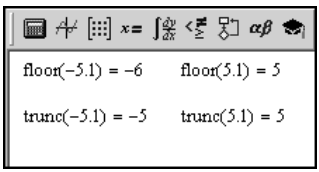


Рис. 2.46. Различия в функциях `trunc()` и `floor()`

Функция `round()`, если ее вызывать только с одним аргументом, в зависимости от значения аргумента, округляет его по правилам, отличным от функций `trunc()` и `floor()`. В данном случае число округляется к ближайшему целому. Документ на рис. 2.47 позволяет составить представление о различии в этих функциях.

Выбор той или иной функции зависит, безусловно, от решаемой задачи.

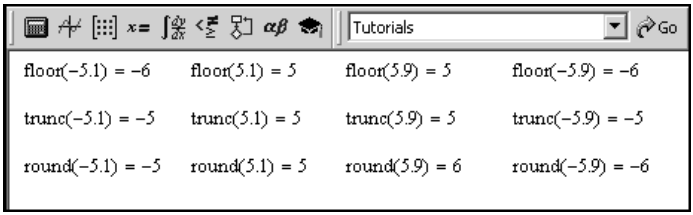


Рис. 2.47. Различные функции округления численных значений

Функции пользователя

Определение функций пользователя осуществляется, как правило, на основе встроенных функций Mathcad. Делается это достаточно просто. В частности, для того, чтобы определить в рабочем документе функцию пользователя, необходимо ввести ее название, в скобках после названия указать аргумент. Если аргументов несколько, они разделяются запятыми. После этого вводится оператор присваивания и выражение, которым непосредственно и задается функция. В этом выражении должны присутствовать аргументы функции, а также могут быть другие переменные, которые следует предварительно инициализировать. В противном случае переменные выделяются цветом, а в рабочем документе появляется сообщение соответствующего характера. Чтобы узнать значение функции для какого-то конкретного аргумента, функция вызывается с этим аргументом и после нее указывается знак равенства.

Пример 2.22. Функции пользователя

В документе на рис. 2.48 приведен пример определения простых функций пользователя с одним и двумя аргументами. Там же представлены команды, с помощью которых вычисляются значения функций для конкретных численных значений аргументов.

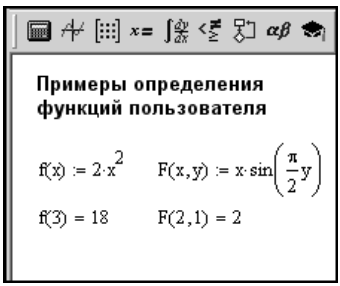


Рис. 2.48. Пример определения функции пользователя

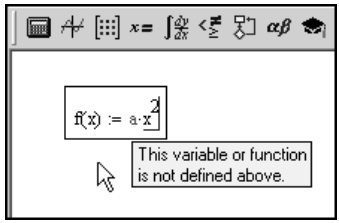


Рис. 2.49. В функции использована неизвестная переменная

На рис. 2.49 проиллюстрирована ситуация, когда в определении функции пользователя встречается переменная, значение которой не определено. В этом случае, как уже отмечалось, появляется соответствующее сообщение.

Выйти из ситуации несложно. Для этого перед определением функции вводят команду присваивания значения переменной (рис. 2.50).

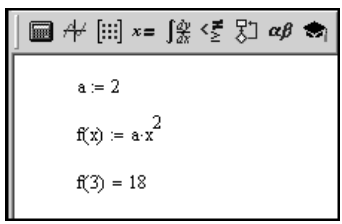


Рис. 2.50. Корректное определение функции

Ранее речь шла о вычислении значения функции при численном значении аргумента (или аргументов).

Пример 2.23. Символьный результат

При попытке вычислить значение функции в символьном виде, т. е. когда аргументами указаны не числа, а символы (названия переменных, которым не присвоены численные значения) с помощью ввода знака равенства после названия функции, результат будет негативным. Однако сделать это можно, если вместо знака равенства ввести оператор вычисления символьного значения (стрелка вправо). Причем в качестве аргументов можно указывать не только отдельные названия, но и целые выражения (рис. 2.51).

Что касается самого оператора вычисления символьного значения, то его можно ввести, нажав комбинацию клавиш <Ctrl>+<.>. Другой способ состоит в выборе этого оператора на панели **Symbolic** (рис. 2.52).

Более подробно вопросы, связанные с символьными вычислениями, обсуждаются в *главе 5*.

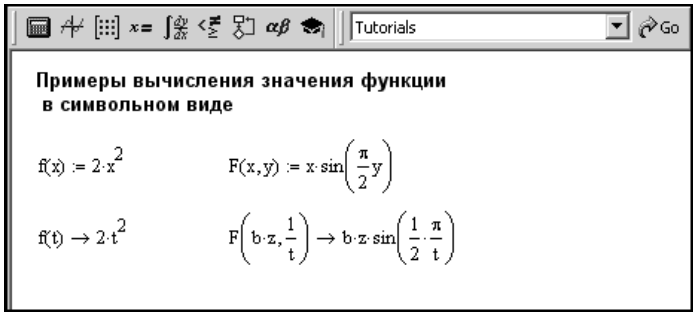


Рис. 2.51. Пример вычисления значений в символьном виде

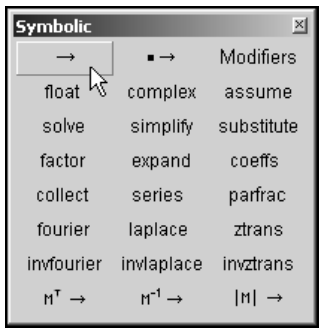


Рис. 2.52. Ввод оператора вычисления символьного значения

Вычисление выражений

Как уже отмечалось, для вычисления численного значения алгебраического выражения в рабочем документе Mathcad, после этого выражения следует поставить знак равенства. Однако не всегда можно ограничиться, например, вычислением суммы или произведения небольшого числа слагаемых. Очень часто выражения имеют достаточно сложную структуру. Помимо этого, приходится вычислять значения выражений или функций, зависящих от какой-то переменной, при разных ее значениях. Не всегда удобно вызывать, например, функцию с разными значениями аргументов — этих значений может быть слишком много или результат нужно получить в компактном виде. Как находить выход из подобных ситуаций, рассказывается в данном разделе.

Вычисление по диапазону значений

В Mathcad переменным в качестве значений можно присваивать не только числа или текстовые строки (об этом рассказывается далее), но и, кроме прочего, диапазоны значений. В данном случае речь идет о последовательности

численных значений, которая присваивается в качестве значения переменной. Если переменную после того, как ей присвоено значение, указать аргументом функции, то функция будет вычислена для каждого элемента последовательности-значения переменной. Таким образом, результат вычисления значения функции для аргумента, имеющего значение-диапазон, является диапазоном значений. Это весьма полезное свойство Mathcad успешно реализуется на практике при решении самых разных задач.

Процедура присваивания диапазона в качестве значения переменной достаточно проста и мало чем отличается от присваивания переменной обычного численного значения. Начинается она с ввода названия переменной, после которого указывается оператор присваивания (комбинация <Shift>+<;>, т. е. следует ввести двоеточие, а знак равенства будет добавлен автоматически). После этого указывается первое значение из диапазона (это может быть либо минимальное, либо максимальное значение). Затем через запятую указывается следующее значение. По этим двум значениям системой определяется шаг изменения переменной. Если первое введенное значение меньше второго, то шаг дискретности положителен, в противном случае — отрицателен. Наконец, после оператора диапазона (это две точки, которые, однако, вводятся нажатием клавиши <;> — если просто ввести с клавиатуры две точки подряд, то они интерпретируются как десятичные и появляется сообщение об ошибке) указывается конечное значение диапазона. Если шаг дискретности положителен, оно должно быть больше первого значения, если отрицателен — меньше. Второе значение можно не указывать. В этом случае шаг дискретности равен по абсолютной величине единице, а его знак выбирается в зависимости от того, какое значение в диапазоне больше — первое или последнее. В любом случае лучше следить за корректностью вводимых значений.

При вводе значений-диапазонов можно применить палитру **Matrix**. На ней, в частности, есть специальная пиктограмма для ввода диапазонных значений (рис. 2.53).

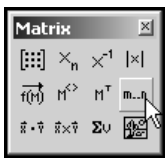


Рис. 2.53. Использование палитры **Matrix** для ввода диапазона значений

После щелчка мышью на пиктограмме в документ вставляется структура из двух заполнителей, между которыми размещен оператор диапазона (две точки) (рис. 2.54).

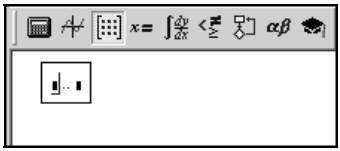


Рис. 2.54. Структурные нули и оператор диапазона

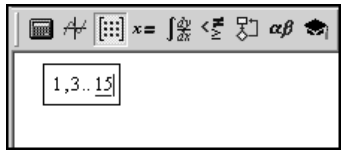


Рис. 2.55. Диапазон в рабочем документе

Вместо структурных заполнителей вводятся границы диапазона и, если необходимо, — второй элемент для определения шага дискретности (рис. 2.55).

Пример 2.24. Диапазон значений

На рис. 2.56 приведен пример присваивания диапазона в качестве значения переменной, которое затем вычисляется (после названия переменной вводится знак равенства).

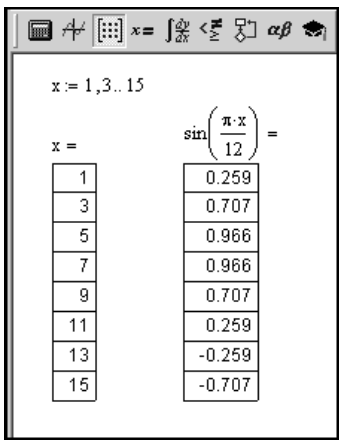


Рис. 2.56. Переменная со значением-диапазоном

Если такую переменную указать в качестве аргумента функции и попытаться проверить значение, то, как уже отмечалось, результат будет вычислен для каждого значения переменной из диапазона и отобразится внизу в виде столбика значений (см. рис. 2.56).

Если функция имеет несколько аргументов, которые принимают значения-диапазоны, то результат вычисляется так: при фиксированных прочих аргументах первый пробегает все значения из диапазона, после этого на шаг дискретности изменяется вторая переменная, а первая снова пробегает значения из своего диапазона. Когда весь диапазон значений пробегает вторая переменная, на шаг дискретности изменяется третья и т. д.

Пример 2.25. Функция двух переменных

В документе на рис. 2.57 приведен пример вычисления функции двух переменных.

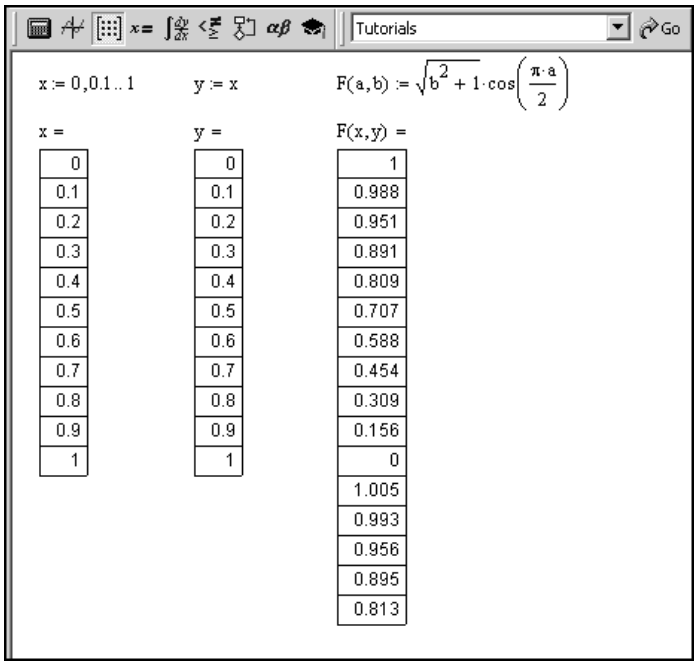


Рис. 2.57. Пример вычисления функции двух переменных

Список со значениями функции может быть достаточно большим, и тогда он отображается не целиком. Чтобы просмотреть все значения, следует выделить этот список в области отображения значений (рис. 2.58).

В случае не полностью отображенного списка значений появляется полоса прокрутки, с помощью которой можно просмотреть список полностью. На рис. 2.59 представлена нижняя часть списка значений функции.

Разумеется, при необходимости размеры отображаемой части списка значений можно изменить: уменьшить или увеличить в зависимости от потребностей. Для этого достаточно навести курсор мыши на рамку вокруг списка, которая появляется при его выделении, в то место, где есть черная узловая точка, и переместить границы в нужном направлении (рис. 2.60).

Аналогичная ситуация имеет место, когда вычисляется значение выражения, в котором есть переменные со значениями-диапазонами.

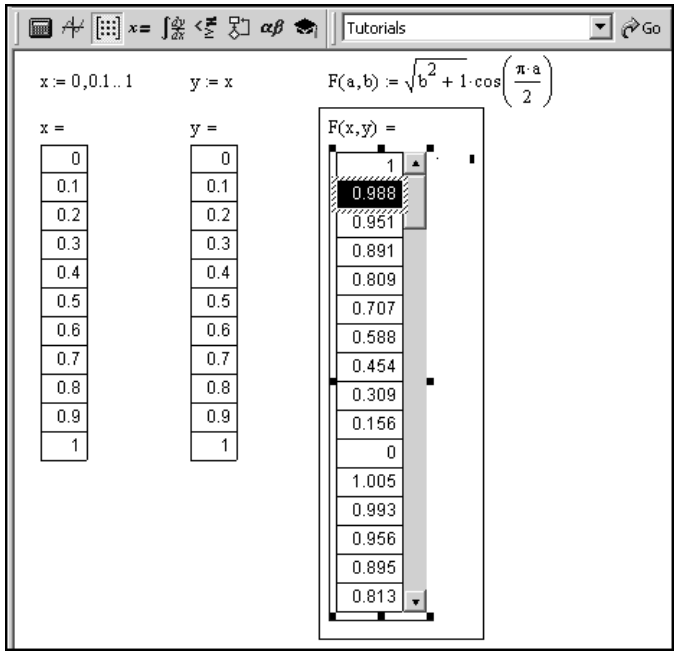


Рис. 2.58. Выделение списка со значениями функции

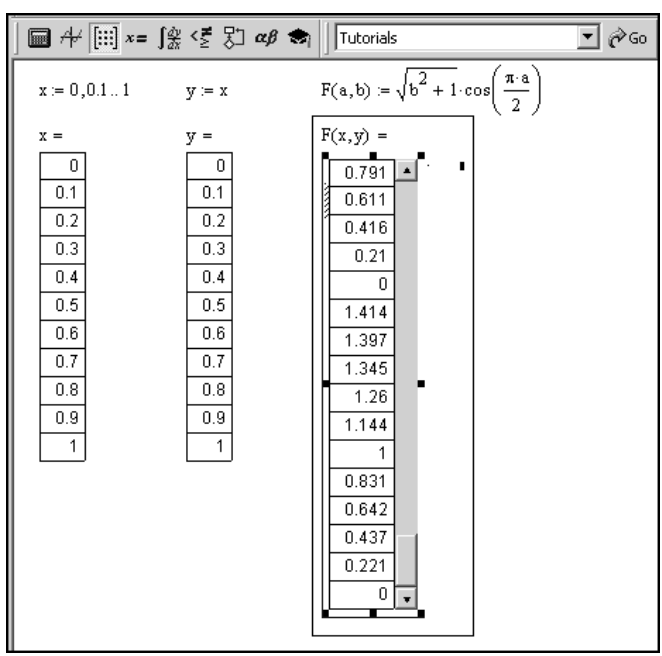


Рис. 2.59. Просмотр списка значений функции

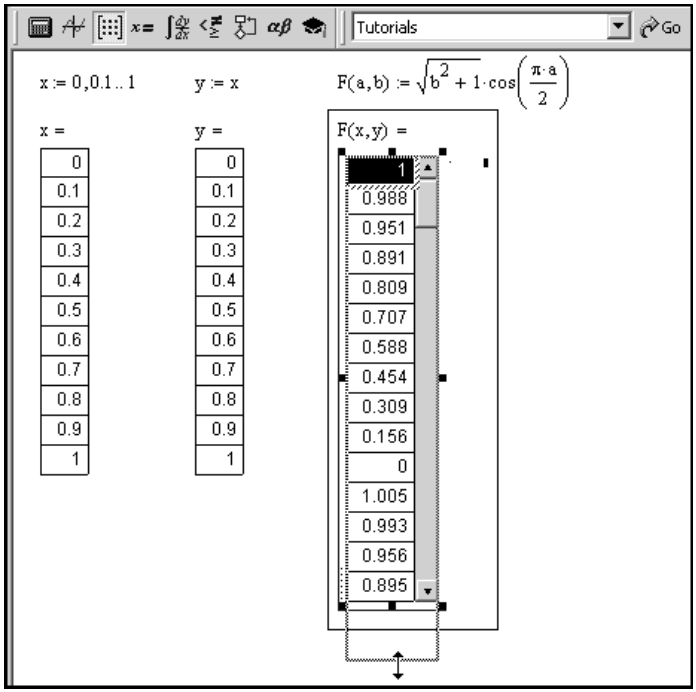


Рис. 2.60. Изменение размеров списка

Вычисление сумм

Оператор сложения удобен, когда слагаемых в выражении не слишком много. В противном случае полезен специальный оператор суммирования, подразумевающий указание индексной переменной суммирования. Слагаемые имеют общий вид, в котором отображена связь с индексной переменной. В зависимости о того, какие значения принимает индексная переменная, существуют два типа оператора вычисления суммы. В первом случае указываются нижняя и верхняя границы диапазона изменения индексной переменной. Шаг дискретности переменной при этом равен единице. Иногда сумму приходится вычислять по значениям индексной переменной, шаг изменения которой отличен от единицы. Тогда можно предварительно индексной переменной в качестве значения присвоить диапазон, после чего воспользоваться альтернативной формой оператора вычисления суммы, в котором переменная суммирования указывается без границ диапазона ее изменения.

Пример 2.26. Вычисление суммы

В документе на рис. 2.61 проиллюстрированы оба подхода.

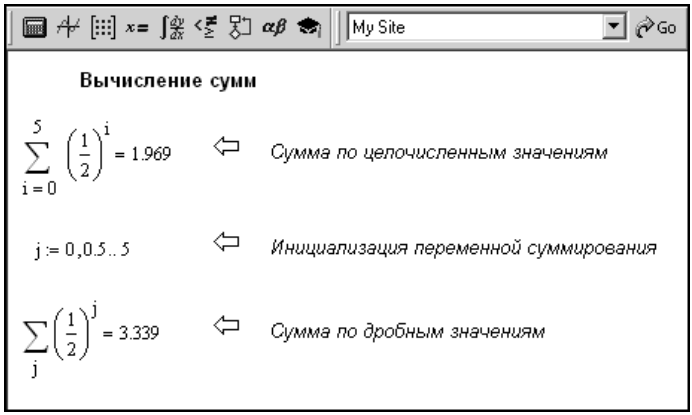


Рис. 2.61. Вычисление сумм различными способами

В первом выражении вычисляется сумма по целочисленным значениям с шагом дискретизации, равным единице. Там употребляется оператор суммирования, в котором, помимо индексной переменной, указываются и границы ее изменения (причем начальное значение может быть меньше конечного). Этот оператор можно ввести, нажав комбинацию клавиш <Ctrl>+<Shift>+<4> либо воспользовавшись палитрой **Calculus** (рис. 2.62).

Если индексная переменная инициализируется со значением-диапазоном, то необходим другой оператор суммирования, который вводится в рабочий документ нажатием комбинации клавиш <Shift>+<4> или выбором пиктограммы на палитре **Calculus** (рис. 2.63).

Рис. 2.62. Вставка оператора суммы с указанием границ диапазона изменения индексной переменной

Рис. 2.63. Вставка оператора суммы без указания границ диапазона изменения индексной переменной

Стоит отметить, что часто задача состоит в вычислении суммы бесконечного числа слагаемых. Если ответ нужно получить в численном виде (т. е. приближенный), разумно ввести в качестве верхней границы достаточно большое число. В противном случае, т. е. для нахождения точного значения ряда из бесконечного числа слагаемых нужно вычислять результат в символьном виде. Об этом речь пойдет в *главе 5*.

Вычисление произведений

Практически все сказанное относительно вычисления сумм большого числа слагаемых с минимальными изменениями относится и к вычислению произведений существенного числа множителей. В частности, в Mathcad есть, помимо стандартного оператора умножения, два оператора вычисления произведений: в одном для индексной переменной границы диапазона изменения указываются, в другом — нет.

Пример 2.27. Вычисление произведений

На рис. 2.64 проиллюстрированы примеры вычисления произведений.

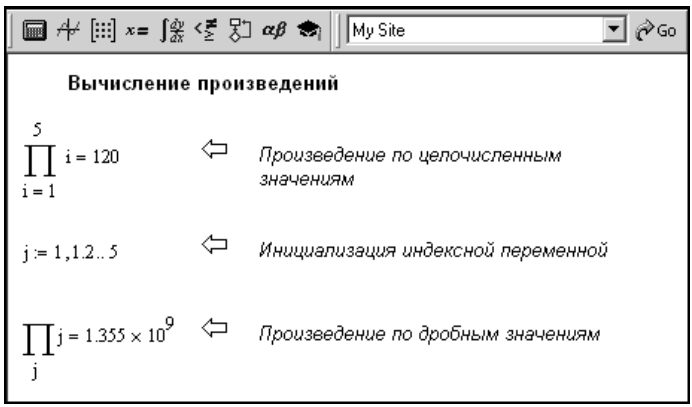


Рис. 2.64. Вычисление произведений различными способами

Оператор вычисления произведения с указанием границ диапазона изменения индексной переменной вводится нажатием комбинации клавиш <Ctrl>+<Shift>+<3>. На рис. 2.65 показано, где на палитре **Calculus** размещена пиктограмма вставки данного оператора.

Оператор вычисления произведения, для которого следует предварительно инициализировать индексную переменную со значением-диапазоном, вводят нажатием комбинации <Shift>+<3> или прибегают к помощи палитры **Calculus** (рис. 2.66).

Что касается описанных ранее операторов (как вычисления суммы, так и произведения), в которых указывается только индексная переменная (но не задаются границы ее изменения), то, строго говоря, предварительно ее можно и не инициализировать. И хотя в численном виде выражение вычислить не удастся, можно попытаться получить символьное значение для суммы или произведения. Однако эти вопросы вынесены в *главу 5*.

Рис. 2.65. Вставка оператора произведения с указанием границ диапазона изменения индексной переменной

Рис. 2.66. Вставка оператора произведения без указания границ диапазона изменения индексной переменной

Использование размерных единиц

Любой серьезный программный пакет ориентирован в первую очередь на решение прикладных задач. Как правило, имеются в виду физические задачи. А, как известно, при этом приходится иметь дело с размерными величинами. Само по себе это проблемой не является. Однако преобразование размерных величин к единой системе размерностей часто становится задачей малоприятной. К счастью, в Mathcad предусмотрена возможность работы с размерными величинами. Подобные операции осуществляются с учетом их размерностей, которые могут быть записаны в разных системах. В дальнейшем переменную, значение которой является величиной размерной, будем называть размерной переменной.

Для инициализации размерной переменной следует ввести ее название, после чего этой переменной присваивается значение. Такая процедура инициализации переменной не отличается от стандартной. После численного значения переменной вводится знак умножения, и далее вместо множителя вставляется символьное выражение размерности. Проще всего это сделать с помощью команды **Insert | Unit** (рис. 2.67).

В результате открывается диалоговое окно **Insert Unit**, представленное на рис. 2.68.

В окне есть два списка: **Dimension** и **Unit**. В **Dimension** представлены категории, по которым сгруппированы размерные величины. Выбрав ту или иную группу, можно существенно сократить список размерных единиц, которые отображаются в списке **Unit**. На рис. 2.69 показано, как можно выбрать размерность времени *минуты*. В этом случае в списке **Dimension** выбирают элемент **Time**, а в **Unit — Minute (min)**.

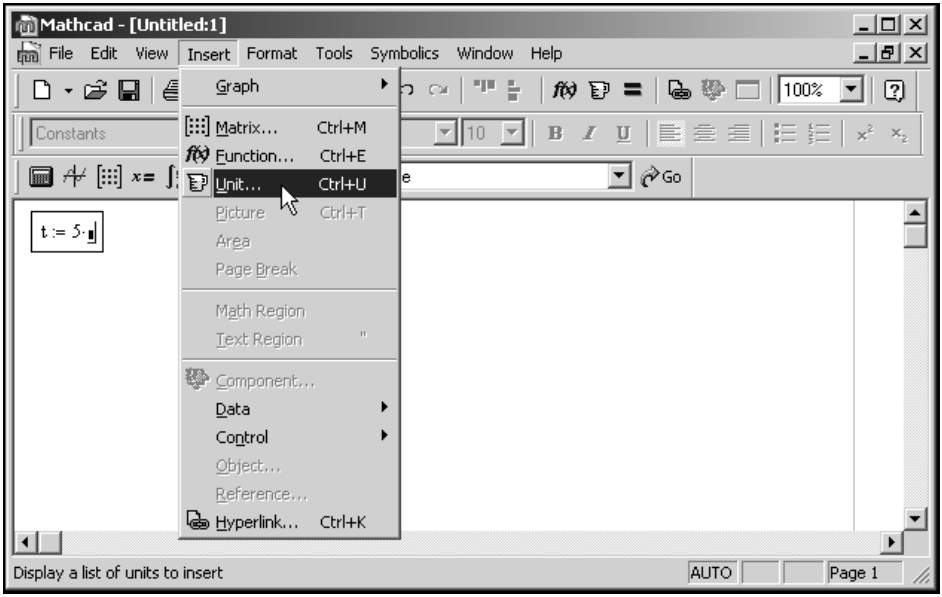


Рис. 2.67. Выбор команды **Insert | Unit**

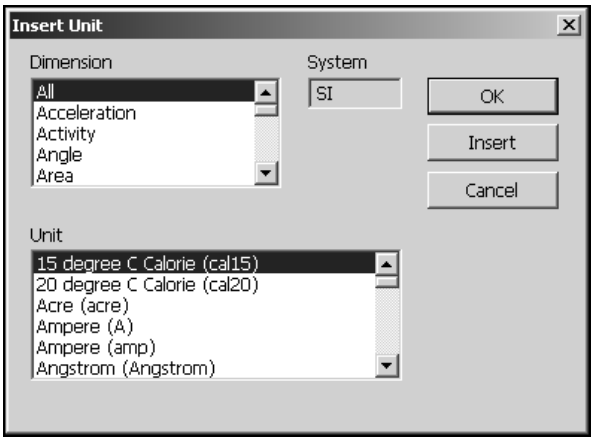


Рис. 2.68. Диалоговое окно **Insert Unit**

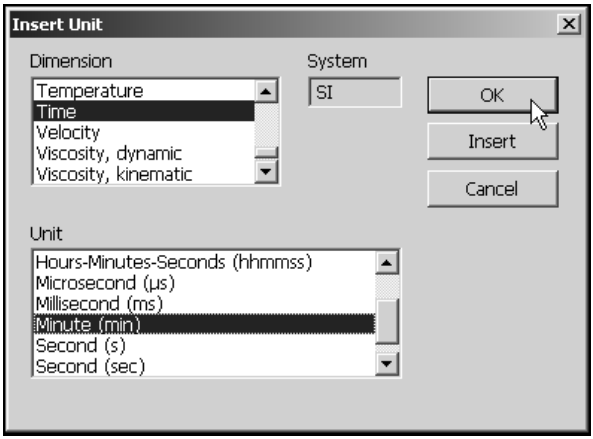


Рис. 2.69. Выбор в качестве размерности минут

Пример 2.28. Размерные величины

В документе на рис. 2.70 представлен результат определения двух размерных величин. Причем одна определена в минутах, а другая — в секундах. Результат их сложения — также величина размерная, причем представлена она в секундах.

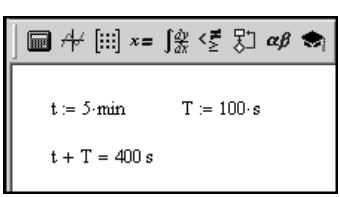


Рис. 2.70. Результат определения размерных величин

Остановимся на основных возможностях по работе с размерными величинами, предоставляемых пользователю приложением Mathcad. Стоит обратить внимание на табл. 2.9, в которой собраны всевозможные типы размерных величин.

Таблица 2.9. *Типы размерных величин*

Категория	Описание
All	Если выбрана эта опция, в списке **Unit** отображаются все доступные размерности
Acceleration	Единицы размерности для ускорения
Activity	Единицы размерности для радиоактивности
Angle	Единицы размерности для углов
Area	Единицы размерности для площади

Таблица 2.9 *(продолжение)*

Категория	Описание
Capacitance	Единицы размерности для электрической емкости
Catalytic Activity	Каталитическая активность
Charge	Единицы размерности для электрического заряда
Conductance	Единицы размерности для электрической проводимости
Current	Единицы размерности для электрического тока
Dose	Доза излучения
Energy	Единицы размерности для энергии
Flow Rate	Единицы потока жидкости
Force	Единицы размерности для силы
Force Density	Объемная плотность силы
Force per Length	Линейная плотность силы
Frequency	Единицы размерности для частоты
Illuminosity	Единицы размерности для освещенности
Inductance	Единицы размерности для индуктивности
Length	Единицы размерности для длины
Luminous Intensity	Единицы размерности для яркости свечения
Magnetic Field Strength	Единицы размерности для силы магнитного поля
Magnetic Flux	Единицы размерности для магнитного потока
Magnetic Flux Density	Единицы размерности для плотности магнитного потока
Mass	Единицы размерности для массы
Permeability	Магнитная проницаемость
Permittivity	Электрическая проницаемость
Potential	Единицы размерности для электрического потенциала
Power	Единицы размерности для мощности
Pressure	Единицы размерности для давления
Resistance	Единицы размерности для электрического сопротивления
Substance	Единицы размерности для количества вещества
Temperature	Единицы размерности для температуры

Таблица 2.9 *(окончание)*

Категория	Описание
Time	Единицы размерности для времени
Velocity	Единицы размерности для скорости
Viscosity, dynamic	Единицы размерности для динамической вязкости
Viscosity, kinematic	Единицы размерности для кинематической вязкости
Volume	Единицы размерности для объема

В зависимости от того, какая выбрана система размерностей, список доступных единиц размерности имеет свои особенности. Что касается выбора системы единиц, то сделать это можно на вкладке **Unit System** в диалоговом окне **Worksheet Options**, выбрав команду **Tools | Worksheet Options** (рис. 2.71).

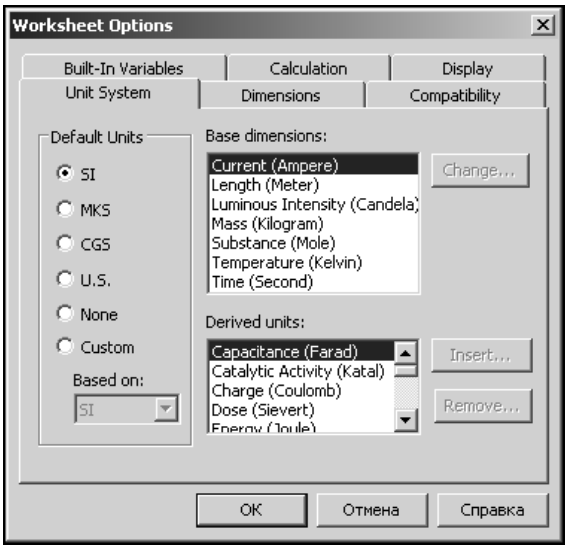


Рис. 2.71. Окно выбора системы единиц

Окно описывалось в *главе 1*. Напомним, что оно содержит группу переключателей **Default Units** из шести элементов. Выбрав один из элементов, можно определить заданную в рабочем документе систему единиц. В частности, в Mathcad предопределены размерные единицы в таких системах: система СИ (элемент **SI (International)**); система МКС (элемент **MKS**) — в этой системе основными элементами длины, массы и времени являются метр, килограмм и секунда соответственно; система СГС (элемент **CGS**) — сокращение

от *сантиметр-грамм-секунда*; американская система единиц (элемент **U.S.**). Наконец, если выбрать элемент **None**, размерные единицы в рабочем документе использоваться не будут, а соответствующие команды по вставке размерностей становятся неактивными.

В табл. 2.10 приведен список основных размерных единиц, которые имеются в Mathcad. Следует отметить, что некоторые размерности доступны только в том случае, если выбрана правильная система единиц. Поэтому здесь же указано, к какой системе (или каким системам) единиц относится та или иная размерность.

Таблица 2.10. *Единицы физических величин*

Обозначение	Описание	Система единиц
`A`	Ампер (стандартное обозначение *А*) — единица силы тока, принятая в СИ	СИ
`a0`	Радиус первой боровской орбиты, равный $5{,}29177\times10^{-11}$ м	СИ, МКС, СГС и американская
`acre`	Акр — площадь, равная примерно 0,4 га или 4047 $м^2$	СИ, МКС, СГС и американская
`amp`	Ампер — сила тока. Эквивалент `A` (см. выше), однако данное обозначение может использоваться в системах единиц, отличных от СИ	СИ, МКС, СГС и американская
`Angstrom`	Ангстрем — единица длины, равная 10^{-10} м	СИ, МКС, СГС и американская
`atm`	Физическая атмосфера (стандартное обозначение *атм*) — единица давления, равная 101,325 кПа	СИ, МКС, СГС и американская
`bar`	Бар — единица давления, равная 10^5 Па	СИ, МКС и американская
`barn`	Барн — единица площади, равная 10^{-28} $м^2$	СИ, МКС, СГС и американская
`bhp`	Паровая лошадиная сила — единица мощности, равная 9809,5 Вт	СИ, МКС, СГС и американская
`bohr`	Радиус первой боровской орбиты, равный $5{,}29177\times10^{-11}$ м	СИ, МКС, СГС и американская
`Bq`	Беккерель (стандартное обозначение *Бк*) — единица радиационной активности	СИ, МКС, СГС и американская
`BTU`	Британская тепловая единица — единица энергии, равная примерно 1055 Дж	СИ, МКС, СГС и американская

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
BTU15	Британская тепловая единица (15-градусная) — единица энергии, равная 1054,8 Дж	СИ, МКС, СГС и американская
c	Скорость света, равная примерно $2{,}998\times10^{8}$ м/с^{2}	СГС
C	Кулон (стандартное обозначение *Кл*) — единица электрического заряда в СИ	СИ
cal	Калория (стандартное обозначение *кал*) — единица энергии, равная 4,1868 Дж	СИ, МКС, СГС и американская
cal15	Калория (15-градусная) — единица энергии, равная 4,1858 Дж	СИ, МКС, СГС и американская
cal20	Калория (20-градусная) — единица энергии, равная 4,1819 Дж	СИ, МКС, СГС и американская
CBTU	Британская тепловая единица (Канада) — единица энергии, равная примерно 1054,6 Дж	СИ, МКС, СГС и американская
cd	Кандела (стандартное обозначение *кд*) — единица силы света, то же, что и одна свеча (1 *св*)	СИ, МКС, СГС и американская
cm	Сантиметр (стандартное обозначение *см*) — единица длины, равная 0,01 м	СИ, МКС, СГС и американская
coul	Кулон (стандартное обозначение *Кл*) — единица электрического заряда, то же самое, что и C, но обозначение может использоваться не только в системе единиц СИ	СИ, МКС, СГС и американская
cubit	Локоть — единица длины, равная примерно 45,7 см	СИ, МКС, СГС и американская
day	Сутки — интервал времени, состоящий из 24 ч или 86400 с	СИ, МКС, СГС и американская
dcal	Диетическая калория — единица энергии, равная 4185,5 Дж	СИ, МКС, СГС и американская
deg	Градус — единица измерения угла, равная 0,0174533 радиан	СИ, МКС, СГС и американская
dyne	Дина (стандартное обозначение *дин*) — единица силы, равная 10 мкН	СИ, МКС, СГС и американская
ε0	Электрическая постоянная, равная $8{,}85418\times10^{-12}$ Ф/м	СИ, МКС, СГС и американская
ehp	Лошадиная сила (электрическая) — единица мощности, равная 746 Вт	СИ, МКС, СГС и американская

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
`erg`	Эрг (стандартное обозначение *эрг*) — единица мощности, равная 10^{-7} Дж	СИ, МКС, СГС и американская
`esu`	Статкулон — единица электрического заряда в СГС: 1 Кл равен 2997924580 статкулон	СГС
`F`	Фарад (стандартное обозначение *Ф*) — единица электрической емкости в СИ	СИ
`farad`	Фарад — единица электрической емкости. Обозначение может использоваться в отличных от СИ системах единиц	СИ, МКС, СГС и американская
`fl_oz`	Жидкая унция — единица объема, равная 29,574 $см^3$	СИ, МКС, СГС и американская
`fortnight`	Промежуток времени в две недели	СИ, МКС, СГС и американская
`ft`	Фут — единица длины, равная 12 дюймам или 0,3048 м	СИ, МКС, СГС и американская
`furlong`	Восьмая часть мили, равная 201 м	СИ, МКС, СГС и американская
`G`	Гаусс, равен 10^{-4} Тл	СИ, МКС и американская
`g`	Ускорение свободного падения относительно ускорения свободного падения на поверхности Земли. Таким образом, ускорение в 1g приближенно равно 9,8 $м/с^2$	СИ, МКС, СГС и американская
`gal`	Галлон (американский) — единица объема, равная примерно 3,785 $дм^3$	СИ, МКС, СГС и американская
`galUK`	Британский галлон — единица объема, равная примерно 4,546 $дм^3$	СИ, МКС, СГС и американская
`gauss`	Гаусс (стандартное обозначение *Гс*) — единица плотности магнитного потока, равная 0,0001 Тл	СИ, МКС и американская
`GHz`	Гигагерц (стандартное обозначение *ГГц*) — единица частоты, равная 10^9 Гц	СИ, МКС, СГС и американская
`gm`	Грамм (стандартное обозначение *гм*) — единица массы, равная 0,001 кг	СИ, МКС, СГС и американская
`gpm`	Единица потока жидкости, равная одному галлону в минуту	СИ, МКС, СГС и американская
`Gy`	Грей — единица дозы излучения	СИ, МКС, СГС и американская

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
H	Генри (стандартное обозначение *Гн*) — единица индуктивности в СИ	СИ
hectare	Гектар (стандартное обозначение *га*) — единица площади, равная 10000 м^2	СИ, МКС, СГС и американская
henry	Генри — единица индуктивности. Это обозначение можно использовать при работе с разными системами единиц	СИ, МКС, СГС и американская
hhp	Лошадиная сила (вода) — единица мощности, равная 746,043 Вт	СИ, МКС, СГС и американская
hp	Лошадиная сила (стандартное обозначение *л.с.*) — единица мощности, равная в Mathcad 745,699871582 Вт	СИ, МКС, СГС и американская
hpUK	Британская лошадиная сила — единица мощности, равная 745,7 Вт	СИ, МКС, СГС и американская
hr	Час (стандартное обозначение *ч*) — интервал времени, равный 3600 с	СИ, МКС, СГС и американская
Hz	Герц (стандартное обозначение *Гц*) — единица частоты в СИ	СИ, МКС, СГС и американская
Hza	Герц (угловая частота) — единица угловой частоты, равная примерно 6,283185307 Гц (это частота, умноженная на 2π)	СИ, МКС, СГС и американская
IBTU	Стандартная британская тепловая единица — единица энергии, равная 1054,5 Дж	СИ, МКС, СГС и американская
in	Дюйм — единица длины, равная 0,0254 м	СИ, МКС, СГС и американская
in_Hg	Дюймов ртутного столба — единица давления, равная примерно 3386,38 Па	СИ, МКС, СГС и американская
J	Джоуль (стандартное обозначение *Дж*) — единица энергии в СИ	СИ
joule	Джоуль — единица энергии. Обозначение можно использовать в различных системах единиц	СИ, МКС, СГС и американская
K	Кельвин — градус термодинамической шкалы Кельвина	СИ, МКС, СГС и американская
KΩ	Килоом (стандартное обозначение *кОм*) — единица сопротивления, равная 1000 Ом	СИ, МКС, СГС и американская

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
kΩ	Килоом, равен 1000 Ом	СИ, МКС, СГС и американская
kA	Килоампер (стандартное обозначение *кА*) — единица силы тока, равная 1000 А	СИ, МКС, СГС и американская
KA	Килоампер — единица силы тока, равная 1000 А. Обозначение используется только в СИ	СИ
katal	Единица скорости каталитической реакции	СИ, МКС, СГС и американская
kcal	Килокалория (стандартное обозначение *ккал*) — единица энергии, равная 1000 кал	СИ, МКС, СГС и американская
kg	Килограмм (стандартное обозначение *кг*) — единица массы	СИ, МКС, СГС и американская
kgf	Килограмм-сила — единица силы, равная 9,80665 Н	СИ, МКС, СГС и американская
kHz	Килогерц (стандартное обозначение *кГц*) — единица частоты, равная 1000 Гц. Обозначение используется в СИ	СИ
KHz	Килогерц — единица частоты, равная 1000 Гц	СИ, МКС, СГС и американская
kip	Единица силы, эквивалентная весу тела массой в 1000 фунтов (453,59 кг)	СИ, МКС, СГС и американская
km	Километр (стандартное обозначение *км*) — единица длины, равная 1000 м	СИ, МКС, СГС и американская
kN	Килоньютон, равен 1000 Н	СИ, МКС, СГС и американская
knot	Морской узел — единица скорости, равная 1,87 км/ч	СИ, МКС и американская
kph	Километры в час (стандартное обозначение *км/ч*) — единица измерения скорости	СИ, МКС, СГС и американская
kV	Киловольт (стандартное обозначение *кВ*) — единица измерения электрического потенциала, равная 1000 В	СИ, МКС, СГС и американская
ksi	Единица давления в расчете 1000 фунтов на квадратный дюйм	СИ, МКС, СГС и американская
KV	Киловольт — единица измерения электрического потенциала, равная 1000 В (обозначение используется в СИ)	СИ

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
kW	Киловатт (стандартное обозначение *кВт*) — единица измерения мощности, равная 1000 Вт (используется в СИ)	СИ
L	Литр (стандартное обозначение *л*) — единица объема, 1000 л составляют 1 $м^3$. Обозначение используется в СИ	СИ
lb	Фунт (торговый британский) — единица массы, равная 0,45359237 кг	СИ, МКС, СГС и американская
lbf	Фунт-сила — единица силы, равная 4,448221615 Н	СИ, МКС, СГС и американская
liter	Литр — единица объема, 1000 л составляют 1 $м^3$. Обозначение используется в разных системах единиц	СИ, МКС, СГС и американская
lm	Люмен (стандартное обозначение *лм*) — единица светового потока	СИ, МКС, СГС и американская
lx	Люкс (стандартное обозначение *лк*) — единица освещенности	СИ, МКС, СГС и американская
m	Метр (стандартное обозначение *м*) — единица длины	СИ, МКС, СГС и американская
mBTU	Средняя британская тепловая единица — единица энергии, равная 1055,87 Дж	СИ, МКС, СГС и американская
mcal	Средняя калория — единица энергии, равная 4,19002 Дж	СИ, МКС, СГС и американская
MΩ	Мегаом (стандартное обозначение *МОм*) — единица сопротивления, равная 10^6 Ом	СИ, МКС, СГС и американская
mA	Миллиампер (стандартное обозначение *мА*) — единица силы тока, равная 10^{-3} А	СИ, МКС, СГС и американская
μA	Микроампер (стандартное обозначение *мкА*) — единица силы тока, равная 10^{-6} А	СИ, МКС, СГС и американская
mg	Миллиграмм (стандартное обозначение *мгм*) — единица массы, равная 10^{-3} г	СИ, МКС, СГС и американская
mH	Миллигерц (стандартное обозначение *мГц*) — единица частоты, равная 10^{-3} Гц	СИ, МКС, СГС и американская
mho	Мо — единица проводимости, равная 1 См	СИ, МКС, СГС и американская
mhp	Метрическая лошадиная сила (стандартное обозначение *л.с.*) — единица мощности, равная 735,499 Вт	СИ, МКС, СГС и американская

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
`MHz`	Мегагерц (стандартное обозначение *МГц*) — единица частоты, равная 10^6 Гц	СИ, МКС, СГС и американская
`mi`	Миля — единица длины, равная 1760 ярдам или 1609,344 м	СИ, МКС, СГС и американская
`micron`	Микрон — единица длины, равная 10^{-6} м	СИ, МКС, СГС и американская
`mil`	Мера длины, равная одной тысячной дюйма	СИ, МКС, СГС и американская
`min`	Минута — единица времени, равная 60 с	СИ, МКС, СГС и американская
`mL`	Миллилитр (стандартное обозначение *мл*) — единица объема, 1000 мл составляют 1 л	СИ, МКС, СГС и американская
`mm`	Миллиметр (стандартное обозначение *мм*) — единица длины, равная 0,001 м	СИ, МКС, СГС и американская
`mol`	Моль — единица количества вещества	СИ, МКС, СГС и американская
`mole`	Моль — единица количества вещества	СИ, МКС, СГС и американская
`MPa`	Мегапаскаль — единица давления, равная 10^6 Па	СИ, МКС, СГС и американская
`mph`	Миль в час — единица измерения скорости, равная 0,44704 м/с	СИ, МКС, СГС и американская
`ms`	Миллисекунда — одна тысячная секунды	СИ, МКС, СГС и американская
`mV`	Милливольт (стандартное обозначение *мВ*) — единица электрического потенциала, равная 0,001 В	СИ, МКС, СГС и американская
`µ0`	Магнитная постоянная, равная $1{,}256637\times10^{-6}$ Гн/м	СИ, МКС, СГС и американская
`µm`	Микрометр — единица длины, равная 10^{-6} м	СИ, МКС, СГС и американская
`µF`	Микрофарад (стандартное обозначение *мкФ*) — единица электрической емкости, равная 10^{-6} Ф	СИ, МКС, СГС и американская
`µH`	Микроньютон — единица силы, равная 10^{-6} Н	СИ, МКС, СГС и американская

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
µs	Микросекунда — одна миллионная секунды	СИ, МКС, СГС и американская
N	Ньютон (стандартное обозначение *Н*) — единица силы в системе СИ	СИ
newton	Ньютон — единица силы, которая может использоваться и в отличных от СИ системах единиц	СИ, МКС, СГС и американская
nm	Нанометр — единица длины, равная 10^{-9} м	СИ, МКС, СГС и американская
nmi	Морская миля, равная 1853,6 м	СИ, МКС, СГС и американская
nF	Нанофарад (стандартное обозначение *нФ*) — единица электрической емкости, равная 10^{-9} Ф	СИ, МКС, СГС и американская
Oe	Эрстед (стандартное обозначение *Э*) — сила магнитного поля в СИ, равная 79,577471546 А/м	СИ
oersted	Эрстед — единица силы магнитного поля. Обозначение используется в разных системах единиц	СИ, МКС, СГС и американская
ohm	Ом (стандартное обозначение *Ом*) — единица сопротивления	СИ, МКС, СГС и американская
oz	Унция — единица массы, равная 0,028349523 кг	СИ, МКС, СГС и американская
Pa	Паскаль (стандартное обозначение *Па*) — единица давления	СИ, МКС, СГС и американская
pF	Пикофарад (стандартное обозначение *пФ*) — единица электрической емкости, равная 10^{-12} Ф	СИ, МКС, СГС и американская
poise	Пуаз (стандартное обозначение *П*) — единица динамической вязкости, равная 0,1 Па·с	СИ, МКС, СГС и американская
psi	Фунтов на квадратный дюйм — единица давления, равная 6894,757293168 Па	СИ, МКС, СГС и американская
R	Градус Ренкина, приближенно равный 0,555555556 К	СИ, МКС, СГС и американская
rad	Радиан — единица измерения углов	СИ, МКС, СГС и американская
S	Сименс (стандартное обозначение *См*) — единица электрической проводимости в СИ	СИ

Таблица 2.10 *(продолжение)*

Обозначение	Описание	Система единиц
`s`	Секунда (стандартное обозначение *сек*) — единица времени в СИ	СИ
`sec`	Секунда — единица времени. Обозначение используется в разных системах единиц	СИ, МКС, СГС и американская
`siemens`	Сименс — единица электрической проводимости. Обозначение используется в различных системах единиц	СИ, МКС, СГС и американская
`slug`	Слаг — единица массы, равная примерно 14,593 кг	СИ, МКС, СГС и американская
`statamp`	Статампер — единица силы тока в СГС: 1 А равен 2997924580 статампер	СГС
`statcoul`	Статкулон — единица электрического заряда в СГС: 1 Кл равен 2997924580 статкулон	СГС
`statfarad`	Статфарад — единица электрической емкости в СГС: 1 Ф равен примерно 898755224014,74 статфарад	СГС
`stathenry`	Статгенри — единица индуктивности в СГС: 1 Гн примерно равен $1{,}113\times10^{-12}$ статгенри	СГС
`statohm`	Статом — единица сопротивления в СГС: 1 Ом равен примерно $1{,}113\times10^{-12}$ статом	СГС
`statseimens`	Статсименс — единица электрической проводимости в СГС: 1 См равен примерно $8{,}988\times10^{11}$ статсименс	СГС
`stattesla`	Статтесла — единица плотности потока магнитной индукции в СГС: 1 Тл примерно равен $2{,}998\times10^{6}$ статтесла	СГС
`statvolt`	Статвольт — единица электрического потенциала в СГС: 1 В составляет порядка $3{,}336\times10^{-3}$ статвольт	СГС
`statweber`	Статвебер — единица потока магнитной индукции в СГС	СГС
`stokes`	Стокс (стандартное обозначение *Ст*) — единица кинематической вязкости	СИ, МКС, СГС и американская
`sr`	Стерадиан — единица измерения телесного угла	СИ, МКС, СГС и американская
`Sv`	Сиверт — единица дозы излучения	СИ

Таблица 2.10 *(окончание)*

Обозначение	Описание	Система единиц
T	Тесла (стандартное обозначение *Тл*) — единица плотности магнитного потока (магнитная индукция) в СИ	СИ
tBTU	Термохимическая британская тепловая единица — единица энергии, равная 1054,35 Дж	СИ, МКС, СГС и американская
tcal	Термохимическая калория — единица энергии, равная примерно 4,184 Дж	СИ, МКС, СГС и американская
tesla	Тесла — единица магнитной индукции (плотность магнитного потока). Обозначение используется в альтернативных к СИ системах единиц	МКС и американская
ton	Короткая тонна — единица массы, примерно равная 907,18474 кг	СИ, МКС, СГС и американская
tonne	Метрическая тонна (стандартное обозначение *т*) — единица массы, равная 1000 кг	СИ, МКС, СГС и американская
torr	Торр — единица давления, равная примерно 133,322 Па	СИ, МКС, СГС и американская
V	Вольт (стандартное обозначение *В*) — единица напряжения (электрический потенциал) в СИ	СИ
volt	Вольт — единица напряжения. Обозначение используется в различных системах единиц	СИ, МКС, СГС и американская
Ω	Ом (стандартное обозначение *Ом*) — единица сопротивления	СИ, МКС, СГС и американская
W	Ватт (стандартное обозначение *Вт*) — единица мощности в СИ	СИ
watt	Ватт — единица мощности. Данное обозначение используется в различных системах единиц	СИ, МКС, СГС и американская
Wb	Вебер (стандартное обозначение *Вб*) — единица магнитного потока в СИ	СИ
weber	Вебер — единица магнитного потока. Обозначение используется в различных системах единиц	СИ, МКС и американская
yd	Ярд — единица длины, равная 3 футам или 0,9144 м	СИ, МКС, СГС и американская
yr	Год — интервал времени, равный 31556926 с	СИ, МКС, СГС и американская

Как видно из таблицы, некоторые размерности дублируются. В основном речь в этих случаях идет об использовании их в разных системах единиц. Более удобны те названия, что воспринимаются не только в СИ.

В какой бы системе единиц ни осуществлялась работа, результат операций в рабочих документах Mathcad неизменно выражается в фундаментальных единицах. С одной стороны, это удобно, а с другой — иногда приходится представлять размерные величины не через комбинацию фундаментальных единиц, а в ином виде, продиктованном особенностями решаемой задачи.

Пример 2.29. Преобразование размерностей

В качестве примера рассмотрим рис. 2.72, на котором сначала переменной присваивается значение в одну метрическую лошадиную силу, после чего значение этой переменной проверяется. Поскольку для работы выбрана система единиц СИ, то значение переменной выражено в ваттах.

Для того чтобы изменить установленную по умолчанию при отображении результата размерность, следует в соответствующей команде выделить структурный заполнитель, размещенный сразу после указателя размерности величины (рис. 2.73).

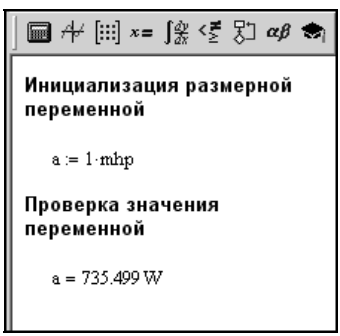


Рис. 2.72. Инициализация размерной переменной и проверка ее значения

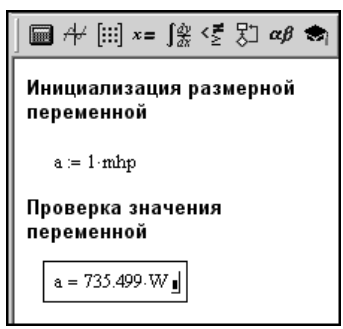


Рис. 2.73. Выделение структурного заполнителя, размещенного за указателем размерности величины

Выделяется только структурный заполнитель, но никак не все выражение — это важно! Далее в месте структурного заполнителя вводится та размерная единица, через которую должен быть выражен результат. В данном случае на рис. 2.74 указаны вольты (т. е. введен символ V).

В результате (после нажатия клавиши <Enter> или щелчка мышью вне области команды) размерный результат выражается через комбинацию размерных единиц, в которой присутствуют вольты (рис. 2.75).

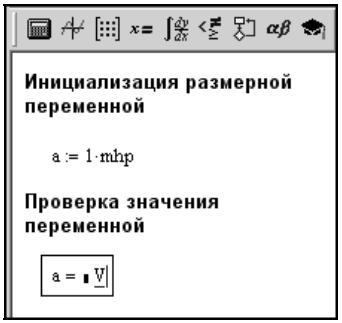


Рис. 2.74. Ввод размерной единицы, через которую должен быть выражен результат

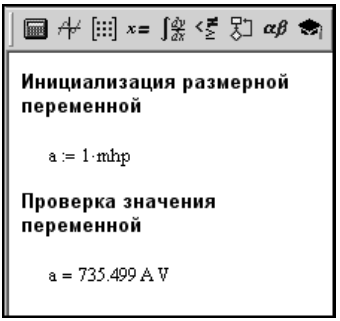


Рис. 2.75. Альтернативный способ отображения значения размерной переменной

В частности, теперь результат (его размерная часть) представлен в виде произведения ампер на вольты, что с физической точки зрения эквивалентно единице мощности (Вт).

Задавать величины в СИ совсем не обязательно. Можно, например, значение переменной, инициализированной со значением в одну метрическую лошадиную силу, выразить в британских лошадиных силах, для чего вместо упоминавшегося ранее структурного заполнителя вводят `hpUK`. Результат представлен на рис. 2.76.

Результат свидетельствует о том, что жители туманного Альбиона достаточно высокого мнения о своих лошадях. Справедливости ради следует отметить, что мнение это разделяют далеко не все.

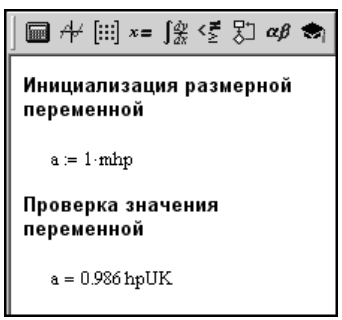


Рис. 2.76. Результат выражен в британских лошадиных силах

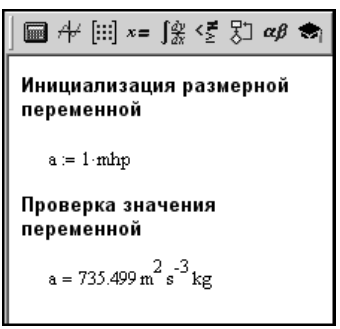


Рис. 2.77. Переменная с размерностью мощности выражена через килограммы, метры и секунды

Еще одна проблема связана с тем, что отображаемая размерность не всегда выглядит эстетично. Особенно это характерно для случаев, когда размерность не принадлежит к одной из фундаментальных для выбранной системы единиц или

преобразована к специальному виду пользователем. На рис. 2.77 переменная с размерностью мощности выражена через килограммы, метры и секунды. Размерные единицы, как несложно убедиться, представлены в виде произведения с соответствующими степенями, в том числе и отрицательными. Чтобы изменить способ отображения размерных единиц, можно дважды щелкнуть на значении переменной, в результате чего открывается диалоговое окно форматирования результата **Result Format**, где следует открыть вкладку **Unit Display** (рис. 2.78).

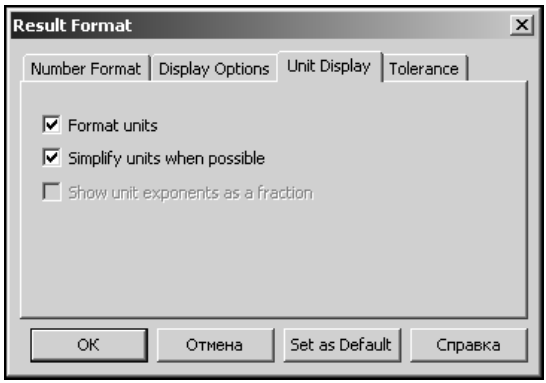


Рис. 2.78. Диалоговое окно **Result Format** открыто на вкладке **Unit Display**

Вкладка содержит опции: **Format units**, **Simplify units when possible** и **Show unit exponents as a fraction**. Если установлен флажок опции **Format units**, выражение для размерной части значения отображается в виде дроби. Флажок опции **Simplify units when possible** устанавливается для автоматического упрощения размерной части значения. На рис. 2.78 установлены флажки обеих опций, в то время как на рис. 2.77 результат показан при отмененных опциях **Format units** и **Simplify units when possible**. Результат после активизации опции **Format units** представлен на рис. 2.79.

Как и ранее, если диалоговое окно **Result Format** открывалось двойным щелчком мыши на результате, то формат применяется только к этому результату. Чтобы форматирование было глобальным, т. е. в рамках всего документа Mathcad, диалоговое окно **Result Format** открывают выбором команды **Format | Result**. Глобальная опция **Show unit exponents as a fraction** может быть установлена только для всего документа в целом и обеспечивает отображение показателей размерностей в виде рациональных дробей.

Важная особенность Mathcad состоит в том, что пользователь может определять собственные размерности. Делается это достаточно просто: вводится название единицы размерности, после чего с помощью оператора присваивания

задается ее размерное значение. Это размерное значение, разумеется, выражается через базовые, встроенные размерные единицы Mathcad. Таким образом, размерность в Mathcad задается в виде обычной размерной переменной.

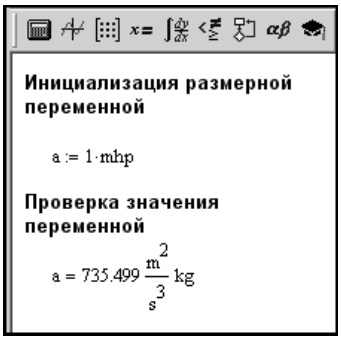


Рис. 2.79. Результат после активизации опции **Format units**

Пример 2.30. Определение размерностей

Пример определения размерностей представлен в документе на рис. 2.80.

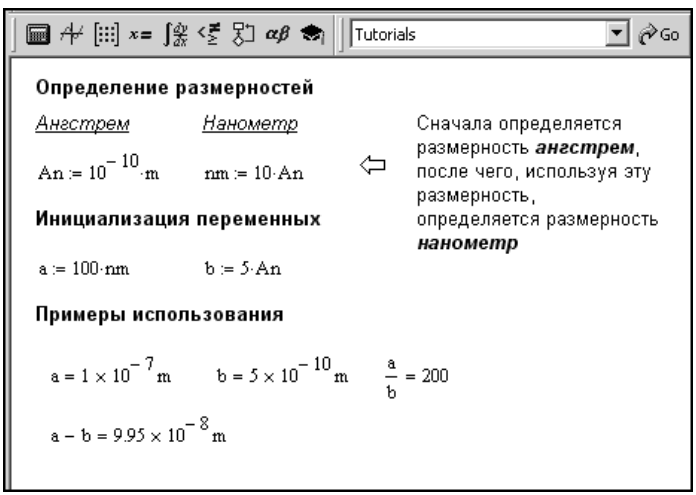


Рис. 2.80. Определение размерностей

В приведенном примере сначала определяется размерность *ангстрем* (переменная `An`). Далее, основываясь на этой размерности, определяется размерность для *нанометров* (один нанометр равен 10 ангстремам). Определенные таким образом размерности можно использовать абсолютно так же, как и встроенные размерности Mathcad.

Пример 2.31. Переопределение размерностей

Описанный способ отображения размерных величин не всегда удобен, причем в силу разных причин. Во-первых, они не всегда и не для всех достаточно информативны. А во-вторых, часто приходится решать задачи, в которых размерности, если так можно выразиться, являются не очень физическими. В Mathcad можно решать физические задачи, но с не меньшим успехом это приложение применяется и для решения экономических задач, в которых численные значения имеют волне конкретную экономическую интерпретацию. Один из способов решения проблемы состоит в переопределении размерных единиц. Для этого вызывают команду **Tools | Worksheet Options**. В открывшемся диалоговом окне **Worksheet Options** следует перейти к вкладке **Dimensions** (рис. 2.81).

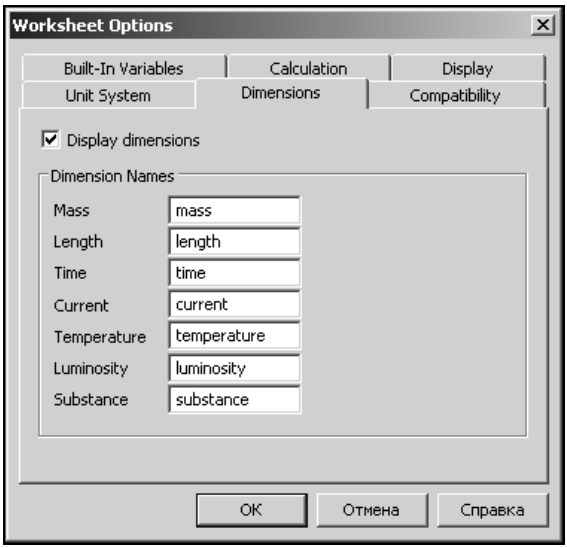


Рис. 2.81. Диалоговое окно **Worksheet Options** открыто на вкладке **Dimensions**

Во вкладке нужно установить флажок опции **Display dimensions**, после чего становится активным набор полей раздела **Dimension Names**, соответствующих основным категориям размерностей, имеющихся в Mathcad. После щелчка на кнопке **OK** в рабочем документе вместо обозначений основных размерностей будут подставлены названия из соответствующих текстовых полей. Причем эти названия можно менять. На рис. 2.82 в поле **Length** введено значение `метр`.

После применения такого формата рассматривавшийся ранее рабочий документ (нижняя его часть, слегка отредактированная) будет выглядеть так, как показано на рис. 2.83.

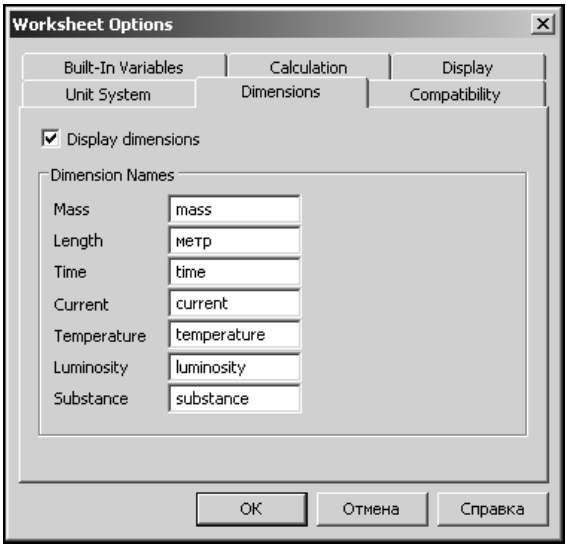


Рис. 2.82. Для длины используется название `метр`

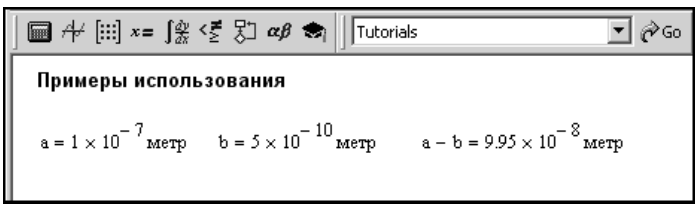


Рис. 2.83. Нижняя часть рабочего документа после применения нового формата

Поскольку в данном случае выбрано кириллическое название, важно, чтобы рабочим шрифтом в документе был шрифт из кириллической группы. Что касается практической пользы от всего этого, то она достаточно очевидна. Предположим, необходимо проводить операции с величинами, которые имеют размерность, не поддерживаемую Mathcad (в первую очередь имеются в виду нефизические размерности). Тогда достаточно переменные определять с какой-то фундаментальной размерностью, а ее отображаемое название переопределить в диалоговом окне **Worksheet Options**.

Матрицы и векторы

В Mathcad предусмотрены широкие возможности для работы с матрицами и векторами. Строго говоря, оба эти объекта реализуются в Mathcad в виде массивов. Каждый массив имеет размерность. В этом смысле матрица является массивом размерностью два, а вектор — один. Задаются матрицы и век-

торы, как и все в Mathcad, достаточно просто, причем предусмотрены различные способы. Массив — набор элементов, каждый из которых может иметь численное значение, текстовое или сам являться массивом. В последнем случае говорят о вложенных массивах. Чтобы получить доступ к какому-то элементу массива, следует указать его индекс (или индексы).

Создание массивов

Для создания массива вызывают команду **Insert | Matrix** (комбинация клавиш <Ctrl>+<M>). Альтернативный вариант — воспользоваться палитрой **Matrix** (рис. 2.84).

Однако какой бы метод ни использовался, в результате откроется диалоговое окно **Insert Matrix**, представленное на рис. 2.85.

Рис. 2.84. Создание массива с помощью палитры **Matrix**

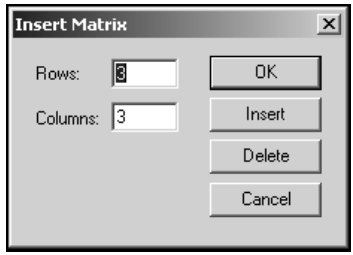


Рис. 2.85. Диалоговое окно **Insert Matrix**

В этом диалоговом окне указывается число строк и столбцов в создаваемом массиве (поля **Rows** и **Columns**). Таким образом, чтобы создать вектор-строку, следует указать в поле **Rows** значение 1, для создания вектора-столбца единицу указывают в поле **Columns**. Если значения в обоих полях отличны от единицы, создаваемый массив с математической точки зрения будет матрицей.

После вызова диалогового окна вставки матрицы **Insert Matrix** и определения размеров массива, в рабочем документе появляется объект, структурно соответствующий создаваемому массиву. Однако вместо элементов в этом объекте отображаются структурные заполнители. Вместо них и следует ввести значения элементов массива. На рис. 2.86 переменной `A` в качестве значения присваивается матрица размерами 3×3.

Пользователю предстоит в явном виде определить все девять элементов данной матрицы.

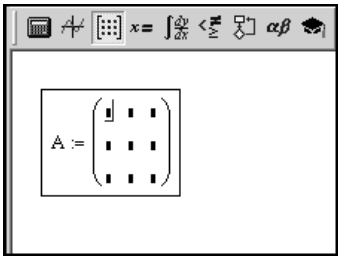


Рис. 2.86. Инициализация матрицы

Пример 2.32. Векторы и матрицы

На рис. 2.87 приведен конечный результат. Там же представлены переменные B и C, которым в качестве значений присвоены вектор-строка и вектор-столбец соответственно.

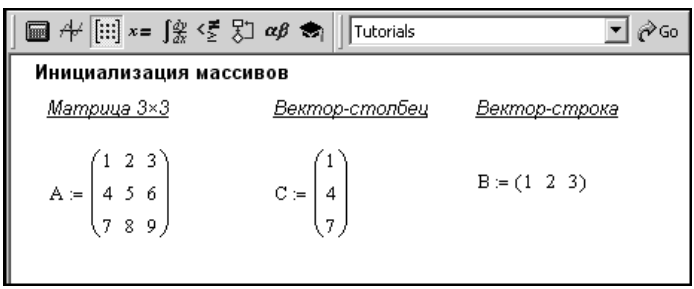


Рис. 2.87. Результат инициализации массивов

Для получения доступа к отдельному элементу массива необходимо указать название соответствующей переменной и индекс (или индексы) элемента в структуре массива. В частности, этим элементам можно присваивать новые значения, переопределяя тем самым исходный массив. Примеры приведены в документе на рис. 2.88.

Проверить значение массива можно так же, как и значение любой другой переменной: для этого вводится название массива, после которого указывается знак равенства. Если число строк или столбцов массива не превышает десяти, то массив отображается полностью. В противном случае видна только часть массива, а прочие элементы можно просмотреть с помощью полос прокрутки. Следует также отметить два важных обстоятельства. Во-первых, по умолчанию индексация элементов массивов в рабочем документе Mathcad начинается с нуля. Во-вторых, даже для одномерных массивов (векторов) все равно указывается два индекса: индексы строки и столбца, на пересечении которых находится элемент. Индексы разделяются запятой. Чтобы перейти

в режим ввода нижних индексов, нажимают клавишу <[> или используют пиктограмму на палитре **Matrix** (рис. 2.89).

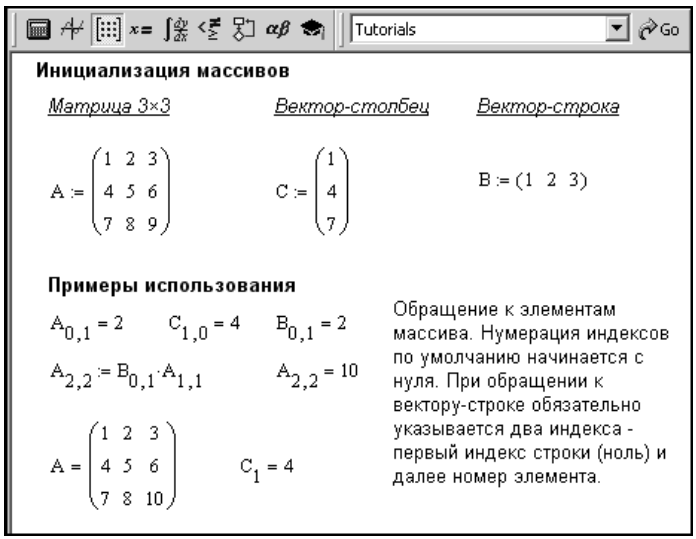


Рис. 2.88. Обращение к элементам массива

Рис. 2.89. Переход в режим ввода индексов с помощью палитры **Matrix**

Таким образом, при обращении к элементу вектора-строки, первым указывается нулевой индекс строки, после чего, через запятую, индекс элемента в этой строке. Для вектора-столбца ситуация противоположная: сначала указывается индекс элемента, после чего нулевой индекс столбца. Однако для вектора-столбца второй индекс указывать не обязательно (по умолчанию он полагается равным нулю), поэтому элемент вектора-столбца вызывают, указав только один индекс. Этот факт нашел свое отражение на рис. 2.88.

Нулевое начальное значение для индексирования элементов массива не очень удобно в первую очередь с той точки зрения, что обычно в линейной алгебре и ее приложениях начальное значение для индексов принимается равным единице. Чтобы избежать путаницы, разумно переопределить настройки Mathcad так, чтобы индексы в рабочих документах начинались со значения 1. Для этого следует переопределить значение системной переменной `ORIGIN`.

Эта переменная ответственна за определение начального индекса элементов массивов в Mathcad и ее значение по умолчанию равно нулю. Переопределить значение этой переменной можно прямо в рабочем документе обычным присваиванием. Второй способ состоит в вызове диалогового окна **Worksheet Options**. Диалоговое окно открывается в результате выбора команды **Tools | Worksheet Options**. В окне нужно перейти к вкладке **Built-In Variables** и установить нужное значение в поле **Array Origin (ORIGIN)**.

Каждый из способов переопределения переменной `ORIGIN` имеет свои преимущества и недостатки. Например, переопределяя переменную с помощью диалогового окна **Worksheet Options**, легко добиться универсальности процедуры индексирования элементов различных массивов во всем рабочем документе. С другой стороны, с помощью команды присваивания значения переменной `ORIGIN` в рабочем документе можно указать для разных массивов различные способы индексирования элементов. Выбор способа должен, очевидно, базироваться на особенностях решаемой задачи.

Интересно то обстоятельство, что значение переменной `ORIGIN` может быть отрицательным. Иногда этим можно воспользоваться, хотя данная возможность, все же, относится к разряду экзотических.

Пример 2.33. Начальный индекс массива

В документе на рис. 2.90 приведены некоторые примеры, иллюстрирующие вышеперечисленные возможности Mathcad.

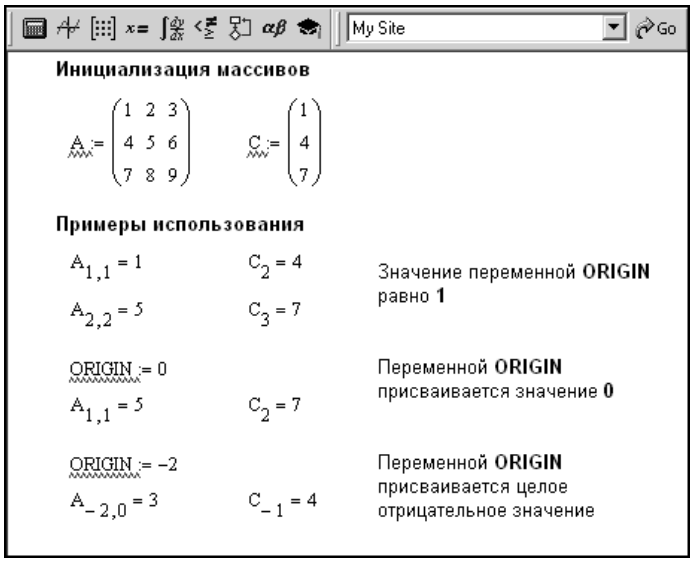


Рис. 2.90. Примеры использования переменной `ORIGIN`

Предварительно переменной `ORIGIN` в окне **Worksheet Options** установлено значение 1. Поэтому элемент с двумя единичными индексами, например, является верхним левым элементом матрицы. После того как переменной `ORIGIN` в рабочем документе присвоено нулевое значение, элемент с единичными индексами уже не будет левым верхним элементом матрицы. Наконец, если переменной `ORIGIN` присвоить целую отрицательную величину –2, то индексы элементов станут начинаться с отрицательного значения. Точнее, начальные их индексы будут совпадать, как и ранее, со значением переменной `ORIGIN`, а последующие индексы получатся добавлением единицы к предыдущему.

Пример 2.34. Определение массива

Описанный способ инициализации массивов не единственно возможный. Часто, например, значение элементов массива является функцией их индексов. В этом случае инициализировать массив можно с помощью переменных со значениями-диапазонами. Пример приведен в документе на рис. 2.91.

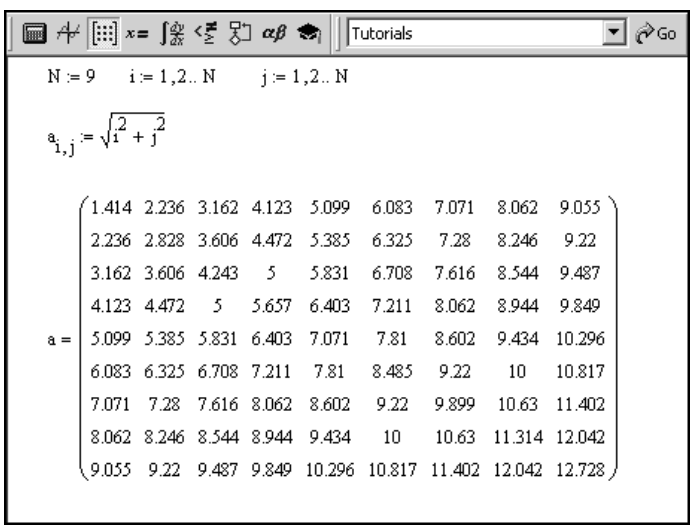


Рис. 2.91. Определение массива с помощью переменных с диапазонными значениями

Значение переменной `ORIGIN` предварительно установлено равным единице (т. е. индексы массивов нумеруются, начиная с единицы). Элемент массива `a` определяется в общем виде как корень квадратный из сумм квадратов своих индексов, а сами индексы предварительно инициализируются со значениями-диапазонами. Если диапазон изменений индексных переменных невелик (на рис. 2.91 это диапазон значений от 1 до 9), то массив, при проверке его зна-

чения, отображается в виде матрицы (см. рис. 2.91). В случае, когда диапазон изменения индексных переменных существенный (а на вкладке **Display Options** диалогового окна **Result Format** в раскрывающемся списке **Matrix display style** выбран элемент **Automatic**), в рабочем документе при проверке значения массива будет видна, как уже ранее отмечалось, только его часть (рис. 2.92).

Tutorials Go

$N := 20 \quad i := 1,2..N \quad j := 1,2..N$

$a_{i,j} := \sqrt{i^2 + j^2}$

a =

	1	2	3	4	5	6	7	8	9
1	1.414	2.236	3.162	4.123	5.099	6.083	7.071	8.062	9.055
2	2.236	2.828	3.606	4.472	5.385	6.325	7.28	8.246	9.22
3	3.162	3.606	4.243	5	5.831	6.708	7.616	8.544	9.487
4	4.123	4.472	5	5.657	6.403	7.211	8.062	8.944	9.849
5	5.099	5.385	5.831	6.403	7.071	7.81	8.602	9.434	10.296
6	6.083	6.325	6.708	7.211	7.81	8.485	9.22	10	10.817
7	7.071	7.28	7.616	8.062	8.602	9.22	9.899	10.63	11.402
8	8.062	8.246	8.544	8.944	9.434	10	10.63	11.314	12.042
9	9.055	9.22	9.487	9.849	10.296	10.817	11.402	12.042	12.728
10	10.05	10.198	10.44	10.77	11.18	11.662	12.207	12.806	13.454
11	11.045	11.18	11.402	11.705	12.083	12.53	13.038	13.601	14.213
12	12.042	12.166	12.369	12.649	13	13.416	13.892	14.422	15
13	13.038	13.153	13.342	13.601	13.928	14.318	14.765	15.264	15.811
14	14.036	14.142	14.318	14.56	14.866	15.232	15.652	16.125	16.643
15	15.033	15.133	15.297	15.524	15.811	16.155	16.553	17	17.493
16	16.031	16.125	16.279	16.492	16.763	17.088	17.464	17.889	18.358

Рис. 2.92. Отображена часть массива

Как несложно видеть, значение массива отображается в виде таблицы. Индексы элемента определяются номерами строки и столбца таблицы, на пересечении которых этот элемент находится. После выделения значения в таблице появляются полосы прокрутки, используя которые можно просмотреть всю таблицу (рис. 2.93).

В Mathcad предусмотрена возможность отображения значения массива в виде матрицы при любых его размерах. Для этого нужно выбрать команду **Format | Result**. В результате открывается диалоговое окно **Result Format**, в котором на вкладке **Display Options** в раскрывающемся списке **Matrix display style** следует выбрать элемент **Matrix** (рис. 2.94).

Tutorials Go

$N := 20 \quad i := 1,2..N \quad j := 1,2..N$

$a_{i,j} := \sqrt{i^2 + j^2}$

a =

	1	2	3	4	5	6	7	8	9
1	1.414	2.236	3.162	4.123	5.099	6.083	7.071	8.062	9.055
2	2.236	2.828	3.606	4.472	5.385	6.325	7.28	8.246	9.22
3	3.162	3.606	4.243	5	5.831	6.708	7.616	8.544	9.487
4	4.123	4.472	5	5.657	6.403	7.211	8.062	8.944	9.849
5	5.099	5.385	5.831	6.403	7.071	7.81	8.602	9.434	10.296
6	6.083	6.325	6.708	7.211	7.81	8.485	9.22	10	10.817
7	7.071	7.28	7.616	8.062	8.602	9.22	9.899	10.63	11.402
8	8.062	8.246	8.544	8.944	9.434	10	10.63	11.314	12.042
9	9.055	9.22	9.487	9.849	10.296	10.817	11.402	12.042	12.728
10	10.05	10.198	10.44	10.77	11.18	11.662	12.207	12.806	13.454
11	11.045	11.18	11.402	11.705	12.083	12.53	13.038	13.601	14.213
12	12.042	12.166	12.369	12.649	13	13.416	13.892	14.422	15
13	13.038	13.153	13.342	13.601	13.928	14.318	14.765	15.264	15.811
14	14.036	14.142	14.318	14.56	14.866	15.232	15.652	16.125	16.643
15	15.033	15.133	15.297	15.524	15.811	16.155	16.553	17	17.493
16	16.031	16.125	16.279	16.492	16.763	17.088	17.464	17.889	18.358

Рис. 2.93. Таблица с полосами прокрутки

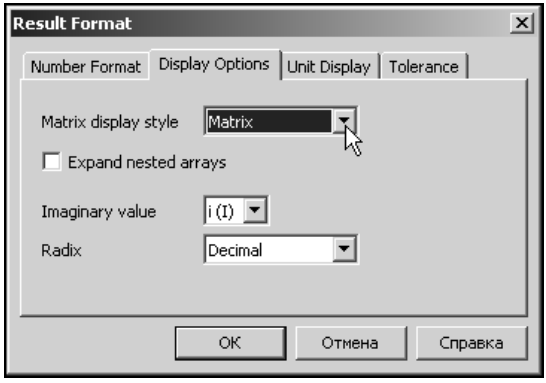


Рис. 2.94. Выбор способа отображения массивов в диалоговом окне **Result Format**

В результате, каких бы размеров ни был массив, он неизменно отображается в виде матрицы (рис. 2.95).

Tutorials Go

$N := 20 \quad i := 1,2..N \quad j := 1,2..N$

$a_{i,j} := \sqrt{i^2 + j^2}$

a =

1.414	2.236	3.162	4.123	5.099	6.083	7.071	8.062	9.055	10.05	11.045	12.042	13.038	14.036	15.033	16.031	17.029	18.028	19.026	20.025
2.236	2.828	3.606	4.472	5.385	6.325	7.28	8.246	9.22	10.198	11.18	12.166	13.153	14.142	15.133	16.125	17.117	18.111	19.105	20.1
3.162	3.606	4.243	5	5.831	6.708	7.616	8.544	9.487	10.44	11.402	12.369	13.342	14.318	15.297	16.279	17.263	18.248	19.235	20.224
4.123	4.472	5	5.657	6.403	7.211	8.062	8.944	9.849	10.77	11.705	12.649	13.601	14.56	15.524	16.492	17.464	18.439	19.416	20.396
5.099	5.385	5.831	6.403	7.071	7.81	8.602	9.434	10.296	11.18	12.083	13	13.928	14.866	15.811	16.763	17.72	18.682	19.647	20.616
6.083	6.325	6.708	7.211	7.81	8.485	9.22	10	10.817	11.662	12.53	13.416	14.318	15.232	16.155	17.088	18.028	18.974	19.925	20.881
7.071	7.28	7.616	8.062	8.602	9.22	9.899	10.63	11.402	12.207	13.038	13.892	14.765	15.652	16.553	17.464	18.385	19.313	20.248	21.19
8.062	8.246	8.544	8.944	9.434	10	10.63	11.314	12.042	12.806	13.601	14.422	15.264	16.125	17	17.889	18.788	19.698	20.616	21.541
9.055	9.22	9.487	9.849	10.296	10.817	11.402	12.042	12.728	13.454	14.213	15	15.811	16.643	17.493	18.358	19.235	20.125	21.024	21.932
10.05	10.198	10.44	10.77	11.18	11.662	12.207	12.806	13.454	14.142	14.866	15.62	16.401	17.205	18.028	18.868	19.723	20.591	21.471	22.361
11.045	11.18	11.402	11.705	12.083	12.53	13.038	13.601	14.213	14.866	15.556	16.279	17.029	17.804	18.601	19.416	20.248	21.095	21.954	22.825
12.042	12.166	12.369	12.649	13	13.416	13.892	14.422	15	15.62	16.279	16.971	17.692	18.439	19.209	20	20.809	21.633	22.472	23.324
13.038	13.153	13.342	13.601	13.928	14.318	14.765	15.264	15.811	16.401	17.029	17.692	18.385	19.105	19.849	20.616	21.401	22.204	23.022	23.854
14.036	14.142	14.318	14.56	14.866	15.232	15.652	16.125	16.643	17.205	17.804	18.439	19.105	19.799	20.518	21.26	22.023	22.804	23.601	24.413
15.033	15.133	15.297	15.524	15.811	16.155	16.553	17	17.493	18.028	18.601	19.209	19.849	20.518	21.213	21.932	22.672	23.431	24.207	25
16.031	16.125	16.279	16.492	16.763	17.088	17.464	17.889	18.358	18.868	19.416	20	20.616	21.26	21.932	22.627	23.345	24.083	24.839	25.612
17.029	17.117	17.263	17.464	17.72	18.028	18.385	18.788	19.235	19.723	20.248	20.809	21.401	22.023	22.672	23.345	24.042	24.759	25.495	26.249
18.028	18.111	18.248	18.439	18.682	18.974	19.313	19.698	20.125	20.591	21.095	21.633	22.204	22.804	23.431	24.083	24.759	25.456	26.173	26.907
19.026	19.105	19.235	19.416	19.647	19.925	20.248	20.616	21.024	21.471	21.954	22.472	23.022	23.601	24.207	24.839	25.495	26.173	26.87	27.586
20.025	20.1	20.224	20.396	20.616	20.881	21.19	21.541	21.932	22.361	22.825	23.324	23.854	24.413	25	25.612	26.249	26.907	27.586	28.284

Рис. 2.95. Отображение массива в виде матрицы

Существует и другая возможность: если в раскрывающемся списке **Matrix display style** выбрать элемент **Table**, то массивы всегда отображаются в виде таблицы (рис. 2.96).

Tutorials Go

$N := 5 \quad i := 1,2..N \quad j := 1,2..N$

$a_{i,j} := \sqrt{i^2 + j^2}$

a =

	1	2	3	4	5
1	1.414	2.236	3.162	4.123	5.099
2	2.236	2.828	3.606	4.472	5.385
3	3.162	3.606	4.243	5	5.831
4	4.123	4.472	5	5.657	6.403
5	5.099	5.385	5.831	6.403	7.071

Рис. 2.96. Отображение массива в виде таблицы

Как именно отображать массивы (в виде матрицы или таблицы), — дело сугубо индивидуальное. С одной стороны, математически более корректно, когда массив в рабочем документе имеет вид, какой ему и надлежит иметь (т. е. представлен в виде матрицы). Кроме того, при таком способе отображения массивов они всегда представляются в документе полностью, поэтому все их элементы видны пользователю. С другой стороны, это же может быть и недостатком, поскольку если массив велик, он займет существенную область в рабочем документе, может перекрыть другие команды на рабочем листе или вообще выйти за пределы видимой области документа. Кроме того, если массив представлен в виде таблицы, индексы элементов массива определяются легко и просто — ведь столбцы и строки таблицы пронумерованы. Здесь, однако, нужно заметить, что режим отображения меток для строк и столбцов таблицы можно отменить. Для этого на таблице щелкают правой кнопкой мыши и в раскрывшемся списке выбирают команду **Properties**. Открывается диалоговое окно **Component Properties**, в котором следует отменить опцию **Show column/row labels**. В результате таблица отображается без нумерации строк и столбцов.

Вложенные массивы

Исключительно мощный инструмент Mathcad — вложенные массивы. Идея, положенная в основу при реализации этих объектов, концептуально достаточно проста — это массивы, элементами которых, в свою очередь, являются массивы. В дальнейшем для простоты будем называть *вложенным массивом* любой массив, среди элементов которого встречаются массивы. Если не возникает путаницы, термин *вложенный* будем понимать в том смысле, что элементы массива имеют разные иерархические уровни, вложенные один в другой. Очень важно, что элементы-массивы вложенного массива могут иметь совершенно различные размерности. Это открывает поистине уникальные возможности. Здесь ограничимся кратким обзором.

Пример 2.35. Вложенные массивы

На рис. 2.97 приведен пример объявления трех различных массивов `a`, `b` и `c`. Эти массивы далее указываются в качестве элементов массива `d`. Последний, таким образом, является вложенным массивом.

По умолчанию задан режим, при котором в качестве значения вложенного массива отображается только его структура без указания в явном виде элементов. Например, для значения массива `d` в документе на рис. 2.97 приведено выражение `({3,1}{4,1}{2,2})`. Это значит, что массив `d` состоит из трех элементов, а они, в свою очередь, являются массивами из трех строк и одного столбца

(первый элемент), четырех строк и одного столбца (второй элемент) и, наконец, двух строк и двух столбцов (третий элемент). Чтобы вложенный массив отображался в явном виде, следует активизировать опцию **Expand nested arrays**, которую можно найти на вкладке **Display Options** диалогового окна **Result Format**. Окно, как уже упоминалось, открывается выбором команды **Format | Result** (рис. 2.98).

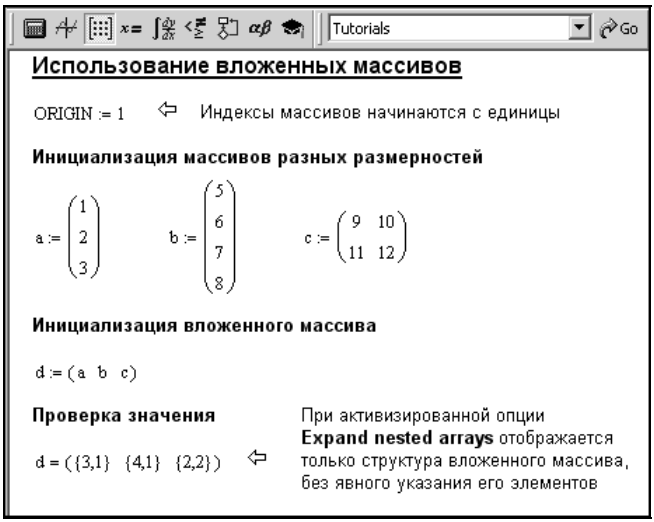


Рис. 2.97. Инициализация вложенного массива

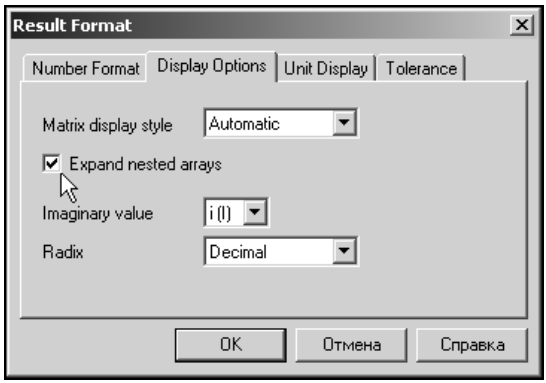


Рис. 2.98. Активизация опции **Expand nested arrays**

При активизированной опции **Expand nested arrays** значение массива d в рабочем листе будет иметь вид, как показано на рис. 2.99.

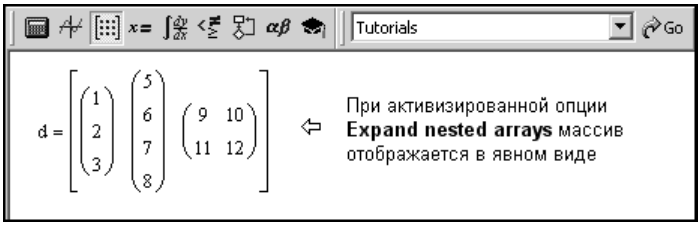


Рис. 2.99. Значение вложенного массива

Доступ к элементам вложенного массива получают так же, как и в случае обычного: следует указать название вложенного массива и индексы элемента. Пример приведен на рис. 2.100.

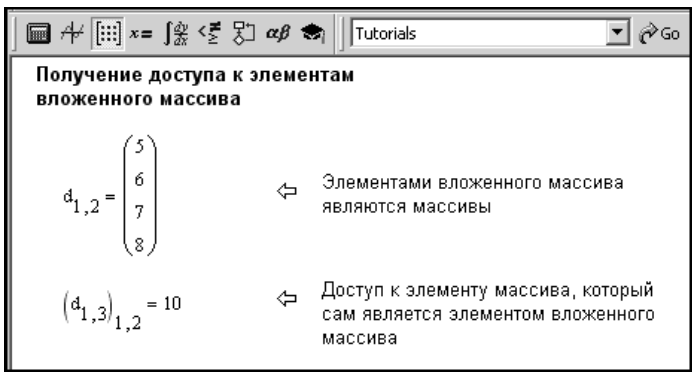


Рис. 2.100. Обращение к элементу вложенного массива

Если просто указать два индекса, то в качестве значения элемента вложенного массива в рассматриваемом случае будет возвращен массив — как и должно быть. Однако часто бывает необходимо добраться до элемента того массива, который сам является элементом вложенного массива. В этом случае после указания индексов, определяющих элемент-массив, указываются также и индексы элемента в структуре выделенного элемента-массива. Последовательность ввода командного кода такая: указывается название исходного вложенного массива, далее переходят в режим ввода индексов (клавиша <[>), и через запятую вводят индексы элемента-массива, а после этого указывают индексы элемента в этом массиве (придется еще раз нажать клавишу <[>). Индексы имеют разный иерархический уровень, т. е. это не четыре указанных через запятую индекса, а индексы для элемента с индексами. Пример можно найти в документе на рис. 2.100. Скобки возле названия элемента-массива добавляются автоматически при вводе второй серии индексов. Если уровней индексов больше, чем два, принцип обращения к элементам остается

таким же: сначала указываются индексы элемента-массива, далее индексы элемента-массива внутри этого массива, после этого снова индексы элемента внутри этого массива и т. д. В этом случае речь идет о вложенном массиве, элементом которого является вложенный массив. Такая структура определена в примере на рис. 2.101.

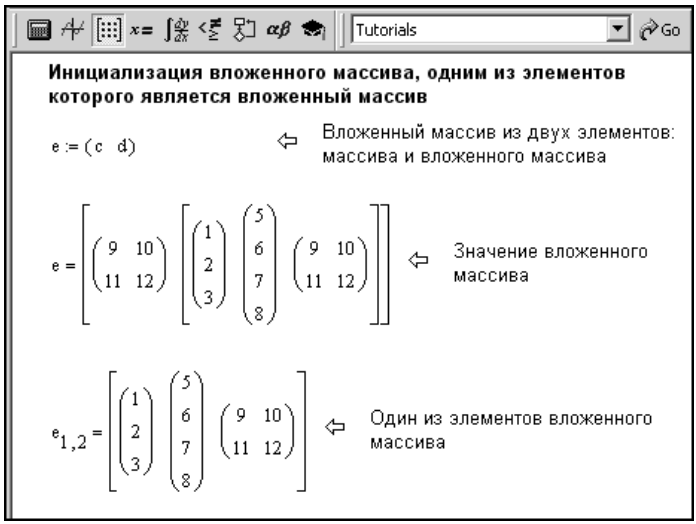


Рис. 2.101. Вложенный массив со сложной структурой

Обращение к элементам такого массива осуществляется так же, как и для более простых структур, рассмотренных ранее.

Операции с массивами

Само собой разумеется, что на объявлении массивов работа не заканчивается, а только начинается. Перефразируя известное выражение, можно сказать, что если массивы в рабочем документе объявляются, значит, для чего-то это нужно. Поэтому разумно рассмотреть те операции, которые можно выполнять с массивами.

В первую очередь отметим, что по отношению к векторам и матрицам могут применяться такие арифметические операции, как сложение, вычитание, деление и умножение. Правда, если операндами служат массивы, они должны удовлетворять некоторым условиям. Так, для того, чтобы можно было сложить два массива, они должны иметь одинаковую размерность и размер. В этом случае массивы складываются так же, как и обычные переменные: например, чтобы вычислить сумму массивов `A` и `B`, в рабочем документе вводят команду `A+B`. При этом в результате получается массив тех же размеров,

элементы которого равны сумме соответствующих элементов исходных массивов. Точно так же вычисляется и разность массивов: `A-B`. Причем последнюю операцию можно рассматривать как сумму массива `A` и массива, противоположного к `B`, т. е. `-B`.

Когда число столбцов массива `A` равно числу строк массива `B`, можно вычислять их произведение с помощью обычного оператора умножения. В результате получается массив, число строк которого равно числу строк первого массива, а число столбцов — числу столбцов второго. Так, если `C = A*B`, то элементы массива `C` вычисляются по формуле $C_{ij} = \sum_k A_{ik} B_{kj}$, а сумма берется по всем индексам k . Таким образом, операция умножения массивов в Mathcad соответствует правилу умножения матриц.

Помимо оператора умножения, при работе с массивами допустим и оператор деления. Правда, в этом случае второй операнд должен быть численным. В результате каждый элемент первого операнда-массива делится на второй операнд-число. К квадратным матрицам (массив с одинаковым количеством строк и столбцов) применима операция возведения в степень (комбинация клавиш <Shift>+<6>). В качестве степени можно указывать любое целое число, в том числе и отрицательное.

Пример 2.36. Операции с массивами

На рис. 2.102 приведен фрагмент рабочего документа, в котором можно найти примеры выполнения упомянутых операций с массивами.

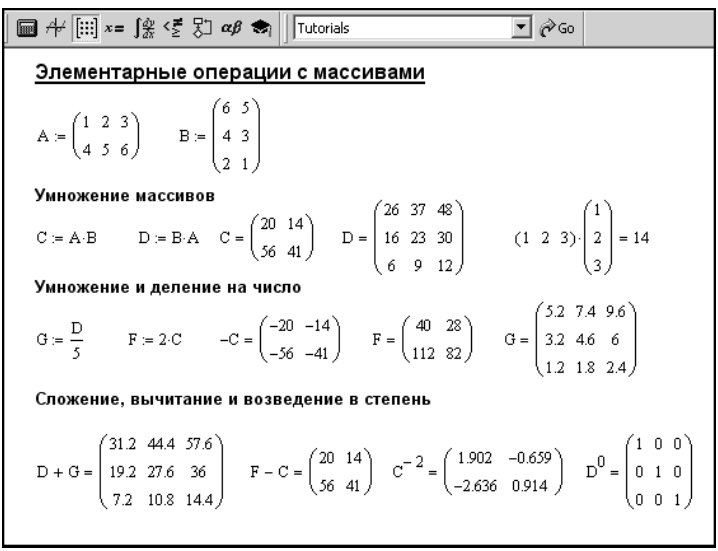


Рис. 2.102. Элементарные операции с массивами

На этих простых операциях с массивами возможности Mathcad, разумеется, не исчерпываются. Далее рассмотрим процедуры транспонирования, обращения матриц, вычисления детерминанта матрицы, а также нахождения векторных произведений.

Пример 2.37. Транспонирование матриц

Как известно, в процессе транспонирования матрицы ее строки и столбцы меняются местами. В Mathcad транспонирование выполняется указанием справа вверху у матрицы символа транспонирования (буква *T*). Это выполняется комбинацией клавиш <Ctrl>+<1> или специальной пиктограммой, размещенной на палитре **Matrix** (рис. 2.103).

В документе на рис. 2.104 приведены простые примеры транспонирования матриц.

Рис. 2.103. Использование палитры **Matrix** для ввода оператора транспонирования

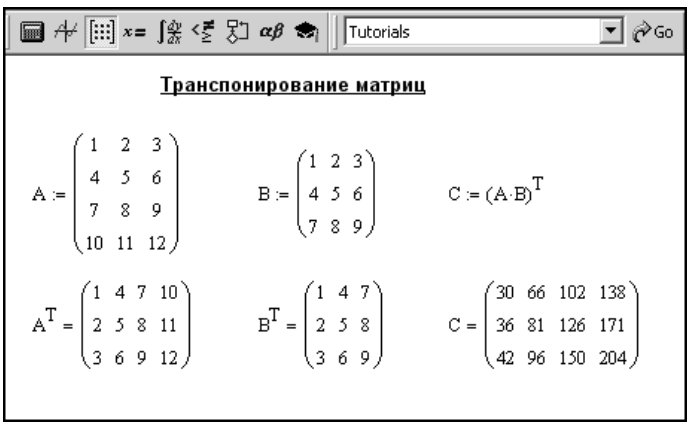


Рис. 2.104. Примеры транспонирования матриц

Очень часто в практических задачах приходится искать обратные матрицы. Как известно, по определению обратной к данной называется такая матрица, которая, будучи умноженной на исходную, в результате дает единичную матрицу (у нее все диагональные элементы равны единице, а недиагональные — нулю). Обратная матрица может быть определена только для невырожденной

квадратной матрицы. Невырожденной, в свою очередь, называется матрица с ненулевым определителем (детерминантом). Как вычисляется детерминант, будет показано далее, а прежде определимся с вычислением обратной матрицы. Эта процедура по сути своей эквивалентна возведению матрицы в минус первую степень, поэтому найти обратную матрицу можно теми методами, что описывались ранее при описании операции возведения матрицы в степень. Однако на палитре **Matrix** есть специальная пиктограмма, воспользовавшись которой можно найти нужную матрицу (рис. 2.105).

Рис. 2.105. Пиктограмма вычисления обратной матрицы

Пример 2.38. Обращение матриц

Примеры вычисления обратных матриц представлены на рис. 2.106.

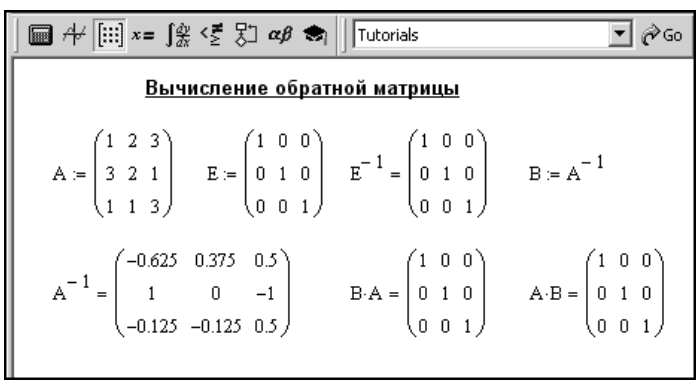


Рис. 2.106. Примеры вычисления обратной матрицы

В Mathcad существует возможность вычислять ряд важных характеристик матриц, среди которых главной, безусловно, является детерминант (или определитель) матрицы. Для выполнения этой операции достаточно заключить название матрицы, для которой вычисляется детерминант, в прямые скобки (комбинация клавиш <Shift>+<\>) — как при вычислении модуля числа. Можно также воспользоваться специальной пиктограммой на палитре **Matrix** (рис. 2.107).

Рис. 2.107. Пиктограмма вычисления детерминанта на палитре **Matrix**

Следует отметить, что детерминант может быть вычислен только для квадратной матрицы.

Пример 2.39. Определитель матрицы

Примеры вычисления детерминанта представлены в документе на рис. 2.108.

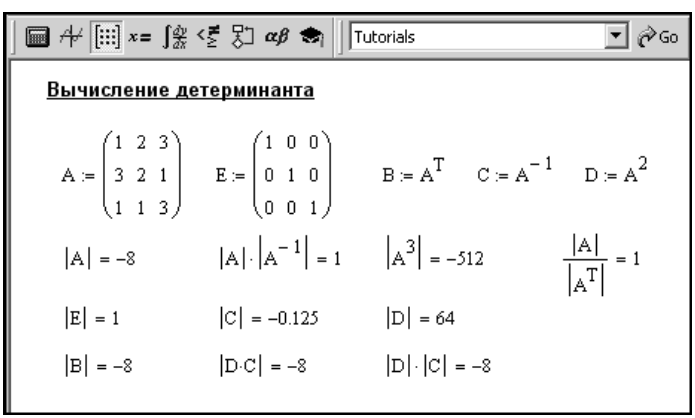


Рис. 2.108. Примеры вычисления детерминанта

Как можно заметить, детерминант единичной матрицы равен единице. Кроме того, при транспонировании матрицы детерминант не меняется, а детерминант обратной матрицы равен единице, деленной на детерминант исходной матрицы. Это свойство — следствие более общего правила, согласно которому детерминант произведения нескольких матриц равен произведению их детерминантов.

Ранее рассказывалось о том, как можно вычислить произведение двух или более матриц и, в частности, двух векторов. Речь шла о так называемом скалярном произведении векторов. Результат операции есть скаляр (число), равный произведению модулей векторов на косинус угла между ними. Кроме скалярного, часто встречается векторное произведение. Это бинарная операция, операндами в которой выступают векторы, а результат — тоже вектор. Направлен он перпендикулярно плоскости, в которой размещены векторы-

операнды, из двух возможных направлений выбирается то, что соответствует правилу правого буравчика (при вращении буравчика от первого вектора ко второму буравчик должен двигаться в направлении вектора-результата операции), а модуль вектора-результата равен произведению модулей векторов-операндов на синус угла между ними.

Векторное произведение в Mathcad можно вычислить с помощью специального оператора, который вводится нажатием комбинации клавиш <Ctrl>+<8> или посредством пиктограммы на палитре **Matrix** (рис. 2.109).

Рис. 2.109. Пиктограмма вычисления векторного произведения на палитре **Matrix**

Применяться операция вычисления векторного произведения может, как уже отмечалось, только к векторам, причем это должны быть векторы-столбцы из трех элементов.

Пример 2.40. Векторное произведение

В документе на рис. 2.110 можно найти примеры вычисления векторных произведений.

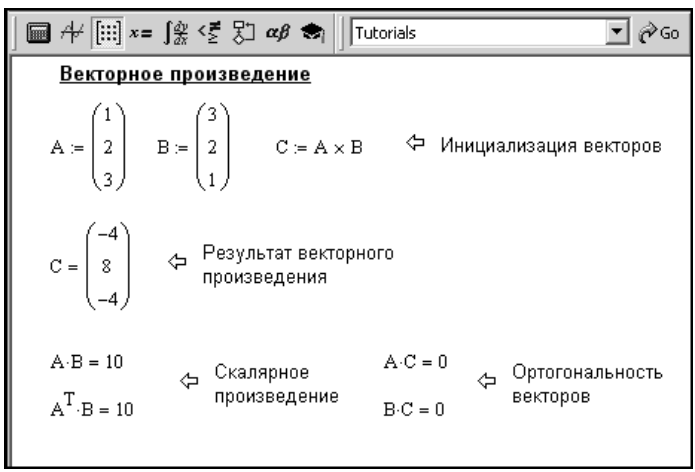


Рис. 2.110. Вычисление векторного произведения

В том, что вектор, получающийся в результате векторного произведения двух других векторов, перпендикулярен им обоим, убедиться достаточно просто — следует вычислить скалярное произведение результирующего вектора на каждый из исходных. Для ортогональных векторов такое произведение, как известно, равно нулю.

В Mathcad на палитре **Matrix** есть пиктограмма, позволяющая ввести в рабочий документ оператор вычисления суммы элементов вектора (рис. 2.111).

Этот же оператор можно ввести при помощи комбинации клавиш <Ctrl>+<4>.

Рис. 2.111. Пиктограмма вставки оператора вычисления суммы элементов вектора на палитре **Matrix**

Пример 2.41. Сумма элементов

Примеры использования оператора представлены на рис. 2.112.

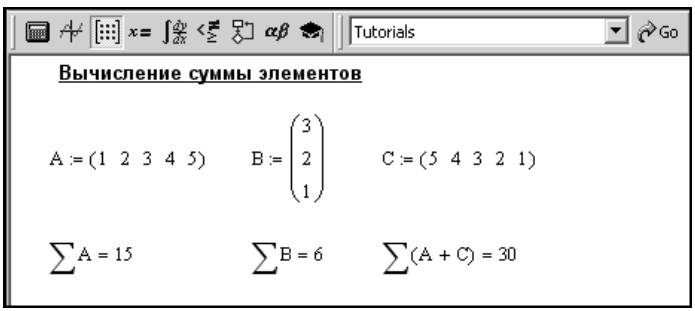


Рис. 2.112. Примеры вычисления суммы элементов векторов

С практической точки зрения вычисление суммы элементов вектора — достаточно экзотическая операция, поскольку, с одной стороны, это операция унарная, и, кроме того, в результате ее выполнения вектору в соответствие ставится скаляр. По этим причинам дать простую физическую интерпретацию данной операции достаточно сложно. На практике такая операция имеет, как правило, несколько модифицированный вид. В качестве примера можно привести процедуру выполнения параллельных расчетов или при определении

нормы вектора (обычно это корень квадратный из суммы квадратов элементов, но может быть и сумма модулей элементов). В любом случае применение процедуры тесно связано с другой, достаточно важной и полезной процедурой — а именно процедурой векторизации.

Если кратко описывать суть последней, то ситуация выглядит следующим образом: массив указывается в качестве аргумента некоторой функции, которая определена для скалярных аргументов. Применяя процедуру векторизации, можно вычислить значение функции для каждого из элементов упомянутого массива. Это исключительно удобный способ выполнения параллельных расчетов, когда одни и те же вычисления проделываются с разными входными значениями. Вместо того чтобы каждый раз вводить одну и ту же функцию с разными аргументами, эти аргументы можно записать в виде элементов некоторого массива, потом указать этот массив в качестве аргумента функции и выполнить процедуру векторизации. В результате будет получен массив тех же размеров, что и указанный аргументом, а его элементы определяют результат вычисления функции для аргумента, стоящего в исходном массиве на соответствующем месте. Для выполнения векторизации на палитре **Matrix** имеется специальная пиктограмма (рис. 2.113).

Рис. 2.113. Пиктограмма векторизации на палитре **Matrix**

Удобна также комбинация клавиш <Ctrl>+<–>. При этом нужно следить, чтобы перед векторизацией линиями редактирования было выделено именно то выражение, по отношению к которому векторизация применяется. В противном случае результат может быть, по меньшей мере, обескураживающим.

Пример 2.42. Процедура векторизации

На рис. 2.114 представлены примеры векторизации различных операций.

После применения процедуры векторизации над соответствующим выражением появляется символ вектора. По нему, кстати, можно впоследствии определить, к какой части команды была применена векторизация.

Tutorials Go

Процедура векторизации

Инициализация массивов

$A := \frac{\pi}{12}\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix}$ $F := \frac{12}{\pi}\begin{pmatrix}6 & 5 & 4\\ 3 & 2 & 1\end{pmatrix}$ $B := (1\ \ -2\ \ 3\ \ -4\ \ 5)$ $C := (1\ \ 2\ \ 3)$ $D := (3\ \ 2\ \ 1)$

Примеры векторизации операций

$\overrightarrow{\sin(A)} = \begin{pmatrix}0.259 & 0.5 & 0.707\\ 0.866 & 0.966 & 1\end{pmatrix}$ ⇦ Аргументом синуса является матрица. В результате вычисляется матрица, элементы которой равны синусам элементов исходной матрицы

$\overrightarrow{(A\cdot F)} = \begin{pmatrix}6 & 10 & 12\\ 12 & 10 & 6\end{pmatrix}$ ⇦ Результатом операции является матрица, элементы которой получаются умножением соответствующих элементов исходных матриц

$\sum B = 3$ ⇦ Сумма элементов вектора. Векторизация не применялась

$\sum \overrightarrow{|B|} = 15$ ⇦ Сумма модулей элементов вектора

$\sqrt{\sum \overrightarrow{B^2}} = 7.416$ ⇦ Корень квадратный из суммы квадратов элементов вектора (модуль вектора)

$D^T\cdot C^T = 10$ ⇦ Скалярное произведение векторов

$\overrightarrow{(D\cdot C)} = (3\ \ 4\ \ 3)$ ⇦ Результатом операции является вектор-строка, элементы которого равны произведению соответствующих элементов исходных векторов

$\overrightarrow{(D^T\cdot C^T)} = \begin{pmatrix}3\\ 4\\ 3\end{pmatrix}$ ⇦ Результатом операции является вектор-столбец, элементы которого равны произведению соответствующих элементов исходных векторов

$\overrightarrow{F^A} = \begin{pmatrix}2.27 & 4.685 & 8.511\\ 12.857 & 14.319 & 8.208\end{pmatrix}$ ⇦ Результатом операции является матрица, элементы которой равны элементам матрицы F, возведенным в степень, определяемую элементами матрицы A

$\overrightarrow{A^F} = \begin{pmatrix}0.048 & 0.048 & 0.128\\ 1.809 & 47.26 & 40.72\end{pmatrix}$ ⇦ Результатом операции является матрица, элементы которой равны элементам матрицы A, возведенным в степень, определяемую элементами матрицы F, возведенными в степень, определяемую, в свою очередь, элементами матрицы A

Рис. 2.114. Примеры выполнения векторизации

Преобразование массивов

Даже если массив состоит из однотипных элементов с простой структурой, на практике часто бывает недостаточно получить доступ к тому или иному элементу массива, приходится обращаться к целым его фрагментам: строкам, столбцам или подматрицам. Все это можно с успехом делать в Mathcad. Далее кратко описываются соответствующие методы.

Пример 2.43. Выделение строк и столбцов

Чтобы выделить из массива (матрицы) столбец, существует специальный оператор, для которого на палитре **Matrix** выделена пиктограмма (рис. 2.115).

Рис. 2.115. Пиктограмма оператора выделения столбца матрицы

То же самое может быть сделано с помощью комбинации клавиш <Ctrl>+<6>. В результате у названия матрицы, столбец которой извлекается, появляется структурный заполнитель для ввода верхнего индекса. Вводимое туда число является индексом столбца массива, к которому должен быть получен доступ. Пример приведен на рис. 2.116.

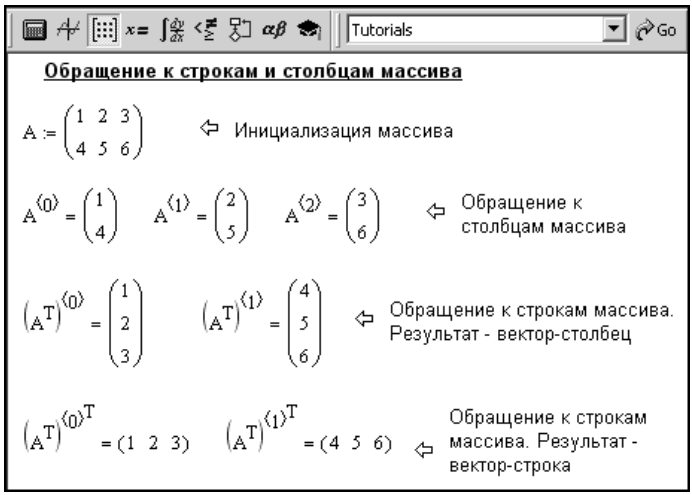


Рис. 2.116. Примеры обращения к строкам и столбцам массива

В том случае, если нужно извлечь строку массива, поступают следующим образом: сначала массив транспонируют, после чего обращаются к столбцу этого транспонированного массива (который совпадает со строкой исходного). Однако в этом случае результат будет вектором-столбцом. Если нужно получить вектор-строку, то результат извлечения столбца транспонированного вектора также транспонируется (см. рис. 2.116).

Иногда в процессе работы возникает необходимость изменить размеры массива. Если для этого к одному массиву следует присоединить другой (возможны несколько способов такого объединения) — применяют специальные функции, которые, среди прочих, описываются в следующем подразделе. Здесь остановимся на вопросах, связанных с добавлением или удалением строк и столбцов массива.

Перед добавлением строк и (или) столбцов в массив (матрицу), следует выделить элемент. Строки будут добавляться сразу под строкой с этим элементом, а столбцы добавляются справа от столбца с выделенным элементом. На рис. 2.117 выделен элемент, стоящий на пересечении первой строки и второго столбца.

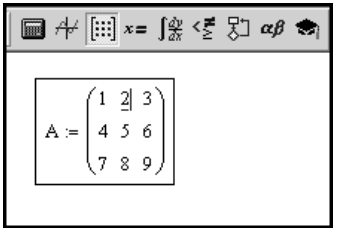


Рис. 2.117. Перед вставкой строк (столбцов) следует выделить элемент

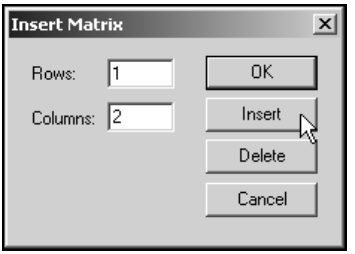


Рис. 2.118. Вставляется одна строка и два столбца

После того как элемент выделен, выбирают команду **Insert | Matrix** (можно воспользоваться пиктограммой на палитре **Matrix**). В открывшемся диалоговом окне **Insert Matrix** в поле **Rows** указывается число вставляемых строк, а в поле **Columns** — столбцов. После этого следует щелкнуть на кнопке **Insert** (рис. 2.118).

В результате исходная матрица меняется в размерах. В частности, на рис. 2.119 в матрицу добавлена одна строка и два столбца.

Удаляются строки и столбцы практически так же. Разница состоит лишь в том, что в диалоговом окне **Insert Matrix** следует щелкнуть не на кнопке **Insert**, а на кнопке **Delete**. На рис. 2.120 в поле **Rows** указано значение 0, а в поле **Columns** — 1.

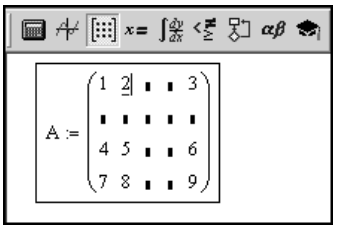


Рис. 2.119. Результат вставки строк и столбцов

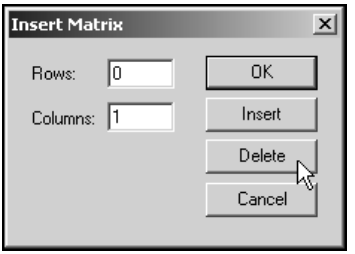


Рис. 2.120. Удаление строк и столбцов

Это соответствует удалению одного столбца (того, в котором находится выделенный перед началом процедуры удаления элемент). Результат удаления можно видеть на рис. 2.121.

Надо отметить, что, как и в случае добавления строк и столбцов, предварительно был выделен элемент в первой строке и втором столбце (см. рис. 2.117). Поэтому, как несложно убедиться из сравнения рис. 2.117 и 2.121, в исходной матрице удален именно второй столбец.

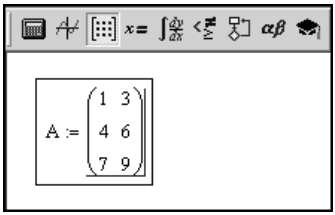


Рис. 2.121. Результат удаления столбца

Функции для работы с массивами

Очень часто встроенные функции Mathcad, предназначенные, по сути, для работы с численными аргументами, корректно обрабатывают и аргументы-массивы. В крайнем случае, можно воспользоваться процедурой векторизации. Несмотря на все это, в Mathcad есть функции, предназначенные для работы именно с массивами. Далее дается их краткий обзор.

Пример 2.44. Функция *matrix()*

Если элементы матрицы определяются через индексы, т. е. могут быть представлены в виде функции от индексов, то для создания матрицы можно воспользоваться функцией `matrix()`. Функция имеет три аргумента: сначала указывается число строк и столбцов создаваемой матрицы, после чего название функции от двух аргументов, согласно которой вычисляются элементы матрицы. Например, командой `matrix(N,M,F)` создается матрица из `N` строк и `M` столбцов, а ее элементы вычисляются по формуле `F(i,j)`, где `i` и `j` — индексы

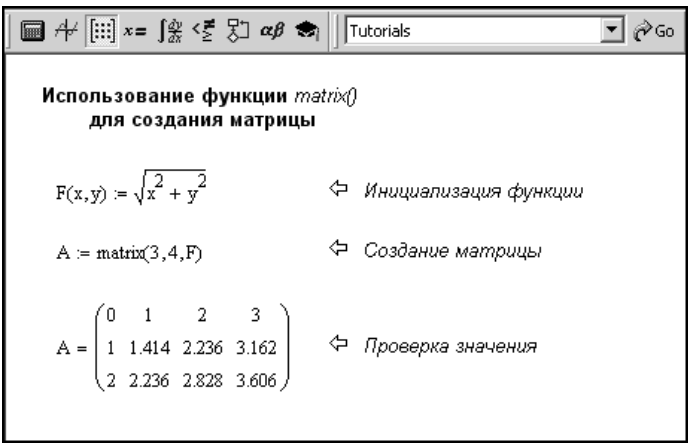


Рис. 2.122. Использование функции `matrix()` для создания матрицы

элемента (по умолчанию `i=0,1,2,…,N-1` и `j=0,1,2,…,M-1`). Указывается только название функции-аргумента `F`, ее аргументы не перечисляются. При этом предварительно данная функция должна быть определена как функция двух аргументов. В документе на рис. 2.122 приведен пример создания матрицы с помощью функции `matrix()`.

Пример 2.45. Функции *stack()* и *augment()*

Очень часто новая матрица создается путем объединения уже существующих матриц разных размеров. Обычно в таких случаях удобны функции `stack()` и `augment()`. Аргументами функций указываются объединяемые матрицы. Функцией `stack()` матрицы последовательно добавляются одна к другой снизу, а функцией `augment()` матрицы объединяются слева направо. Чтобы матрицы можно было указывать аргументами функции `stack()`, они должны иметь одинаковое число столбцов. Для функции `augment()` матрицы-аргументы должны иметь одинаковое число строк. Количество аргументов в обеих функциях произвольно. Примеры представлены в документе на рис. 2.123.

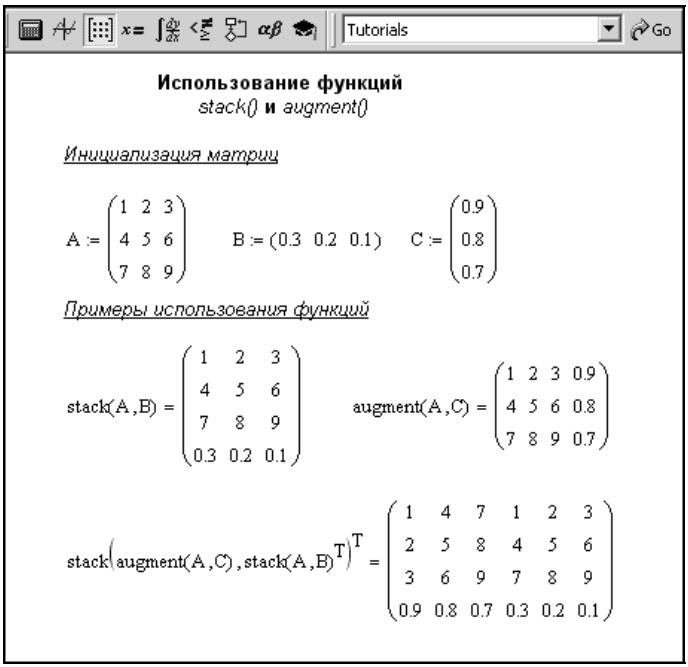


Рис. 2.123. Примеры использования функций `stack()` и `augment()`

Пример 2.46. Функция *submatrix()*

Практически важная задача — извлечение подматрицы из некоторой заданной матрицы. В этом случае прибегают к помощи функции `submatrix()`. У нее пять аргументов. Первым указывается исходная матрица, из которой извлекается подматрица. Два следующих аргумента — это начальная и конечная строки извлекаемой подматрицы из базовой матрицы, указанной первым аргументом функции `submatrix()`. Два последних аргумента функции — столбцы (начальный и конечный) извлекаемой подматрицы. Пример вызова функции `submatrix()` можно видеть на рис. 2.124.

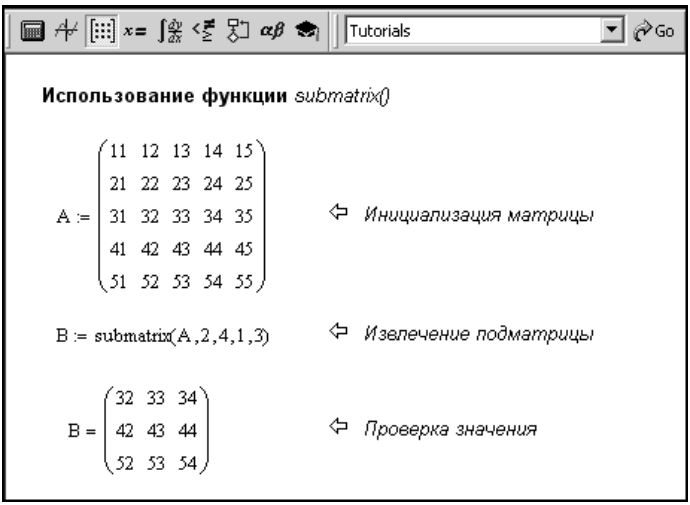


Рис. 2.124. Пример извлечения подматрицы

Помимо этого, в Mathcad есть ряд функций для создания матриц специальных типов. Соответствующие функции (вместе с аргументами) приведены в табл. 2.11.

Таблица 2.11. *Функции создания матриц специального типа*

Функции и их аргументы	Описание
`diag(v)`	Функцией создается диагональная матрица, диагональные элементы которой определяются вектором `v`
`geninv(A)`	Функцией возвращается левосторонняя обратная матрица к действительной матрице `A`. Число строк матрицы `A` должно быть не меньше числа ее столбцов

Таблица 2.11 *(окончание)*

Функции и их аргументы	Описание
`identity(n)`	Функцией в качестве результата возвращается единичная матрица ранга `n`
`rref(A)`	В качестве результата функцией возвращается матрица, получающаяся из действительной матрицы `A` последовательным вычитанием строк

Некоторые функции в Mathcad предназначены для определения размеров массивов и связанных с ними характеристик. В табл. 2.12 перечислены эти функции и дано их краткое описание: как и с какими аргументами их следует вызывать и какая характеристика массива возвращается в качестве результата.

Таблица 2.12. *Функции определения размеров массивов*

Функции и их аргументы	Описание
`cols(A)`	Возвращает число столбцов матрицы `A`
`last(v)`	Возвращает индекс последнего элемента вектора `v`
`length(v)`	Возвращает число элементов вектора `v`
`rows(A)`	Возвращает число строк матрицы `A`

Важными характеристиками матриц являются такие, как след (сумма диагональных элементов), собственные числа, собственные векторы и т. п. Часть функций, полезных при вычислении таких характеристик, можно найти в табл. 2.13.

Таблица 2.13. *Функции вычисления основных характеристик матриц*

Функции и их аргументы	Описание
`rank(A)`	Возвращает ранг матрицы `A`
`tr(M)`	Возвращает сумму диагональных элементов (след) квадратной матрицы `M`
`eigenvals(M)`	Вычисляет собственные числа квадратной матрицы `M`. Результат представляется в виде вектора
`eigenvecs(M)`	Вычисляет нормированные на единицу собственные векторы квадратной матрицы `M`. Результат возвращается в виде матрицы. Собственные векторы формируют столбики этой матрицы. Соответствие между собственными векторами и собственными числами устанавливается по строкам
`eigenvec(M,z)`	Результат вызова функции — собственный вектор квадратной матрицы `M`, соответствующий собственному числу `z` этой матрицы

Пример 2.47. Характеристики матриц

Примеры вызова некоторых из перечисленных функций представлены в документе на рис. 2.125.

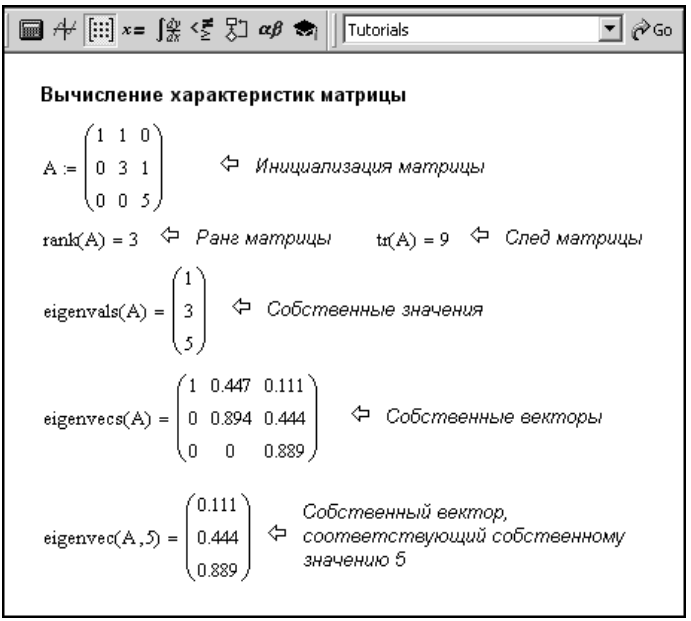


Рис. 2.125. Примеры вычисления основных характеристик матриц

Безусловно, на этом функции Mathcad для работы с массивами не исчерпываются — их существенно больше. Они будут описываться в последующих главах книги по мере необходимости. Это же относится и к примерам вызова тех функций, что рассматривались в данном подразделе.

Решение уравнений и систем

В данном разделе кратко рассматриваются функции и методы, которые могут использоваться в Mathcad для численного решения алгебраических уравнений и систем. Сразу следует отметить, что далее в книге соответствующие приемы еще будут обсуждаться: в частности, в главе, посвященной численным методам и при обсуждении символьных вычислений. Здесь же приводятся элементарные сведения по данному вопросу и только в той части, что касается поиска приближенного численного решения.

Для решения уравнений предназначена функция `root()`. Первым аргументом указывается функциональная зависимость, которая должна обращаться в нуль,

а вторым — переменная, относительно которой решается уравнение. Например, вызов команды `root(f(x),x)` приведет к решению уравнения `f(x)=0` относительно переменной `x`. Иногда корень уравнения следует искать на каком-то определенном интервале значений переменной. В этом случае у функции `root()` указывается еще два дополнительных аргумента — границы диапазона, на котором ищутся решения. Так, если решение упомянутого уравнения следует искать на диапазоне значений переменной `x` от `a` до `b`, то соответствующая команда будет иметь вид `root(f(x),x,a,b)`. Причем если два последних аргумента функции `root()` не указаны, то для поиска решения уравнения предварительно переменной, относительно которой это уравнение решается, следует присвоить начальное значение. Чем ближе оно окажется к искомому корню уравнения, тем эффективнее будет проходить процесс вычисления этого корня.

Отдельный и довольно часто встречающийся класс выражений составляют полиномы. Искать корни полинома можно с помощью функции `polyroots()`. Аргументом последней указывается вектор-столбец коэффициентов полинома (первый элемент соответствует нулевой степени переменной полинома, второй элемент — первой степени и т. д.).

Системы линейных уравнений решаются посредством функции `lsolve()`, которая имеет два аргумента: первый аргумент является матрицей коэффициентов системы уравнений, а второй — вектором-столбцом правых частей этих уравнений.

Пример 2.48. Решение уравнений и систем

Примеры решения уравнений и линейных систем представлены в документе на рис. 2.126.

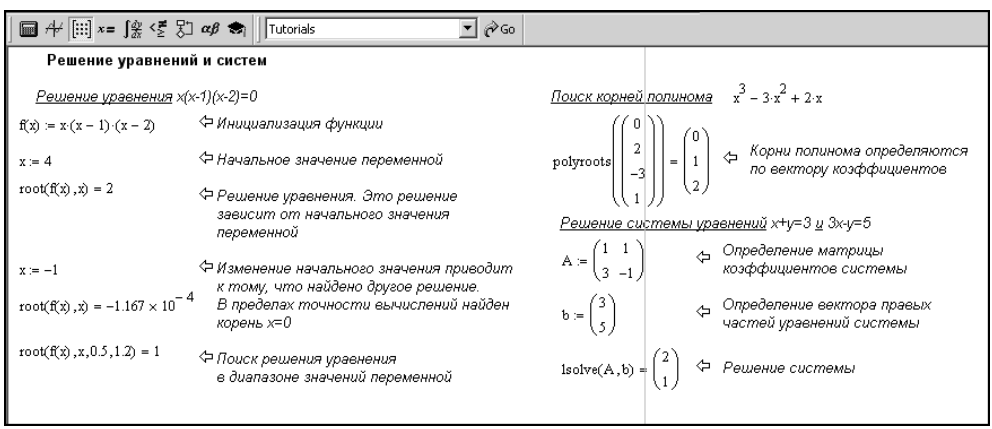


Рис. 2.126. Решение уравнений и линейных систем

Очень многие прикладные задачи связаны с решением систем уравнений и, к сожалению (или к счастью — кому как больше нравится), эти системы обычно линейными не являются. Методы решения нелинейных систем методологически существенно сложнее, однако в Mathcad есть ряд утилит, позволяющих успешно справляться и с этим заданием.

Обычно системы нелинейных уравнений решают с помощью функции `Find()`. Ее аргументами указываются переменные, относительно которых решается система. Ссылки на саму систему в аргументах функции нет, поэтому ее определяют непосредственно перед обращением к функции `Find()`. Другими словами, сначала записывается система уравнений, а уже после этого вызывается функция `Find()` с соответствующими аргументами.

Пример 2.49. Решение нелинейной системы

На рис. 2.127 представлен пример решения системы уравнений с использованием функции `Find()`.

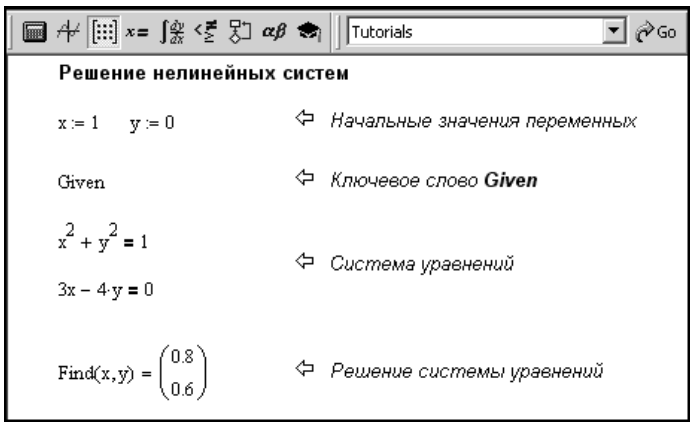


Рис. 2.127. Решение нелинейных систем

Первыми двумя командами переменным `x` и `y`, относительно которых будет решаться система уравнений, присваиваются начальные значения. Далее в рабочий документ вводится обязательное ключевое слово `Given`. Вводится оно как обычное название переменной. После этого ключевого слова следует ввести уравнения системы. В данном случае уравнений два. Вообще же число уравнений должно равняться числу переменных.

Относительно ввода уравнений нужно сделать одно важное замечание. Оно касается способа ввода знака равенства. Здесь необходим оператор так называемого логического равенства — он вводится нажатием комбинации клавиш

<Ctrl>+<=>. Можно также воспользоваться пиктограммой на палитре **Boolean** (рис. 2.128).

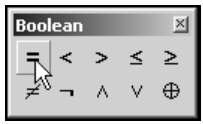


Рис. 2.128. Использование пиктограммы палитры **Boolean** для вставки логического оператора равенства

После того как система уравнений задана, вызывается функция `Find()`, а ее аргументами будут переменные, относительно которых система решается (см. рис. 2.127).

Представленных сведений бывает вполне достаточно для решения большей части задач. Однако любое серьезное исследование подразумевает привлечение и методик соответствующего уровня.

Вычисление производных

В этой главе дадим также краткий обзор методов вычисления в численном виде производных и определенных интегралов.

Оператор вычисления производной вводится нажатием комбинации клавиш <Shift>+</>. Можно с этой целью воспользоваться специальной пиктограммой на палитре **Calculus** (рис. 2.129).

Сам оператор имеет вид, показанный на рис. 2.130. Он содержит два структурных заполнителя. Вместо одного (того, что в знаменателе) вводится название переменной, по которой выполняется дифференцирование. Другой заполнитель нужен для ввода дифференцируемого выражения или названия функции (с аргументом).

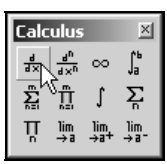


Рис. 1.129. Вставка оператора вычисления производной с помощью пиктограммы на палитре **Calculus**

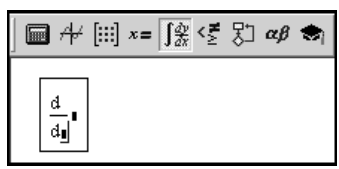


Рис. 2.130. Оператор вычисления производной

Пример 2.50. Вычисление производных

На рис. 2.131 приведен пример вычисления производной в численном виде от функции и полиномиального выражения.

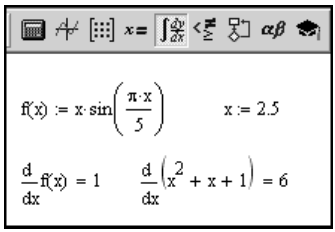


Рис. 2.131. Вычисление производной

В рассмотренном примере сначала определяется функция и задается значение переменной, и только после этого вычисляется производная. Предварительное определение значения переменной, по которой вычисляется производная, — момент принципиальный. Численное значение производной вычисляется, как и ранее, вводом знака равенства.

В Mathcad можно вычислять в численном виде и производные более высоких порядков. К сожалению, порядок производной не может превышать 5. Соответствующий оператор вводится нажатием комбинации клавиш <Ctrl>+<Shift>+</> или с помощью пиктограммы на палитре **Calculus** (рис. 2.132).

На рис. 2.133 показано, как будет выглядеть оператор вычисления производной высокого порядка в рабочем документе Mathcad.

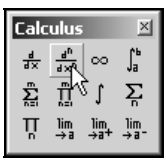


Рис. 2.132. Пиктограмма ввода оператора вычисления производной высокого порядка на палитре **Calculus**

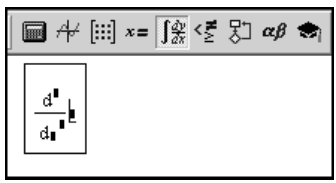


Рис. 2.133. Оператор вычисления производной высокого порядка

По сравнению с оператором вычисления производной первого порядка, в данном случае есть еще два дополнительных структурных заполнителя — однако число вводится только в один из них, а второй при этом заполняется автоматически. Как несложно догадаться, вводимое число — это порядок производной. В документе на рис. 2.134 приведены примеры вычисления производных различных порядков.

Наконец, если нужно вычислить производные для набора значений переменной, то разумно в качестве значения этой переменной присвоить диапазон. Все остальные команды, по большому счету, изменений не претерпевают. Пример приведен на рис. 2.135.

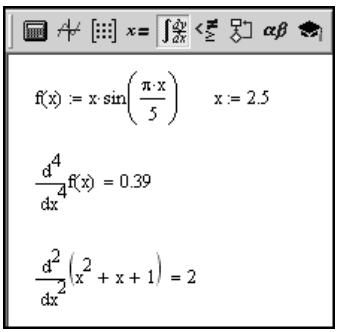


Рис. 2.134. Вычисление производных различных порядков

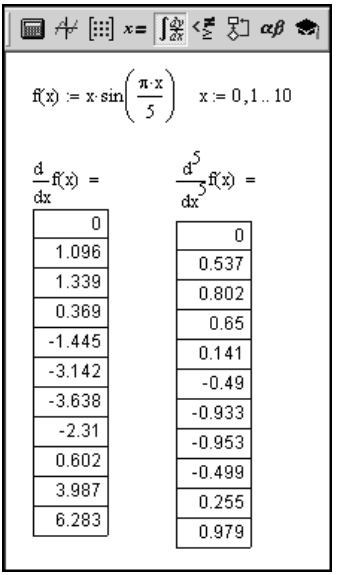


Рис. 2.135. Вычисление производных для диапазона значений

Ранее были рассмотрены примеры работы с функцией, зависящей от одной переменной. Если функция имеет несколько аргументов, производная вычисляется точно так же, следует только корректно указывать тот аргумент, по которому проводится дифференцирование, и для всех аргументов должны быть заданы значения.

Вычисление интегралов

Чтобы вычислить определенный интеграл (в численном виде), существует специальный оператор, который в рабочем документе выглядит абсолютно так же, как стандартное математическое обозначение для определенного интеграла. Вставка соответствующего оператора в рабочий документ Mathcad осуществляется нажатием комбинации клавиш <Shift>+<7>. Можно, разумеется, воспользоваться пиктограммой на палитре **Calculus**, благо пиктограмм на ней хватает на все случаи жизни (рис. 2.136).

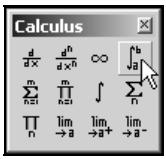


Рис. 2.136. Вставка оператора вычисления определенного интеграла

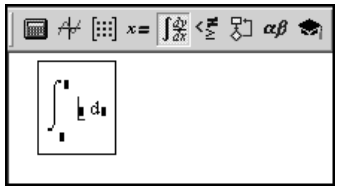


Рис. 2.137. Оператор вычисления определенного интеграла

В рабочем документе оператор с незаполненными структурными заполнителями выглядит так, как показано на рис. 2.137.

В операторе, как несложно заметить, четыре структурных заполнителя: два для ввода верхней и нижней границ интегрирования (под и над символом интеграла), один заполнитель для ввода подынтегральной функции или выражения, и еще один для указания переменной интегрирования (после символа дифференциала).

Пример 2.51. Вычисление интеграла

Примеры вычисления определенных интегралов представлены в документе на рис. 2.138.

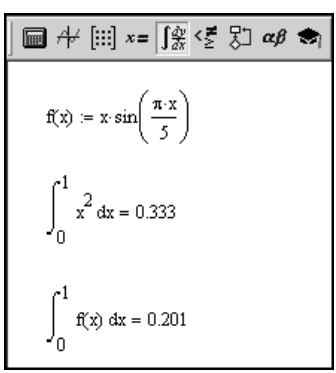


Рис. 2.138. Вычисление определенных интегралов

Следует отметить, что определенные интегралы можно вычислять и в символьном виде. Соответствующие методы описаны в *главе 5*.

Полезные советы и возможности

Существует ряд полезных возможностей, которые существенно упрощают работу с приложением. Они связаны с самыми разными аспектами работы с Mathcad. В этом разделе некоторым из них и будет уделено внимание.

Редактирование выражений

В первую очередь следует отметить, что методы ввода и редактирования математических выражений и команд в Mathcad имеют существенные особенности по сравнению с другими похожими и не очень похожими приложениями. Иногда бывает достаточно сложно предугадать поведение системы при нажатии той или иной клавиши, поэтому далее перечислены некоторые советы, которые помогут читателю лучше понять встроенные алгоритмы работы системы в части редактирования кода.

- При вставке оператора он размещается после того выражения (или подвыражения), которое выделено линиями редактирования, если выражение находится слева от вертикальной линии редактирования. Если оно находится справа от вертикальной линии редактирования, то оператор вставляется перед выражением. При необходимости автоматически добавляются скобки.
- Перемещать вертикальную линию редактирования из начала выражения в конец и обратно можно с помощью нажатия клавиши <Ins>.
- При удалении бинарных операторов вместо них часто отображается метка в виде прямоугольника. Она свидетельствует о том, что в данном месте должен быть введен оператор.
- Для заключения выражения или подвыражения в скобки, следует нажать клавишу <’>. В этом случае все, что было выделено линиями редактирования, заключается в скобки.
- Следует иметь в виду, что при удалении одной скобки, автоматически удаляется и другая.
- Чтобы указать введенное в рабочий документ выражение аргументом функции, можно поступить следующим образом: выражение выделяется и заключается в скобки (клавиша <’>), вертикальная линия редактирования переводится в положение слева от выражения (можно нажать для этого клавишу <Ins>), после чего вводится название функции.

Использование нестандартных названий

В названиях переменных в Mathcad допустимы довольно специфические символы. И это совсем не блажь. Ведь обычно в Mathcad приходится решать физические или инженерные задачи, и в этом случае важно, чтобы названия переменных и функций наиболее полно соответствовали физическому смыслу соответствующих характеристик. Поэтому в названии сложно ограничиться исключительно стандартными символами. К стандартным в данном случае относятся все буквы (строчные и прописные, в том числе и греческие), цифры, нижние индексы, символ подчеркивания и процента, символ бесконечности (комбинация клавиш <Ctrl>+<Shift>+<Z>) и штрих. При этом следует иметь в виду, что, например,

символ бесконечности может быть только первым символом в названии. В Mathcad названия, набранные в разных регистрах, различимы. Другими словами, одна и та же строчная и прописная буква в Mathcad интерпретируется как два разных символа. Мало того, даже одни и те же названия, но выделенные разным стилем, интерпретируются как разные переменные.

Пример 2.52. Операторы в названиях

Чтобы инициализировать название, в котором формально присутствуют операторы, следует выполнить два шага.

1. Нажать комбинацию клавиш <Ctrl>+<Shift>+<J>, в результате чего в рабочий документ будет вставлена пара квадратных скобок со структурным заполнителем между ними (рис. 2.139).
2. В эти квадратные скобки ввести названия, содержащие самые разнообразные операторы. Небольшой пример представлен на рис. 2.140.

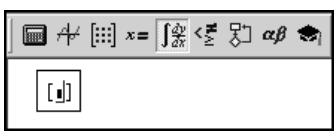


Рис. 2.139. Скобки, в которых вводится название переменной

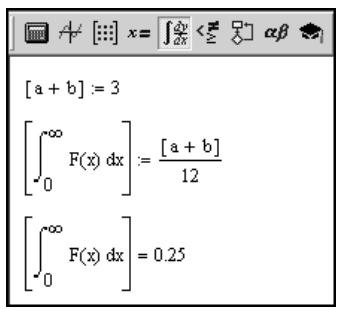


Рис. 2.140. Использование операторов в названиях переменных

Само собой разумеется, что таким же образом можно задавать и названия функций.

Пример 2.53. Символы-операторы

Есть еще один способ, позволяющий привлекать для формирования названий переменных и функций символы операторов. Обычно речь идет о бинарных операторах вроде сложения или умножения. Дело в том, что если просто ввести эти операторы в название, то автоматически будет предпринята попытка выполнить соответствующую операцию, что в рассматриваемом контексте крайне нежелательно. Чтобы этого не случилось, переходят в специальный режим, при котором операторы при вводе интерпретируются как символы. Для перехода в этот режим нажимают комбинацию клавиш <Ctrl>+<Shift>+<K>. При этом линии редактирования из синих становятся красными. И это всего-навсего признак перехода в указанный режим — ничего более. Затем вводится нужное

название и осуществляется обратный переход в нормальный режим: следует снова нажать комбинацию клавиш <Ctrl>+<Shift>+<K>. Линии редактирования восстанавливают исходную окраску. Примеры применения описанного режима можно найти на рис. 2.141.

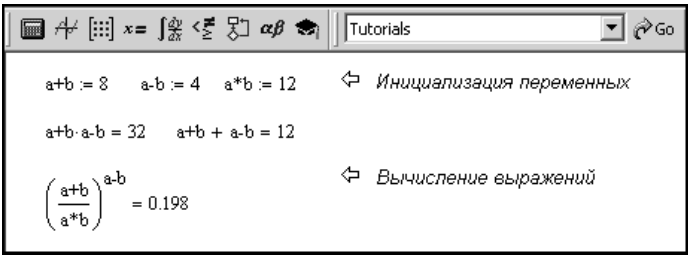


Рис. 2.141. Примеры использования названий с символами операторов

Несмотря на наличие таких экзотических возможностей Mathcad, все же хочется предостеречь читателя от чрезмерного увлечения ими — экзотика не должна становиться самоцелью.

Определение операторов пользователя

В заключение остановимся еще на одной существенной возможности Mathcad. Она состоит в том, что пользователь может определять собственные унарные и бинарные операторы. Определяются они практически так же, как и обычные функции в Mathcad. Правда, свои особенности все же имеются. В частности, для определения оператора для него сначала следует ввести символ. Далее сразу после символа создаваемого оператора в скобках указываются названия для операнда (для унарного оператора) или операндов (двух — для бинарного оператора). Затем после оператора присваивания вводится выражение, определяющее действие оператора на операнд или операнды.

Пример 2.54. Операторы пользователя

В документе на рис. 2.142 можно найти примеры определения унарного и бинарного операторов, а также возможные способы их применения в вычислениях.

Бинарным оператором, для которого выбран символ *звездочка* (*), вычисляется корень квадратный из суммы квадратов операндов. Унарный оператор, для которого выбран символ *шляпка* (^), действует на операнд согласно правилу: вычисляется сумма единицы и единицы, деленной на операнд, после чего все это возводится в степень, определяемую операндом. Таким образом, при увеличении операнда результат действия оператора должен приближаться к экспоненте ($e \approx 2.72$).

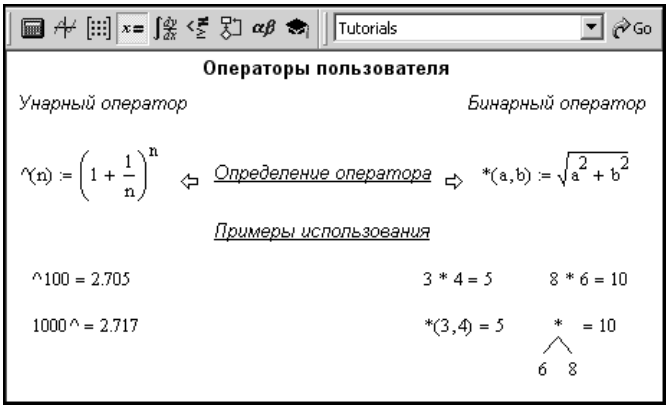


Рис. 2.142. Примеры определения операторов пользователя

Упомянутые выше символы (* и ^) следует вводить, перейдя предварительно в специальный режим ввода названий с символами. Он обсуждался ранее, и такой переход осуществляется нажатием комбинации клавиш <Ctrl>+<Shift>+<K>. Это же относится и к последующему использованию операторов в выражениях и командах рабочего документа.

Бинарный оператор при вводе команд можно вводить практически как обычный оператор, однако лучше через палитру **Evaluation**. Унарный оператор может указываться перед операндом или после него. В первом случае выбирают пиктограмму с изображением литер **f** и **x** (именно в таком порядке) на палитре **Evaluation** (рис. 2.143).

В результате в рабочем документе появится структура из двух заполнителей (рис. 2.144).

Рис. 2.143. Пиктограмма вставки оператора перед операндом

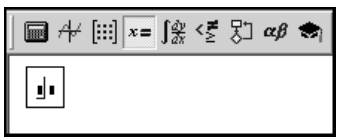


Рис. 2.144. Ввод унарного оператора перед операндом

Вместо первого заполнителя вводится символ оператора, а вместо второго — операнд. Практически такая же ситуация складывается, когда унарный оператор вводится после операнда (результат от этого не меняется). Следует, разумеется, выбирать другую пиктограмму на палитре **Evaluation** (на ней буква **x** следует перед буквой **f**) (рис. 2.145).

На первый взгляд, в документ при этом вставляется такая же конструкция, как и в предыдущем случае (рис. 2.146). Однако в заблуждение впадать не следует, поскольку здесь сначала вводится операнд, а после этого — оператор.

Для бинарных операторов также предназначены две пиктограммы. Часто удобно вводить бинарный оператор как стандартный встроенный оператор Mathcad, вроде умножения или сложения. В этом случае используют пиктограмму с тремя литерами **xfy** (рис. 2.147).

Теперь предстоит заполнить три позиции: первый операнд, оператор и второй операнд (рис. 2.148).

Наконец, команда с бинарным оператором может быть введена несколько иначе: в виде ветвящейся структуры с бинарным оператором в вершине и нижними ветвями, указывающими на операнды. Соответствующая пиктограмма на палитре **Evaluation** так же, как и в предыдущем случае, содержит три буквы **xfy**, однако литера **f** несколько смещена вверх (рис. 2.149).

Какой вид имеет вставляемая при этом в рабочий документ структура, ясно из рис. 2.150.

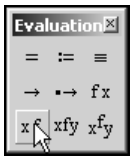


Рис. 2.145. Пиктограмма вставки оператора после операнда

Рис. 2.146. Ввод унарного оператора после операнда

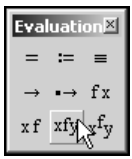


Рис. 2.147. Пиктограмма вставки бинарного оператора

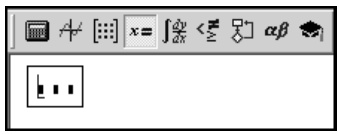


Рис. 2.148. Ввод бинарного оператора

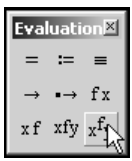


Рис. 2.149. Пиктограмма вставки бинарного оператора в виде ветвистой структуры

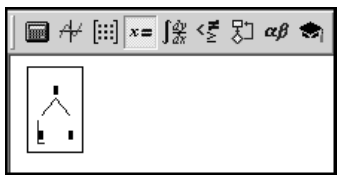


Рис. 2.150. Ввод бинарного оператора с ветвистой структурой

В нижние заполнители вводятся операнды, а верхний предназначен для ввода оператора. Кроме этого, бинарный оператор можно вызывать как обычную функцию: указывается название оператора, а после этого в скобках приводятся аргументы. Все это нашло свое отражение в примерах на рис. 2.142.

Глава 3

Графика и визуализация результатов в Mathcad

Очень важный момент в любом исследовании, помимо выполнения вычислений, — визуализация получаемых результатов. Часто с этой целью используются графики, диаграммы, схемы и т. п. Для решения подобных задач в Mathcad предусмотрены специальные утилиты, которые обсуждаются в данной главе. Если точнее, то речь пойдет о работе с графикой, создании анимации и выполнении других манипуляций, имеющих отношение к представлению данных в рабочих документах.

Создание двумерных графиков

Наиболее просто, безусловно, создаются двумерные графики. Обычно в этом случае подразумевают отображение в графическом виде зависимости функции одной переменной от ее аргумента. Для вставки в рабочий документ двумерного графика вызывают команду **Insert | Graph | X-Y Plot** (рис. 3.1).

В результате в рабочий документ будет вставлена область графика, содержащая несколько структурных заполнителей (рис. 3.2).

Необходимо вместо двух основных заполнителей ввести переменную (заполнитель в нижней части будущего графика) и название функции (или выражение), для которой (которого) строится график — для этого предназначен заполнитель в левой части области графика. Функция определяется заранее в рабочем документе, или это может быть встроенная функция Mathcad. В любом случае, ее аргументом указывается переменная, введенная вместо заполнителя под нижней горизонтальной рамкой, ограничивающей область графика. На рис. 3.3 в качестве переменной указано `x`, а функцией, график которой отображается, является синус (соответственно заполнитель слева от рамки графика заменен на выражение `sin(x)`).

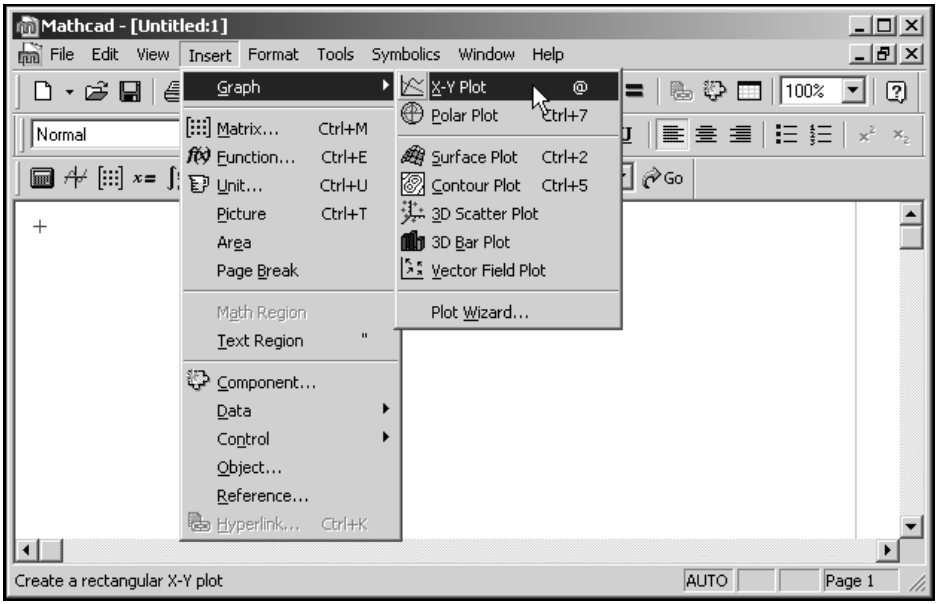


Рис. 3.1. Вызов команды вставки двумерного графика в рабочий документ

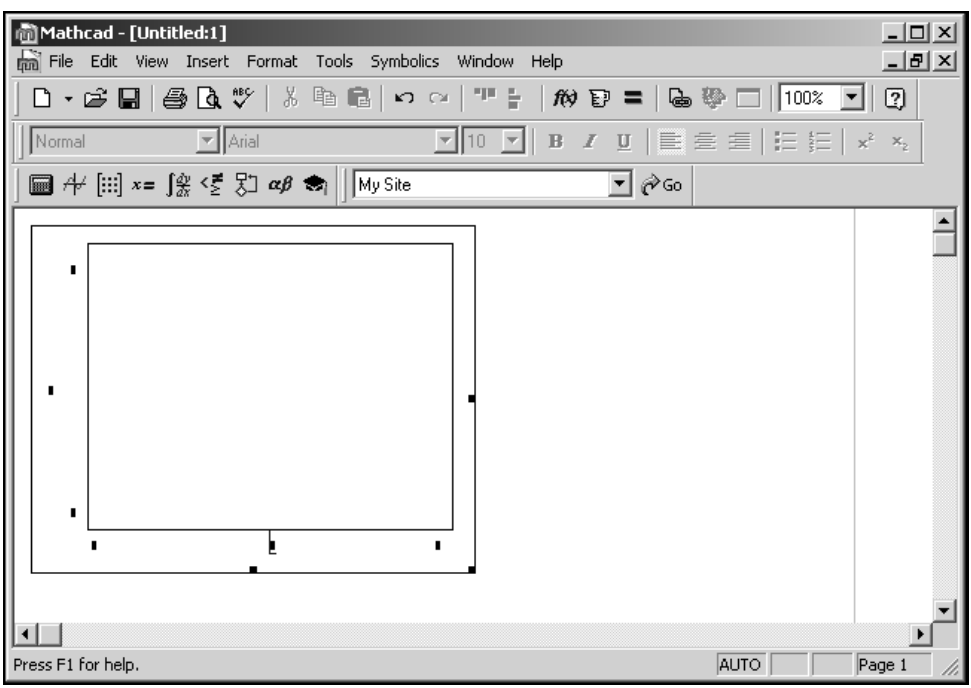


Рис. 3.2. Область отображения графика в рабочем документе

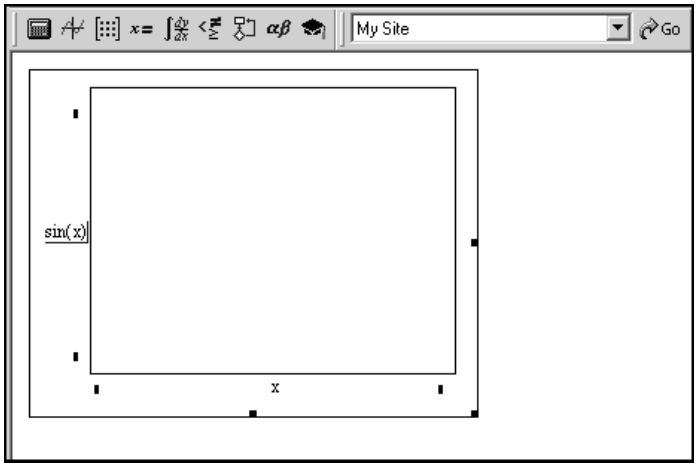


Рис. 3.3. Ввод переменной и функции

Если границы изменения переменной не указаны, то график по умолчанию строится для области изменения переменной в диапазоне от –10 до 10. Это значит, что если после ввода названия переменной и функции щелкнуть мышью где-нибудь в рабочем документе вне области графика, то будет построен график синусоиды, который можно видеть на рис. 3.4.

Шкала оси ординат автоматически масштабируется так, чтобы график функции полностью помещался в области отображения, — от минимального до максимального значения функции.

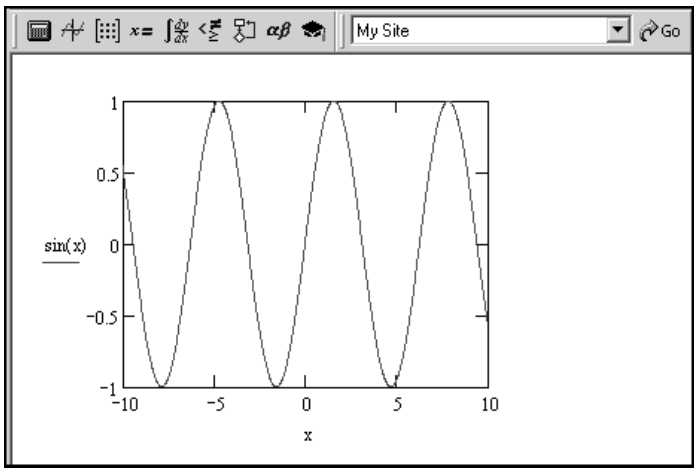


Рис. 3.4. График синусоиды

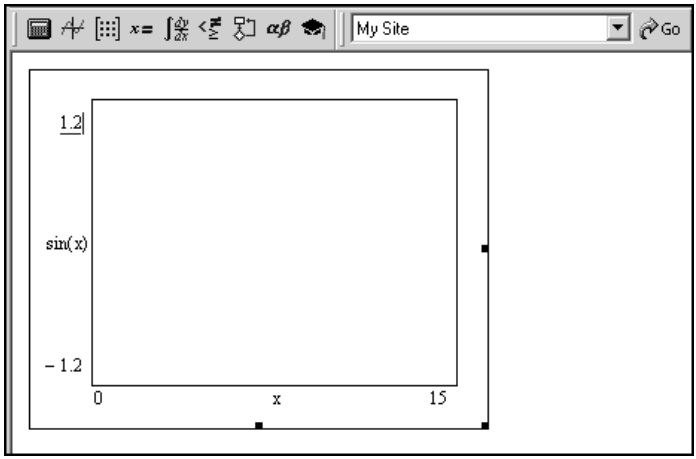


Рис. 3.5. Определение границ изменения переменной и диапазона отображаемых значений функции

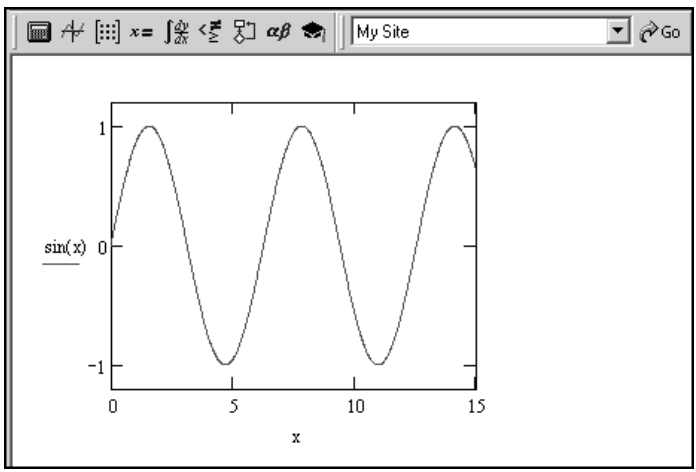


Рис. 3.6. Результат внесения изменений в график

С одной стороны, то, что границы изменения переменной можно не указывать — удобно. С другой стороны, важно иметь возможность самому определять не только область изменения переменной, но и область отображения графика (последняя как раз и задается с помощью минимального и максимального значений вдоль вертикальной координатной оси). Было бы удивительно, если в Mathcad такая возможность не предусматривалась. Более того, соответствующие границы задаются исключительно просто. Для этого достаточно выделить график и ввести необходимые численные значения вместо

заполнителей, расположенных у углов возле соответствующих координатных осей. Если график уже создан и осуществляется его редактирование, то границы изменения функции вводятся вместо тех численных значений, что на данный момент отображены на графике как граничные значения переменной и функции. На рис. 3.5 область изменения переменной задана от 0 до 15, а вертикальная координатная ось охватывает диапазон значений от –1,2 до 1,2.

После внесенных изменений график будет выглядеть так, как показано на рис. 3.6.

Пример 3.1. График функции

Иной способ определения границ изменения переменной состоит в том, чтобы предварительно инициализировать ее со значением-диапазоном. Тогда при построении графика достаточно указать название переменной, а область ее значений будет определена автоматически. Такая ситуация проиллюстрирована на рис. 3.7.

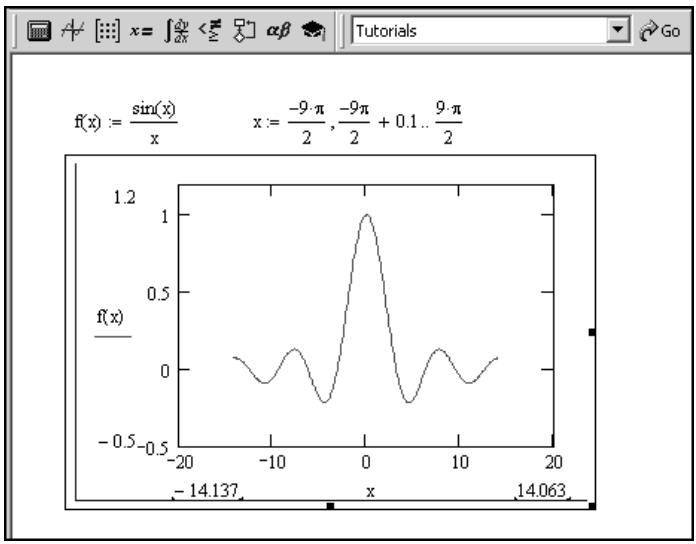


Рис. 3.7. Использование переменной со значением-диапазоном

Сначала определяется функция, равная отношению синуса к его аргументу (т. е. $f(x) = \sin(x)/x$), а переменная-аргумент инициализируется со значением-диапазоном. Здесь она изменяется от $-9\pi/2$ до $9\pi/2$ с шагом 0,1. Далее, при создании графика, помимо функции, для которой строится график, указывается только название переменной, а граничные значения переменной не вводятся. Как несложно заметить на рис. 3.7, в результате график строится для значений переменной, которые определены при ее инициализации. Однако

область отображения вдоль горизонтальной оси несколько превышает необходимый интервал. В данном случае область отображения по горизонтали лежит в пределах от –20 до 20. Граничные значения диапазона изменения переменной, для которого строится график, отображаются под горизонтальной координатной осью при выделении графика (см. рис. 3.7).

Пример 3.2. Графики нескольких функций

Типична ситуация, когда на одном рисунке необходимо отобразить графики нескольких функций одновременно. В этом случае процедура создания графиков, по сравнению с рассмотренной ранее, претерпевает минимальные изменения. В частности, нужно после ввода названия функции и ее аргумента, в левой части области отображения графика нажать клавишу <,>. Запятая в результате не появится, но зато внизу под названием уже введенной функции размещается новый структурный заполнитель, вместо которого и вводится название дополнительной функции (рис. 3.8).

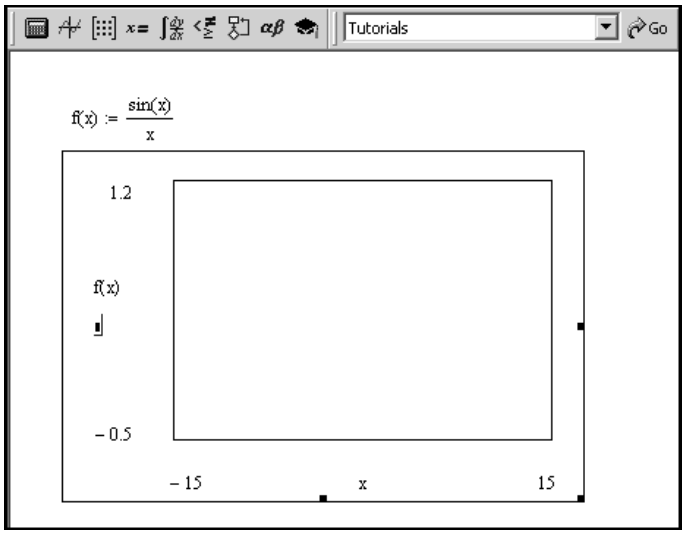


Рис. 3.8. Ввод названия дополнительной функции для отображения ее графика

После ввода выражения или названия функции следует, как обычно, щелкнуть мышью вне области рисунка или нажать клавишу <Enter>. На рис. 3.9 показан результат отображения графиков сразу двух функций. Первая задается названием (она определена заранее), а вторая — выражением.

В принципе, для функций можно задавать разные аргументы. Такая ситуация показана на рис. 3.10. Там аргументом заранее описанной функции указана переменная z, а второе выражение, для которого строится график, записано через переменную x.

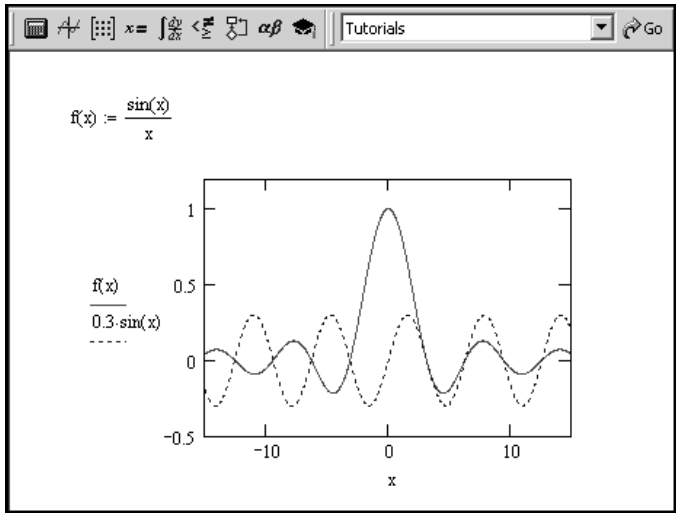


Рис. 3.9. Отображение графиков двух функций

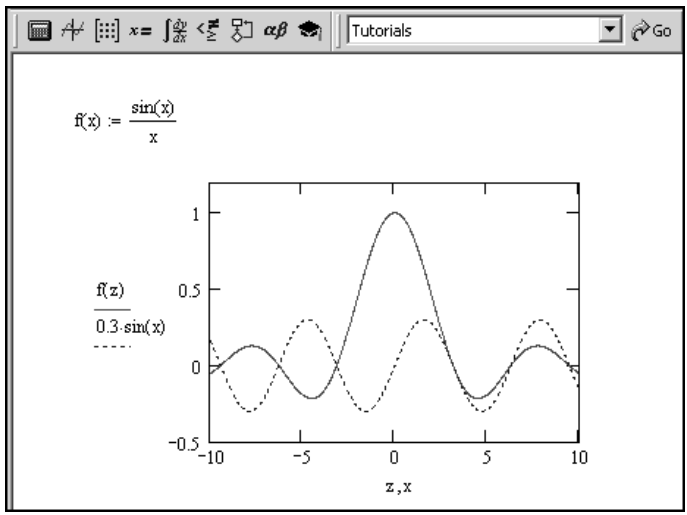


Рис. 3.10. Использование разных аргументов функций

Если этим переменным предварительно не присвоить значения и не указать в явном виде границы изменения, то по умолчанию каждая из них будет изменяться от –10 до 10, что и видно на рис. 3.10. При этом должно соблюдаться строгое соответствие: порядок следования названий переменных справа налево должен соответствовать порядку следования функций или выражений (сверху вниз), для которых отображаются графики. При вводе названий переменных у горизонтальной координатной оси они разделяются запятыми.

Если в явном виде задать минимальное и максимальное значения вдоль горизонтальной координатной оси, то они будут автоматически интерпретироваться как границы изменения аргументов графически отображаемых функций. Пример такой ситуации приведен на рис. 3.11. Там диапазон изменения обеих переменных указан от –20 до 20.

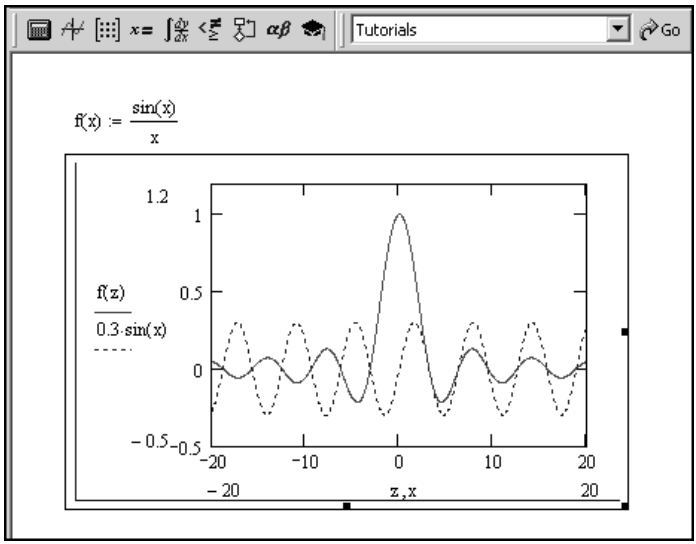


Рис. 3.11. Явное определение границ изменения аргументов функций

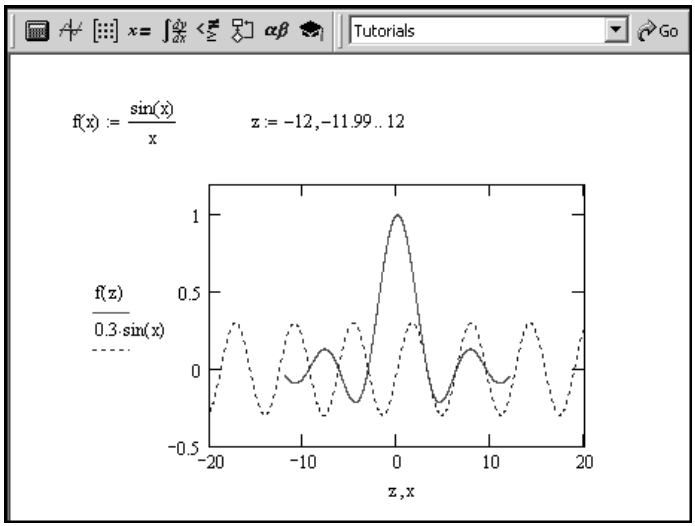


Рис. 3.12. Разные границы изменения аргументов

Главное удобство, связанное с возможностью указания разных аргументов для отображаемых на одном рисунке функциональных зависимостей, состоит в том, что для разных переменных можно задавать разные границы изменения. Например, если предварительно переменной z присвоить диапазонное значение в пределах от –12 до 12 с шагом 0,01, а явно указанные границы вдоль горизонтальной координатной оси оставить без изменения, т. е. от –20 до 20, то график первой функции будет построен для значений аргумента от –12 до 12, а для второго выражения — для значений от –20 до 20 (рис. 3.12).

Обычно такой подход удобен в тех случаях, когда отображаемые в графическом виде функции имеют разные области определения.

Настройка параметров графика

Как несложно заметить из предыдущих примеров, при выводе дополнительного графика изменяется тип линии, с помощью которой он отображается. Меняется также и ее цвет — с красного (для первого графика) на синий (для второго). Если одновременно отображать больше двух графиков, ситуация будет аналогичной: новые графики строятся с помощью линий нового типа и цвета (хотя, разумеется, такая новизна рано или поздно заканчивается). Иногда возникает желание, а то и просто необходимость, взять весь этот процесс под свой контроль. Имеется в виду возможность выбирать тип, цвет и толщину линий, посредством которых и отображаются графики функций. Кроме того, общий вид создаваемого рисунка далек от совершенства, поэтому часто приходится повышать его презентабельность, что называется, собственноручно.

Вносить изменения в настройки параметров графика (или графиков) в документе Mathcad можно несколькими способами. Для определенности будем предполагать, что рисунок с необходимым графиком (графиками) уже создан и далее его нужно немного отредактировать. Для этого следует дважды щелкнуть мышью на этом рисунке. В результате откроется диалоговое окно **Formatting Currently Selected X-Y Plot** (рис. 3.13).

Это же окно можно открыть, выбрав команду **Format | Graph | X-Y Plot**. При этом должен быть выделен редактируемый рисунок. Если нет, то открывшееся окно будет несколько отличаться от описываемого здесь.

Диалоговое окно **Formatting Currently Selected X-Y Plot** содержит четыре вкладки (**X-Y Axes**, **Traces**, **Labels** и **Defaults**), наиболее важны из которых, пожалуй, первые две. На вкладке **X-Y Axes** выполняются настройки, связанные, в основном, с видом и типом координатной системы, используемой при отображении графиков. Там содержится три раздела: формально эквивалентные **X-Axis** и **Primary Y Axis** (**Secondary Y Axis**), расположенные в верхней части диалогового окна и содержащие опции настройки основных параметров

координатной системы, а также раздел **Axis Style** в нижней части диалогового окна, в котором выбирается, с помощью группы переключателей, тип координатной системы. Опции в разделах **X-Axis** и **Primary Y Axis** (**Secondary Y Axis**) абсолютно идентичны, а разница состоит лишь в том, что выполняемые настройки применяются к горизонтальной и вертикальной (основной и вспомогательной) координатным осям соответственно. В частности, установив флажок опции **Log scale**, можно перейти к логарифмической шкале. Это можно делать только в том случае, если значения вдоль оси, для которой применяется логарифмический масштаб, принимают положительные значения.

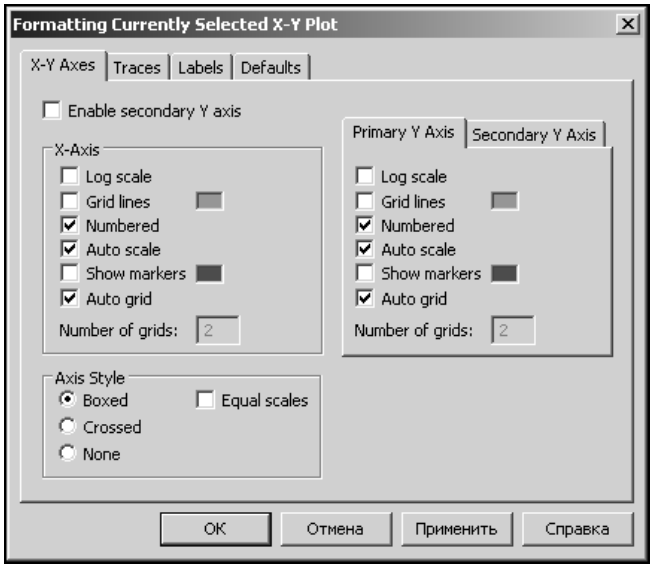


Рис. 3.13. Диалоговое окно **Formatting Currently Selected X-Y Plot** открыто на вкладке **X-Y Axes**

Опция **Grid lines** активизируется в том случае, если нужно отобразить координатную сетку. Цвет выбирается после щелчка на цветной кнопке палитры справа от опции. Результат применения этого режима к созданному ранее рисунку с графиками двух функций показан на рис. 3.14.

Координатная сетка здесь отображается для обеих осей, т. е. в диалоговом окне **Formatting Currently Selected X-Y Plot** флажок поставлен у каждой из двух опций **Grid lines**.

Опция **Numbered** отвечает за нумерацию меток вдоль координатных осей. Если флажок этой опции не выставлен, после отмены выделения рисунка численные значения вдоль меток делений координатных осей отображаться не будут. По умолчанию при создании графика активизированы обе опции **Autoscale**.

В результате вдоль каждой из координатных осей масштабирование осуществляется автоматически. Чтобы отключить этот режим, достаточно убрать флажки упомянутых опций. Полезна в практическом плане опция **Show markers**. По умолчанию она неактивна ни для одной из осей. Если флажок у опции поставить, то появится возможность отображать на графике дополнительные горизонтальные или вертикальные линии. После активизации этой опции для обеих осей при выделении рисунка с графиками появляется по два структурных заполнителя возле каждой из координатных осей (рис. 3.15).

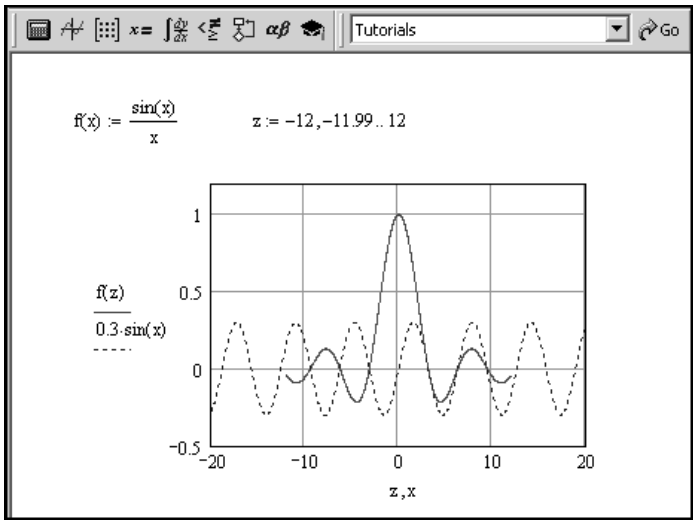


Рис. 3.14. Отображение координатной сетки

Вместо этих заполнителей вводится число, являющееся горизонтальной или вертикальной координатой для проводимой маркировочной линии — в соответствии с тем, у какой оси размещен структурный заполнитель. Нет необходимости вводить значения для каждого из четырех заполнителей. Отображается ровно столько вспомогательных линий, сколько нужно (правда, не более четырех). На рис. 3.16 показана ситуация, когда отображается одна горизонтальная линия с координатой 12 и одна вертикальная с координатой 0.8.

Результат представлен на рис. 3.17. Видно, что кроме двух ранее созданных графиков, на рисунке отображены еще две линии, возле которых (верхняя правая часть рисунка) приведены соответствующие численные значения — координаты маркировочных линий.

Если в дальнейшем понадобится убрать одну из маркировочных линий, следует в области отображения графика выделить численное значение координаты, соответствующее удаляемой линии, как это показано на рис. 3.18, и удалить его.

Наконец, если установлен флажок опции **Auto grid**, то число меток вдоль координатных осей системой выбирается автоматически. Если флажок опции не установлен, пользователю придется самостоятельно указать в поле **Number of grids**, на какое число интервалов нужно разбить весь диапазон значений вдоль соответствующей координатной оси.

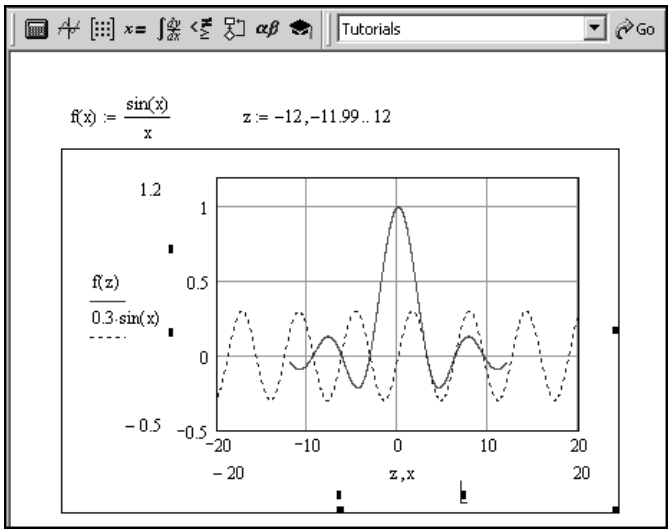


Рис. 3.15. Дополнительные структурные заполнители появляются после активизации опции **Show markers**

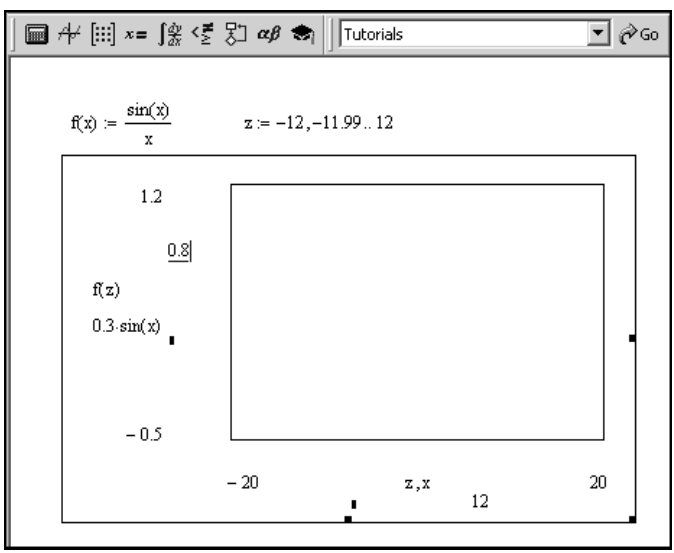


Рис. 3.16. Ввод численных значений, определяющих координаты маркировочных линий

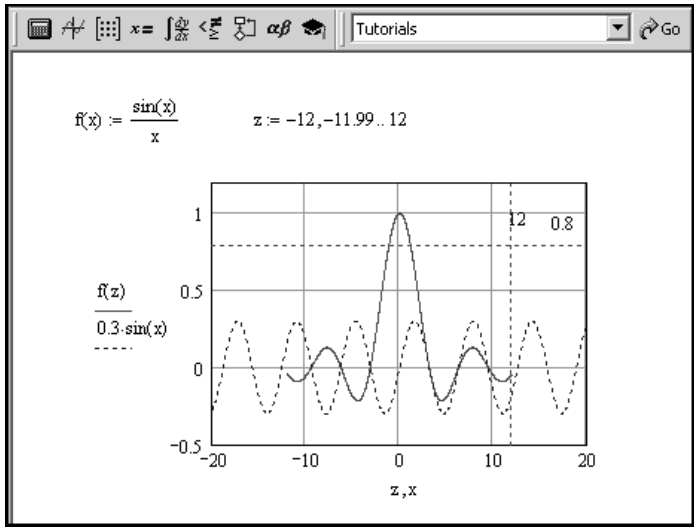


Рис. 3.17. Отображение маркировочных линий

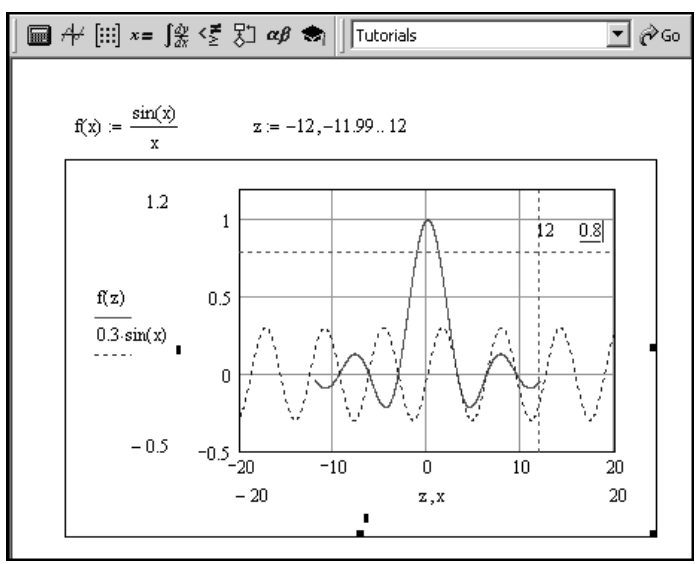


Рис. 3.18. Удаление маркировочной линии

Раздел **Axis Style** содержит группу из трех переключателей (**Boxed**, **Crossed** и **None**) и опцию **Equal scales**. Выбирая один из переключателей, можно задать тип координатной системы. Переключатель **Boxed** выбран по умолчанию и соответствует координатной системе, когда вся область отображения графики заключена в прямоугольную рамку. Если переключатель установлен

в положение **Crossed**, координатные оси пересекаются в начале координат. Пример такой системы координат приведен на рис. 3.19.

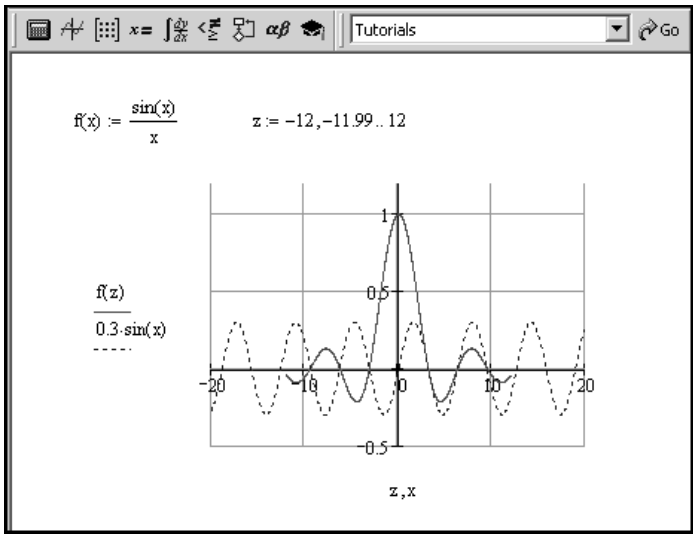


Рис. 3.19. Переключатель выбора координатной системы установлен в положение **Crossed**

Можно вообще не отображать координатную систему, для чего достаточно установить переключатель в положение **None**. А если нужно, чтобы масштаб вдоль горизонтальной и вертикальной осей был одинаков, устанавливают флажок опции **Equal scales**.

Комбинируя различные настройки, можно добиваться желаемого эффекта при работе с графиками. На рис. 3.20 показан вид диалогового окна **Formatting Currently Selected X-Y Plot** (вкладка **X-Y Axes**) со следующими настройками: по каждой из осей отображается координатная сетка, нумеруются метки и автоматически выполняется масштабирование. Число интервалов для отображения линий координатной сетки равно 5, система координат с пересечением координатных осей в начальной точке и одинаковым масштабом по обеим осям.

На этом же рисунке показан курсор мыши, размещенный в области над кнопкой **Применить**, а не **OK**. Если щелкнуть на этой кнопке, то внесенные изменения вступят в силу, диалоговое окно закрыто не будет (в отличие от случая, когда щелкают на кнопке **OK**), но при этом можно видеть результат применения настроек, который представлен на рис. 3.21. Для удобства существенно сужена область изменения переменных-аргументов отображаемых функций.

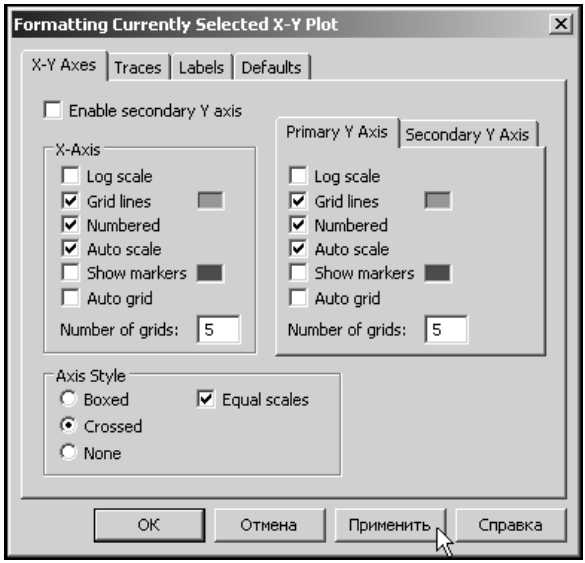


Рис. 3.20. Настройка параметров графика

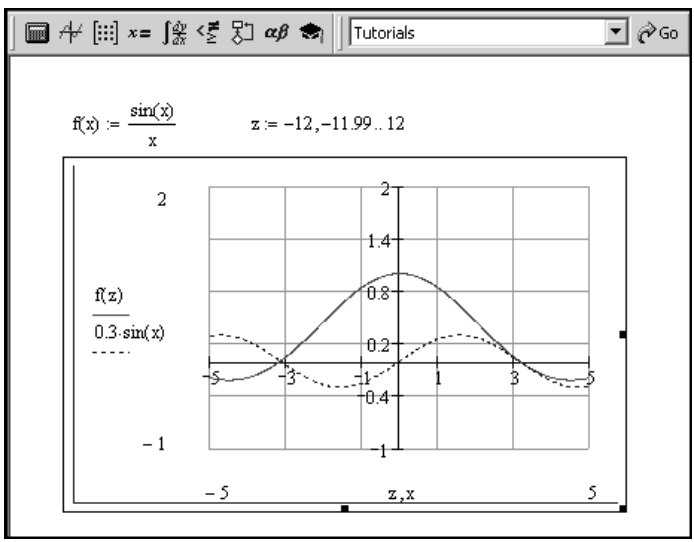


Рис. 3.21. Результат применения настроек

Как упоминалось ранее, область настройки типа и цвета вертикальной координатной оси содержит две идентичные вкладки: **Primary Y Axis** и **Secondary Y Axis** (см. рис. 3.20). Для того чтобы вспомогательная координатная ось была доступна, следует установить флажок опции **Enable secondary Y axis** в верхней части вкладки **X-Y Axes** (см. рис. 3.20). В этом

случае в правой части графика появится структурный заполнитель, куда вводят имя и аргумент функции для построения графика. Там же (т. е. справа) отображается еще одна координатная ось с индивидуальными настройками, аналогичными тем, что описывались выше. Иногда это бывает удобно.

Настройки, относящиеся непосредственно к способу отображения функциональных зависимостей, выполняются на вкладке **Traces** диалогового окна **Formatting Currently Selected X-Y Plot**. Данное диалоговое окно, открытое на вкладке **Traces**, представлено на рис. 3.22.

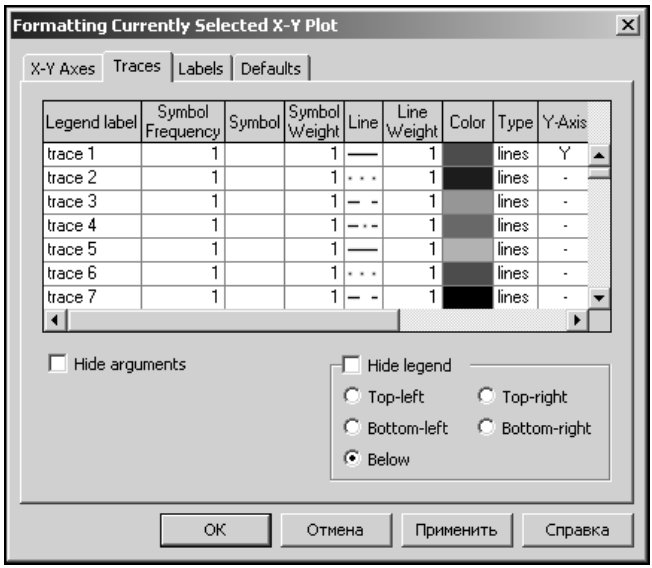


Рис. 3.22. Диалоговое окно **Formatting Currently Selected X-Y Plot** открыто на вкладке **Traces**

Главный элемент на этой вкладке — список в центральной части диалогового окна. Элементы списка разбиты на девять колонок, для которых приводятся названия. В колонке **Legend label** отображаются метки для легенд графиков, которые одновременно являются и индикаторами графиков функций. По умолчанию они формируются объединением слова **trace** и порядкового номера графика. Всего в списке шестнадцать элементов, а это значит, что одновременно на одном рисунке можно отображать до шестнадцати различных графиков. Следующие две колонки имеют названия **Symbol Frequency** и **Symbol**. В колонке **Symbol** указывается тип базового символа, с помощью которого формируются кривые для графиков зависимостей (по умолчанию в этой колонке отображается значение **none**), а в поле колонки **Symbol Frequency** указывается плотность (частота) его использования (отображения

на кривой). Размеры символов задаются в полях колонки **Symbol Weight**. Колонка **Line** содержит сведения о том, какого характера линия используется для отображения графика, а ее толщина устанавливается в полях колонки **Line Weight**. Для разных графиков используются разные линии. Поскольку линии могут быть всего четырех типов (сплошная, пунктирная, штрихованная и штрихпунктирная), то они применяются поочередно и периодически. В силу этих причин становится очевидным, что только с помощью типа линии можно выделить лишь четыре графика. Если графиков на одном рисунке больше четырех, то для них обычно задают еще и разные цвета. Цвет (для данной линии) отображается в колонке с названием **Color**.

Однако совсем необязательно отображать график (или графики) с помощью линий. Способ отображения графической зависимости можно выбрать в колонке **Type**. Наконец, последняя колонка **Y-Axis** определяет тип вертикальной координатной оси. Под каждой колонкой расположено поле или раскрывающийся список, в которое или из которого соответственно вводится или выбирается значение. Между колонками существует взаимосвязь — доступность элементов из раскрывающихся списков значений свойства зависит от выбранных значений для прочих свойств.

Название-легенда для графика отображается в той части рисунка, что определяется положением переключателя в разделе **Hide legend**. Чтобы группа переключателей была доступна, у соответствующей опции должен отсутствовать флажок. Соответствие между расположением легенды и состоянием переключателей такое:

- **Top-left** — слева вверху;
- **Bottom-left** — слева внизу;
- **Below** — внизу;
- **Top-right** — справа вверху;
- **Bottom-right** — справа внизу.

Наличие флажка опции **Hide arguments** приводит к тому, что на рисунке будут скрыты названия аргумента (или аргументов) и функций вдоль координатных осей (однако при выделении рисунка они видны).

Возвращаясь к выбору базового символа, следует отметить, что соответствующий раскрывающийся список с возможными значениями доступен только в том случае, если значением в колонке **Type** является **lines**, **points** или **stem**. Полный список значений, которые можно увидеть в колонке **Type**, приведен в табл. 3.1.

Таблица 3.1. *Значения в колонке* ***Type***

Значение	Описание
lines	График зависимости строится с помощью линии: сначала определяются базовые точки эскиза графика, после чего они соединяются линиями
points	Отображаются только базовые точки эскиза. Точки линиями не соединяются
error	Это значение может применяться только для пар графиков. Причем эти два графика должны следовать один за другим в списке графиков вкладки **Traces**. В этом случае соответствующие базовые точки эскизов графиков соединяются отрезками, такими, как при выделении области ошибки измерений для экспериментальных данных. Базовые точки эскиза каждого отдельного графика при этом линиями не соединяются. При выборе этого значения свойства из колонок **Symbol**, **Line**, **Line Weight** и **Symbol Weight** становятся неактивными. Соответствующее значение разумно применять в тех случаях, когда зависимость строится по экспериментальным данным. Один график задает верхнюю границу значений исследуемой величины, а другой — нижнюю
bar	В этом случае графическая зависимость отображается в виде столбиков, подобно тому, как строятся гистограммы. Как и в предыдущем случае, колонки **Symbol**, **Line**, **Line Weight** и **Symbol Weight** не используются
step	График зависимости отображается в виде непрерывной ступенчатой функции. Свойства в колонках **Symbol**, **Line**, **Line Weight** и **Symbol Weight** неактивны
draw	Используется рисованная линия для отображения графиков функциональных зависимостей. В отличие от случая, когда указано значение **lines**, в данной ситуации нет возможности выделять базовые узлы эскиза графика специальными символами
stem	Базовые точки эскиза графика отмечаются символами, выбранными в колонке **Symbol**. Эти точки соединяются вертикальными линиями с горизонтальной координатной осью. Сами базовые точки линией не соединяются (колонка **Line** неактивна)
solidbar	Зависимость отображается с помощью заполненных прилегающих друг к другу столбиков

В зависимости от решаемых задач выбирается способ графического отображения данных. Некоторые из них встречаются чаще, другие — реже. Наиболее примечательными являются способы отображения данных в виде

отдельных точек с выделением их специальными символами. Обычно это делается при работе с экспериментальными данными. Правда, в этом случае необходимы специальные методы, о которых речь пойдет в последующих главах.

Различные графики можно выделять на рисунке также с помощью линий разной толщины или символов разного размера. Соответствующая настройка выполняется в раскрывающихся списках под колонками **Line Weight** и **Symbol Weight**. Списки содержат числа от **1** до **9**, а также значение **p**. Числа задают толщину линий или размер символов, **p** соответствует оптимальной толщине или размеру в один пиксел. Как правило, перечисленных средств вполне хватает для однозначной идентификации практически любого числа графиков, вплоть до максимально возможных шестнадцати.

Пример 3.3. Способы отображения графиков

Выбор разных настроек в разных комбинациях приводит к различным визуальным эффектам. В табл. 3.2 представлены некоторые наиболее характерные ситуации.

Таблица 3.2. *Использование разных настроек*

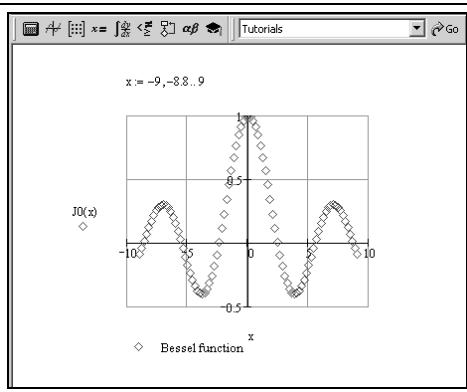
Основные настройки	Результат
На рисунке отображается графическая зависимость для функции Бесселя нулевого индекса (команда `J0(x)`). Переменной-аргументу предварительно присваивается значение-диапазон, в соответствии с чем аргумент функции изменяется от –9 до 9 с шагом дискретности 0,2. В качестве легенды в диалоговом окне **Formatting Currently Selected X-Y Plot** на вкладке **Traces** введено название **Bessel function** для легенды графика (колонка **Legend label**), график отображается в виде отдельных точек (в колонке **Type** выбрано значение **points**), точки представляются в виде ромбов (колонка **Symbol**), а в колонке **Symbol Weight** выбрано значение 2, определяющее размер символов	

Таблица 3.2 *(продолжение)*

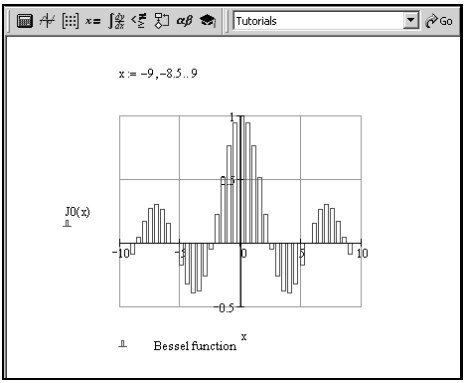
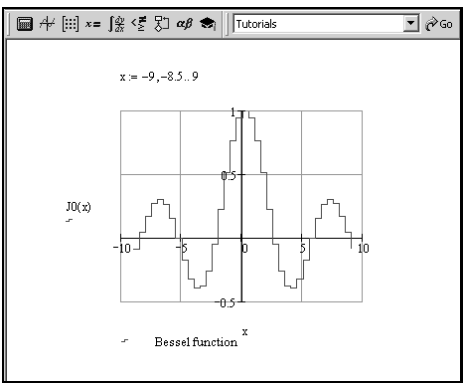
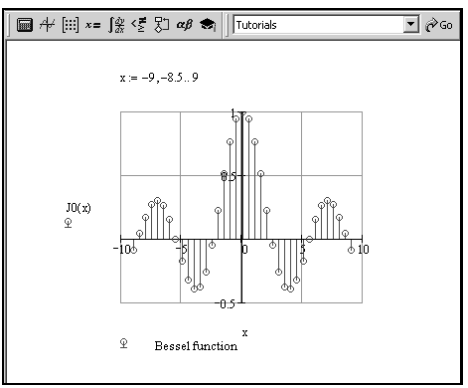
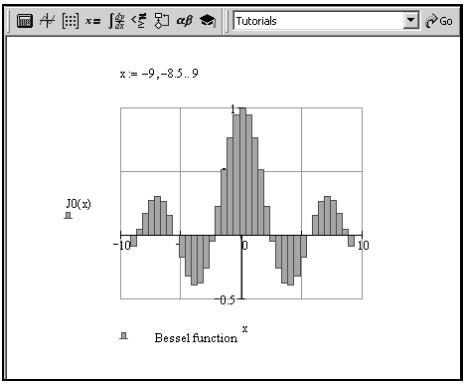
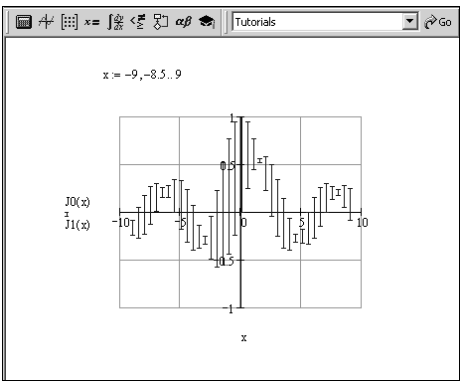
Основные настройки	Результат
Значение в колонке **Type** изменено на **bar**. В результате график функции отображается в виде незаполненных прямоугольных столбиков. Для удобства и большей наглядности увеличен шаг дискретности аргумента функции Бесселя (в данном случае и далее он составляет 0,5)	
Значение в колонке **Type** установлено равным **step**. График функции имеет ступенчатый вид	
Пример применения значения **stem** (колонка **Type**). В колонке **Symbol** изменен тип символа. Поэтому базовые точки отображаются в виде кружков. Центры этих точек соединены линиями с горизонтальной координатной осью. Изменяя значение в колонке **Symbol Weight**, можно варьировать размер символов и толщину линий	

Таблица 3.2 *(окончание)*

Основные настройки	Результат
В колонке **Type** установлено значение **solidbar**. График в этом случае имеет вид закрашенных вплотную размещенных друг к другу прямоугольных столбиков	
Как уже отмечалось, значение **error** в колонке **Type** устанавливается для двух последовательных графиков. В качестве второго дополнительного графика указана функция Бесселя первого индекса (команда `J1(x)`). В результате базовые точки двух графиков соединяются вертикальными линиями с горизонтальными метками по краям	

В диалоговом окне **Formatting Currently Selected X-Y Plot**, помимо перечисленных, есть еще две вкладки. На рис. 3.23 данное диалоговое окно открыто на вкладке **Labels**.

На этой вкладке задается заголовок создаваемого графика, а также надписи у координатных осей. Заголовок вводится в поле **Title**. Однако для того чтобы заголовок отображался на рисунке, необходимо установить флажок опции **Show Title**. Заголовок можно отображать над или под графиком. В первом случае устанавливают переключатель **Above**, а во втором — **Below**.

Надпись для оси абсцисс (горизонтальная координатная ось) вводится в поле **X-Axis**, а для оси ординат (вертикальная координатная ось) — в поле **Y-Axis** (для вспомогательной оси поле **Y2-Axis**). Для того чтобы надпись отображалась, необходимо установить флажок опции, размещенной у соответствующего поля.

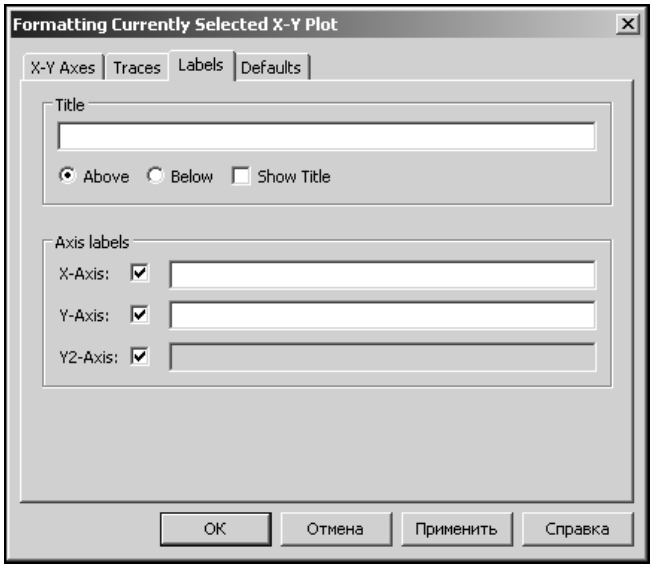


Рис. 3.23. Диалоговое окно **Formatting Currently Selected X-Y Plot** открыто на вкладке **Labels**

Пример 3.4. График с заголовком

На рис. 3.24 приведен пример заголовка графика с выполнением надписей вдоль координатных осей.

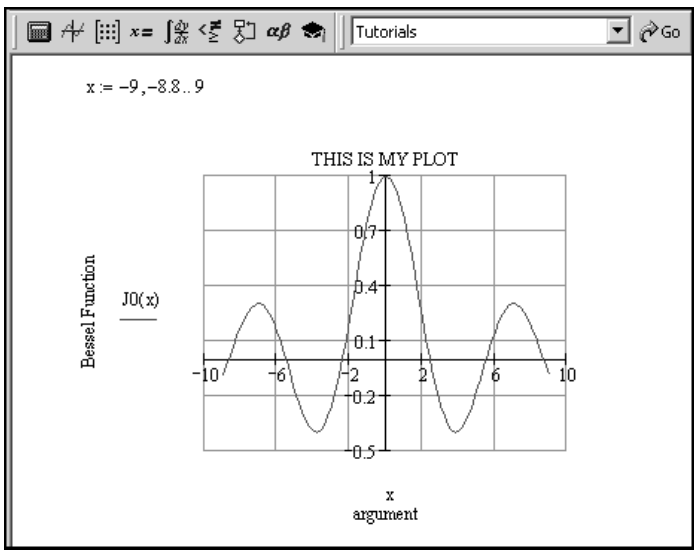


Рис. 3.24. Пример использования заголовка графика функции

На рис. 3.25 представлено диалоговое окно **Formatting Currently Selected X-Y Plot** с настройками, которые следует выполнить на вкладке **Labels**.

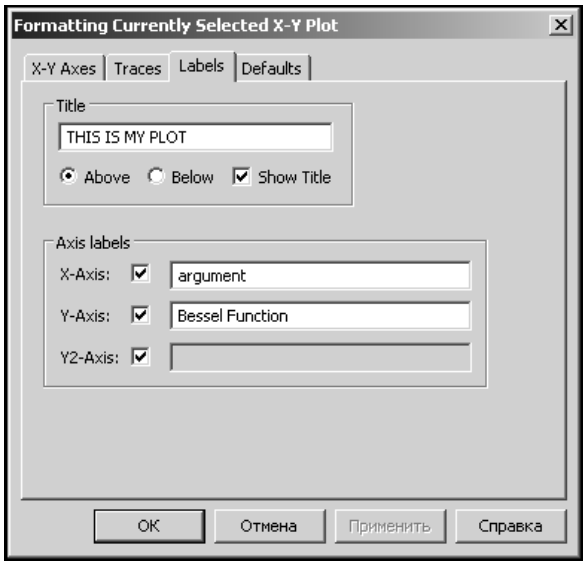


Рис. 3.25. Настройки на вкладке **Labels**

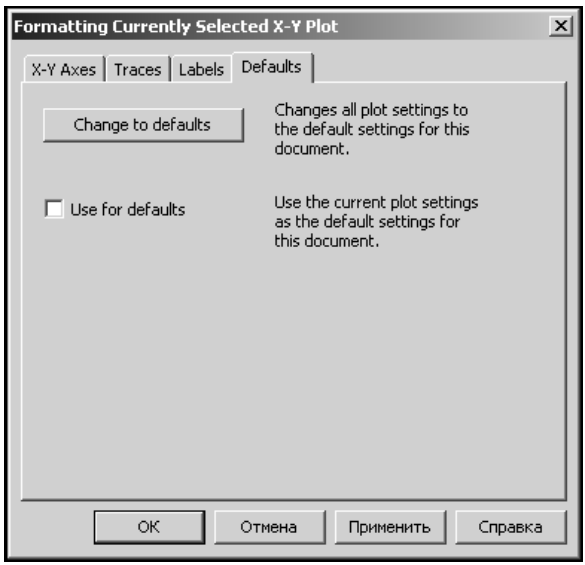


Рис. 3.26. Вкладка **Defaults**

Четвертая вкладка **Defaults** диалогового окна **Formatting Currently Selected X-Y Plot** необходима в двух случаях: если нужно отменить выполненные настройки и перейти к тем, что определены в Mathcad по умолчанию, или если выполненные настройки для данного графика следует применять по умолчанию ко всем графикам в рабочем документе. Как выглядит данная вкладка, показано на рис. 3.26.

Вкладка содержит всего два функциональных элемента: кнопку **Change to defaults** и опцию **Use for defaults**. Для применения настроек по умолчанию к выбранному графику, щелкают на кнопке **Change to defaults**. Наличие флажка опции **Use for defaults** приводит к тому, что настройки данного графика принимаются по умолчанию для всех графиков в данном рабочем документе.

Графики в полярных координатах

Большой популярностью пользуется полярная система координат. Как известно, в этом случае координата точки на плоскости задается с помощью двух параметров: расстояния r от точки до начала координат и угла φ между горизонтальной координатной осью и линией, соединяющей точку с началом координат. Связь между декартовыми и полярными координатами дается соотношениями $x = r\cos\varphi$ и $y = r\sin\varphi$. Часто уравнения, определяющие кривые функциональных зависимостей, записываются сразу в полярной системе координат (иногда это бывает удобно). При этом для отображения графика функциональной зависимости существует две возможности. Первая состоит в том, чтобы перейти в уравнении к декартовым координатам. Однако в этом случае, к сожалению, не всегда можно разрешить полученное уравнение относительно одной из декартовых переменных, в силу чего соответствующая функция будет задана в неявном виде. Вторая возможность связана с построением графика в полярной системе координат. И это предусмотрено в Mathcad. В данном случае нет необходимости переписывать исходное уравнение в декартовых координатах, что существенно упрощает задачу.

Пример 3.5. Спираль Архимеда

В качестве небольшого примера рассмотрим процедуру построения графика кривой, которая называется спиралью Архимеда. Соответствующее уравнение имеет вид $r = a\varphi$, где a — константа, а угол φ изменяется в пределах от 0 до 2π.

Процедура построения графика в полярной системе координат мало чем отличается от создания обычного двумерного графика, однако есть и ряд особенностей. В частности, после нажатия комбинации клавиш <Ctrl>+<7> или

выбора команды **Insert | Graph | Polar Plot** в рабочий документ будет вставлена область создания полярного графика, в центре которой изображен круг, снизу и слева от которого размещены два структурных заполнителя (рис. 3.27).

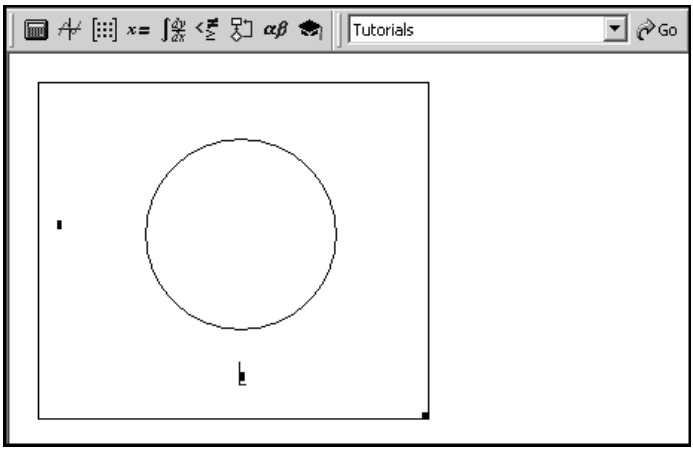


Рис. 3.27. Область создания полярного графика

Вместо нижнего заполнителя, как и в случае графика в декартовых координатах, указывается аргумент функции. Тот заполнитель, что находится слева от центрального круга в области создания графика, заменяется на название функции с указанием ее аргумента (или выражения, зависящего от переменной-аргумента, указанной вместо нижнего структурного заполнителя). Если указывается функция, она должна быть предварительно определена, как это сделано в примере на рис. 3.28 (команда $r(\varphi) := 5\cdot\varphi$).

При этом в правой части области создания графика можно видеть еще два структурных заполнителя, которые пока что лучше не трогать. Их назначение станет ясно в дальнейшем.

После щелчка мышью вне области создаваемого рисунка строится кривая для спирали Архимеда (рис. 3.29).

Если диапазон изменения переменной-аргумента функции не указан, то по умолчанию он устанавливается в пределах от 0 до 2π, как и в рассмотренном примере.

Общий вид рисунка, создаваемого на основе настроек по умолчанию, не очень привлекателен, поэтому обычно приходится выполнять настройку самостоятельно. Как и раньше, редактирование подразумевает двойной щелчок мышью в области полярного графика. В результате открывается диалоговое окно **Formatting Currently Selected Polar Plot**, которое очень сильно напо-

минает окно **Formatting Currently Selected X-Y Plot** — только в данном случае опций и переключателей поменьше (рис. 3.30).

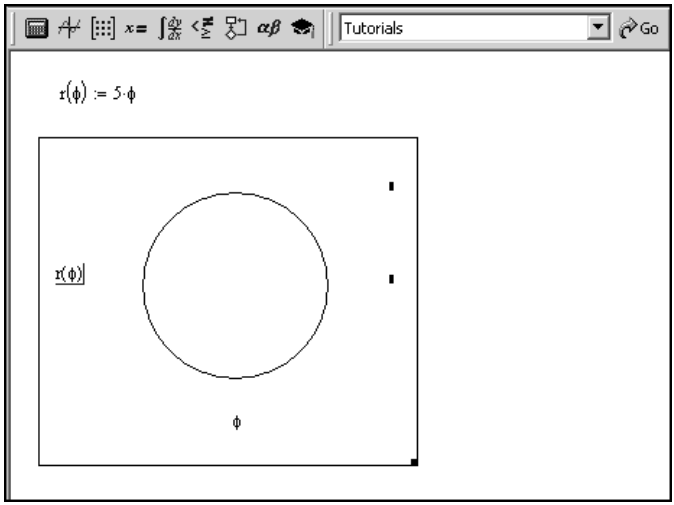


Рис. 3.28. Ввод функции и переменной-аргумента

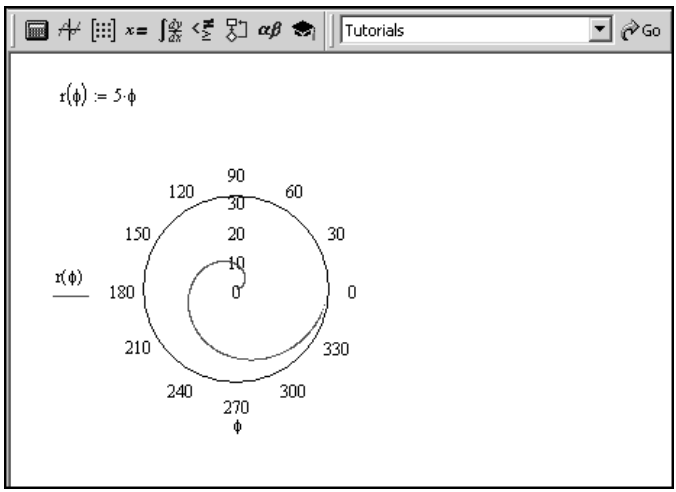


Рис. 3.29. Спираль Архимеда

Особой необходимости подробно описывать опции диалогового окна **Formatting Currently Selected Polar Plot** нет. Особенности настроек, выполняемых в этом окне, связаны с тем, что создаваемый график строится в полярной системе координат, — это накладывает некоторые ограничения. Например, в случае полярного графика логарифмическое масштабирование

можно применять только по отношению к радиальной переменной r (в рассматриваемом примере это функция от азимутальной переменной φ). Зато так же, как и для декартовой системы координат, можно отображать координатную сетку (на рис. 3.30 установлен флажок опций **Grid lines**). Результат применения этой настройки представлен на рис. 3.31.

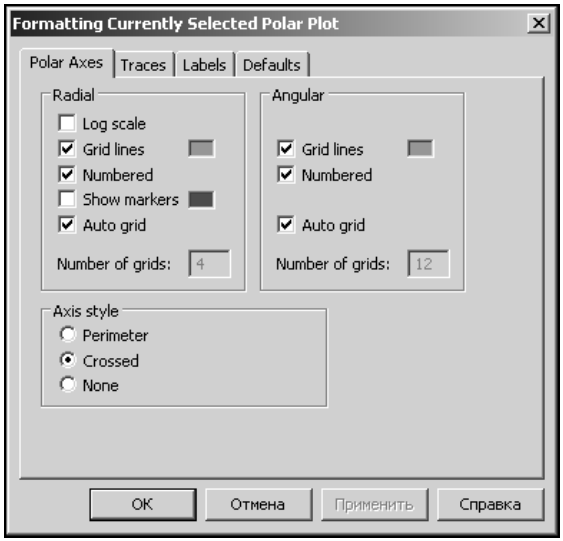


Рис. 3.30. Диалоговое окно **Formatting Currently Selected Polar Plot**

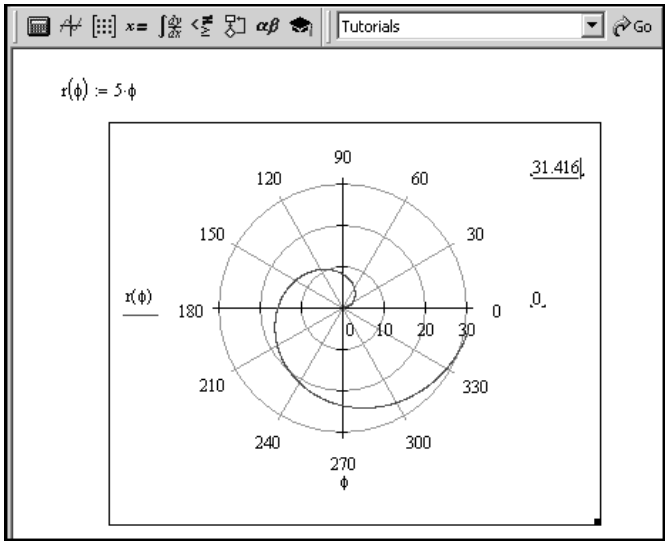


Рис. 3.31. Отображение координатной сетки

На этом же рисунке применена настройка **Crossed**, в силу чего координатные оси отображаются с точкой пересечения в центре рисунка.

Возвращаясь к маркерам заполнения в правой верхней и центральной частях рисунка, которые были оставлены в неприкосновенности, на рис. 3.31 несложно заметить, что в них отображаются значения функции при минимальном и максимальном значениях переменной-аргумента. Если вместо этих значений ввести другие, определяемые пользователем, то график будет отображаться так, что значения функции не выходят за пределы диапазона, определяемого соответствующими величинами. На рис. 3.32 введены значения `10` и `20` для нижней и верхней границ области изменения радиальной переменной.

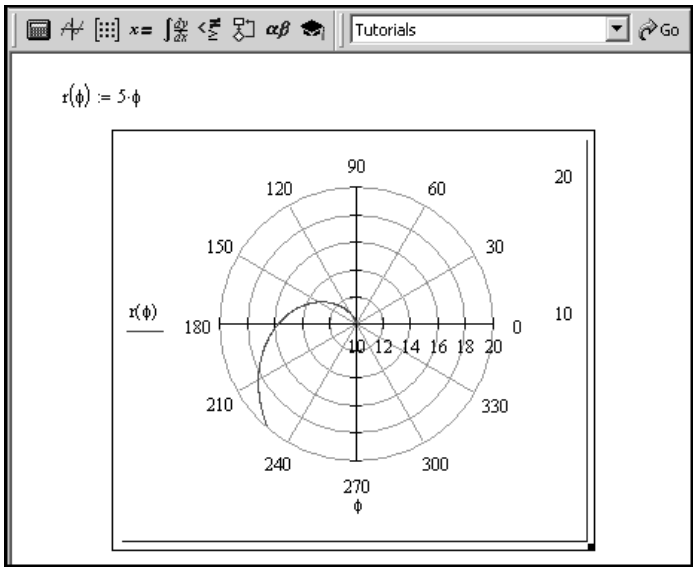


Рис. 3.32. Отображение значений функции в предопределенном диапазоне

Прочие настройки для полярных графиков выполняются и применяются аналогично тому, как это было для декартовых координат. Следует только помнить, что из-за специфики системы координат некоторые настройки могут придать графику экзотический вид. На рис. 3.33 для той же самой функции переменная-аргумент инициализирована со значением-диапазоном с шагом дискретности `0.3` (это сделано для того, чтобы структуру графика легче было проследить по базовым точкам). На вкладке **Traces** диалогового окна **Formatting Currently Selected Polar Plot** выбран тип кривой **stem** (колонка **Type**), а в качестве базовых символов в колонке **Symbol** указаны квадраты.

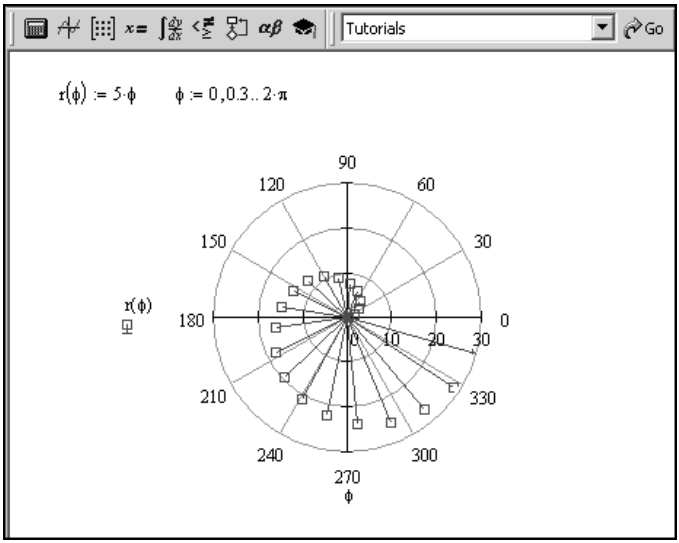


Рис. 3.33. Использование настройки **stem** для полярного графика

Вполне логично было бы предположить, что для примененной настройки базовые линии должны соединяться прямой линией с точкой начала координат, в чем несложно убедиться, взглянув на рис. 3.33.

Основные методы работы с графиками

Далее остановимся на некоторых базовых методах работы с графиками, которые полезно иметь в виду при решении задач с помощью Mathcad. Эти методы позволяют наиболее эффективно реализовать возможности графического представления данных и их интерпретации. В первую очередь следует упомянуть возможность изменения размеров рисунка с графиком (или графиками) в рабочем документе Mathcad. Для этого рисунок предварительно выделяют. Часто размеры рисунка нужно изменить так, чтобы сохранялись исходные пропорции (отношение высота-ширина). В этом случае курсор мыши наводят на правую нижнюю точку области выделения рисунка (эта точка отмечена небольшим черным прямоугольным квадратом) и, нажав левую кнопку мыши, перемещают курсор, задавая тем самым приемлемый размер рисунка (рис. 3.34).

Если нужно изменить только ширину или высоту рисунка, то курсор мыши наводят на правую или нижнюю центральные точки области выделения рисунка соответственно. Дальнейшая процедура такая же: нажимают левую кнопку мыши и растягивают или сжимают рисунок в выбранном направлении до нужных границ. На рис. 3.35 продемонстрировано, как увеличить ширину области с графиком.

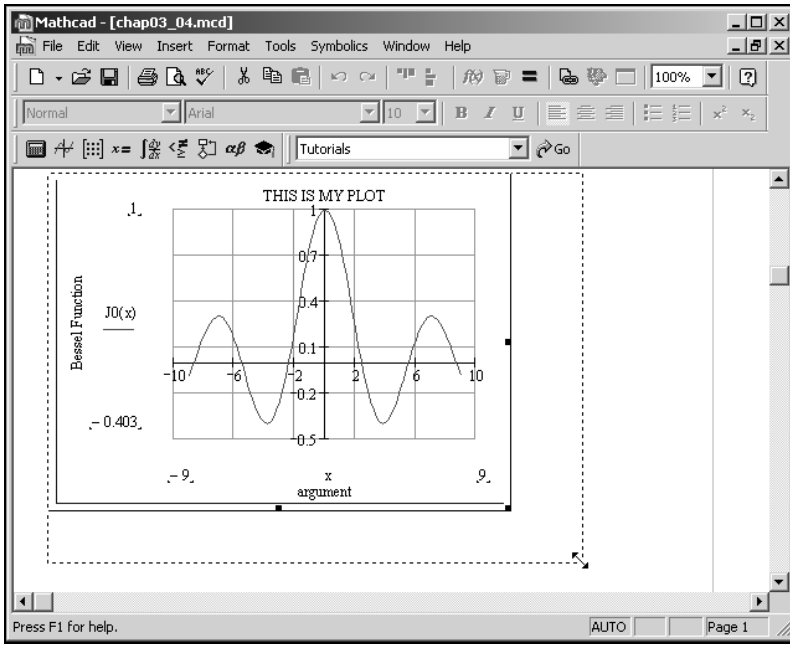


Рис. 3.34. Пропорциональное изменение размеров рисунка

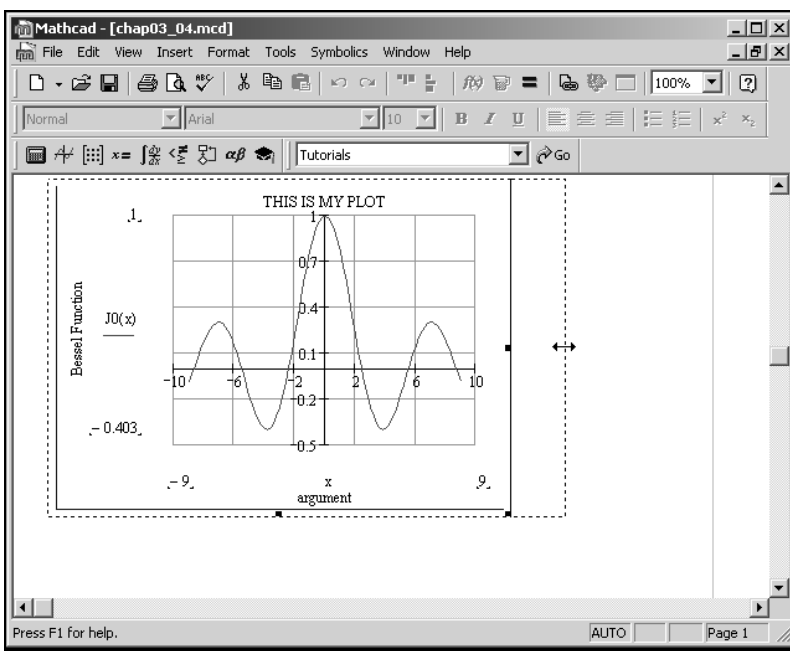


Рис. 3.35. Изменение ширины рисунка

Не представляет никаких трудностей и процедура перемещения рисунка по рабочему документу. Рисунок, как и в предыдущем случае, выделяется, после чего курсор мыши наводят на границу области выделения (в принципе в любом месте). Если курсор размещен правильно, он будет отображаться в виде небольшой ладони (рис. 3.36).

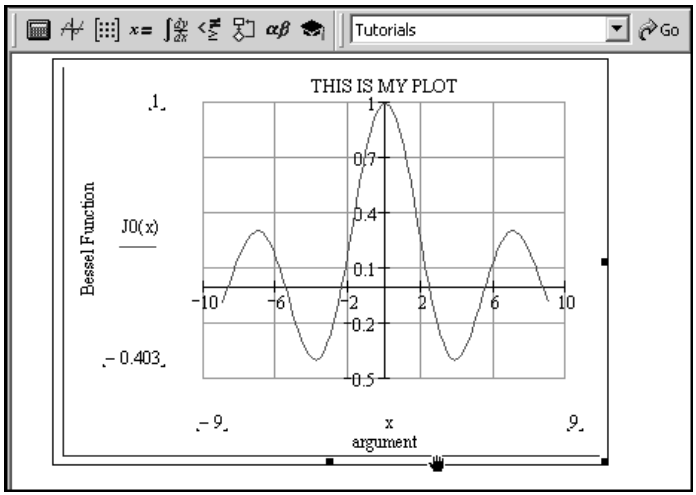


Рис. 3.36. Курсор мыши наведен на границу области выделения рисунка

Поле этого нажимают левую кнопку мыши и, удерживая ее, перемещают рисунок в нужное место рабочего документа. Новое место расположения рисунка выделяется при перемещении курсора пунктирной рамкой (рис. 3.37).

Следует только иметь в виду, что рисунок с графиком функции должен перемещаться по документу так, чтобы инициализация функций и переменных, используемых при построении графика (или графиков) осуществлялась в рабочем документе до непосредственного отображения графика.

В Mathcad существует возможность увеличивать не только весь рисунок с графиком целиком, но и некоторые его части. Это бывает весьма полезно, особенно при работе с быстро осциллирующими функциями, когда для детального отслеживания характера функциональной зависимости приходится рассматривать график на небольших интервалах изменения аргумента. Для выполнения подобной процедуры следует выделить рисунок с графиком и щелкнуть на нем правой кнопкой мыши. В результате открывается список вспомогательного меню, в котором нужно выбрать элемент **Zoom** (рис. 3.38).

После щелчка мышью на этом элементе откроется диалоговое окно **X-Y Zoom**, содержащее четыре неактивных поля и пять кнопок (рис. 3.39).

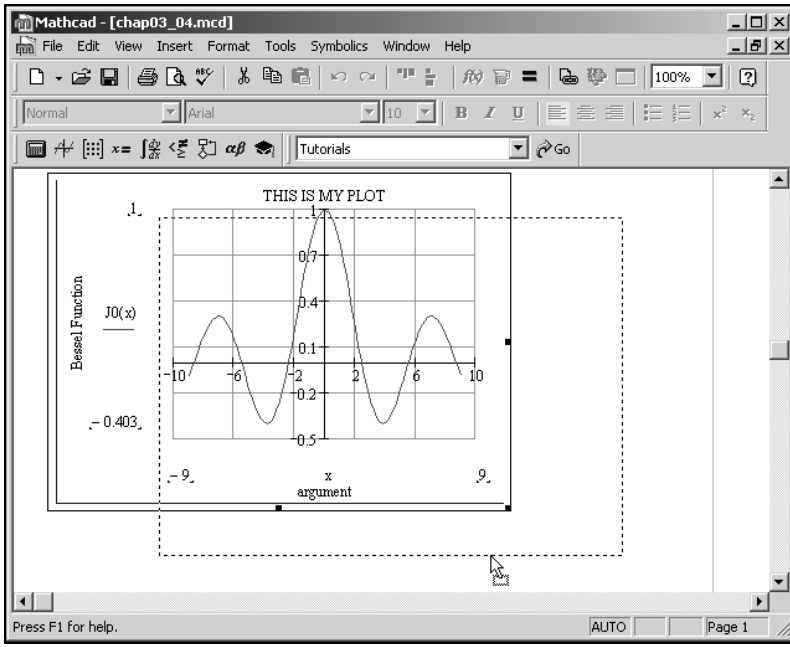


Рис. 3.37. Перемещение рисунка по рабочему документу

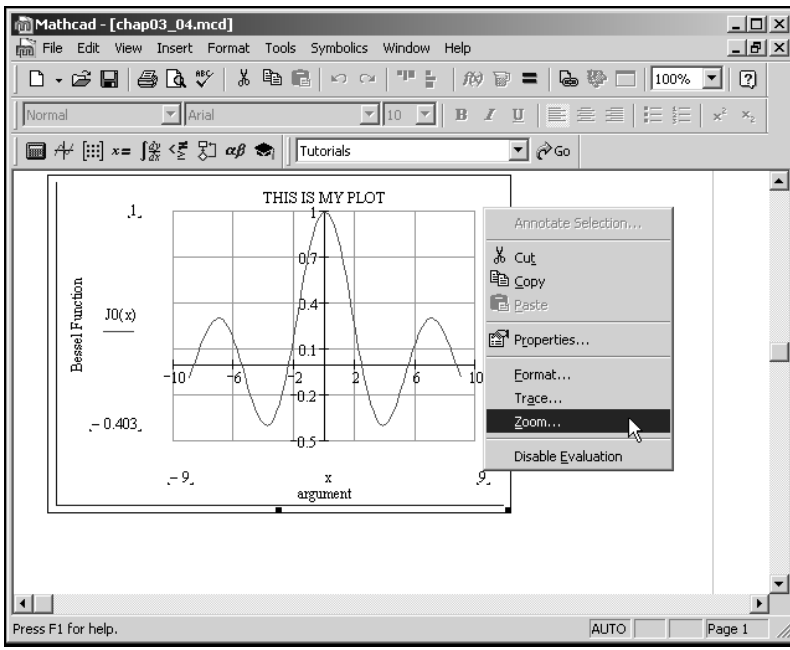


Рис. 3.38. Выбор элемента **Zoom** в списке раскрывающегося меню

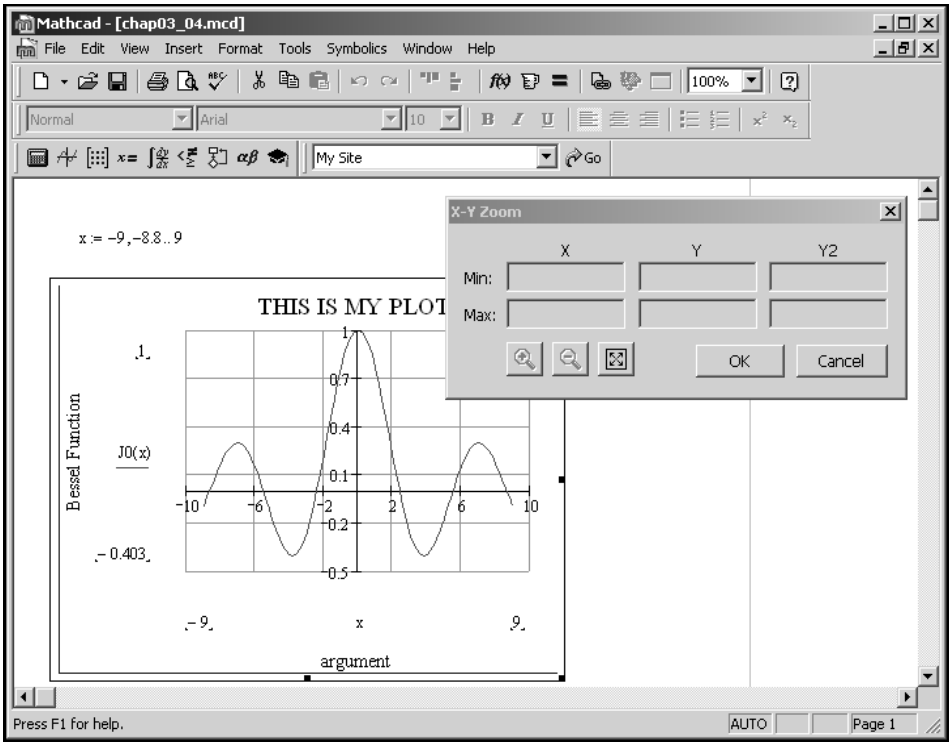


Рис. 3.39. В рабочем документе открыто диалоговое окно **X-Y Zoom**

Далее с помощью мыши нужно выделить ту область на рисунке с графиком, которую предстоит увеличить (рис. 3.40).

При этом в полях диалогового окна **X-Y Zoom** отображаются наибольшие и наименьшие значения координат для данной области. В частности, в строке **Min** отображаются наименьшие значения, а в строке **Max** — наибольшие. Координата вдоль оси абсцисс представлена в полях столбца **X**, а ордината — в полях столбца **Y** (см. рис. 3.40). Для увеличения выделенной области следует щелкнуть на кнопке **Zoom** (кнопка с изображением лупы и знака плюс) в диалоговом окне **X-Y Zoom**. Результат будет таким, как показано на рис. 3.41.

Чтобы оставить изменения в силе, щелкают на кнопке **OK**. Для отмены увеличения используют кнопку **Unzoom** (кнопка с изображением лупы и знака минус). Процесс увеличения можно выполнять несколько раз подряд. При этом сначала выделяется область и увеличивается, затем в этой увеличенной области снова выделяется подобласть и увеличивается и т. д. Чтобы вернуться в первоначальный режим отображения графика, в диалоговом окне **X-Y Zoom** следует щелкнуть на кнопке **Full View** (кнопка с изображением диагональных стрелок, см. рис. 3.41).

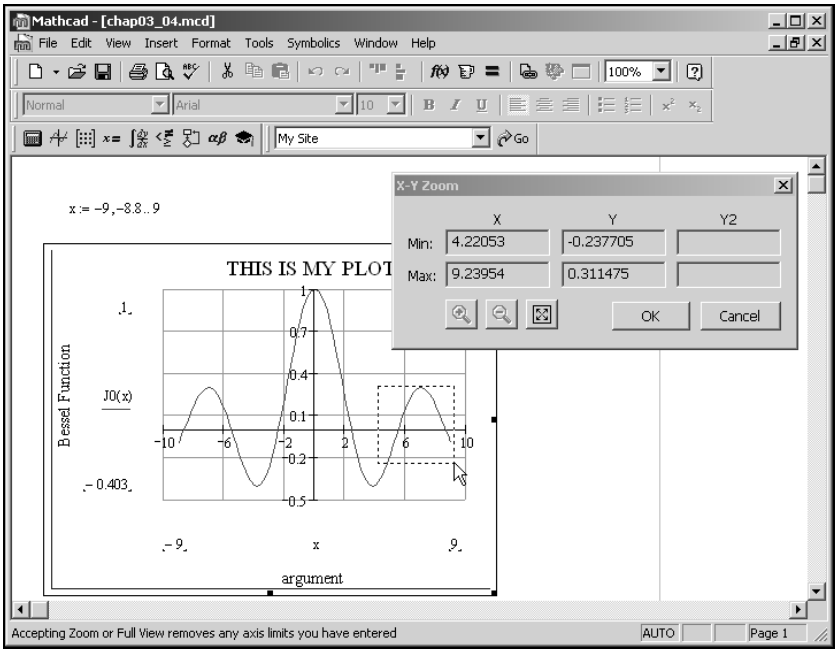


Рис. 3.40. Выделение области для последующего увеличения

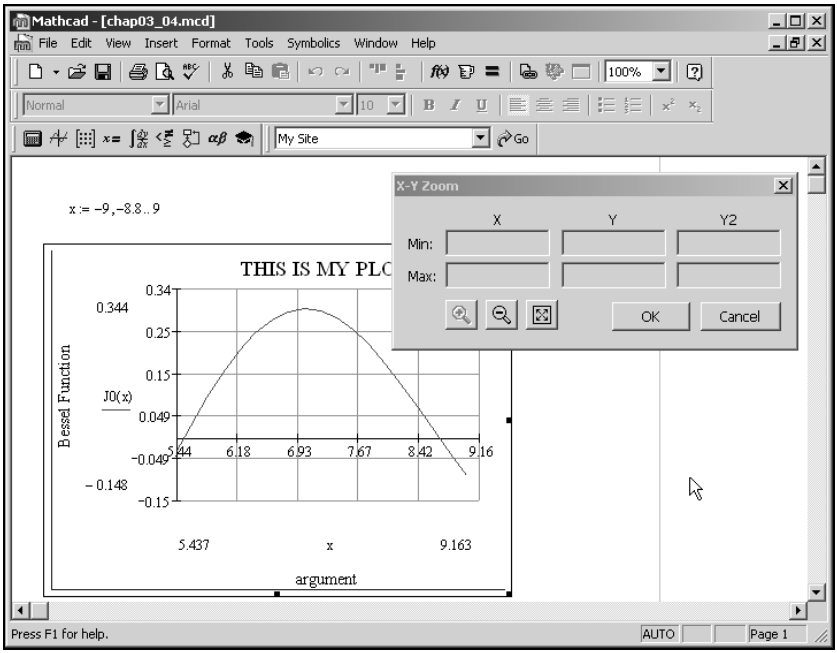


Рис. 3.41. Результат увеличения выделенной области

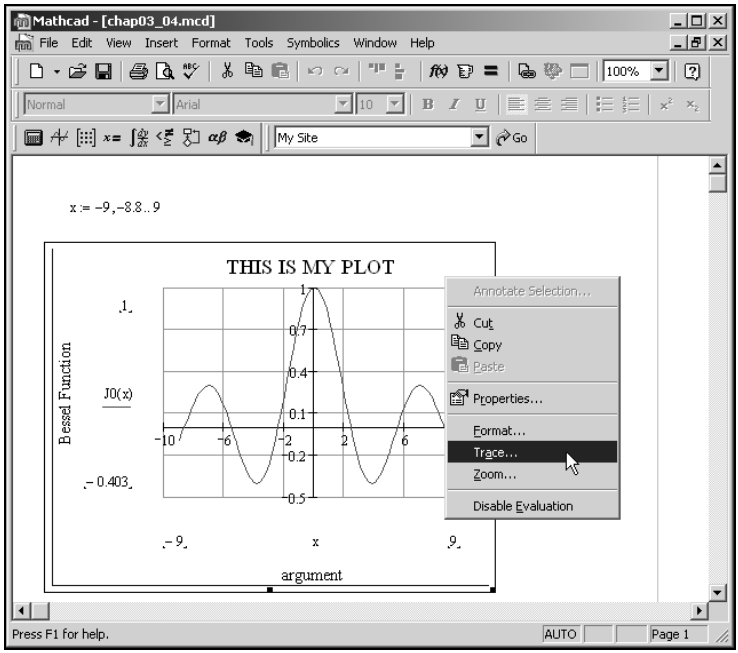


Рис. 3.42. Выбор элемента **Trace** из раскрывающегося списка

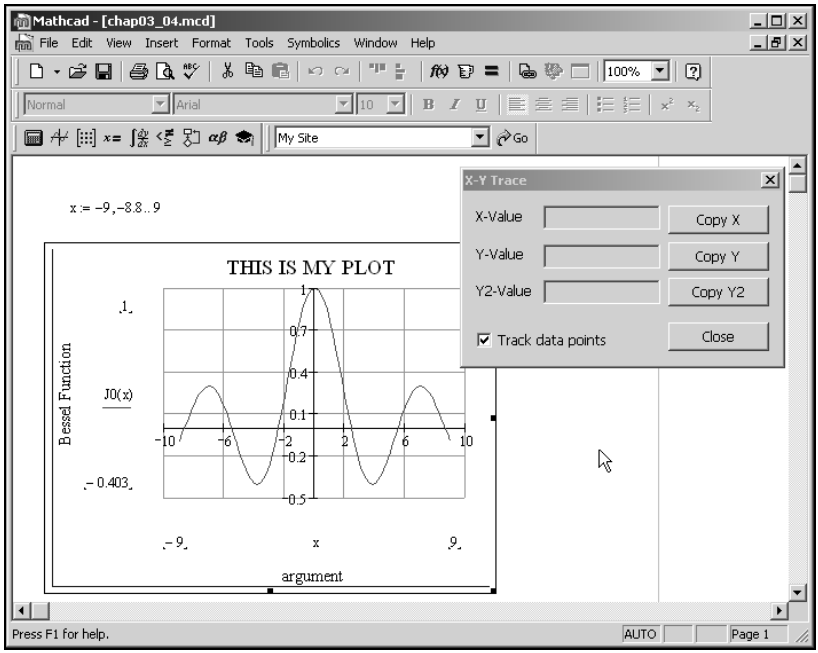


Рис. 3.43. В рабочем документе открыто диалоговое окно **X-Y Trace**

Другая важная проблема, которая с успехом решается в Mathcad, состоит в следующем. Очень часто нужно знать абсциссу и ординату точек, лежащих на кривой графика функции. Иногда такую оценку можно выполнить, что называется, на глаз, однако это не всегда возможно, да и точность подобных оценок оставляет желать лучшего. В Mathcad, как уже отмечалось, это не проблема. В качестве подтверждения рассмотрим следующую процедуру. Выделим рисунок с графиком функции, после чего щелкнем на нем правой кнопкой мыши. В уже знакомом раскрывающемся списке меню выберем элемент **Trace** (рис. 3.42).

В результате в рабочей области документа открывается диалоговое окно **X-Y Trace**, имеющее всего два неактивных поля, три кнопки и одну опцию (рис. 3.43).

Если теперь щелкнуть левой кнопкой мыши в области графика функции, то, во-первых, на графике будет выделена точка, ближайшая к точке размещения курсора (при установленном флажке опции **Track data points**), и, во-вторых, координаты этой точки отображаются в полях диалогового окна **X-Y Trace**: в поле **X-Value** можно видеть абсциссу точки, а в поле **Y-Value** — ординату. На графике эта точка выделяется горизонтальной и вертикальной пунктирными линиями (рис. 3.44).

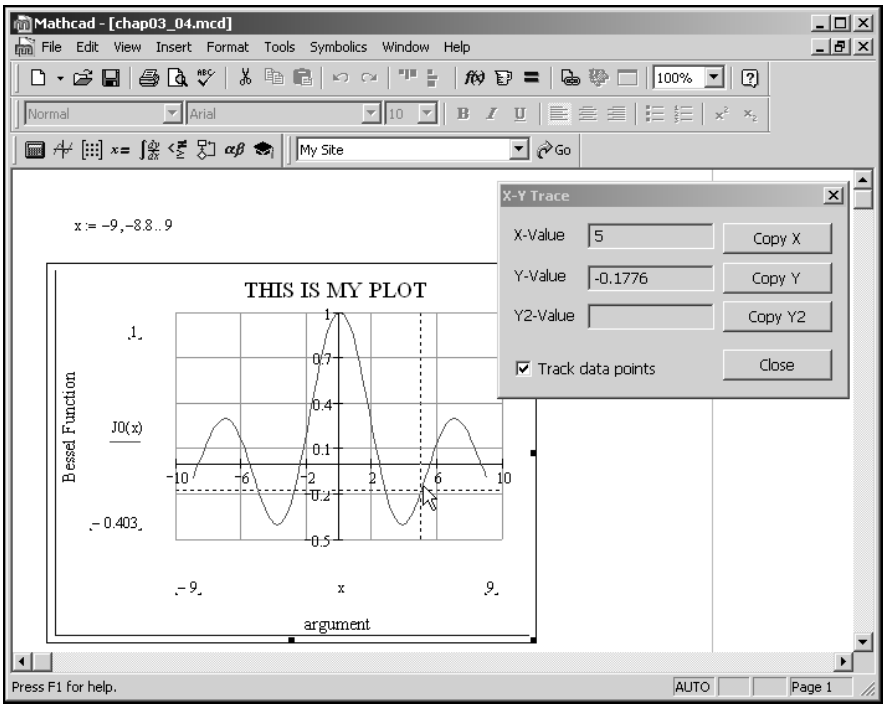


Рис. 3.44. Определение координат точки на графике

Если значения абсциссы или ординаты нужно скопировать, щелкают соответственно на кнопке **Copy X** или **Copy Y**. Флажок опции **Track data points** устанавливают в том случае, если нужно перемещаться вдоль кривой графика (см. рис. 3.44). Иногда бывает необходимо просто определить координаты точки в области отображения графика (при том, что сама точка на кривой графика не лежит). В этом случае необходимо отменить опцию **Track data points**.

Отображение массивов данных

Помимо непрерывных функциональных зависимостей, в Mathcad на графиках с успехом можно представлять и дискретные наборы данных. Обычно речь в этом случае идет о данных, являющихся элементами массивов. Далее рассматривается простой пример, позволяющий составить некоторое представление о возможностях Mathcad в этом вопросе. Более сложные задачи приводятся в последующих главах книги.

Пример 3.6. Отображение значений массива

В самом простом варианте процедура отображения значений элементов массива на графике выглядит следующим образом: при создании графика вместо переменной-аргумента указывается индексная переменная, а в качестве функции указывают название массива (с указанием индекса в символьном виде). Предварительно этот массив должен быть определен, а индексной переменой заблаговременно следует присвоить значение-диапазон. Пример приведен на рис. 3.45.

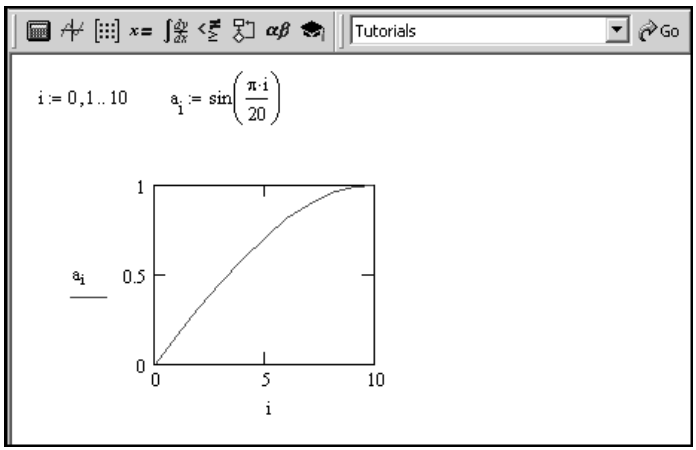


Рис. 3.45. Построение графика по точкам

По умолчанию даже в том случае, если график строится по точкам, он отображается в виде кривой линии, т. е. базовые точки соединяются линиями. В частности, в рассмотренном случае сначала определяется индексная переменная `i` со значением-диапазоном (значения от 0 до 10 с шагом 1), после чего инициализируется массив `a`, элементы которого определяются через их индекс `i` согласно формуле `sin(πi/20)`. Далее при создании графика у горизонтальной оси указывается индексная переменная `i`, а вместо функции вводится выражение a_i. Таким образом, если формально рассматривать индексную переменную как аргумент, то элемент массива в данном случае следует интерпретировать как функцию от этого аргумента (индекса).

Что касается непосредственно графика, то к нему можно применять те же настройки, что и к обычным графикам, которые рассматривались ранее. В качестве примера на рис. 3.46 показан тот же график, что и на рис. 3.45, но только в данном случае он несколько увеличен в размерах, отображаются только базовые точки (в виде символа косого креста), а также добавлена координатная сетка и явно указана область отображения графика вдоль координатных осей.

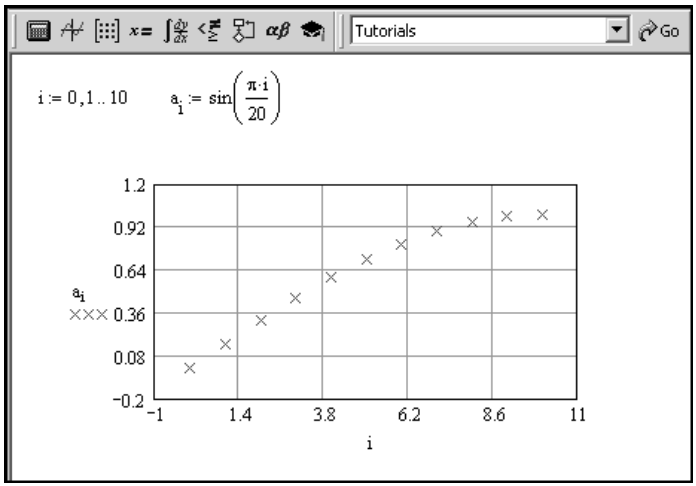


Рис. 3.46. Изменение настроек графика

Встречается еще одна практически важная ситуация, когда и абсциссы, и ординаты точек, которые следует отобразить на графике, являются элементами массивов. Пусть горизонтальные координаты точек — элементы массива `b`, а вертикальные — массива `a`. Для отображения точек на рисунке необходимо у горизонтальной координатной оси указать название массива `b`, а у вертикальной — массива `a` (рис. 3.47).

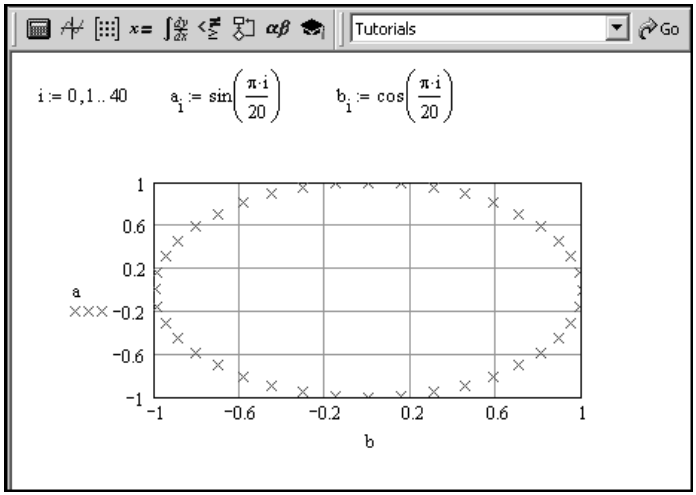


Рис. 3.47. Координаты точек являются элементами массивов

В рассмотренном примере все точки лежат на дуге окружности единичного радиуса с центром в начале координат. На рис. 3.47 эта фигура скорее напоминает эллипс. Причина проста и состоит в том, что масштаб вдоль горизонтальной и вертикальной координатных осей различен. В завершение хочется отметить, что все рассмотренное в этом разделе с минимальными изменениями относится и к полярным графикам.

Создание трехмерных графиков

Задача построения в Mathcad трехмерного графика или, если быть более корректным, поверхности, сложнее. Во-первых, в этом случае количество возможных типов таких графиков больше. Во-вторых, в силу очевидных причин становится более громоздкой процедура определения функций, переменных и т. п. Однако в целом прослеживается тот же концептуальный подход, что и при работе с двумерной графикой. В Mathcad можно создавать следующие типы трехмерных графиков:

- ❒ график-поверхность, в том числе заданный в параметрическом виде;
- ❒ контурный график;
- ❒ трехмерная гистограмма;
- ❒ трехмерный график рассеяния;
- ❒ трехмерный график векторного поля.

В зависимости от типа графика, процедура его создания имеет свои особенности. Сначала рассмотрим, как создаются графические поверхности.

Создание поверхностей

График-поверхность создается для функций, зависящих от двух аргументов. При этом по горизонтальным осям обычно отображаются значения независимых переменных, а вдоль вертикальной оси отсчитывают значение зависимой переменной, т. е. функции. Эта функция может быть задана как в явном виде, так и в параметрическом. Кроме того, можно создавать поверхности по массивам данных, что с практической точки зрения весьма важно и удобно.

Пример 3.7. График функции двух переменных

Простой и быстрый способ построения графика-поверхности подразумевает на первом этапе определение функции двух переменных (рис. 3.48).

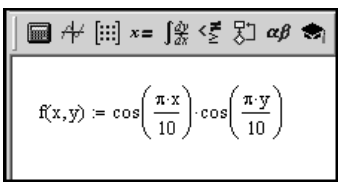


Рис. 3.48. Определение функциональной зависимости для последующего графического отображения

Далее следует выбрать команду **Insert | Graph | Surface Plot** (или нажать комбинацию клавиш <Ctrl>+<2>). В результате в рабочую область документа будет вставлена область трехмерного графика-поверхности, содержащая три перпендикулярные координатные оси и структурный заполнитель в левом нижнем углу (рис. 3.49).

Вместо структурного заполнителя следует ввести название (без аргументов!) функции, определенной ранее. После щелчка мышью вне области графика, системой создается соответствующая поверхность (рис. 3.50).

Несложно заметить, что область изменения аргументов для отображаемой функциональной зависимости нигде в явном виде не указывалась. Предлагаемый системой по умолчанию диапазон значений, для которого строится зависимость, можно изменить уже после создания графика. Для этого выполняется двойной щелчок в области графика и в открывшемся диалоговом окне **3-D Plot Format** переходят к вкладке **QuickPlot Data** (рис. 3.51).

Вкладка состоит из трех разделов, размещенных по горизонтали. Первые два идентичных раздела называются **Range 1** и **Range 2** и содержат по три поля каждое. В этих полях указываются минимальное (**start**) и максимальное (**end**) значения для диапазона отображения функциональной зависимости по первой и второй координате. В поле **# of Grids** указывается число базовых узлов по координатным осям, на основе которых строится график.

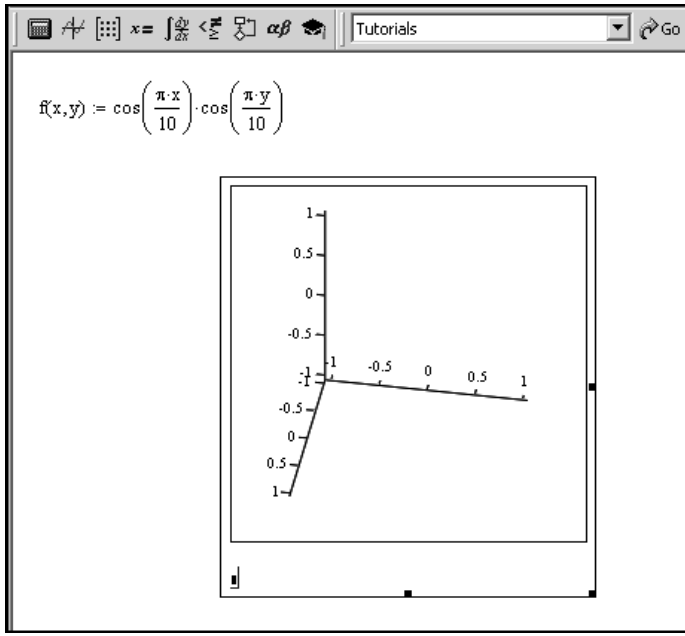


Рис. 3.49. Область создания трехмерного графика

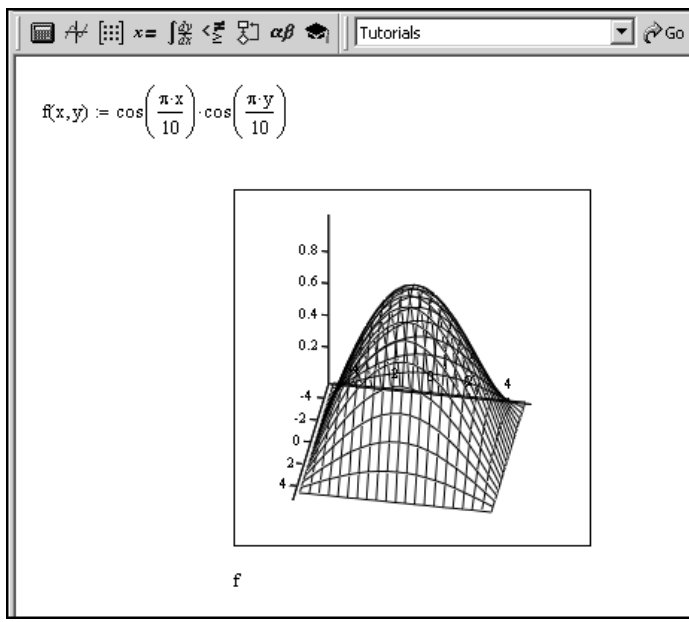


Рис. 3.50. Трехмерный график-поверхность

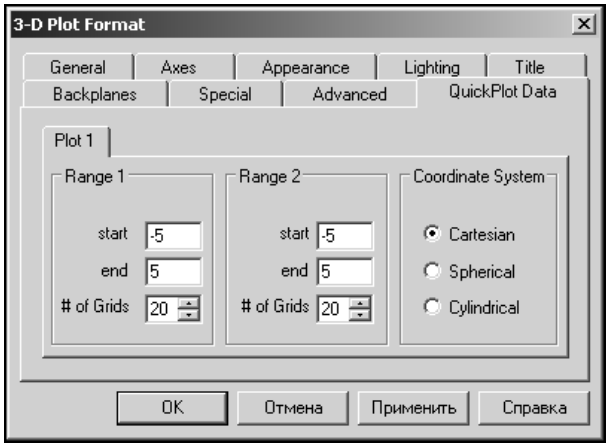


Рис. 3.51. Диалоговое окно **3-D Plot Format** открыто на вкладке **QuickPlot Data**

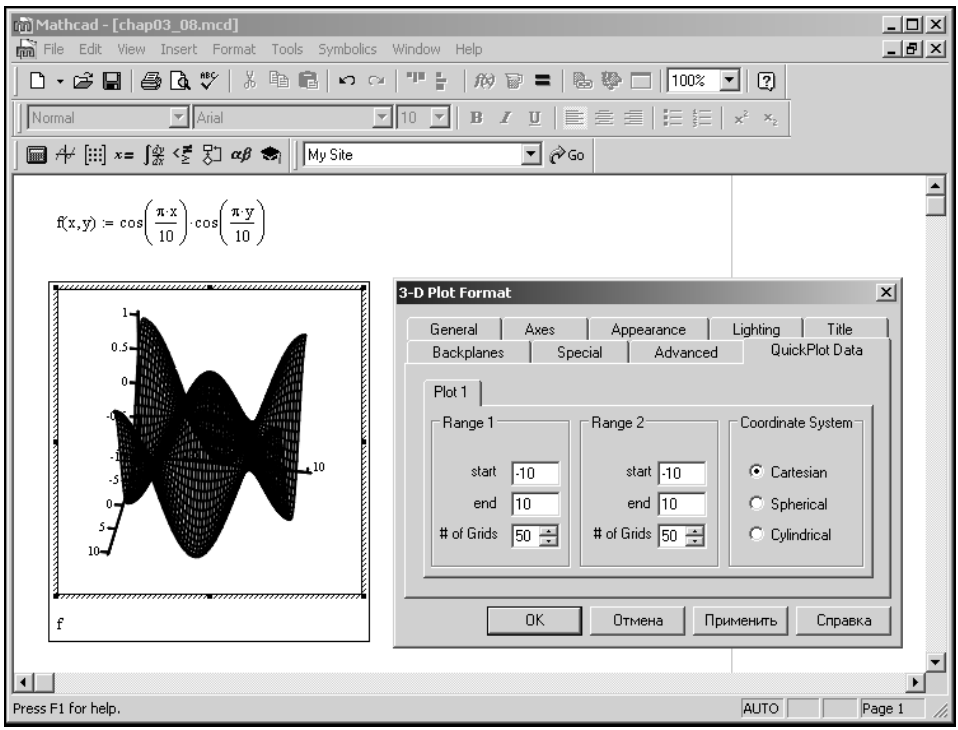


Рис. 3.52. Изменение настроек трехмерного графика

На рис. 3.52 показано, как будет выглядеть созданный ранее график, если его строить в диапазоне изменения переменных от –10 до 10 при 50 узловых точках по каждой из осей.

В разделе **Coordinate System** вкладки **QuickPlot Data** размещена группа переключателей на три позиции, с помощью которой выбирается тип координатной системы: декартова (положение переключателя **Cartesian**), сферическая (положение **Spherical**) или цилиндрическая (положение **Cylindrical**). Вкладка **QuickPlot Data** может сама состоять из нескольких (для каждой поверхности) вкладок. В рассмотренном примере строилась одна поверхность, поэтому описанные поля и переключатели размещены на единой вкладке.

Основные настройки трехмерного графика

Добиться желаемого вида для трехмерного графика можно, выполнив некоторые настройки на вкладках диалогового окна **3-D Plot Format**. На рис. 3.53 показано это диалоговое окно, открытое на вкладке **General**.

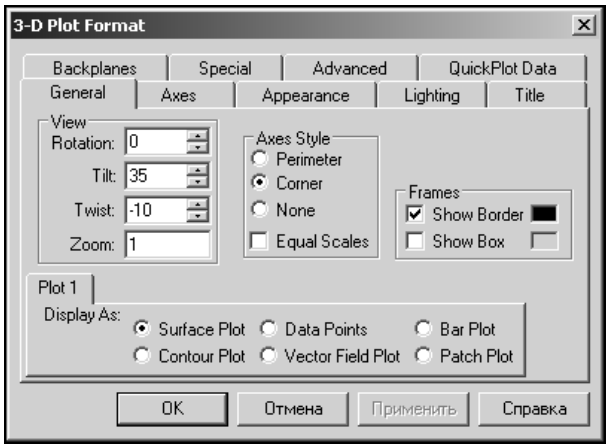


Рис. 3.53. Диалоговое окно **3-D Plot Format** открыто на вкладке **General**

В разделе **View** на этой вкладке размещены четыре поля: **Rotation**, **Tilt**, **Twist** и **Zoom**. В трех полях **Rotation**, **Tilt** и **Twist** указываются углы поворотов системы координат при отображении, а в поле **Zoom** вводится коэффициент масштабирования (он может быть и больше единицы). Ориентация системы определяется, фактически, углами Эйлера. В табл. 3.3 более детально описан способ определения этих углов.

В разделе **Axes Style** задается способ отображения координатных осей. Раздел состоит из переключателя на три положения и опции **Equal Scales**. Флажок опции устанавливается в том случае, если нужно, чтобы масштаб вдоль всех координатных осей был единым. Установив переключатель в положение **Perimeter**, можно перейти в режим, при котором координатные оси отобра-

жаются на переднем плане в области рисунка. По умолчанию переключатель установлен в положение **Corner**. В этом случае координатные оси пересекаются в углу, в начале системы координат. Чтобы не отображать координатные оси совсем, переключатель устанавливают в положение **None**.

Таблица 3.3. *Углы ориентации системы координат*

Поле для ввода угла	Описание
Rotation	Угол поворота системы координат относительно неподвижной координатной системы вокруг оси z. Значением может быть число в диапазоне от –360 до 360°
Tilt	Угол поворота вокруг оси x системы координат, получающейся после поворота вокруг оси z на угол, указанный в поле **Rotation**. Значением может быть число в диапазоне от 0 до 180°
Twist	Угол поворота вокруг оси z системы координат, получающейся сначала поворотом вокруг оси z неподвижной системы (угол поворота указан в поле **Rotation**) и последующим поворотом вокруг новой оси x (угол этого поворота указан в поле **Tilt**). Значением может быть число в диапазоне от –360 до 360°

Раздел **Frames** содержит всего две опции: **Show Border** и **Show Box**. Наличие флажка опции приводит к отображению границы области рисунка при его выделении и рамок ограничивающего рисунок параллелепипеда соответственно. Справа от опций расположены поля выбора цвета. При активной опции щелчок на этом поле приводит к открытию специального диалогового окна, в котором выбирается необходимый цвет.

Вкладка в нижней части диалогового окна **3-D Plot Format**, открытого на вкладке **General**, содержит группу переключателей **Display As**. Переключатель может находиться в одном из шести положений, и в каждом случае будет задаваться определенный тип графика.

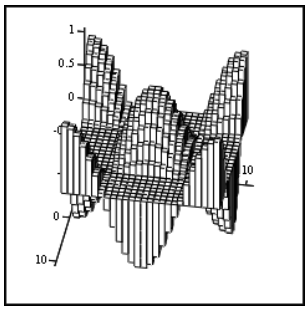
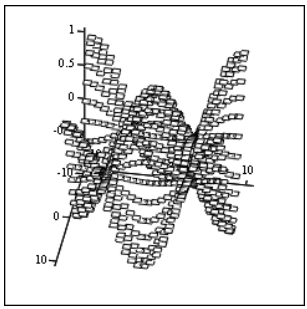
Пример 3.8. Графики разных типов

В табл. 3.4 приведены сведения о том, как положение переключателя влияет на способ отображения графика и примеры применения описываемых настроек к созданному ранее графику. Для большей наглядности предварительно на вкладке **QuickPlot Data** число базовых узлов вдоль координатных осей, по которым строится график зависимости, установлено равным 25.

Таблица 3.4. *Типы трехмерных графиков*

Положение переключателя	Описание	Пример применения
Surface Plot	Функциональная зависимость отображается в виде графика-поверхности	
Contour Plot	Контурный график. Плоское изображение, на котором по периметру откладываются значения координат-аргументов функции, а сам график представляет собой набор линий одинаковых значений функции. Другими словами, вдоль отдельной линии функция принимает одно и то же значение	
Data Points	На графике отображаются только значения функции в узловых точках	
Vector Field Plot	График векторного поля. О нем речь пойдет несколько позже	Для рассматриваемой зависимости этот тип неприменим

Таблица 3.4 *(окончание)*

Положение переключателя	Описание	Пример применения
Bar Plot	Область под поверхностью графика разбивается на прямоугольные параллелепипеды	
Patch Plot	Контур графика отображается с помощью плоских квадратных меток	

На вкладке **Axes** выполняются настройки, имеющие отношение к способу отображения координатных осей. Эта вкладка сама состоит из трех вкладок: **X-Axis**, **Y-Axis** и **Z-Axis** (рис. 3.54).

Вкладки абсолютно идентичны и служат для настройки трех координатных осей (в соответствии с названиями вкладок). В левой части каждой из вкладок находится раздел **Grids**, основными элементами которого являются опции **Draw Lines**, **Draw Ticks** и **Auto Grid** (см. рис. 3.54). Наличие флажка опции приводит соответственно к отображению на графике линий координатной сетки для соответствующей оси, отображению засечек вдоль координатной оси и автоматическому разбиению оси на интервалы координатной сетки. Если последний режим не активизирован (отсутствует флажок опции **Auto Grid**), то в поле **Number** можно в явном виде указать число интервалов разбиения для координатной сетки. При активной опции **Draw Lines** доступными станут поле **Line Weight**, в котором указывается толщина для линий координатной сетки, а также элемент выбора цвета линий для координатной сетки **Line Color**. Для выбора цвета линий необходимо щелкнуть на цветовом

поле слева от метки **Line Color**. Открывается специальное диалоговое окно с палитрой цветов, откуда выбирается цвет линии.

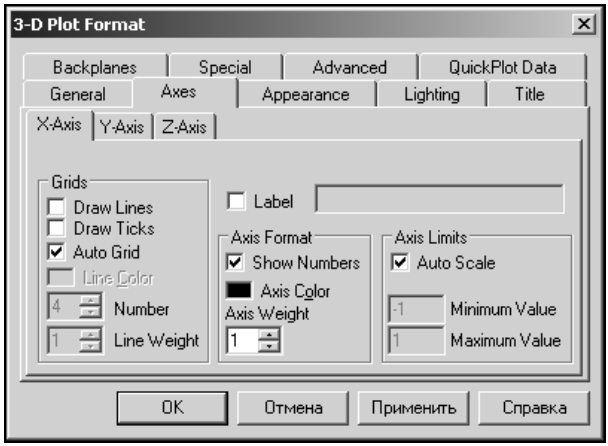


Рис. 3.54. Диалоговое окно **3-D Plot Format** открыто на вкладке **Axes**

Часто бывает необходимо выполнить надписи вдоль координатных осей, например, в явном виде указать название переменной. В этом случае устанавливают флажок опции **Label**. В результате доступным становится поле справа от опции, куда вводится надпись, отображаемая вдоль соответствующей координатной оси.

Формат осей настраивается в разделе **Axis Format**. Здесь представлена опция **Show Numbers**, наличие флажка которой приводит к отображению численных значений у засечек вдоль оси. Помимо этой опции, там есть элемент выбора цвета линии координатной оси **Axis Color** и поле установки толщины линии координатной оси **Axis Weight**. Наконец, в разделе **Axis Limits** определяются границы значений для отображения вдоль координатной оси. Если установлен флажок опции **Auto Scale**, эти границы определяются системой автоматически. При отсутствии флажка в полях **Minimum Value** и **Maximum Value** указывается минимальное и максимальное значения диапазона отображения для данной координатной оси (см. рис. 3.54).

Заголовок для графика вводится в поле **Graph Title** на вкладке **Title** диалогового окна **3-D Plot Format** (рис. 3.55).

Помимо поля **Graph Title** на вкладке **Title** имеется группа переключателей. Если переключатель находится в положении **Above**, заголовок отображается над графиком. Чтобы заголовок отображался под графиком, переключатель должен находиться в положении **Below**. Заголовок не отображается, если выбрано положение **Hide**.

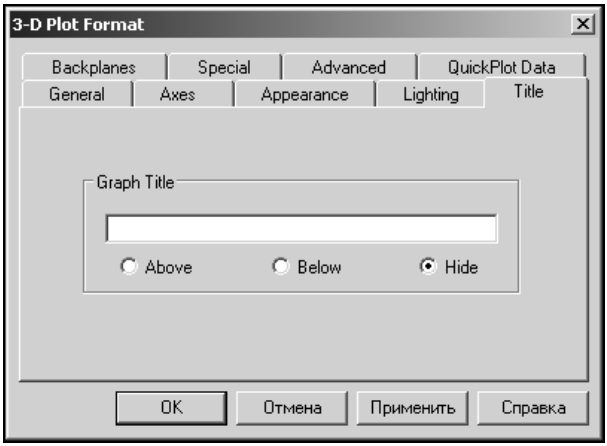


Рис. 3.55. Диалоговое окно **3-D Plot Format** открыто на вкладке **Title**

Пример 3.9. График с заголовком

На рис. 3.56 приведен пример рабочего документа с заголовком, линиями координатной сетки и засечками вдоль координатных осей. Диапазон отображения значений вдоль вертикальной координатной оси здесь и далее установлен от –1,5 до 1,5.

Цветовые настройки и способ заполнения областей графика определяются на вкладке **Appearance** диалогового окна **3-D Plot Format** (рис. 3.57).

Вкладка имеет три раздела (**Fill Options**, **Line Options** и **Point Options**), каждый из которых содержит в нижней части подраздел **Color Options** для выполнения цветовых настроек. Раздел **Fill Options** содержит группу переключателей на три положения и две опции. Если переключатель находится в положении **Fill Surface**, поверхность графика закрашивается одним цветом (оттенками основного, выбранного в поле **Solid Color**), если в соответствующем положении находится переключатель в подразделе **Color Options**, или различными цветами, если переключатель находится в положении **Colormap** (кстати, назначение одноименных переключателей в остальных подразделах аналогичное). Чтобы поверхность закрашивалась в соответствии с контурными линиями графика, переключатель устанавливают в положение **Fill Contours**. Наконец, поверхность не закрашивается цветом при выбранном положении переключателя **No Fill**. При установленном флажке опции **Alternate Mesh** поверхность графика разбивается на квадраты, закрашенные только наполовину. Для перехода в режим сглаживания как цветовой гаммы, так и самой поверхности, устанавливают флажок опции **Smooth Shading**.

Рис. 3.56. Создание заголовка графика

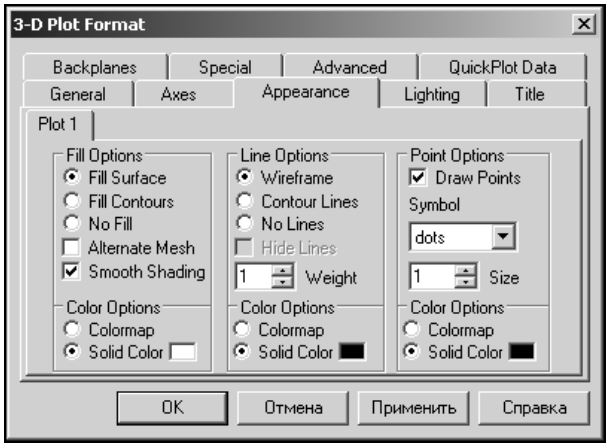


Рис. 3.57. Диалоговое окно **3-D Plot Format** открыто на вкладке **Appearance**

В разделе **Line Options** настраиваются параметры для контурных линий графика. На поверхности графика можно отображать основную сетку (положение переключателя **Wireframe**), контурные линии (положение переключателя **Contour**) и не отображать линии совсем (положение переключателя **No Lines**). В поле **Weight** указывается толщина линий на графике. Кроме того, если в разделе **Fill Options** переключатель установлен в положение **No Fill** (в силу чего поверхность графика не закрашивается), в разделе **Line Options** доступной становится опция **Hide Lines**. Если установить флажок этой опции, график станет непрозрачным в том смысле, что все линии за графиком отображаться не будут.

Помимо основной сетки графика, на нем еще можно отображать и базовые узловые точки. Для этого следует установить флажок опции **Draw Points** из раздела **Point Options** (см. рис. 3.57). Все прочие функциональные элементы этого раздела доступны, только когда данная опция активна. В раскрывающемся списке **Symbol** выбирается символ для отображения узловых точек. В поле **Size** устанавливается их размер. Результат применения некоторых из этих настроек показан на рис. 3.58.

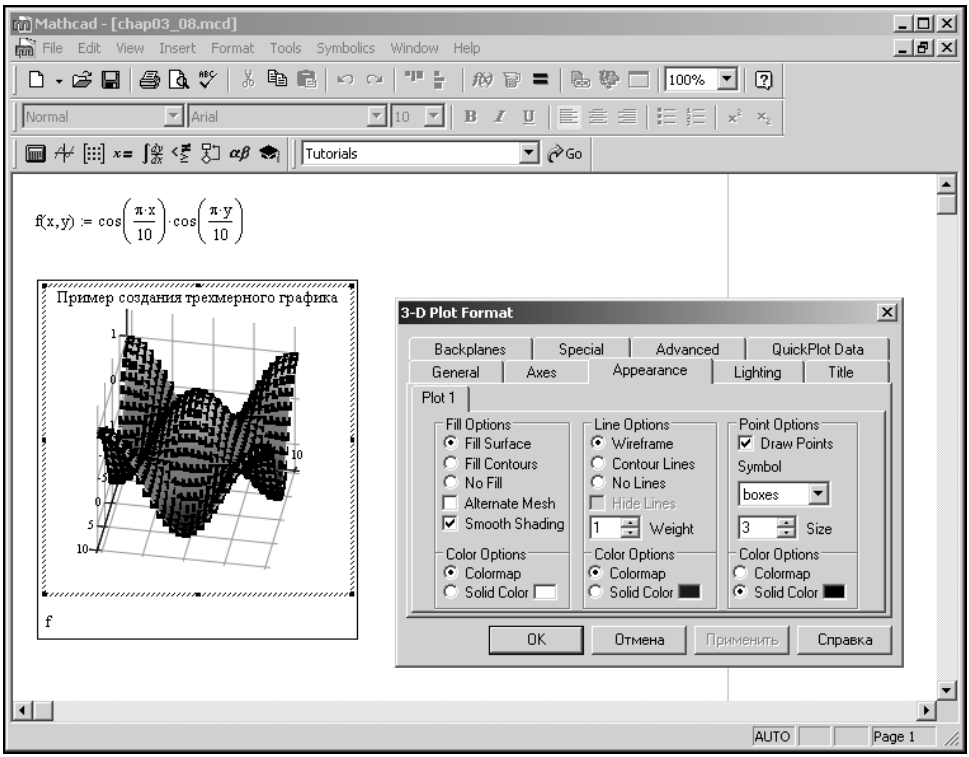


Рис. 3.58. Изменение способа отображения графика

Узловые точки здесь отображаются в виде квадратов (размером 3) черным цветом, выведена основная сетка, а поверхность графика закрашена с использованием различных цветов.

При отображении трехмерного графика можно включать режим подсветки. Основная идея состоит в том, что реализуется специальное выделение графика цветом и яркостью закраски, как если бы он освещался внешним источником света. Причем моделируется освещение как рассеянным светом, так и подсветкой от направленного источника. Настройка этого режима производится на вкладке **Lighting** (рис. 3.59).

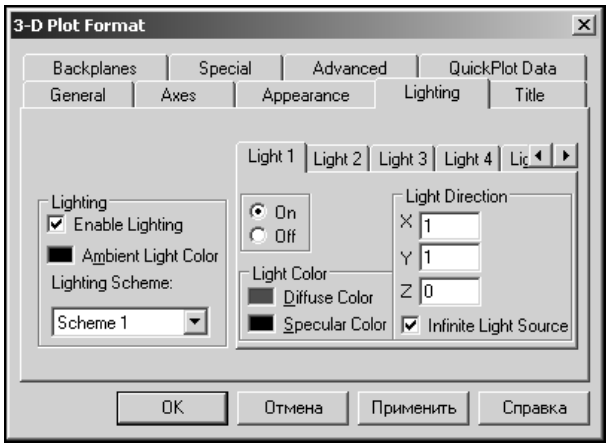


Рис. 3.59. Диалоговое окно **3-D Plot Format** открыто на вкладке **Lightning**

Чтобы элементы на вкладке стали активными, следует установить флажок опции **Enable Lighting**, находящейся в разделе **Lighting**. Кроме нее, там можно найти элемент **Ambient Light Color** для выбора цвета рассеянного света, а также раскрывающийся список **Lighting Scheme**, в котором можно выбрать одну из предопределенных схем освещения. В правой части вкладки имеется раздел, состоящий из восьми идентичных вкладок, названия которых изменяются последовательно от **Light 1** до **Light 8** (справа еще есть кнопки со стрелками для прокрутки вкладок, см. рис. 3.59). Это раздел настройки параметров прямого света. Переключатель на два положения **On** и **Off** позволяет включить и выключить режим использования данного цвета. С помощью элемента **Diffuse Color** выбирают цвет для диффузионного освещения, а цвет для компонента прямого света, отражаемой зеркально от поверхности графика, устанавливают с помощью элемента **Specular Color**. Значения в полях подраздела **Light Location** определяют направление на источник прямого света в данной координатной системе. Наконец, если установлен флажок

опции **Infinite Light Source**, моделируемый источник света предполагают размещенным на бесконечности.

На вкладке **Advanced** можно активизировать ряд специальных эффектов, которые делают график намного более привлекательным. Диалоговое окно **3-D Plot Format**, открытое на этой вкладке, показано на рис. 3.60.

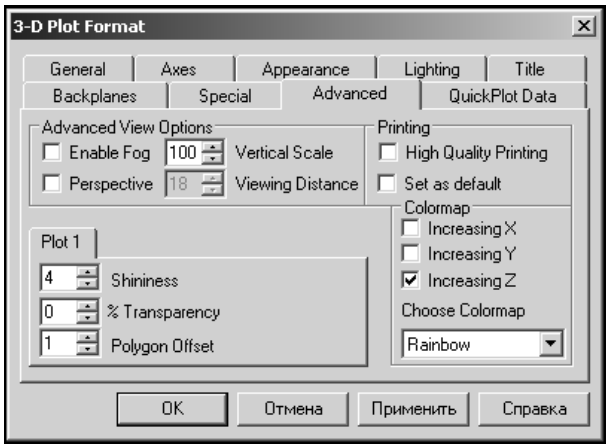


Рис. 3.60. Диалоговое окно **3-D Plot Format** открыто на вкладке **Advanced**

Вкладка состоит из четырех разделов. В разделе **Advanced View Options** представлены две опции (**Enable Fog** и **Perspective**) и два поля (**Vertical Scale** и **Viewing Distance**). Установив флажок опции **Enable Fog**, переходят в режим тумана. Эффект перспективы включается, если активна опция **Perspective**. При этом доступным становится поле **Viewing Distance**, где в безразмерных единицах устанавливается расстояние, с которого условно просматривается график. В поле **Vertical Scale** устанавливается (в процентах) коэффициент масштабирования вдоль вертикальной координатной оси. В нижней части вкладки размещено три поля. В поле **Shininess** указывается значение для определения отражающей способности поверхности графика — чем выше значение, тем сильнее поверхность отражает свет. Значение в поле **Transparency** определяет (в процентах) коэффициент прозрачности графика. Поле **Polygon Offset** необходимо в тех случаях, когда на фоне закрашенной поверхности следует выделить основные линии сетки графика.

В разделе **Colormap** размещены три опции (**Increasing X**, **Increasing Y** и **Increasing Z**) и раскрывающийся список **Choose Colormap**. Наличие флажка опции приводит к применению выбранного способа цветового выделения в направлении координатных осей в соответствии с названиями опций.

В раскрывающемся списке **Choose Colormap** выбирают схему (основную палитру) закраски. Наконец, две опции в разделе **Printing** позволяют перейти в режим высокого разрешения (точнее, разрешения принтера, и для этого следует установить флажок опции **High Quality Printing**) и сохранить эту настройку для всех новых создаваемых трехмерных графиков по умолчанию (опция **Set as default**).

Пример 3.10. График в тумане

Пример применения некоторых из упоминавшихся настроек показан на рис. 3.61.

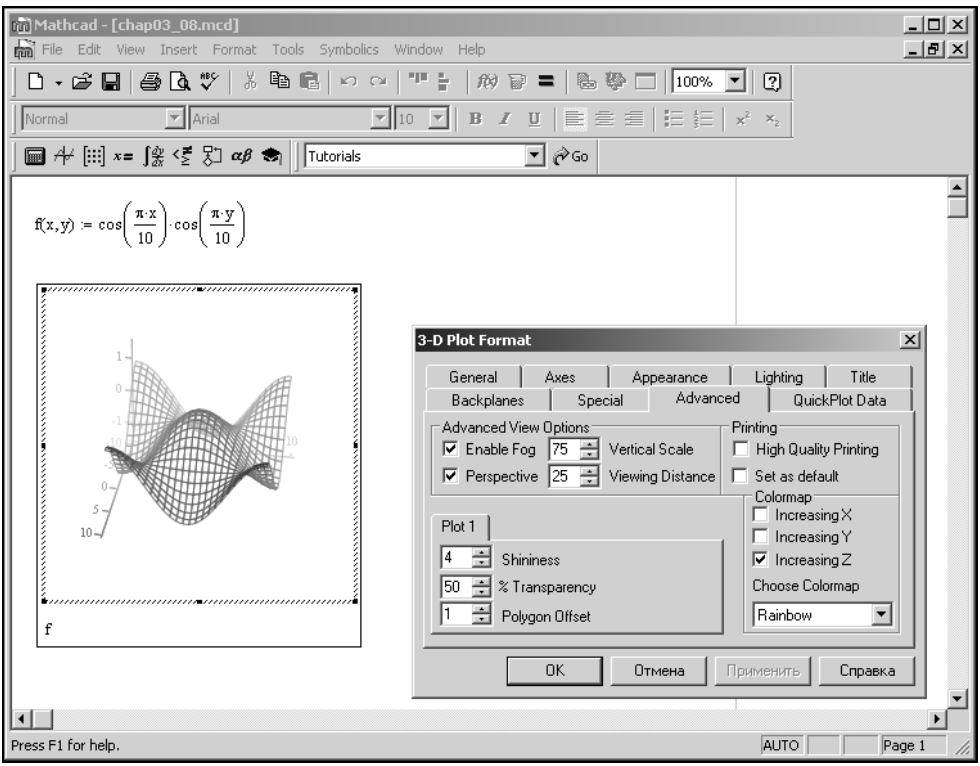


Рис. 3.61. Использование дополнительных эффектов отображения

К графику применен режим тумана, использован эффект перспективы, изображение масштабировано вдоль вертикальной оси и установлена 50%-ная прозрачность.

Доступность элементов, находящихся на вкладке **Special**, существенно зависит от типа создаваемого графика (рис. 3.62).

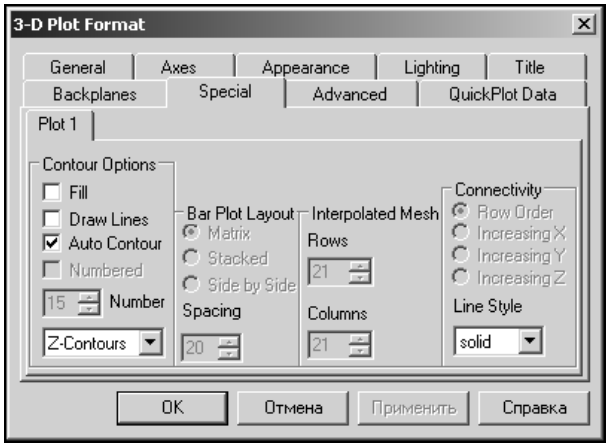


Рис. 3.62. Диалоговое окно **3-D Plot Format** открыто на вкладке **Special**

В данном случае доступных элементов не очень много, но они регулируют достаточно важные настройки. Тип контурных линий и линий основной сетки графика определяется в раскрывающемся списке **Line Style**. Это может быть сплошная (элемент **solid**), пунктирная (**dotted**), штрихованная (**dashed**) и штрихпунктирная (**da-dot**) линии. Опция **Fill** отвечает за закраску поверхности графика в соответствии с контурными линиями, опция **Draw Lines** активизируется для отображения контурных линий, а количество контурных линий определяется автоматически при установленном флажке опции **Auto Contour** (в противном случае его указывают в поле **Number**). По умолчанию контурные линии отображаются для вертикальной оси. Однако ситуацию можно изменить. Для этого в раскрывающемся списке в нижней части раздела **Contour Options** выбирают нужную ось (см. рис. 3.62). Контурные линии можно отображать сразу для нескольких осей. Широкие возможности предоставлены пользователю и при настройке способа отображения координатных плоскостей. Настройки выполняются на вкладке **Backplanes** (рис. 3.63), которая состоит из трех практически идентичных вкладок (различие только в названии).

Каждая такая вкладка, в свою очередь, состоит из двух разделов, разделенных на два подраздела. В верхней части вкладки размещено две опции и один элемент выбора цвета. Если установлен флажок опции **Fill Backplane**, координатная плоскость будет закрашена цветом, заданным с помощью элемента **Color**. Чтобы отображалась граница области отображения координатной плоскости, необходимо установить флажок опции **Backplane Border**.

В разделе **Grids** выполняются настройки для основных линий координатной сетки, а в разделе **Sub-Grids** — для вспомогательных линий. Подразделы этих разделов соответствуют разным координатным осям. Их элементы имеют одинаковое название и назначение. Флажок опции **Draw Lines** устанавливается для отображения линий координатной сетки. При активной опции можно выбрать цвет линии с помощью элемента **Line Color**, а толщина линии устанавливается в поле **Line Weight**. Наконец, для отображения засечек вдоль координатной оси активизируют опцию **Draw Ticks** (см. рис. 3.63).

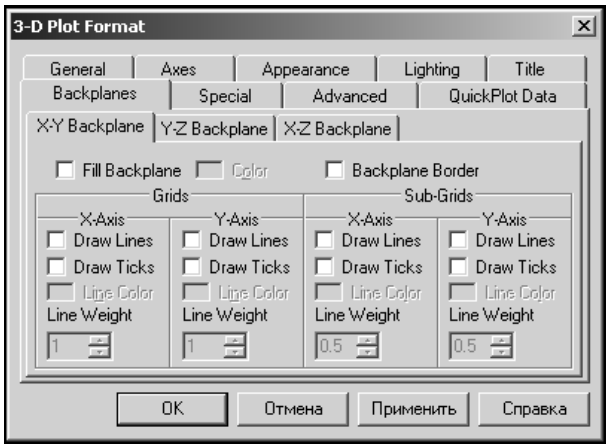


Рис. 3.63. Диалоговое окно **3-D Plot Format** открыто на вкладке **Backplanes**

Пример 3.11. Закраска цветом

Пример закраски координатных плоскостей цветом с одновременным отображением основных и вспомогательных линий координатной сетки представлен на рис. 3.64.

Хочется обратить внимание на то, что некоторые элементы на разных вкладках диалогового окна **3-D Plot Format** имеют если не одинаковое, то очень похожее назначение и часто бывает, что настройки, выполненные на одной вкладке, автоматически влияют на установки, сделанные ранее на других вкладках этого окна. Кроме того, совсем не обязательно ориентацию системы координат задавать в диалоговом окне **3-D Plot Format**. Обычно удобнее бывает развернуть нужным образом координатную систему с помощью мыши, благо, в Mathcad есть такая возможность.

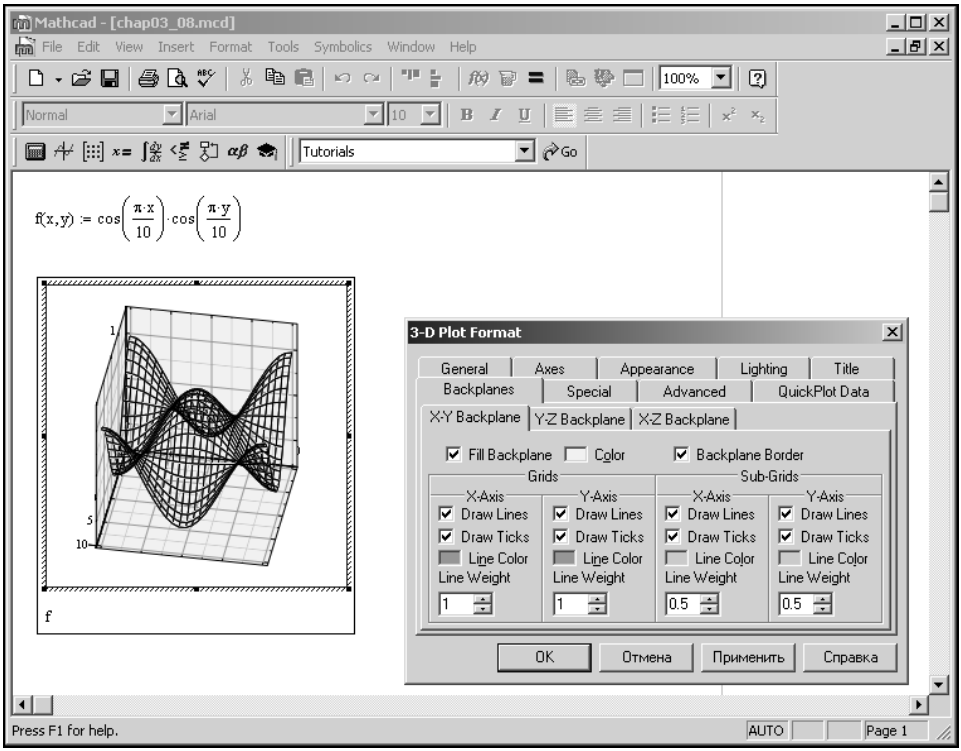


Рис. 3.64. Отображение основных и вспомогательных линий координатной сетки и закраска координатных плоскостей цветом

Отображение параметрических поверхностей

В предыдущем подразделе рассматривался пример функции двух переменных, заданной явно. Далеко не всегда существует возможность задать функциональную зависимость в явном виде. Самый элементарный пример — поверхность сферы. Обычно такая поверхность определяется в параметрическом виде. Это значит, что каждая из координат является функцией двух независимых параметров. Для всех трех координат эти параметры одни и те же. Поскольку параметров два, а координат — три, то естественным образом между этими координатами существует некоторая взаимозависимость, или функциональная связь. Такое положение дел, казалось бы, создает определенные трудности. С одной стороны, это действительно так. Но с другой, на основе параметрических зависимостей можно построить поверхности практически произвольной формы. В Mathcad предусмотрена возможность создания поверхностей, заданных в параметрическом виде.

Наиболее быстрый и простой способ отображения поверхности, заданной в параметрическом виде, состоит в следующем. Сначала определяются зави-

симости каждой из трех координат от переменных-параметров, после чего выбирается команда **Insert | Graph | Surface Plot** и в области создания графика вместо структурного заполнителя вводятся (в круглых скобках) три координаты, разделенные запятыми. Это именно те координаты, что предварительно определяются как функции двух параметров. Однако вводятся только их названия, без указания параметров-аргументов.

Пример 3.12. Эллипсоид

На рис. 3.65 приведен пример создания эллипсоида.

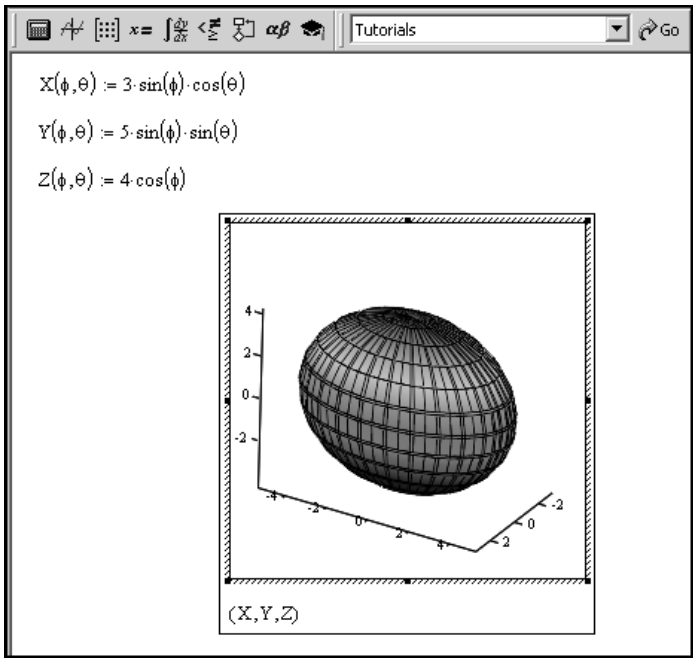


Рис. 3.65. Создание эллипса

Функциональные зависимости декартовых координат от параметров φ и θ в этом случае имеют такой вид: $x(\varphi,\theta) = 3\sin\varphi\cos\theta$, $y(\varphi,\theta) = 5\sin\varphi\sin\theta$ и $z(\varphi,\theta) = 4\cos\theta$. Именно эти зависимости и заданы в начальной части рабочего документа.

Настройка параметрического графика осуществляется практически так же, как и графика, построенного на основе явной зависимости. В случае создания эллипсоида, для того, чтобы он на экране имел правильную форму (а не выглядел как сфера), нужно установить флажок опции **Equal Scales** на вкладке

General диалогового окна **3-D Plot Format**. Прочие настройки существенного значения не имеют и достаточно очевидны.

Пример 3.13. Геликоид

В Mathcad с успехом строятся и более замысловатые поверхности, чем сфера или эллипсоид. На рис. 3.66 приведен пример создания поверхности геликоида.

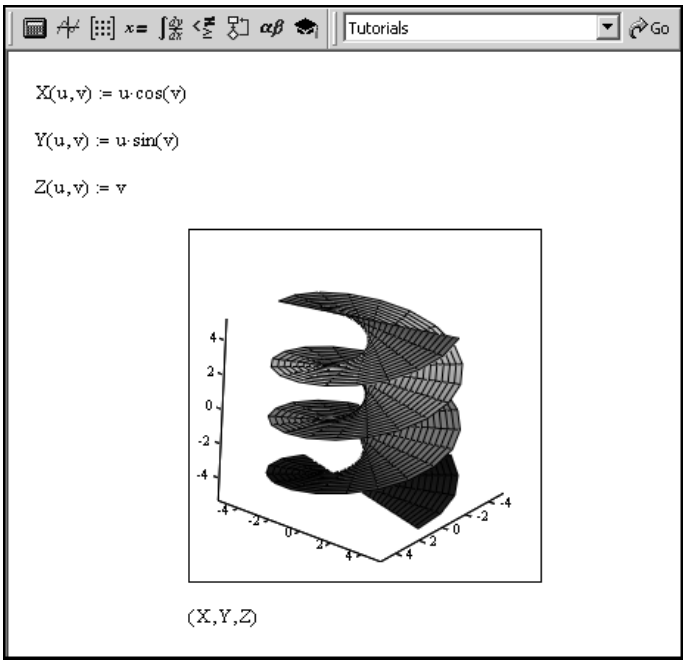


Рис. 3.66. Создание геликоида

В параметрическом виде соответствующая зависимость может быть представлена следующим образом: $x(u,v) = u\cos v$, $y(u,v) = u\sin v$ и $z(u,v) = v$, где u и v — параметры.

Практически так же создаются и пространственные кривые. В этом случае следует учесть два принципиальных отличия по сравнению с созданием параметрических поверхностей. Во-первых, каждая из трех координат зависит не от двух, а от одного параметра. Во-вторых, создается не поверхность, а кривая. Поэтому для создания такого графика выбирают команду **Insert | Graph | 3D Scatter Plot**. Можно, правда, воспользоваться и прежней командой **Surface Plot**, однако после создания графика нужно будет на вкладке **General** диалогового окна **3-D Plot Format** переключатель **Display As** установить в положение **Scatter Plot**.

Пример 3.14. Винтовая линия

На рис. 3.67 изображена коническая винтовая линия.

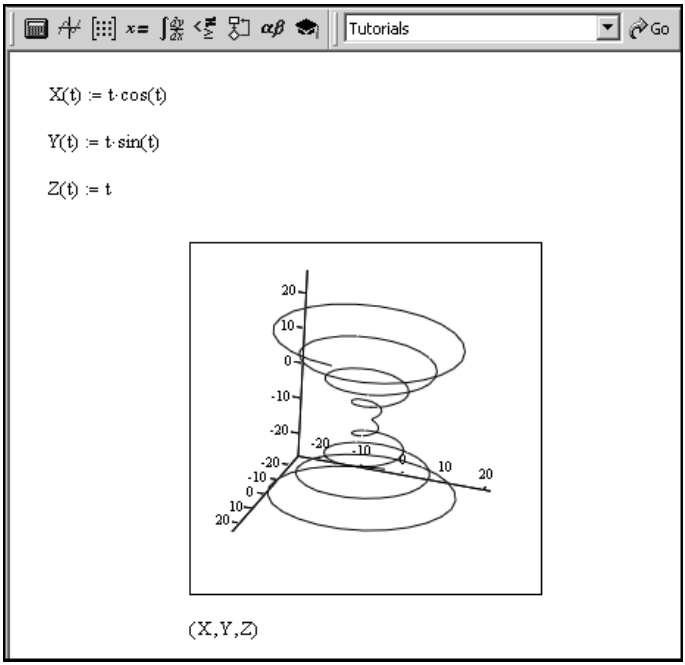


Рис. 3.67. Создание конической винтовой линии

При создании этой линии зависимость координат от параметра t была задана в виде $x(t) = t\cos t$, $y(t) = t\sin t$ и $z(t) = t$. Для того чтобы точки соединялись линией, переключатель в разделе **Line Options** на вкладке **Appearance** диалогового окна **3-D Plot Format** устанавливают в положение **Lines**, а если линия недостаточно плавная, увеличивают число узловых точек, по которым строится кривая (значение вводится в поле **# of Grids** на вкладке **QuickPlot Data** и должно быть не более 200).

Помимо определения функциональных зависимостей, графики можно строить по численным данным, или, другими словами, по точкам. Каждая из параметрических зависимостей координат от параметра реализуется в виде матрицы. Матрица является набором значений соответствующей координаты для различных значений параметров. Это, пожалуй, единственное отличие по сравнению с процедурой создания параметрической зависимости на основе алгебраических зависимостей. Далее выбирается команда **Insert | Graph | Surface Plot** и вместо структурного заполнителя вводится тройка координат, но только теперь это названия матриц.

Пример 3.15. Конус

На рис. 3.68 проиллюстрирована процедура создания параметрической зависимости (конуса) на основе узловых точек, которые занесены в матрицы.

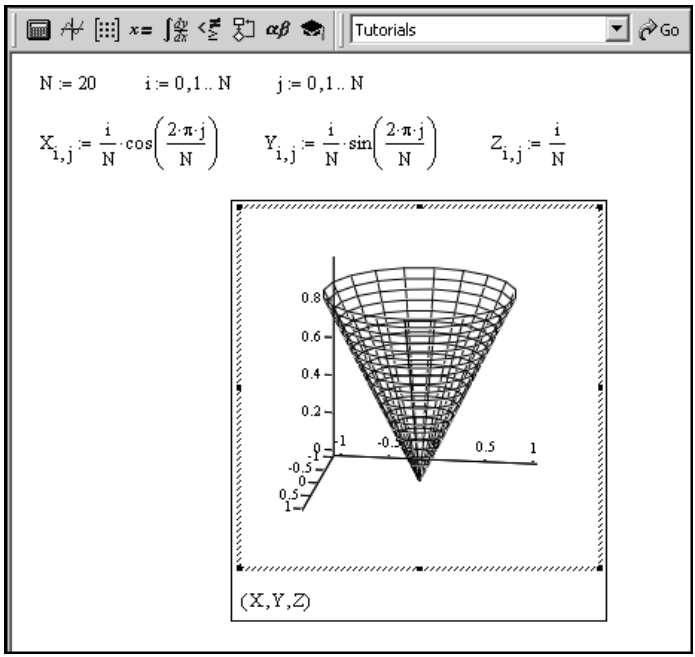


Рис. 3.68. Создание конуса

Число точек по каждому из параметров определяется переменной N (значение 20). Эта переменная определяет верхнюю границу изменения индексов i и j, которые пробегают целочисленные значения от 0 до N. Элементы координатных матриц определяются соотношениями $X_{i,j} = \frac{i}{N}\cos\left(\frac{2\pi j}{N}\right)$, $Y_{i,j} = \frac{i}{N}\sin\left(\frac{2\pi j}{N}\right)$ и $Z_{i,j} = \frac{i}{N}$, задавая тем самым контуры конуса. Настройка графика выполняется абсолютно аналогично рассматривавшимся ранее примерам.

Совершенно логично возникает вопрос о том, нельзя ли похожим образом строить пространственные кривые, т. е. задавать их в параметрическом виде не посредством уравнений, а набором точек. В Mathcad есть такая возможность. Несложно догадаться, что для создания пространственной кривой по точкам зависимость координат этой кривой от параметра реализуется в виде

вектора. Таким образом, необходимо создать три вектора, после чего воспользоваться командой **Insert | Graph | 3D Scatter Plot**, указать тройку векторов в качестве координат кривой и выполнить настройку графика (в первую очередь убедиться, что точки соединяются линией, в противном случае будут отображаться одни узловые точки).

Пример 3.16. Спираль

Результат выполнения этих процедур представлен на рис. 3.69, на котором можно видеть фрагмент спирали.

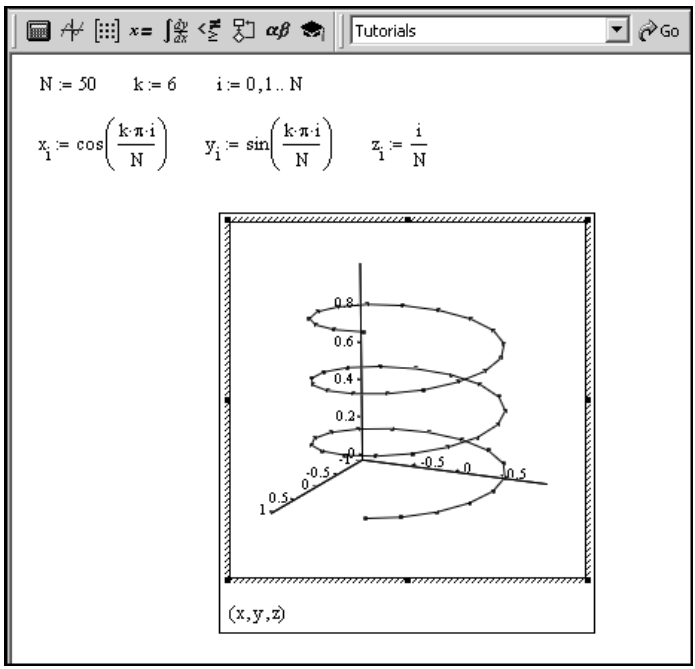


Рис. 3.69. Создание спирали

Число узловых точек определяется переменной N (значение 50). Меняя величину k (в рабочем документе равно 6), можно отображать разное количество витков спирали (при фиксированном N). Индекс, играющий роль переменной-параметра, пробегает целочисленные значения от 0 до N. Спиральная линия задается тремя векторами, элементы которых определяются через переменную-параметр такими соотношениями: $x_i = \cos\left(\frac{k\pi i}{N}\right)$, $y_i = \sin\left(\frac{k\pi i}{N}\right)$ и $z_i = \frac{i}{N}$.

Особенности построения трехмерных графиков

Трехмерные графики практически любого типа в Mathcad создаются точно так же, как и обычный график-поверхность. Исключение — график векторного поля. Во всех остальных случаях принципиальным моментом является выбор корректной команды из подменю **Graph**. Но даже если команда выбрана не та, что предполагалась с самого начала, ситуация вполне поправима. Как было показано ранее, в диалоговом окне **3-D Plot Format** на вкладке **General** можно изменить тип графика, установив правильным образом переключатель **Display As**. Разумеется, некоторые настройки специфичны для каждого конкретного типа графика. Однако большинство из них типовые. Например, при работе с контурными графиками обычно у контурных линий, определяющих профиль распределения функции, отображают численные значения, которые функция принимает на этих контурных линиях. В этом случае на вкладке **Special** диалогового окна **3-D Plot Format** устанавливают флажок опции **Numbered** (при работе с графиком-поверхностью эта опция недоступна).

Пример 3.17. Контурные графики

Результат выбора такого формата для графика показан на рис. 3.70.

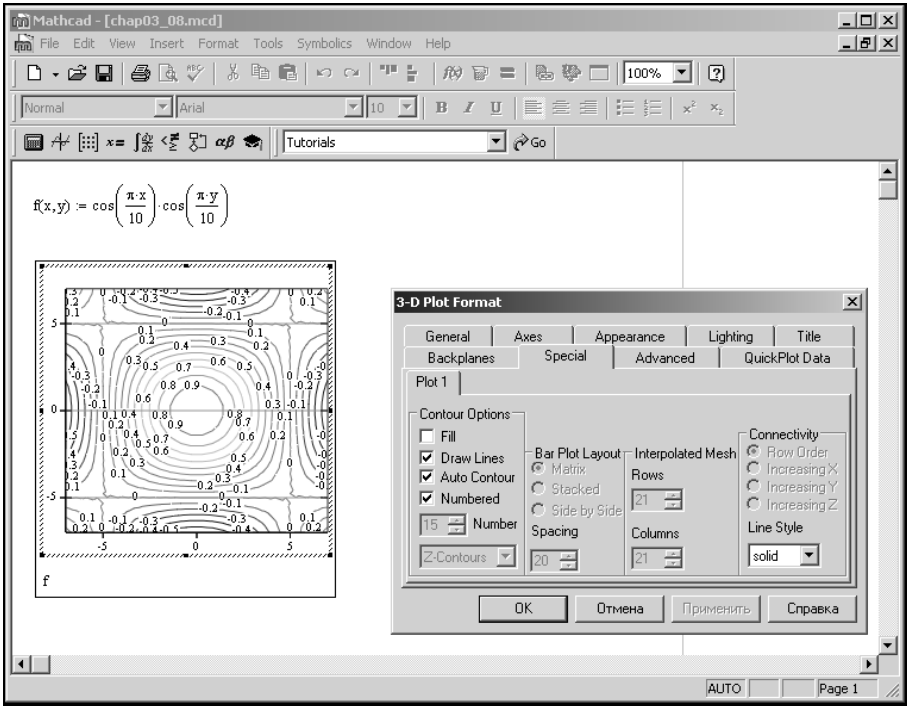


Рис. 3.70. Отображение численных значений функции на контурном графике

Реализация для контурного графика остальных настроек, описанных ранее применительно к поверхностям, также вполне очевидна. На рис. 3.71 представлен результат закраски графика различными цветами предопределенной палитры.

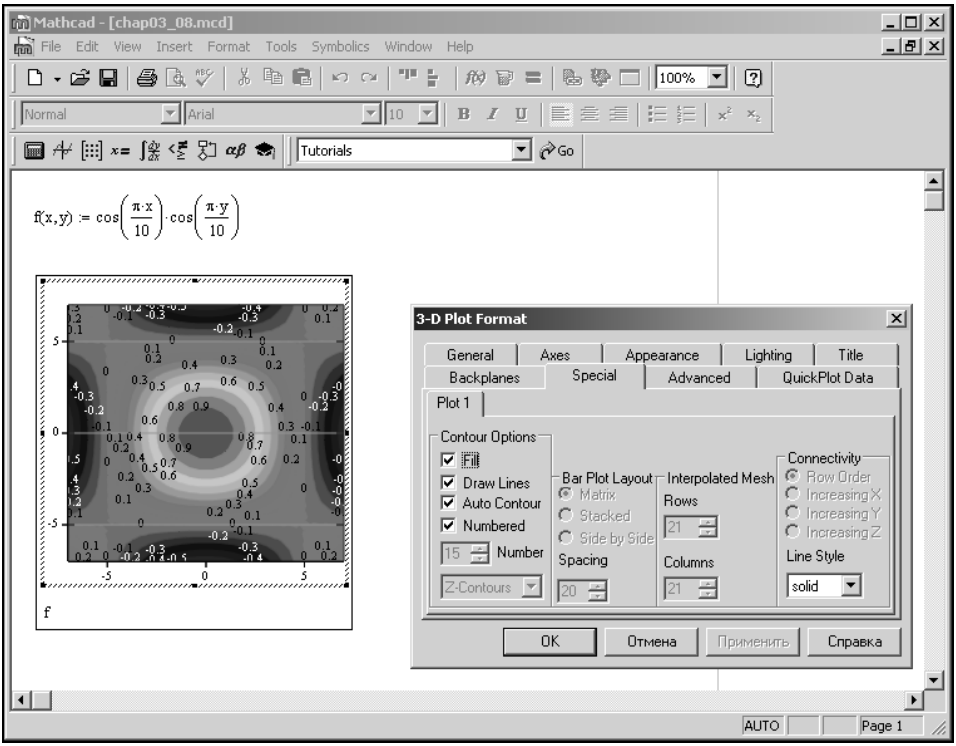


Рис. 3.71. Закраска области графика цветом

График векторного поля по сравнению с рассмотренными примерами создается несколько отличными методами. В первую очередь это относится к способу определения функции, для которой строится график. Точнее, график обычно строится на основе матрицы значений. Причем в данном случае на двумерном графике отображаются векторы. Вектор на плоскости, как известно, имеет два компонента. Один из способов задать сразу два компонента — прибегнуть к помощи комплексных чисел. Действительная и мнимая части комплексного числа могут интерпретироваться как компоненты вектора. Именно такой подход применяется в Mathcad при создании графиков векторных полей. При этом формируется комплексная матрица, т. е. матрица, элементы которой — комплексные числа. Название матрицы вводится вместо структурного заполнителя в области создания графика векторного поля.

Пример 3.18. Векторное поле

Пример такого графика приведен в документе на рис. 3.72.

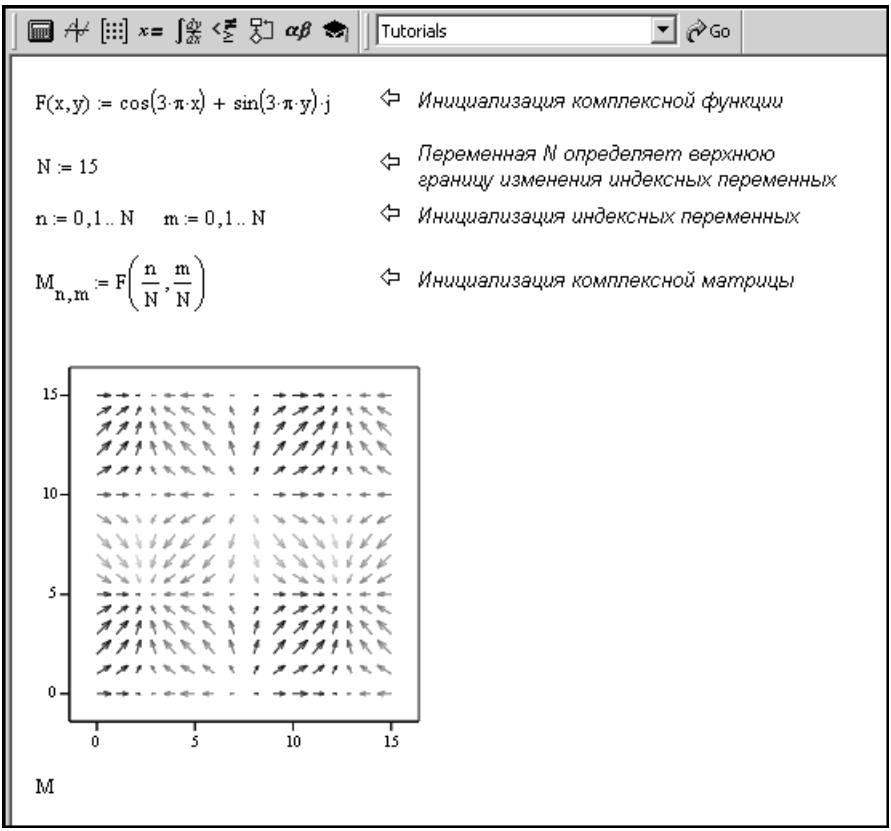


Рис. 3.72. График векторного поля

Сначала в рабочем документе определяется комплексная функция, на основе которой затем создается комплексная матрица. Однако при создании графика векторного поля можно обойтись и без комплексных чисел. При этом компоненты векторов задаются в виде двух разных матриц, а затем эти матрицы в скобках, разделенные запятой, указываются вместо структурного заполнителя в области создания графика. В документе на рис. 3.73 проиллюстрирован процесс создания того же графика, что и ранее (не применялись только настройки цветового выделения), но уже на основе двух действительных матриц.

Сначала создается векторная функция, т. е. массив из двух элементов, где каждый элемент есть функция двух аргументов. Далее создается вложенный массив. Это матрица, элементы которой определяются значением определен-

ной ранее векторной функции, т. е. элементы матрицы сами являются векторами. Наконец, формируются две матрицы: первая на основе первых компонентов векторов-элементов вложенного массива (компонент с нулевым индексом), а вторая — на основе вторых компонентов (они имеют индекс 1). Далее эти матрицы указываются в паре при создании графика векторного поля. Результат такой же, как и при одной комплексной матрице.

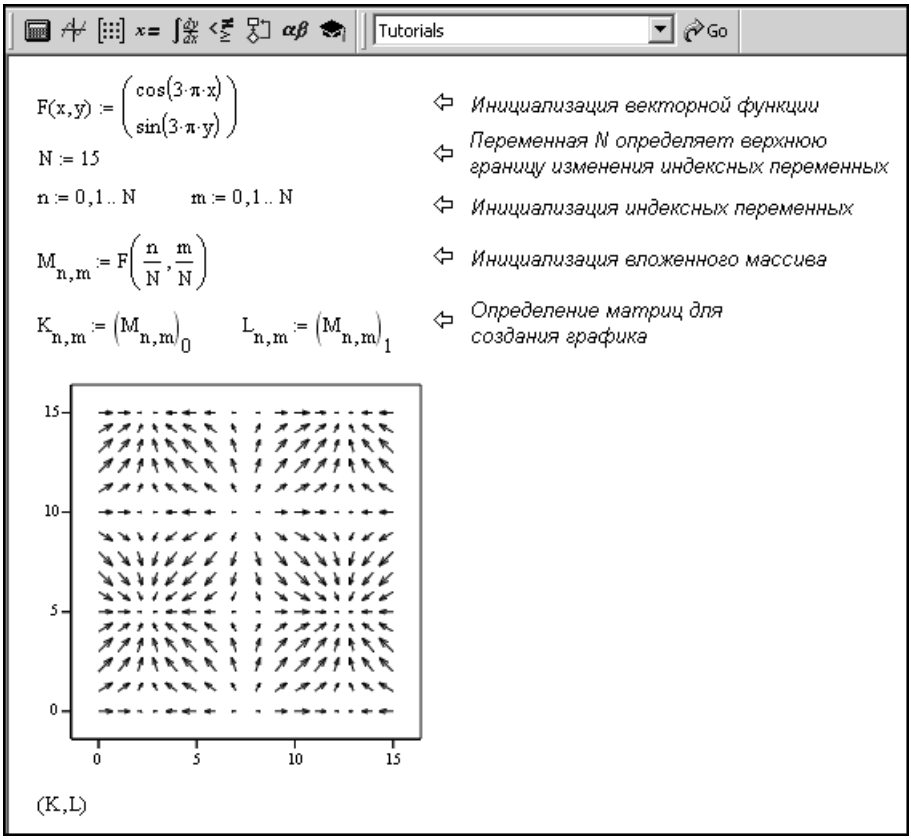


Рис. 3.73. Создание графика векторного поля с помощью векторной функции

Создание диаграмм

Данные, на основе которых в Mathcad могут строиться диаграммы, обычно реализуются в виде таблиц или матриц. Для построения диаграммы достаточно выбрать команду **Insert | Graph | 3D Bar Plot** и в области создания графика вместо структурного заполнителя указать название соответствующей матрицы или таблицы.

Пример 3.19. Диаграмма

На рис. 3.74 показан результат создания диаграммы на основе данных, взятых из таблицы.

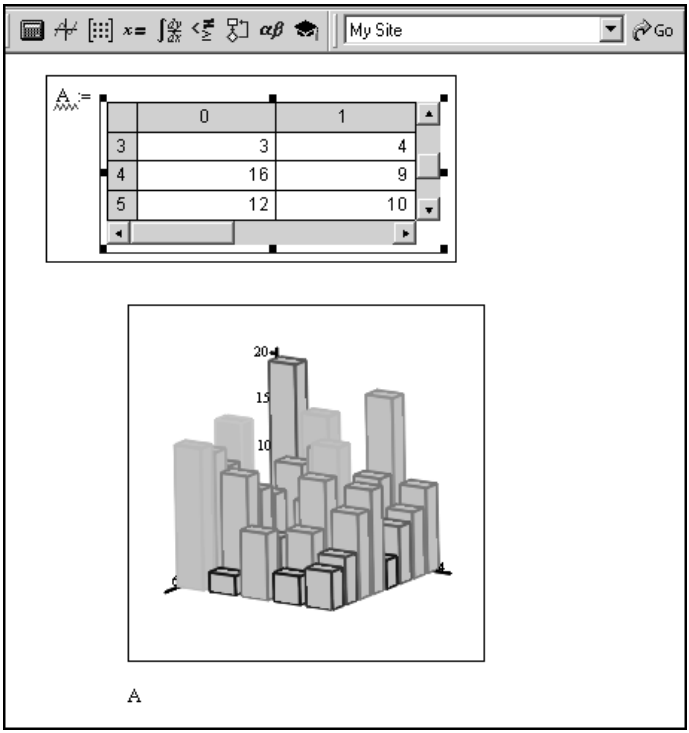


Рис. 3.74. Создание диаграммы на основе данных таблицы

Чтобы создать таблицу, следует выбрать команду **Insert | Data | Table** и заполнить ячейки таблицы. Для изменения размеров таблицы она выделяется, и с помощью мыши ее границы перемещаются до нужных пределов. Кроме самой таблицы, при выборе упомянутой команды в рабочий документ вставляется структурный заполнитель (после которого идет оператор присваивания и непосредственно таблица), вместо которого для создаваемой таблицы указывается название.

Практически так же диаграмма создается на основе матрицы. Сначала в документе определяется матрица, после чего ее название вводят вместо структурного заполнителя в области создания графика. Пример показан на рис. 3.75.

Как для таблицы, так и для матрицы, на диаграмме вдоль вертикальной координатной оси откладываются фактические значения из матрицы или таблицы, а вдоль горизонтальных — индексы соответствующих элементов матрицы

или таблицы. Настройки диаграммы выполняются с помощью диалогового окна **3-D Plot Format**, которое описывалось ранее. Правда, большинство настроек в случае работы с диаграммой недоступно. Открыть это диалоговое окно можно, помимо прочего, щелкнув правой кнопкой мыши в области диаграммы и выбрав из раскрывающегося списка команду **Properties**.

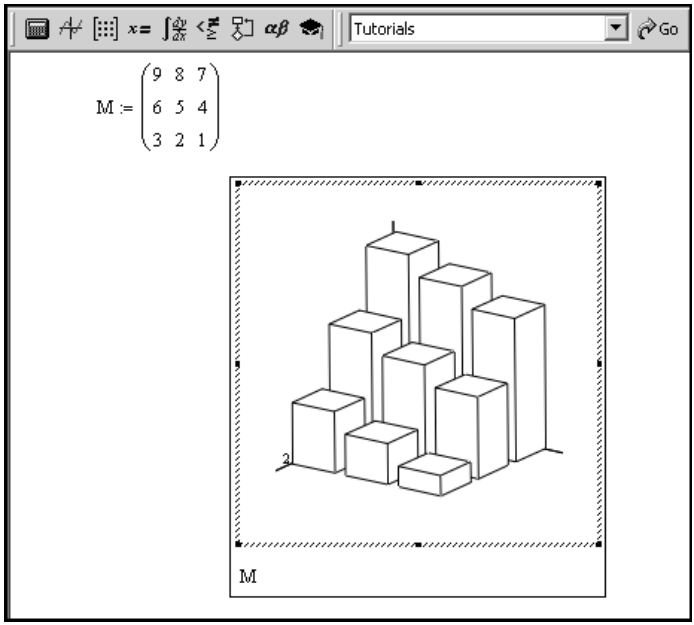


Рис. 3.75. Создание диаграммы на основе матрицы

Отображение нескольких графиков

Иногда возникает необходимость отображать на графике сразу несколько поверхностей. Рецепт в этом случае достаточно прост: структуры, на основе которых строятся поверхности (это могут быть названия функций, матицы или тройки матриц, заключенные в круглые скобки), вводятся через запятую вместо структурного заполнителя в области создания диаграммы. Далее приведены некоторые простые примеры построения нескольких поверхностей, кривых и диаграмм на одном графике.

Пример 3.20. Поверхности

На рис. 3.76 представлен фрагмент рабочего документа Mathcad, в котором описываются две функции от двух переменных, после чего на графике строятся

соответствующие поверхности (параболоид вращения $f(x, y) = x^2 + y^2$ и плоскость $F(x, y) = x + y$).

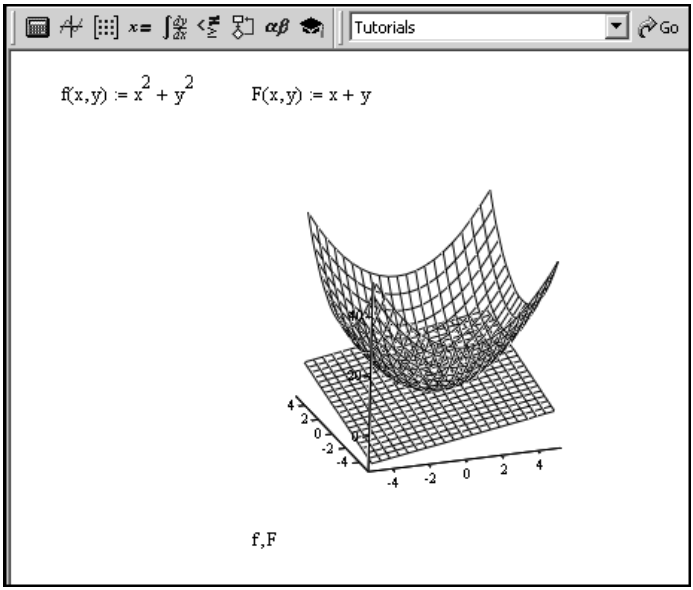


Рис. 3.76. Параболоид вращения и плоскость

Для построения поверхностей выбиралась команда **Insert | Graph | Surface Plot**. Вместо структурного заполнителя через запятую вводятся названия функций.

Пример 3.21. Поверхность и диаграмма

Совсем не обязательно, чтобы поверхности задавались одинаковым образом. Более того, нет необходимости каждую зависимость отображать в виде поверхности. Так, на рис. 3.77 одна зависимость задана функцией, а вторая (точнее, данные для отображения) — с помощью матрицы.

По данным, записанным в матрицу, строится диаграмма. Так зависимость, что определена функцией ($F(x, y) = 0,5x + 0,1y + 5$), соответствует плоскости, которая также отображена на рис. 3.77. В этом случае в области создания графика вместо структурного заполнителя через запятую указываются название матрицы и функции. Далее следует выполнить ряд настроек в диалоговом окне **3-D Plot Format**. Если на графике отображена более чем одна зависимость, это диалоговое окно содержит по нескольку вкладок в тех местах, где выполняется настройка параметров, относящихся к способу отображения конкретной зависимости. Например, на рис. 3.77 на вкладке **General** в нижней

части, где выбирается способ отображения зависимости, уже две, а не одна вкладка, как ранее. Вкладки называются **Plot 1** и **Plot 2**, по количеству зависимостей и в соответствии с порядком их следования в месте структурного заполнителя в области создания графика. Так вот, чтобы данные, собранные в матрице, представлялись в виде диаграммы, необходимо на вкладке **Plot 1** переключатель **Display As** установить в положение **Bar Plot**. Многие другие настройки также могут быть выполнены отдельно для каждой из зависимостей.

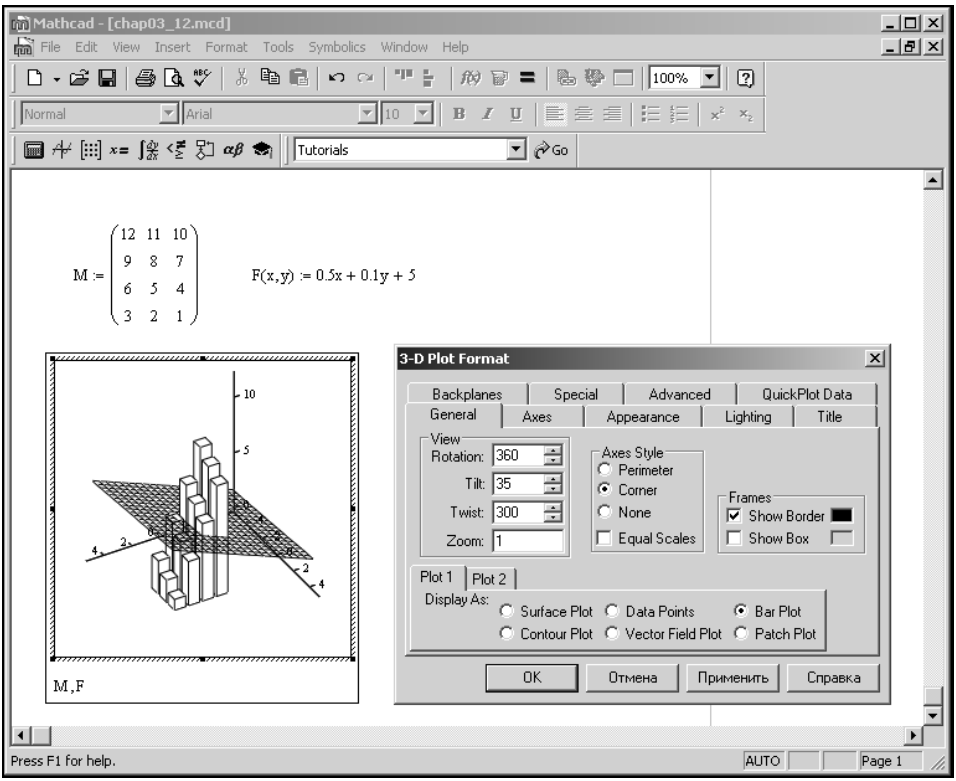


Рис. 3.77. Диаграмма и плоскость

Пример 3.22. Конус и сфера

При построении сложных пространственных фигур часто прибегают к параметрическому способу описания поверхностей, как это сделано в документе, фрагмент которого изображен на рис. 3.78.

В этом случае каждая поверхность (а это сфера радиусом 2 и симметричная коническая поверхность с точкой инверсии в начале координат) задана

в параметрическом виде. Координаты конической поверхности определяются через параметры u и v следующим образом: $x(u,v) = u\cos v$, $y(u,v) = u\sin v$ и $z(u,v) = u$. Сфера задается так: $X(u,v) = a\sin v\cos u$, $Y(u,v) = a\sin v\sin u$ и $Z(u,v) = a\cos v$, где радиус сферы определяется переменной a, которой предварительно присвоено значение 2. Каждая поверхность задается заключенной в скобки тройкой из названий координат (точнее, названий функций, определяющих зависимость координат от параметров).

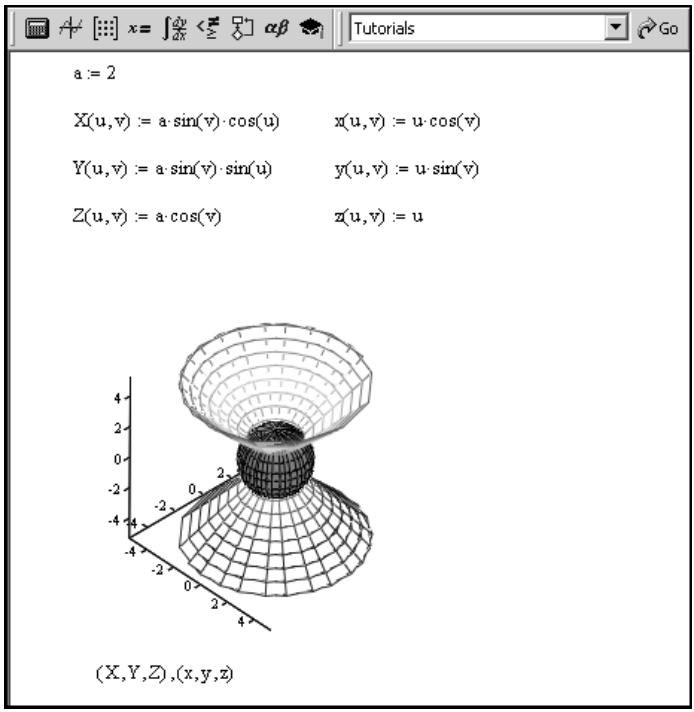


Рис. 3.78. Сфера и конусы

Пример 3.23. Тень от поверхности

Иногда бывает полезно на одном графике дважды отобразить одну и ту же зависимость, но в разном виде. Такой пример представлен на рис. 3.79.

На этом графике изображена поверхность параболоида вращения (функция $f(x,y) = x^2 + y^2$). Несложно заметить, что название функции дважды через запятую указано в месте структурного заполнителя, а кроме поверхности параболоида отображен еще и контурный график. Добиться

такого эффекта можно, установив разный способ отображения зависимости. Для одной это будет поверхность, для другой — контурный график. Следует обратить внимание, что для разных зависимостей допускается устанавливать разные диапазоны изменения переменных, что и было сделано в этом примере. В результате контурный график занимает бóльшую площадь, чем проекция в плоскость переменных части параболоида вращения, построенной на рис. 3.79.

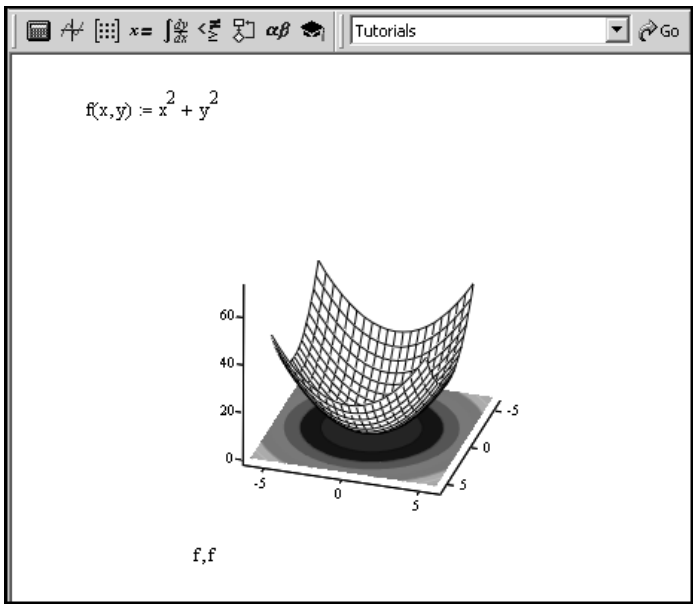


Рис. 3.79. Отображение одной зависимости в разном виде

Пример 3.24. Пространственные кривые

Практически так же отображаются на одном рисунке несколько пространственных кривых. На рис. 3.80 изображены винтовая кривая и прямая линия.

Винтовая кривая задается в параметрическом виде: $x(t) = 5\cos t$, $y(t) = 6\sin t$ и $z(t) = t$, где t параметр. Прямая линия задается тремя векторами с координатами точек прямой: $X_i = 0,5i$, $Y_i = 0,1i$, $Z_i = i$, а индексная переменная i принимает значения в диапазоне от 0 до 20. После создания графика были выполнены дополнительные настройки (точки соединяются линиями, установлен режим единого масштаба для координатных осей и т. д.).

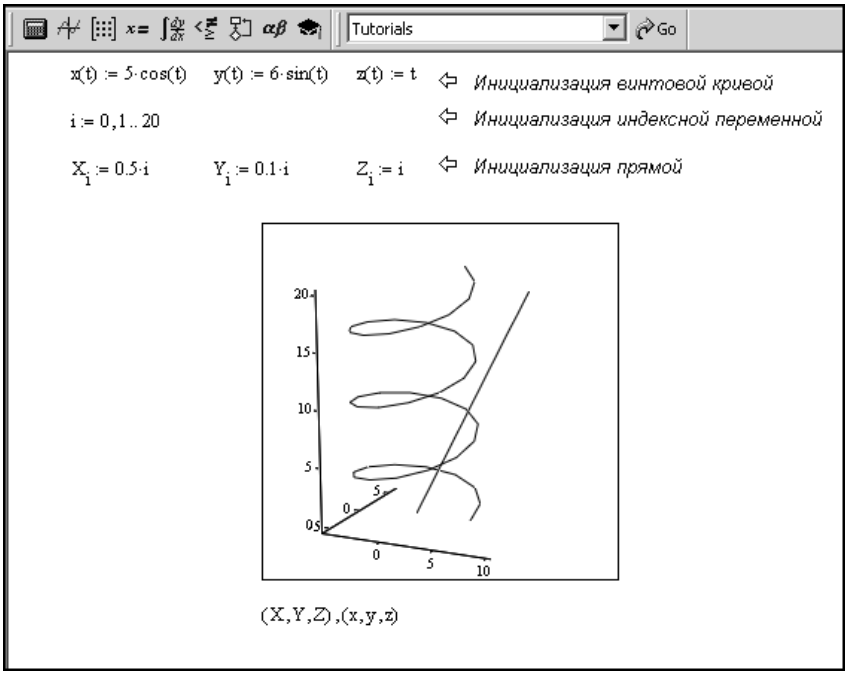


Рис. 3.80. Две пространственные кривые

Работа с утилитой создания графиков

Удобный способ создания трехмерных графиков — использование специальной утилиты Mathcad, которая запускается выбором команды **Insert | Graph | Plot Wizard**. В результате открывается диалоговое окно **Plot Type**, показанное на рис. 3.81.

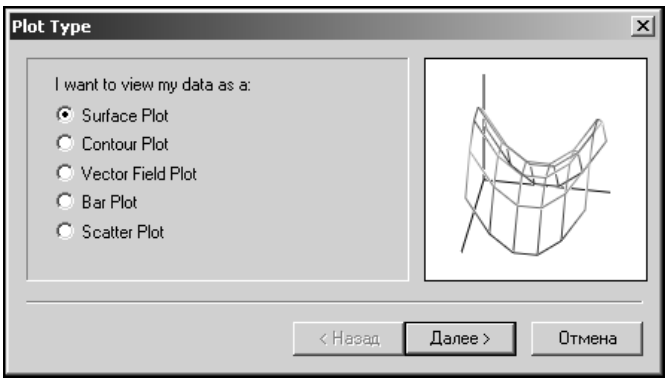


Рис. 3.81. Диалоговое окно **Plot Type**

В этом диалоговом окне выбирается тип создаваемого графика. После того как выбор сделан (переключатель установлен в правильное положение), выполняется щелчок на кнопке **Далее**. Открывается диалоговое окно **Appearance** (рис. 3.82).

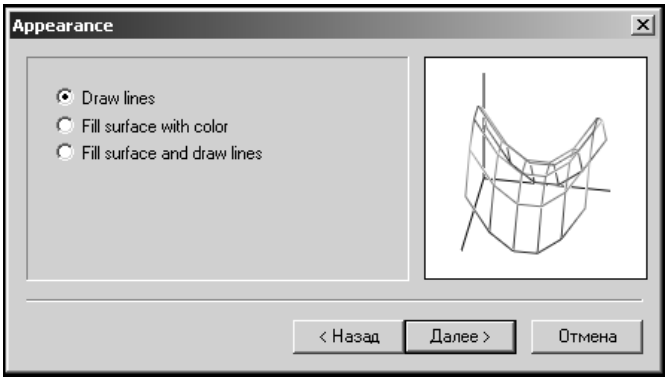


Рис. 3.82. Диалоговое окно **Appearance**

Окно позволяет настроить способ отображения графика. В зависимости от выбранного типа графика, элементы в диалоговом окне **Appearance** могут несколько отличаться от тех, что представлены на рис. 3.82. Затем, если тип графика позволяет, выполняются некоторые цветовые настройки в диалоговом окне **Coloring** (рис. 3.83).

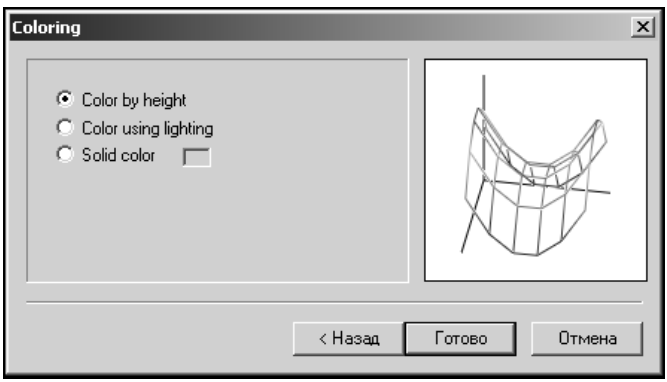


Рис. 3.83. Диалоговое окно **Coloring**

Для подтверждения сделанных настроек щелкают на кнопке **Готово**. В противном случае можно перейти к предыдущему диалоговому окну с помощью кнопки **Назад**. Таким образом, в последовательно открывающихся диалоговых

окнах можно выполнить все основные настройки для создаваемого графика — те настройки, что обычно выполняются в диалоговом окне **3-D Plot Format**. Назначение элементов диалоговых окон при этом такое же, как и одноименных элементов на вкладках в окне **3-D Plot Format**.

Работа с изображениями

Помимо графиков функциональных зависимостей, в процессе работы часто приходится иметь дело с различными изображениями, в том числе экспортируемыми из внешних источников данных. Для вставки рисунка в документ Mathcad выбирается команда **Insert | Picture** (или нажимается комбинация клавиш <Ctrl>+<T>), в результате чего в рабочий документ вставляется область отображения рисунка, в которой всего один структурный заполнитель, куда вводится название переменной, которой в качестве значения присвоено изображение, или название файла (заключенное в двойные кавычки), из которого экспортируется рисунок.

Пример 3.25. Вставка изображения

На рис. 3.84 в документе отображается файл Girl.jpg, размещенный в каталоге D:\MCAD. Указан полный путь к файлу, а название файла приведено без расширения.

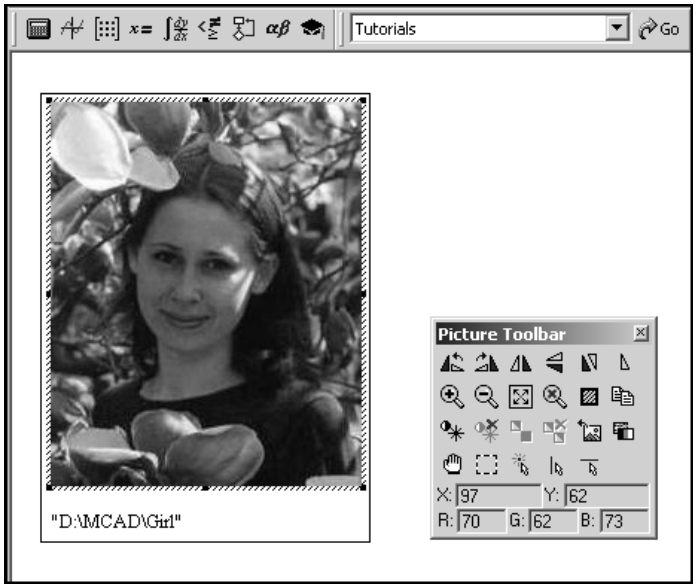


Рис. 3.84. Отображение в рабочем документе рисунка

Если рисунок выделен, кроме него отображается еще и панель **Picture Toolbar** для работы с изображениями. Панель содержит достаточно много кнопок, назначение которых кратко описано в табл. 3.5.

Таблица 3.5. *Кнопки панели* ***Picture Toolbar***

Кнопка	Описание
	Поворот изображения на 90° против часовой стрелки
	Поворот изображения на 90° по часовой стрелке
	Зеркальное отображение изображения относительно вертикали
	Зеркальное отображение изображения относительно горизонтали
	Транспонирование изображения (отображение относительно главной диагонали)
	Восстановление способа ориентирования изображения
	Увеличение масштаба отображения рисунка
	Уменьшение масштаба отображения рисунка
	Выбор масштаба отображения такого, чтобы оно соответствовало области окна рисунка
	Восстановление исходного масштаба отображения рисунка
	Выделение всего изображения
	Копирование выделенного фрагмента изображения в буфер обмена
	Настройка контраста изображения
	Восстановление настроек контраста изображения
	Настройка карты оттенков серого цвета изображения
	Восстановление настроек карты оттенков серого цвета для изображения
	Восстановление всех настроек отображения
	Обновить изображение
	Переход в режим перемещения изображения с помощью мыши
	Переход в режим выделения фрагмента изображения
	Переход в режим выбора отдельного пиксела в области изображения
	Переход в режим выбора вертикального ряда пикселов в области изображения

Таблица 3.5 *(окончание)*

Кнопка	Описание
	Переход в режим выбора горизонтального ряда пикселов в области изображения
X: 261	В поле отображается горизонтальная координата точки изображения, в которой на текущий момент находится курсор мыши
Y: 4	В поле отображается вертикальная координата точки изображения, в которой на текущий момент находится курсор мыши
R: 92	В поле отображается значение красной составляющей в матрице изображения для той его точки, в которой на текущий момент находится курсор мыши
G: 92	В поле отображается значение зеленой составляющей в матрице изображения для той его точки, в которой на текущий момент находится курсор мыши
B: 92	В поле отображается значение синей составляющей в матрице изображения для той его точки, в которой на текущий момент находится курсор мыши

Используя панель **Picture Toolbar**, можно существенно изменить способ отображения рисунка в рабочем документе. Однако при реализации серьезных проектов желательно иметь доступ к более или менее надежным характеристикам изображения. Во всяком случае, на упомянутую панель рассчитывать можно не всегда.

Изображение можно вставлять в рабочий документ с помощью ссылки, как было описано ранее, если соответствующий файл имеет один из форматов BMP, GIF, JPG, PCX или TGA. Информацию о рисунке можно получить из матрицы (таблицы) значений пикселов. Чтобы вычислить эту матрицу, для цветных изображений используют функцию `READRGB()`, а для черно-белых — `READBMP()`. Аргумент обеих функций — название файла изображения.

Пример 3.26. Считывание изображения

На рис. 3.85 матрица значений пикселов изображения вычислена с помощью функции `READRGB()`.

Проверить значение матрицы можно, указав после ее названия знак равенства. На рис. 3.85 соответствующая таблица показана в свернутом виде, поэтому чтобы просмотреть ее полностью (а она достаточно большая), ее следует выделить, а затем воспользоваться полосами прокрутки. Если указать вычисленную матрицу вместо структурного заполнителя, то для черно-белого изображения будет отображен исходный рисунок. Однако в том случае, если

матрица вычислялась для цветного изображения (как в рассмотренном примере), оно будет отображено трижды подряд, причем в черно-белом цвете (см. рис. 3.85). Дело в том, что цвет каждого пиксела в цветном изображении определяется тремя числами, определяющими долю красного, зеленого и синего цветов (*RGB* — это сокращение от *Red*, *Green* и *Blue*, то есть *красный*, *зеленый* и *синий*). Поэтому вычисленная матрица для цветного изображения состоит, фактически, из трех матриц, поэтому изображений тоже три. Чтобы по такой матрице построить исходное цветное изображение, необходимо выделить из нее три подматрицы, соответствующие красному, зеленому и синему цветам, и указать эти матрицы через запятую в месте структурного заполнителя области отображения рисунка.

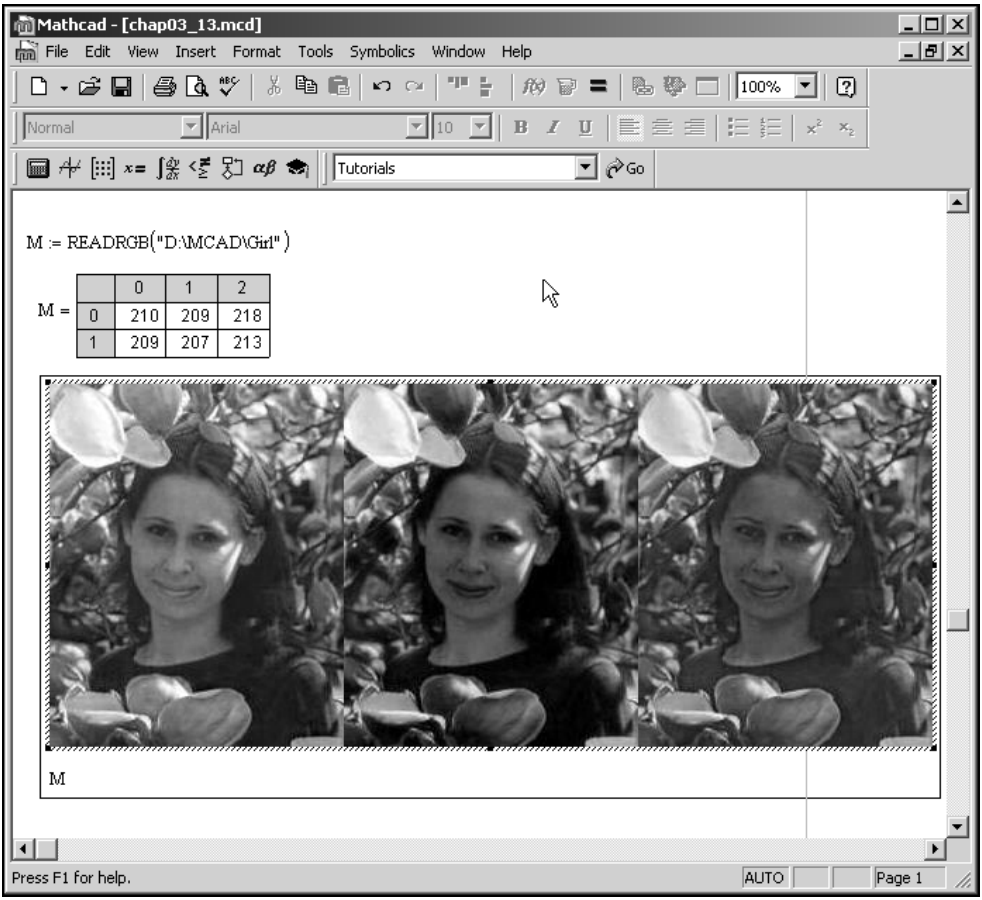


Рис. 3.85. Вычисление матрицы изображения

Подматрица может быть выделена из матрицы с помощью функции `submatrix()`, аргументами которой последовательно указываются матрица, из которой извлекается подматрица, индекс начальной строки подматрицы, индекс конечной строки подматрицы, индекс начального столбца подматрицы и индекс конечного столбца подматрицы. Число столбцов матрицы определяется функцией `cols()`, а строк — `rows()`. Матрица для красного цвета присваивается в качестве значения переменной `R`, матрица для зеленого цвета присваивается как значение переменной `G`, а значением переменной `B` является матрица для синего цвета. Следует отметить, что если просто указать одну из этих матриц, например `R`, вместо структурного заполнителя, то изображение будет черно-белым (рис. 3.86).

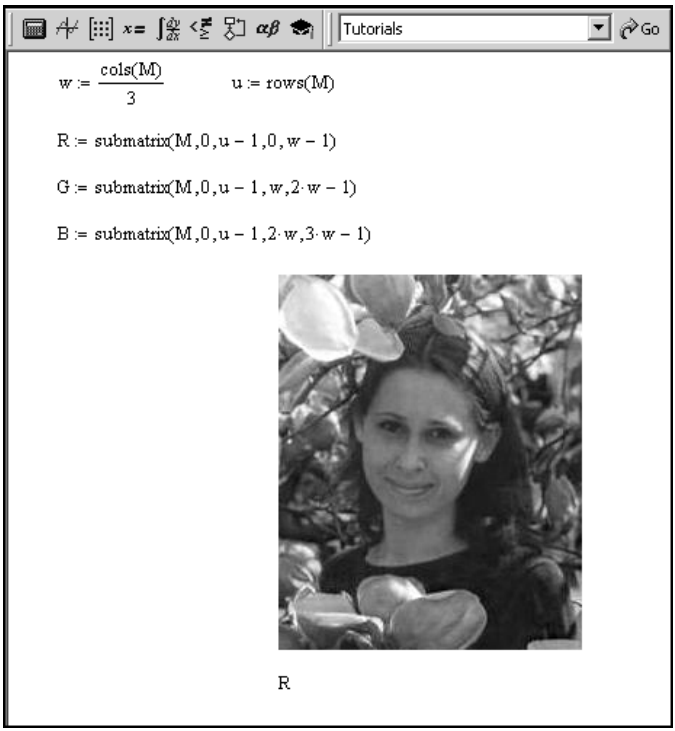


Рис. 3.86. Черно-белое изображение

Как отмечалось, цветное изображение строится, если вместо структурного заполнителя вводятся названия всех трех матриц, причем в соответствующем порядке (рис. 3.87).

Первая матрица дает сведения о доле красного цвета в изображении, вторая — о доле зеленого и третья — синего цвета. Поэтому если поменять порядок матриц, расцветка изменится.

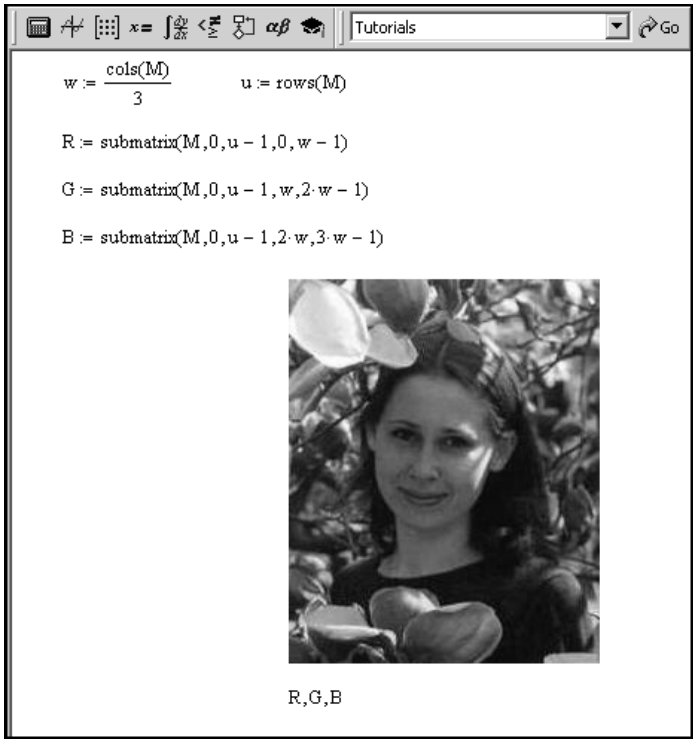


Рис. 3.87. Создание цветного изображения

Описанный способ извлечения подматриц из матрицы цветного изображения может использоваться для создания его черно-белых копий, причем есть возможность выделять только определенный цвет спектра. Например, воспользовавшись функцией `WRITEBMP()`, можно создать новый файл с черно-белым изображением. Название создаваемого файла (полный путь) указывается аргументом функции, после чего вводится оператор присваивания и далее переменная (матрица значений), на основе которой создается файл с изображением. На рис. 3.88 приведена соответствующая команда.

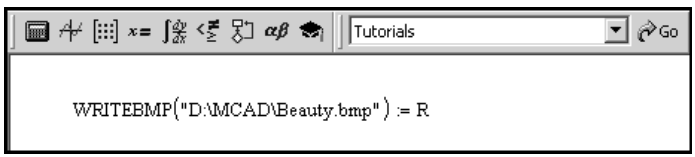


Рис. 3.88. Создание файла изображения

Для создания цветных изображений предусмотрена функция `WRITERGB()`. Как несложно догадаться, извлекая подматрицы из матриц изображений, можно

копировать тем самым фрагменты исходного изображения. Например, из созданной ранее матрицы R выделяется подматрица N (индексы ее начальной и конечной строк 40 и 200, а столбцов — 30 и 170), и изображение, построенное на основе этой матрицы (рис. 3.89), очевидно, является фрагментом исходного, построенного на основе матрицы R (см. рис. 3.86).

Рис. 3.89. Копирование фрагмента изображения и создание зеркального отражения

Если в функции submatrix() начальный и конечный индексы строк или столбцов подматрицы указать в обратном порядке (то есть сначала больший индекс, а затем меньший), данные в строках или столбцах подматрицы также будут следовать в обратном порядке, и в результате изображение отображается зеркально (относительно горизонтали или вертикали — в соответствии с тем, для строк или столбцов изменен порядок индексов). На рис. 3.89 создана матрица P, практически такая же, как матрица N, только при ее создании в обратном порядке были указаны индексы столбцов при извлечении подматрицы. Изображения, созданные на основе матриц P и N, являются зеркальными.

Пример 3.27. Создание изображений

Таким образом, создав матрицу численных значений, в дальнейшем на ее основе можно построить изображение, причем элементами матрицы могут быть нецелые числа. В документе на рис. 3.90 приведен пример создания такой

матрицы. Черно-белое изображение, формируемое на ее основе, напоминает дифракционную картину.

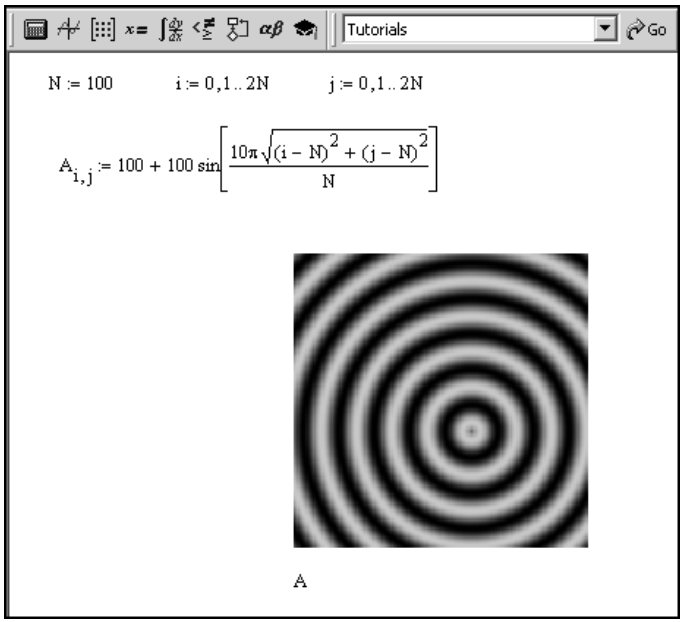


Рис. 3.90. Создание рисунка на основе матрицы

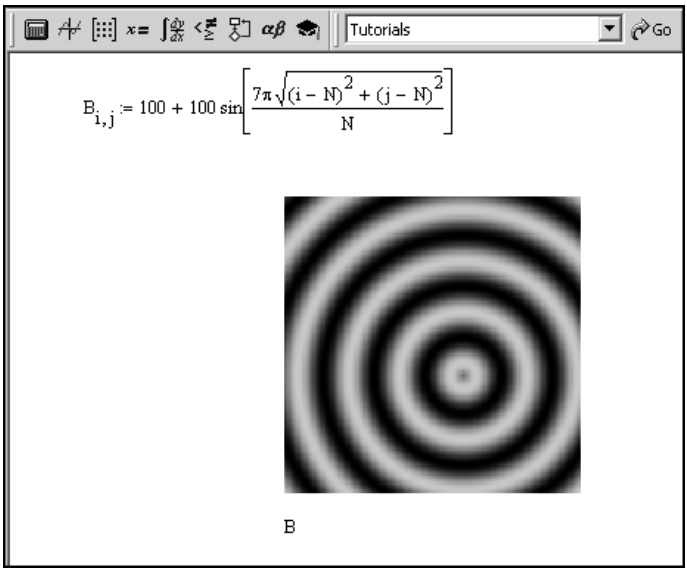


Рис. 3.91. Второе изображение

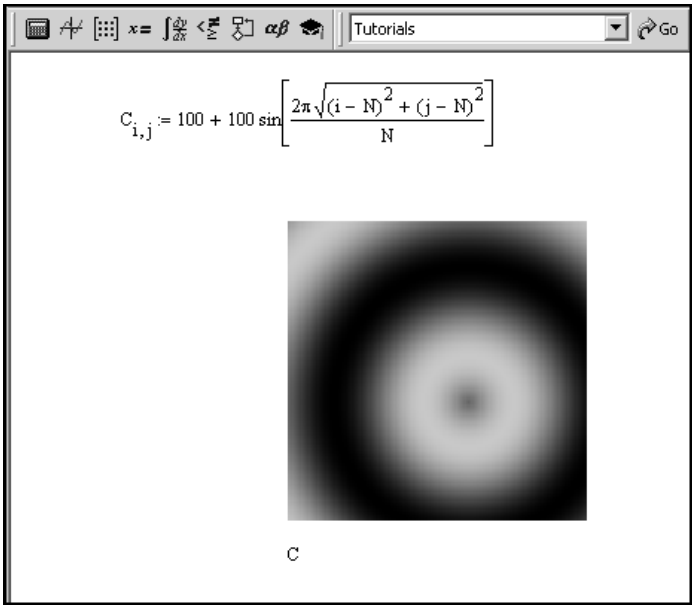


Рис. 3.92. Третье изображение

Далее создаются еще две матрицы. По сравнению с первой матрицей, принципиальное отличие состоит в коэффициенте у аргумента синуса. В результате соответствующие изображения имеют разный масштаб. Второе изображение с командой определения матрицы показано на рис. 3.91, а третье — на рис. 3.92.

Указав все три матрицы на месте структурного заполнителя в области создания рисунка, можно создать цветную картинку с круговыми линиями различного цвета (рис. 3.93).

Выполняя различные преобразования с матрицами изображений, можно добиваться самых разных эффектов, в том числе и достаточно нетривиальных. Например, можно выполнить инверсию цвета в черно-белом изображении. Черный цвет определяется значением 0, а белый — 255. Если исходное изображение определяется матрицей `A`, то, очевидно, для инверсии цветовой палитры следует рассмотреть матрицу `D = 255 - A` (согласно этому соотношению элементы матрицы `D` получаются вычитанием из числа 255 соответствующих элементов матрицы `A`). Результат показан на рис. 3.94.

Таким образом, в Mathcad можно создавать практически любые изображения. На рис. 3.95 приведен пример построения простой черно-белой картинки: белый круг на фоне черного квадрата.

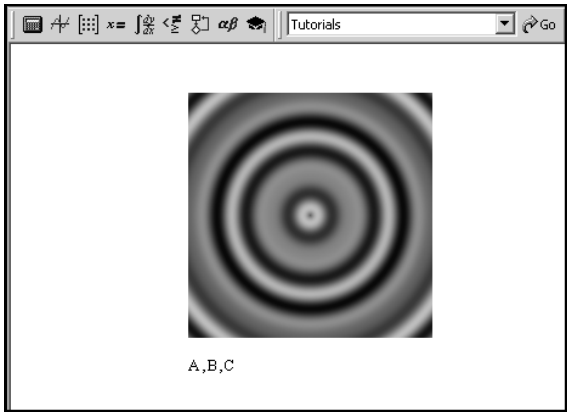


Рис. 3.93. Цветное изображение

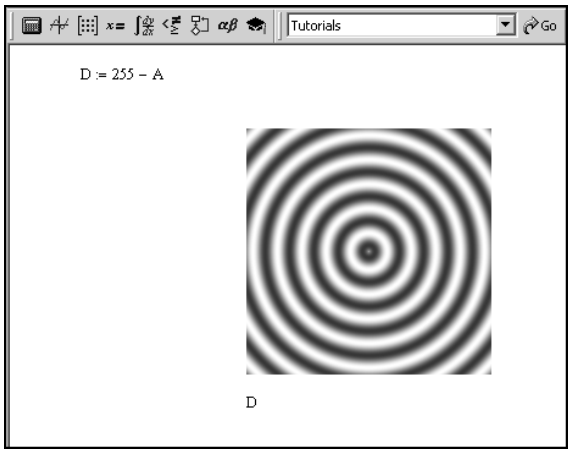


Рис. 3.94. Инверсия цветовой палитры

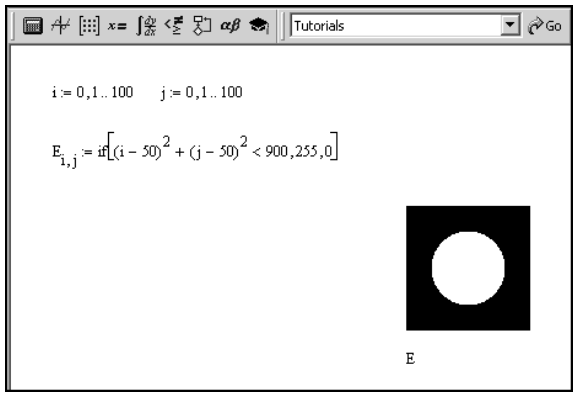


Рис. 3.95. Создание простой картинки

При определении базовой матрицы изображения была использована функция `if()`, у которой три аргумента: проверяемое условие (в данном случае это $(i-50)^2+(j-50)^2<900$, где индексные переменные i и j принимают значения в диапазоне от 0 до 100, и поэтому соответствующее условие задает внутренние точки круга радиуса в 30 пикселов, размещенного в центре квадрата размерами 101×101 пиксел), значение, возвращаемое функцией при выполнении условия (255, что соответствует белому цвету), а также значение, возвращаемое функцией при невыполнении условия (0, то есть черный цвет). Обратим внимание на два обстоятельства, связанных со способом формирования цветовой палитры на основе матрицы значений. Во-первых, уже отмечалось, что в этой матрице элементы могут быть нецелыми числами. В этом случае перед созданием изображения элементы матрицы округляются. Во-вторых, основные оттенки серого цвета задаются целыми числами в диапазоне от 0 (черный) до 255 (белый). Если число в матрице значений не попадает в указанный диапазон, то цвет определяется числом, равным остатку от деления элемента матрицы на 256.

Анимация

Анимация обычно создается для различного рода зависимостей, в которых присутствует временная переменная. Другими словами, в процессе создания анимации генерируются несколько изображений, которые последовательно выводятся в области отображения графики. Сразу следует подчеркнуть, что в анимационные картинки могут быть включены не только графики функциональных зависимостей, но и различного рода выражения, которые меняют свое значение со временем. Непременное условие успешного создания анимации в Mathcad — выбор в качестве временного параметра переменной `FRAME`. Это системная переменная, и как-то определять или инициализировать ее в рабочем документе нет необходимости. Просто вместо переменной, которая символизирует время, следует всегда указывать `FRAME`. Процедура создания анимации в Mathcad подразумевает запись анимационного клипа, который, при условии его сохранения, в последующем может просматриваться вне зависимости от содержимого рабочего документа, в котором этот клип был записан. Рассмотрим простой пример.

Пример 3.28. Создание анимации

На рис. 3.96 показан фрагмент рабочего документа, в котором определена функция одной переменной (это `y(x)`). Она определяется через функцию Бесселя нулевого индекса `J0()`, аргументом которой указана умноженная на 2π разность аргумента функции `x` и переменной `FRAME`, умноженной на 0,1.

Диапазон изменения аргумента функции установлен от 0 до 10. Далее для этой функциональной зависимости строится график.

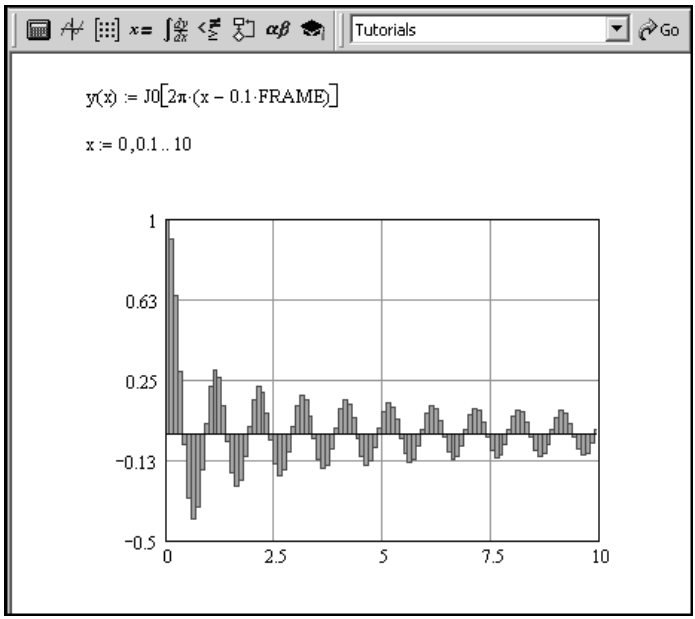


Рис. 3.96. Анимация создается на основе графика функции

По вертикали на графике указана область изменения от –0,5 до 1, отображается координатная сетка, а сама зависимость представлена в виде диаграммных столбиков (см. рис. 3.96). Стоит обратить внимание на то, что в описании функции была использована переменная `FRAME`, но как аргумент функции она не указывалась. На основе этого графика и с учетом определенной в начале документа функции и будет создаваться анимационный клип.

В общем случае для записи клипа выбирается команда **Tools | Animation | Record**, в результате чего открывается диалоговое окно **Record Animation**, которое представлено на рис. 3.97.

В диалоговом окне есть три поля: **From**, **To** и **At**. В полях **From** и **To** указываются начальное и конечное значения для переменной `FRAME`. По умолчанию переменная принимает значения в диапазоне от 0 до 9. В поле **At** вводится число отображаемых за секунду кадров. Для начала записи клипа необходимо, при открытом диалоговом окне **Record Animation** с правильно заполненными полями, выделить область в рабочем окне, для которой создается анимация, и щелкнуть на кнопке **Animate**. Процесс выделения области анимации в рабочем документе показан на рис. 3.98.

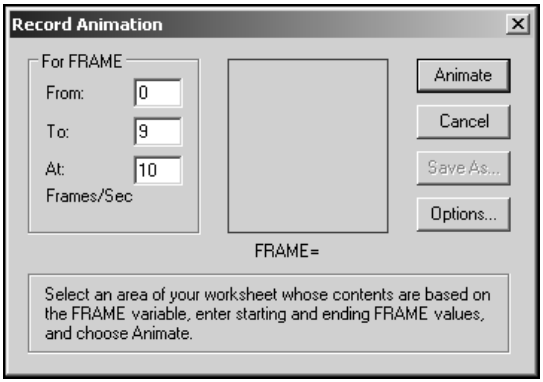


Рис. 3.97. Диалоговое окно записи анимации **Record Animation**

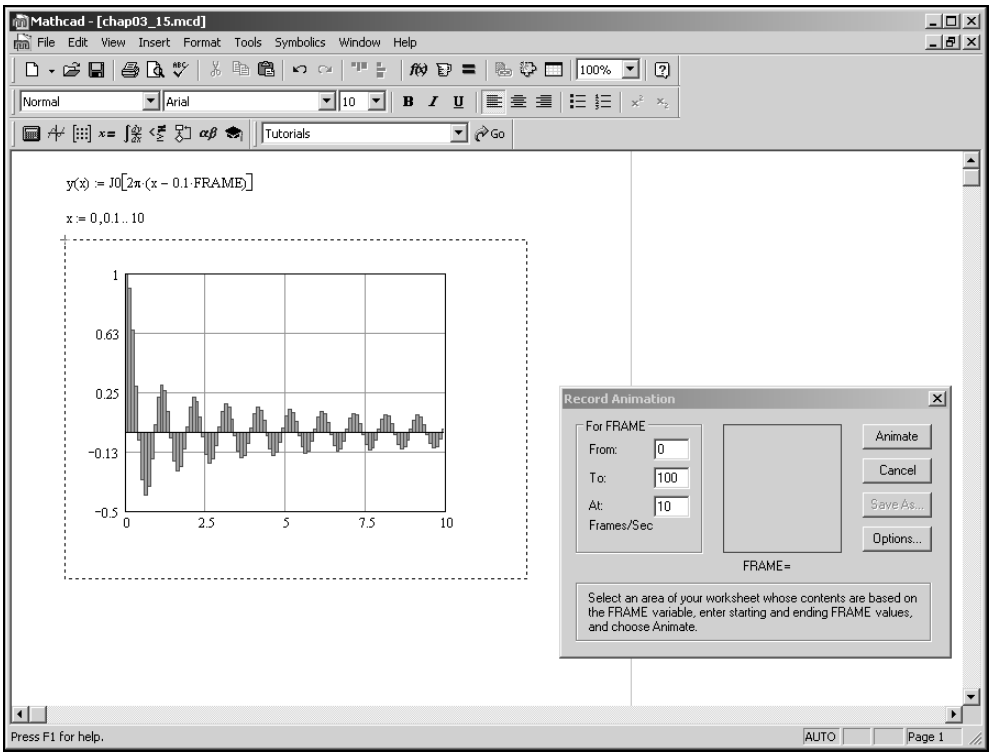


Рис. 3.98. Выбор области для создания анимации

После щелчка на кнопке **Animate**, если все было сделано правильно, начинается покадровая запись клипа. Записываемые клипы отображаются в диалоговом окне **Record Animation** в большом центральном поле. Для каждого

отображаемого кадра внизу указывается и соответствующее значение переменной `FRAME`. На рис. 3.99 показано, как может выглядеть диалоговое окно **Record Animation** в процессе записи клипа.

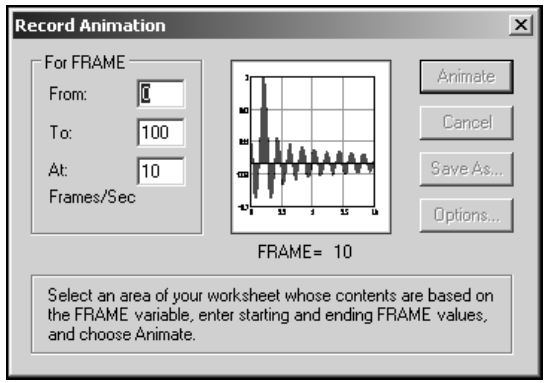


Рис. 3.99. Диалоговое окно создания анимации в процессе записи

После того как все кадры записаны, открывается окно просмотра анимации **Play Animation** (рис. 3.100).

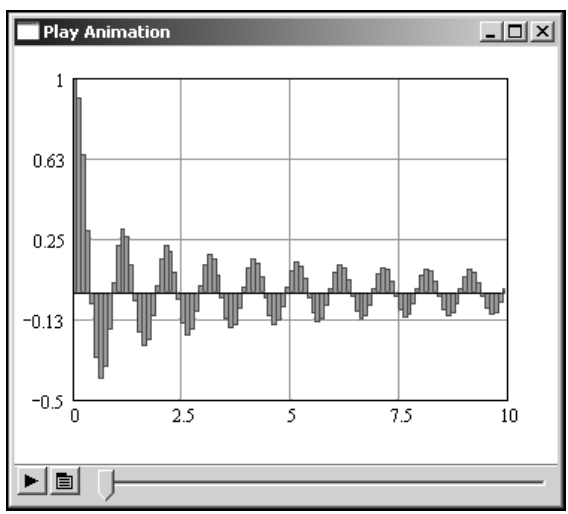


Рис. 3.100. Окно просмотра анимации

Окно, помимо области отображения анимации, содержит две кнопки: кнопку начала просмотра (кнопка со стрелкой) и кнопку для открытия файлов анимации

(кнопка с пиктограммой, напоминающей страницу блокнота). Кроме этого, в нижней части размещена полоса прокрутки кадров анимации. Для запуска анимации, таким образом, достаточно щелкнуть на кнопке со стрелкой. В табл. 3.6 представлены некоторые отдельные кадры, созданные в процессе записи клипа. По этим кадрам можно проследить процесс распространения условной волны.

Таблица 3.6. *Просмотр анимации*

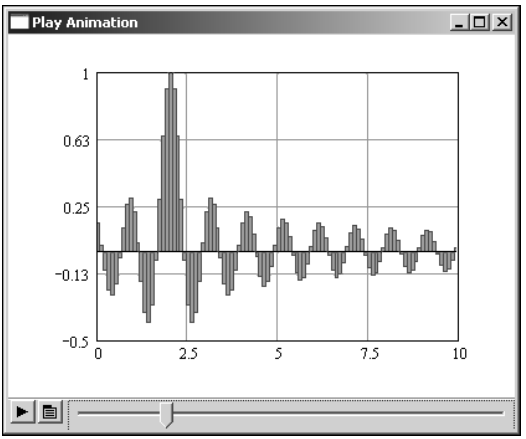
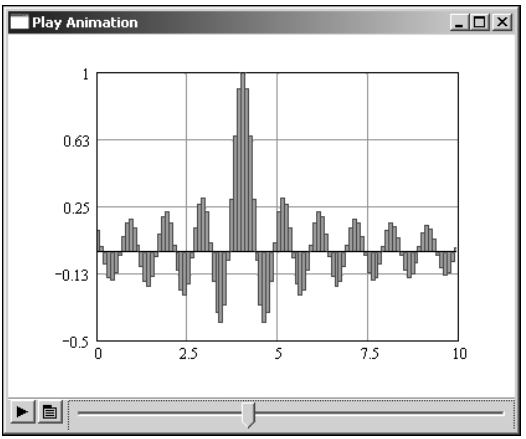
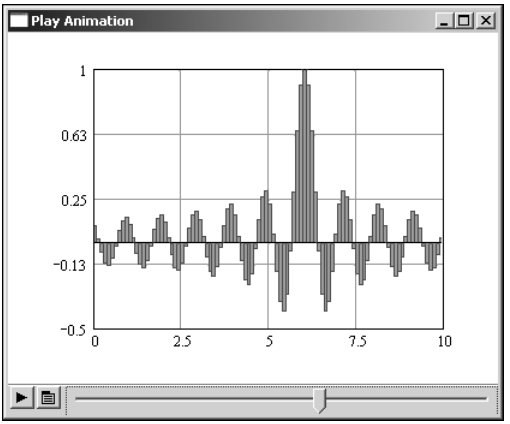
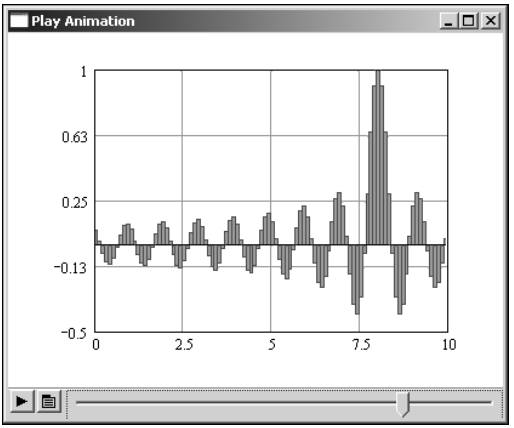
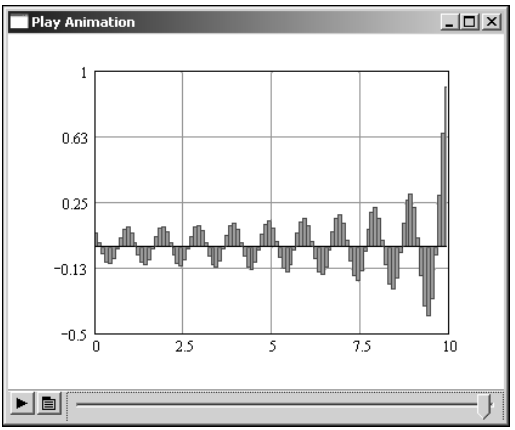
Кадр	Изображение
Двадцатый кадр	
Сороковой кадр	

Таблица 3.6 *(окончание)*

Кадр	**Изображение**
Шестидесятый кадр	
Восьмидесятый кадр	
Сотый кадр	

Чтобы сохранить созданный клип, следует щелкнуть на кнопке **Save As** в диалоговом окне **Record Animation**. Кнопка **Options** в этом диалоговом окне служит для настройки программы записи видео.

Рассмотренный пример достаточно прост. Однако такие несложные ситуации встречаются редко. Обычно анимация создается для пространственных изображений. В качестве иллюстрации на рис. 3.101 показана трехмерная поверхность. Соответствующая функциональная зависимость, как и ранее, задается функцией Бесселя, только аргумент у нее теперь другой. Общий периодический по аргументу множитель определяет временную зависимость функции. Сам процесс можно интерпретировать, с некоторой натяжкой, как колебания круглой мембраны.

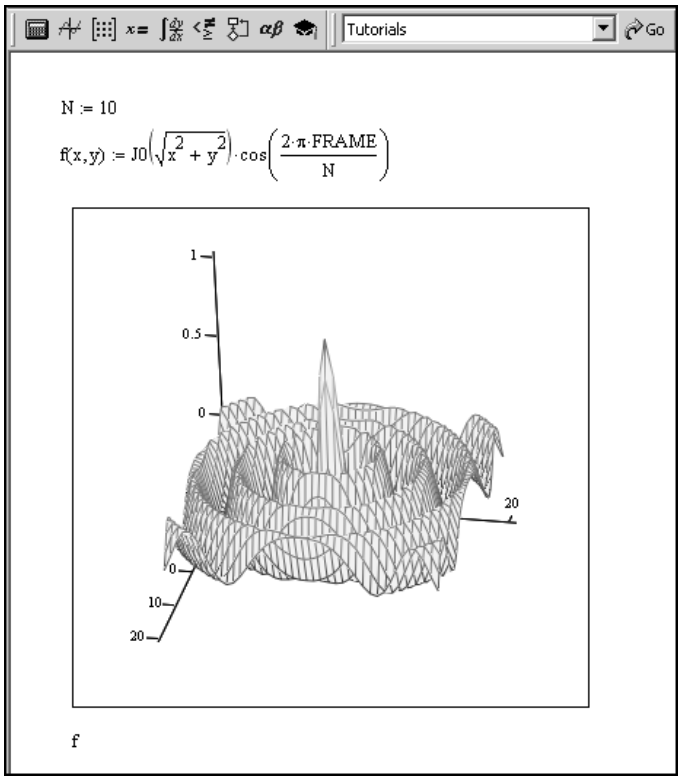


Рис. 3.101. Анимация создается на основе графика поверхности

Анимация создается на основе всего десяти кадров. Параметры базовой функциональной зависимости подобраны так, чтобы изображения на начальном и конечном кадрах совпадали, т. е. за рассматриваемый временной период система совершила полный цикл. Наиболее характерные кадры приведены в табл. 3.7.

Таблица 3.7. *Некоторые кадры анимации*

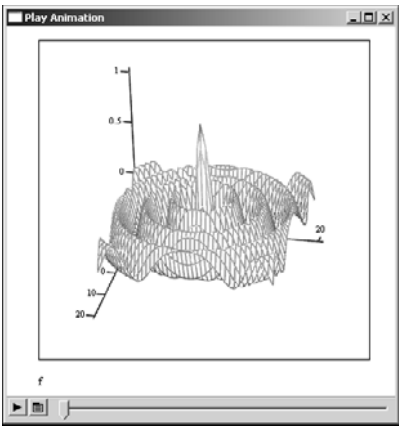
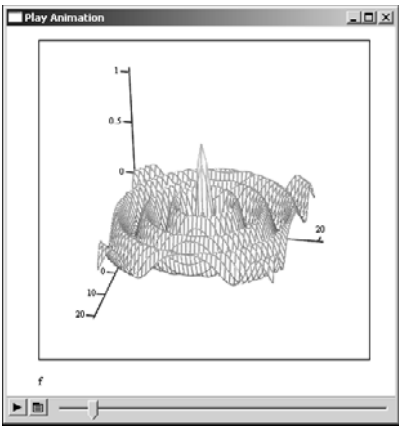
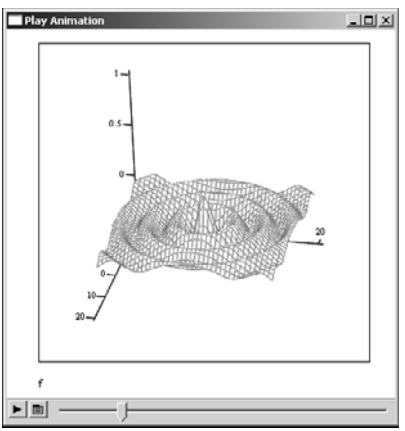
Кадр	**Изображение**
Начальный кадр	
Первый кадр	
Второй кадр	

Таблица 3.7 *(продолжение)*

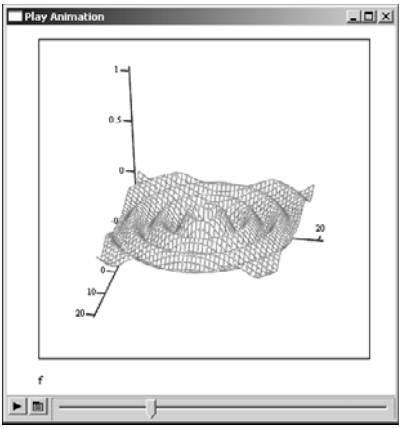
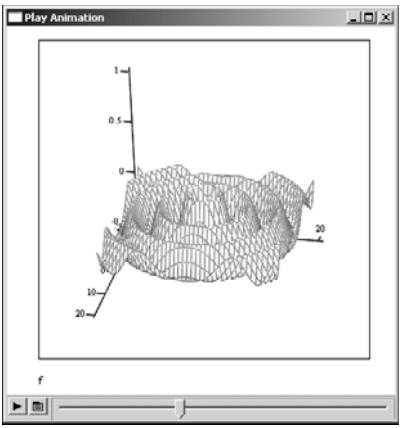
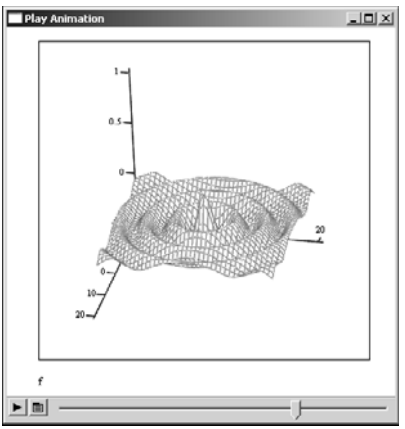
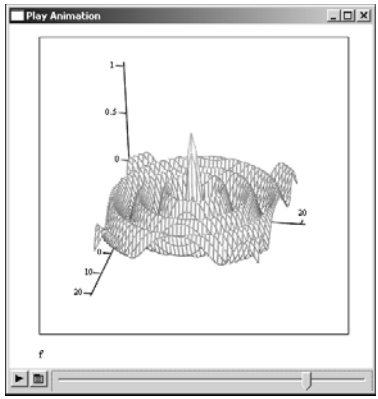
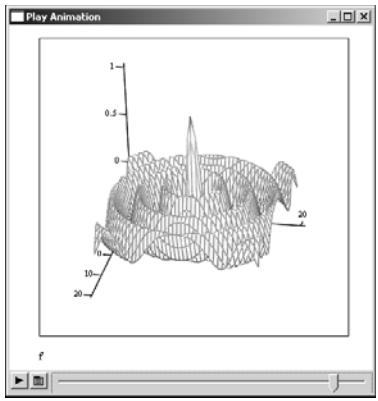
Кадр	Изображение
Третий кадр	
Четвертый кадр	
Восьмой кадр	

Таблица 3.7 *(окончание)*

Кадр	Изображение
Девятый кадр	
Десятый кадр	

Выбор числа кадров при создании анимации — достаточно важный момент, особенно если рассматриваемый процесс является периодическим или квазипериодическим. В последнем случае желательно подбирать диапазон записи анимации так, чтобы он совпадал с периодом системы. Кроме того, кадров должно быть достаточно для того, чтобы на них были зафиксированы основные состояния системы. Наиболее неприятна ситуация, когда временной интервал между записями кадров совпадает или близок к периоду системы: сколько бы кадров ни создавалось, динамику системы отследить при этом практически невозможно.

Глава 4

Программирование в Mathcad

Приложение Mathcad — не только достаточно мощный пакет для расчетов и символьных вычислений, но и удобная среда для составления программ. Причем специфика Mathcad такова, что в некоторых аспектах он превосходит другие популярные программные среды.

Основы программирования в Mathcad

Программой, как известно, называется упорядоченная последовательность инструкций, объединенных в единую логическую структуру. Программа обычно подразумевает более чем одну последовательность действий и этим отличается от формального набора команд. В каждой программной среде есть свои правила записи инструкций, которые образуют язык программирования. Для создания программ в Mathcad предусмотрены специальные утилиты, а для ввода в рабочем документе структур, посредством которых записываются программы, имеется специальная палитра (рис. 4.1).

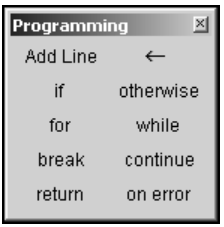


Рис. 4.1. Палитра инструкций программирования **Programming**

На палитре **Programming** представлено десять пиктограмм для ввода различных инструкций. Кратко каждая пиктограмма описана в табл. 4.1. Более подробно они обсуждаются в контексте рассматриваемых в этой главе вопросов.

Таблица 4.1. *Кнопки палитры* ***Programming***

Пиктограмма	Инструкция или оператор	Комбинация клавиш	Комментарий
Add Line	Оператор создания подпрограммы или добавления новой строки программы `Add Line`	<]>	Программа реализуется в специальном модуле, который технически выделяется утолщенной вертикальной линией. Пиктограмма служит для создания такой структуры (то есть линии с заполнителями) или добавления новых структурных заполнителей
if	Инструкция условного оператора `if`	<Shift>+<]>	Особенностью программы является наличие точек ветвления алгоритма, которые реализуются обычно с помощью условных операторов. Пиктограмму используют при создании таких операторов
for	Инструкция оператора цикла `for`	<Ctrl>+<Shift>+<'>	С помощью пиктограммы в рабочий документ добавляется структура оператора цикла с указанием диапазона изменения индексной переменой
break	Инструкция оператора прерывания `break`	<Ctrl>+<Shift>+<[>	Вставка оператора завершения программы. Оператор обычно применяется вместе с условными операторами и позволяет завершить выполнение программы
return	Инструкция оператора возвращения результата `return`	<Ctrl>+<Shift>+<\>	В качестве результата программой возвращается значение переменной, указанной последней командой в программном модуле. В некоторых случаях удобнее бывает оператор возвращения результата, который позволяет возвращать результат в процессе выполнения программы

Таблица 4.1 *(окончание)*

Пиктограмма	Инструкция или оператор	Комбинация клавиш	Комментарий
←	Оператор локального присваивания	<Shift>+<[>	Оператор присваивания в программном модуле
otherwise	Инструкция `otherwise`	<Ctrl>+<Shift>+<]>	Инструкция условного оператора, необходимая для определения последовательности действий (команд), выполняемых при невыполнении условия в условном операторе
while	Инструкция оператора цикла `while`	<Ctrl>+<]>	Вставка в рабочий документ структуры оператора цикла с инструкцией условия продолжения итерационного процесса
continue	Инструкция оператора перехода к следующей итерации `continue`	<Ctrl>+<[>	Оператор прерывания текущей итерации и перехода к следующей. Часто используется вместе с условным оператором и позволяет пропускать некоторые итерации в операторе цикла
on error	Инструкция оператора отслеживания ошибок `on error`	<Ctrl>+<'>	С помощью этого оператора можно отслеживать ошибки, возникающие в процессе расчета выражений и возвращать в этом случае, вместо ошибки, предопределенное значение

Каждая программа в рабочем документе Mathcad реализуется в виде отдельного блока или модуля. Структура этого блока вставляется в рабочий документ щелчком на пиктограмме **Add Line** на палитре **Programming** или нажатием клавиши <]>. Результат показан на рис. 4.2.

Вместо структурных заполнителей вводятся инструкции (команды) программы, которые должны быть выполнены. Для того чтобы добавить еще один структурный заполнитель (чтобы потом вместо него можно было ввести очередную команду), необходимо снова щелкнуть на пиктограмме **Add Line** (рис. 4.3).

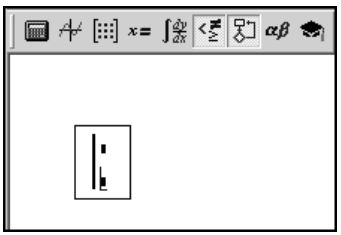


Рис. 4.2. Вставка программного блока в документ

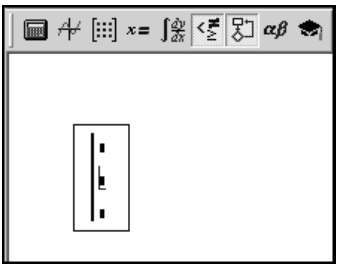


Рис. 4.3. Добавление в программный блок структурного заполнителя

Для удаления лишнего структурного заполнителя достаточно выделить его и нажать клавишу <Delete>. Обычно программой возвращается значение (хотя и не обязательно). Этот результат, к концу выполнения программы, как правило, записан в качестве значения некоторой переменной, которая указывается в последней команде программы. Следует также иметь в виду, что в теле программы используется оператор локального присваивания, который имеет вид левонаправленной стрелки и вводится с помощью панели инструментов **Programming** или нажатием комбинации клавиш <Shift>+<[>.

Пример 4.1. Дважды два

Исключительно простой пример программы, в которой присутствует оператор локального присваивания, приведен на рис. 4.4.

Программа состоит из трех команд-инструкций. Первой инструкцией локальной переменной `i` в качестве значения присваивается число 2. Второй командой эта переменная возводится в квадрат (после этого значение переменной `i` равно 4). Наконец, третья команда состоит только из названия локальной переменной, и это именно тот результат, который возвращается программой (то есть 4). Для того чтобы увидеть результат выполнения программы, программный блок следует выделить (линиями редактирования) и нажать клавишу <=>, т. е. ввести оператор равенства. Ситуация проиллюстрирована на рис. 4.5.

Разумеется, в реальных ситуациях необходимость в создании таких исключительно простых программ не возникает. Обычно программы более замысловаты. При этом помимо описывавшихся ранее функций, применяются еще и специальные операторы, о которых речь пойдет далее. Но прежде имеет смысл остановиться на логических операторах, которые часто входят в состав условных операторов. Условные же операторы исключительно важны, поскольку именно с их помощью в программах реализуется ветвление алгоритмов (преимущество, которое выгодно отличает программы от обычного набора команд).

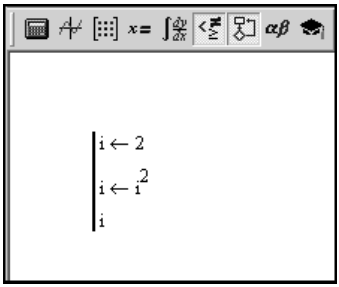


Рис. 4.4. Пример простой программы

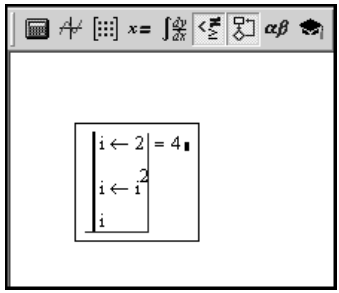


Рис. 4.5. Вычисление значения согласно созданной программе

Логические операторы

Логические операторы необходимы в основном для сравнения выражений. Их полный набор можно увидеть на палитре **Boolean**, которая изображена на рис. 4.6.

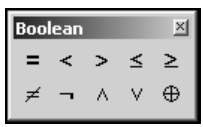


Рис. 4.6. Палитра **Boolean**

Назначение пиктограмм палитры **Boolean** приведено в табл. 4.2.

Таблица 4.2. *Пиктограммы панели инструментов* ***Boolean***

Пиктограмма	Оператор	Комбинация клавиш	Комментарий
=	Логическое равенство	<Ctrl>+<=>	Оператор имеет два операнда (бинарный оператор) и используется в операциях логического сравнения. Если равенство верно, в качестве значения возвращается 1, в противном случае — 0
<	Строгое логическое неравенство *меньше*	<Shift>+<,>	Бинарный оператор (имеет два операнда). Если левый операнд меньше правого, в качестве значения возвращается 1, в противном случае — 0

Таблица 4.2 *(окончание)*

Пиктограмма	Оператор	Комбинация клавиш	Комментарий
>	Строгое логическое неравенство *больше*	<Shift>+<.>	Оператор имеет два операнда. Если правый операнд меньше левого, в качестве значения возвращается 1, в противном случае — 0
≤	Нестрогое логическое неравенство *меньше либо равно*	<Ctrl>+<9>	Бинарный оператор. Если левый операнд не больше правого, в качестве значения возвращается 1, в противном случае — 0
≥	Нестрогое логическое неравенство *больше либо равно*	<Ctrl>+<0>	У оператора два операнда. Значение 1 возвращается, если правый операнд не больше левого операнда. Если это не так, возвращается 0
≠	Логическое неравенство *не равно*	<Ctrl>+<3>	Бинарный оператор. В качестве результата возвращается 1 для разных операндов и 0 для одинаковых
¬	Логическое отрицание *НЕ*	<Ctrl>+<Shift>+<1>	Унарный оператор (имеет один операнд). В качестве результата возвращается 1, если операнд равен 0. В противном случае возвращается 0
∧	Логическое *И*	<Ctrl>+<Shift>+<7>	Бинарный оператор. В качестве значения возвращается 0, если хотя бы один из двух операндов равен 0. Во всех остальных случаях возвращается 1
∨	Логическое *ИЛИ*	<Ctrl>+<Shift>+<6>	Бинарный оператор. Значением является 0, если оба операнда равны 0. Во всех остальных случаях возвращается 1
⊕	Логическое *исключающее ИЛИ*	<Ctrl>+<Shift>+<5>	Бинарный оператор. Единица возвращается в качестве результата в случае, если оба операнда различны и один из них при этом равен 0. Во всех прочих ситуациях возвращается 0

Пример 4.2. Логические операторы

Некоторые примеры вызова логических операторов (с проверкой результата) представлены в документе на рис. 4.7.

Рис. 4.7. Примеры вызова логических операторов

Для того чтобы проверить значение логического оператора, достаточно, как и в случае обычной команды, указать после него знак равенства (нажатием клавиши <=>).

Условный оператор

Главной отличительной чертой условного оператора является инструкция `if`, справа и слева от которой размещаются два структурных заполнителя. В правом заполнителе указывается проверяемое условие, а в левом — вычисляемое выражение. Обычно определяется также выражение (или оператор), вычисляемое (или выполняемый) при невыполнении условия в условном операторе. Альтернативное действие указывается вместе с инструкцией `otherwise`. Выполняемый оператор вводится вместо заполнителя слева от инструкции.

Пример 4.3. Условный оператор

На рис. 4.8 представлен пример программного блока с условным оператором и инструкцией `otherwise`.

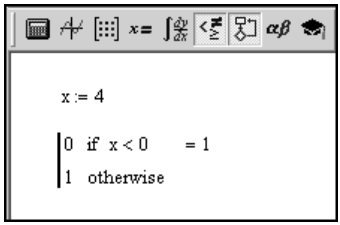


Рис. 4.8. Программный блок с условным оператором

В программном блоке проверяется условие, которое состоит в том, что переменная `x` меньше нуля. Если это так, то в качестве результата возвращается 0, в противном случае — 1 (значение, указанное слева от инструкции `otherwise`). Проверить результат выполнения программного блока можно, указав после него знак равенства. Предварительно, правда, необходимо определить значение переменной `x` (что и было сделано, как это видно из рис. 4.8).

Программные блоки с успехом используются при определении функциональных зависимостей, не сводящихся к суперпозиции элементарных математических функций. В этом случае после названия создаваемой функции (с указанием аргумента или аргументов) вводится оператор присваивания, после которого следует программный блок, где в соответствии со значением аргумента функции вычисляется ее значение.

Пример 4.4. Функция с условным оператором

На рис. 4.9 уже знакомый программный блок описывает функцию, которая возвращает два значения: 0 при отрицательном аргументе и 1 — в противном случае. Здесь же приведены примеры вызова созданной функции.

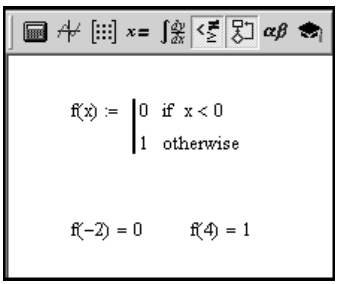


Рис. 4.9. Определение функции с использованием условного оператора

Часто в одном программном блоке реализуют сразу несколько условных операторов, которые могут последовательно размещаться один за другим или быть вложенными.

Пример 4.5. Несколько условных операторов

Пример последовательного размещения условных операторов приведен на рис. 4.10.

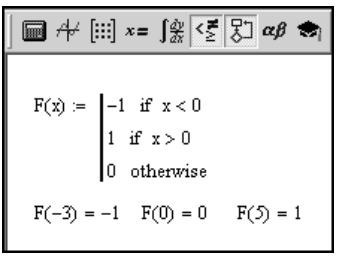


Рис. 4.10. Определение функции с использованием нескольких условных операторов

В соответствующем программном блоке определяется функция одного аргумента, которая может принимать три значения: –1 при отрицательном аргументе, 1 — при положительном и 0, если аргумент функции равен нулю. Программный блок реализуется с помощью двух условных операторов. Первым оператором проверяется, меньше ли нуля аргумент функции, и если это так, в качестве результата возвращается значение –1. Далее проверяется условие, положителен ли аргумент функции. В этом случае возвращается единица. Перед инструкцией `otherwise` указано нулевое значение. Поэтому если не выполняется ни одно из двух проверяемых условий (то есть аргумент функции не является ни положительным, ни отрицательным и, следовательно, равен нулю), в качестве значения возвращается нуль. Таким образом, если перед инструкцией `otherwise` следует несколько условных операторов, то оператор, соответствующий этой инструкции, выполняется только если не выполнено ни одно из условий в условных операторах.

Условный оператор допускается и без инструкции `otherwise`. Ведь ситуация, когда некоторое действие следует выполнить только при выполнении (или невыполнении) определенного условия, встречается на практике достаточно часто.

Пример 4.6. Использование условного оператора

Рассмотрим функцию, которая в качестве значения возвращает свой аргумент, если он неотрицателен, и число, на единицу большее, если аргумент функции отрицателен. Программный блок с описанием этой функции показан на рис. 4.11.

Первая команда соответствующего программного блока — инициализация локальной переменной `i`, которой в качестве значения присваивается аргумент `x`, передаваемый функции `y()`. Далее проверяется, меньше нуля

значение этой переменной или нет. Если переменная отрицательна, ее значение увеличивается на единицу. Последним оператором в программном блоке указана переменная `i`, которая и возвращается в качестве значения функции. Строго говоря, такую функцию можно было описать без ввода специальной локальной переменной. Фрагмент документа с альтернативным способом определения функции показан на рис. 4.12.

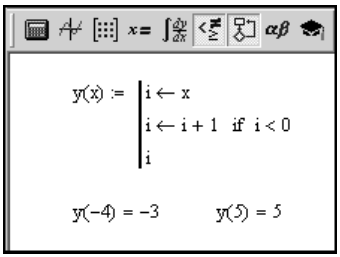


Рис. 4.11. Определение функции с использованием условного оператора и локальной переменной

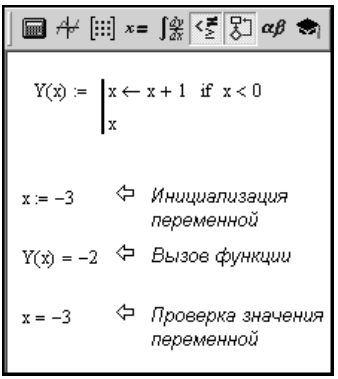


Рис. 4.12. Альтернативный способ определения функции

В отличие от предыдущего варианта, в данном случае программный блок содержит всего два оператора. Первым оператором аргумент функции увеличивается на единицу при условии, что этот аргумент меньше нуля. Аргумент возвращается в качестве значения функции. Ниже приведен пример вызова такой функции. Сначала переменной `x` присвоено отрицательное значение –3. Далее переменная указывается аргументом функции, после чего проверяется значение переменной-аргумента. Несложно заметить, что значение этой переменной не меняется. Дело все в том, что в теле программы использовался оператор локального присваивания. Поэтому все операции по изменению значений переменных, в том числе и глобальных (то есть таких, что инициализированы вне программного блока), имеют силу только в пределах данной программы.

Несколько иначе реализуется программа с вложенными условными операторами. Речь идет о том, что одна или более команд, выполняемых в рамках условного оператора, реализована с помощью условного оператора. Обычно к такой схеме построения программы прибегают в случаях, когда проверяется большое количество взаимосвязанных условий. Классический пример — определение кусочно-непрерывной функции.

Пример 4.7. Кусочно-непрерывная функция

В этом случае искомая функция может записываться через достаточно простые элементарные функции, однако, в зависимости от диапазона, в который попадает аргумент, формулы, определяющие ее значение, разные. Такая ситуация показана на рис. 4.13. Сначала там определена кусочно-непрерывная функция, а далее, чтобы можно было представить характер функциональной зависимости, построен график.

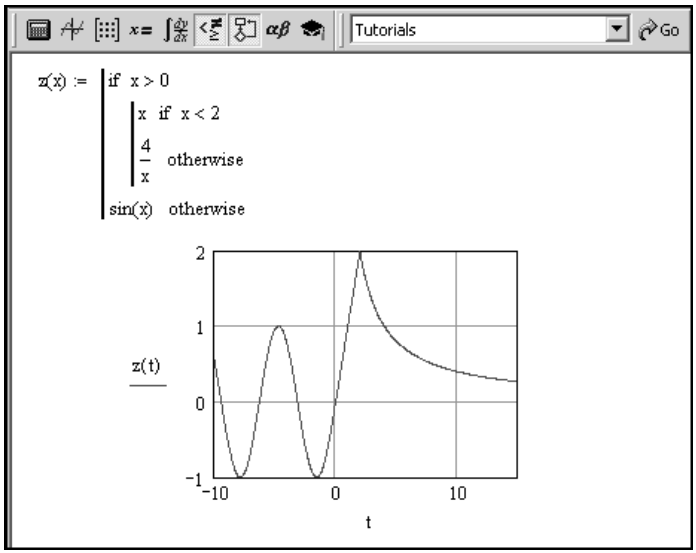


Рис. 4.13. Вложенные условные операторы использованы при определении кусочно-непрерывной функции

Программа реализуется по следующему алгоритму: сначала проверяется условие `x > 0`, т. е. больше ли нуля аргумент функции. Если это так, то далее проверяется условие `x < 2`. При верном втором условии значение функции равно ее аргументу, т. е. `x`. Таким образом, если аргумент функции попадает в интервал от 0 до 2, то функция задается линейной зависимостью. В ситуации, когда второе условие не выполняется (но выполняется первое), функция определяется зависимостью `4/x`. Именно такое выражение указано вместе с инструкцией `otherwise` в блоке второго, внутреннего условного оператора. Эта формула, следовательно, соответствует значениям аргумента, большим либо равным двум. Наконец, если не выполнено условие первого, внешнего условного оператора, значение функции вычисляется по формуле `sin(x)`. Разумеется, существует много способов проверки нескольких условий, в том числе с помощью вложенных условных операторов. На рис. 4.14 показан

фрагмент документа с описанием кусочно-непрерывной функции, полностью аналогичной той, что была представлена ранее. Однако в данном случае несколько иначе организован программный блок, в котором непосредственно задается функция. В частности, в первую очередь проверяется, попадает ли аргумент функции в интервал значений от 0 до 2. При этом использован логический оператор И (должны одновременно выполняться два неравенства). Второй, вложенный условный оператор указан в качестве команды для инструкции `otherwise`.

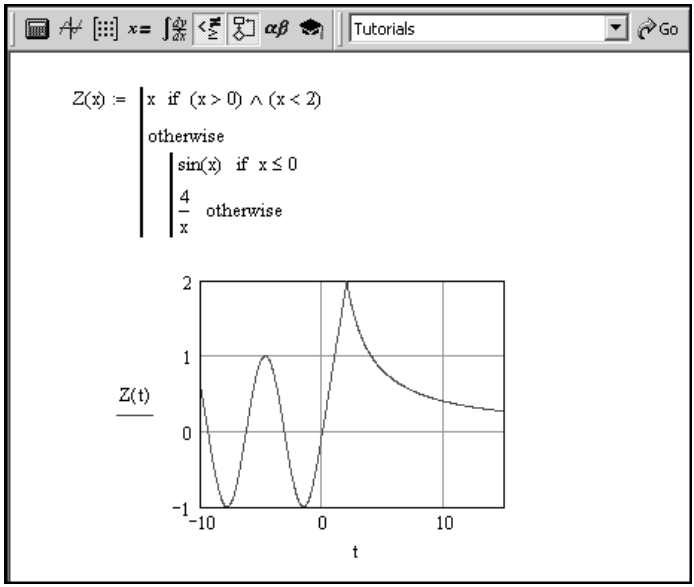


Рис. 4.14. Альтернативный способ определения кусочно-непрерывной функции

Убедиться в том, что в программном блоке описана та же функция, что и созданная ранее, можно по графику, приведенному в нижней части рис. 4.14. В обоих случаях вложенный условный оператор вводился в отдельном блоке, отделенном вертикальной линией слева. Чтобы вставить в тело программы такой блок, нужно выделить структурный заполнитель (слева от инструкции `if` условного оператора или инструкции `otherwise` — в соответствии с тем, где блок должен быть добавлен), после чего выполняется щелчок на пиктограмме **Add Line** палитры **Programming**.

Пример 4.8. Вложенные операторы

Разницу между последовательным размещением условных операторов и ситуацией, когда один из них вызывается внутри другого, иллюстрирует рис. 4.15.

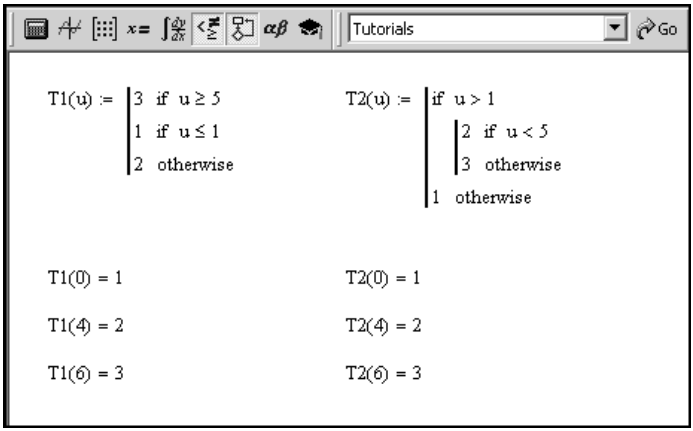


Рис. 4.15. Различные способы размещения условных операторов в теле программы

При определении первой функции `T1()` использовано два последовательных условных оператора. В первом аргумент функции сравнивается с числом 5. Если аргумент не меньше этого числа, в качестве значения функции возвращается 3. Вторым оператором проверяется условие, состоящее в том, что аргумент функции меньше либо равен 1. В этом случае возвращается значение 1. При невыполнении ни одного из условий в условных операторах значение функции равно 2. Ниже под описанием функции приведены примеры ее вызова. В правой части документа описана функция `T2()`. В теле блока описания этой функции в первом операторе аргумент сравнивается с единицей. Если он строго больше единицы, выполняется другой условный оператор. В соответствии с этим вложенным оператором для аргументов функции, меньших 5, в качестве значения возвращается число 2, если это не так, то число 3. Наконец, для значений аргумента функции, не превышающего 1, значение функции равно 1 (команда у инструкции `otherwise` для внешнего оператора). Сразу под блоком программы приведены примеры вызова функции `T2()`. Несложно заметить, что оба программных блока фактически описывают одну и ту же зависимость: для значений аргумента, не превышающих 1, значение функции равно 1; если аргумент функции лежит в интервале от 1 до 5, функция принимает значение 2; при аргументе, принимающем значение от 5 и более, в качестве результата возвращается число 3. Разница между функциями `T1()` и `T2()` состоит лишь в способе реализации такой зависимости. В частности, условие второго условного оператора в функции `T1()` проверяется всегда, вне зависимости от того, выполнено ли условие первого условного оператора. Напротив, в функции `T2()` условие второго вложенного условного оператора проверяется только в том случае, если выполнено условие первого оператора. При написании небольших программных блоков такая

разница обычно мало сказывается на конечном результате. Однако когда программа достаточно большая или в процессе ее реализации предполагается выполнить существенное число итераций, такие нюансы следует принимать во внимание, поскольку при этом важно не только корректно составить алгоритм (последовательность команд), но и реализовать наиболее оптимальные схемы.

Еще одна особенность условных операторов, на которую хочется обратить внимание читателя, связана с интерпретацией численных значений при работе с логическими операторами. Любое численное значение, отличное от нуля, интерпретируется, если оно указано в логическом операторе, как `TRUE`. Нулевое значение, соответственно, интерпретируется как `FALSE`. Поэтому в условных операторах в качестве условия допустимы не только логические операторы, но и обычные арифметические выражения.

Пример 4.9. Арифметическое выражение в условии

Пример арифметического выражения в условном операторе представлен на рис. 4.16.

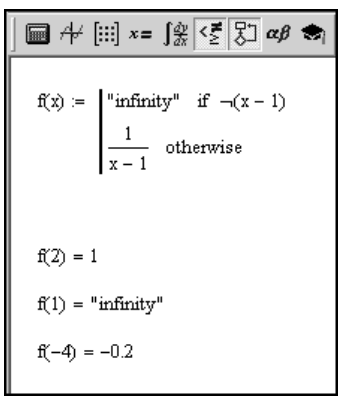


Рис. 4.16. Пример использования в условном операторе арифметического выражения

В приведенном документе описывается функция, которая возвращает результат, равный единице, деленной на разность аргумента функции и единицы. Эта функция, очевидно, определена для любого аргумента, за исключением единицы. При таком аргументе функцией в качестве результата возвращается текстовая строка `"infinity"`. Соответствующий программный код реализуется с помощью всего одного условного оператора. В качестве проверяемого условия, при выполнении которого значением является упомянутая текстовая строка, указано арифметическое выражение (разность аргумента функции

и единицы, т. е. выражение в знаменателе при вычислении значения функции). Точнее, это выражение является аргументом логического оператора отрицания. Если это выражение равно нулю, результатом вызова функции должна быть текстовая строка. Нулевое значение, как отмечалось ранее, интерпретируется как логическое значение `FALSE`, поэтому возникает ситуация, будто условие в условном операторе не выполнялось. Поскольку выражение является аргументом логического оператора отрицания, то при нулевом выражении результат всей логической операции равен `TRUE` (в противном случае — `FALSE`). Поэтому слева от инструкции `if` условного оператора указана текстовая строка, возвращаемая функцией в качестве результата в точке разрыва. Для всех прочих значений аргумента результат вызова функции вычисляется по формуле, указанной слева от инструкции `otherwise`. Разумеется, можно было, например, просто записать в условном операторе в качестве условия упомянутое выражение (без оператора отрицания). Тогда следует поменять местами текстовую строку и формулу для вычисления значения функции.

Операторы циклов

Наряду с условными операторами, особое место занимают операторы циклов. Их назначение состоит в основном в том, чтобы в краткой форме кодировать серии большого числа однотипных операций. В Mathcad существует несколько способов организации циклов. Рассмотрим их.

Часто при выполнении оператора цикла каждый итерационный шаг можно отождествить с некоторой переменной, которую далее будем называть индексной. В этом случае процесс реализации оператора цикла состоит в том, что индексная переменная пробегает значения из определенного множества, а выполняемая последовательность действий при данном фиксированном значении индексной переменной обычно зависит от индексной переменной (но это не обязательно). Такого рода циклы в Mathcad создаются с помощью конструкции `For`, при вставке которой (с помощью пиктограммы на панели программирования) автоматически добавляется блок с двумя структурными заполнителями и символом принадлежности множеству. В левом структурном заполнителе указывается индексная переменная, а вместо правого структурного заполнителя вводится интервал или множество значений, которые она пробегает.

Пример 4.10. Сумма квадратов

На рис. 4.17 приведен фрагмент документа с кодом процедуры для вычисления суммы квадратов натуральных чисел. Верхняя граница суммы указывается аргументом процедуры.

```
MySum(N) := | s ← 0
            | for i ∈ 1..N
            |   s ← s + i^2
            | s

MySum(10) = 385
```

Рис. 4.17. Процедура вычисления суммы квадратов натуральных чисел

В начале процедуры с нулевым значением инициализируется локальная переменная `s`. Далее индексная переменная пробегает значения от `1` до `N` (аргумент функции, определяющий верхнюю границу суммы) и квадрат значения индексной переменной прибавляется к значению локальной переменной `s`. После выполнения всех итераций значение локальной переменной возвращается как результат выполнения процедуры (которую, кстати, можно рассматривать и как функцию).

Нередко встречается инструкция `continue`, которая служит указанием для выхода из цикла. Точнее, после этой инструкции с помощью ключевого слова `if` указывают условие, при котором выполнение цикла прекращается.

Пример 4.11. Сумма квадратов с ограничением

В качестве примера разместим в рассматривавшемся ранее операторе цикла команду `continue if i > 10` (рис. 4.18).

```
MySum2(N) := | s ← 0
             | for i ∈ 1..N
             |   | continue  if i > 10      ⇦ Измененный код процедуры
             |   | s ← s + i^2
             | s

MySum2(5) = 55
                                            ⇦ Примеры вызова
MySum2(10) = 385

MySum2(15) = 385
```

Рис. 4.18. Измененная процедура с примерами вызова

В этом случае сумма будет увеличиваться до тех пор, пока индексная переменная не превысит значение 10. Другими словами, процедурой будет вычисляться сумма квадратов натуральных чисел с верхней границей, равной минимальному из чисел 10 и аргумента функции. В последнем несложно убедиться по результатам вычисления суммы для разных передаваемых процедуре аргументов. Если аргумент превышает 10, результат не меняется.

Операторы цикла в Mathcad организуются также с помощью инструкции `while`. Цикл продолжается до тех пор, пока выполняется условие, указанное после этого ключевого слова. Главная особенность состоит в том, что теперь пользователю самостоятельно приходится определять процедуру изменения индексной переменной или задавать иной способ взаимосвязи между проверяемым условием и процессом выполнения итераций.

Пример 4.12. Еще один оператор цикла

Процедуру вычисления суммы квадратов натуральных чисел на основе инструкции `while` можно было бы определить так, как показано во фрагменте программного кода на рис. 4.19.

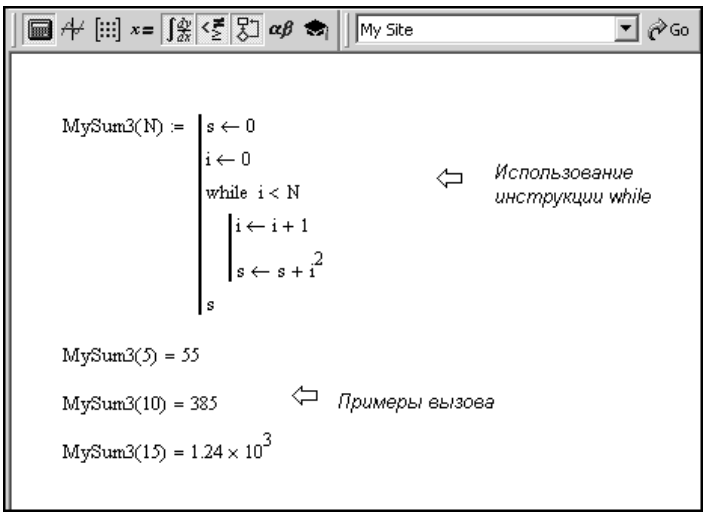


Рис. 4.19. Использование в теле процедуры инструкции `while`

Теперь приходится в явном виде инициализировать не только переменную для записи значения суммы, но и индексную переменную. Проверяемым условием является неравенство: индексная переменная не должна превышать верхнюю границу, т. е. аргумент процедуры. Выполняемых операций в пре-

делах каждой отдельной итерации также две: сначала на единицу изменяется значение индексной переменной, а затем квадрат этого значения прибавляется к переменной, определяющей сумму квадратов натуральных чисел. Хотя такой код, по сравнению с первоначальным, выглядит более громоздким, реализация оператора цикла посредством инструкции `while` часто бывает предпочтительней, поскольку позволяет создавать более гибкий алгоритм.

Пример 4.13. Использование инструкции *break*

Как и ранее, при работе с инструкцией `while` можно пропускать некоторые итерации или вообще преждевременно завершать выполнение программного кода. Пример приведен в документе на рис. 4.20.

```
MySum4(N) := | s ← 0
             | i ← 0
             | while i < N
             |   | i ← i + 1
             |   | break if i > 10
             |   | s ← s + i^2
             | s

MySum4(5) = 55
MySum4(10) = 385
MySum4(15) = 385
```

⇦ *Использование инструкции break*

⇦ *Примеры вызова*

Рис. 4.20. Использование инструкции `break`

После команды увеличения индексной переменной добавлена инструкция `break` и условный оператор. В условном операторе проверяется, превышает ли индексная переменная значение 10. Если это так, то выполнение процедуры прекращается. Здесь легко прослеживается главное отличие в работе инструкций `continue` и `break` — если в первом случае пропускается соответствующая итерация, то во втором — завершается работа.

Рекурсия

Под рекурсией в общем случае подразумевают ситуацию, когда в теле описания процедуры (или функции) вызывается эта же процедура (или функция), но уже с другим аргументом. Иными словами, если функция или процедура ссылается на саму себя, имеет место рекурсия. Часто рекурсия бывает следствием ошибки, допущенной пользователем. Но если рекурсию реализовать с умом, то можно создавать эффективные, компактные и достаточно элегантные программные коды.

Пример 4.14. Двойной факториал

Как иллюстрацию к использованию рекурсии рассмотрим процедуру вычисления двойного факториала — произведения чисел "через один", т. е. для четной границы это произведение всех четных чисел, начиная с 2, а для нечетной границы — нечетных, начиная с 1. Фрагмент программного кода такой процедуры приведен на рис. 4.21.

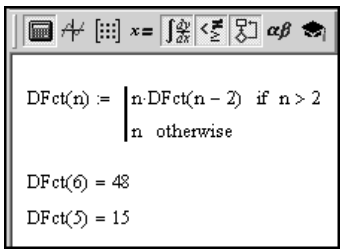


Рис. 4.21. Рекурсия при вычислении двойного факториала

Как видим, код состоит всего из одного условного оператора. Проверяемым является условие, что аргумент больше 2. Если это условие выполнено, вычисляется произведение аргумента на значение, возвращаемое процедурой для аргумента, уменьшенного на 2. При невыполнении условия в качестве значения возвращается аргумент. Таким образом, процедура будет по рекурсии вызывать (с уменьшением аргумента с шагом 2) сама себя до тех пор, пока аргументом не станет число, не большее 2 — т. е. 2 или 1. Наличие при каждом рекурсивном вызове процедуры множителя обеспечивает корректное вычисление двойного факториала. Подтверждением тому служат примеры, приведенные внизу под блоком описания процедуры.

Рекурсия позволяет существенно сократить программный код. При этом вопрос о том, упрощается ли алгоритм выполнения программы, остается открытым. Чего однозначно нельзя отрицать, — так это простоты решения целого ряда задач с помощью рекурсивного описания процедур.

Пример 4.15. Вычисление суммы через рекурсию

В качестве примера можно привести задачи о вычислении сумм и произведений. На рис. 4.22 представлен пример документа с кодом процедуры для вычисления суммы $\sum_{n=0}^{N} \frac{1}{2^n}$. Аргумент процедуры — верхняя граница индексной переменной.

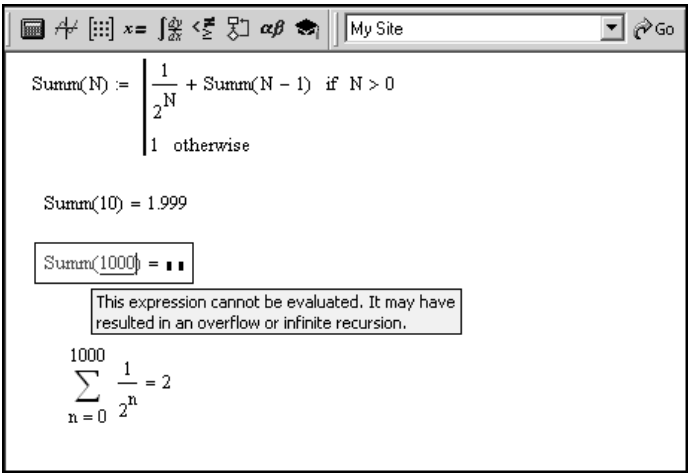


Рис. 4.22. Вычисление суммы с помощью рекурсии

Как и при вычислении двойного факториала, в данном случае программа состоит всего из одного условного оператора. При условии отличного от нуля аргумента вычисляется сумма значения, возвращаемого процедурой для уменьшенного на единицу аргумента, и слагаемого с показателем степени, совпадающем с аргументом процедуры. При нулевом значении аргумента в качестве результата возвращается единица. Код, таким образом, достаточно простой и элегантный, однако увлекаться рекурсией без крайней необходимости все же не стоит. Пояснение к этому можно найти ниже, под описанием процедуры, где приведены примеры ее вызова. Дело в том, что она работает только с относительно небольшими аргументами. Если же аргумент велик, вместо ожидаемого результата (близкого к значению 2) появляется сообщение об ошибке. Причем стандартная встроенная процедура для вычисления суммы дает вполне корректное значение (см. рис. 4.22). Причина в том, что рекурсия подразумевает существенную загрузку системных ресурсов. Именно это имелось в виду, когда утверждалось, что простота кода не всегда означает эффективность алгоритма.

Элементы управления

В рабочих документах Mathcad допускается вставка так называемых элементов управления — специальных кнопок, переключателей, полей и т. п. Их не очень много, но они достаточно функциональны. Можно создавать специальные сценарии, с помощью которых определяется взаимодействие объектов рабочего документа Mathcad и элементов управления. Вначале кратко опишем основные элементы управления и методы работы с ними. Далее будут изложены основные принципы составления кодов сценариев, после чего приводятся примеры применения элементов управления в рабочих документах Mathcad.

Вставка элементов управления в рабочий документ может осуществляться с помощью панели инструментов **Controls** либо путем выбора команды **Insert | Control**, с последующим выбором нужного элемента. Читатель мог обратить внимание при описании соответствующей панели, что предлагаемых элементов управления совсем немного — всего шесть. Среди них три кнопки (обычная, кнопка-опция и кнопка-переключатель), текстовое поле, список и полоса прокрутки. На рис. 4.23 проиллюстрирован процесс вставки кнопки в рабочий документ.

Рис. 4.23. Вставка кнопки в рабочий документ

Для вставки кнопки необходимо щелкнуть на соответствующей пиктограмме панели **Controls**. В результате вставляется не только элемент управления, но и структурный заполнитель, вместо которого вводится название переменной (или переменных), определяемых в процессе взаимодействия с элементом управления. В частности, после вставки кнопки рабочий документ будет иметь такой вид, как показано на рис. 4.24.

Здесь кратко отметим ряд настроек, которые могут применяться к элементам управления. Разумеется, в зависимости от того, какой элемент вставлен в рабочий документ, различны и настройки. Однако принципы их выполнения одинаковы для всех элементов. Сразу отметим два важных направления, по которым осуществляется настройка элементов управления в рабочем документе. Во-первых, это внешний вид элемента управления и формальные атрибуты: количество так называемых входных и выходных элементов.

Входные элементы — это те, что передаются как аргументы для выполнения процедур, связанных с функциональностью элемента управления. Выходными являются те переменные, значения которых определяются непосредственно в ходе реализации вышеозначенных процедур. Во-вторых, обычно приходится редактировать (или создавать) код для процедур-обработчиков событий, т. е. фактически тех процедур, с помощью которых изменяются или определяются значения выходных переменных.

Рис. 4.24. Результат вставки кнопки в рабочий документ

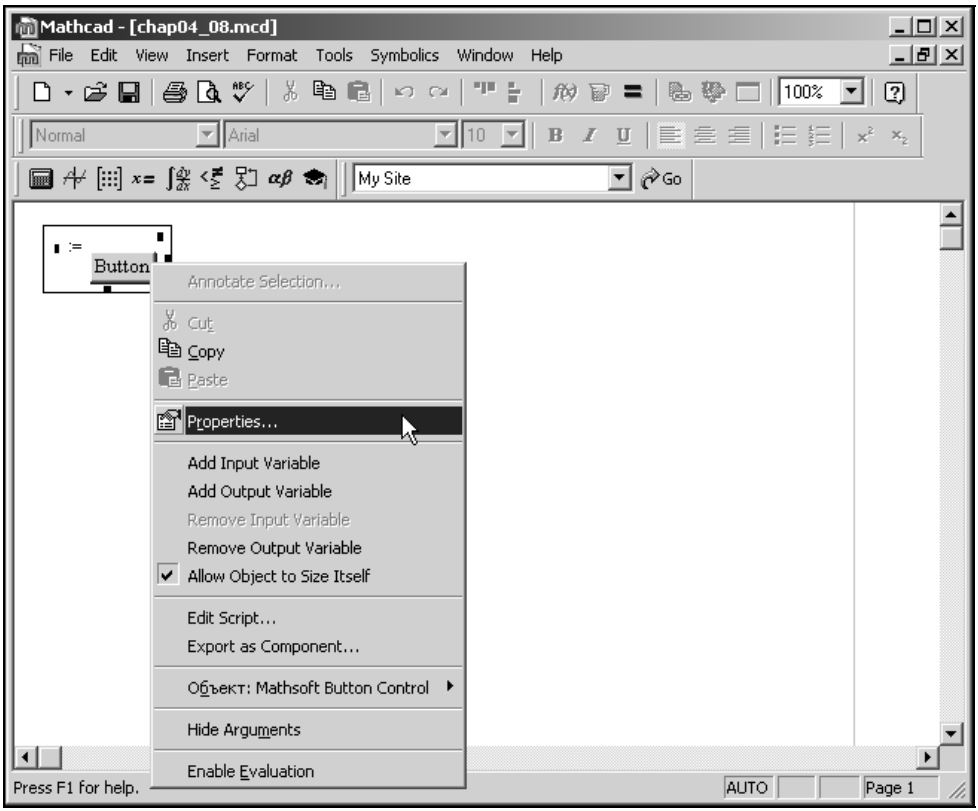


Рис. 4.25. Команды настройки элемента управления из раскрывающегося списка

Большинство необходимых действий по настройке элемента управления начинаются со щелчка правой кнопки мыши. В раскрывающемся списке много команд, среди которых есть достаточно интересные (рис. 4.25). Но сначала необходимо выделить именно элемент управления, а не структурный заполнитель. Списки в обоих случаях будут практически одинаковыми, а результат — не очень.

Если в раскрывшемся списке выбрать команду **Properties**, откроется диалоговое окно **Component Properties**, представленное на рис. 4.26.

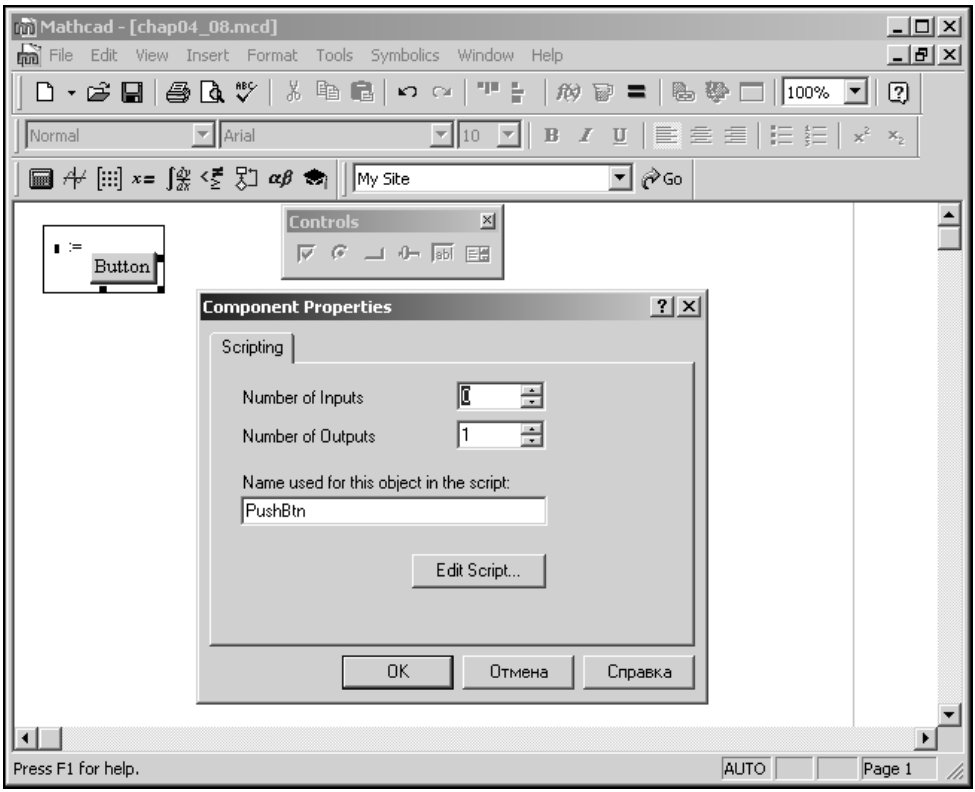


Рис. 4.26. Диалоговое окно **Component Properties**

Окно содержит всего одну вкладку **Scripting**, на ней три поля и кнопка. В поле **Number of Inputs** указывают число входных переменных (т. е. аргументов), значение по умолчанию — 0. Поле **Number of Outputs** предназначено для указания числа выходных переменных (по умолчанию это значение равно 1). В поле **Name used for this object in the script** вводится название для объекта (элемента управления), по которому на этот объект будут выпол-

няться ссылки в коде сценария. Без крайней необходимости название, предлагаемое по умолчанию, менять не следует. Наконец, кнопка **Edit Script** предназначена для перехода к редактору сценария. На рис. 4.27 показано окно **Script Editor**, которое открывается после щелчка на этой кнопке.

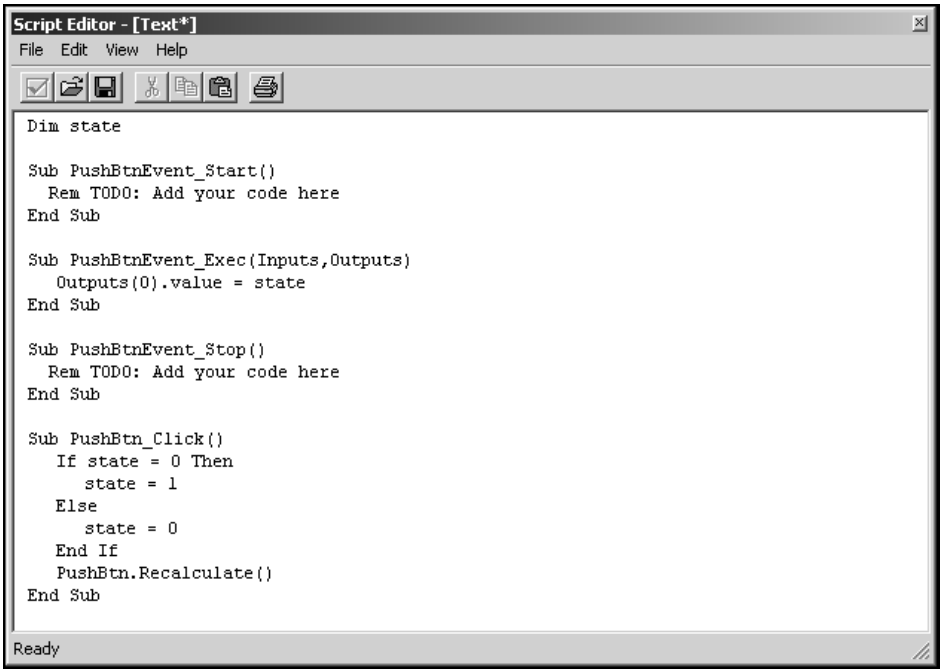


Рис. 4.27. Окно редактора сценариев **Script Editor**

Это же окно может быть открыто, если в раскрывающемся списке выбрать команду **Edit Script**. Как работать с кодами сценариев, рассказывается несколько позже. Здесь же остановимся на том, как быстро и легко выполнять самые простые, но необходимые настройки внешнего вида элемента управления.

Внешние параметры элемента управления

Процедуру настройки внешних параметров элементов управления рассмотрим на примере такого объекта, как кнопка. В частности, если в раскрывающемся списке меню выбрать подменю **Объект: Mathsoft Button Control** (см. рис. 4.25) и затем команду **Properties**, откроется окно, которое показано на рис. 4.28.

Окно **Свойства** содержит три вкладки: **General**, **Шрифты** и **Рисунки**. На вкладке **General** размещены опции, группа переключателей, несколько полей и списков. Раздел **Type** содержит группу переключателей на три положения.

В зависимости от положения переключателя, кнопка может быть обычной (положение **Push Button**), кнопкой-опцией (положение **Check Box**) или кнопкой-переключателем (положение **Radio Button**). Таким образом, даже после вставки элемента управления — обычной кнопки — далее можно будет поменять ее тип. Поля **Group ID** и **Button ID** в разделе **Radio Group** доступны только при работе с кнопками-переключателями, и в этих полях указываются идентификаторы группы кнопок и непосредственно кнопок. В списках раздела **Alignment** выбирается способ выравнивания текста с названием кнопок по горизонтали (список **Horizontal**) и по вертикали (список **Vertical**). Раздел **Style** содержит семь опций, назначение которых таково:

- **Auto Size** — масштабирование размеров кнопки в соответствии с названием;
- **Left text** — отображение названия кнопки слева от значка опции или переключателя;
- **Multiline** — отображение названия в несколько строк;
- **Push-like** — отображение кнопок-опций и кнопок-переключателей с использованием стиля обычных кнопок;
- **Bitmap** — загрузка изображения для отображения его на кнопке;
- **Icon** — загрузка пиктограммы для отображения на кнопке;
- **Flat** — отображение кнопки без объемных эффектов, т. е. в этом случае кнопка является невыпуклой на фоне рабочего листа.

Пожалуй, наиболее важный элемент на вкладке **General** — поле **Caption**, куда вводится имя элемента управления, непосредственно отображаемое на нем. Оно может отличаться от названия элемента, которое упоминалось ранее и через которое в сценарии выполняется ссылка на элемент управления.

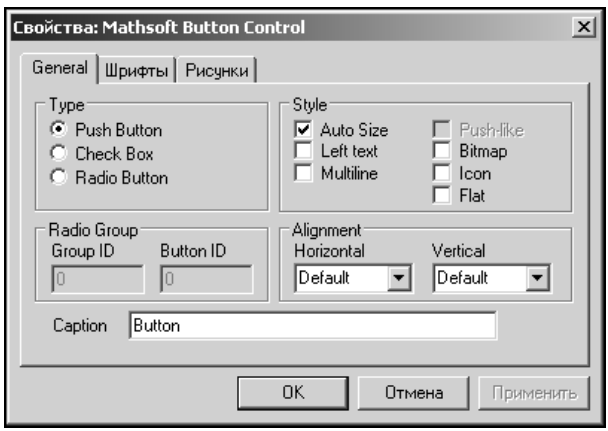


Рис. 4.28. Диалоговое окно **Свойства** открыто на вкладке **General**

На вкладке **Шрифты** выполняются настройки, связанные с типом, размером и прочими атрибутами шрифтов, заданных при отображении элемента управления. Диалоговое окно **Свойства**, открытое на вкладке **Шрифты**, показано на рис. 4.29.

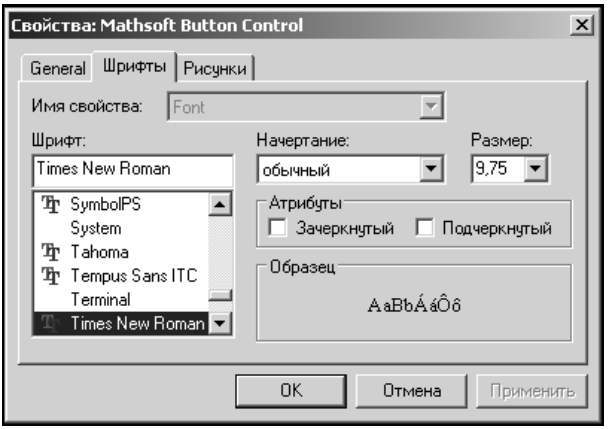


Рис. 4.29. Диалоговое окно **Свойства** открыто на вкладке **Шрифты**

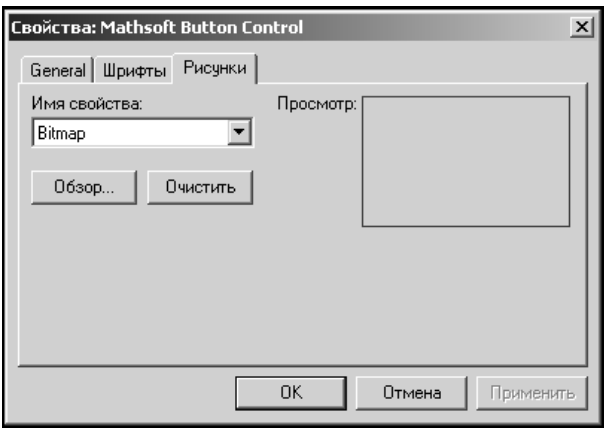


Рис. 4.30. Диалоговое окно **Свойства** открыто на вкладке **Рисунки**

Элементы на этой вкладке стандартные для всех окон подобного типа, поэтому думается, особой необходимости комментировать их назначение нет. Наконец, на вкладке **Рисунки** можно определить пиктограмму или изображение для отображения на кнопке (рис. 4.30).

В списке **Имя свойства** выбирают **Bitmap** для загрузки изображения или **Icon** — для пиктограммы. Далее кнопкой **Обзор** осуществляют поиск нужного файла в системе.

Пример 4.16. Кнопка

На рис. 4.31 показано, как будет выглядеть кнопка, если в качестве ее названия указать **Щелкать здесь**, шрифт установлен Arial размера 12 полужирный подчеркнутый, выбран режим автоматического масштабирования (опция **Auto Size**), а вместо структурного заполнителя введено `MyButton`.

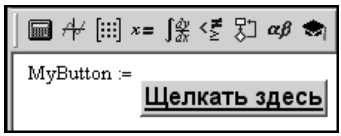


Рис. 4.31. Кнопка после применения настроек

Теперь не мешало бы выяснить, что получено в результате. Для этого поступим следующим образом: проверим значение переменной `MyButton`, которой фактически присваивался элемент управления. Это значение, как несложно заметить, равно нулю (рис. 4.32).

Интересная ситуация складывается, если щелкнуть на созданной кнопке. Значение переменной `MyButton` изменится с нулевого на единичное (рис. 4.33).

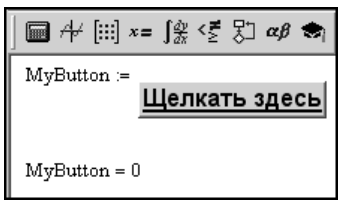


Рис. 4.32. Проверка значения переменной `MyButton`

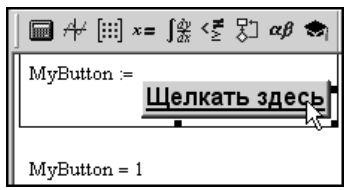


Рис. 4.33. Изменение значения переменной `MyButton` в результате щелчка на кнопке

Если на кнопке щелкнуть еще раз, значение переменной снова станет нулевым, и т. д. Почему происходит именно так, а не иначе, станет понятно после прочтения следующего раздела. Здесь заметим только, что настройки безопасности приложения могут заблокировать выполнение макросов. Чтобы увидеть, как значение переменной `MyButton` меняется при нажатии кнопки, придется вызвать команду **Enable Evaluation** например, из раскрывающегося списка справки для кнопки или из меню **Tools**.

Язык сценариев VBScript

Сценарии при работе в Mathcad могут использоваться достаточно широко, причем не только с элементами управления. Обычно для этого выбирают один из двух языков: JScript или VBScript. По умолчанию при составлении

сценариев для элементов управления в Mathcad установлен язык VBScript. По большому счету VBScript — это язык Visual Basic (сокращенно VB), адаптированный для работы с документами гипертекстовой разметки, т. е. HTML. Весьма полезным для читателя окажется знакомство с такой популярной разновидностью языка Visual Basic, как VBA (сокращение от Visual Basic for Applicaions). Это язык составления макросов для приложений пакета Microsoft Office. Далее приводятся краткие сведения по языку VB. Дело в том, что на практике сценарии, создаваемые для приложения Mathcad, подразумевают методы, характерные именно для языка VB (или VBA).

Переменные и константы

Общей для всех языков программирования является концепция переменных и констант. Под переменной понимают область памяти, к которой можно ссылаться по имени. В этой области могут быть записаны различные данные, изменяющиеся в процессе выполнения кода. Константы отличаются от переменных тем, что их значения в ходе выполнения программы не меняются. Для того чтобы использовать переменные и константы в сценарии, их следует описать или инициализировать. Инициализация переменных осуществляется с помощью ключевого слова `Dim`. Точнее, указывается ключевое слово `Dim`, а после него имя переменной. Так можно описывать сразу несколько переменных. В этом случае их имена разделяются запятыми. Константы описываются практически так же, с той лишь разницей, что ключевым словом будет `Const`, и, кроме этого, вслед за именем константы после знака равенства указывают ее значение. Это так называемые константы пользователя, они определяются непосредственно пользователем. Кроме них есть еще и встроенные константы. Их значение определять нет необходимости. Эти константы предопределены системой и к ним можно обращаться по имени. Существуют самые различные константы, которые предназначены для определения цвета, логических значений, даты, текста, а также для работы с диалоговыми окнами. Далее в табл. 4.3—4.9 приводятся константы VBScript. Для удобства они разбиты по группам. В табл. 4.3 перечислены основные константы VBScript для определения цвета.

Таблица 4.4 содержит константы для работы с датой и временем.

Кроме этих констант, при работе с датой и временем часто нужны константы форматирования, позволяющие определять способ отображения данных типа "дата" и "время" (табл. 4.5).

Текстовые константы, т. е. константы для работы с текстовыми символами, приведены в табл. 4.6.

Таблица 4.3. *Константы VBScript для определения цвета*

Имя	Значение (шестнадцатеричные числа)	Описание (Цвет)
vbBlack	&h00	Черный
vbRed	&hFF	Красный
vbGreen	&hFF00	Зеленый
vbYellow	&hFFFF	Желтый
vbBlue	&hFF0000	Синий
vbMagenta	&hFF00FF	Алый
vbCyan	&hFFFF00	Голубой
vbWhite	&hFFFFFF	Белый

Таблица 4.4. *Константы VBScript для работы с датой и временем*

Имя	Значение (число)	Описание
vbSunday	1	Воскресенье
vbMonday	2	Понедельник
vbTuesday	3	Вторник
vbWednesday	4	Среда
vbThursday	5	Четверг
vbFriday	6	Пятница
vbSaturday	7	Суббота
vbUseSystemDayOfWeek	0	День недели, который в выбранной системе настроек соответствует первому дню недели
vbFirstJan1	1	Неделя, на которую выпадает 1 января
vbFirstFourDays	2	Первая неделя в году, в которой, по крайней мере, четыре дня (т. е. хотя бы четыре дня недели попадают на новый год)
vbFirstFullWeek	3	Первая полная неделя в году

Таблица 4.5. *Константы VBScript для работы с форматами даты и времени*

Имя	Значение (число)	Описание
vbGeneralDate	0	Отображение даты и времени по значению действительного числа. Целая часть действительного числа определяет дату, а дробная — время
vbLongDate	1	Отображение даты в "длинном" формате, предусмотренном системными локальными (региональными) настройками
vbShortDate	2	Отображение даты в "коротком" формате, предусмотренном системными локальными (региональными) настройками
vbLongTime	3	Отображение времени в "длинном" формате, предусмотренном системными локальными (региональными) настройками
vbShortTime	4	Отображение времени в "коротком" формате, предусмотренном системными локальными (региональными) настройками

Таблица 4.6. *Текстовые константы VBScript*

Имя	Значение (символ, возвращаемый по коду)	Описание
vbCr	Chr(13)	Возврат каретки
VbCrLf	Chr(13) & Chr(10)	Возврат каретки с переходом строки
vbFormFeed	Chr(12)	Символ перехода формы. В Microsoft Windows не используется
vbLf	Chr(10)	Символ перехода строки
vbNewLine	Chr(13) & Chr(10) или Chr(10)	Символ новой строки. Зависит от настроек системы
vbNullChar	Chr(0)	Символ, имеющий значение 0
vbNullString	Нулевая строка, имеет значение 0	Не одно и то же, что пустая строка. Используется при вызове внешних процедур
vbTab	Chr(9)	Горизонтальная табуляция
vbVerticalTab	Chr(11)	Вертикальная табуляция. Не используется в Microsoft Windows

Помимо этих констант, при работе с VBScript широко используются константы процедур сравнения: константа бинарного сравнения `vbBinaryCompare` со значением 0 и константа `vbTextCompare` текстового сравнения со значением 1. Полезными могут быть константы `vbTrue` (логическое значение "истина"), `vbFalse` (логическое значение "ложь"), а также `vbUseDefault` (значение по умолчанию в соответствии с региональными настройками системы). Константа же `vbObjectError` (значение –2147221504) дает нижнюю границу для определяемых пользователем кодов ошибок.

Существуют константы, определяющие подтипы переменных (в VBScript все переменные имеют тип `Variant`, однако у него есть подтипы). Для того чтобы константы были доступны в проекте, должна быть явная ссылка на библиотеки, где соответствующие подтипы описаны. При работе с VBScript константы из табл. 4.7 приходится описывать явно.

Таблица 4.7. *Константы VBScript для определения подтипов*

Имя	Значение	Описание
`vbEmpty`	0	Неинициализированная переменная (по умолчанию)
`vbNull`	1	Переменная не содержит данных
`vbInteger`	2	Целочисленный подтип `Integer`
`vbLong`	3	Подтип `Long` — целочисленный удлиненный
`vbSingle`	4	Подтип `Single` — действительное число
`vbDouble`	5	Подтип `Double` — действительное число двойной точности
`vbCurrency`	6	Подтип `Currency` — специальный подтип (действительное число) для работы с валютными расчетами
`vbDate`	7	Подтип `Date` для работы с датой и временем
`vbString`	8	Подтип `String` — текстовое значение
`vbObject`	9	Объектный подтип `Object`
`vbError`	10	Подтип `Error` для работы с ошибками
`vbBoolean`	11	Логический подтип `Boolean`
`vbVariant`	12	Подтип `Variant`, используемый для массивов
`vbDataObject`	13	Подтип объекта данных
`vbDecimal`	14	Подтип `Decimal`
`vbByte`	17	Подтип `Byte`
`vbArray`	8192	Подтип, определяющий массив

Наконец, в табл. 4.8 приведены константы, являющиеся аргументами функции `MsgBox()`. Сама функция описывается позже. Отметим только, что с ее помощью выводятся окна сообщений и поэтому она достаточно популярна.

Таблица 4.8. *Константы VBScript для использования с функцией* `MsgBox()`

Имя	Значение	Описание
`vbOKOnly`	`0`	Отображение в окне сообщения единственной кнопки **OK**
`vbOKCancel`	`1`	Отображение в окне сообщения кнопок **OK** и **Cancel**
`vbAbortRetryIgnore`	`2`	Отображение кнопок **Abort**, **Retry** и **Ignore**
`vbYesNoCancel`	`3`	Отображение кнопок **Yes**, **No** и **Cancel**
`vbYesNo`	`4`	Отображение кнопок **Yes** и **No**
`vbRetryCancel`	`5`	Отображение кнопок **Retry** и **Cancel**
`vbCritical`	`16`	Отображение пиктограммы как в сообщении об ошибке
`vbQuestion`	`32`	Отображение пиктограммы окна запроса
`vbExclamation`	`48`	Отображение пиктограммы окна предупреждения
`vbInformation`	`64`	Отображение пиктограммы информационного окна
`vbDefaultButton1`	`0`	Активна по умолчанию первая кнопка
`vbDefaultButton2`	`256`	Активна по умолчанию вторая кнопка
`vbDefaultButton3`	`512`	Активна по умолчанию третья кнопка
`vbDefaultButton4`	`768`	Активна по умолчанию четвертая кнопка
`vbApplicationModal`	`0`	Пользователь должен закрыть диалоговое окно для продолжения работы с приложением
`vbSystemModal`	`4096`	Пользователь должен закрыть диалоговое окно для продолжения работы с системой

Таким образом, окно сообщения может иметь несколько кнопок, поэтому необходимо как-то определять, какая кнопка была выбрана пользователем. В этом случае полезны константы из табл. 4.9. К сожалению, для работы с ними в VBScript их нужно предварительно инициализировать в явном виде.

Таблица 4.9. *Константы VBScript определения результата для использования с функцией* `MsgBox()`

Имя	Значение	Описание
`vbOK`	1	Выбрана кнопка **OK**
`vbCancel`	2	Выбрана кнопка **Cancel**
`vbAbort`	3	Выбрана кнопка **Abort**
`vbRetry`	4	Выбрана кнопка **Retry**
`vbIgnore`	5	Выбрана кнопка **Ignore**
`vbYes`	6	Выбрана кнопка **Yes**
`vbNo`	7	Выбрана кнопка **No**

Важным моментом при определении констант и переменных является область их доступности. Следует отметить, что переменные и константы определяются в процедурах, функциях и модулях. Правило простое: переменная или константа доступна в области того блока, где она определена. Если она определена в процедуре, то вызывать эту переменную можно только в теле этой процедуры. Такие переменные называют локальными. Если переменная описана в модуле, то на нее можно ссылаться в каждой процедуре или функции этого модуля. В этом случае говорят, что переменная глобальная. Кроме того, на область доступности объектов (не только переменных и констант, но функций и процедур) существенно влияет наличие или отсутствие в описании ключевых слов `Public` и `Private`. Если переменная описана одним из этих ключевых слов, то в ключевом слове `Dim` нет необходимости. Переменная, инициализированная с ключевым словом `Private`, доступна только в пределах того сценария, где она описана. Чтобы переменная была доступна во всех процедурах всех сценариев, прибегают к помощи ключевого слова `Public`.

Заметим, что в сценариях VBScript переменные можно и не объявлять, а сразу вставлять в программный код. В этом случае говорят о неявном объявлении переменной. Данный подход прост, однако не очень эстетичен. Если в программном коде присутствует много имен (функций, процедур, переменных, констант и т. п.), без явного описания в них бывает крайне сложно разобраться.

Массивы

Несколько особое место занимают такие структуры данных, как массивы. В общем случае это набор однотипных переменных с одинаковым именем, а к отдельной переменной из набора можно обращаться по имени массива и индексу (или индексам) переменной в этом массиве. При описании массива

необходимо указать его размерность и, как правило, размер. Размерность массива определяет количество индексов, по которым однозначно идентифицируется переменная массива. Размер определяет области изменения индексов. Если размер массива известен до начала выполнения программы, говорят, что массив статический. В противном случае массивы являются динамическими, и число элементов в таком массиве определяется уже в процессе реализации программы.

Описываются массивы практически так же, как и обычные переменные, с той лишь разницей, что вместо имени переменной указывается имя массива, а в скобках за ним — верхние индексы по каждой из размерностей (в случае статического массива). Если массив динамический, то скобки после имени массива остаются пустыми. Например, инструкцией `Dim A(10)` определяется массив `A` из 11 переменных. Элементов 11 потому, что по умолчанию нижний индекс — 0, а верхний индекс указан в данном случае как 10.

Пример объявления трехмерного массива может быть таким: `Dim B(5,4,7)`. В этом массиве первый индекс изменяется от 0 до 5, второй — от 0 до 4 и третий — от 0 до 7. Чтобы обратиться к элементу массива, нужно после имени массива указать соответствующие индексы. Командой `B(0,2,5) = 1` элементу с индексами 0, 2 и 5 в качестве значения присваивается число 1.

Как отмечалось, динамические массивы инициализируются без указания границ изменения индексов. Инструкцией `Dim D()` описывается динамический массив. Размерность и размер этого массива определяется в процессе выполнения кода. Предположим, что массив одномерный и верхняя граница изменения индекса задается целочисленной переменной `N`, которая, в свою очередь, вводится пользователем в процессе выполнения программы. Таким образом, размер массива становится известен уже после того, как программный код запущен на выполнение. Для рассматриваемого динамического массива после того, как значение переменной `N` установлено, командой `ReDim D(N)` определяют размер массива. Инструкцию `ReDim` можно использовать столько, сколько нужно в теле программы, однако следует помнить, что при переопределении параметров динамического массива вся информация о его текущих элементах теряется. Поэтому временами полезно расширять размеры массива с помощью инструкции `ReDim Preserve`. Например, командой `ReDim Preserve D(N+5)` размер массива увеличивается на 5 элементов, при этом первоначальные элементы массива не теряются.

Функции VBScript

В VBScript по сравнению, например, с VBA, функций, конечно, меньше, однако для эффективной работы вполне хватает. В табл. 4.10 эти функции перечислены в алфавитном порядке. Там же есть их краткое описание. Те функции,

что будут часто встречаться в примерах книги, описываются более подробно по мере их использования.

Таблица 4.10. *Функции VBScript*

Функция	Описание
`Abs()`	Функция вычисления абсолютного значения
`Array()`	Функция создания массива данных
`Asc()`	Функция в качестве результата возвращает численный код первого символа текстовой строки, указанной аргументом функции
`Atn()`	Арктангенс числа, указанного аргументом функции
`CBool()`	Преобразование аргумента функции к логическому подтипу `Boolean`
`CByte()`	Преобразование аргумента функции к численному подтипу `Byte`
`CCur()`	Преобразование аргумента функции к численному подтипу `Currency`
`CDate()`	Преобразование аргумента функции к подтипу даты `Date`
`CDbl()`	Преобразование аргумента функции к численному подтипу `Double`
`Chr()`	Функция по коду символа возвращает текстовый символ
`CInt()`	Преобразование аргумента функции к численному подтипу `Integer`
`CLng()`	Преобразование аргумента функции к численному подтипу `Long`
`Cos()`	Функция вычисления косинуса
`CreateObject()`	Функция создания объекта
`CSng()`	Преобразование аргумента функции к численному подтипу `Single`
`CStr()`	Преобразование аргумента функции к текстовому подтипу `String`
`Date()`	Функция в качестве значения возвращает текущую системную дату
`DateAdd()`	Функция возвращает дату, получающуюся из третьего аргумента, путем добавления интервала (второй аргумент функции). Первый аргумент — символ, определяющий размерность указанного вторым аргументом интервала

Таблица 4.10 *(продолжение)*

Функция	Описание
`DateDiff()`	Функция вычисляет временной интервал между двумя датами
`DatePart()`	Функция возвращает определенную часть даты (год, месяц, неделя и т. п.), указанную аргументом функции
`DateSerial()`	Функцией по году, месяцу и дню (аргументы функции) возвращается дата
`DateValue()`	Функция по аргументу (дате в текстовом формате) возвращает дату
`Day()`	По аргументу (дате) вычисляется количество дней
`Escape()`	Перекодировка строки в символы кода ASCII
`Eval()`	Функция возвращает результат вычисления выражения
`Exp()`	Экспоненциальная функция
`Filter()`	Функция возвращает массив, выделенный из строки на основе фильтра (критерия)
`FormatCurrency()`	Возвращает денежное значение с использованием символа валюты (определяется системными настройками)
`FormatDateTime()`	Результат вычисления выражения возвращается в формате даты или времени
`FormatNumber()`	Результат вычисления выражения возвращается в формате числа
`FormatPercent()`	Результат вычисления выражения возвращается в процентном формате
`GetLocale()`	Возвращает текущее значение локального идентификатора
`GetObject()`	Возвращает ссылку на объект
`GetRef()`	Возвращает ссылку на процедуру, связываемую с событием
`Hex()`	Преобразование числа к шестнадцатеричной системе счисления
`Hour()`	По текущему времени определяется полное число часов
`InputBox()`	Отображение диалогового окна с полем ввода значения, которое затем считывается

Таблица 4.10 *(продолжение)*

Функция	Описание
`InStr()`	Функция возвращает позицию, начиная с которой одна строка повторяется в другой
`InStrRev()`	Функция возвращает позицию с конца строки, начиная с которой одна строка повторяется в другой
`Int()`	Функция возвращает целую часть числа, указанного аргументом функции. Если число отрицательное, то оно округляется до целого числа, не большего аргумента
`Fix()`	Функция возвращает целую часть числа, указанного аргументом функции. Если число отрицательное, то оно округляется до целого числа, не меньшего аргумента
`IsArray()`	Проверка, является ли аргумент функции массивом
`IsDate()`	Проверка возможности преобразования аргумента функции к значению типа дата
`IsEmpty()`	Проверка выполнения инициализации переменной
`IsNull()`	Проверка, содержит ли аргумент данные
`IsNumeric()`	Проверка, можно ли преобразовать аргумент к значению численного типа
`IsObject()`	Проверка, является ли аргумент переменной объектного типа
`Join()`	Функция формирует строку путем объединения подстрок
`LBound()`	Функция возвращает нижнюю границу массива
`LCase()`	Перевод символов строки в нижний регистр
`Left()`	Выделение подстроки из строки, содержащей указанное число символов от начала строки
`Len()`	Длина строки в символах или байтах
`LoadPicture()`	Возвращается объект рисунка
`Log()`	Функция вычисления натурального логарифма от аргумента функции
`LTrim()`	Копия строки без начальных пробелов
`Mid()`	Выделение из строки (первый аргумент) подстроки определенной длины (второй аргумент), начиная с указанного символа (третий аргумент)

Таблица 4.10 *(продолжение)*

Функция	Описание
`Minute()`	По текущему времени определяется полное число минут
`Month()`	По дате вычисляется полное число месяцев
`MonthName()`	Функция по номеру месяца возвращает его текстовое название
`MsgBox()`	Отображение диалогового окна сообщения
`Now()`	Функция возвращает текущую системную дату и время
`Oct()`	Преобразование числа к восьмеричной системе счисления
`Replace()`	С помощью функции в текстовой строке выполняется замена одних подстрок на другие
`RGB()`	Возвращает числовое значение, определяющее цвет в формате RGB
`Right()`	Функция из строки, начиная с указанной позиции, возвращает подстроку установленной длины
`Rnd()`	Функция генерирования случайного числа в диапазоне значений от 0 до 1. Аргумент функции может не указываться. Если он указан, то используется в качестве базового значения для формирования последовательности случайных чисел
`Round()`	Округление числа (первый аргумент) до количества знаков, указанных вторым аргументом
`RTrim()`	Копия строки без конечных пробелов
`ScriptEngine()`	Возвращает текстовую строку, сообщающую о том, какой язык сценариев используется
`ScriptEngineBuildVersion()`	Номер версии устройства обработки сценариев
`ScriptEngineMajorVersion()`	Основной номер версии устройства обработки сценариев
`ScriptEngineMinorVersion()`	Дополнительный номер версии устройства обработки сценариев
`Second()`	Функция по времени определяет количество секунд
`SetLocale()`	Установка локального идентификатора

Таблица 4.10 *(продолжение)*

Функция	Описание
`Sgn()`	Функция возвращает знак числа — аргумента функции. Для нуля возвращается нулевое значение
`Sin()`	Функция вычисления синуса
`Space()`	Функция возвращает строку, состоящую из указанного аргументом функции числа пробелов
`Split()`	Функция разбивает текстовую строку на подстроки, которые формируют массив. Элементами массива являются подстроки. Строка на подстроки разбивается в соответствии с размещением в строке символов-разделителей
`Sqr()`	Корень квадратный из числа — аргумента функции
`StrComp()`	Функция сравнения строк. Результатом является число
`String()`	Функция формирует текстовую строку, получающуюся путем повторения символа (второй аргумент). Число повторений определяется первым аргументом функции
`StrReverse()`	Функция возвращает строку, получающуюся из указанной аргументом функции строки обращением порядка следования символов
`Tan()`	Функция вычисления тангенса
`Time()`	Функция возвращает системное время
`Timer()`	Функция возвращает число секунд, начиная с полуночи до текущего момента
`TimeSerial()`	По аргументам функции (часы, минуты, секунды) возвращается момент времени
`TimeValue()`	Функция преобразует время в текстовом формате к формату даты `Date`
`Trim()`	Копия строки без начальных и конечных пробелов
`TypeName()`	Функция возвращает сведения о переменной, указанной аргументом функции
`UBound()`	Функция возвращает верхнюю границу массива
`UCase()`	Перевод символов строки в верхний регистр
`Unescape()`	Декодировка текстовой строки, перекодированной с помощью функции `Escape()`

Таблица 4.10 *(окончание)*

Функция	Описание
`VarType()`	Функция возвращает целое число, определяющее подтип указанной аргументом функции переменной
`Weekday()`	Функция по дате возвращает день недели (число)
`WeekdayName()`	Функция по дате возвращает день недели (текстовую строку)
`Year()`	Функция по дате определяет полное число лет

На основе встроенных функций VBScript можно создавать собственные. Примером могут служить производные математические функции, такие как гиперболический синус и косинус, секанс и косеканс и ряд других.

Что касается вызова функций (и процедур) в программном коде, то делается это как обычно, указанием имени функции (процедуры) с передачей аргументов (если необходимо) в круглых скобках. Однако есть и другой способ передачи аргументов. Их можно указывать после названия функции (без скобок), разделяя запятыми. В обоих случаях следует строго соблюдать предопределенный порядок аргументов. Однако если у функции или процедуры аргументы именованные (т. е. аргументы имеют свое имя), то передавать их функции можно в произвольном порядке. В этом случае указываются разделенные запятыми пары: имя аргумента и его значение. Имя и значение разделяются специальными операторами: двоеточие и знак равенства (`:=`).

Операторы VBScript

В VBScript существуют самые разные операторы, как правило, соответствующие основным математическим и логическим операциям. В табл. 4.11 представлены арифметические операторы VBScript.

Таблица 4.11. *Арифметические операторы VBScript*

Оператор	Пример вызова	Описание
`+`	`a + b`	Оператор сложения. Результатом операции является сумма чисел-операндов выражения
`-`	`a - b`	Оператор вычитания. Результатом является разность операндов, входящих в соответствующее выражение

Таблица 4.11 *(окончание)*

Оператор	Пример вызова	Описание
`*`	`a * b`	Оператор умножения. Результат — произведение операндов выражения
`/`	`a / b`	Оператор деления. Результатом является частное операндов выражения
`\`	`a \ b`	Оператор целочисленного деления. Результат — целая часть частного операндов выражения
`^`	`a ^ b`	Оператор возведения в степень. Результатом является число, равное первому операнду в степени, определяемой вторым операндом
`Mod`	`a Mod b`	Оператор деления по модулю. Результат — остаток от целочисленного деления операндов выражения
`=`	`a = b`	Оператор присваивания. Переменной, указанной в левой части оператора, присваивается значение, определяемое выражением в правой части оператора

Приоритетность операторов определяется приоритетностью соответствующих математических операций. Однако для надежности можно порекомендовать скобки, которые не только позволяют изменять естественный порядок выполнения операций, но и структурируют программный код, делая его более удобным для чтения.

Переходными между арифметическими и логическими операторами являются операторы сравнения, которые можно найти в табл. 4.12.

Таблица 4.12. *Операторы сравнения VBScript*

Оператор	Пример вызова	Описание
`=`	`a = b`	Оператор *равно*. Результат — значение `True`, если оба операнда имеют одинаковые значения. В противном случае — `False`
`>`	`a > b`	Оператор *больше*. Результат — значение `True`, если первый операнд строго больше второго. В противном случае — `False`

Таблица 4.12 *(окончание)*

Оператор	Пример вызова	Описание
`<`	`a < b`	Оператор *меньше*. Результат — значение `True`, если первый операнд строго меньше второго. В противном случае — `False`
`>=`	`a >= b`	Оператор *не меньше*. Результат — значение `True`, если первый операнд не меньше второго. В противном случае — `False`
`<=`	`a <= b`	Оператор *не больше*. Результат — значение `True`, если первый операнд не больше второго. В противном случае — `False`
`<>`	`a <> b`	Оператор *не равно*. Результат — значение `True`, если первый операнд не равен второму. В противном случае — `False`
`Is`	`a Is b`	Оператор сравнения объектов. Оба операнда должны быть ссылками на объект, т. е. иметь тип `Object`. Результат равен `True`, если оба операнда ссылаются на один и тот же объект. В противном случае — `False`

Результатом вычисления выражений, содержащих операторы сравнения, может быть значение `Null`, если хотя бы один из операндов равен `Null` (не имеет значения).

Логических операторов в VBScript всего шесть (табл. 4.13).

Таблица 4.13. *Логические операторы VBScript*

Оператор	Пример вызова	Описание
`And`	`a And b`	Логическое *И*. Результат равен `True`, если оба операнда равны `True`. В противном случае — `False`
`Eqv`	`a Eqv b`	Оператор логического равенства. Результат равен `True`, если значения операндов совпадают, и `False` в противном случае

Таблица 4.13 *(окончание)*

Оператор	Пример вызова	Описание
`Imp`	`a Imp b`	Оператор импликации. Результат равен `False` только в том случае, если первый операнд равен `True`, а второй равен `False`. В противном случае результатом будет `True`
`Not`	`Not a`	Логическое отрицание. Это унарный оператор. Результат его выполнения равен `True`, если операнд равен `False`, и наоборот: если операнд равен `False`, результатом будет `True`
`Or`	`a Or b`	Логическое *ИЛИ*. Результат равен `True`, если хотя бы один из операндов равен `True`. В противном случае результат равен `False`
`Xor`	`a Xor b`	Логическое исключающее *ИЛИ*. Результат равен `True`, если только один из операндов равен `True`. В противном случае результат равен `False`

Таблица значений логических операторов, при условии, что один или оба оператора принимают значение `Null`, имеет некоторые особенности. Останавливаться здесь на этом не будем, однако если такая ситуация встретится (т. е. одним из значений может оказаться `Null`), следует помнить, что не всегда, если один из операндов равен `Null`, будет таким же и конечный результат.

Кроме упомянутых, успешно используется в программных кодах оператор конкатенации, с помощью которого выполняется объединение текстовых строк. Он обозначается символом `&`. Также нужно учесть, что если по сравнению к текстовым строкам применяется операция арифметического сложения, то сложение в такой ситуации интерпретируется как конкатенация.

Условные операторы VBScript

В любой мало-мальски приличной программе есть точки ветвления алгоритма. Часто они реализуются с помощью условных операторов. В условном операторе проверяется условие, и если оно истинно, выполняется оператор или последовательность операторов; если условие ложно, выполняется другой оператор или последовательность операторов. Существует упрощенный вариант условного оператора, когда оператор или последовательность операторов выполняется при истинном условии (т. е. на случай несоблюдения

условия операторы не предусмотрены). В VBScript условный оператор реализуется с помощью конструкции `If…Then…Else` (упрощенному варианту условного оператора соответствует конструкция `If…Then`).

После ключевого слова `If` указывается условие, проверяемое на предмет истинности. После этого условия следует ключевое слово `Then`. В случае истинности условия будут выполнены все операторы, указанные (с новой строки) вслед за ключевым словом `Then`. Операторы, выполняемые при несоблюдении условия, указываются (также с новой строки) за ключевым словом `Else`. Заканчивается условный оператор инструкцией `End If`.

Существует упрощенный вариант условного оператора. Все инструкции и операторы размещаются в одну строку. Конечная команда `End If` при этом не нужна.

Нередко при составлении программ условные операторы бывают вложенными друг в друга, т. е. в теле условного оператора вызываются другие условные операторы и т. д. Несколько вложенных условных операторов существенно усложняют восприятие кода. Полезным может оказаться модифицированный условный оператор `If…Then…ElseIf`. Оператор реализуется по следующей схеме. Сначала проверяется условие после инструкции `If`. Если оно истинно, выполняется группа операторов в блоке `Then`. После инструкции `Then` следует инструкция `ElseIf` с условием, которое проверяется при неистинности первого условия. Операторы, указанные в этом блоке после инструкции `Then`, выполняются при истинности второго условия. Далее может следовать еще один блок `ElseIf`, операторы которого (после инструкции `Then`) выполняются при соблюдении соответствующего условия, и т. д. Последний блок операторов после инструкции `Else` выполняется, если среди всех проверенных условий не нашлось истинных.

На практике структуры из большого числа вложенных операторов встречаются редко. Альтернативой может стать оператор выбора `Select…Case`. Синтаксис вызова этого оператора следующий. Сначала указываются ключевые слова `Select Case`. После этой инструкции записывается переменная (или выражение), значение которой (или которого) проверяется. Возможные варианты перечислены после инструкций `Case`. Там же указаны операторы, выполняемые в случае совпадения значений. При этом проверочных значений после инструкции `Case` может быть несколько (они разделяются запятыми). Последней в операторе выбора может быть инструкция `Case Else`. Проверочное значение для этой инструкции не указывается, а соответствующие операторы выполняются, если проверяемое значение не совпало ни с одним предыдущим проверочным значением. Заканчивается оператор выбора инструкцией `End Select`.

Операторы циклов VBScript

Часто программы составляются для того, чтобы автоматизировать процесс многократного повторения однотипных действий. Незаменимы при этом операторы циклов. В VBA есть несколько стандартных способов их реализации. Вначале рассмотрим конструкцию `For…Next`.

Оператор цикла `For…Next` обычно удобен в тех случаях, когда нужно последовательно повторить определенное число однотипных действий. В операторе имеется переменная-счетчик, которая с начальным значением указывается после ключевого слова `For`. Далее следует ключевое слово `To`, после которого указывают конечное значение переменной-счетчика. Наконец, после ключевого слова `Step` можно указать шаг изменения переменной-счетчика. Если шаг не указан, он по умолчанию полагается равным единице. После этого следует серия операторов, выполняемых при фиксированном значении переменной-счетчика. Заканчивается последовательность операторов инструкцией `Next`, после которой записывают название переменной-счетчика, хотя последнее делать необязательно.

Не всегда последовательность выполняемых операторов удается связать с переменной-счетчиком. Нередко складывается такая ситуация, когда группа операций выполняется по отношению к объектам из определенного множества. Специально для этого предназначен оператор `For Each…Next`. Синтаксис вызова оператора следующий. После инструкции `For Each` указывается имя переменной для перебора объектов из базовой коллекции (множества) объектов. Это множество следует за ключевым словом `In`. Далее перечисляют выполняемые над объектами коллекции операции, а заканчивается этот блок инструкцией `Next` (можно указать переменную перебора объектов).

Ранее рассматривались операторы цикла, в которых критерием окончания процесса было достижение переменной-счетчиком граничного значения и перебор всех объектов из базовой коллекции. Еще одним критерием может быть выполнение определенного условия. Другими словами, последовательность действий выполняется до тех пор, пока некоторое условие не будет выполнено. Реализовано это может быть так. Сначала, например, проверяется условие, и если оно не является истинным, выполняется последовательность операторов. После этого условие проверяется снова. Если оно остается неистинным, снова выполняются операторы, и так до тех пор, пока ситуация не изменится и условие не станет истинным. В VBA по такой схеме работает оператор `Do…Loop`. В рамках этого цикла вначале или конце указывается проверяемое условие. Условие может указываться после ключевого слова `While` или `Until`. Если условие указано после ключевого слова `While`, то оператор цикла выполняется, пока условие не станет ложным, т. е. для выполнения

оператора нужно, чтобы условие было истинным. Чтобы оператор цикла выполнялся при невыполнении условия, задают ключевое слово `Until`. Если условие указано в начале оператора цикла, то оно проверяется перед непосредственным выполнением команд цикла. Теоретически возможна ситуация (в зависимости от истинности условия), когда до выполнения команд дело вообще не дойдет. Если условие размещено в конце оператора цикла, то серия команд будет выполнена, по крайней мере, один раз, и только после этого проверяется условие.

Операторы циклов `Do…Loop` — мощный инструмент при создании программных кодов, который таит, однако, и скрытые опасности. В основном они связаны с корректной обработкой и описанием условий выхода из цикла. Ведь даже малейшая неточность может привести к серьезным сбоям в работе программы.

Процедуры и функции пользователя

Процедуры и функции VBScript выделяются в специальные программные блоки. Для процедур инструкцией начала блока является ключевое слово `Sub`. Функция обычно начинается с ключевого слова `Function` (обычно, потому что перед ключевыми словами `Sub` и `Function` могут размещаться инструкции `Private` и `Public`). Заканчивается блок процедуры инструкцией `End Sub`, а функции — `End Function`.

Имя функции или процедуры указывается после ключевого слова начала соответствующего блока. В круглых скобках перечисляются аргументы, если такие есть. Конкретный код процедур и функций существенно зависит от того, для чего эти процедуры и функции создаются. Далее будут рассмотрены небольшие примеры.

Что касается ключевых слов `Private` и `Public`, то их назначение и результат применения при работе с процедурами и функциями практически такие же, как и в случае переменных. Процедура, описанная как `Private`, доступна в пределах содержащего ее сценария. Чтобы расширить область доступности, рекомендуется указать ключевое слово `Public`.

Отображение диалоговых окон

Далеко не последний момент при работе с различными приложениями, в том числе и Mathcad, — взаимодействие и обмен данными пользователя и приложения. Исключительно удобно делать это с помощью диалоговых окон. В сценариях VBScript обычно программными методами открывают окна двух типов: окно сообщения и окно ввода значения. Остановимся на этом подробнее.

Для отображения окна сообщения применяют функцию `MsgBox()`. Она в общем случае может иметь до пяти именованных аргументов, из которых обязателен только один, — первый. Аргументы функции перечислены в табл. 4.14.

Таблица 4.14. *Аргументы функции* `MsgBox()`

Имя аргумента	Описание
`prompt`	Обязательный аргумент. Определяет текст сообщения, выводимого в диалоговом окне
`buttons`	Необязательный аргумент. Значение, равное сумме чисел, определяющих вид окна сообщения, точнее, количество и тип отображаемых в окне кнопок, тип пиктограмм, а также кнопку, активную по умолчанию. В качестве аргумента можно указать одну из встроенных констант VBA (см. табл. 4.5). По умолчанию (если этот аргумент не указан) отображается окно с единственной кнопкой подтверждения **OK**
`title`	Необязательный текстовый аргумент. Определяет название окна сообщения. Если аргумент не указан, в строке названия окна отображается название приложения
`helpfile`	Необязательный аргумент. Текстовое выражение, определяющее файл справки. Этот аргумент указывается только вместе с аргументом `context`
`context`	Необязательный аргумент. Численное значение контекста справки. Аргумент указывается только вместе с аргументом `helpfile`

Какие аргументы можно передавать этой функции, уже неявно упоминалось при описании констант, передаваемых в качестве аргументов функции `MsgBox()` (см. табл. 4.8). Задавая те или иные аргументы, можно получить окно практически любого вида.

Окно ввода характеризуется тем, что в нем есть специальное поле для ввода пользователем значения. Обычно так передаются аргументы вызываемых в программном коде функций и процедур. Вызывается окно ввода с помощью функции `InputBox()`, аргументы которой перечислены в табл. 4.15. Они указаны в том порядке, в котором их следует передавать.

Таблица 4.15. *Аргументы функции* `InputBox()`

Аргумент	Описание
`prompt`	Обязательный аргумент. Текстовое сообщение, отображаемое в диалоговом окне в качестве подсказки
`title`	Необязательный аргумент. Текстовое значение, используемое в качестве названия диалогового окна
`default`	Необязательный аргумент. Текстовое выражение, отображаемое по умолчанию в строке ввода
`xpos`	Необязательный аргумент. Служит для позиционирования диалогового окна по горизонтали. Определяет расстояние от левого верхнего угла диалогового окна до левого края экрана. Если аргумент не указан, окно центрируется по горизонтали
`ypos`	Необязательный аргумент. Служит для позиционирования диалогового окна по вертикали. Определяет расстояние от левого верхнего угла диалогового окна до верхнего края экрана. Если аргумент не указан, окно центрируется по вертикали
`helpfile`	Необязательный аргумент. Аргумент содержит ссылку на файл справки. Используется только вместе с аргументом `context`
`context`	Необязательный аргумент. Численное контекстное значение темы справки. Аргумент используется в паре с аргументом `helpfile`

Объекты, методы и свойства

Если рассматривать VBScript с точки зрения перспективы применения этого языка для работы с приложением Mathcad, то, очевидно, не последнее место занимает необходимость взаимодействия с элементами рабочих документов Mathcad. Причем здесь имеются в виду не только элементы управления, но и всевозможные переменные (математические, логические, текстовые и т. п.) в рабочем листе Mathcad. Другими словами, в процессе реализации сценария бывает крайне важно знать состояние переключателей, кнопок, значения переменных, результаты вычисления выражений. Все, что здесь упоминалось, в VBScript, как и во всех объектно-ориентированных языках, рассматривается в виде объектов со своими свойствами и методами. Свойства определяют характеристики объекта, а методы — допустимые операции, которые можно выполнять с объектами. Чтобы получить доступ к свойству или методу, его указывают после ссылки на объект. Ссылки на объект и свойство (или метод) разделяются точкой.

Рабочему документу Mathcad соответствует объект `Worksheet`. В табл. 4.16 представлены свойства, которые поддерживаются этим объектом.

Методы этого объекта кратко описаны в табл. 4.17.

Таблица 4.16. *Свойства объекта* `Worksheet`

Свойство	Описание
`Application`	Свойство определяет приложение, которым управляется данный рабочий документ (объект `Worksheet`)
`FullName`	Свойство определяет полный путь к рабочему документу
`Name`	Свойство определяет название рабочего документа
`NeedsSave`	Логическое значение, определяющее наличие несохраненных изменений в рабочем документе
`Options`	Свойство определяет параметры объекта `Worksheet`
`Parent`	Свойство определяет родительский объект для объекта `Worksheet`
`Path`	Путь к рабочему документу, без указания имени файла рабочего документа
`Regions`	Свойство определяет коллекцию объектов `Region` объекта `Worksheet`
`Windows`	Свойство определяет коллекцию объектов `Window` объекта `Worksheet`

Таблица 4.17. *Методы объекта* `Worksheet`

Метод	Описание
`Close`	Закрытие документа, соответствующего объекту `Worksheet`
`GetOption`	Возвращается текущее значение выбранного параметра
`GetValue`	Возвращается значение переменной (имя переменной указывается аргументом метода) в рабочем документе объекта `Worksheet`
`PrintAll`	Вывод на печать документа объекта `Worksheet` с использованием принтера, выбранного по умолчанию
`Recalculate`	Пересчет рабочего документа
`Save`	Сохранение документа
`SaveAs`	Сохранение документа с новым именем или в ином формате. Новое имя файла и формат файла указываются аргументами метода
`SetOption`	Установка значения параметра объекта `Worksheet`. Параметр и его значение указываются аргументами метода
`SetValue`	Установка значения переменной в рабочем документе. Для установки значения элемента матрицы используют метод `MatrixValue.SetElement`. Имя переменной и значение — аргументы метода

Помимо свойств и методов, с объектами связывают также и события. События — это, фактически, те действия (со стороны пользователя или системы), на которые объект может реагировать. Процедуры, посредством которых реализуется реакция объекта на события, называются обработчиками событий. С объектом `Worksheet` связывают два события: `Recalculated` (произошел пересчет рабочего документа) и `Changed` (внесены изменения в документ).

Кратко остановимся также на некоторых иных объектах, классах и коллекциях. Они могут оказаться полезными при решении задач. Для тех, кто не очень хорошо знаком с основными подходами объектно-ориентированного программирования (ООП), отметим некоторые особенности классов и коллекций. Так, под классом обычно понимают некоторую совокупность элементов или объектов. Это одно из базовых понятий в ООП. В некотором смысле класс схож с типом, однако, это все же более общая категория. Класс может содержать подклассы. Об объекте, который принадлежит классу, говорят, что это экземпляр (или член) класса. Коллекцией объектов называют группы сходных объектов. К элементам коллекции (т. е. объектам, относящимся к коллекции) можно обращаться по имени или по индексу. Это очень удобно. Сама коллекция также является объектом. В качестве примера можно привести коллекцию `Worksheets`, имеющую два свойства. Свойство `Application` определяет приложение, к которому относится коллекция объектов `Worksheet`, а `Parent` — отвечает за родительский объект коллекции. Метод у коллекции один. Он позволяет открывать рабочие документы. Имя документа указывается аргументом метода `Open`. Ранее упоминался родительский объект. Дело в том, что одни объекты могут содержать в себе другие. Объект, содержащий текущий объект, называется родительским по отношению к текущему.

Класс `Application` полезен при создании экземпляров объекта приложения Mathcad. Он поддерживает ряд интересных свойств, которые перечислены в табл. 4.18.

Таблица 4.18. *Свойства класса* `Application`

Свойство	Описание
`ActiveWindow`	Доступ к текущему активному окну
`ActiveWorksheet`	Доступ к рабочему листу активного окна
`Application`	Определение приложения, к которому относится объект
`DefaultFilePath`	Используемый по умолчанию путь для открытия и сохранения файлов рабочих документов
`FullName`	Имя и полный путь к файлу приложения Mathcad

Таблица 4.18 *(окончание)*

Свойство	Описание
`Height`	Высота в пикселах основного окна приложения
`Left`	Расстояние в пикселах от левого края экрана до окна приложения
`Name`	Имя приложения, к которому относится объект
`Parent`	Родительский объект
`Path`	Путь к приложению Mathcad
`Top`	Расстояние в пикселах от верхнего края экрана до окна приложения
`Version`	Строка с номером версии приложения
`Visible`	Режим отображения главного окна Mathcad (отображается или нет)
`Width`	Ширина в пикселах основного окна приложения
`Windows`	Коллекция объектов `Window` — всех открытых рабочих окон Mathcad
`Worksheets`	Коллекция объектов `Worksheet` — всех открытых рабочих документов Mathcad

Методов у класса `Application` всего два: `CloseAll` (закрытие всех рабочих документов) и `Quit` (завершение сеанса работы с приложением с сохранением рабочих документов). Обрабатываемые события для класса `Application` приведены в табл. 4.19.

Таблица 4.19. *События для класса* `Application`

Событие	Описание
`Quit`	Закрытие приложения
`WindowActivated`	Активизация окна
`WindowDeactivated`	Деактивизация окна
`WorksheetClosing`	Закрытие рабочего документа
`WorksheetOpened`	Открытие документа

Класс `Window` поддерживает свойства и методы, полезные при работе с окнами рабочих документов. Свойства класса собраны и кратко описаны в табл. 4.20.

Таблица 4.20. *Свойства класса* `Window`

Свойство	Описание
`Height`	Высота окна в пикселах
`Left`	Позиция левого края окна по отношению к основному окну
`Top`	Позиция верхнего края окна по отношению к основному окну
`Width`	Ширина окна в пикселах
`WindowState`	Состояние рабочего окна
`Zoom`	Масштаб отображения окна

Для активизации окна служит метод `Activate`. Прокрутка содержимого окна осуществляется посредством метода `ScrollTo` с указанием в качестве аргументов горизонтальной и вертикальной координат для прокрутки. Наконец, для прокрутки содержимого окна до определенного места (диапазона), вызывают метод `ScrollToRegion`, аргументом которого указывается соответствующий диапазон. Сами диапазоны представляются объектами класса `Region`. Класс имеет всего три свойства: `Tag` (текст диапазона) `X` (координата левой границы диапазона) и `Y` (координата верхней границы диапазона).

С точки зрения обработки численных значений важными представляются следующие три класса. Так, класс `NumericValue` представляет все численные значения. Его свойства можно найти в табл. 4.21.

Таблица 4.21. *Свойства класса* `NumericValue`

Свойство	Описание
`AsString`	Возвращается значение в виде текстовой строки
`Imag`	Мнимая часть комплексного значения в формате числа двойной точности
`Integer`	Действительная часть числа в целочисленном формате
`Real`	Действительная часть комплексного значения в формате числа двойной точности
`Type`	Свойство определяет, к какому типу относится значение

У класса `StringValue`, представляющего текстовые значения в рабочем документе Mathcad, всего три свойства: `AsString` (значение в текстовом формате), `Type` (тип значения) и `Value` (значение).

При работе с матрицами и массивами необходимы методы и свойства класса `MatrixValue`, которые перечислены в табл. 4.22.

Таблица 4.22. *Свойства класса* `MatrixValue`

Свойство	Описание
`AsString`	Возвращается значение в виде текстовой строки
`Cols`	Число столбцов матрицы
`Rows`	Число строк матрицы
`Type`	Тип значения

У этого класса есть два метода: `GetElement` — для получения значения элемента матрицы и `SetElement` — для установки значения элемента матрицы. Аргументами метода `GetElement` указывают индексы элемента, а у метода `SetElement` есть еще третий аргумент — присваиваемое элементу значение.

Работа с программным кодом

Рассмотрим программный код сценария, с помощью которого реализуется функциональность вставленной ранее (см. рис. 4.31) в рабочий документ Mathcad кнопки (листинг 4.1).

Листинг 4.1. Программный код сценария для элемента управления `PushBtn`

```
Dim state
Sub PushBtnEvent_Start()
  Rem TODO: Add your code here
End Sub
Sub PushBtnEvent_Exec(Inputs,Outputs)
    Outputs(0).value = state
End Sub
Sub PushBtnEvent_Stop()
  Rem TODO: Add your code here
End Sub
Sub PushBtn_Click()
     If state = 0 Then
         state = 1
     Else
        state = 0
     End If
     PushBtn.Recalculate()
End Sub
```

Листинг состоит из инструкции описания переменной `state` (в самом начале) и четырех процедур. Следует отметить, что среди поддерживаемых методов кнопки есть метод `Recalculate`, с помощью которого осуществляется пере-

счет элемента управления (другими словами, обновляются параметры). Процесс условно состоит из событий начала пересчета (событие `Start`), выполнения пересчета (событие `Exec`) и его окончания (событие `Stop`). Кнопка также может реагировать (обычно так и делает) на щелчок, что соответствует событию `Click`. Упомянутые четыре процедуры есть не что иное, как обработчики соответствующих событий. Названия процедур содержат имя элемента управления и названия события.

Фактический код содержат только процедуры `PushBtnEvent_Exec()` и `PushBtn_Click()` (`Rem` означает комментарий). В процедуре `PushBtnEvent_Exec()` всего одна команда `Outputs(0).value = state`. Этой командой для переменной, которой в качестве значения была присвоена кнопка, устанавливается значение, равное переменной `state`. В данном случае `Outputs` — это массив значений, возвращаемый элементом управления. Поскольку возвращается одно значение, то в массиве всего один элемент с нулевым индексом. Свойство `value` отвечает за значение соответствующего элемента массива.

Код процедуры `PushBtn_Click()` состоит из условного оператора, в котором значение переменной `state` меняется с 0 на 1 и наоборот, и вызова метода пересчета `Recalculate`. Следовательно, схема выполнения сценария такова: при щелчке на кнопке переменная `state` принимает поочередно одно из двух значений (0 или 1), начинается пересчет элемента, для чего вызывается метод `Recalculate`. Как следствие вызова этого метода запускается обработчик события `PushBtnEvent_Exec()`, в рамках выполнения которого определяется результат. Вооружившись этими сведениями, внесем изменения в код сценария.

Пример 4.17. Подсчет количества щелчков на кнопке

В качестве очень простого примера изменения сценария рассмотрим задачу по подсчету количества щелчков на кнопке. Для этого в процедуре `PushBtn_Click()` меняем условный оператор на команду `state = state + 1`, которой при каждом щелчке значение переменной `state` увеличивается на единицу. Процедура `PushBtn_Click()` будет иметь при этом вид, показанный в листинге 4.2.

Листинг 4.2. Программный код процедуры `PushBtn_Click()`

```
Sub PushBtn_Click()
state = state + 1
PushBtn.Recalculate()
End Sub
```

Результат работы элемента управления в этом случае иллюстрируется на рис. 4.34, где показан фрагмент рабочего документа с элементом управления после нескольких (точнее, пяти) щелчков на кнопке.

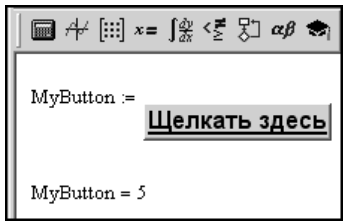


Рис. 4.34. Рабочий документ после нескольких щелчков на кнопке

Если сохранить документ, закрыть, а затем снова открыть, значением переменной `MyButton` будет нуль.

Пример 4.18. Изменение атрибутов кнопки

Достаточно нагляден пример, в котором при щелчке на кнопке изменяются ее внешние атрибуты. Отредактируем код сценария так, чтобы при щелчке на кнопке менялось ее название. Новый код процедуры `PushBtn_Click()` приведен в листинге 4.3.

Листинг 4.3. Измененный код процедуры `PushBtn_Click()`

```
Sub PushBtn_Click()
state = state Mod 3 + 1
With PushBtn
Select Case state
Case 1    Text="Один"
Case 2    Text="Два"
Case Else Text="Три"
End Select
End With
PushBtn.Recalculate()
End Sub
```

Теперь новое значение, присваиваемое переменной `state`, равняется остатку от деления предыдущего на три плюс единица. Поэтому соответствующая переменная принимает при каждом щелчке последовательно значения один, два и три (еще ноль в самом начале при открытии документа). Далее следует

оператор `With…End With`. В теле этого оператора с помощью другого оператора выбора `Slect Case` в зависимости от состояния переменной `state` устанавливается название для кнопки (свойство `PushBtn.Text`). Результат выполнения такого кода показан на рис. 4.35 —4.37.

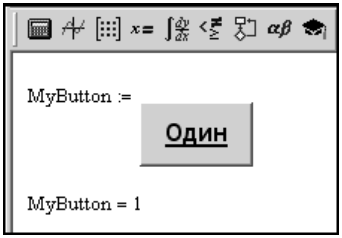


Рис. 4.35. Название кнопки "Один"

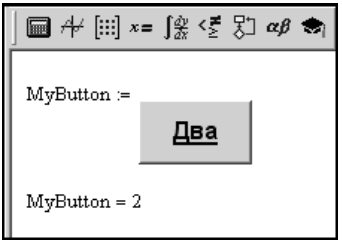


Рис. 4.36. Название кнопки "Два"

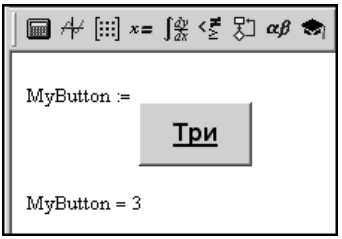


Рис. 4.37. Название кнопки "Три"

Чтобы кнопка при изменении названия не меняла размер, следует отменить опцию автоматического масштабирования (об этом рассказывалось ранее). Можно было бы выполнить такую операцию и программными методами. За автоматическое масштабирование отвечает свойство `AutoSize`. Это и некоторые другие свойства класса `Button` перечислены в табл. 4.23.

Таблица 4.23. *Свойства класса* `Button`

Свойство	Описание
`Auto`	Логическое значение, определяющее, задается ли состояние кнопки автоматически или для этого нужны специальные обработчики событий
`AutoSize`	Установка или снятие режима автоматического масштабирования
`ButtonID`	Применяется с кнопками-переключателями. Это индекс кнопки в группе переключателей, по которому можно ссылаться на соответствующую кнопку
`ButtonStyle`	Тип кнопки. Значение 0 соответствует обычной кнопке, 1 — кнопке-опции и 2 — кнопке-переключателю

Таблица 4.23 *(окончание)*

Свойство	Описание
`Check`	Применяется с кнопками-опциями и кнопками-переключателями. Свойство (логическое значение) определяет, выделена (выбрана) кнопка или нет
`Enabled`	Логическое значение, определяющее доступность элемента
`Flat`	Режим отображения кнопки: плоская или трехмерная
`GroupID`	Применяется с кнопками-переключателями. Идентификатор группы кнопок-переключателей
`LeftText`	Режим отображения текста слева (по отношению, например, к символу опции или переключателя)
`Multiline`	Режим отображения названия кнопки в несколько строк
`PushLike`	Свойство, определяющее вид кнопки как кнопки-опции
`Text`	Название кнопки
`UseBitmap`	Логическое значение, определяющее, используется ли изображение для отображения на кнопке
`UseIcon`	Логическое значение, определяющее, используется ли пиктограмма для отображения на кнопке

Что касается методов класса `Buttons`, то, кроме упоминавшегося уже `Recalculate`, стоит обратить внимание еще на два: `AboutBox` (отображение диалогового окна `About` с информацией об элементе управления) и `SelectedButton` (возвращает индекс выбранного элемента-переключателя в группе). Они иногда бывают полезны. Эти свойства и методы относятся ко всем объектам кнопочного типа: обычным кнопкам, кнопкам-опциям и кнопкам-переключателям. Правда, в зависимости от типа кнопки работа с тем или иным свойством может иметь свои особенности.

Свойства и методы элементов управления

Здесь кратко рассмотрим основные свойства и методы прочих элементов управления, доступных в рабочих документах Mathcad. Начнем с раскрывающихся списков. Списки относятся к классу `ListBox`, свойства которого приведены в табл. 4.24.

Методы класса `ListBox` приведены в табл. 4.25.

Таблица 4.24. *Свойства класса* `ListBox`

Свойство	Описание
`Border`	Отображение границы
`ClientEdge`	Отображение границы области (внутренняя серая рамка)
`Count`	Число элементов в списке
`CurSel`	Индекс выбранного элемента в списке
`DisableNoScroll`	Логическое значение, определяющее, нужно ли отображать неактивную полосу прокрутки, если элементов в списке недостаточно, чтобы заполнить все его поле
`HorizontalScroll`	Логическое значение, определяющее, отображается или нет горизонтальная полоса прокрутки
`LBERR`	Целое число, возвращаемое в случае, если ни один из элементов списка не выбран
`LBERRSPACE`	Используется для проверки наличия достаточной памяти
`ModalFrame`	Отображение рамки списка
`NoIntegralHeight`	Определяет, нужно ли отображать только те элементы списка, что целиком (по высоте) помещаются в видимую область
`RightAlignText`	Выравнивание текста по правому краю
`RightToLeftReadingOrder`	Режим чтения текста справа налево
`Sort`	Отображение элементов списка в алфавитном порядке
`StaticEdge`	Отображение статической границы для списка
`Transparent`	Прозрачность списка: определяется, видны ли объекты, которые попадают за список при его раскрытии
`VerticalScroll`	Логическое значение, определяющее, отображается или нет вертикальная полоса прокрутки

Таблица 4.25. *Методы класса* `ListBox`

Метод	Описание
`AddString`	Добавление строки, указанной аргументом метода, в конец списка
`DeleteString`	Удаление из списка элемента с индексом, указанным аргументом метода

Таблица 4.25 *(окончание)*

Метод	Описание
`FindString`	Определяется индекс первого элемента списка, содержащего указанную строку (как минимум)
`FindStringExact`	Определяется индекс первого элемента списка, совпадающего с указанной строкой
`GetText`	По индексу элемента определяется его текстовое название
`InsertString`	Вставка строки указанной вторым аргументом метода в место списка, соответствующего индексу, указанному первым аргументом
`Recalculate`	Пересчет элемента управления
`ResetContent`	Очистка списка
`SelectString`	Выделение строки по индексу

Много похожих свойств имеют текстовые поля, которые являются членами класса `TextBox`. Свойства этого класса приведены в табл. 4.26.

Таблица 4.26. *Свойства класса* `TextBox`

Свойство	Описание
`AutoHScroll`	Режим автоматической горизонтальной прокрутки
`AutoSize`	Логическое значение, определяющее, активен или нет режим автоматического масштабирования
`AutoVScroll`	Режим автоматической вертикальной прокрутки
`BackColor`	Цвет заднего фона
`Border`	Отображение границы
`ClientEdge`	Внутренняя рамка
`DisableNoScroll`	Отображение неактивной полосы прокрутки в поле с отображением текста в несколько строк в случае, если текст не полностью заполняет поле (по высоте)
`Enabled`	Доступность элемента
`ForeColor`	Цвет переднего фона
`HelpContextID`	Идентификатор контекста файла справки для объекта
`HorizontalScroll`	Отображение горизонтальной полосы прокрутки

Таблица 4.26 *(окончание)*

Свойство	Описание
`hWind`	Возвращается описатель окна, содержащего элемент управления
`LeftScrollBar`	Отображение вертикальной полосы прокрутки слева
`Lowercase`	Режим нижнего регистра
`Multiline`	Режим отображения текста в текстовом поле в несколько строк
`NoHideSelection`	Режим сохранения выделенного фрагмента при передаче фокуса другому элементу управления
`OEMConvert`	Специальный режим работы с символами в кодировке, отличной от ANSI
`Password`	Режим сокрытия вводимых символов в текстовое поле (символы отображаются звездочками)
`ReadOnly`	Режим, в котором текст доступен только для чтения
`RightAlignedText`	Выравнивание текста по правому краю
`RightToLeftReadingOrder`	Чтение текста справа налево
`Text`	Текстовое содержимое текстового поля
`Uppercase`	Режим верхнего регистра
`VerticalScroll`	Отображение вертикальной полосы прокрутки справа
`WantsReturn`	Отображение символа возврата каретки

Методов у текстового поля нет, да это и понятно, в отличие от цифровой полосы прокрутки. Этот элемент управления с точки зрения ООП является экземпляром объекта `Slider`. Метод у объекта `Slider` всего один — `Recalculate`. А вот свойств несколько больше. Они перечислены в табл. 4.27.

Таблица 4.27. *Свойства объекта* `Slider`

Свойство	Описание
`AutoTicks`	Отображение всех меток делений шкалы вдоль полосы прокрутки с шагом, устанавливаемым свойством `TickFrequency`
`TickFrequency`	Шаг между делениями
`Ticks`	Режим отображения меток делений
`Border`	Отображение границы

Таблица 4.27 *(окончание)*

Свойство	Описание
ClientEdge	Внутренняя рамка
StaticEdge	Отображение статической границы
NumTicks	Количество отображаемых меток (только чтение)
Orientation	Ориентация полосы прокрутки (0 означает ориентацию по горизонтали и 1 — по вертикали)
Point	Тип ползунка цифровой полосы прокрутки (возможные значения 0, 1 и 2)
Transparent	Режим прозрачности элемента управления
EnableSelection	Режим, позволяющий пользователю выбирать диапазон значений
LineSize	Интервал перемещения при управлении ползунком с помощью клавиатуры
PageSize	Интервал перемещения при управлении ползунком с помощью клавиш <Page Up>, <Page Down> или колеса мыши
MaxRange	Максимальное значение шкалы
MinRange	Минимальное значение шкалы
Position	Текущее значение
SelectionLow	Нижняя граница выбранного диапазона значений
SelectionHigh	Верхняя граница выбранного диапазона значений

Вне всяких сомнений, разбираться в том, как работают элементы управления и что с их помощью можно делать, лучше на примерах. Далее рассматривается пример, в котором задействованы сразу несколько элементов управления.

Пример 4.19. Совместное использование элементов управления

В качестве примера совместного использования нескольких элементов управления в одном рабочем листе рассмотрим документ, фрагмент которого представлен на рис. 4.38.

Назначение у документа простое: там отображается график одной из трех предопределенных функций. Функция может отображаться с масштабным множителем, который, в свою очередь, может быть экспоненциальной или показательной функцией. В этом множителе фигурирует один параметр, задаваемый пользователем. Фактически, данный параметр определяет степень

множителя масштабирования. Реализуется все это посредством нескольких элементов управления.

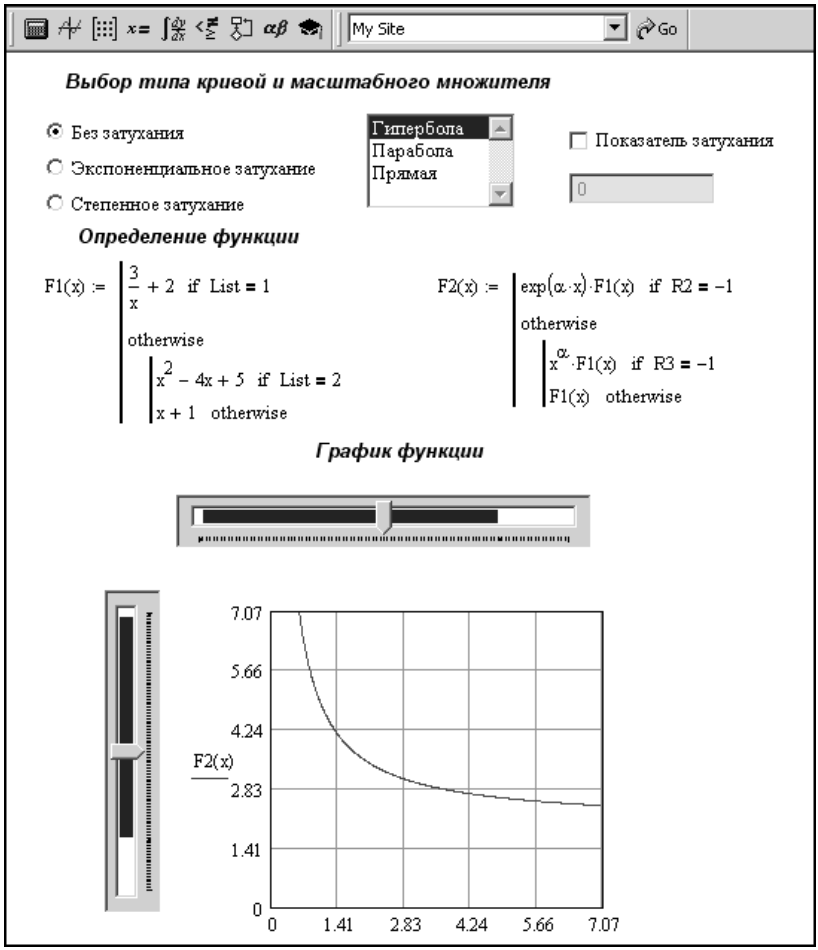


Рис. 4.38. Документ с несколькими элементами управления

В верхней части документа размещена группа переключателей на три положения. Выбрав одно из положений, можно установить экспоненциальный, степенной масштабный множитель или его отсутствие. Список в правой части предназначен для выбора типа кривой. Элементы этого списка — наименования **Гипербола**, **Парабола** и **Прямая**. Еще правее находится кнопка-опция **Показатель затухания** и поле под ней. Если флажок опции установлен, то в поле внизу можно ввести численное значение для показателя затухания. Если флажок не установлен, поле недоступно (при попытке редактирования измене-

ния в силу не вступают) и выделено желтым цветом. По умолчанию показатель затухания равен нулю. Далее следуют два блока, определяющие отображаемую функцию, и непосредственно рисунок с изображением графика выбранной функции с учетом показателя затухания. Вверху и слева от рисунка размещены полосы прокрутки, с помощью которых устанавливаются верхние границы вдоль горизонтальной и вертикальной координатных осей.

На рис. 4.38 все элементы управления отображены без указания переменных, значениями которых они являются. Чтобы было легче разобраться в том, как все это работает, имеет смысл рассмотреть документ, так сказать, в "первозданном" виде. Правда, для этого придется несколько раздвинуть элементы, поэтому рабочий документ увеличится и его содержимое далее показано на двух рисунках. На рис. 4.39 показана верхняя часть документа.

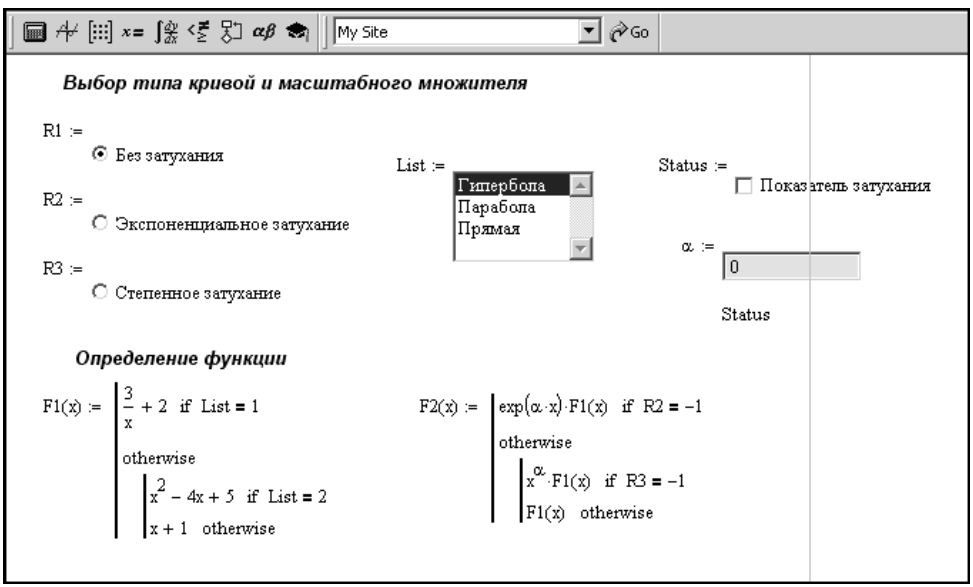


Рис. 4.39. Верхняя часть документа

Нижняя часть документа, содержащая график функции, показана на рис. 4.40.

Важно подчеркнуть, что документы на рис. 4.38, 4.39 и 4.40 функционально абсолютно эквиваленты — это просто один и тот же документ, просто режим отображения элементов различный. Теперь сделаем несколько замечаний относительно механизма взаимодействия элементов управления. Прежде всего, расскажем, как переходят в режим, при котором переменные для элементов управления не отображаются. Делается это просто: достаточно выделить соответствующий элемент, щелкнуть правой кнопкой мыши и в раскрывшемся списке выбрать команду **Hide Arguments** (рис. 4.41).

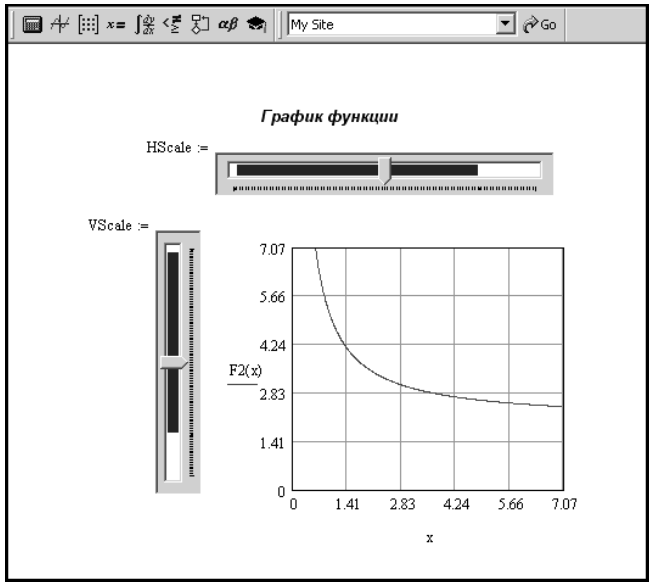


Рис. 4.40. Нижняя часть документа

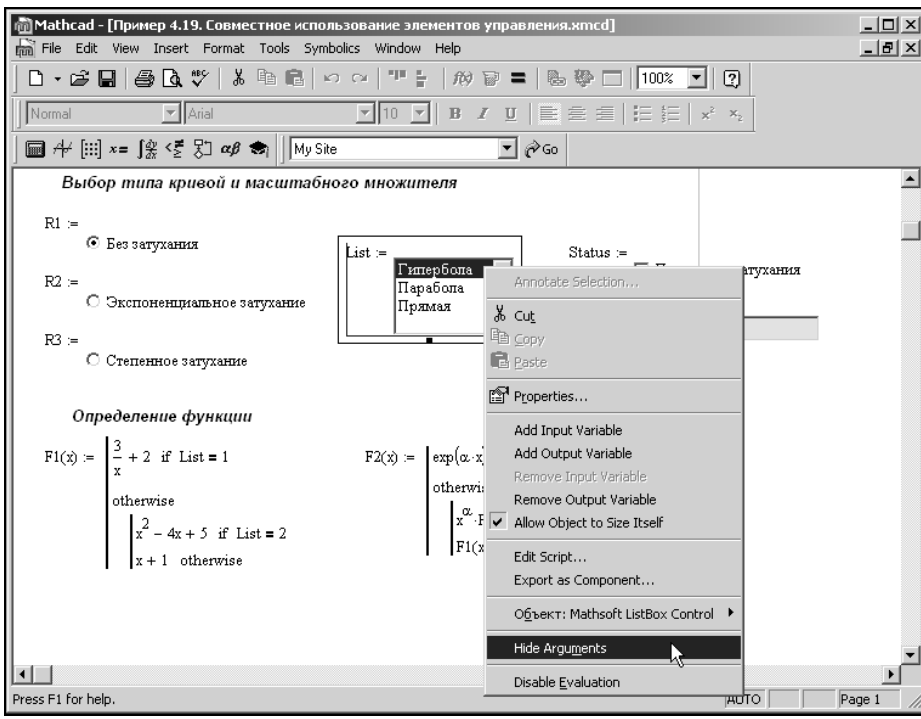


Рис. 4.41. Переход в режим скрытых переменных для элемента управления

Если переменные для элемента управления скрыты, то соответствующая команда в раскрывающемся списке меняется на **Show Arguments**. Ее используют для отображения переменных для элемента управления (рис. 4.42).

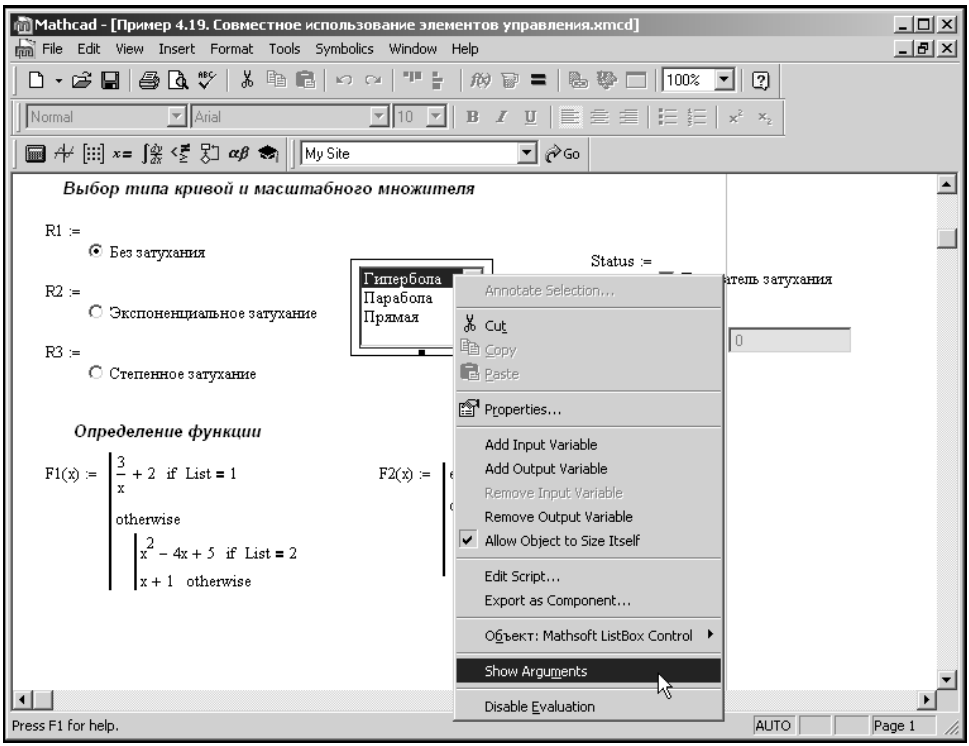


Рис. 4.42. Выход их режима скрытых переменных для элемента управления

Далее разберемся с каждым элементом управления, размещенным в рабочем документе, по отдельности. Начнем с определения отображаемой на рисунке функции.

Функция описывается, что называется, в два захода — исключительно ради удобства. В первом блоке определяется непосредственно отображаемая функция `F1(x)`. Блок состоит из двух вложенных условных операторов. Проверяется значение переменной `List`, которое равно индексу элемента списка, выбранного пользователем. Если выбран первый элемент списка (значение переменной `List` равно `1`), строится график функции $f(x) = 3/x + 2$. Для второго элемента (`List = 2`) отображается график параболы $f(x) = x^2 - 4x + 5$. Наконец, для `List = 3`, строится прямая $f(x) = x + 1$. Во втором блоке описана функция `F2(x)`, которая получается из `F1(x)` умножением на масштабный множитель.

Множителем может быть $\exp(\alpha x)$ или x^{α}, либо множитель вообще отсутствует. Параметр α указывается в текстовом поле, а тип множителя (или его отсутствие) определяется в соответствии с состоянием группы переключателей. Переменные, которые соответствуют различным переключателям (это переменные `R1`, `R2` и `R3`), возвращают в качестве своих значений –1 (это численный эквивалент значения `True`), если переключатель находится в соответствующем положении, и 0 (численный эквивалент значения `False`) в противном случае. Блок также состоит из вложенных условных операторов. Именно функция `F2(x)` введена вместо структурного заполнителя в левой части рисунка с графиком (см. рис. 4.40).

Назначение полос прокрутки, как отмечалось, состоит в выборе верхних границ отображения графика функции. Переменные, введенные для полос прокрутки, имеют названия `HScale` и `VScale` для горизонтальной и вертикальной полос, соответственно (см. рис. 4.40). Если быть более точным, то в качестве верхних границ указаны квадратные корни от этих переменных (рис. 4.43).

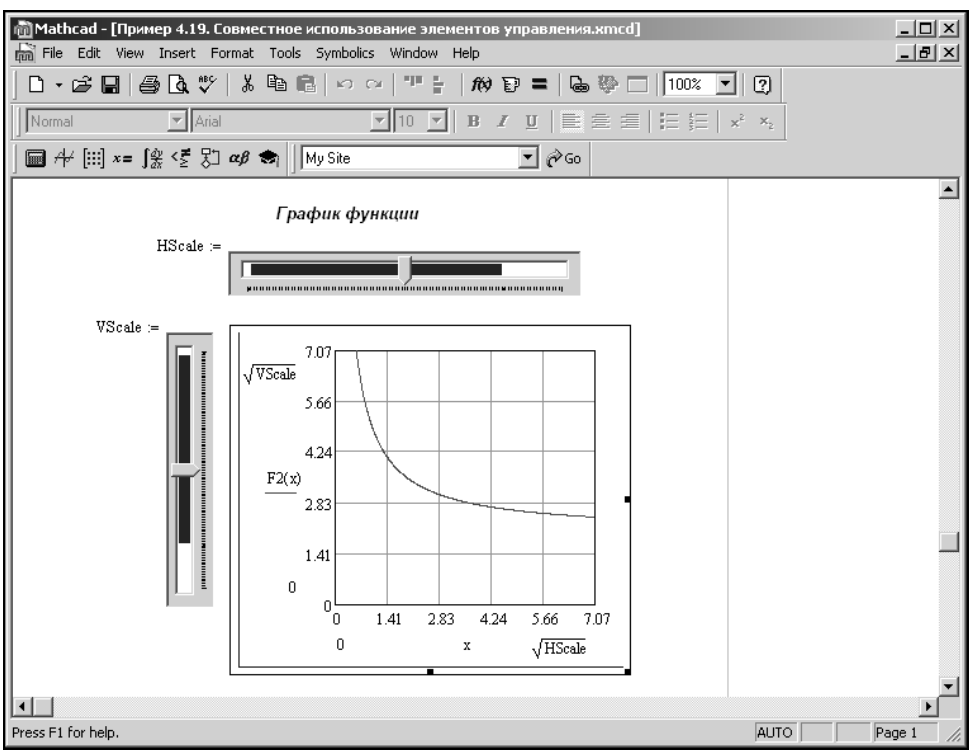


Рис. 4.43. Верхние границы диапазонов отображения графика функции

Сделано это для того, чтобы при перемещении ползунка (увеличении верхней границы) эффективная цена деления возрастала. Что касается настроек полос прокрутки, то это будет легче понять, если взглянуть на рис. 4.44, на котором показано диалоговое окно свойств, где собственно настройки и выполняются.

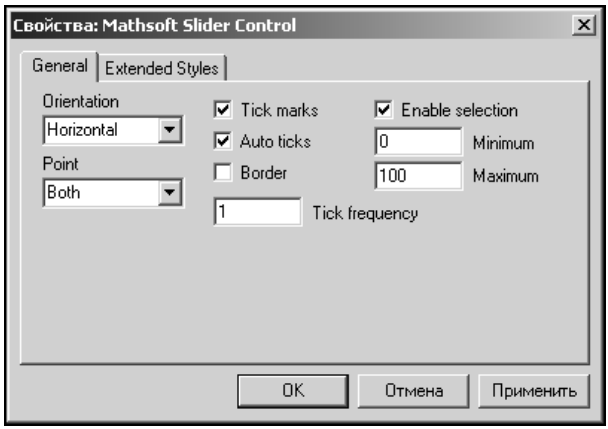


Рис. 4.44. Диалоговое окно настройки полосы прокрутки открыто на вкладке **General**

Отметим наиболее важные элементы. В списке **Orientation** выбирают способ ориентирования полосы прокрутки (в данном случае горизонтальная полоса). В поле **Minimum** указывают минимальное значение, соответствующее крайнему левому положению ползунка, а в поле **Maximum** — максимальное, которое установлено равным 100. Однако поскольку границей является корень квадратный от этого значения, то верхняя граница на графике может лежать в пределах от 0 до 10. Цена деления шкалы вводится в поле **Tick frequency**. На рис. 4.44 установлен флажок опции **Enable selection**. Это позволяет отображать вдоль полосы прокрутки специальную полосу выделения диапазонов. Об том речь пойдет далее при обсуждении программного кода. Окно свойств имеет еще одну вкладку — **Extended Styles** (рис. 4.45).

Назначение соответствующих опций описывать особого смысла не имеет, поскольку они достаточно точно совпадают с названиями свойств объекта `Slider`, который рассматривался ранее. Ряд настроек выполняется с помощью программного кода. Код для горизонтальной полосы прокрутки приведен в листинге 4.4.

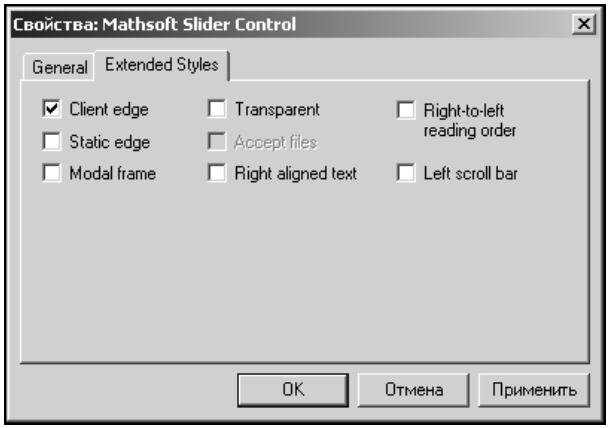


Рис. 4.45. Диалоговое окно настройки полосы прокрутки открыто на вкладке **Extended Styles**

Листинг 4.4. Код для горизонтальной полосы прокрутки

```
With Slider
Position = (MaxRange + MinRange)/2
SelectionLow = MinRange + 2*Sqr(MinRange) + 1
SelectionHigh = MaxRange - 2*Sqr(MaxRange) + 1
End With
Sub SliderEvent_Exec(Inputs,Outputs)
  Outputs(0).Value = Slider.Position
End Sub
Sub Slider_ValueChanged()
  Slider.Recalculate()
End Sub
```

Код для вертикальной полосы прокрутки аналогичен. В этом коде оставлены только те процедуры, в которых выполняются вычисления и настройки — лишние удалены. Блок команд в начале модуля предназначен для исходных установок при открытии документа. Этими командами ползунок устанавливается в центральное положение (команда `Position = (MaxRange + MinRange)/2`), а также задается упоминавшаяся полоса выделения диапазона. Она настраивается так, что начальный белый отрезок соответствует единичному значению вдоль координатной оси на графике, а конечный — также единичному значению вдоль оси, но уже с другой стороны координатной оси. Другими словами, если переместить ползунок на левую границу полосы

выделения, график будет отображаться в диапазоне от 0 до 1. Если переместить ползунок к правой границе полосы выделения, график будет отображаться в диапазоне от 0 до 9. Для того чтобы кнопка-переключатель в качестве значения своей переменной возвращала состояние переключателя, изменяем программный код этого элемента так, как показано в листинге 4.5.

Листинг 4.5. Код для первой кнопки-переключателя

```
RadioBtn.Check = True
Sub RadioBtnEvent_Exec(Inputs,Outputs)
    Outputs(0).Value = RadioBtn.Check()
End Sub
Sub RadioBtn_Click()
    RadioBtn.Recalculate()
End Sub
Sub RadioBtn_SelectionChange()
    RadioBtn.Recalculate()
End Sub
```

Поскольку первая кнопка-переключатель выделена по умолчанию при загрузке документа, в верхней части модуля для первой кнопки указана команда `RadioBtn.Check = True`. Для прочих кнопок эту команду не указывают. Возвращаемый результат вычисляется командой `Outputs(0).Value = RadioBtn.Check()` (по умолчанию предлагается команда `Outputs(0).Value = RadioBtn.SelectedButton()`).

Программный код для списка приведен в листинге 4.6.

Листинг 4.6. Программный код для списка

```
ListBox.ResetContent()
ListBox.AddString("Гипербола")
ListBox.AddString("Парабола")
ListBox.AddString("Прямая")
ListBox.CurSel = 0
Sub ListBoxEvent_Exec(Inputs,Outputs)
    Outputs(0).Value = ListBox.CurSel + 1
End Sub
Sub ListBox_SelChanged()
    ListBox.Recalculate()
```

```
End Sub
Sub ListBox_DblClick()
    ListBox.Recalculate()
End Sub
```

Сначала очищают список (команда `ListBox.ResetContent()` — в данном случае она необязательна, все будет работать и без нее). Далее добавляются элементы списка, и выделяется первый элемент (команда `ListBox.CurSel = 0`). В качестве результата возвращается порядковый номер выбранного элемента, который больше индекса этого элемента на единицу. При загрузке документа флажок кнопки-опции отменен, поэтому соответствующий код начинается с команды `CheckBox.Check = False` (листинг 4.7).

Листинг 4.7. Программный код для кнопки-опции

```
CheckBox.Check = False
Sub CheckBoxEvent_Exec(Inputs,Outputs)
     If CheckBox.check Then
         Outputs(0).Value = 1
     Else
         Outputs(0).Value = 0
     End If
End Sub
Sub CheckBox_Click()
     CheckBox.Recalculate()
Rem Worksheet.Recalculate()
End Sub
```

В том случае, если установлен флажок опции, в качестве значения соответствующей переменной `Status` возвращается `1`, а в противном случае — `0`. Кроме этого, стоит обратить внимание на команду пересчета рабочего документа `Worksheet.Recalculate()` — она выделена как комментарий. Дело в том, что переменная `Status` существенно влияет на функциональность рассматриваемого далее текстового поля. Чтобы при ее модификации изменения вносились и в настройки этого поля, можно, например, в явном виде задать команду пересчета рабочего листа или указать переменную `Status` в качестве входной переменной текстового поля (далее использован именно такой подход — входная переменная добавляется с помощью команды **Add Input Variable** из раскрывающегося списка). Программный код для текстового поля приведен в листинге 4.8.

Листинг 4.8. Программный код для текстового поля

```
TextBox.Text = 0
TextBox.BackColor = vbYellow
TextBox.Enabled = Worksheet.GetValue("Status")
Sub MyColor(enable)
If enable Then
          TextBox.BackColor = vbWhite
          Else
          TextBox.BackColor = vbYellow
          TextBox.Text = "0"
End If
End Sub
Sub TextBoxEvent_Exec(Inputs,Outputs)
Dim a
a = WorkSheet.GetValue("Status")
TextBox.Enabled = a
MyColor(a)
    If a Then Outputs(0).Value = CDbl(TextBox.Text)
End Sub
```

В модуле, помимо прочего, описывается процедура `MyColor()`, с помощью которой по аргументу устанавливается фон текстового поля. Для желтого фона (он соответствует ситуации, когда элемент недоступен) в поле программными методами вводится также и нулевое значение. В качестве результата текстовым полем возвращается число. Оно получается преобразованием введенного в текстовое поле значения к числовому формату (функция `CDbl()`). Доступ к переменной `Status` в программном коде реализуется методом `Worksheet.GetValue()`.

Эта глава является особой, поскольку в ней, по большому счету, речь шла не столько о Mathcad, сколько о языке сценариев VBScript. С одной стороны, современные тенденции развития прикладных математических пакетов таковы, что их все более тесная интеграция с другими программными средствами, по-видимому, будет со временем усиливаться. Это накладывает свой отпечаток на особенности работы с такими пакетами. А с другой стороны, если разработка проекта подразумевает привлечение серьезных внешних программных ресурсов, естественным образом возникает вопрос, правильно ли выбрано базовое приложение, в котором этот проект реализуется? Поэтому, несмотря на всю наглядность и эффективность, с написанием разного рода сценариев для приложений типа Mathcad следует быть крайне осторожным. Это как раз та ситуация, когда средства должны точно соотноситься с конечной целью.

Глава 5

Символьные вычисления

Составить серьезную конкуренцию на рынке математических программных продуктов приложение может только в том случае, если позволяет пользователю выполнять символьные (аналитические) операции. Во всяком случае, это относится к таким пакетам, как Mathematica, Maple и, разумеется, Mathcad. Причем из этой замечательной тройки в области символьных вычислений Mathcad, пожалуй, обладает самыми скромными возможностями. Более того, в Mathcad реализована та же технология выполнения символьных преобразований, что и в Maple, но только упрощенный ее вариант. Фактически, в Mathcad оставлены лишь основные процедуры. Тем не менее, за счет простоты их реализации в рабочих документах и наглядности получаемого результата Mathcad обеспечивает себе неизменно высокую популярность среди специалистов.

Реализация символьных вычислений в Mathcad может осуществляться выбором команд из меню **Symbolics** либо их вводом непосредственно в рабочий документ. В начале этой главы описываются основные символьные операции, которые можно выполнять в рабочих документах Mathcad. Сначала дается описание соответствующих команд меню **Symbolics** в контексте их использования для решения поставленной задачи, после чего рассказывается, как те же операции выполняются с помощью команд в рабочем документе. Эта глава является обзорной, и в ней приведены достаточно простые примеры. Более сложные задачи рассматриваются в *главе 6*, где основное внимание уделяется математической стороне вопроса.

Методы дифференцирования и интегрирования

В рабочих документах Mathcad можно аналитически вычислять производные, интегрировать, раскладывать функции в ряд Тейлора, выполнять преобразования Лапласа и Фурье, упрощать выражения и выполнять ряд других действий. Для этого соответствующее выражение предварительно вводится в рабочий документ, затем оно (или его часть) выделяется и из меню выбирается нужная команда. В зависимости от выполняемых действий процедура может иметь свои особенности, которые подробно рассмотрим далее.

Вычисление производных

Производной, как известно, называется предел отношения приращения функции к приращению ее аргумента. Математически производная функции $f(x)$ в точке x определяется как $f'(x) = \lim\limits_{\Delta x \to 0} (f(x + \Delta x) - f(x))/\Delta x$.

В Mathcad для вычисления производной предусмотрена команда **Symbolics | Variable | Differentiate**. Для того чтобы данная команда была доступна, в рабочем документе в дифференцируемом выражении предварительно следует выделить переменную, по которой производная должна быть вычислена. Можно просто навести на эту переменную маркер ввода (для чего, например, достаточно на этой переменной щелкнуть кнопкой мыши).

Пример 5.1. Производная от выражения

На рис. 5.1 представлено выражение $ax^2 + \sin(bx)$, которое будет дифференцироваться по переменной x. Это выражение было предварительно введено в рабочий документ, и в нем маркер ввода наведен на переменную x в аргументе синуса.

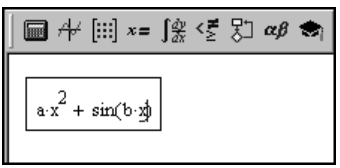


Рис. 5.1. Дифференцируемое выражение

После того как переменная в выражении выделена, выбираем команду **Symbolics | Variable | Differentiate** (рис. 5.2).

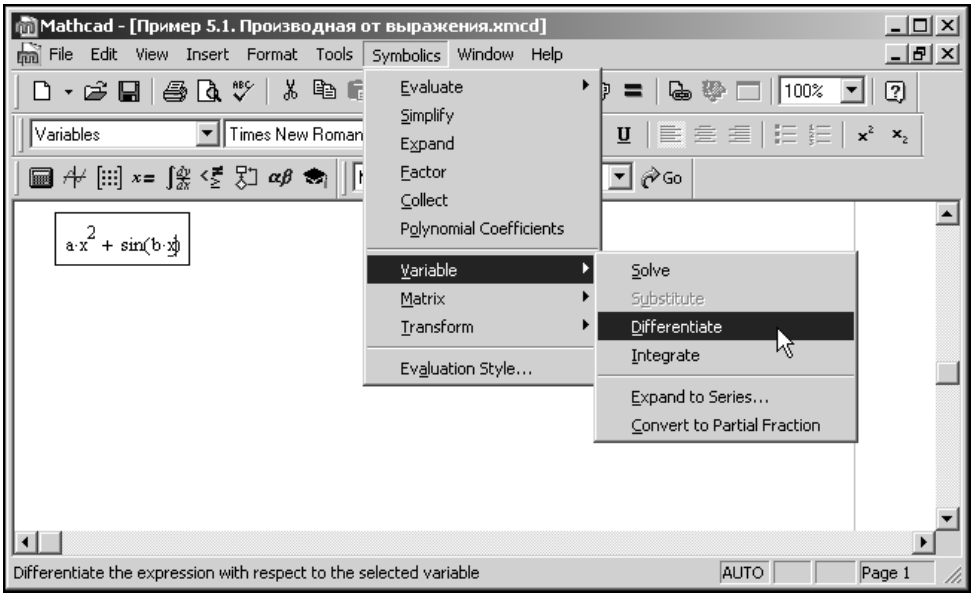


Рис. 5.2. Выбор команды для вычисления производной

Результат выполнения команды по умолчанию отображается под тем выражением, от которого вычисляется производная (рис. 5.3).

С полученным выражением можно работать так же, как и с обычным, введенным пользователем в рабочий лист. Например, от этого выражения можно вычислить еще одну производную. Для разнообразия пускай это будет производная по b. Выделяем эту переменную (рис. 5.4) и в очередной раз выполняем команду **Symbolics | Variable | Differentiate**.

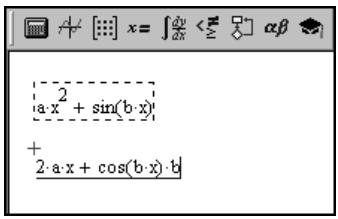


Рис. 5.3. Результат вычисления производной по x

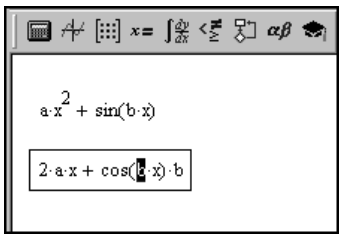


Рис. 5.4. Выделение переменной в выражении

В результате будет вычислена новая производная, которая отображается внизу под исходным выражением (это третье сверху выражение на рис. 5.5).

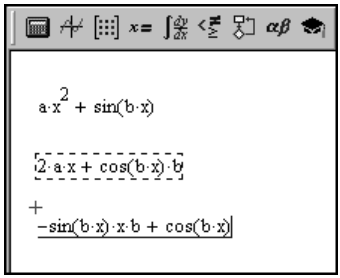


Рис. 5.5. Результат вычисления производной по *b*

Таким образом, в документе можно вычислять производные практически от любых выражений. Хотя такой метод вычисления производных достаточно прост, он связан с рядом неудобств, которые становятся очевидными при решении более сложных задач. Прежде всего, этот метод не пригоден для вычисления производной от функции, определенной пользователем в рабочем листе. При попытке вычислить производную с помощью команды **Differentiate** результат будет представлен как формальная запись производной от функции. Чтобы получить значение производной в явном виде, придется прибегнуть к другим методам (в частности, использовать оператор вычисления символьного значения). Еще один обширный класс задач связан с вычислением производных высоких порядков. В этом случае придется несколько раз вызывать команду **Differentiate**, что не всегда приемлемо. Поэтому на практике для вычисления производных часто обращаются не к помощи меню, а вводят команды непосредственно в рабочий лист. При этом очень удобны математические палитры **Calculus** и **Evaluation**. Далее в качестве примера рассматривается процедура вычисления производной от выражения. При этом вместо команд меню **Symbolics** задействованы упомянутые математические палитры.

Пример 5.2. Символьная производная

Вычислим производную от выражения, не прибегая к командам меню **Symbolics**. Для этого на палитре **Calculus** выберем кнопку с изображением символа производной (рис. 5.6).

В результате этот символ с двумя заполнителями появляется в рабочем документе (рис. 5.7).

На месте нижнего заполнителя вводится переменная, по которой вычисляется производная, а на месте второго — дифференцируемое выражение (рис. 5.8).

Если не прибегать к помощи команд меню **Symbolics**, то после ввода дифференцируемого выражения следует ввести оператор вычисления символьного значения (стрелка вправо). Это можно сделать, щелкнув на соответствующей кнопке палитры **Evaluation** (рис. 5.9).

Рис. 5.6. Вставка символа производной

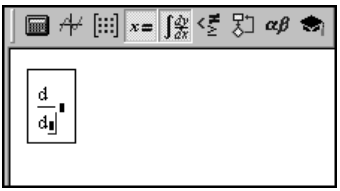


Рис. 5.7. Символ производной

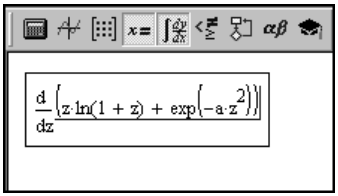


Рис. 5.8. Выражение для вычисления производной

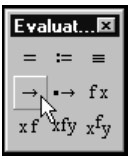


Рис. 5.9. Ввод оператора вычисления символьного значения

После ввода оператора вычисления символьного значения рабочий документ будет иметь вид, как на рис. 5.10.

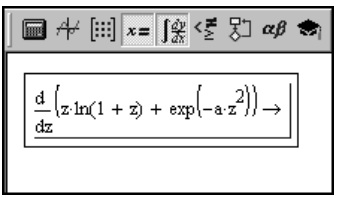


Рис. 5.10. Ввод оператора вычисления символьного значения

Производная будет вычислена, если щелкнуть курсором мыши вне области вводимого выражения или нажать клавишу <Enter>. Результат вычисления производной представлен на рис. 5.11.

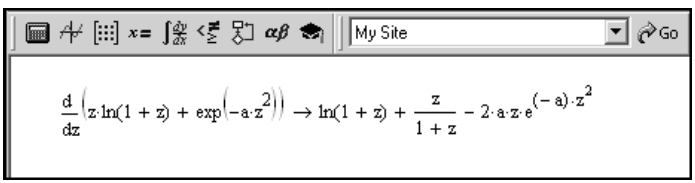


Рис. 5.11. Результат вычисления производной

Несложно проверить, что точно такой же результат может быть получен, если воспользоваться командой **Symbolics | Evaluate | Symbolically** или нажать комбинацию клавиш <Shift>+<F9> (рис. 5.12).

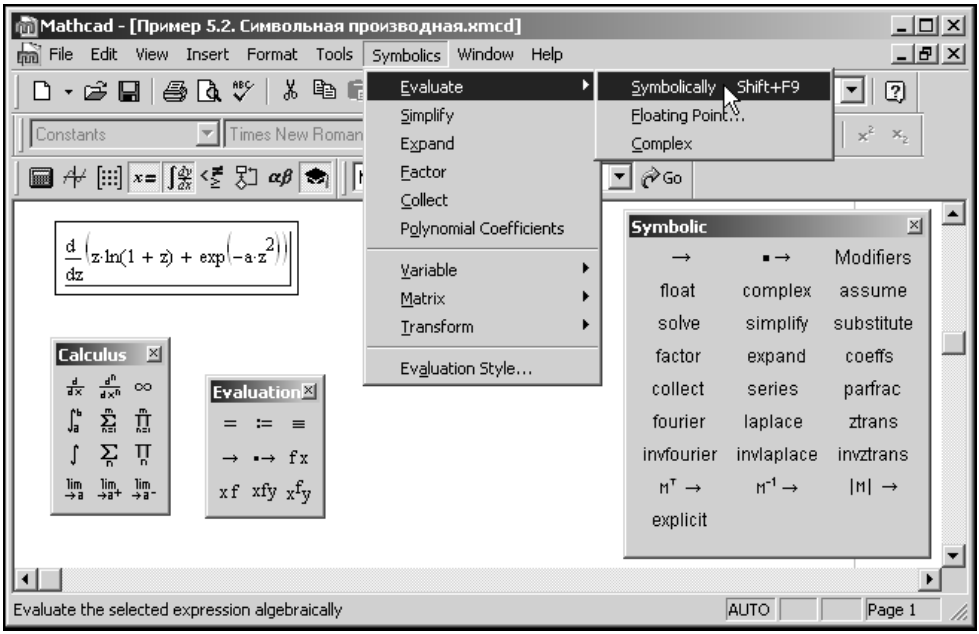


Рис. 5.12. Использование команд меню **Symbolics** для вычисления производной

Правда, в этом случае результат отображается не справа от вычисляемого выражения, а под ним (рис. 5.13). Кроме того, между вычислением символьного результата с помощью команд меню и ключевых слов (т. е. инструкций, вводимых непосредственно в вычисляемые или преобразуемые символьные выражения посредством палитры **Evaluation** или **Symbolic**) существует одна принципиальная разница: если выражение вычислялось через команды меню и затем в документ были внесены изменения, то результат таких вычислений не пересчитывается. При использовании ключевых слов результат обновляется автоматически.

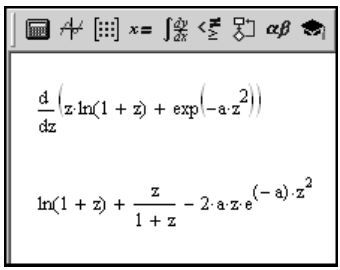


Рис. 5.13. Результат вычисления производной

Пожалуй, одним из главных преимуществ вычисления производных с помощью символьных операторов, которые вводятся непосредственно в рабочем документе, является то, что при этом допускаются функции пользователя. Часто бывает необходимо вычислить производную от функции, определенной ранее в рабочем документе. В качестве иллюстрации рассмотрим следующий пример.

Пример 5.3. Производная от функции

Прежде всего, в рабочем листе определяем функцию пользователя $f(z)$. В качестве такой функции рассмотрим зависимость $f(z) = z\ln(z) + \exp(-az^2)$. В документе функция пользователя определяется следующим образом: сначала вводится название функции, а в скобках после него указывается аргумент (или аргументы). После этого вводится оператор присваивания (двоеточие и знак равенства, причем для ввода этого оператора достаточно нажать комбинацию клавиш <Shift>+<:> — знак равенства после двоеточия отображается автоматически). Вслед за оператором присваивания вводится математическое выражение, определяющее функциональную зависимость (рис. 5.14).

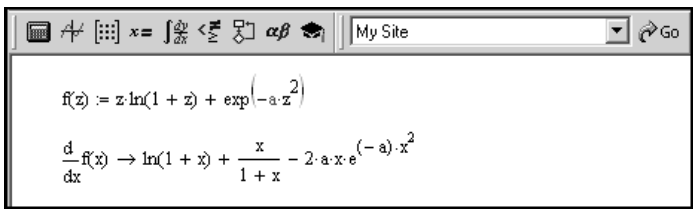


Рис. 5.14. Определение функции пользователя и вычисление производной

Для вычисления производной функция пользователя указывается сразу после символа дифференцирования. Если затем ввести оператор вычисления символьного значения и нажать клавишу <Enter>, производная от функции будет получена (см. рис. 5.14). Стоит обратить внимание, что при вычислении производной, как и в других аналогичных случаях, можно указывать в качестве аргумента переменную, отличную от заданной при определении функции. В рассмотренном примере функция содержит символьный параметр `a`. В этом случае, если предварительно, до определения функции, параметру значение присвоено не было, соответствующий символ в определении функции будет выделен специальным образом (по умолчанию красным цветом). В подобной ситуации необходимо быть предельно аккуратным при выборе переменной для аргумента функции. Например, если в описанной ранее функции указать аргументом переменную `a`, а затем по этой переменной вычислить производную,

то в исходной функциональной зависимости параметр a рассматривается как переменная со всеми вытекающими отсюда последствиями (рис. 5.15).

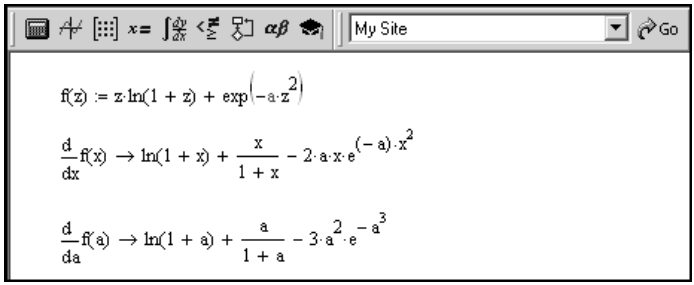


Рис. 5.15. Аргумент и параметр функции совпадают

Кроме этого производную от функции можно вычислять по параметру, даже если он не указан явно как аргумент функции. Ситуация проиллюстрирована на рис. 5.16 (последнее выражение). Там в качестве переменной, по которой вычисляется производная, указан параметр функции a.

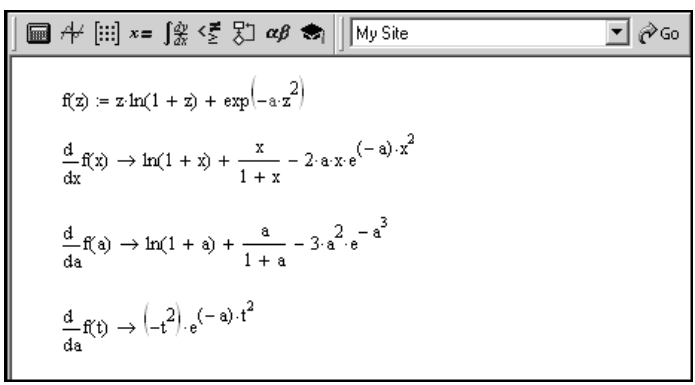


Рис. 5.16. Вычисление производной по параметру функции

Пример 5.4. Производные высоких порядков

Примерно так же вычисляются и производные более высоких порядков. Разумеется, можно несколько раз подряд продифференцировать выражение или функцию с помощью рассмотренных методов. Однако такой подход неприемлем, когда необходимо вычислить производную достаточно высокого порядка. Для этого на панели **Calculus** выбирается пиктограмма с изображением символа производной *n*-го порядка (рис. 5.17).

В результате в рабочем документе появится символ вычисления производной высокого порядка. В отличие от производной первого порядка, здесь необходимо в явном виде указать порядок производной. Поэтому заполнителей не два, как раньше, а четыре (рис. 5.18).

Рис. 5.17. Выбор символа производной произвольного порядка

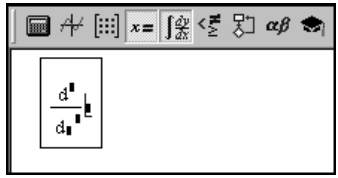


Рис. 5.18. Ввод символа вычисления производной высокого порядка в документ

Причем из двух заполнителей, соответствующих порядку производной (он, как известно, указывается у обоих дифференциалов в числителе и знаменателе выражения), заполняется только нижний. Верхний заполнитель автоматически принимает такое же значение (рис. 5.19).

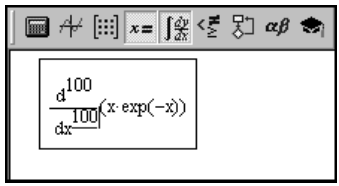


Рис. 5.19. Ввод дифференцируемого выражения и порядка производной

В данном случае вычисляется производная 100-го порядка от выражения $x\exp(-x)$ по переменной x. Именно это выражение вводится в качестве дифференцируемого (см. рис. 5.19). Во всем остальном процесс расчета производной высокого порядка в символьном виде такой же, как и для обычной производной первого порядка: вводится оператор вычисления символьного значения, после чего выполняется щелчок мышью вне области выражения (рис. 5.20).

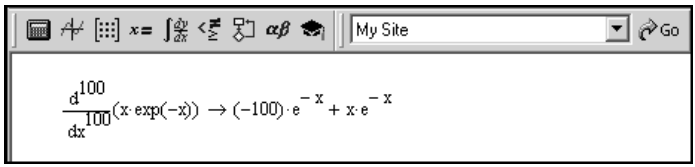


Рис. 5.20. Результат вычисления сотой производной

Как несложно заметить, здесь не только вычислена производная, но и результат представлен в достаточно компактном виде. Хотя, справедливости ради, следует отметить, что дифференцировалось далеко не самое сложное выражение. Совершенно аналогично вычисляются производные высоких порядков от функций пользователя. Пример приведен в документе на рис. 5.21.

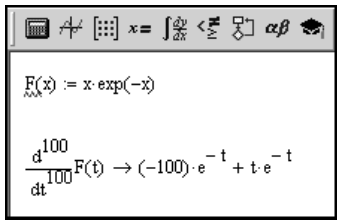


Рис. 5.21. Вычисление производной высокого порядка от функции пользователя

В данном случае сначала была определена функция $F(x) = x\exp(-x)$. От этой функции вычисляется 100-я производная. При вычислении производной в аргументе функции заменена переменная. Результат, как несложно заметить, такой же, как и при вычислении производной непосредственно от выражения.

Интегрирование

Практически так же, как производные, вычисляются и интегралы. Для интегрирования выражения в рабочем документе существует команда **Symbolics | Variable | Integrate**.

Пример 5.5. Вычисление интегралов

Проиллюстрируем ситуацию на примере вычисления интеграла от выражения $\exp(ax)\cos(bx)$ по переменной x. Для этого соответствующее выражение следует предварительно ввести в рабочий документ, а переменную, по которой вычисляется интеграл, выделить (рис. 5.22).

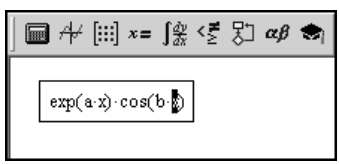


Рис. 5.22. Ввод в рабочий документ интегрируемого выражения

После того как переменная выделена, становится активной команда **Integrate**. Поэтому чтобы вычислить интеграл, выбираем команду **Symbolics | Variable | Integrate** (рис. 5.23).

Результат интегрирования отображается внизу под интегрируемым выражением (рис. 5.24).

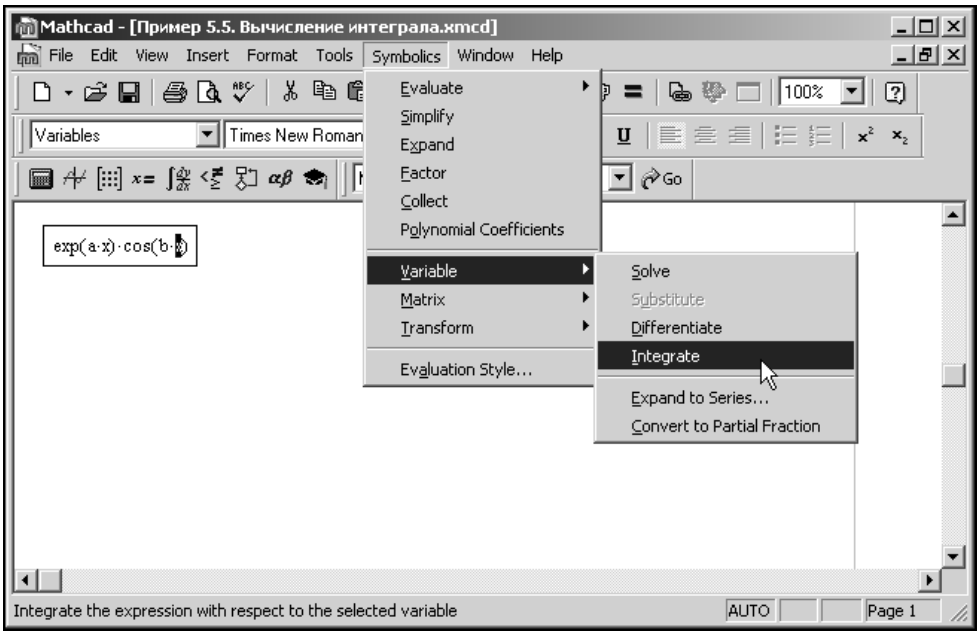


Рис. 5.23. Выбор команды интегрирования выражения

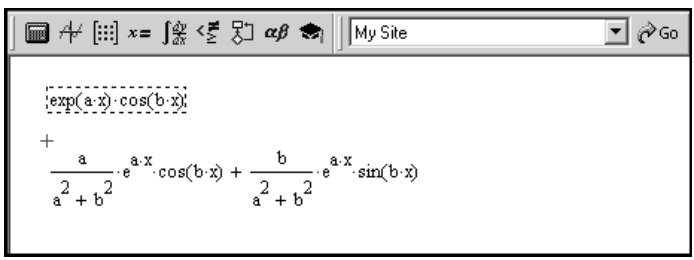


Рис. 5.24. Результат вычисления интеграла

Вообще же следует отметить, что ситуация с вычислением интегралов практически такая же, как и с дифференцированием функций и выражений. Например, если определить функцию пользователя, а затем попытаться вычислить от нее интеграл, то результат будет негативным — интеграл отображается

в символьном виде без вычисления. Поэтому для вычисления интегралов разумнее вводить соответствующие команды прямо в рабочий документ. Для этого используют палитру **Calculus**, где имеются две пиктограммы, полезные при вычислении интегралов. Одна применяется при вычислении неопределенных интегралов, другая — определенных. Это пиктограммы с изображением символа интеграла и отличаются они тем, что пиктограмма для определенного интеграла содержит символы верхней и нижней границ. На рис. 5.25 показана панель **Calculus** с выделенной пиктограммой для вычисления неопределенного интеграла.

После щелчка мышью на этой пиктограмме в рабочем документе появляется символ неопределенного интеграла с двумя заполнителями для ввода интегрируемого выражения и переменной, по которой вычисляется интеграл (рис. 5.26).

Рис. 5.25. Панель **Calculus** с выделенной пиктограммой вычисления неопределенного интеграла

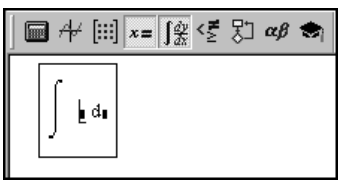


Рис. 5.26. Символ интеграла с заполнителями

Интегрировать будем то же самое выражение, что и ранее. Его вводим вместо первого заполнителя, а вместо второго — переменную интегрирования. После этого вводится оператор вычисления символьного значения. На этом этапе выражение будет иметь вид, как показано на рис. 5.27.

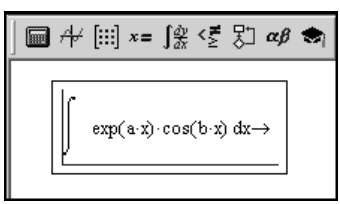


Рис. 5.27. Ввод выражения и оператора вычисления символьного значения

После щелчка мышью вне области выражения (или нажатия кнопки <Enter>) интеграл будет вычислен (рис. 5.28).

Заметим, что результат вычисления интеграла представлен не в самом компактном виде. В подобных ситуациях обычно прибегают к командам упрощения выражений. Подробнее об этом речь будет идти далее.

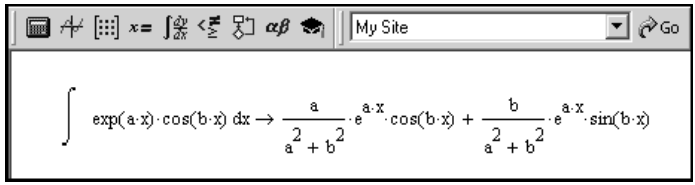


Рис. 5.28. Результат вычисления интеграла

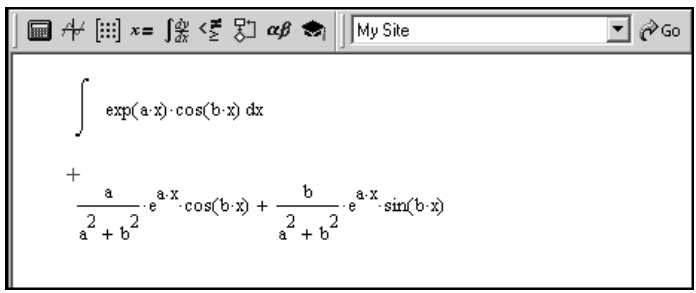


Рис. 5.29. Отображение результата вычисления интеграла после вызова команды **Evaluate | Symbolically**

Если не вводить оператор вычисления символьного значения, а выделить выражение с интегралом и вызвать команду **Symbolics | Evaluate | Symbolically**, интеграл также будет получен, однако результат при этом отображается внизу под исходным выражением (рис. 5.29).

При вычислении определенного интеграла прибегают к помощи другой пиктограммы (рис. 5.30).

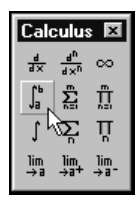


Рис. 5.30. Выбор пиктограммы для вычисления определенного интеграла

В случае определенного интеграла помимо интегрируемого выражения и переменной интегрирования указываются также пределы интегрирования. Соответствующие заполнители отображаются в верхней и нижней частях интеграла (рис. 5.31).

Во всем остальном процедура вычисления определенного интеграла мало чем отличается от предыдущей. Результат вычисления определенного интеграла представлен на рис. 5.32.

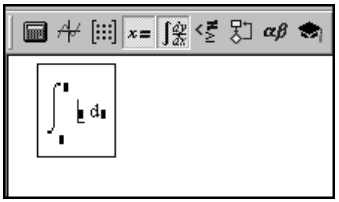


Рис. 5.31. Ввод данных для вычисления определенного интеграла

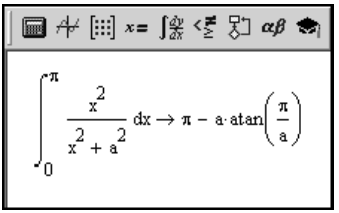


Рис. 5.32. Результат вычисления определенного интеграла

В данном случае интегрировалось выражение $x^2/(x^2+a^2)$ по переменной x в пределах от 0 до π. Для ввода числа π в качестве верхней границы интеграла можно воспользоваться палитрой **Calculator**. Хотя интеграл определенный, вычисляется он в аналитическом виде. Поэтому при вычислении определенных интегралов можно прибегнуть к помощи команды **Symbolics | Evaluate | Symbolically**. В этом случае оператор вычисления символьного значения не нужен.

Как уже отмечалось, обычно приходится иметь дело с функциями пользователя. Вычисление определенных и неопределенных интегралов при этом также возможно. В обоих случаях функция указывается в качестве подынтегрального выражения. На рис. 5.33 показан результат вычисления интеграла Пуассона (с точностью до общего множителя).

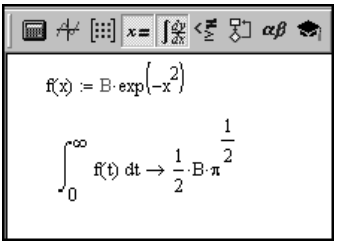


Рис. 5.33. Результат вычисления интеграла Пуассона

Подынтегральная функция — $f(x) = B\exp(-x^2)$. Интегрирование выполняется в пределах от нуля до бесконечности. Символ бесконечности (∞) можно ввести посредством палитры **Calculus**.

Использование команд Maple

В версии Mathcad 11 помимо перечисленных возможностей по дифференцированию и интегрированию функций и выражений при работе с документами приложения Mathcad доступны и команды Maple. Причина такого либера-

лизма достаточно очевидна и состоит в том, что символьные вычисления в Mathcad реализуются, как уже отмечалось, посредством ядра Maple. В частности, для вычисления производных предназначена функция `diff()`. Ее первым аргументом указывается дифференцируемая функция или выражение, вторым — переменная, по которой производная вычисляется. Например, для вычисления производной от $\frac{d}{dx}\left(ax^2+bx+c\right)$, в рабочий документ Mathcad 11 достаточно ввести команду `diff(a*x*x+b*x+c,x)`, после которой указать оператор вычисления символьного значения. Результат ее выполнения представлен на рис. 5.34.

Стоит обратить внимание на то, что вторая степень вводится путем последовательного умножения переменной. Однако возможен и обычный для Mathcad синтаксис представления выражений. Часто бывает удобным сначала определить дифференцируемое выражение в виде функции, после чего дифференцировать уже эту функцию (рис. 5.35).

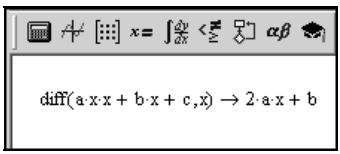


Рис. 5.34. Использование команды Maple для вычисления производной

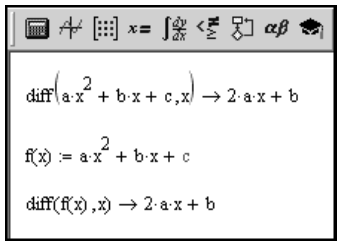


Рис. 5.35. Использование синтаксиса Mathcad в командах Maple

Это же относится и к вычислению интегралов. В Maple для этих целей предусмотрена команда `int()`. Ее аргументы — это интегрируемое выражение и переменная, по которой следует интегрировать. Замечание о предварительном определении подынтегральной функции также остается справедливым. В качестве примера рассмотрим уже упоминавшийся ранее интеграл $\int \exp(ax)\cos(bx)dx$. Для его вычисления сначала определяем функцию $f(x)=\exp(ax)\cos(bx)$ (команда `f(x):=exp(a*x)*cos(b*x)`), после чего вводим в рабочий документ Mathcad 11 команду `int(f(x),x)`. Далее следует ввести оператор вычисления символьного значения. Результат можно видеть на рис. 5.36.

Как несложно заметить, полученное в результате выражение несколько громоздко. Это достаточно распространенная ситуация при вычислении интегралов, особенно в символьном виде. Поэтому подобного рода выражения приходится упрощать. В частности, в Maple для упрощения выражений

существует команда `simplify()`, аргументом которой указывается упрощаемое выражение. Эту же команду можно задействовать и в рабочем документе Mathcad 11. В этом случае команда интегрирования сама указывается аргументом команды упрощения выражений `simplify()` (рис. 5.37).

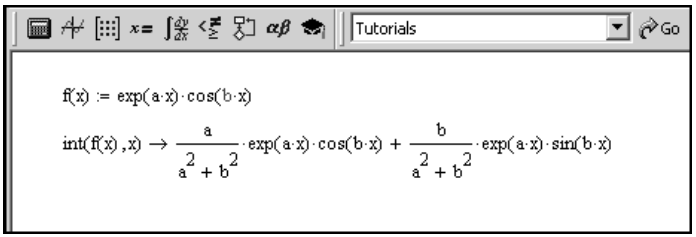


Рис. 5.36. Вычисление интеграла с помощью команды Maple

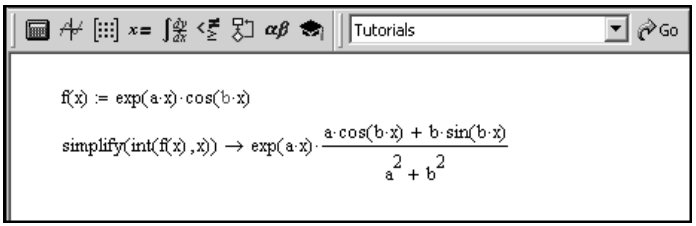


Рис. 5.37. Интегрирование и упрощение результата

Очевидно, что в результате выражение выглядит более компактно. К сожалению, добиться существенного упрощения выражений такими простыми методами удается далеко не всегда.

В Maple команда `int()` пригодна также и для вычисления определенных интегралов. При этом для переменной интегрирования указывается диапазон ее изменения. Проблема в том, что такой диапазон вводится с помощью знака равенства. Однако Mathcad любую попытку ввода знака равенства интерпретирует по-своему, поэтому определенные интегралы лучше все же вычислять методами, специально предусмотренными для этого в Mathcad.

Обращаем внимание читателя на то, что работа с командами Maple является рудиментом, в определенном смысле экзотикой. В последующих после Mathcad 11 версиях такие возможности не предусмотрены.

Работа с выражениями

В процессе решения задач приходится работать с выражениями самого различного рода. Эти выражения, как правило, преобразуются, упрощаются, разбиваются на слагаемые и т. п. В Mathcad для этих целей предусмотрены специальные утилиты, о которых речь пойдет в данном разделе.

Пример 5.6. Упрощение выражений

Обычно выражения упрощаются на промежуточных этапах вычислений. Для этого имеется команда `simplify()`, которая, как отмечалось, заимствована из Maple. В исходном выражении следует выделить упрощаемое подвыражение (или все выражение, если оно упрощается целиком), после чего выбрать команду **Symbolics | Simplify** (рис. 5.38).

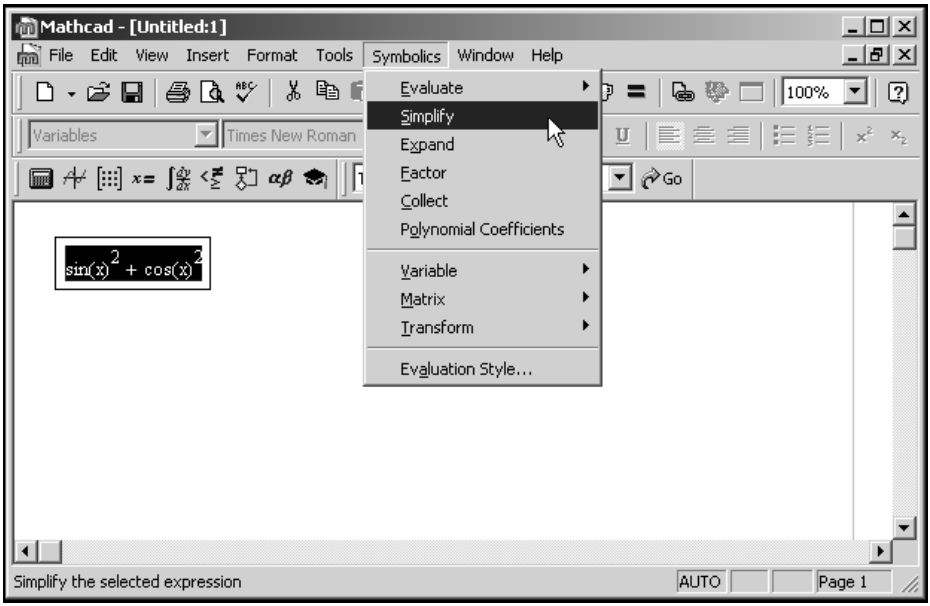


Рис. 5.38. Выбор команды **Simplify** для упрощения выражения

Результат отображается внизу под упрощаемым выражением. В частности, в рассматриваемом случае упрощается выражение $\sin^2(x)+\cos^2(x)$, которое, как известно, тождественно равняется единице (рис. 5.39).

Точно такой же результат можно получить в Mathcad 11, если упрощаемое выражение указать аргументом команды `simplify()` в рабочем документе. Чтобы команда упрощения выражений выполнилась, вводят оператор вычисления символьного значения (рис. 5.40).

Существует возможность упрощать выражения через команды палитры символьных вычислений. Для этого выбирается упрощаемое выражение, а на панели символьных вычислений щелкают на пиктограмме с названием команды **simplify** (рис. 5.41).

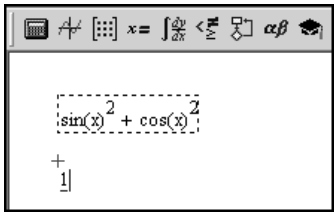


Рис. 5.39. Результат упрощения выражения

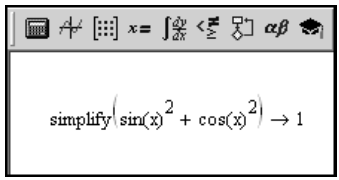


Рис. 5.40. Ввод команды упрощения в рабочем документе Mathcad 11

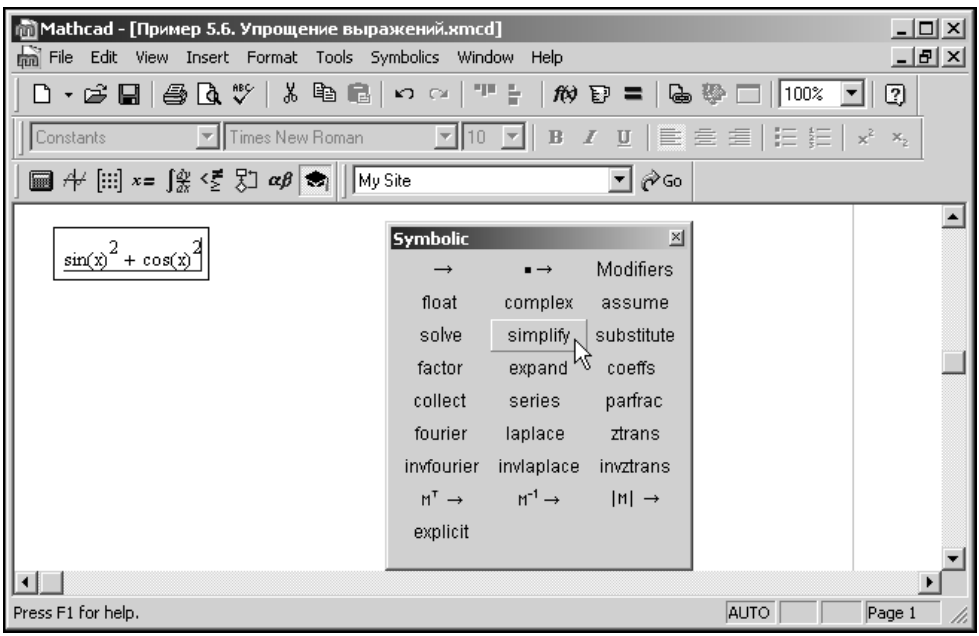


Рис. 5.41. Упрощение выражения с помощью пиктограмм панели символьных вычислений

После щелчка на пиктограмме за выражением отображается ключевое слово `simplify` с последующим оператором вычисления символьного значения (рис. 5.42).

Чтобы увидеть результат упрощения, следует щелкнуть мышью вне области выражения (рис. 5.43).

Заметим, что результат упрощения выражений может быть достаточно обескураживающим. Пример приведен на рис. 5.44.

В этом случае предпринималась попытка упростить выражение $1-2\sin^2(x)$, которое равно $\cos(2x)$. Однако в результате упрощения получено выражение $2\cos^2(x)-1$, которое хотя и является корректным, однако вряд ли может счи-

таться более простым по сравнению с исходным, упрощаемым выражением. Вообще же процедура упрощения существенно зависит от тех задач, в рамках которых она выполняется. А это достаточно индивидуально, и полная автоматизация здесь маловероятна.

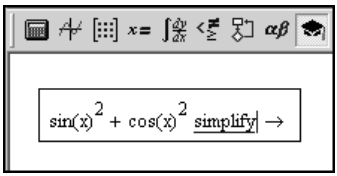


Рис. 5.42. Рабочий документ после щелчка на пиктограмме панели символьных вычислений

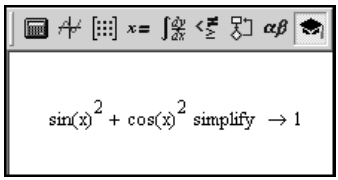


Рис. 5.43. Результат упрощения выражения

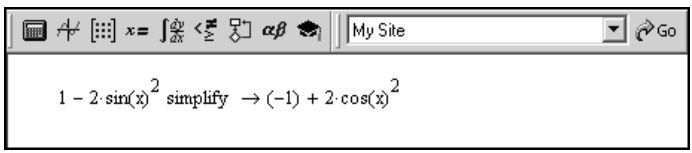


Рис. 5.44. Упрощение тригонометрического выражения

Нередко приходится выполнять преобразования при некоторых дополнительных условиях. Если они в принципе позволяют хоть как-то упростить выражение, разумно передать информацию о дополнительных условиях системе. Разумеется, не всякое условие можно описать в терминах, воспринимаемых приложением Mathcad. Однако часто это сделать удается.

Пример 5.7. Упрощение при дополнительных условиях

Если упрощаемое или преобразуемое выражение существенно зависит от некоторой переменной, область допустимых значений которой ограничена, вводят ключевое слово (инструкцию) `assume` с указанием накладываемого ограничения. В качестве иллюстрации рассмотрим простой пример. Соответствующий документ Mathcad показан на рис. 5.45.

Чтобы ввести ключевое слово `assume`, следует воспользоваться пиктограммой на палитре **Symbolic** (рис. 5.46).

В первом случае при извлечении квадратного корня в качестве ограничения было указано неравенство, свидетельствующее о том, что выражение, квадрат которого находится под символом радикала, положительно. Во втором случае условием является принадлежность переменной определенному диапазону значений (функция `RealRange()`), в силу чего можно упростить (причем существенно) выражения, размещенные под знаком модуля. Кстати,

ключевые слова можно вводить с помощью специальной палитры, которая открывается после выбора команды **View | Toolbars | Modifier** или щелчка на пиктограмме **Modifiers** на палитре **Symbolic** (см. рис. 5.46). Палитра **Modifier** показана на рис. 5.47.

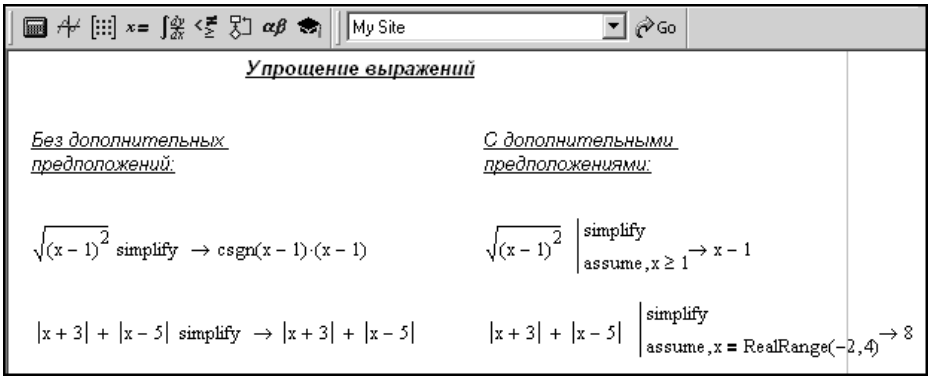


Рис. 5.45. Упрощение выражений с дополнительными условиями

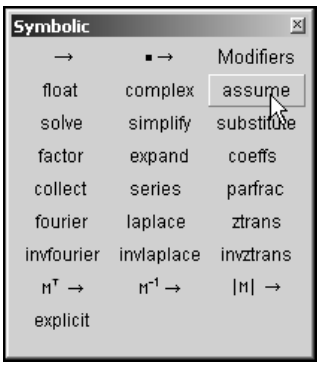


Рис. 5.46. Выбор пиктограммы **assume** на палитре **Symbolic**

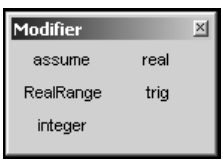


Рис. 5.47. Палитра **Modifier**

Кроме уже знакомого ключевого слова **assume** и **RealRange**, там есть пиктограммы с ключевыми словами **real**, **integer** и **trig**, которые соответствуют упрощению выражений с действительными, целыми и тригонометрическими переменными.

Что такое упрощение выражения — на самом деле вопрос философский. Одна и та же операция в одном случае может рассматриваться как упрощение, а в другом — нет. Многое зависит от конкретной решаемой задачи. Кроме того, встроенные алгоритмы упрощения выражений не всегда очевидны, поэтому приходится более или менее явно указывать, какого рода действия

следует выполнить. Например, если при работе с полиномами, представленными в виде произведений, следует разложить выражение на отдельные слагаемые, выбирают инструкцию `expand`.

Пример 5.8. Преобразование выражений

На рис. 5.48 показан пример с этой инструкцией для разложения выражений на слагаемые.

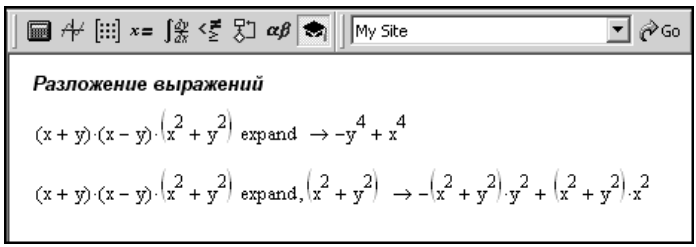


Рис. 5.48. Разложение выражений на слагаемые

Инструкцию `expand` используют по-разному. Преобразуемое выражение вводят слева от инструкции. Кроме этого, справа от инструкции иногда указывают часть выражения (множитель), которую при преобразовании следует оставить в неизменном виде. На рис. 5.48 в таком режиме преобразуется второе выражение. Выполнить обратную операцию, т. е. представить выражение в виде произведения, можно с помощью инструкции `factor`. Примеры представлены на рис. 5.49.

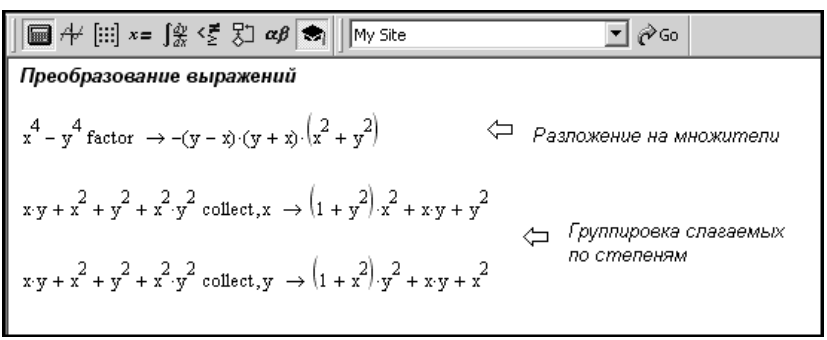


Рис. 5.49. Преобразование выражений

Там же показана работа с инструкцией `collect`. Слева указывается полиномиальное выражение, в котором следует сгруппировать слагаемые по степеням одной из переменных, а справа — та переменная, по степеням которой следует выполнять группировку.

Пример 5.9. Коэффициенты полинома

Что касается выражений полиномиального типа, то часто бывает важно определить его коэффициенты, т. е. множители при степенных слагаемых. Вектор коэффициентов полинома возвращается с помощью инструкции `coeffs` (рис. 5.50).

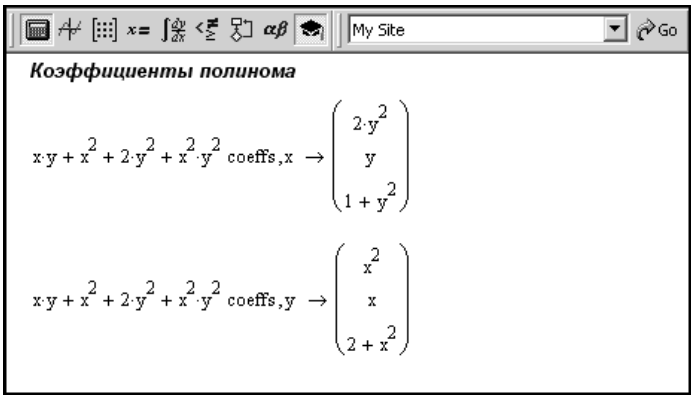


Рис. 5.50. Вычисление коэффициентов полиномиального выражения

В зависимости от того, по какой переменной группируются слагаемые, различными будут и коэффициенты.

Пример 5.10. Замена переменных

Еще одна важная процедура, к которой прибегают при символьном преобразовании выражений, — замена переменных или подстановки, которые в Mathcad можно осуществлять посредством инструкции `substitute`. На рис. 5.51 вместо переменной `y` введено выражение `sin(z)`.

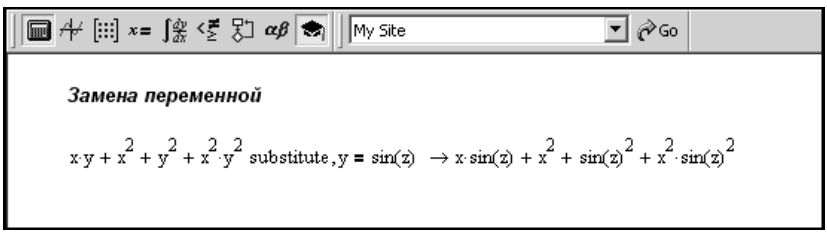


Рис. 5.51. Замена переменной в выражении

Пример 5.11. Разложение на простые дроби

На рис. 5.52 показано, как, применяя инструкцию `parfrac`, можно раскладывать рациональные выражения на простые дроби.

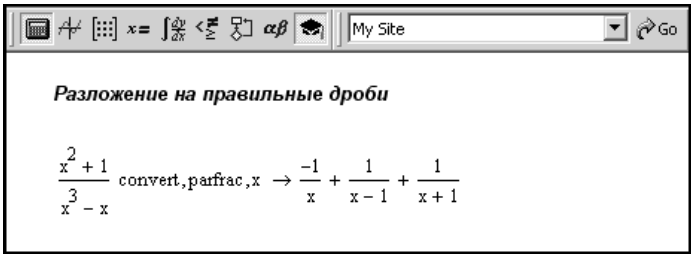


Рис. 5.52. Разложение выражения на правильные дроби

Выполняется щелчок на пиктограмме **parfrac** палитры **Symbolic**, инструкция `convert` при этом добавляется автоматически.

Пример 5.12. Вычисление выражений

Наконец, при обработке символьных выражений на некоторых этапах работы важно бывает вычислить или оценить в численном виде значение этого выражения. Для вычисления выражения в численном виде после него указывают инструкцию `float`, справа от которой вводится число, определяющее точность вычисления выражения. Кроме этого, имеется возможность вычислять выражения на множестве комплексных чисел. Соответствующая инструкция носит название `complex`. Примеры вычисления выражений можно видеть в документе на рис. 5.53.

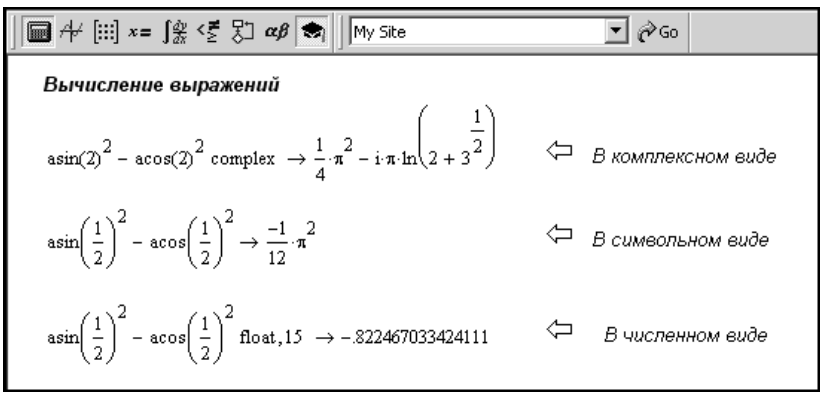


Рис. 5.53. Вычисление выражений в комплексном, символьном и численном виде

При вычислении арксинуса или арккосинуса от числа, большего по абсолютной величине единицы, результат будет комплексным.

Символьные операции с матрицами

Возможности по выполнению символьных преобразований при работе с матрицами не очень велики. Однако ряд основных, наиболее общих операций в Mathcad с матрицами делать можно. Это процедуры транспонирования, нахождения обратной матрицы и вычисления определителя (детерминанта).

Пример 5.13. Операции с матрицами

На рис. 5.54 показана начальная часть документа, где матрица сначала инициализируется в символьном виде, а затем транспонируется.

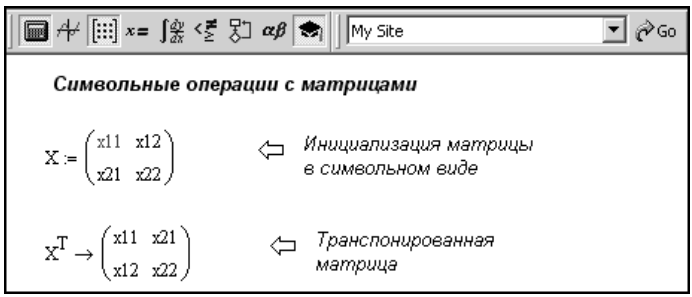


Рис. 5.54. Инициализация и транспонирование матрицы

Для транспонирования матрицы на палитре **Symbolic** выбирают специальную пиктограмму с символом транспонирования (рис. 5.55).

Чтобы найти обратную матрицу, на палитре **Symbolic** выбирают пиктограмму, соседнюю с пиктограммой транспонирования (рис. 5.56).

Результат обращения матрицы представлен на рис. 5.57. Там же показано, как обращается (в символьном виде) матрица с численными значениями.

Основная особенность обращения численной матрицы в символьном виде состоит в том, что (см. рис. 5.57) элементы обратной матрицы записаны в виде рациональных дробей, т. е. к формату чисел с плавающей точкой они не преобразуются.

Как отмечалось ранее, рассматриваемые операции по символьному преобразованию можно выполнять не только с помощью палитры **Symbolic**, но и посредством команд меню **Symbolics**. В частности, чтобы вычислить детерминант матрицы в символьном виде, достаточно выбрать команду **Symbolics | Matrix | Transpose**. Результат вычисления детерминанта матрицы с помощью этой команды показан на рис. 5.58.

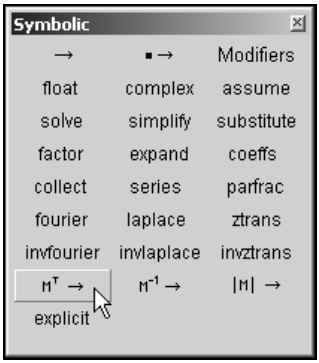


Рис. 5.55. Выбор пиктограммы для транспонирования матрицы

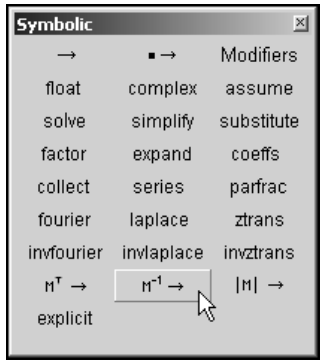


Рис. 5.56. Выбор пиктограммы для вычисления обратной матрицы

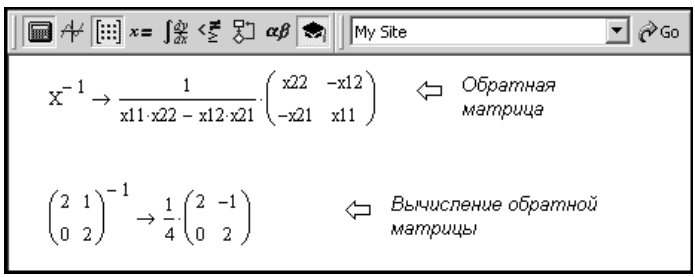


Рис. 5.57. Вычисление обратной матрицы

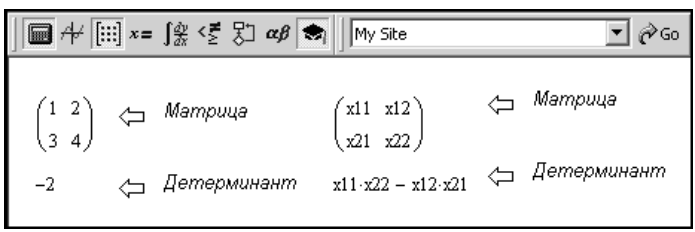


Рис. 5.58. Вычисление детерминанта матрицы

Ситуация с использованием пиктограммы вычисления детерминанта на палитре **Symbolic** достаточно интересная, поскольку соответствующая команда (рис. 5.59) успешно может применяться и для вычисления модуля комплексных чисел в символьном виде.

Более того, для определения детерминанта допустимо воспользоваться пиктограммой вычисления детерминанта матрицы на палитре **Matrix** (рис. 5.60) и затем командой вычисления выражения в символьном виде (пиктограмма со стрелкой на палитре **Symbolic**).

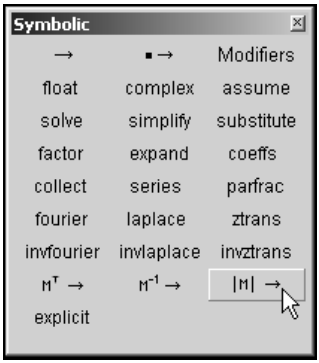


Рис. 5.59. Выбор команды вычисления детерминанта на палитре **Symbolic**

Рис. 5.60. Выбор пиктограммы вычисления детерминанта на палитре **Matrix**

На рис. 5.61 показан результат вычисления модуля комплексного числа в символьном виде с помощью пиктограммы палитры **Symbolic** и детерминанта матрицы.

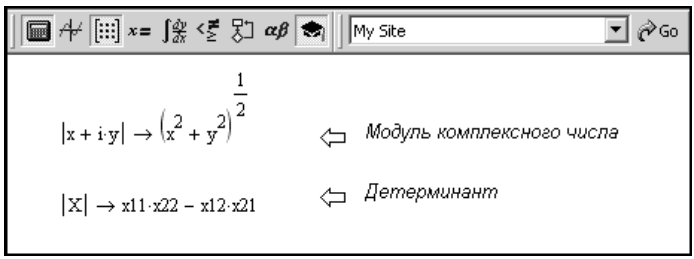


Рис. 5.61. Результат вычисления определителя матрицы

Каким способом вычислять детерминант матрицы — это, разумеется, решать пользователю.

Интегральные преобразования и разложение в ряды

Важный метод, часто встречающийся при решении уравнений и целого класса практических задач, — интегральные преобразования. Наиболее известны и популярны преобразования Лапласа и Фурье. Существуют прямое и обратное преобразования. Прямое преобразование Фурье для функции $f(t)$ обычно осуществляется по формуле $F(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)\exp(-i\omega t)dt$. При таком опре-

делении прямого преобразования Фурье обратное дается соотношением $f(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} F(\omega)\exp(i\omega t)d\omega$. В Mathcad прямое преобразование определяется как $F(\omega) = \int_{-\infty}^{+\infty} f(t)\exp(-i\omega t)dt$, в силу чего для обратного преобразования используется формула $f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)\exp(i\omega t)d\omega$. Для прямого преобразования Лапласа принято соотношение $F(p) = \int_{0}^{+\infty} f(t)\exp(-pt)dt$. Для обратного преобразования выполняется интегрирование по комплексной области, поэтому приводить соответствующее выражение для интеграла не будем. Отметим лишь, что в результате этого по функции $F(p)$ можно восстановить $f(t)$.

Несложно заметить, что формальные выражения для интегралов в обоих преобразованиях достаточно схожи. Правда, в преобразовании Лапласа переменная p в общем случае комплексная, но это сейчас не принципиально. Важно то, что у этих преобразований достаточно схожи и основные свойства. Поэтому на практике критерием того, какое из двух преобразований выбрать, является, как ни странно, возможность выполнить преобразование. Дело в том, что преобразование Фурье можно выполнить далеко не для каждой функции. Например, для полинома (частным случаем которого является линейная функция) соответствующий интеграл расходится. Преобразование Фурье часто применяется при анализе сигналов, в том числе и типа "елочка". Однако эти сигналы, хотя и характеризуются линейной временной зависимостью, но она ограничена во времени. Другими словами, соответствующая функция является кусочно-гладкой, а это не одно и то же, что линейная зависимость.

Для выполнения преобразования Фурье в Mathcad предусмотрено ключевое слово `fourier`. Слева указывается преобразуемая функция, а справа — переменная, по которой следует выполнять преобразование. Для обратного преобразования Фурье можно прибегнуть к помощи инструкции `invfourier`. Обе инструкции можно ввести посредством палитры **Symbolic**.

Пример 5.14. Интегральные преобразования

Примеры выполнения прямого и обратного преобразований показаны в документе на рис. 5.62.

Преобразование Фурье

$f(t) := \exp(-t^2)$ $\quad F(\omega) := \sqrt{\pi}\cdot\exp\left(\frac{-\omega^2}{4}\right)$ ⇦ *Инициализация функций*

$f(t)\ \text{fourier}, t \rightarrow \pi^{\frac{1}{2}}\cdot e^{\frac{-1}{4}\cdot\omega^2}$ ⇦ *Прямое преобразование*

$F(\omega)\ \text{invfourier}, \omega \rightarrow \frac{1}{2}\cdot 4^{\frac{1}{2}}\cdot e^{-t^2}\ \text{simplify} \rightarrow e^{-t^2}$ ⇦ *Обратное преобразование*

Проверка результата

$\int_{-\infty}^{\infty} f(t)\cdot\exp(-i\cdot\omega\cdot t)\,dt \rightarrow \pi^{\frac{1}{2}}\cdot e^{\frac{-1}{4}\cdot\omega^2}$ $\quad \frac{1}{2\cdot\pi}\int_{-\infty}^{\infty} F(\omega)\cdot\exp(i\cdot\omega\cdot t)\,d\omega \rightarrow e^{-t^2}$

Рис. 5.62. Преобразование Фурье

Практически так же выполняется преобразование Лапласа. Пример выполнения такого преобразования приведен в документе на рис. 5.63.

Преобразование Лапласа

$f(t) := \exp(-t)\cdot\sin(t)$ $\quad F(p) := \frac{1}{(p+1)^2+1}$ ⇦ *Инициализация функций*

$f(t)\ \text{laplace}, t \rightarrow \frac{1}{(s+1)^2+1}$ ⇦ *Прямое преобразование*

$F(p)\ \text{invlaplace}, p \rightarrow e^{-t}\cdot\sin(t)$ ⇦ *Обратное преобразование*

Проверка результата

$\int_{0}^{\infty} f(t)\cdot\exp(-p\cdot t)\,dt\ \text{assume}, p > 0 \rightarrow \frac{1}{2+2\cdot p+p^2}$

Рис. 5.63. Преобразование Лапласа

Если преобразование (Лапласа или Фурье) выполняется с помощью команд меню, важно не забыть, что предварительно в преобразуемом выражении следует выделить ту переменную, по которой выполняется преобразование. Процесс выполнения преобразования с помощью команд меню **Symbolics** проиллюстрирован на рис. 5.64.

Результат преобразования показан на рис. 5.65.

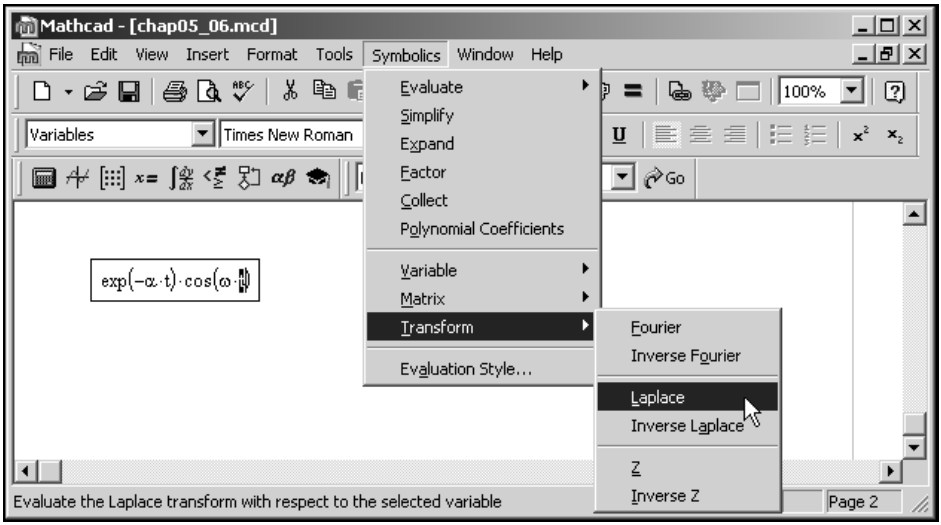


Рис. 5.64. Выполнение преобразования Лапласа с помощью команд меню

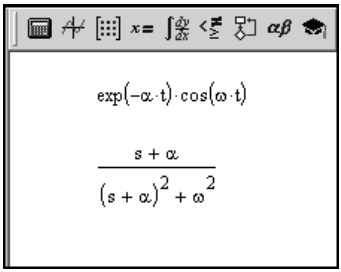


Рис. 5.65. Результат выполнения преобразования Лапласа с помощью команд меню

Как обычно при выполнении символьных преобразований посредством команд меню, результат отображается рядом с исходным выражением (в данном случае снизу). При изменении исходного выражения результат преобразования останется неизменным. При этом корректный выбор переменной преобразования (т. е. переменной, по которой выполняется интегрирование)

является принципиальным, поскольку для разных переменных результаты преобразования абсолютно разные. На рис. 5.66 помечена (линиями редактирования) переменная, отличная от той, что выделялась ранее.

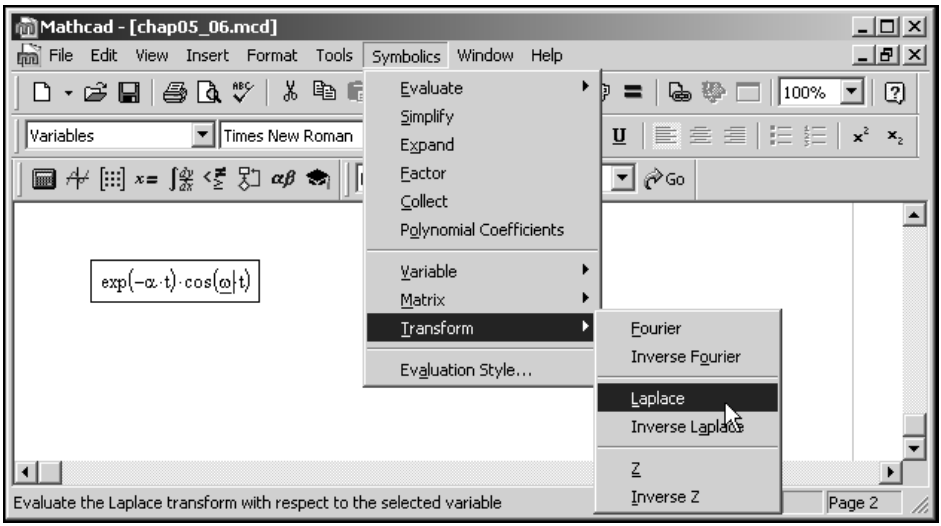


Рис. 5.66. Выделена переменная в выражении

Результат преобразования Фурье в этом случае показан на рис. 5.67. Как и следовало ожидать, он изменился.

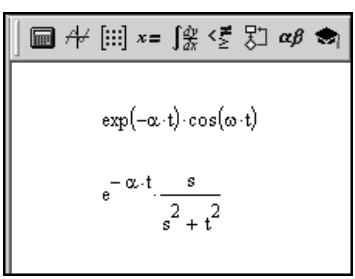


Рис. 5.67. Результат преобразования зависит от переменной преобразования

Хотя преобразования Лапласа и Фурье достаточно продуктивны в плане их применения на практике, это далеко не единственные преобразования, которые полезны при решении задач. В Mathcad существует также возможность выполнять так называемое Z-преобразование (прямое и обратное). Для функции целочисленного аргумента $f(n)$ Z-преобразование выполняется по фор-

муле $F(z) = \sum_{n=0}^{\infty} \frac{f(n)}{z^n}$. Отсюда легко заметить, что, во-первых, соответствующее выражение есть степенной ряд по переменной $x = z^{-1}$ и, во-вторых, если значения $f(n)$ определяют производные (деленные на $n!$) для некоторой функции, то Z-преобразование будет рядом Тейлора для этой функции (ее аргументом является $1/z$).

Пример 5.15. Z-преобразование

Например, если $f(n) = 1/n!$, то $F(z) = \exp(1/z)$. В том, что это действительно так, несложно убедиться, взглянув на рис. 5.68.

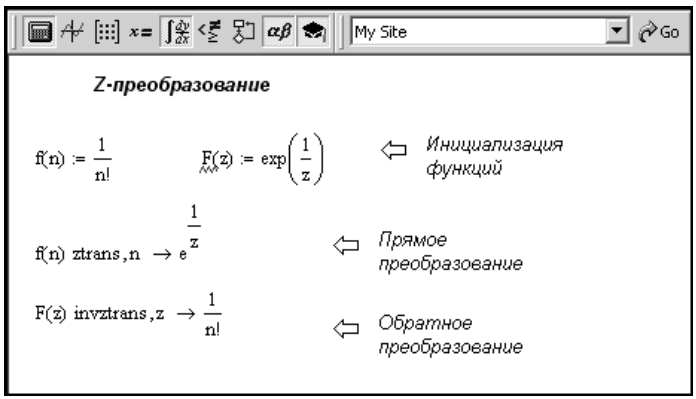


Рис. 5.68. Выполнение Z-преобразования

Прямое преобразование выполнено с помощью инструкции `ztrans`, а обратное — с помощью `invztrans`. Еще один пример преобразования, результат которого можно предугадать, — для функции $f(n) = 1$. Действительно, поскольку $\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$, то должно быть $F(z) = \frac{1}{1-1/z}$ (рис. 5.69).

Что касается обратного преобразования, то в этом случае вычисляется криволинейный интеграл по замкнутой комплексной области. Как и для преобразования Лапласа, приводить его не будем. Важно то, что по функции $F(z)$ можно восстановить $f(n)$. Для этого, кстати, можно было бы разложить $F(z)$ в ряд по степеням $1/z$. Убедимся в этом. Для разложения функций в ряд Тейлора воспользуемся инструкцией `series`.

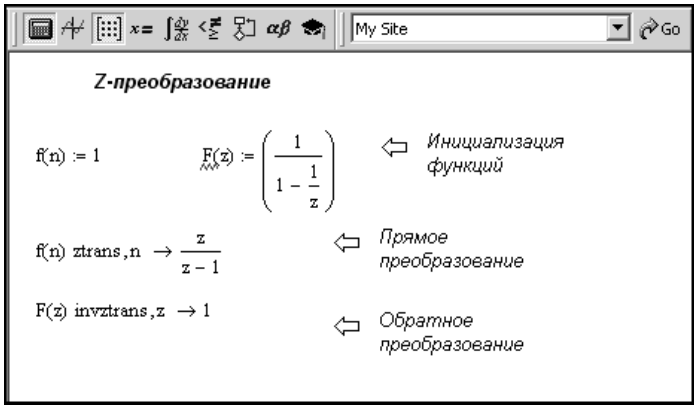


Рис. 5.69. Результат некоторых преобразований легко предугадать

Пример 5.16. Ряд Тейлора

Результат разложения полученных ранее функций в ряд Тейлора показан на рис. 5.70.

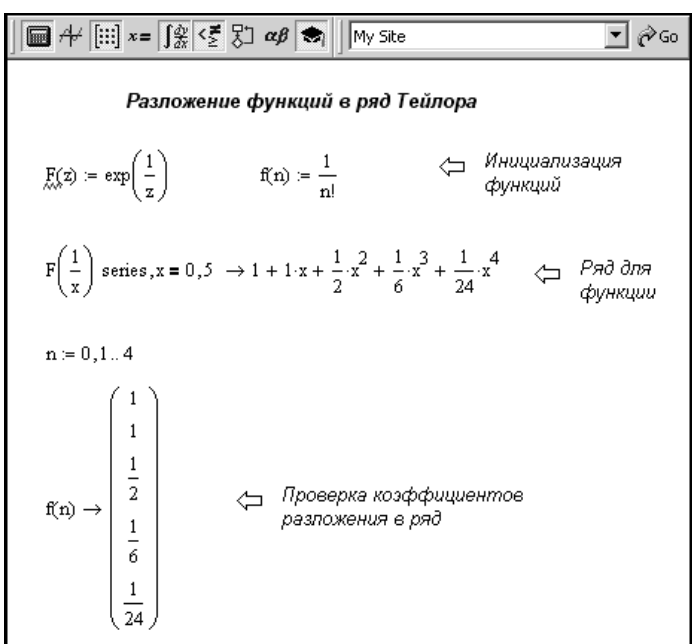


Рис. 5.70. Разложение функции в ряд

Слева от инструкции `series` указывается раскладываемая в ряд функция, а справа — переменная со значением, в окрестности которого функция рас-

кладывается в ряд (если разложение выполняется в окрестности нуля, ряд называется рядом Маклорена). Далее через запятую можно указать число вычисляемых слагаемых в ряде. В нижней части документа на рис. 5.70 приведено соответствующее число начальных значений для функции, которая ранее подвергалась Z-преобразованию. Поскольку в ряд раскладывалась функция, полученная в результате этого преобразования, значения совпадают с коэффициентами разложения. В том, что такое совпадение далеко не случайно, свидетельствуют и данные документа, показанного на рис. 5.71.

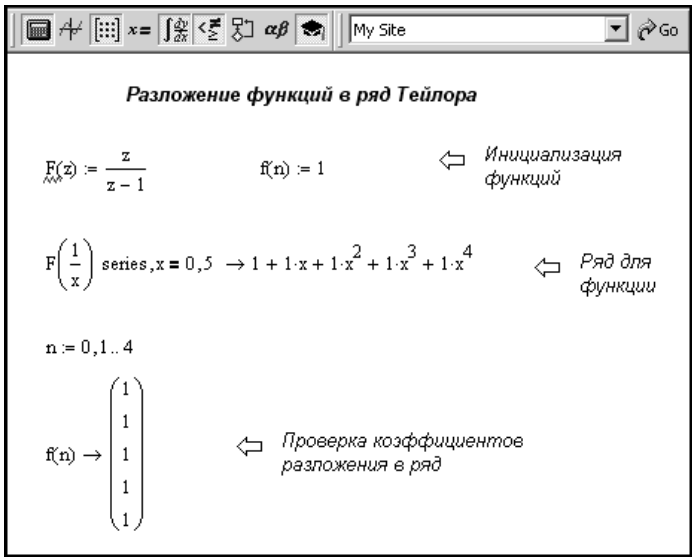


Рис. 5.71. Разложение в ряд и проверка коэффициентов разложения

Выбирая различные преобразования и комбинируя их, можно решать самые разнообразные задачи. Частично такие подходы будут использованы далее при рассмотрении практических примеров.

Решение уравнений в символьном виде

В Mathcad уравнения можно решать не только в численном, но и в символьном виде. В рабочий документ вводится уравнение, после чего с помощью пиктограммы **solve** на палитре **Symbolic** вводится одноименная инструкция `solve` (рис. 5.72).

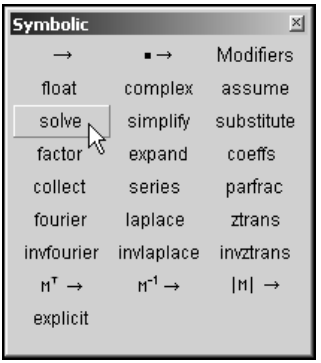


Рис. 5.72. Пиктограмма для решения уравнения

Пример 5.17. Решение уравнений и систем

Результат применения этой инструкции рассмотрим на примере решения квадратного уравнения $x^2 - 5x + 6 = 0$. Соответствующий документ представлен на рис. 5.73.

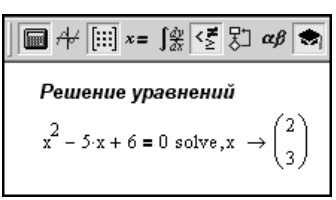


Рис. 5.73. Решение квадратного уравнения

Справа от инструкции `solve` указывают ту переменную, относительно которой решается уравнение. Не следует также забывать, что в качестве знака равенства в уравнении вводится оператор логического равенства (его пиктограмма размещена на палитре **Boolean**). Строго говоря, правую часть уравнения можно вообще не указывать. Если слева от инструкции `solve` введено выражение, то решается уравнение, получающееся приравниванием этого выражения нулю.

Если нужно решить систему уравнений, то возможны два принципиально разных подхода. Один состоит в том, что уравнения задаются в виде элементов вектора. Этот вектор затем указывают слева от инструкции `solve`, а справа от нее через запятую перечисляются переменные, относительно которых решается система. Результат возвращается в виде вектора решений. На рис. 5.74 приведен пример такого способа решения системы уравнений $x^2 + y^2 = 1$ и $y - x^2 = 1$.

Рис. 5.74. Решение системы уравнений, записанной в виде вектора

Как видим, системой найдены все решения, в том числе и комплексные. Альтернативный способ решения системы уравнений заключается в использовании специального блока, который начинается ключевым словом `Given` (вводится с клавиатуры) и заканчивается функцией `Find()` (не обязательно этой, но именно она в данном случае представляет интерес). Аргументом функции указывают те переменные, относительно которых решается система уравнений. Сами уравнения должны быть записаны в блоке (т. е. между ключевым словом `Given` и командой поиска решения `Find()`). Пример решения линейной системы уравнений таким методом представлен на рис. 5.75.

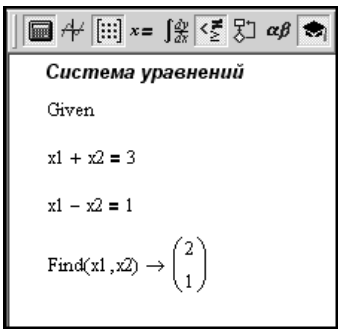


Рис. 5.75. Решение линейной системы уравнений

Очевидно, что решение найдено правильно. Практически так же решаются неравенства и системы неравенств.

Суммирование рядов, вычисление границ и произведений

Достаточно несложными являются операции аналитического расчета рядов, произведений и границ. Рассмотрим кратко особенности каждой из этих операций. Однако сразу отметим, что особенность вычисления соответствующих выражений в символьном виде (по сравнению с определением того же выражения в формате числа с плавающей точкой) состоит, в основном, в выборе соответствующего оператора — в данном случае следует вводить стрелку вычисления символьного результата.

Пример 5.18. Вычисление бесконечных сумм и произведений

На рис. 5.76 приведены примеры вычисления в символьном виде сумм бесконечных рядов.

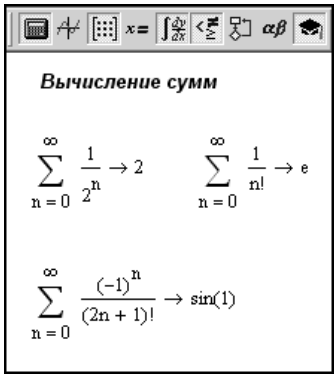


Рис. 5.76. Вычисление рядов в символьном виде

Практически так же вычисляются и бесконечные произведения. Примеры представлены в документе на рис. 5.77.

Символы вычисления суммы и произведения вводятся с помощью пиктограмм палитры **Calculus**. Здесь же имеется целых три пиктограммы для вычисления пределов. Дело в том, что для разрывных функций предел определяется справа или слева (для непрерывных оба эти предела совпадают). Для нахождения обычного (двустороннего) предела выбирают пиктограмму в левом нижнем углу палитры **Calculus** (при определенной форме палитры, рис. 5.78).

Пиктограммы для вычисления правосторонних и левосторонних пределов выделены на рис. 5.79 и рис. 5.80.

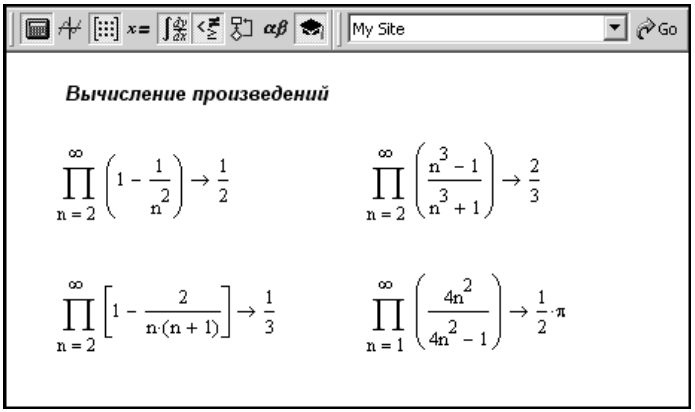


Рис. 5.77. Вычисление бесконечных произведений

Рис. 5.78. Пиктограмма вычисления двустороннего предела

Рис. 5.79. Пиктограмма вычисления правостороннего предела (предел сверху)

Рис. 5.80. Пиктограмма вычисления левостороннего предела (предел снизу)

Пример 5.19. Вычисление пределов

Примеры вычисления пределов показаны в документе на рис. 5.81.

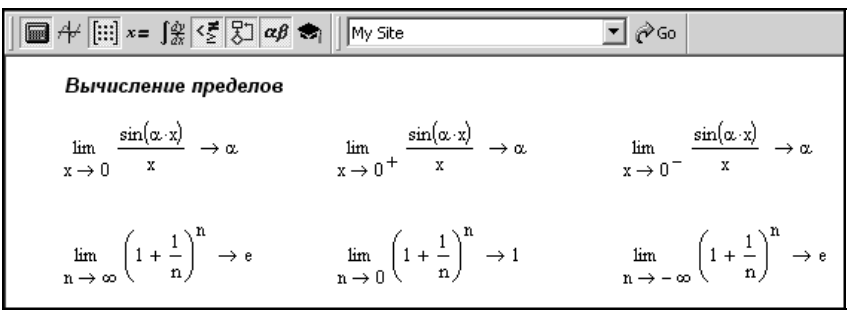


Рис. 5.81. Вычисление пределов

Среди прочих, там приведены левосторонние и правосторонние пределы. Поскольку функция, для которой определяется предел, является непрерывной,

все такие пределы совпадают. В документе на рис. 5.82 приведен фрагмент документа, в котором задана разрывная функция. Функция может принимать всего два целочисленных значения: –1 при отрицательном аргументе и 1 — в противном случае. Далее приводятся значения пределов (двустороннего, правостороннего и левостороннего) для этой функции.

Рис. 5.82. Вычисление предела разрывной функции

Для сравнения приведены также значения функции в окрестности нулевой точки.

Использование команд меню

Большинство рассмотренных операций описывалось с акцентом на использование при расчете аналитических выражений оператора вычисления символьного значения. Ранее отмечалось, что данный подход имеет то преимущество, что результаты вычислений в этом случае автоматически обновляются при изменениях в рабочем документе. Однако иногда бывает быстрее воспользоваться командами меню **Symbolics**. Практически для всех приведенных примеров, существуют альтернативные способы их решения с помощью команд меню **Symbolics**. Выбор способа определяется обычно исходя из соображений удобства. При работе с командами меню предусмотрена возможность определения пользователем вида отображения результата. По умолчанию результат символьного вычисления появляется внизу под исходным выражением. Чтобы изменить положение вещей, можно воспользоваться командой **Symbolics | Evaluation Style**. В результате откроется диалоговое окно, показанное на рис. 5.83.

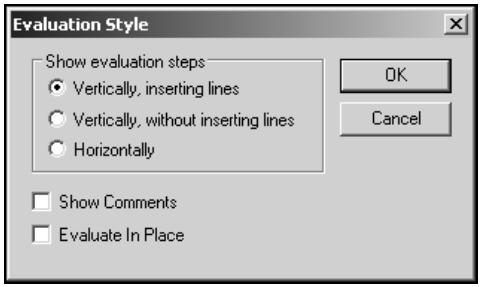


Рис. 5.83. Диалоговое окно **Evaluation Style**

В диалоговом окне размещена группа переключателей на три положения и две опции. Положение переключателя определяет, где будет отображаться результат символьных вычислений: снизу под вычисляемым выражением или справа от него. При отображении результата снизу существует два режима: с добавлением или без добавления строки. Если переключатель установлен в положение **Vertically, inserting lines** (положение по умолчанию), то после вычисления результат не только добавляется под исходным выражением, но и все прочие данные под этим выражением смещаются вниз. При положении переключателя **Vertically, without inserting lines** этого не происходит — результат появляется снизу под выражением, но данные не смещаются. Наконец, чтобы результат выводился справа от вычисляемого выражения, переключатель должен быть установлен в положение **Horizontally**.

При установленном флажке опции **Show Options** в результате выполнения символьных вычислений появится не только результат, но и комментарий к нему. Иногда это бывает исключительно удобно. А с опцией **Evaluate In Place** ситуация более сложная. Если флажок опции установлен, все прочие элементы управления диалогового окна **Evaluation Style** будут недоступными, а результат символьных вычислений отобразится вместо вычисляемого выражения.

Пример 5.20. Использование команд меню

На рис. 5.84 и рис. 5.85 показан результат выполнения нескольких операций по вычислению выражений в символьном виде. Вычисления осуществлялись с помощью команд меню **Symbolics** в режиме отображения комментариев.

Настройки диалогового окна **Evaluation Style** показаны на рис. 5.86.

Для упрощения выражений выбирают команду **Symbolics | Simplify**. Разложение осуществляется посредством команды **Symbolics | Expand**. Процедура по представлению выражения в виде произведения (или дроби с общим знаменателем) осуществляется выбором команды **Symbolics | Factor**. Обратная операция, позволяющая разбить дробь на сумму простых дробей, может быть выполнена, если выбрать команду **Symbolics | Variable | Convert to**

Partial Fraction. При этом в преобразуемом выражении должна быть выделена соответствующая переменная.

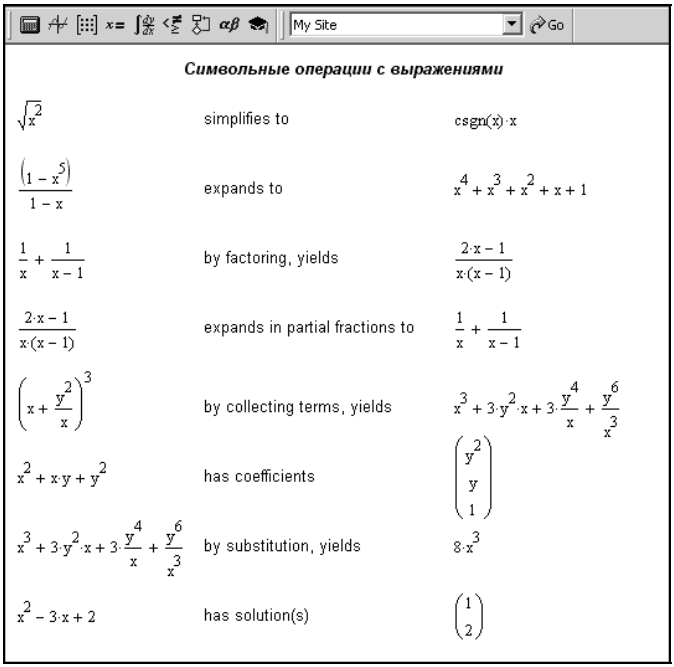


Рис. 5.84. Результаты символьных вычислений и комментарии к ним

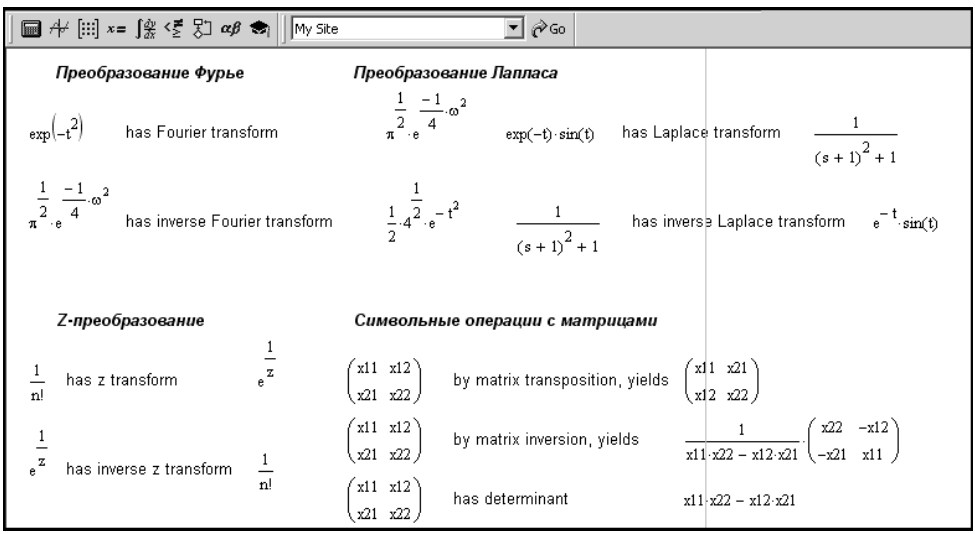


Рис. 5.85. Символьные преобразования с комментариями

Рис. 5.86. Настройки диалогового окна **Evaluation Style**

Если выделить в выражении переменную и выбрать команду **Symbolics | Collect**, получим (если это возможно) полином по этой переменной. Коэффициенты полинома определяют с помощью команды **Symbolics | Polynomial Coefficients**. Здесь также сначала выделяется полиномиальная переменная.

Чтобы осуществить замену переменных в выражении, значение, на которое будет меняться переменная, копируется в буфер обмена, после чего заменяемая переменная выделяется и выбирается команда **Symbolics | Variable | Substitute**.

Решать уравнения с помощью команд меню также достаточно просто. В выражении (это не обязательно должно быть равенство) выделяется переменная, относительно которой уравнение будет решаться. Далее прибегают к команде **Symbolics | Variable | Solve**.

Весь набор прямых и обратных преобразований выполняется путем выбора соответствующей команды из подменю **Symbolics | Transforms**. Названия команд однозначно указывают на тип выполняемого преобразования. Наконец, символьные манипуляции с матрицами (вычисление транспонированной и обратной матрицы, а также детерминанта матрицы) становятся доступными через команды **Transpose** (транспонирование), **Invert** (обращение матрицы) и **Determinant** (детерминант матрицы) подменю **Symbolics | Matrix**. Результаты интегральных преобразований (прямых и обратных) и операций с матрицами были представлены в документе на рис. 5.85.

В заключение этой главы хочется отметить, что здесь были описаны далеко не все важные и полезные методы символьных вычислений. Упоминались только наиболее простые, те, которые могут быть выполнены в Mathcad, что называется, в одно действие (или почти так). Возможности Mathcad этим совсем не ограничиваются. Спектр задач, решаемых в символьном виде, существенно шире. Часть из них будет описываться далее в этой книге, но уже как один из этапов в решении более сложной задачи.

Глава 6

Задачи высшей математики

В этой главе рассматриваются способы решения задач высшей математики с использованием приложения Mathcad. В начале главы приводятся примеры, связанные с вычислением производных, в том числе от функций, заданных в неявном и параметрическом виде. Во всех случаях речь идет о получении результата в символьном виде. Численные методы обсуждаются в *главе 7*.

Кроме производных, здесь можно также найти сведения о вычислении интегралов: определенных и неопределенных, двойных, тройных и контурных. Кратко рассматриваются дифференциальные уравнения и задачи физики. Проиллюстрированы методы разложения функций в ряды, в том числе в ряды Фурье, а также интегральные преобразования.

Вычисление производных

В *главе 5* о вычислении производных речь уже шла. Если функция задана в явном виде, ее дифференцирование особой сложности не представляет.

Производная явной функции

Как известно, в наиболее простом случае функциональной зависимостью устанавливается связь между двумя числами (если речь идет о числовых однозначных функциях) — аргументом функции и ее значением. Эта зависимость может быть задана разными способами. Обычно указывается выражение, согласно которому по аргументу функции вычисляется ее значение. Тогда говорят, что функция задана в явном виде.

Задача 6.1. Найти производную функции $y(x) = \dfrac{1 + x - x^2}{1 - x + x^2}$.

Вычисление производной от функции, заданной в явном виде, — достаточно тривиальная задача. Можно дифференцировать непосредственно выражение, определяющее функцию, либо сначала определить функциональную зависимость, а затем продифференцировать функцию. Последнее предпочтительнее, особенно если вычисление производной — всего лишь промежуточный этап при решении более сложной задачи. В этом случае созданный документ легко может стать шаблоном для решения других подобных задач. Обычно бывает достаточно внести изменения в команду определения дифференцируемой функции. Результаты остальных команд в документе будут пересчитаны автоматически. На рис. 6.1 приведен фрагмент документа с решением поставленной задачи.

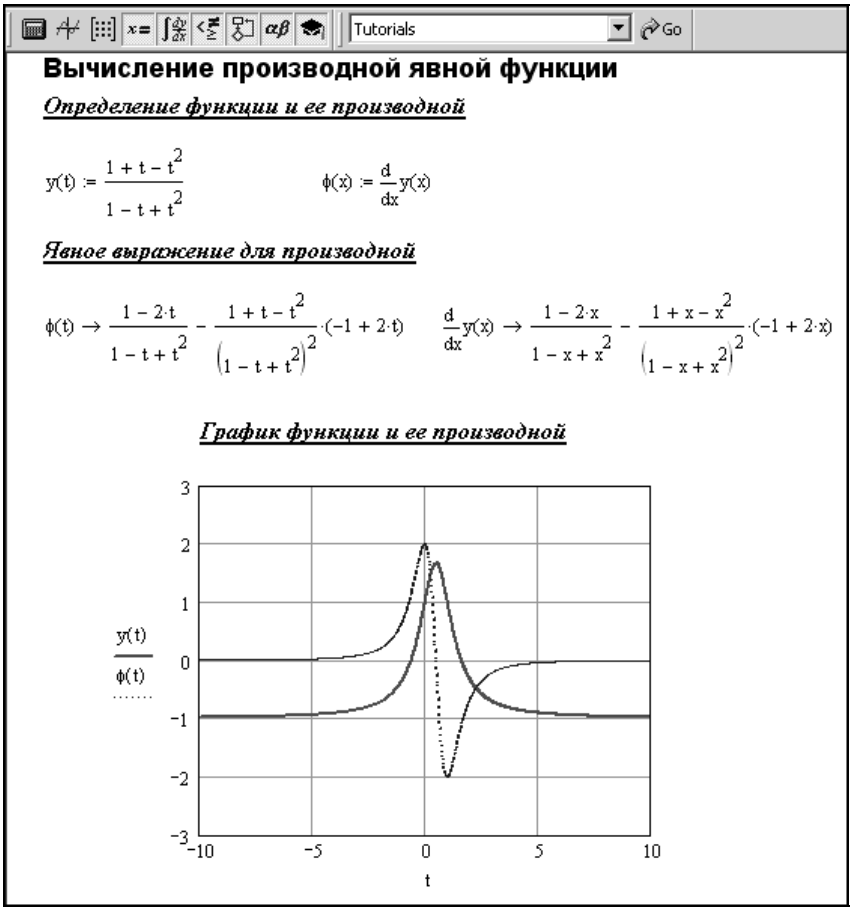


Рис. 6.1. Вычисление производной явной функции

В *главе 5* в качестве примеров уже рассматривались явные функциональные зависимости, поэтому решение задачи 6.1 особых комментариев не требует. Тем не менее, далее приведены основные команды.

Команда, с помощью которой определяется функция для дальнейшего дифференцирования:

$$y(t) := \frac{1 + t - t^2}{1 - t + t^2}$$

Команда определения производной от исходной функции:

$$\phi(x) := \frac{d}{dx} y(x)$$

Вычисление в явном виде выражения для производной. Для этого после названия функции-производной вводится оператор вычисления символьного значения:

$$\phi(t) \rightarrow \frac{1 - 2\cdot t}{1 - t + t^2} - \frac{1 + t - t^2}{\left(1 - t + t^2\right)^2}\cdot(-1 + 2\cdot t)$$

Вычисление в явном виде выражения для производной, но на этот раз оператор вычисления символьного значения вводится после команды дифференцирования. С точностью до обозначения аргумента функции оба выражения (это и предыдущее) совпадают (как и должно быть):

$$\frac{d}{dx} y(x) \rightarrow \frac{1 - 2\cdot x}{1 - x + x^2} - \frac{1 + x - x^2}{\left(1 - x + x^2\right)^2}\cdot(-1 + 2\cdot x)$$

Следует отметить тривиальный с математической точки зрения факт, что производная от функции сама является функцией. В Mathcad это правило остается в силе. Поэтому, например, можно построить график для функции и ее производной, что и было сделано в документе, фрагмент которого показан на рис. 6.1.

Производная от неявно заданной функции

К сожалению, представить функциональную зависимость в явном виде удается далеко не всегда. Но даже если явное выражение (или выражения) для вычисления значения функции по ее аргументу и известно, оно может быть весьма неудобным для практических расчетов. Часто встречается и имеет большое значение неявный способ определения функциональной зависимости. Обычно это делается с помощью уравнения, в которое входит сама функция

и ее аргумент, т. е. в уравнении, например, присутствует две переменных, одна из которых — независимая, а вторая принимает такие значения, чтобы уравнение превращалось в тождество. Таким образом определяется функция от одной переменной. В отличие от явного определения, здесь функция может быть неоднозначной, т. е. одному значению аргумента может соответствовать несколько значений функции.

Задача 6.2. Вычислить производную неявной функции $z(x)$, заданной равенством $x^2 + 2xz - z^2 - 2x = 0$.

Главное правило, которому следуют при вычислении производной от неявно заданной функции, состоит в том, что равенство, определяющее функциональную зависимость, дифференцируется по переменной-аргументу неявной функции. При этом подразумевается, что вторая переменная является функцией от той переменной, по которой вычисляется производная.

В рабочем документе, представленном на рис. 6.2, сначала определяется функция двух переменных, с помощью которой задается неявная функциональная зависимость. Далее эта функция дифференцируется по независимой переменной (первому аргументу), причем второй аргумент указан как функция первого. Полученное в результате выражение должно тождественно равняться нулю. В общем случае выражение содержит независимую переменную (аргумент неявной функции), зависимую переменную (неявную функцию) и производную от неявной функции по ее аргументу. Это уравнение вместе с исходным уравнением, определяющим неявную функциональную зависимость, образует систему, из которой может быть определена как функция, так и ее производная.

Далее приведены команды, использованные при вычислении производной от неявной функции.

Команда определения функции двух переменных. Эта функция, будучи приравненная к нулю, формирует уравнение, с помощью которого задается неявная функциональная зависимость:

$$F(x,z) := x^2 + 2\cdot x\cdot z - z^2 - 2\cdot x$$

Вычисление производной от левой части равенства, определяющего неявную функциональную зависимость. Левая часть уравнения ранее была задана в виде функции двух переменных. Первая переменная, по которой вычисляется производная, полагается независимой и является аргументом неявной функции. Вторым аргументом указано название неявной функции, и в скобках приведен ее аргумент:

$$\frac{d}{dx}F(x,z(x)) \rightarrow 2\cdot x + 2\cdot z(x) + 2\cdot x\cdot\frac{d}{dx}z(x) - 2\cdot z(x)\cdot\frac{d}{dx}z(x) - 2$$

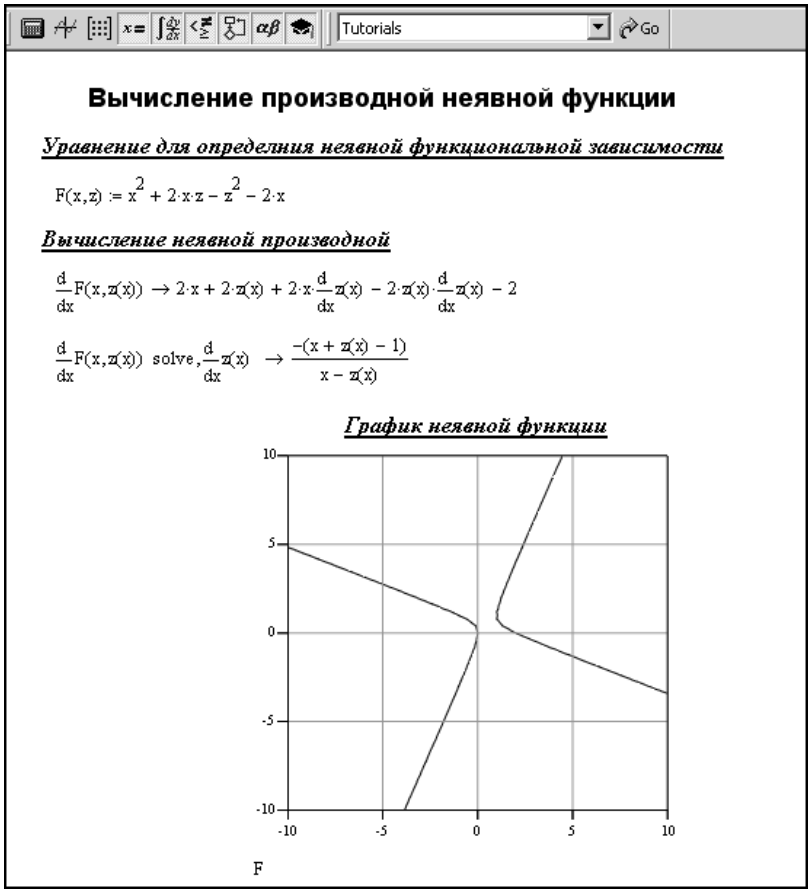


Рис. 6.2. Производная от неявной функции

С помощью инструкции `solve` уравнение (его левая часть), полученное в результате дифференцирования исходного равенства для неявной зависимости, решается относительно производной неявной функции. Из исходного уравнения по известному аргументу неявной функции определяется ее значение. Подставив эти результаты в полученное выражение для производной неявной функции, можно найти и ее значение (при данном аргументе неявной функции):

$$\frac{d}{dx}F(x, z(x))\ \text{solve}, \frac{d}{dx}z(x) \ \rightarrow \frac{-(x + z(x) - 1)}{x - z(x)}$$

В нижней части рис. 6.2 отображен график функции, неявно заданной с помощью исходного уравнения в условии задачи. Процедура его создания в данном случае имеет свои особенности. В частности, сначала строился контурный график для функции двух переменных $F(x, y)$ (левая часть равенст-

ва, задающего неявную функцию). Очевидно, что контур, соответствующий нулевому значению функции $F(x, y)$, совпадает с искомой неявной функциональной зависимостью. Таким образом, среди всех контурных линий на графике следует оставить только ту, что соответствует нулевому значению функции. Для этого на вкладке **Special** диалогового окна **3-D Plot Format** следует установить в поле **Number** значение 1. При этом отображается только одна контурная линия. Чтобы убедиться, что эта линия соответствует нулевому значению, достаточно на той же вкладке установить флажок опции **Numbered**. Остальные настройки не так существенны и обсуждались в *главе 3*.

Параметрическая производная

Еще один достаточно популярный способ определения функциональной зависимости подразумевает наличие, помимо функции и ее аргумента, еще и третьего параметра. В явном виде задаются зависимости аргумента функции от этого параметра и функции от параметра. Таким образом, каждому значению параметра из области его возможных значений соответствуют значения аргумента функции и самой функции. Соотношение между аргументом и функцией, устанавливаемое по такой схеме, называется параметрической зависимостью. Наряду с определением функции в неявном виде, параметрический способ реализации функциональной зависимости широко применяется при работе с неоднозначными функциями. В задаче 6.3 рассматривается вычисление производной от функции, заданной в параметрическом виде.

Задача 6.3. Найти производную функции $y(x)$, заданной с помощью соотношений $x(t) = \sin(t)$ и $y(t) = \cos^4(t)$, где параметр t принимает значения из диапазона $[-\pi/2, \pi/2]$.

При дифференцировании неявно заданной функции, производная также определялась в неявном виде, поскольку выражение для производной содержало, помимо аргумента функции, еще и саму неявную функцию. Несложно догадаться, что при дифференцировании функции, заданной параметрически, производная также вычисляется в параметрическом виде. Как отмечалось, параметрическая функциональная зависимость дается системой из двух уравнений: зависимостью аргумента функции от параметра и зависимостью функции от параметра. Для производной зависимость аргумента от параметра не меняется, а вот зависимость функции-производной от параметра определяется через отношение производных по параметру от исходной функции и аргумента. Другими словами, если параметрическая зависимость задана равенствами $x = x(t)$ и $y = y(t)$, то производная y'_x задается равенствами $x = x(t)$ и $y'_x(t) = y'_t(t)/x'_t(t)$. В дальнейшем, чтобы избежать путаницы,

условимся переменную x называть аргументом функции y, а переменную t — параметром, с помощью которого задана зависимость функции от аргумента.

Фрагмент рабочего документа на рис. 6.3 содержит набор команд для вычисления производной функции, заданной в параметрическом виде. Основной момент здесь — вычисление производных по параметру от аргумента функции и самой функции, поскольку далее ищется отношение соответствующих выражений. Все это реализуется с помощью создаваемой в документе функции трех аргументов: первый и второй аргументы — названия (без указания параметра!) аргумента функции и функции, а третий аргумент — название параметра. После создания такой функции она используется для вычисления выражения, определяющего зависимость функции-производной от параметра.

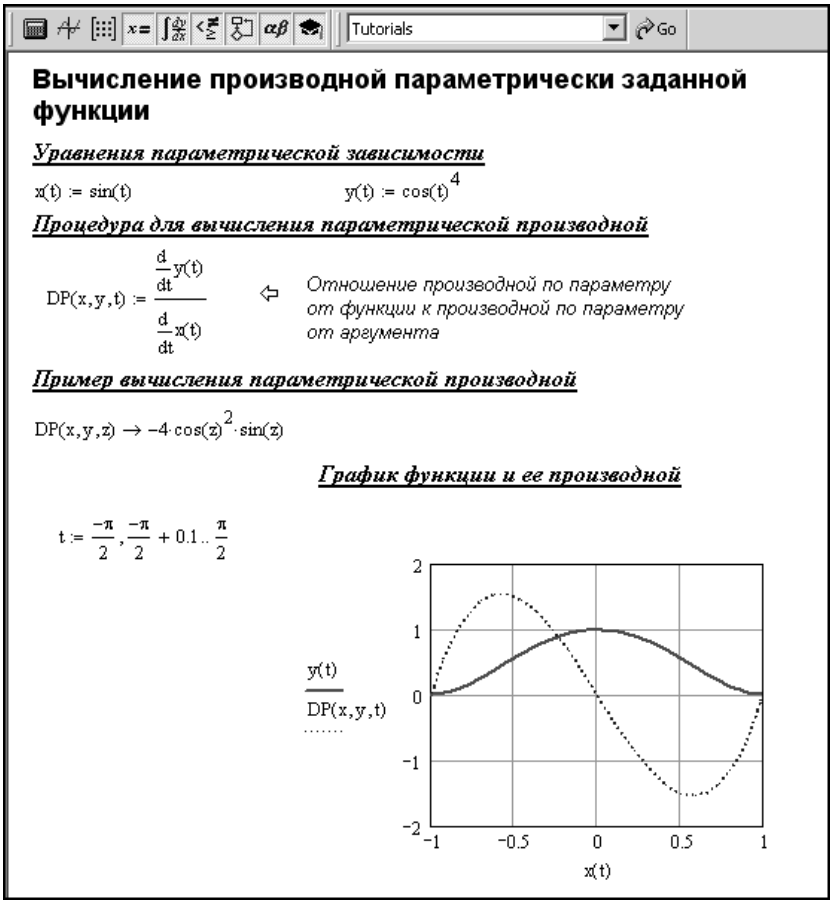


Рис. 6.3. Производная от параметрически заданной функции

Далее приведено более подробное описание назначения каждой из команд рабочего документа, в котором вычисляется параметрическая производная.

Командой задается функциональная зависимость аргумента, по которому вычисляется производная, от параметра:

$$x(t) := \sin(t)$$

Зависимость от параметра функции, для которой вычисляется производная:

$$y(t) := \cos(t)^4$$

Функция трех аргументов. Первые два аргумента — названия функций, которые задают параметрическую зависимость дифференцируемой функции от ее аргумента. Третий аргумент есть не что иное, как название переменной-параметра:

$$DP(x, y, t) := \frac{\frac{d}{dt}y(t)}{\frac{d}{dt}x(t)}$$

Пример вызова созданной ранее функции для вычисления параметрической производной:

$$DP(x, y, z) \rightarrow -4\cdot\cos(z)^2\cdot\sin(z)$$

Команда задает диапазон изменения переменной-параметра и шаг дискретности для построения графика функции и ее производной:

$$t := \frac{-\pi}{2}, \frac{-\pi}{2} + 0.1 .. \frac{\pi}{2}$$

В нижней части рис. 6.3 построены графики заданной в неявном виде дифференцируемой функции и ее производной. Для этого в явном виде задается диапазон изменения переменной-параметра, а вместо структурного заполнителя у горизонтальной координатной оси графика, где обычно указывается аргумент, вводится название функции (с указанием аргумента), определяющей зависимость аргумента от параметра. Возле другой оси можно видеть выражения, определяющие зависимость исходной функции и ее производной от параметра.

Производная функции, заданной в полярных координатах

В некоторых задачах функции заданы в полярных координатах. Обычно это делается, когда соответствующая зависимость, выраженная через полярные координаты, имеет более простой вид. Как известно, в полярной системе координат положение точки на плоскости задается двумя параметрами: рас-

стоянием r от точки начала координат до точки на плоскости и углом φ между направлением на точку и горизонтальной (декартовой) координатной осью. Связь между декартовыми координатами (x, y) точки и ее полярными координатами (r, φ) дается соотношениями $x = r\cos\varphi$ и $y = r\sin\varphi$. В этом случае к двум равенствам, связывающим полярные и декартовы координаты точки, следует добавить также и соответствующее равенство для упомянутой функциональной зависимости. Если, например, известна зависимость радиальной координаты r от угловой φ для точек некоторой кривой, то эту кривую вполне можно считать заданной в параметрическом виде. При этом независимая полярная координата играет роль параметра. В качестве иллюстрации рассмотрим следующую задачу.

Задача 6.4. Вычислить производную y'_x от функции, заданной в полярной системе координат уравнением $r(\theta) = \sin^2\theta$.

На рис. 6.4 приведен фрагмент рабочего документа, состоящего из двух частей. В первой определяется в полярной системе координат функция, и после этого задаются уравнения перехода от полярных координат к декартовым. Таким образом, с учетом функциональной зависимости, декартовы координаты точек на соответствующей кривой являются функциями одного параметра, в данном случае полярного угла.

Далее, во второй части документа, согласно общему правилу вычисления параметрической производной, задается функция, равная отношению производных по полярному углу от y- и x-координаты. При этом первый аргумент — исследуемая функциональная зависимость. Другими словами, в явном виде указано, что радиальная полярная координата является функцией полярного угла. После этого значение производной (которая будет, как и исходная функция, представлена через полярные координаты) можно вычислить, что и делается двумя последующими командами. Далее приведены команды рассмотренного рабочего документа.

Командой задается функция в полярной системе координат. Независимая переменная — полярный угол, а радиальная координата — функция от этого угла:

$$r(\theta) := \sin(\theta)^2$$

Декартовая координата точки (абсцисса) определяется как функция полярных координат. Соответствующее уравнение устанавливает связь между полярными и декартовыми координатами:

$$x(r, \theta) := r\cdot\cos(\theta)$$

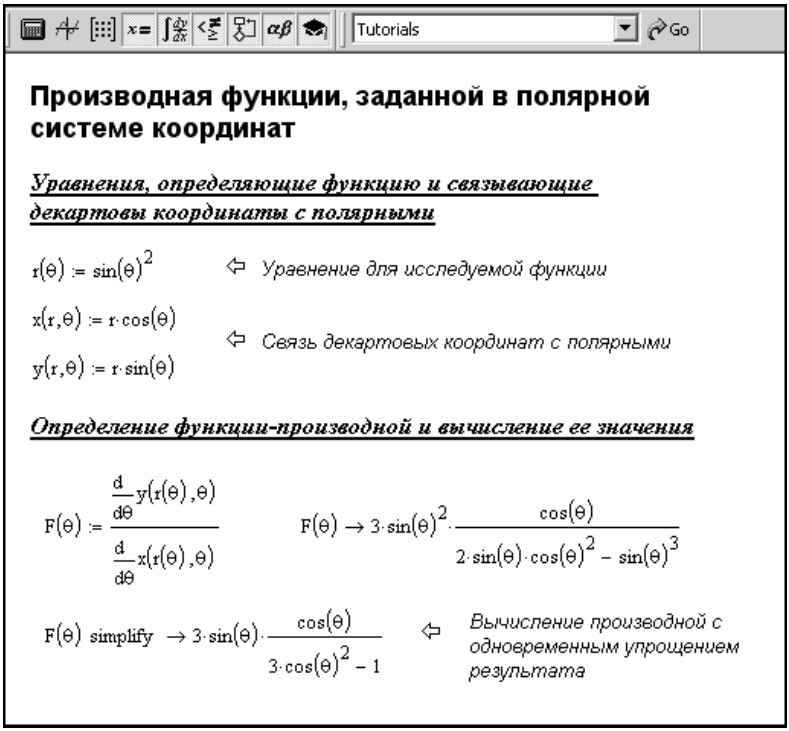


Рис. 6.4. Производная от функции в полярной системе координат

Ордината точки определяется как функция от полярных координат. Эта и предыдущая функции обеспечивают переход от полярных координат к декартовым:

$$y(r,\theta) := r\cdot\sin(\theta)$$

Команда определения параметрической производной. Эта производная реализуется в виде функции, зависящей от одного аргумента — полярного угла, который играет роль параметра при параметрическом описании функции. При вычислении производных по этому параметру следует иметь в виду, что радиальная полярная координата — функция угла, следовательно, после названия радиальной переменной в явном виде указан аргумент. Радиальная переменная, в свою очередь, является аргументом в функциях перехода от полярных координат к декартовым:

$$F(\theta) := \frac{\frac{d}{d\theta}y(r(\theta),\theta)}{\frac{d}{d\theta}x(r(\theta),\theta)}$$

Команда вычисления в символьном виде параметрической производной. Значение вычисляется корректно, однако выглядит несколько громоздко:

$$F(\theta) \rightarrow 3\cdot\sin(\theta)^2\cdot\frac{\cos(\theta)}{2\cdot\sin(\theta)\cdot\cos(\theta)^2 - \sin(\theta)^3}$$

Вычисление параметрической производной в символьном виде с одновременным упрощением результата. По сравнению с предыдущим конечное выражение выглядит компактнее:

$$F(\theta)\ \text{simplify}\ \rightarrow 3\cdot\sin(\theta)\cdot\frac{\cos(\theta)}{3\cdot\cos(\theta)^2 - 1}$$

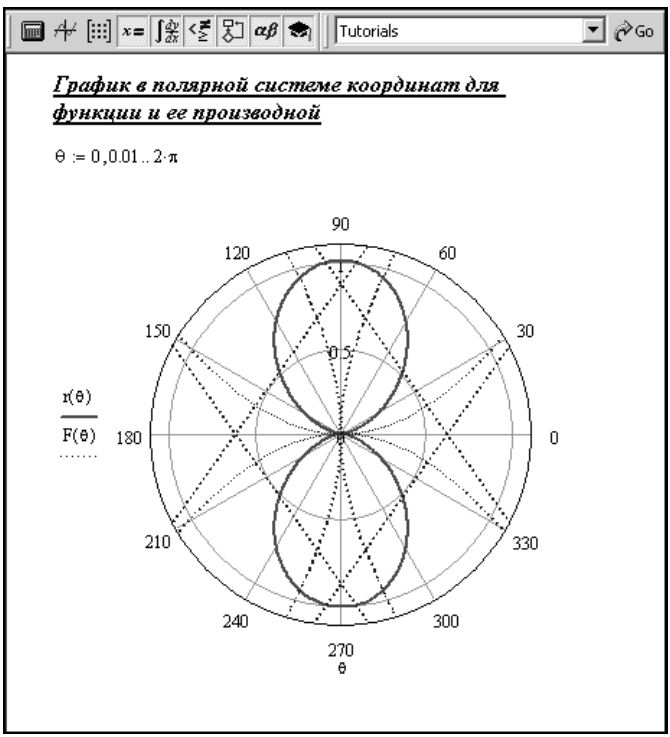


Рис. 6.5. График функции и ее производной в полярной системе координат

Несмотря на то, что уравнение для исходной функции и ее производной может быть получено в полярных координатах в явном виде, этого не всегда достаточно для получения четкого представления о функции и ее производной. Наиболее наглядным, как правило, является графический способ представления данных. Поэтому разумно построить графики функции и ее произ-

водной, причем в полярной системе координат. Для этого следует выбрать команду **Insert | Graph | Polar Plot** и выполнить ряд простых настроек. График функции и ее производной в полярной системе координат представлен на рис. 6.5.

В частности, отдельной командой указывается диапазон и дискретность изменения полярного угла. Вместо структурного заполнителя в нижней части области создания полярного графика вводится имя полярной переменной, а слева последовательно указываются названия функции и ее производной. Функции записываются вместе с аргументом.

Производные высоких порядков

Если требуется вычислить производную более высокого порядка от функции, заданной в параметрическом виде, поступают следующим образом. Сначала вычисляют производную первого порядка, которая, как и исходная функция, задана в параметрическом виде. Соответствующие зависимости (аргумента и производной от параметра) являются исходными для вычисления второй производной и т. д. Другими словами, параметрическая производная второго порядка определяется системой зависимостей $x = x(t)$ и $y''_{xx}(t) = \dfrac{dy'_x(t)}{dt} \Big/ \dfrac{dx(t)}{dt}$.

Задача 6.5. Вычислить третью производную y'''_{xxx} от функции $x(t) = \cos(t)$ и $y(t) = \sin^4(t)$.

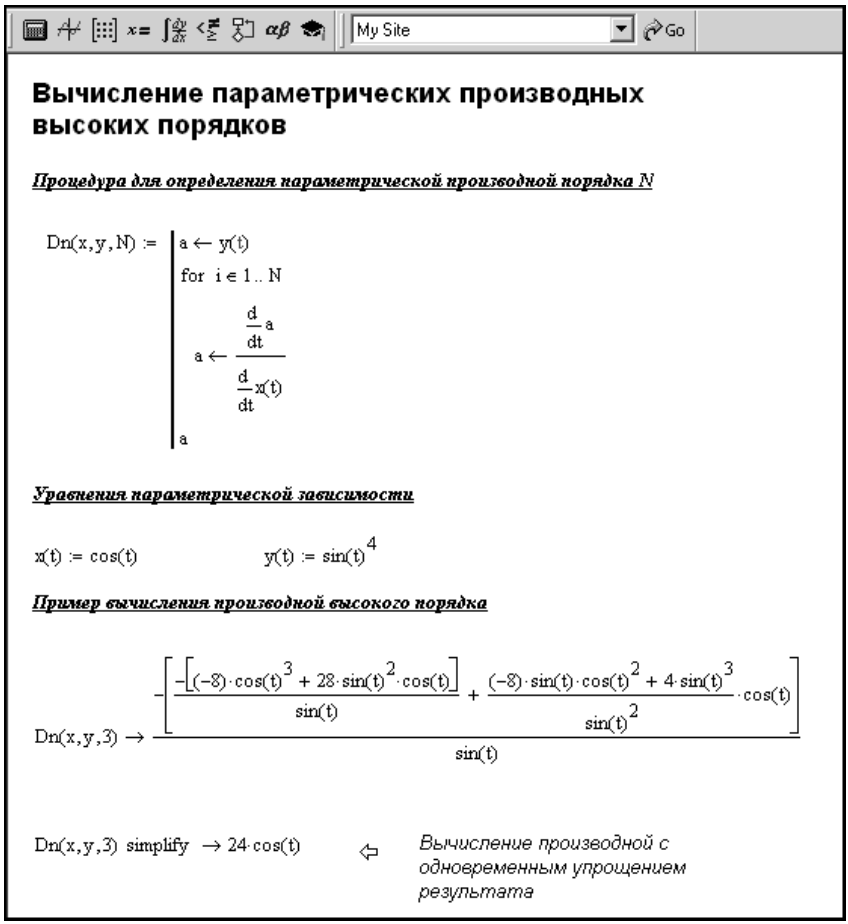
Для вычисления параметрической производной высокого порядка определяется специальная процедура, имеющая три аргумента: переменная-аргумент `x`, переменная-функция `y` и порядок производной `N`. Код процедуры представлен в документе на рис. 6.6.

Первой командой в теле процедуры инициализируется переменная `a`, которой в качестве значения присваивается функция (второй аргумент) от параметра (он должен иметь название `t`), т. е. `y(t)`. Далее в теле оператора цикла эта переменная переопределяется как отношение производной по параметру `t` от предыдущего значения производной по `t` от аргумента параметрически заданной функции. Именно переменная `a` возвращается как результат вычисления процедуры. Далее приведены команды рабочего документа.

Процедура вычисления параметрической производной высокого порядка. Сначала инициализируется локальная переменная `a` со значением `y(t)`. Далее размещен оператор цикла, в котором переменная-счетчик пробегает значения от `1` до `N`, где `N` — порядок производной. Новое значение переменной `a` присваивается как частное от деления производной от `a` по `t` на производную

от `x(t)` по `t`. Тем самым реализуется последовательность вычисления производной в параметрическом виде:

$$Dn(x,y,N) := \left| \begin{array}{l} a \leftarrow y(t) \\ \text{for } i \in 1..N \\ \quad a \leftarrow \dfrac{\dfrac{d}{dt}a}{\dfrac{d}{dt}x(t)} \\ a \end{array} \right.$$



Вычисление параметрических производных высоких порядков

Процедура для определения параметрической производной порядка N

$$Dn(x,y,N) := \left| \begin{array}{l} a \leftarrow y(t) \\ \text{for } i \in 1..N \\ \quad a \leftarrow \dfrac{\dfrac{d}{dt}a}{\dfrac{d}{dt}x(t)} \\ a \end{array} \right.$$

Уравнения параметрической зависимости

$x(t) := \cos(t)$ $\qquad$ $y(t) := \sin(t)^4$

Пример вычисления производной высокого порядка

$$Dn(x,y,3) \rightarrow \frac{-\left[\dfrac{-\left[(-8)\cdot\cos(t)^3 + 28\cdot\sin(t)^2\cdot\cos(t)\right]}{\sin(t)} + \dfrac{(-8)\cdot\sin(t)\cdot\cos(t)^2 + 4\cdot\sin(t)^3}{\sin(t)^2}\cdot\cos(t)\right]}{\sin(t)}$$

$Dn(x,y,3)$ simplify $\rightarrow 24\cdot\cos(t)$ ⇦ *Вычисление производной с одновременным упрощением результата*

Рис. 6.6. Параметрические производные высоких порядков

Параметрическая зависимость `x` от параметра `t`:

$$x(t) := \cos(t)$$

Параметрическая зависимость `y` от параметра `t`:

$$y(t) := \sin(t)^4$$

Пример вычисления производной третьего порядка с помощью созданной ранее процедуры. Результат довольно громоздок:

$$Dn(x,y,3) \rightarrow \frac{-\left[\frac{-\left[(-8)\cdot\cos(t)^3 + 28\cdot\sin(t)^2\cdot\cos(t)\right]}{\sin(t)} + \frac{(-8)\cdot\sin(t)\cdot\cos(t)^2 + 4\cdot\sin(t)^3}{\sin(t)^2}\cdot\cos(t)\right]}{\sin(t)}$$

Пример вызова созданной функции (аргументы те же, что и выше) с одновременным упрощением результата:

$$Dn(x,y,3) \text{ simplify } \rightarrow 24\cdot\cos(t)$$

Там же на рис. 6.6 приведен пример вычисления производной третьего порядка с помощью созданной процедуры. Для однозначного определения зависимости соответствующей функции от аргумента необходимо дополнить полученный результат еще и зависимостью аргумента от параметра.

Вычисление интегралов

О вычислении интегралов в символьном виде кратко шла речь в *главе 5*. Там, правда, рассматривались достаточно простые задачи. Простые в том смысле, что для их решения достаточно прямого применения встроенных утилит Mathcad. Здесь будут описаны более замысловатые задачи, хотя и в них ничего особо сложного нет. Большинство приведенных задач решается в несколько этапов, каждый из которых имеет свои особенности.

Пример 6.1. Определенные и неопределенные интегралы

Для начала напомним, как в символьном виде вычисляются неопределенные и определенные интегралы. В документе на рис. 6.7 приведены соответствующие примеры.

Сначала определяется функция, которая затем и интегрируется. Первой командой берется в символьном виде неопределенный интеграл от этой функции. После этого от той же функции вычисляется интеграл в указанных пределах. Наконец, через определенный интеграл определяется функциональная зависимость `F(z)`. В подынтегральном выражении, помимо переменной интегрирования, присутствует еще переменная-параметр. В этом случае

говорят об интеграле с параметром. В Mathcad это вполне легальный и эффективный способ определения функции. Функция `F(z)` ничем не отличается от обычной. В частности, можно построить ее график (см. рис. 6.7). Далее приведены основные команды рабочего документа.

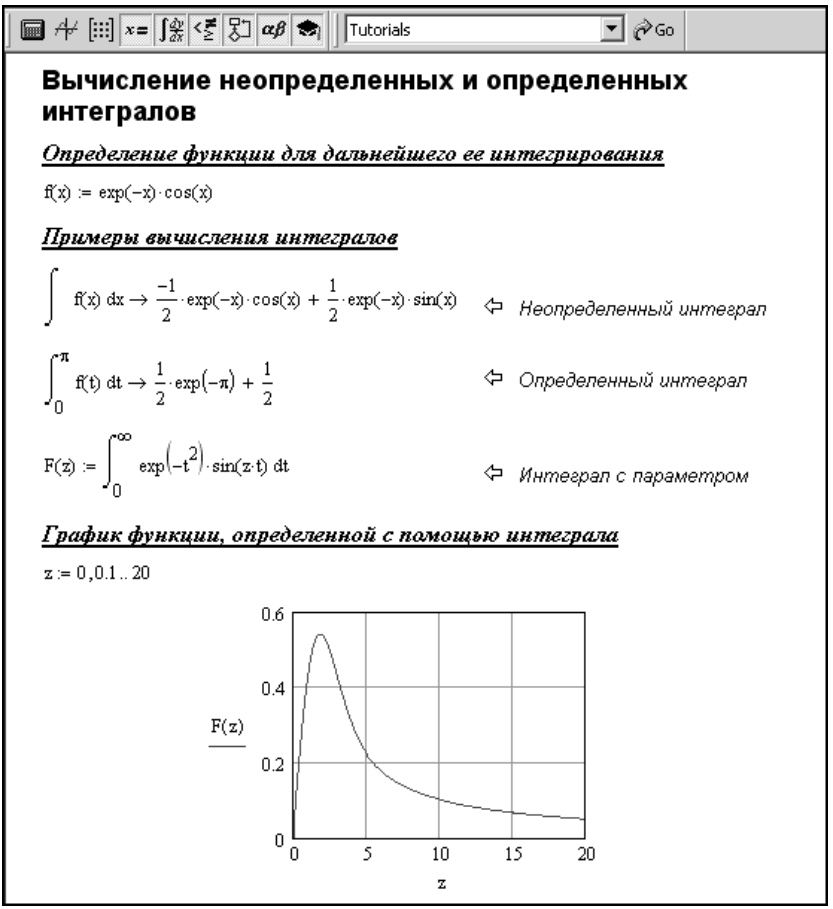


Рис. 6.7. Вычисление неопределенных и определенных интегралов

Инициализация функции для дальнейшего интегрирования:

$$f(x) := \exp(-x)\cdot\cos(x)$$

Вычисление неопределенного интеграла в символьном виде:

$$\int f(x)\,dx \rightarrow \frac{-1}{2}\cdot e^{-x}\cdot\cos(x) + \frac{1}{2}\cdot e^{-x}\cdot\sin(x)$$

Вычисление определенного интеграла в символьном виде:

$$\int_0^{\pi} f(t)\,dt \rightarrow \frac{1}{2}\cdot e^{-\pi} + \frac{1}{2}$$

Инициализация функции через интеграл с параметром. Параметр интеграла является аргументом функции:

$$F(z) := \int_0^{\infty} \exp\left(-t^2\right)\cdot \sin(z\cdot t)\,dt$$

Далее рассмотрим более сложные задачи, связанные с вычислением различных интегралов.

Криволинейные интегралы

Криволинейный интеграл от функции нескольких переменных — интеграл, вычисленный вдоль определенного контура, называемого контуром интегрирования. Существуют криволинейные интегралы первого и второго рода. В зависимости от типа интеграла различны и способы его вычисления. В любом случае задача по вычислению криволинейного интеграла, как правило, подразумевает параметризацию контура интегрирования. Правда, очень часто этот контур сразу задан в параметрическом виде.

Вычисление криволинейного интеграла первого рода по контуру C, заданному в параметрическом виде с помощью соотношений $x = x(t)$, $y = y(t)$ и $z = z(t)$ от функции $f(x, y, z)$, сводится к расчету следующего выражения:

$$\int_C f(x, y, z)dl = \int_{\alpha}^{\beta} f(x(t), y(t), z(t))\sqrt{x_t'^2 + y_t'^2 + z_t'^2}\,dt,$$

где α и β — пределы изменения параметра t. Если интерпретировать функцию $f(x, y, z)$ как линейную плотность кривой вдоль контура интегрирования, то результатом вычисления криволинейного интеграла первого рода от этой функции по указанному контуру будет масса соответствующей кривой. Значение криволинейного интеграла первого рода не зависит от направления обхода кривой при интегрировании.

Задача 6.6. Вычислить криволинейный интеграл первого рода $\int_C y^2 dl$ по контуру, заданному в параметрическом виде $x(t) = a(t - \sin(t))$ и $y(t) = a(1 - \cos(t))$, $0 \le t \le 2\pi$.

Часто контур интегрирования лежит в одной плоскости. Тогда ситуация несколько упрощается. Например, если контур интегрирования размещен в плоскости XY, то криволинейный интеграл вычисляется согласно соотношению $\int\limits_C f(x,y)dl = \int\limits_\alpha^\beta f(x(t),y(t))\sqrt{x_t'^2 + y_t'^2}\,dt$. Именно эта формула и использована для вычисления интеграла в документе на рис. 6.8.

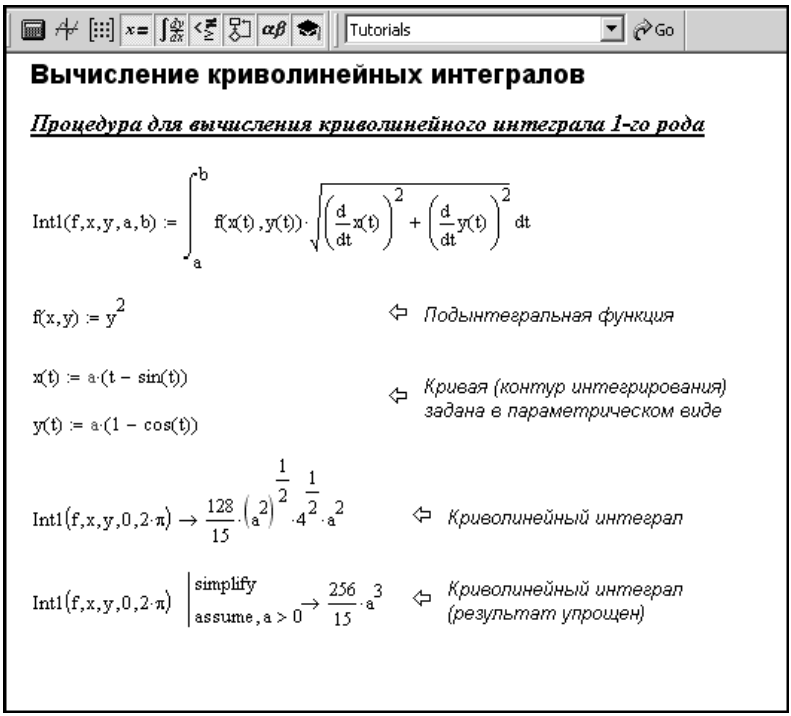


Рис. 6.8. Вычисление криволинейных интегралов первого рода

Определяется специальная процедура, с помощью которой по ее аргументам (а это названия подынтегральной функции и функций, определяющих зависимость от переменной-параметра горизонтальной и вертикальной координат вдоль контура интегрирования, а также нижняя и верхняя границы изменения переменной-параметра) вычисляется криволинейный интеграл первого рода. Процедура определена через интеграл. Два последних ее аргумента указаны нижним и верхним пределами интегрирования. У подынтегральной функции аргументы сами являются функциями переменной интегрирования. Кроме того, эта подынтегральная функция умножается на корень квадратный из суммы квадратов производных от функций-аргументов подынтегральной

функции по переменной интегрирования. Далее на рис. 6.8 для проверки корректности работы описанной процедуры инициализируются три функции: подынтегральная функция (та, от которой вычисляется криволинейный интеграл), а также параметрические зависимости, определяющие контур интегрирования. В данном случае контуром интегрирования является циклоида ($x(t) = a(t - \sin(t))$, $y(t) = a(1 - \cos(t))$, $0 \le t \le 2\pi$). После этого дважды вызывается созданная процедура с одними и теми же аргументами. Во втором случае результат автоматически упрощается, если считать неотрицательным символьный параметр a, присутствующий в параметрических зависимостях. Далее приведены команды рабочего документа.

Процедура вычисления криволинейного интеграла первого рода:

$$\mathrm{Int1}(f,x,y,a,b) := \int_a^b f(x(t),y(t))\cdot\sqrt{\left(\frac{d}{dt}x(t)\right)^2 + \left(\frac{d}{dt}y(t)\right)^2}\,dt$$

Инициализация подынтегральной функции:

$$f(x,y) := y^2$$

Параметрическая зависимость x от параметра t:

$$x(t) := a\cdot(t - \sin(t))$$

Параметрическая зависимость y от параметра t:

$$y(t) := a\cdot(1 - \cos(t))$$

Использование процедуры для вычисления криволинейного интеграла первого рода. Результат достаточно громоздкий:

$$\mathrm{Int1}(f,x,y,0,2\cdot\pi) \rightarrow \frac{128}{15}\cdot\left(a^2\right)^{\frac{1}{2}}\cdot 4^{\frac{1}{2}}\cdot a^2$$

Результат вызова процедуры расчета интеграла первого рода после упрощения:

$$\mathrm{Int1}(f,x,y,0,2\cdot\pi)\ \left|\begin{array}{l}\text{simplify}\\ \text{assume}, a > 0\end{array}\right. \rightarrow \frac{256}{15}\cdot a^3$$

Несколько иначе вычисляется криволинейный интеграл второго рода. Его интерпретация может быть такой: это работа, выполненная некоторой силой (векторная подынтегральная функция) вдоль пространственной кривой (контур интегрирования). Отсюда очевидно, что при замене направления обхода контура значение интеграла должно менять знак на противоположный.

Если нас интересует элементарная работа δA, выполненная силой $\vec{F}$ на элементарном перемещении $d\vec{r}$, то это, очевидно, есть скалярное произведение силы на перемещение: $\delta A = \vec{F}d\vec{r}$. Чтобы вычислить полную работу, нужно проинтегрировать по контуру. Предположим, что $\vec{F} = (P, Q, R)$, где $P(x, y, z)$, $Q(x, y, z)$ и $R(x, y, z)$ есть функции пространственных координат. Тогда криволинейный интеграл второго рода дается выражением $\int\limits_C \vec{F}d\vec{r} = \int\limits_C Pdx + Qdy + Rdz$. Если контур интегрирования C задан в параметрическом виде $x = x(t)$, $y = y(t)$ и $z = z(t)$, то легко получить $\int\limits_C Pdx + Qdy + Rdz = \int\limits_\alpha^\beta \left(Px'(t) + Qy'(t) + Rz'(t) \right) dt$, при этом, как и ранее α и β — пределы изменения параметра t, а во втором интеграле аргументами функций $P(x, y, z)$, $Q(x, y, z)$ и $R(x, y, z)$ указывают зависимости от параметра пространственных координат вдоль контура интегрирования.

Задача 6.7. Вычислить криволинейный интеграл второго рода $\int\limits_C (x + y)dx + (y - x)dy$ по контуру, заданному в параметрическом виде $x(t) = a\cos(t)$ и $y(t) = b\sin(t)$, $0 \le t \le 2\pi$.

В случае плоского контура интегрирования криволинейный интеграл второго рода может быть вычислен согласно соотношению $\int\limits_C Pdx + Qdy = \int\limits_\alpha^\beta \left(Px'(t) + Qy'(t) \right) dt$.

В верхней части документа, представленного на рис. 6.9, определена процедура, позволяющая вычислять криволинейные интегралы второго рода.

Далее приведены команды, содержащиеся в этом документе.

Процедура для вычисления криволинейного интеграла второго рода:

$$\mathrm{Int2}(P, Q, x, y, a, b) := \int_a^b \left(P(x(t), y(t)) \cdot \frac{d}{dt} x(t) + Q(x(t), y(t)) \cdot \frac{d}{dt} y(t) \right) dt$$

Инициализация первой подынтегральной функции:

$$P(x, y) := x + y$$

Инициализация второй подынтегральной функции:

$$Q(x, y) := y - x$$

Параметрическая зависимость для координаты вдоль оси абсцисс:

$$x(t) := a \cdot \cos(t)$$

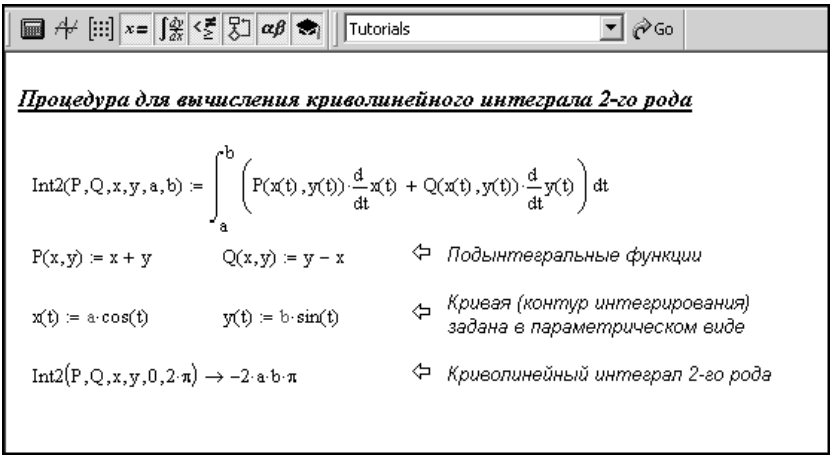


Рис. 6.9. Вычисление криволинейных интегралов второго рода

Параметрическая зависимость для координаты вдоль оси ординат:

$y(t) := b \cdot \sin(t)$

Использование созданной процедуры для вычисления криволинейного интеграла второго рода:

$\mathrm{Int2}(P, Q, x, y, 0, 2 \cdot \pi) \rightarrow -2 \cdot a \cdot b \cdot \pi$

Сначала в документе по соответствующим правилам описывается процедура, с помощью которой в дальнейшем будут вычисляться криволинейные интегралы второго рода. После этого инициализируются интегрируемые функции и параметрические зависимости, определяющие контур интегрирования. Остается проверить процедуру на эффективность. Этот тест она, очевидно, проходит.

Полезность криволинейных интегралов второго рода заключается еще и в том, что с их помощью можно вычислять площади фигур. Обычно площадь вычисляется с помощью двойного интеграла по области. Часто область задается ограничивающей ее кривой, которая записывается в параметрическом виде. Именно в такой ситуации для вычисления площадей оправдано применение криволинейных интегралов второго рода.

Задача 6.8. Вычислить с помощью криволинейного интеграла площадь фигуры, граница которой задана в параметрическом виде: $x(t) = a\sin^3(t)$ **и** $y(t) = b\cos^3(t)$**,** $0 \le t \le 2\pi$ **.**

Если плоская фигура ограничена некоторой кривой, то площадь этой фигуры может быть вычислена по одной из следующих формул: $S = \oint_C xdy = -\oint_C ydx = \frac{1}{2}\oint_C xdy - ydx$, где все указанные интегралы вычисля-

ются по замкнутому контуру — границе области, в таком направлении обхода, чтобы область оставалась слева. В частности, с учетом параметрического определения контура интегрирования, можем записать $S = \int_{\alpha}^{\beta} x(t)y'(t)dt$. Этой формулой и воспользуемся. На рис. 6.10 приведен пример документа, в котором площадь области рассчитывается с помощью криволинейного интеграла.

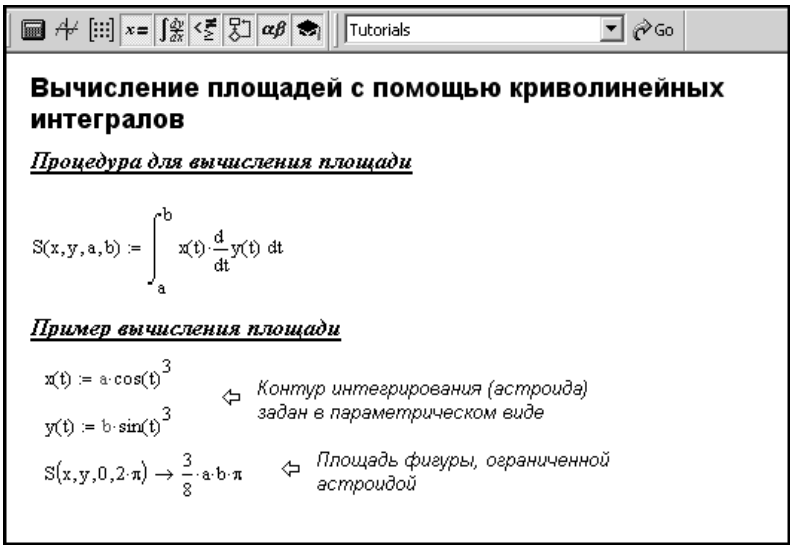


Рис. 6.10. Вычисление площади через контурный интеграл

Документ содержит процедуру вычисления площади фигуры через криволинейный интеграл второго рода, а также команды, задающие контур интегрирования и команду вычисления площади, которые приведены далее.

Процедура для вычисления площади фигуры через криволинейный интеграл второго рода:

$$S(x,y,a,b) := \int_a^b x(t)\cdot\frac{d}{dt}y(t)\,dt$$

Параметрическая зависимость, определяющая горизонтальные координаты точек контура интегрирования:

$$x(t) := a\cdot\cos(t)^3$$

Параметрическая зависимость, определяющая вертикальные координаты точек контура интегрирования:

$$y(t) := b \cdot \sin(t)^3$$

Проверка работы процедуры:

$$S(x, y, 0, 2 \cdot \pi) \rightarrow \frac{3}{8} \cdot a \cdot b \cdot \pi$$

Что касается самой фигуры, площадь которой была найдена, то представление о ней можно составить по рис. 6.11.

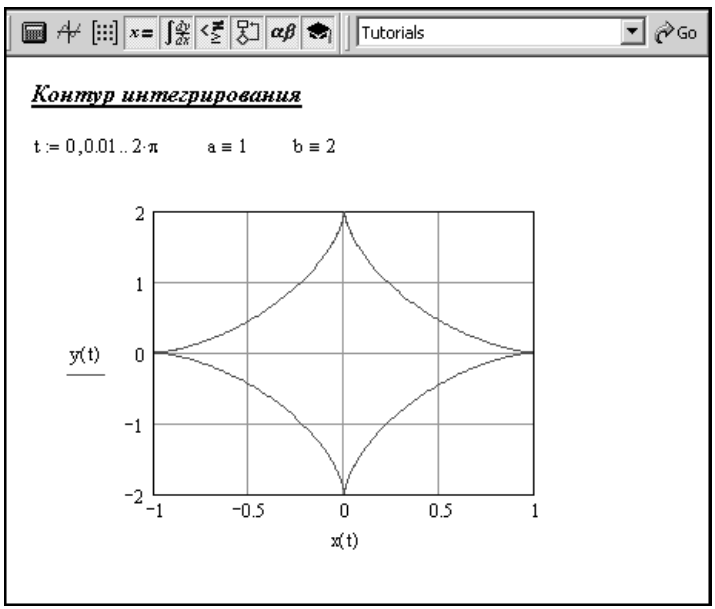


Рис. 6.11. Контур интегрирования

В качестве структурных заполнителей вдоль координатных осей на графике указаны параметрические зависимости, определяющие контур, ограничивающий область.

Полученные результаты легко обобщаются на случай пространственных кривых, когда при интегрировании следует учитывать третью координату. На рис. 6.12 представлены процедуры для вычисления криволинейных интегралов первого и второго рода по пространственным кривым.

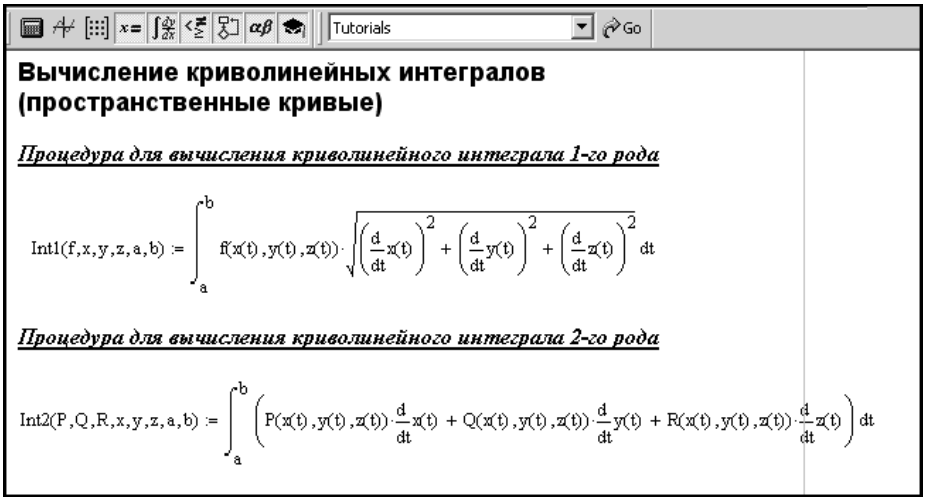


Рис. 6.12. Процедуры для вычисления криволинейных интегралов по пространственной кривой

Задача 6.9. Вычислить длину пространственной кривой, заданной в параметрическом виде $x(t) = \exp(-t)\cos(t)$, $y(t) = \exp(-t)\sin(t)$ **и** $z(t) = \exp(-t)$, $0 \le t < +\infty$.

Чтобы вычислить длину кривой с помощью криволинейного интеграла первого рода, следует, очевидно, в качестве подынтегральной функции указать единицу. Результат применения процедуры расчета интеграла первого рода по пространственной кривой для вычисления ее длины представлен на рис. 6.13.

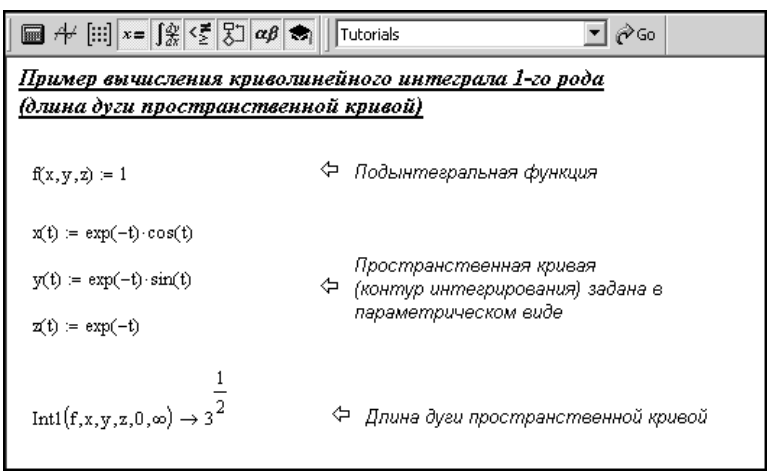


Рис. 6.13. Вычисление криволинейного интеграла первого рода по пространственной кривой

Далее приведены команды из этого рабочего документа.

Инициализация подынтегральной функции. При вычислении длины кривой эта функция должна тождественно равняться единице:

$f(x, y, z) := 1$

Параметрическая зависимость координаты *x* для рассматриваемой кривой:

$x(t) := \exp(-t)\cdot\cos(t)$

Параметрическая зависимость координаты *y* для рассматриваемой кривой:

$y(t) := \exp(-t)\cdot\sin(t)$

Параметрическая зависимость координаты *z* для рассматриваемой кривой:

$z(t) := \exp(-t)$

Вычисление криволинейного интеграла первого рода с помощью специальной, созданной ранее процедуры:

$\mathrm{Int1}(f, x, y, z, 0, \infty) \rightarrow 3^{\frac{1}{2}}$

Задача 6.10. Вычислить криволинейный интеграл второго рода $\int\limits_C ydx + zdy + xdz$ **вдоль контура, заданного в параметрическом виде:** $x(t) = \alpha\cos(t)$, $y(t) = \alpha\sin(t)$ **и** $z(t) = \beta t$, $0 \le t \le 2\pi$.

На рис. 6.14 представлен пример вычисления криволинейного интеграла второго рода по пространственной кривой.

Далее приведены команды из этого рабочего документа.

Первая подынтегральная функция:

$P(x, y, z) := y$

Вторая подынтегральная функция:

$Q(x, y, z) := z$

Третья подынтегральная функция:

$R(x, y, z) := x$

Первое уравнение параметрической зависимости:

$x(t) := \alpha\cdot\cos(t)$

Второе уравнение параметрической зависимости:

$y(t) := \alpha\cdot\sin(t)$

Третье уравнение параметрической зависимости:

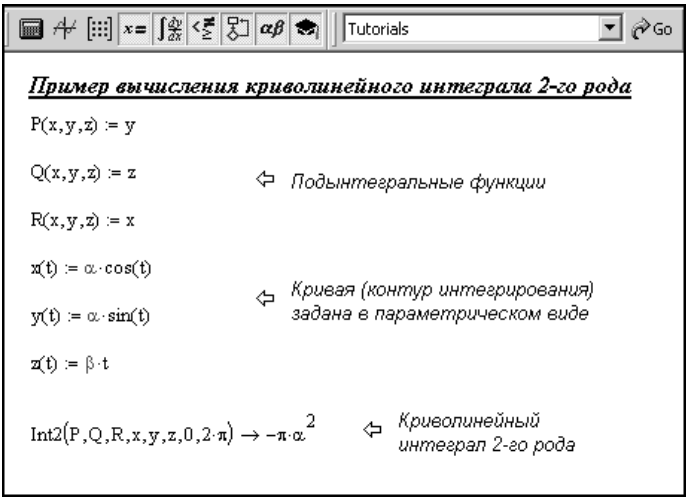


Рис. 6.14. Вычисление криволинейного интеграла второго рода по пространственной кривой

Вызов созданной ранее процедуры для вычисления криволинейного интеграла второго рода:

$$\mathrm{Int2}(P, Q, R, x, y, z, 0, 2\cdot\pi) \rightarrow -\pi\cdot\alpha^2$$

Фактически, по сравнению с плоскими контурами, при интегрировании по пространственной кривой мало что меняется — выражения становятся более громоздкими, но основная идея вычислений сохраняется.

Двойные интегралы

В общем случае двойной интеграл $\iint\limits_{\Omega} f(x,y)dxdy$ от функции $f(x,y)$ по области Ω определяется как число — предел двойной суммы произведений функции двух аргументов $f(x,y)$ на малые приращения аргументов при условии стремления этих приращений к нулю. Сумма берется по точкам области Ω, которая называется областью интегрирования. Обычно двойной интеграл вычисляется сведением к повторному интегралу: сначала вычисляется определенный интеграл по одной переменной (пределы интегрирования могут быть функциями другой переменной), а затем полученное выражение интегрируется по оставшейся переменной. В этом случае, например, может иметь место соотношение $\iint\limits_{\Omega} f(x,y)dxdy = \int\limits_{a}^{b} dx \int\limits_{\varphi(x)}^{\xi(x)} f(x,y)dy$.

Другой способ подразумевает замену переменных интегрирования. Если переход к новым переменным (u,v) осуществляется согласно соотношениям $x = x(u,v)$ и $y = y(u,v)$, то двойной интеграл в новых переменных может быть вычислен по формуле $\iint\limits_{\Omega} f(x,y)dxdy = \iint\limits_{\Omega'} f(x(u,v), y(u,v)) \,|\, J(u,v) \,|\, dudv$, где якобиан перехода $J(u,v)$ есть детерминант матрицы, составленной из частных производных от старых переменных по новым. Обычно новые переменные выбирают таким образом, чтобы упростить область интегрирования.

Задача 6.11. Вычислить двойной интеграл $\iint\limits_{\Omega} (x^2 + y^2)dxdy$ по области, ограниченной $y = a$, $y = 3a$, $y = x$ и $y = x + a$.

Решение этой задачи представлено в документе на рис. 6.15. Здесь всего три команды и график, на котором с помощью четырех прямых линий выделена область интегрирования.

Далее приведены команды из этого рабочего документа.

Подынтегральная функция:

$$f(x,y) := x^2 + y^2$$

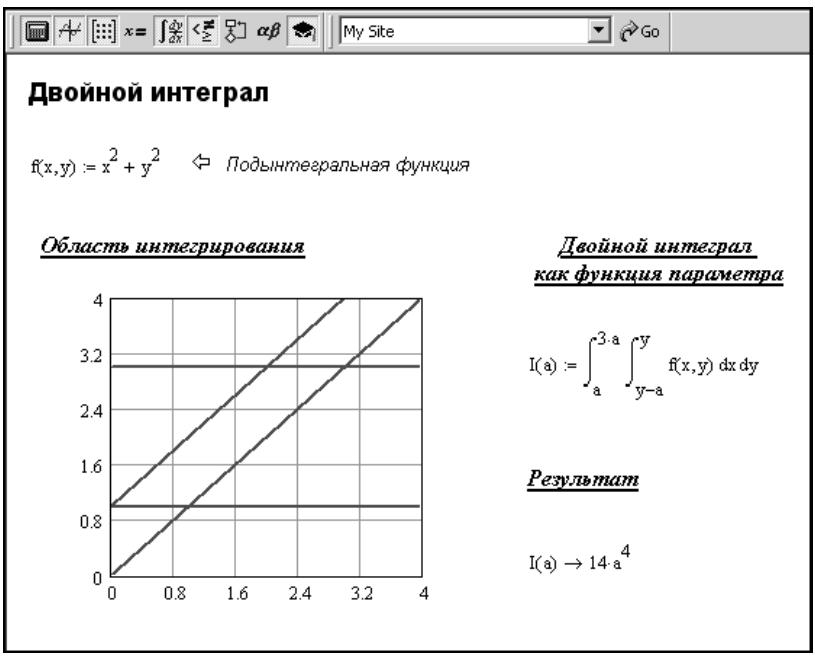


Рис. 6.15. Вычисление двойного интеграла

Двойной интеграл записан через повторный и задан как функция от параметра a, входящего в уравнения для границ области интегрирования:

$$I(a) := \int_{a}^{3\cdot a} \int_{y-a}^{y} f(x,y)\,dx\,dy$$

Команда вычисления в символьном виде двойного интеграла:

$$I(a) \rightarrow 14\cdot a^{4}$$

В данном случае, поскольку область интегрирования (точнее, ее границы) задана достаточно просто (уравнения несложные), двойной интеграл сразу может быть записан через повторный. Сначала выполняется интегрирование по переменной x, границы изменения которой определяются из двух последних уравнений в условии задачи и зависят от y. Интегрирование по y выполняется в пределах, определяемых первыми двумя уравнениями в условии задачи.

Задача 6.12. Вычислить площадь области, ограниченной кривой, заданной в параметрическом виде: $x(t) = a(t - \sin(t))$, $y(t) = a(1 - \cos(t))$, $0 \le t \le 2\pi$ **при условии** $y \ge 0$.

Для решения этой задачи, по сути, нужно перейти к новой переменной интегрирования, т. е. вместо x к t. Это частный случай общей замены переменных в двойном интеграле. Однако здесь можно будет обойтись без вычисления якобиана перехода. В частности, будем сначала интегрировать по y, а после этого — по t. Пределы интегрирования по y устанавливаются от нуля до значения, определяемого параметрической зависимостью $y(t)$ в условии задачи. Пределы интегрирования по параметру t — от 0 до 2π. Кроме того, поскольку переменная x — функция параметра, справедливо соотношение $dx = x'(t)dt$. Процесс вычисления двойного интеграла реализован в документе, представленном на рис. 6.16.

Чтобы избежать путаницы, при вычислении первого внутреннего интеграла название переменной интегрирования заменено на z (чтобы переменная интегрирования формально не совпадала с верхней границей интеграла). Далее приведены команды из этого документа.

Уравнения параметрической зависимости:

$$x(t) := a\cdot(t - \sin(t))$$

$$y(t) := a\cdot(1 - \cos(t))$$

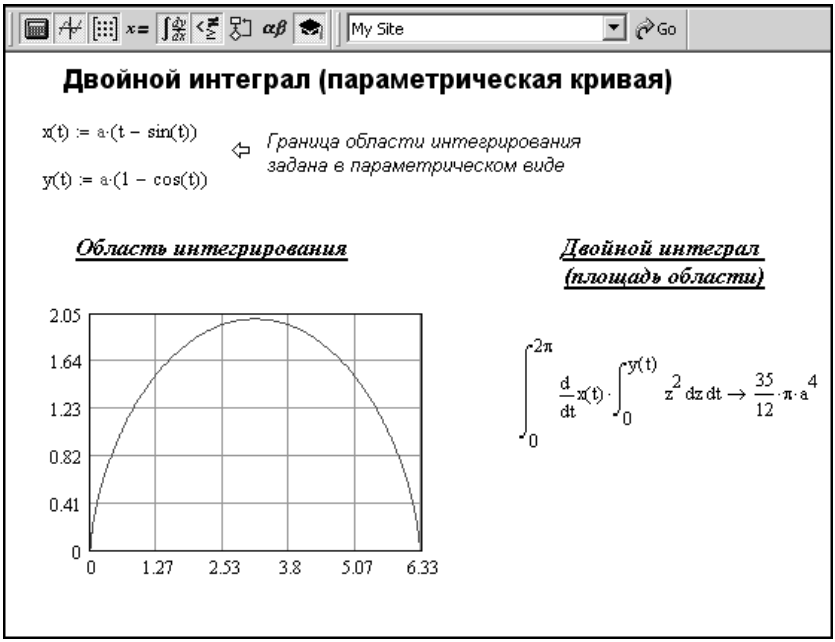


Рис. 6.16. Граница области интегрирования задана в параметрическом виде

Команда вычисления двойного интеграла через повторный. Во внутреннем интеграле заменена переменная интегрирования (чтобы ее название отличалось от названия функции на верхней границе внутреннего интеграла):

$$\int_0^{2\pi} \frac{d}{dt}x(t)\cdot\int_0^{y(t)} z^2\,dz\,dt \rightarrow \frac{35}{12}\cdot\pi\cdot a^4$$

Однако далеко не всегда двойной интеграл так просто можно свести к повторному. В качестве иллюстрации рассмотрим пример, в котором двойной интеграл вычисляется путем замены переменных.

Задача 6.13. Вычислить площадь области, ограниченной кривыми $xy = a^2$, $xy = 2a^2$, $y = x$ **и** $y = 2x$.

Область, площадь которой нужно вычислить, ограничена четырьмя кривыми (рис. 6.17).

Команды, с помощью которых реализуется решение этой задачи, представлены на рис. 6.18. В начале документа записаны уравнения, определяющие границу области интегрирования. В документе они нигде не используются, поэтому имеют скорее иллюстративный характер. Новые переменные u и v вводятся

согласно соотношениям $u^2 = xy$ и $v^2 = \dfrac{y}{x}$. При таком выборе новых координат область интегрирования будет прямоугольной. Другими словами, пределы интегрирования по каждой из переменных — константы, не зависящие от переменных интегрирования. По переменной u нужно интегрировать в пределах от a до $\sqrt{2}a$, а по v — от 1 до $\sqrt{2}$. Исходные соотношения определяют зависимость новых переменных от старых. Для вычисления якобиана перехода нужно решить соответствующие уравнения относительно старых переменных, т. е. выразить их в виде функций переменных u и v. Это и делается. Далее вычисляется якобиан перехода и непосредственно двойной интеграл, который в документе сразу записан через повторный интеграл — благо простота области интегрирования в новых переменных позволяет это легко сделать.

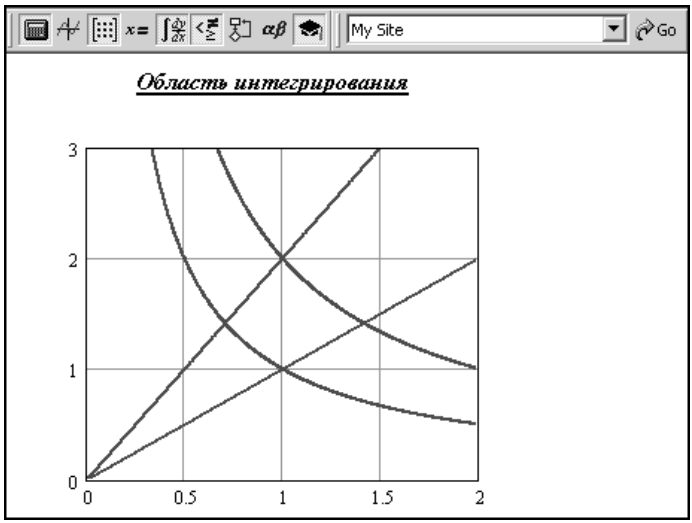


Рис. 6.17. Область интегрирования при вычислении двойного интеграла

Далее приведены некоторые комментарии к наиболее существенным командам рабочего документа на рис. 6.18.

Решение системы уравнений, определяющих способ перехода от старых переменных интегрирования к новым. Найдено два решения. Выбираем то решение, которое соответствует положительным значениям переменных:

$$\begin{pmatrix} u^2 = x\cdot y \\ v^2 = \dfrac{y}{x} \end{pmatrix} \text{ solve}, \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} \dfrac{u}{v} & v\cdot u \\ \dfrac{-u}{v} & -v\cdot u \end{pmatrix}$$

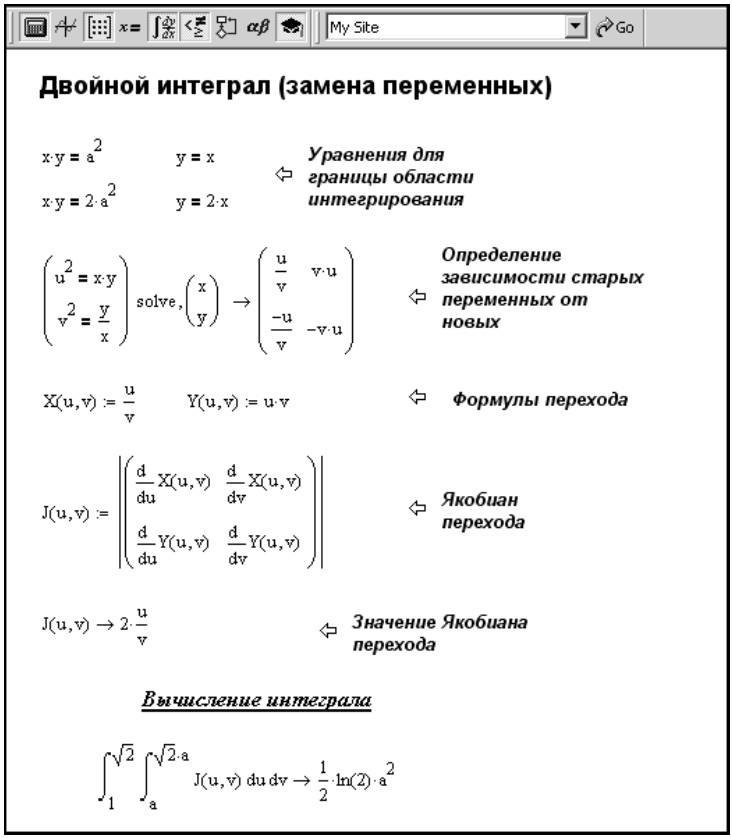


Рис. 6.18. Вычисление двойного интеграла с заменой переменных

Закон перехода от новых переменных `u` и `v` к переменной `x`. Задается в соответствии с найденным на предыдущем шаге решением системы уравнений:

$$X(u,v) := \frac{u}{v}$$

Закон перехода от новых переменных `u` и `v` к переменной `y`. Задается в соответствии с найденным на предыдущем шаге решением системы уравнений:

$$Y(u,v) := u \cdot v$$

Якобиан перехода определяется как детерминант матрицы, составленной из производных от старых переменных по новым:

$$J(u,v) := \left| \begin{pmatrix} \frac{d}{du} X(u,v) & \frac{d}{dv} X(u,v) \\ \frac{d}{du} Y(u,v) & \frac{d}{dv} Y(u,v) \end{pmatrix} \right|$$

Проверка значения якобиана перехода:

$$J(u,v) \rightarrow 2\cdot\frac{u}{v}$$

Вычисление двойного интеграла через повторный, записанный в новых переменных:

$$\int_1^{\sqrt{2}}\int_a^{\sqrt{2}\cdot a} J(u,v)\,du\,dv \rightarrow \frac{1}{2}\cdot\ln(2)\cdot a^2$$

Таким же (или почти таким же) образом могут решаться и более сложные задачи, подразумевающие вычисление двойного интеграла. Кроме того, рассмотренные методы без труда расширяются на случай интегралов более высокой кратности.

Тройные интегралы

Принципиально тройной интеграл отличается от двойного дополнительной переменной интегрирования. От трех аргументов зависит и подынтегральная функция. Методы вычисления двойных и тройных интегралов совпадают, нужно только сделать поправку на число переменных интегрирования. Если область интегрирования V такая, что $a \le x \le b$, $\varphi(x) \le y \le \xi(x)$ и $h(x,y) \le z \le H(x,y)$, то тройной интеграл по соответствующей области может быть вычислен через повторный интеграл $\iiint\limits_V f(x,y,z)dxdydz = \int\limits_a^b dx \int\limits_{\varphi(x)}^{\xi(x)} dy \int\limits_{h(x,y)}^{H(x,y)} f(x,y,z)dz$. При вычислении тройных интегралов с успехом применяется метод замены переменных. Якобиан перехода в этом случае определяется как детерминант матрицы размером 3×3, составленной из производных от старых координат по новым. Рассмотрим вычисление тройных интегралов на примерах.

Задача 6.14. Вычислить объем области, ограниченной поверхностями $z=0$, $z=x^2+y^2$, $y=0$, $y=x^2$, $x=0$ **и** $x=1$.

Решение задачи представлено на рис. 6.19. Там всего две команды и эскиз поверхности.

Эскиз дает представление о верхней границе интегрирования по переменной z. Эта поверхность ограничивает фигуру сверху. Прочие поверхности не показаны, чтобы не загромождать изображение. Поскольку в данном случае определяется объем области, то в качестве подынтегральной функции при

вычислении тройного интеграла указываем единицу. Далее приведены команды из документа.

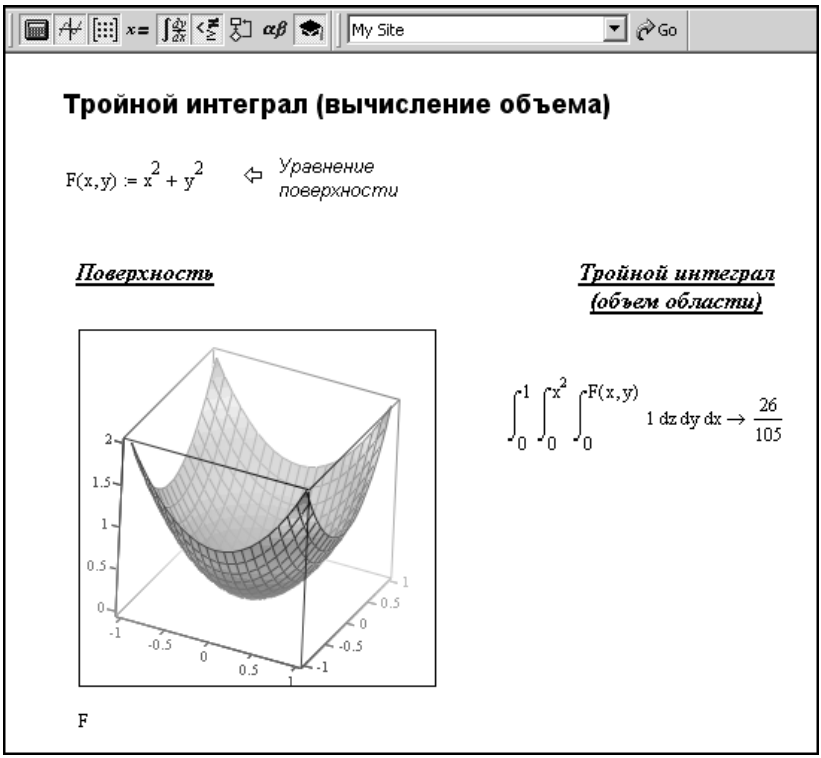


Рис. 6.19. Вычисление объема с помощью тройного интеграла

Инициализация функции двух переменных, определяющей верхнюю границу интегрирования по переменной z:

$$F(x,y) := x^2 + y^2$$

Вычисление объема (тройной интеграл с подынтегральной функцией, равной единице) через повторный интеграл:

$$\int_0^1 \int_0^{x^2} \int_0^{F(x,y)} 1\,dz\,dy\,dx \rightarrow \frac{26}{105}$$

Это пример исключительно прост. В Mathcad можно решать и более сложные задачи.

Задача 6.15. Вычислить тройной интеграл $\iiint\limits_V \sqrt{1-\frac{x^2}{a^2}-\frac{y^2}{b^2}-\frac{z^2}{c^2}}dxdydz$ **по области** V**, заданной соотношением** $\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}\le 1$.

В данном случае при вычислении интеграла разумно выполнить замену переменных. В частности, удобно перейти к модифицированным сферическим координатам согласно соотношениям $x = ar\sin\theta\cos\varphi$, $y = br\sin\theta\sin\varphi$ и $z = cr\cos\theta$. В соответствии с этой заменой вычисляется значение подынтегральной функции и якобиан перехода. Решение представлено в документе на рис. 6.20.

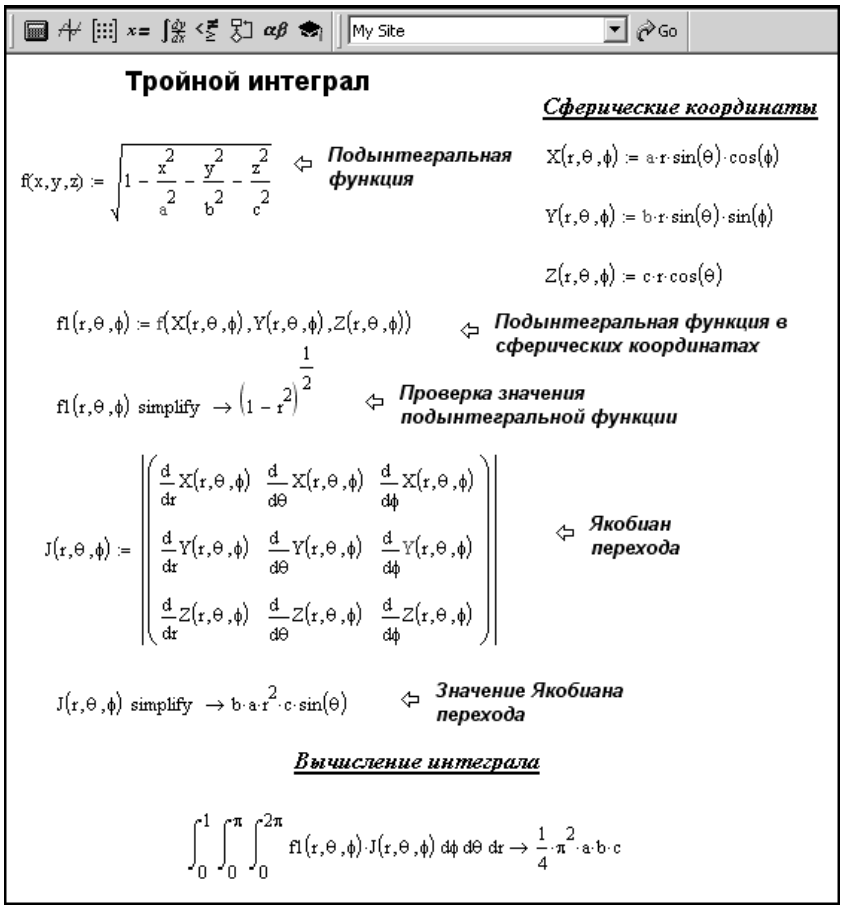


Рис. 6.20. Вычисление тройного интеграла

Сначала инициализирована подынтегральная функция, задан закон перехода к новым переменным, затем вычислен якобиан перехода и, наконец, рассчитан тройной интеграл — общая схема и последовательность действий такая же, как и при вычислении двойных интегралов. Далее приведено описание упомянутых команд.

Подынтегральная функция:

$$f(x,y,z) := \sqrt{1 - \frac{x^2}{a^2} - \frac{y^2}{b^2} - \frac{z^2}{c^2}}$$

Декартовые координаты X, Y и Z выражены через сферические:

$$X(r,\theta,\phi) := a\cdot r\cdot \sin(\theta)\cdot \cos(\phi)$$

$$Y(r,\theta,\phi) := b\cdot r\cdot \sin(\theta)\cdot \sin(\phi)$$

$$Z(r,\theta,\phi) := c\cdot r\cdot \cos(\theta)$$

Определена новая функция, которая является выражением для подынтегральной функции, но в сферических координатах:

$$f1(r,\theta,\phi) := f(X(r,\theta,\phi), Y(r,\theta,\phi), Z(r,\theta,\phi))$$

Проверка значения подынтегральной функции в полярных координатах с упрощением результата:

$$f1(r,\theta,\phi) \text{ simplify } \rightarrow \left(1 - r^2\right)^{\frac{1}{2}}$$

Якобиан перехода — детерминант матрицы размером 3×3. Элементы матрицы — частные производные от старых декартовых по новым сферическим координатам:

$$J(r,\theta,\phi) := \left|\begin{pmatrix} \frac{d}{dr}X(r,\theta,\phi) & \frac{d}{d\theta}X(r,\theta,\phi) & \frac{d}{d\phi}X(r,\theta,\phi) \\ \frac{d}{dr}Y(r,\theta,\phi) & \frac{d}{d\theta}Y(r,\theta,\phi) & \frac{d}{d\phi}Y(r,\theta,\phi) \\ \frac{d}{dr}Z(r,\theta,\phi) & \frac{d}{d\theta}Z(r,\theta,\phi) & \frac{d}{d\phi}Z(r,\theta,\phi) \end{pmatrix}\right|$$

Проверка значения якобиана перехода с одновременным упрощением результата:

$$J(r,\theta,\phi) \text{ simplify } \rightarrow a\cdot \sin(\theta)\cdot b\cdot r^2\cdot c$$

Вычисление тройного интеграла. Интеграл записан через повторный в сферических координатах. Верхние пределы интегрирования по углам θ и ϕ составляют соответственно π и 2π — это стандартные пределы изменения этих углов. Что касается пределов изменения по r, то здесь верхняя граница равна единице, и это следует, например, из выражения для подынтегральной функции в сферической системе координат (выражение под знаком квадратного корня не может быть отрицательным). Нижняя граница по этой переменной по умолчанию принимается равной нулю:

$$\int_0^1 \int_0^{\pi} \int_0^{2\pi} fl(r,\theta,\phi)\cdot J(r,\theta,\phi)\, d\phi\, d\theta\, dr \rightarrow \frac{1}{4}\cdot\pi^2\cdot a\cdot b\cdot c$$

Возвращаясь к области интегрирования, можем отметить, что в рассмотренном примере это эллипсоид. Соответствующая поверхность, ограничивающая область интегрирования, показана на рис. 6.21.

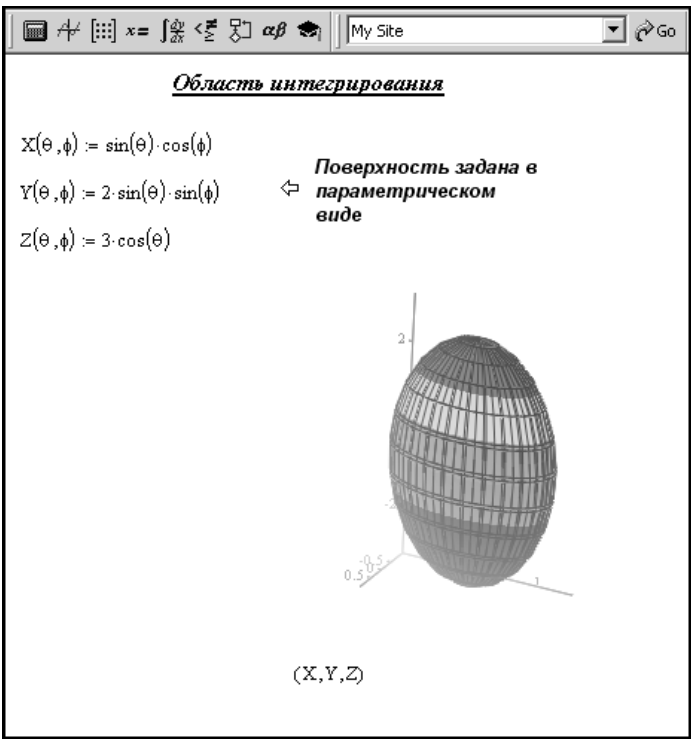


Рис. 6.21. Область интегрирования

Особенность рассмотренной задачи состоит в том, что подынтегральная функция определяет, фактически, и область интегрирования. Ведь интегри-

рование выполнялось по всем значениям переменных, при которых определена подынтегральная функция (точнее, функция является действительной, т. е. выражение под корнем больше нуля). В общем случае при замене координат следует проверять, как в новых координатах запишется область интегрирования. Это необходимо для корректного указания пределов интегрирования по каждой из переменных.

Ряды Фурье

Еще один класс задач, который мы рассмотрим в этой главе — разложение функций в ряды Фурье. Ранее уже описывалось интегральное преобразование Фурье. К рядам Фурье оно имеет непосредственное отношение — интегральное преобразование можно рассматривать как граничный случай ряда Фурье при стремлении интервала разложения функции в ряд к бесконечности. В данном случае ограничимся решением задачи разложения функции $f(x)$ в ряд Фурье на симметричном интервале $(-L, L)$. Напомним, что рядом Фурье функции $f(x)$ называется ряд $f(x) = \frac{a_0}{2} + \sum_{m=1}^{\infty}(a_m \cos(2\pi mx/L) + b_m \sin(2\pi mx/L))$, при этом коэффициенты разложения $a_m = 1/L\int_{-L}^{L} f(x)\cos(2\pi mx/L)dx$ и $b_m = 1/L\int_{-L}^{L} f(x)\sin(2\pi mx/L)dx$. Таким образом, задача сводится к вычислению этих коэффициентов.

Пример 6.2. Ряд Фурье

Для нахождения коэффициентов ряда можно создать специальные процедуры, вид которых представлен в документе на рис. 6.22.

Здесь три процедуры. Первой процедурой вычисляются коэффициенты разложения у косинусов, второй — у синусов. Обе эти процедуры очень схожи. Аргументы у них одинаковые. Это раскладываемая в ряд функция, граница (правая) симметричного интервала разложения и номер коэффициента. Код процедуры состоит, в общем-то, из команды вычисления интеграла: в первой процедуре раскладываемая в ряд функция интегрируется с косинусом, а во второй — с синусом. Перед интегралом стоят соответствующие множители. Третьей процедурой (аргументами которой являются раскладываемая в ряд функция; точка, в которой вычисляется ряд Фурье; правая граница интервала разложения и число слагаемых в ряде) непосредственно вычисляется ряд

(точнее, конечное число слагаемых ряда). В эту процедуру входят созданные ранее процедуры вычисления коэффициентов разложения. Далее рабочий документ на рис. 6.22 содержит примеры вычисления рядов Фурье для двух функций. Ряды вычисляются в символьном виде. В этом случае в качестве точки, в которой вычисляется ряд, можно указать символьное название переменной, а в качестве оператора вычисления значения — стрелку (оператор вычисления результата в символьном виде).

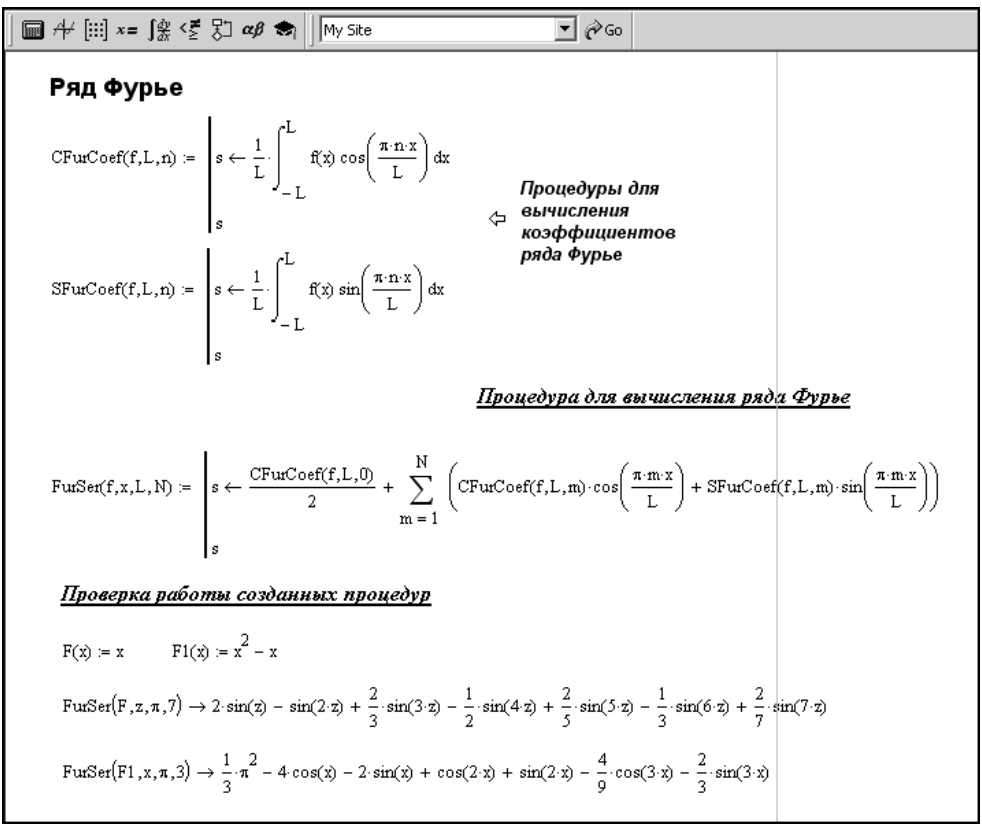


Рис. 6.22. Процедуры для формирования ряда Фурье

Оценивать точность разложения лучше не по формуле, а все же по графику. На рис. 6.23 показан результат аппроксимации линейной функции $F(x) = x$ рядом Фурье на интервале $(-\pi, \pi)$. Аппроксимация выполняется урезанными рядами из двух (тонкая линия) и семи (утолщенная линия) слагаемых.

Можно отметить два обстоятельства. Во-первых, даже относительно небольшим числом слагаемых в ряде можно довольно неплохо аппроксимировать

функцию. Во-вторых, ряд определяет периодическую функцию, совпадающую (в пределе, при бесконечном числе слагаемых) с аппроксимируемой (раскладываемой в ряд) функцией только на интервале разложения. Но это общее свойство рядов Фурье. Другими словами, заменять исходную функцию рядом Фурье можно только на интервале разложения. Исключение имеет место, если раскладываемая в ряд функция периодична с периодом $2L$.

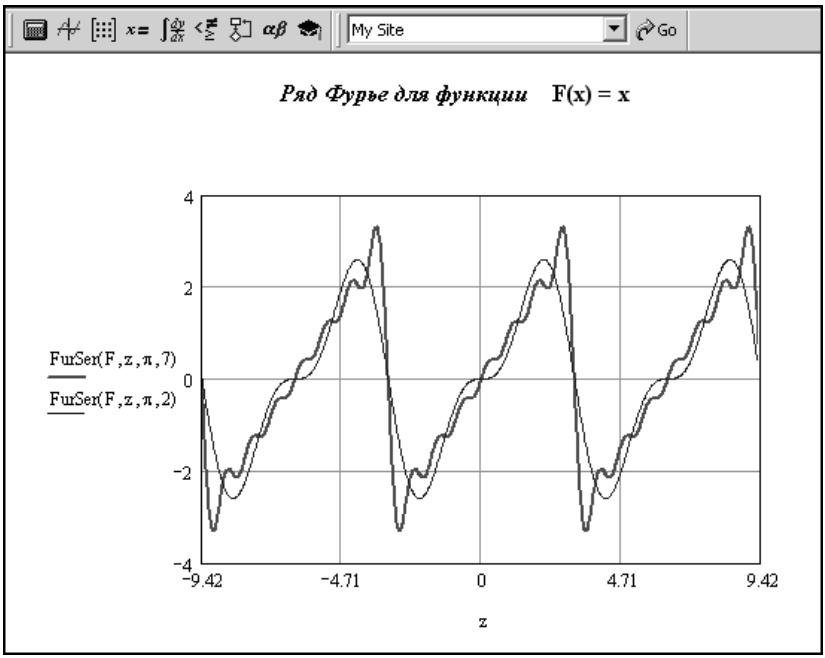


Рис. 6.23. Аппроксимация рядом Фурье линейной функции

Убедиться в сказанном можно из рис. 6.24. Там представлена парабола $F(x) = x(x-1)$ и пять слагаемых ряда Фурье для нее.

Функция раскладывается в ряд на интервале (–1, 1), где, как и следовало ожидать, они достаточно близки. За пределами интервала от этой близости не остается даже воспоминания, — кривые расходятся, как в море корабли. При увеличении числа слагаемых ряда точность аппроксимации на интервале разложения возрастает (рис. 6.25).

Описанным методом можно создавать процедуры и для формирования рядов по ортогональным наборам функций. Думается, читатель сможет это сделать самостоятельно. Еще одно замечание относится к работоспособности предложенных процедур. Дело в том, что если попытаться вычислить ряд Фурье

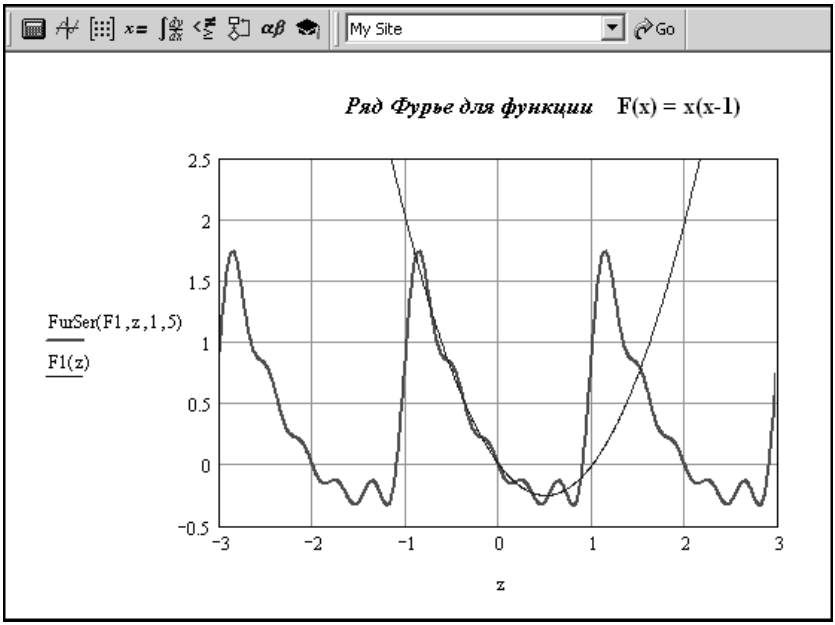


Рис. 6.24. Разложение параболы в ряд Фурье

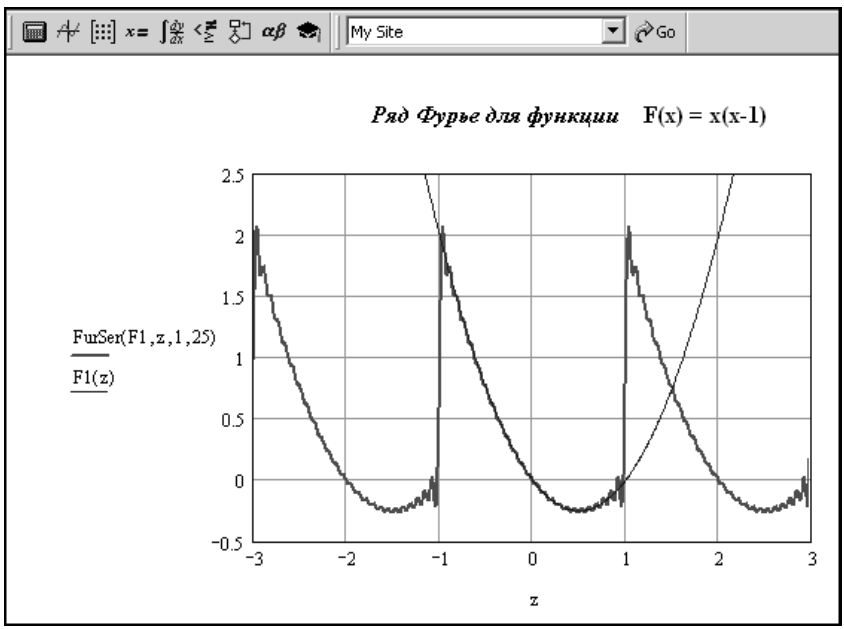


Рис. 6.25. Увеличение количества слагаемых ряда Фурье приводит к более точной аппроксимации

для какого-нибудь простого с математической точки зрения выражения (например, суммы тригонометрических функций), то желаемого результата иногда можно и не достичь. Проблема обычно связана с вычислением коэффициентов разложения, а именно с вычислением интегралов. Поскольку здесь они вычисляются в символьном виде (если речь идет о символьном вычислении ряда), то у пользователя отсутствует возможность принимать участие в упрощении подынтегральных выражений.

На практике ряды обычно служат для аппроксимации раскладываемых функций, а значит допустимы не символьные выражения, а численные значения в формате с плавающей точкой. Тогда при вызове соответствующей процедуры указывают знак обычного равенства, а в качестве переменной для ряда вводят число. Подробнее численные методы описываются в *главе 7*.

Дифференциальные уравнения

Решение дифференциальных уравнений в символьном виде в Mathcad имеет свои особенности. Главная проблема при этом связана с отсутствием универсального способа решения. Дело в том, что метод решения во многом зависит от типа дифференциального уравнения, а решаемые в символьном виде уравнения составляют на практике лишь небольшую группу. К тому же сами символьные вычисления не относятся к приоритетным в Mathcad, а служат скорее вспомогательным инструментом, хотя и очень полезным. Из сказанного не следует, что в Mathcad дифференциальные уравнения решать сложно. Просто здесь нужно проявлять немножко больше изобретательности, чем при решении прочих задач. Перейдем к основным подходам, применимым при этом.

Сначала рассмотрим обыкновенные дифференциальные уравнения — уравнения, в которые входят неизвестные функции одного аргумента и их производные по этому аргументу. Способ решения уравнения зависит от его типа и порядка (определяется максимальным порядком производных). Как и ранее, рассмотрим конкретные примеры.

Задача 6.16. Найти общее решение уравнения $y(x)y'(x) = x^2$.

Это дифференциальное уравнение первого порядка с разделяющимися переменными. Действительно, его можно записать в виде $ydy = x^2dx$. Именно этим выражением и воспользуемся для нахождения общего интеграла. Решение представлено на рис. 6.26.

Далее приведены команды, использованные при решении этой задачи.

Решаемое уравнение:

$$y \cdot dy = x^2 \cdot dx$$

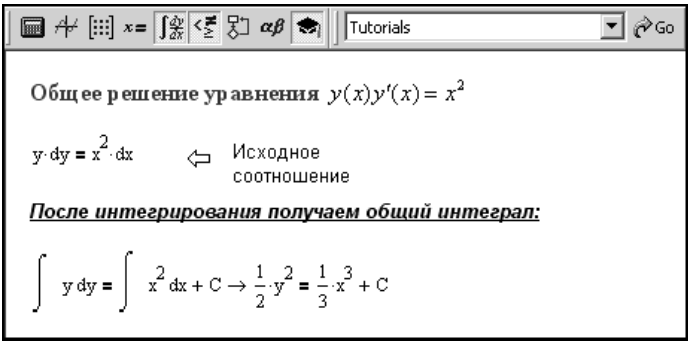


Рис. 6.26. Решение уравнения первого порядка

Общий интеграл уравнения вычислен после интегрирования соотношения, полученного методом разделения переменных:

$$\int y\,dy = \int x^2\,dx + C \rightarrow \frac{1}{2} \cdot y^2 = \frac{1}{3} \cdot x^3 + C$$

Для поиска общего интеграла достаточно проинтегрировать правую и левую части исходного равенства. Не следует забывать, что интегралы вычисляются с точностью до константы, которая после интегрирования добавляется к правой части. При наличии начальных условий (для уравнения первого порядка условие одно) эту константу можно вычислить. Поскольку в формулировке задачи требовалось найти общее решение, то в данном случае это не требуется.

Задача 6.17. Найти общее решение уравнения $(x^2 + y^2(x)) + xy(x)y'(x) = 0$.

Особенность этого уравнения по сравнению с предыдущим состоит в том, что для его решения разумно сделать замену, т. е. ввести новую функцию. Документ с основными командами для решения данного уравнения представлен на рис. 6.27.

В дифференциальном уравнении выполняется замена функции согласно соотношению $y(x) = xz(x)$. Полученное в результате уравнение для новой функции решается относительно производной. В таком виде уравнение допускает применение процедуры разделения переменных. Найденное таким образом соотношение интегрируется. Поскольку оно содержит новую функцию, введенную ранее в результате замены, необходимо перейти к исходной функции. Далее эти команды описаны более подробно.

Исходное решаемое уравнение:

$$x \cdot y(x) \cdot \frac{d}{dx} y(x) = x^2 + y(x)^2$$

Tutorials Go

Общее решение уравнения $(x^2+y^2(x))+xy(x)y'(x)=0$

$$x\cdot y(x)\cdot\frac{d}{dx}y(x)=x^2+y(x)^2$$

⇦ ***Исходное соотношение***

Замена:

$$y(x):=z(x)\cdot x$$

Подстановка в уравнение:

$$x\cdot y(x)\cdot\frac{d}{dx}y(x)=x^2+y(x)^2\rightarrow x^2\cdot z(x)\cdot\left(\frac{d}{dx}z(x)\cdot x+z(x)\right)=x^2+z(x)^2\cdot x^2$$

Преобразование:

$$x^2\cdot z(x)\cdot\left(\frac{d}{dx}z(x)\cdot x+z(x)\right)=x^2+z(x)^2\cdot x^2\ \text{solve},\frac{d}{dx}z(x)\ \rightarrow\frac{-\left(z(x)^2-1\right)}{x\cdot z(x)}$$

Конечное уравнение:

$$\frac{d}{dx}z(x)=\frac{-\left(z(x)^2-1\right)}{x\cdot z(x)}$$

Разделение переменных и интегрирование:

$$\frac{dx}{x}=\frac{zdz}{1+2z^2}\qquad\int\frac{1}{x}dx=\int\frac{z}{1+2\cdot z^2}dz+C\rightarrow\ln(x)=\frac{1}{4}\cdot\ln\left(1+2\cdot z^2\right)+C$$

Обратная замена:

$$z:=\frac{Y}{x}$$

Переход к исходной функции:

$$\ln(x)=\frac{1}{4}\cdot\ln\left(1+2\cdot z^2\right)+C\rightarrow\ln(x)=\frac{1}{4}\cdot\ln\left(1+2\cdot\frac{Y^2}{x^2}\right)+C$$

Рис. 6.27. При решении уравнения используется замена функции

Замена, выполняемая в дифференциальном уравнении:

$$y(x):=z(x)\cdot x$$

Подстановка выражения для старой функции через новую в дифференциальное уравнение:

Полученное уравнение решается (алгебраически) относительно производной:

$$x^2\cdot z(x)\cdot\left(\frac{d}{dx}z(x)\cdot x+z(x)\right)=x^2+z(x)^2\cdot x^2\ \text{solve},\frac{d}{dx}z(x)\ \rightarrow\frac{-\left(z(x)^2-1\right)}{x\cdot z(x)}$$

Такой вид имеет решаемое уравнение, записанное через новую введенную функцию:

$$\frac{d}{dx}z(x) = \frac{-\left(z(x)^2 - 1\right)}{x \cdot z(x)}$$

Соотношение, полученное в результате применения процедуры разделения переменных:

$$\frac{dx}{x} = \frac{zdz}{1 + 2z^2}$$

В результате интегрирования выражения, найденного на предыдущем этапе, получаем общий интеграл для дифференциального уравнения. Это выражение записано через новую функцию:

$$\int \frac{1}{x}\,dx = \int \frac{z}{1 + 2 \cdot z^2}\,dz + C \rightarrow \ln(x) = \frac{1}{4} \cdot \ln\left(1 + 2 \cdot z^2\right) + C$$

Замена, выполняемая для перехода к исходной функции. Поскольку обозначение для этой функции присутствовало в операции присваивания значения, чтобы не возникло рекуррентной ссылки, необходимо новое название (та же буква, но прописная):

$$z := \frac{Y}{x}$$

Переход в общем интеграле дифференциального уравнения к исходной функции, через которую формулировалась задача:

$$\ln(x) = \frac{1}{4} \cdot \ln\left(1 + 2 \cdot z^2\right) + C \rightarrow \ln(x) = \frac{1}{4} \cdot \ln\left(1 + 2 \cdot \frac{Y^2}{x^2}\right) + C$$

В предыдущих примерах решение уравнений определялось в общем виде. На практике обычно приходится решать задачу Коши, т. е. дополнительно учитывать начальные условия.

Задача 6.18. Найти решение задачи Коши: $y'(x) + 4y(x) = \exp(x)$ **и** $y(0) = 0$.

Соответствующее неоднородное уравнение будем решать методом вариации постоянной. При этом, как известно, сначала ищется общее решение соответствующего однородного уравнения, и в полученном решении константа интегрирования заменяется на неизвестную функцию. Вся эта конструкция под-

ставляется в исходное дифференциальное уравнение. В результате получаем уравнение относительно этой неизвестной функции-множителя. Константа интегрирования определяется из начальных условий. Рабочий документ с решением задачи показан на рис. 6.28.

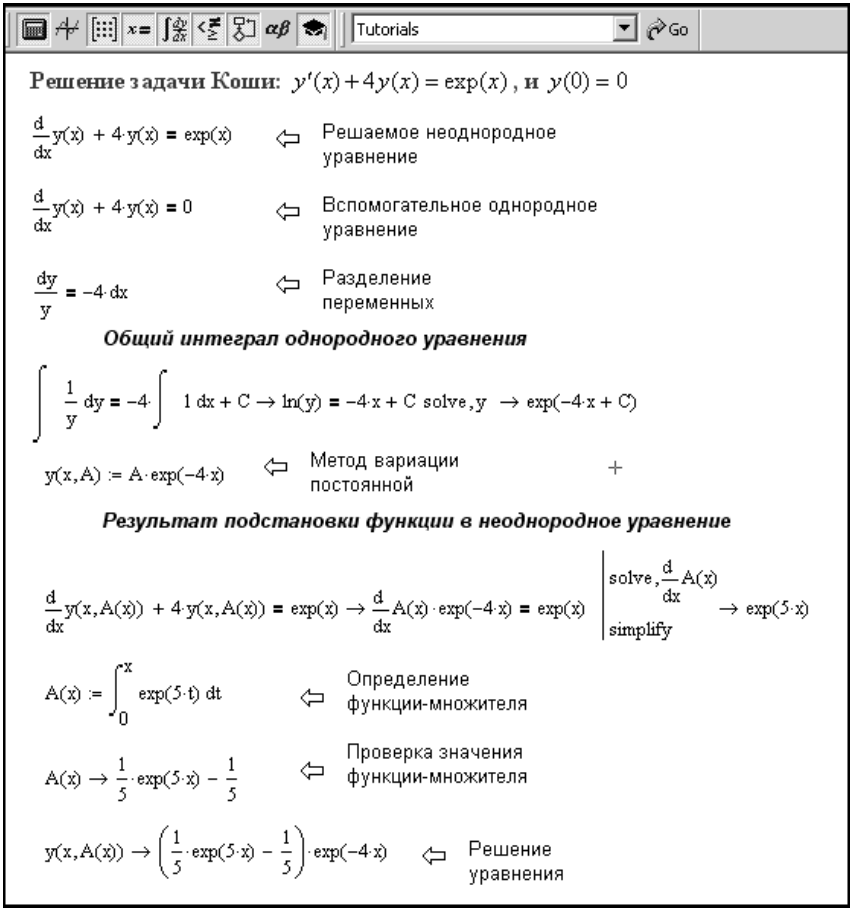


Рис. 6.28. Метод вариации постоянной

Решение однородного уравнения ищется методом разделения переменных. Далее приведены команды, дающие представление о процессе поиска решения.

Исходное решаемое неоднородное уравнение:

$$\frac{d}{dx}y(x) + 4\cdot y(x) = \exp(x)$$

Вспомогательное однородное уравнение:

$$\frac{d}{dx}y(x) + 4\cdot y(x) = 0$$

Выражение, получаемое в результате применения метода разделения переменных для вспомогательного однородного уравнения:

Общий интеграл для вспомогательного однородного уравнения:

$$\int \frac{1}{y}\,dy = -4\cdot\int 1\,dx + C \rightarrow \ln(y) = -4\cdot x + C \text{ solve}, y \rightarrow \exp(-4\cdot x + C)$$

Метод вариации постоянной. Искомая функция определяется как зависящая от двух аргументов. Вторым аргументом определена константа интегрирования. Впоследствии эта функция будет вызываться со вторым аргументом-неизвестной:

$$y(x, A) := A\cdot\exp(-4\cdot x)$$

После подстановки в исходное уравнение получаем уравнение для функции-множителя (точнее, ее производной), которое алгебраически решается относительно производной неизвестной функции-множителя, с одновременным упрощением результата:

$$\frac{d}{dx}y(x, A(x)) + 4\cdot y(x, A(x)) = \exp(x) \rightarrow \frac{d}{dx}A(x)\cdot\exp(-4\cdot x) = \exp(x) \left| \begin{array}{l} \text{solve}, \frac{d}{dx}A(x) \\ \text{simplify} \end{array} \right. \rightarrow \exp(5\cdot x)$$

Функция-множитель определяется через определенный интеграл. Границы интегрирования выбраны так, чтобы удовлетворялись начальные условия (в рассматриваемом примере это легко сделать):

$$A(x) := \int_0^x \exp(5\cdot t)\,dt$$

Функция-множитель найдена:

$$A(x) \rightarrow \frac{1}{5}\cdot\exp(5\cdot x) - \frac{1}{5}$$

Решение задачи Коши:

К сожалению, далеко не каждое дифференциальное уравнение первой степени можно простыми методами свести к виду, допускающему разделение переменных. Задача существенно усложняется, если степень уравнения достаточно высока. Однако и арсенал методов решения уравнений не ограничивается описанным ранее подходом. Перейдем к следующей задаче, в которой приходится решать систему дифференциальных уравнений, правда, не слишком сложную.

Задача 6.19. Найти траекторию движения брошенного под углом к горизонту тела, если на него помимо силы тяжести действует сила сопротивления воздуха, пропорциональная скорости тела.

Сначала следует математически корректно сформулировать задачу. В данном случае воспользуемся вторым законом Ньютона, связывающим ускорения тела с силами, действующими на него. Таким образом, речь идет о системе дифференциальных уравнений, которую предстоит решить. Реализуя решение этой задачи в Mathcad, по возможности будем стараться получать результаты в символьном (аналитическом) виде. Фрагмент соответствующего документа (верхняя его часть) показан на рис. 6.29.

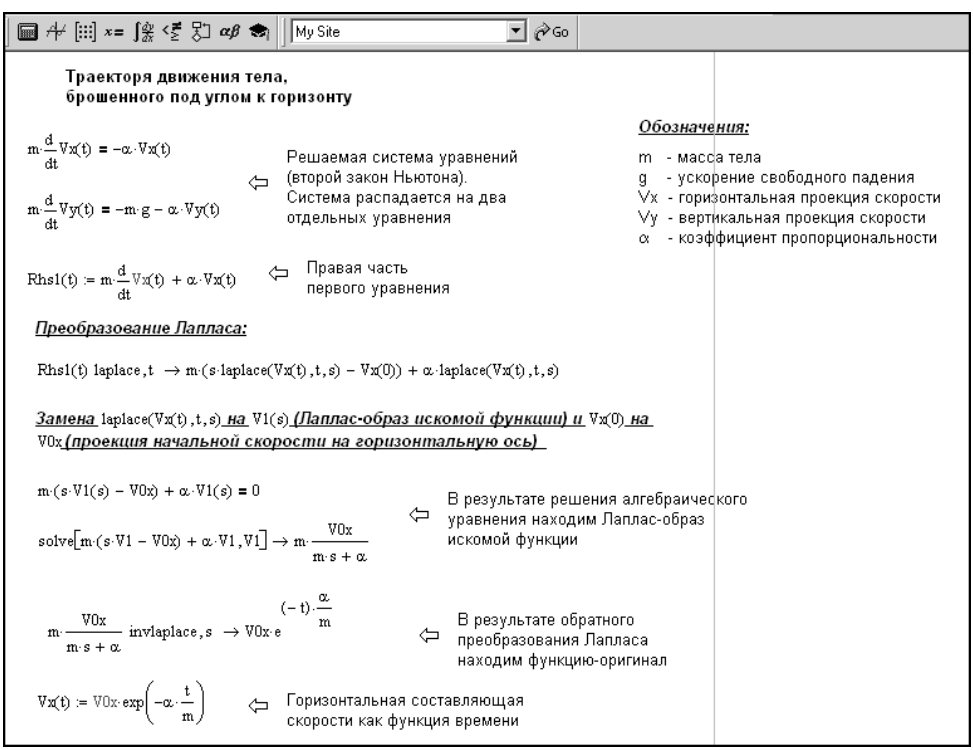


Рис. 6.29. Первая часть документа с решением задачи о полете тела (решение первого уравнения)

В качестве исходных функций выбираем проекции скорости тела на горизонтальную и вертикальную оси. Ускорение записывается как производная от скорости, а силы (их в данном случае две — силы тяжести и сопротивления воздуха) либо постоянны (сила тяжести), либо определяются через скорость тела (сила сопротивления воздуха). Записав проекции каждой из двух сил на координатные оси, получаем уравнения Ньютона. Они записаны в верхней части документа (для дальнейших расчетов удобнее все слагаемые в этих уравнениях перенести в одну часть и рассматривать ее как функцию времени). Далее это выражение подвергается преобразованию Лапласа, находится образ искомой функции, а после обратного преобразования и сама функция. Эта же процедура должна быть проделана и со вторым уравнением. В результате получаем функциональные зависимости горизонтальной и вертикальной составляющих скорости тела от времени. Вторая часть документа, в которой получено выражение для вертикальной составляющей скорости, показана на рис. 6.30.

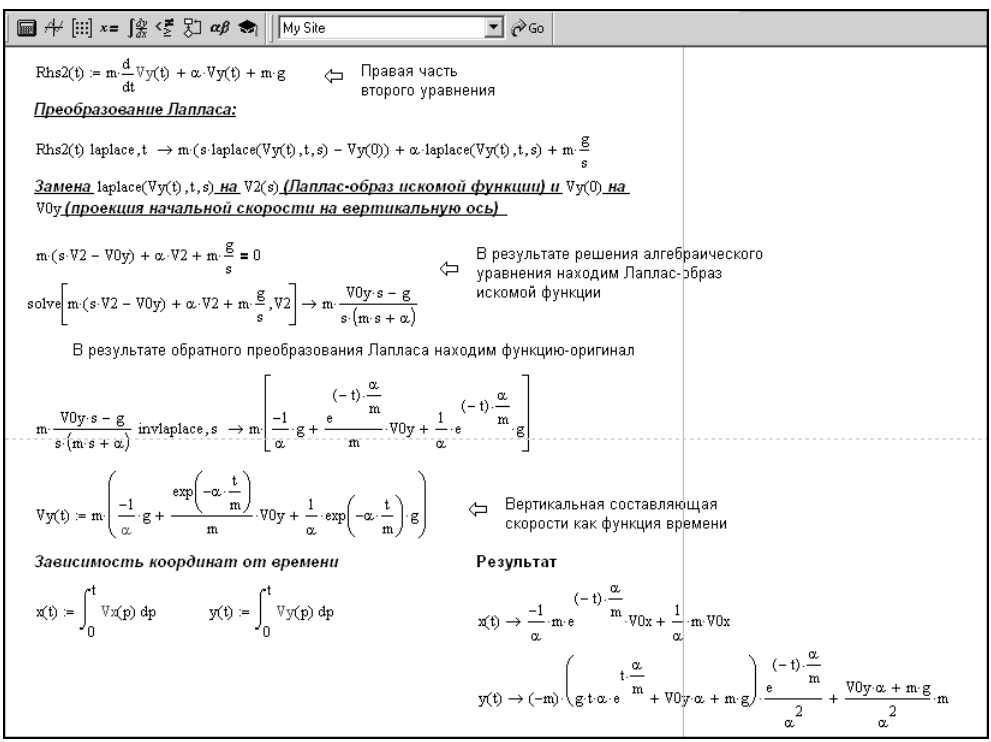


Рис. 6.30. Вторая часть документа с решением задачи о полете тела (решение второго уравнения)

Помимо скорости, вычисляется также и зависимость координат от времени, именно эти выражения служат основой для определения траектории. Это не сложно, достаточно проинтегрировать компоненты скорости и принять во внимание, что в нулевой момент времени тело находится в начале координат. Далее приведены команды рабочего документа.

Первое уравнение, определяющее движение тела вдоль горизонтальной оси. В левой части — произведение массы на ускорение (первая производная от скорости по времени), в правой — проекция силы сопротивления воздуха на горизонтальную ось. Эта сила по условию задачи пропорциональна скорости:

$$m\cdot\frac{d}{dt}Vx(t) = -\alpha\cdot Vx(t)$$

Второе уравнение, определяющее движение тела вдоль вертикальной оси. В левой части — произведение массы на ускорение (первая производная от скорости по времени), в правой — проекция силы тяжести и силы сопротивления воздуха на вертикальную ось. Сила тяжести равна по абсолютной величине произведению массы на ускорение свободного падения:

$$m\cdot\frac{d}{dt}Vy(t) = -m\cdot g\cdot -\alpha\cdot Vy(t)$$

Выражение, определяющее первое уравнение, записано в виде функции времени:

$$Rhs1(t) := m\cdot\frac{d}{dt}Vx(t) + \alpha\cdot Vx(t)$$

По отношению к первому уравнению выполнено преобразование Лапласа:

$$Rhs1(t)\ \text{laplace}\ ,t\ \rightarrow m\cdot(s\cdot\text{laplace}(Vx(t),t,s) - Vx(0)) + \alpha\cdot\text{laplace}(Vx(t),t,s)$$

В выражении, полученном после преобразования Лапласа, формально заменена функция и ее значение в начальный момент времени:

$$m\cdot(s\cdot V1(s) - V0x) + \alpha\cdot V1(s) = 0$$

В результате решения алгебраического уравнения найден Лаплас-образ искомой функции:

$$\text{solve}\left[m\cdot(s\cdot V1 - V0x) + \alpha\cdot V1, V1\right] \rightarrow m\cdot\frac{V0x}{m\cdot s + \alpha}$$

После обратного преобразования Лапласа получаем зависимость горизонтальной составляющей скорости тела от времени:

$$m\cdot\frac{V0x}{m\cdot s + \alpha}\ \text{invlaplace}, s\ \rightarrow V0x\,e^{(-t)\cdot\frac{\alpha}{m}}$$

Горизонтальная скорость тела задана в виде функции времени:

Выражение, определяющее второе уравнение, представлено в виде функции времени:

$$Rhs2(t) := m\cdot\frac{d}{d\ t}Vy(t) + \alpha\cdot Vy(t) + m\cdot g$$

По отношению ко второму уравнению выполнено преобразование Лапласа:

$$Rhs2(t)\ laplace, t \rightarrow m\cdot(s\cdot laplace(Vy(t), t, s) - Vy(0)) + \alpha\cdot laplace(Vy(t), t, s) + m\cdot\frac{g}{s}$$

В выражении, полученном после преобразования Лапласа, формально заменена функция и ее значение в начальный момент времени:

$$m\cdot(s\cdot V2 - V0y) + \alpha\cdot V2 + m\cdot\frac{g}{s} = 0$$

В результате решения алгебраического уравнения найден Лаплас-образ искомой функции:

$$solve\left[m\cdot(s\cdot V2 - V0y) + \alpha\cdot V2 + m\cdot\frac{g}{s}, V2\right] \rightarrow m\cdot\frac{V0y\cdot s - g}{s\cdot(m\cdot s + \alpha)}$$

После обратного преобразования Лапласа получаем зависимость вертикальной составляющей скорости тела от времени:

$$m\cdot\frac{V0y\cdot s - g}{s\cdot(m\cdot s + \alpha)}\ invlaplace, s \rightarrow m\cdot\left[\frac{-1}{\alpha}\cdot g + \frac{e^{(-t)\cdot\frac{\alpha}{m}}}{m}\cdot V0y + \frac{1}{\alpha}\cdot e^{(-t)\cdot\frac{\alpha}{m}}\cdot g\right]$$

Вертикальная скорость тела задана в виде функции времени:

$$Vy(t) := m\cdot\left(\frac{-1}{\alpha}\cdot g + \frac{\exp\left(-\alpha\cdot\frac{t}{m}\right)}{m}\cdot V0y + \frac{1}{\alpha}\cdot\exp\left(-\alpha\cdot\frac{t}{m}\right)\cdot g\right)$$

Вычисление горизонтальной и вертикальной координат тела в виде функции времени:

$$x(t) := \int_0^t Vx(p)\ dp$$

$$y(t) := \int_0^t Vy(p)\ dp$$

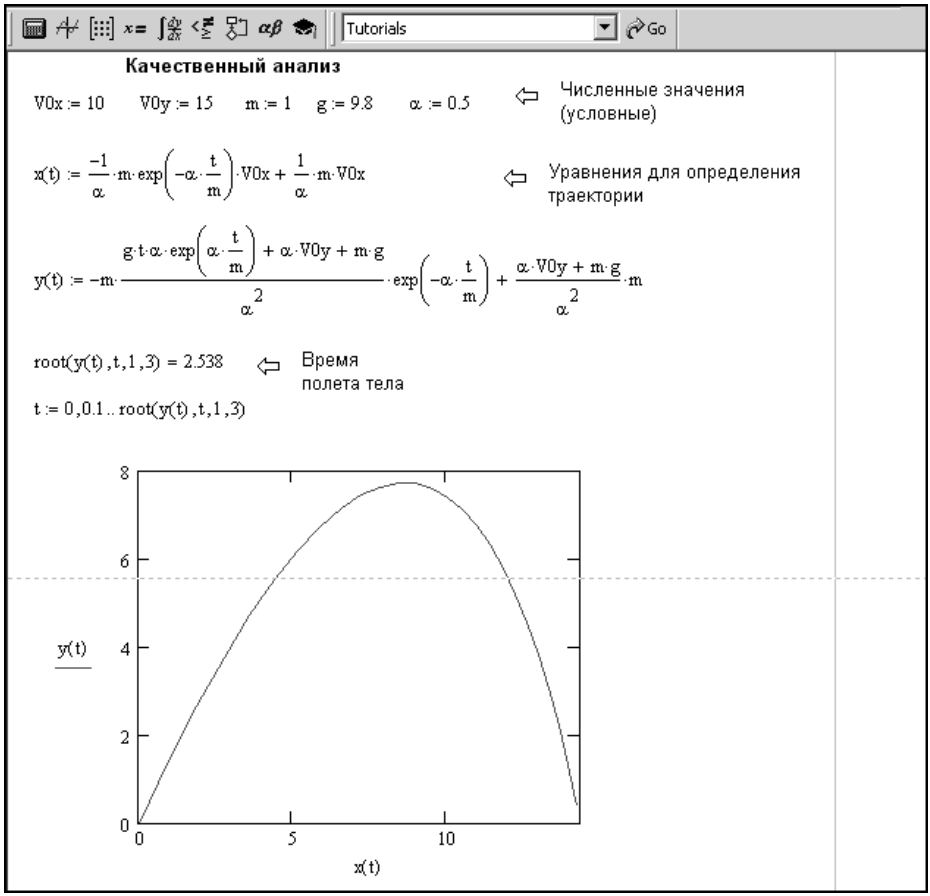


Рис. 6.31. Траектория полета тела

Аналитическая зависимость горизонтальной и вертикальной координат от времени:

$$x(t) \rightarrow \frac{-1}{\alpha}\cdot m\cdot \exp\left(-\alpha\cdot\frac{t}{m}\right)\cdot V0x + \frac{1}{\alpha}\cdot m\cdot V0$$

$$y(t) \rightarrow -m\cdot\frac{g\cdot t\cdot\alpha\cdot\exp\left(\alpha\cdot\frac{t}{m}\right) + \alpha\cdot V0y + m\cdot g}{\alpha^2}\cdot\exp\left(-\alpha\cdot\frac{t}{m}\right) + \frac{\alpha\cdot V0y + m\cdot g}{\alpha^2}\cdot m$$

Хотя формально речь шла о системе уравнений, на самом деле последовательно решались два отдельных уравнения. Это стало возможным благодаря тому, что проекции на координатные оси сил, действующих на тело, зависят только от проекций соответствующих скоростей. Такой вариант — скорее

исключение из правил, поэтому в общем случае с системой дифференциальных уравнений придется повозиться.

Несмотря на то, что аналитическое решение получено, траекторию движения тела хотелось бы представить наглядно. Для этого строим график, предварительно задав параметрам задачи (массе тела, начальной скорости, коэффициенту пропорциональности и т. п.) численные значения, а также определяем зависимости координат от времени. Эти зависимости указываются у координатных осей на графике (рис. 6.31).

Важно правильно выбрать интервал времени, на протяжении которого отслеживается движение тела. Если он слишком большой, траектория уйдет ниже уровня горизонта. Положив горизонтальную координату равной нулю, можно найти время полета тела. Решений, очевидно, два; то, которое соответствует начальному моменту времени, в расчет не принимаем.

Параметры задачи подобраны так, что траектория тела на рис. 6.31 существенно отличается от параболы (в случае, если сопротивления воздуха нет, брошенное под углом к горизонту тело движется по параболе). С физической точки зрения этот результат вполне ожидаем и объясним.

В следующей задаче также будем решать систему дифференциальных уравнений, но только теперь на отдельные уравнения она уже не распадается.

Задача 6.20. В область пространства с постоянными электрическим и магнитным полями влетает заряженная частица. Векторы электрической и магнитной индукций взаимно перпендикулярны. Вектор начальной скорости частицы также перпендикулярен электрическому и магнитному векторам. Определить траекторию движения частицы. Гравитационными силами пренебречь.

Как и ранее, здесь для определения траектории частицы воспользуемся уравнениями Ньютона. На частицу (если не учитывать силы гравитации) действуют две силы: со стороны электрического и магнитного полей. Сила, действующая на частицу со стороны электрического поля, пропорциональна произведению заряда частицы на вектор напряженности электрического поля. С этой силой проблем обычно не возникает (имеется в виду, конечно же, решение системы уравнений). Хуже дела обстоят с силой Лоренца. Она определяется через векторное произведение скорости частицы на вектор напряженности магнитного поля (все это еще умножается на заряд частицы). Как следствие система уравнений, описывающая динамику частицы, уже не распадается на отдельные уравнения, что несколько усложняет задачу, однако не делает ее неразрешимой.

Документ, в котором представлено решение задачи из-за его громоздкости, разбит на три части. Первые две содержат команды вычисления функций

координат, а на последней схематично представлен график траектории частицы. Первую часть документа можно видеть на рис. 6.32.

Траектория заряженной частицы

$X0 := 0 \quad Y0 := 0 \quad Z0 := 0$ ⇦ Начальные координаты частицы

$dX0 := 0 \quad dY0 := V0 \quad dZ0 := 0$ ⇦ Начальная скорость частицы

$E := \begin{pmatrix} E0 \\ 0 \\ 0 \end{pmatrix} \quad B := \begin{pmatrix} 0 \\ 0 \\ B0 \end{pmatrix}$ ⇦ Векторы электрического и магнитного полей

Обозначения:
M - масса
e0 - заряд
E - электрическое поле
B - магнитное поле
V0 - начальная скорость

Выражение, определяющее уравнение в векторном виде:

$$M\cdot\begin{pmatrix} \frac{d^2}{dt^2}x(t) \\ \frac{d^2}{dt^2}y(t) \\ \frac{d^2}{dt^2}z(t) \end{pmatrix} = e0\cdot E + e0\cdot\begin{pmatrix} \frac{d}{dt}x(t) \\ \frac{d}{dt}y(t) \\ \frac{d}{dt}z(t) \end{pmatrix} \times B$$

⇦ Система уравнений в векторном виде

$$M\cdot\begin{pmatrix} \frac{d^2}{dt^2}x(t) \\ \frac{d^2}{dt^2}y(t) \\ \frac{d^2}{dt^2}z(t) \end{pmatrix} - \left[e0\cdot E + e0\cdot\begin{pmatrix} \frac{d}{dt}x(t) \\ \frac{d}{dt}y(t) \\ \frac{d}{dt}z(t) \end{pmatrix} \times B \right] \rightarrow \begin{pmatrix} \frac{d}{dt}\frac{d}{dt}x(t)\cdot M - E0\cdot e0 - \frac{d}{dt}y(t)\cdot e0\cdot B0 \\ \frac{d}{dt}\frac{d}{dt}y(t)\cdot M + \frac{d}{dt}x(t)\cdot e0\cdot B0 \\ \frac{d}{dt}\frac{d}{dt}z(t)\cdot M \end{pmatrix}$$

Функции, определяющие отдельные уравнения

$$R1(t) := \frac{d}{dt}\frac{d}{dt}(x(t)\cdot M) - E0\cdot e0 - \frac{d}{dt}(y(t)\cdot e0\cdot B0)$$

$$R2(t) := \frac{d}{dt}\frac{d}{dt}(y(t)\cdot M) + \frac{d}{dt}(x(t)\cdot e0\cdot B0)$$

$$R3(t) := \frac{d}{dt}\frac{d}{dt}z(t)\cdot M$$

Преобразование Лапласа:

$$R1(t)\ \text{laplace},t \rightarrow M\cdot\left[s\cdot(s\cdot\text{laplace}(x(t),t,s) - x(0)) - \left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}x(t) \end{matrix}\right.\right] - E0\cdot\frac{e0}{s} - e0\cdot B0\cdot(s\cdot\text{laplace}(y(t),t,s) - y(0))$$

$$R2(t)\ \text{laplace},t \rightarrow M\cdot\left[s\cdot(s\cdot\text{laplace}(y(t),t,s) - y(0)) - \left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}y(t) \end{matrix}\right.\right] + e0\cdot B0\cdot(s\cdot\text{laplace}(x(t),t,s) - x(0))$$

$$R3(t)\ \text{laplace},t \rightarrow M\cdot\left[s\cdot(s\cdot\text{laplace}(z(t),t,s) - z(0)) - \left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}z(t) \end{matrix}\right.\right]$$

Рис. 6.32. Первая часть документа с расчетом траектории частицы

Поскольку в общем случае решение задачи будет выглядеть громоздко, некоторые упрощения делаем сразу. Во-первых, примем во внимание, что в начальный момент времени частица находится в начале координат. Кроме того, систему координат выберем так, чтобы координатные оси (направление) совпадали с векторами скорости и напряженности электрического и магнитного полей. В связи с этим на первом этапе определяются параметры начальных координат и скоростей частицы. Векторы электрического и магнитного полей инициализированы так, что имеют по одному ненулевому элементу. Направление вектора электрического поля совпадает с осью O*x*, вектора магнитной индукции — с осью O*z*, а начальной скорости — с осью O*y*.

В общем случае второй закон Ньютона для отдельной частицы дает одно уравнение в векторной форме или три (для трехмерной задачи) — в проекциях. В предыдущей задаче отдельно выписывалось каждое уравнение для проекций. Здесь запишем одно векторное уравнение. Такой подход имеет существенное преимущество, поскольку позволяет легко и просто вычислить силу Лоренца, которая, как упоминалось, определяется через векторное произве-

дение векторов — в Mathcad для этого есть специальная команда (на палитре **Matrix**). Поэтому с записью исходных уравнений проблем не возникает. Получаем систему дифференциальных уравнений второго порядка с тремя неизвестными функциями (функции задают зависимость от времени координат частицы, скорость определяется через первые производные от этих функций, а ускорение — через вторые производные). Для решения системы к каждому из трех уравнений применяем преобразование Лапласа. В результате для образов функций получаем систему линейных уравнений, которую и предстоит решить. Сначала выделяем для данной линейной системы матрицу коэффициентов и вектор правых частей. Далее система решается и затем выполняется обратное преобразование Лапласа.

На рис. 6.33 показана вторая часть документа с решением задачи. Там размещены команды, начиная с замены в уравнениях для образов функций (как и в предыдущей задаче, перед решением системы уравнений вводятся более удобные обозначения для образов функций, их значений и производных в начальный момент).

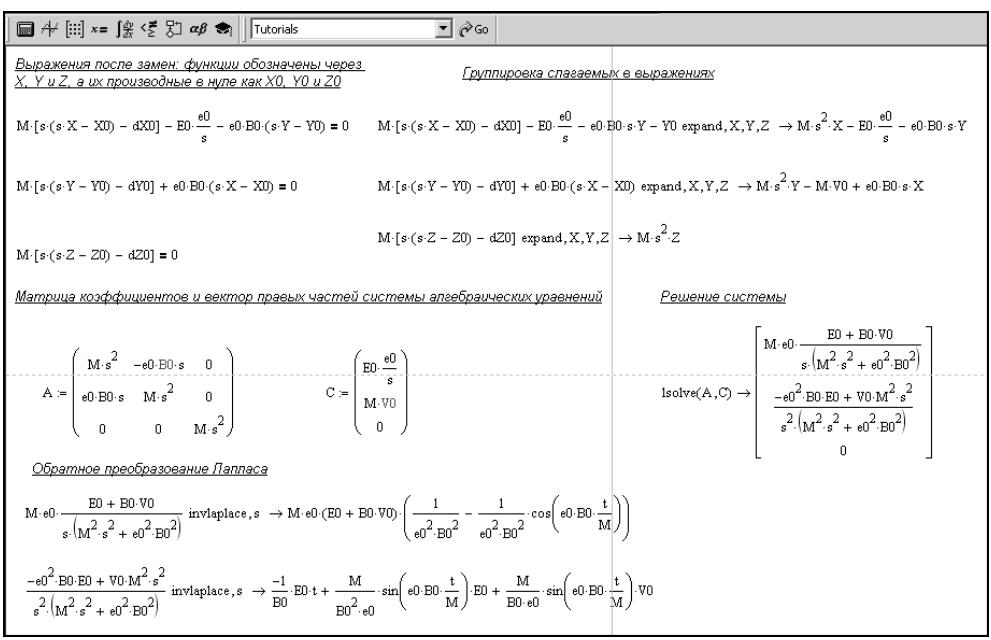


Рис. 6.33. Вторая часть документа с расчетом траектории частицы

Далее приведены некоторые комментарии по поводу команд рабочего документа.

Значения координат частицы и проекций скорости на координатные оси в начальный момент времени. Все координаты равны нулю. Только одна из

проекций скорости на координатную ось отлична от нуля. Координатная система выбрана так, что это ось O*y*:

$$X0 := 0,\ Y0 := 0,\ Z0 := 0,\ dX0 := 0,\ dY0 := V0,\ dZ0 := 0$$

Вектор напряженности электрического поля. В силу выбора координатной системы вектор имеет один отличный от нуля элемент (для оси O*x*):

$$E := \begin{pmatrix} E0 \\ 0 \\ 0 \end{pmatrix}$$

Вектор напряженности магнитного поля. В силу выбора координатной системы вектор имеет один отличный от нуля элемент (для оси O*z*):

$$B := \begin{pmatrix} 0 \\ 0 \\ B0 \end{pmatrix}$$

Базовое уравнение (второй закон Ньютона) в векторной форме. Сила Лоренца определяется через векторное произведение скорости на напряженность магнитного поля:

$$M\cdot\begin{pmatrix} \frac{d^2}{d\,t^2}x(t) \\ \frac{d^2}{d\,t^2}y(t) \\ \frac{d^2}{d\,t^2}z(t) \end{pmatrix} = e0\cdot E + e0\cdot\begin{pmatrix} \frac{d}{d\,t}x(t) \\ \frac{d}{d\,t}y(t) \\ \frac{d}{d\,t}z(t) \end{pmatrix} \times B$$

Выражение, определяющее исходное уравнение:

$$M\cdot\begin{pmatrix} \frac{d^2}{dt^2}x(t) \\ \frac{d^2}{dt^2}y(t) \\ \frac{d^2}{dt^2}z(t) \end{pmatrix} - \left[e0\cdot E + e0\cdot\begin{pmatrix} \frac{d}{dt}x(t) \\ \frac{d}{dt}y(t) \\ \frac{d}{dt}z(t) \end{pmatrix} \times B \right] \rightarrow \begin{pmatrix} \frac{d}{dt}\frac{d}{dt}x(t)\cdot M - E0\cdot e0 - \frac{d}{dt}y(t)\cdot e0\cdot B0 \\ \frac{d}{dt}\frac{d}{dt}y(t)\cdot M + \frac{d}{dt}x(t)\cdot e0\cdot B0 \\ \frac{d}{dt}\frac{d}{dt}z(t)\cdot M \end{pmatrix}$$

Функции, определяющие первое, второе и третье уравнения:

$$R1(t) := \frac{d}{dt}\frac{d}{dt}x(t)\cdot M - E0\cdot e0 - \frac{d}{dt}y(t)\cdot e0\cdot B0$$

$$R2(t) := \frac{d}{dt}\frac{d}{dt}y(t)\cdot M + \frac{d}{dt}x(t)\cdot e0\cdot B0$$

$$R3(t) := \frac{d}{dt}\frac{d}{dt}z(t)\cdot M$$

Преобразования Лапласа для первого, второго и третьего уравнений:

$$R1(t)\ \text{laplace}, t \rightarrow M\cdot\left[s\cdot(s\cdot\text{laplace}(x(t),t,s) - x(0)) - \left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}x(t) \end{matrix}\right.\right] - E0\frac{e0}{s} - e0\cdot B0(s\cdot\text{laplace}(y(t),t,s) - y(0))$$

$$R2(t)\ \text{laplace}, t \rightarrow M\cdot\left[s\cdot(s\cdot\text{laplace}(y(t),t,s) - y(0)) - \left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}y(t) \end{matrix}\right.\right] + e0\cdot B0\cdot(s\cdot\text{laplace}(x(t),t,s) - x(0))$$

$$R3(t)\ \text{laplace}, t \rightarrow M\cdot\left[s\cdot(s\cdot\text{laplace}(z(t),t,s) - z(0)) - \left|\begin{matrix} t \leftarrow 0 \\ \frac{d}{dt}z(t) \end{matrix}\right.\right]$$

В выражении, полученном при преобразовании Лапласа, для первого уравнения заменены обозначения для функций (это `X`, `Y` и `Z`), их начальных значений (`X0`, `Y0` и `Z0`) и начальных значений их производных (`dX0`, `dY0` и `dZ0`):

$$M\cdot[s\cdot(s\cdot X - X0) - dX0] - E0\cdot\frac{e0}{s} - e0\cdot B0\cdot(s\cdot Y - Y0) = 0$$

То же, что и в предыдущем случае, но для второго и третьего уравнений:

$$M\cdot[s\cdot(s\cdot Y - Y0) - dY0] + e0\cdot B0\cdot(s\cdot X - X0) = 0$$

$$M\cdot[s\cdot(s\cdot Z - Z0) - dZ0] = 0$$

Группировка слагаемых в первом выражении. Это необходимо для того, чтобы затем легче было определить матрицу коэффициентов и вектор правых частей для системы линейных уравнений относительно образов искомых функций:

$$M\cdot[s\cdot(s\cdot X - X0) - dX0] - E0\frac{e0}{s} - e0\cdot B0\cdot s\cdot Y - Y0\ \text{expand}, X, Y, Z \rightarrow M\cdot s^2\cdot X - E0\frac{e0}{s} - e0\cdot B0\cdot s\cdot Y$$

Группировка слагаемых во втором и третьем выражениях:

$$M\cdot[s\cdot(s\cdot Z - Z0) - dZ0]\ \text{expand}, X, Y, Z \rightarrow M\cdot s^2\cdot Z$$

Матрица коэффициентов для системы линейных алгебраических уравнений:

$$A := \begin{pmatrix} M\cdot s^2 & -e0\cdot B0\cdot s & 0 \\ e0\cdot B0\cdot s & M\cdot s^2 & 0 \\ 0 & 0 & M\cdot s^2 \end{pmatrix}$$

Вектор правых частей:

$$C := \begin{pmatrix} E0\cdot\dfrac{e0}{s} \\ M\cdot V0 \\ 0 \end{pmatrix}$$

Решение системы линейных уравнений относительно образов искомых функций:

$$\text{lsolve}\ (A\ ,C) \rightarrow \begin{bmatrix} M\cdot e0\cdot\dfrac{E0\ +\ B0\cdot V0}{s\cdot\left(M^2\cdot s^2\ +\ e0^2\cdot B0^2\right)} \\ \dfrac{-e0^2\cdot B0\cdot E0\ +\ V0\cdot M^2\cdot s^2}{s^2\cdot\left(M^2\cdot s^2\ +\ e0^2\cdot B0^2\right)} \\ 0 \end{bmatrix}$$

После обратного преобразования Фурье получаем выражение для координаты частицы в виде функции времени:

$$M\cdot e0\cdot\frac{E0 + B0\cdot V0}{s\cdot\left(M^2\cdot s^2 + e0^2\cdot B0^2\right)}\ \text{invlaplace}\,,s\ \rightarrow$$

$$\rightarrow M\cdot e0\cdot(E0 + B0\cdot V0)\cdot\left(\frac{1}{e0^2\cdot B0^2} - \frac{1}{e0^2\cdot B0^2}\cdot\cos\left(e0\cdot B0\cdot\frac{t}{M}\right)\right)$$

То же, что и в предыдущем случае, но для второй координаты:

$$\frac{-e0^2\cdot B0\cdot E0 + V0\cdot M^2\cdot s^2}{s^2\cdot\left(M^2\cdot s^2 + e0^2\cdot B0^2\right)}\ \text{invlaplace}\,,s\ \rightarrow$$

$$\rightarrow \frac{-1}{B0}\cdot E0\cdot t + \frac{M}{B0^2\cdot e0}\cdot\sin\left(e0\cdot B0\cdot\frac{t}{M}\right)\cdot E0 + \frac{M}{B0\cdot e0}\cdot\sin\left(e0\cdot B0\cdot\frac{t}{M}\right)\cdot V0$$

В процессе решения найдена зависимость координат частицы от времени, причем движение осуществляется в плоскости, перпендикулярной вектору

магнитного поля. В этом нет ничего удивительного, поскольку сила Лоренца направлена перпендикулярно к векторам скорости и магнитной индукции. Поэтому все действующие на частицу силы лежат в плоскости, перпендикулярной оси O*z*.

Чтобы построить траекторию, зададим численные значения для параметров задачи (абсолютно условные). Результат показан на рис. 6.34.

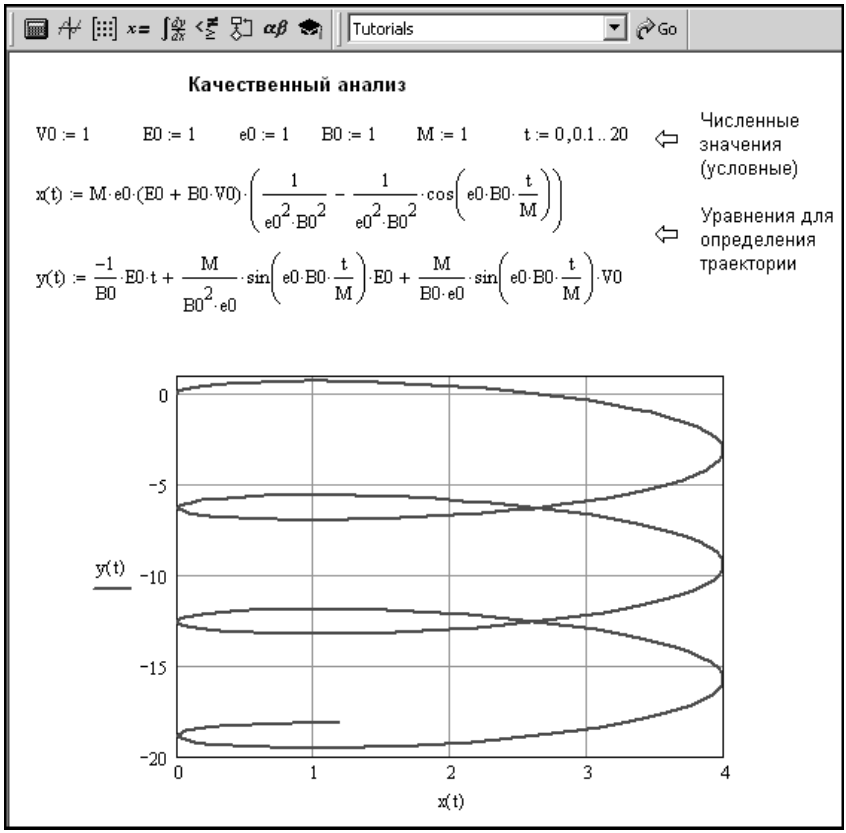
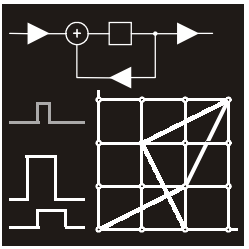


Рис. 6.34. Траектория частицы в электрическом и магнитном полях

На этом иллюстрацию возможностей Mathcad в области символьных вычислений заканчиваем. Рассмотренные здесь методы решения задач, безусловно, не являются исчерпывающими. Тем не менее, они дают некоторое представление о практике работы с Mathcad.



Глава 7

Численные и статистические методы

Несмотря на все успехи в реализации символьных методов, Mathcad тем не менее остается приложением, ориентированным на выполнение численных расчетов. Спектр решаемых при этом задач чрезвычайно широк. Имеются в виду, прежде всего, встроенные возможности Mathcad. Кроме того, пользователь при желании может создавать собственные процедуры, функции, а то и целые рабочие документы, выполняющие те или иные действия. Здесь будут затронуты вопросы, связанные с решением в численном виде алгебраических и дифференциальных уравнений и систем, кратко рассмотрены задачи численного интегрирования и дифференцирования, а также методы интерполяции. Поскольку очень часто интерес представляет не столько конечный результат, а способ его получения (это актуально для студентов), в некоторых случаях предлагаются алгоритмы, реализованные с помощью программных средств Mathcad, позволяющие альтернативными методами решать упомянутые задачи.

Решение алгебраических уравнений и систем

В этом разделе рассмотрим основные методы, которые существуют в Mathcad для решения алгебраических уравнений и систем. Многие подходы, применяемые в случае решения алгебраических уравнений, с минимальными изменениями расширяются для систем. Поэтому вполне логично начать с рассмотрения алгебраических уравнений.

Алгебраические уравнения

В Mathcad решение алгебраических уравнений осуществляется несколькими способами. Можно для этих целей воспользоваться процедурой `root()`.

Аргументами указывают решаемое уравнение (точнее, выражение, которое приравнивается нулю) и переменную, относительно которой это уравнение решается. Процедура может вызываться с указанием начального приближения для поиска корня уравнения или диапазона, на котором ищется решение.

Пример 7.1. Решение уравнений

Примеры вызова процедуры приведены в документе на рис. 7.1.

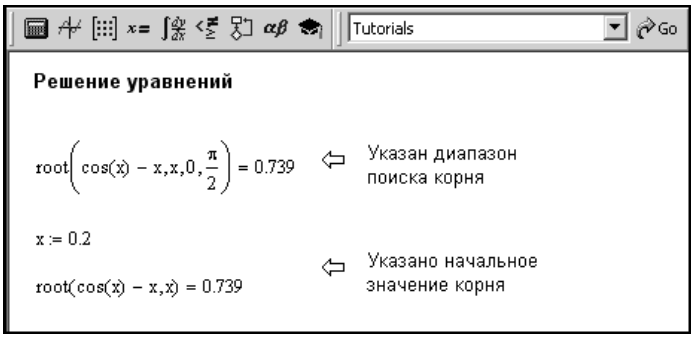


Рис. 7.1. Решение уравнения с помощью процедуры `root()`

В первом случае задается диапазон, на котором ищется корень, во втором — перед вызовом процедуры определяется начальное приближение для корня. В силу этого третий и четвертый аргументы процедуры `root()`, определяющие интервал поиска решения, не указываются. Способ вызова процедуры `root()` неявно определяет метод, используемый для вычисления корня. В качестве иллюстрации можно привести данные на рис. 7.2.

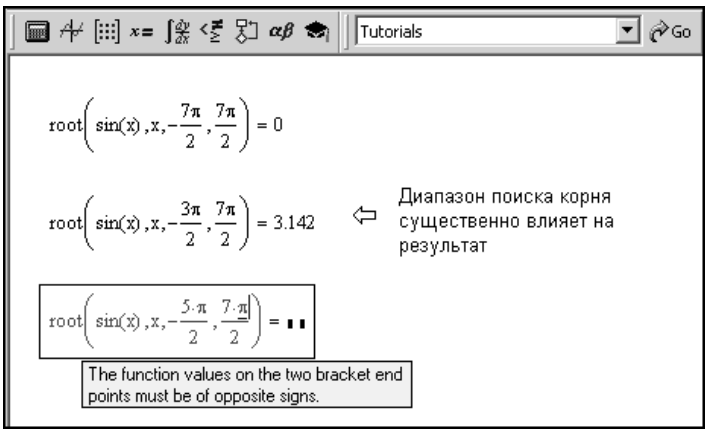


Рис. 7.2. Границы диапазона поиска корня существенно влияют на результат

Там процедура `root()` трижды вызывается для решения одного и того же уравнения, но каждый раз с новыми границами поиска решения. При этом результаты совершенно различны. В последнем случае вообще выводится сообщение об ошибке, хотя во всех трех случаях указанные диапазоны поиска однозначно накрывают очевидное нулевое решение. Дело в том, что на указанных диапазонах существует несколько решений уравнения, поэтому в зависимости от выбранного диапазона вычисляются разные корни. Случай с сообщением об ошибке также объясняется достаточно просто. Известно, что если функция непрерывна и на границах диапазона принимает значения разных знаков, то на этом диапазоне она, по крайней мере, в одной точке обращается в нуль. Это достаточное условие, но не необходимое. Если оно выполнено, процедура `root()` позволяет найти корень уравнения. В противном случае необходимы другие методы решения. Ситуация очень схожа с той, что складывается при решении уравнений методом дихотомии (половинного деления) или методом хорд. Кстати, соответствующие процедуры в случае необходимости без труда могут быть созданы с помощью программных средств Mathcad.

Задача 7.1. Метод половинного деления

На рис. 7.3. представлен фрагмент рабочего документа, в котором создается процедура вычисления корня уравнения методом половинного деления.

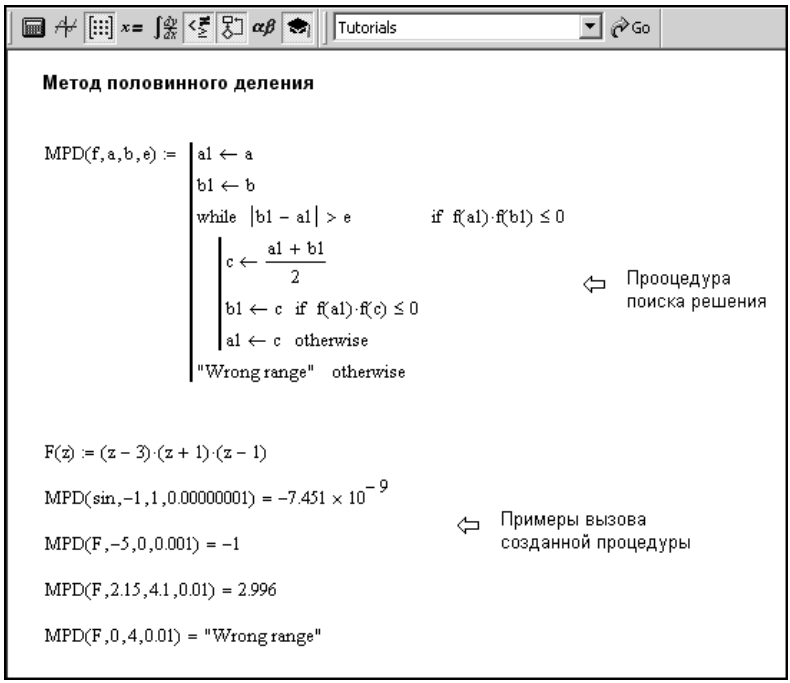


Рис. 7.3. Поиск корня методом половинного деления

Суть метода состоит в следующем. Сначала на границах диапазона поиска решения вычисляется функция, определяющая решаемое уравнение. Она должна принимать на границах значения разных знаков. Если это так, то функция вычисляется в центральной точке диапазона поиска решения (рис. 7.4).

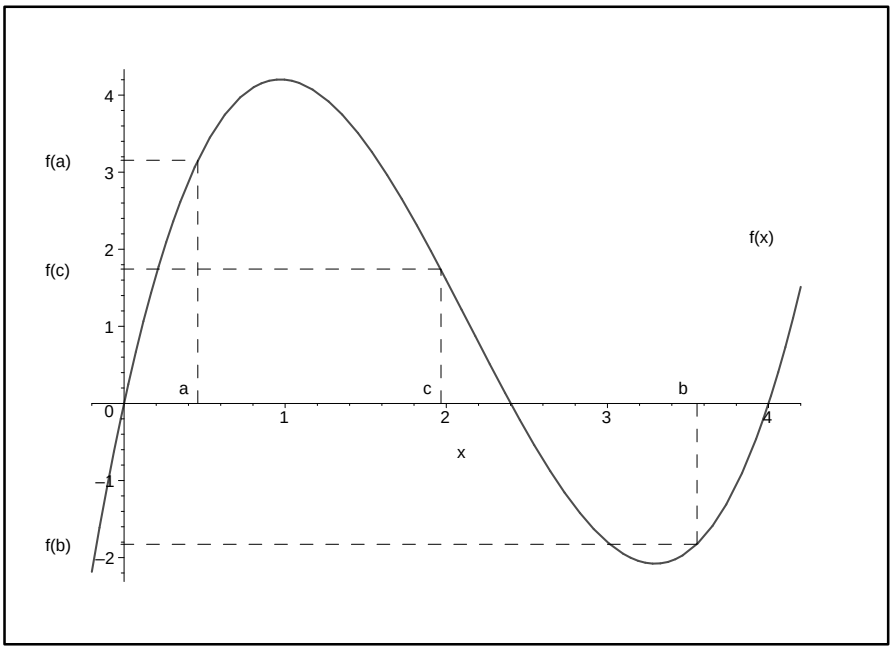


Рис. 7.4. Выбор точки по методу половинного деления

Далее в эту точку перемещается та граница диапазона, значение функции на которой по знаку соответствует значению в центральной точке. В результате диапазон поиска уменьшается вдвое, а задача сводится к предыдущей. Повторяя указанную процедуру определенное число раз, можно с заданной точностью локализовать корень уравнения. Именно этот алгоритм и реализован в приведенном примере.

У процедуры четыре аргумента: функция, определяющая уравнение; левая и правая границы диапазона, на котором ведется поиск корня, и точность вычисления корня уравнения. Первыми двумя командами в теле процедуры определяются две локальные переменные, значения которых в дальнейшем будут определять границы диапазона, на котором находится искомое решение. Вначале они совпадают со значениями левой и правой границ исходного диапазона (второй и третий аргументы процедуры). Далее следует условный

оператор. Проверяемым условием является неположительность произведения функции на границах диапазона. Если условие выполняется (т. е. произведение меньше либо равно нулю), выполняется оператор цикла `while`. При невыполнении условия (а это означает, что на границах функция отлична от нуля и имеет один знак), возвращается текстовое значение `"Wrong range"`, свидетельствующее о том, что для поиска корня на данном диапазоне процедура непригодна (что, кстати, совсем не означает, что корня на этом диапазоне нет).

Оператор `while` выполняется до тех пор, пока длина диапазона, на котором находится корень, превышает погрешность (четвертый аргумент функции). Оператор цикла в свою очередь состоит из двух операторов: определяется центральная точка диапазона и затем с помощью условного оператора одна из границ диапазона смещается в эту центральную точку. Какую именно границу следует смещать, определяем по произведению значений функции на левой границе и в центре диапазона. Если произведение не превышает нуля, смещать нужно правую границу, в противном случае — левую.

После создания процедуры приведены примеры ее применения. Следует отметить одно важное обстоятельство, касающееся аргументов процедуры. Дело в том, что в отличие от встроенной процедуры `root()`, в данном случае функция, определяющая решаемое уравнение, указывается без аргумента. Поэтому нет необходимости указывать и переменную, относительно которой решается уравнение. Это накладывает ограничение и на способ реализации уравнения. Для решения с использованием созданной процедуры правая часть уравнения (та, что приравнивается нулю) должна быть определена как функция одной переменной — именно той, относительно которой решается уравнение.

Созданная процедура применяется для решения уравнения $\sin(x) = 0$ на интервале от –1 до 1. Первым аргументом указывается, как отмечалось, название функции, т. е. `sin`.

Кроме этого, в документе на рис. 7.3 показаны примеры вычисления с разной точностью целочисленных корней полинома третьей степени. Среди прочих, там есть и пример, когда указан некорректный диапазон для поиска корня. Несмотря на то, что корней на этом диапазоне несколько, с помощью разработанной процедуры вычислить ни один из них не удастся. Для этого нужно как минимум изменить начальный диапазон.

Задача 7.2. Метод хорд

Внеся небольшие изменения, созданную ранее процедуру можно модифицировать так, чтобы она определяла корень уравнения по методу хорд. Принципиальное отличие метода хорд от предыдущего состоит в выборе пробной точки, в которую смещается одна из границ диапазона поиска решения. Эта точка

выбирается как пересечение с координатной осью хорды, проходящей через точки, определяемые значениями функции на границе (рис. 7.5).

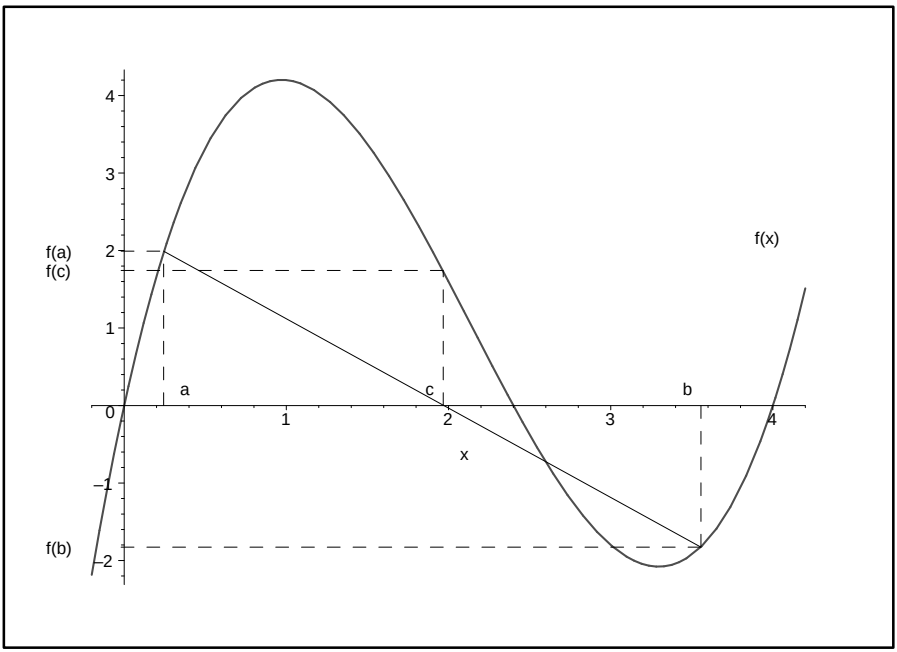


Рис. 7.5. Выбор точки по методу хорд

Так, если ищется решение уравнения $f(x) = 0$ в диапазоне значений от a до b, то пробная точка c определяется соотношением $c = \dfrac{af(b) - bf(a)}{f(b) - f(a)}$.

По сравнению с дихотомией (половинным делением), метод хорд обеспечивает ускоренную сходимость алгоритма. Пример соответствующей процедуры с примерами ее вызова приведен в документе на рис. 7.6.

В отличие от предыдущей процедуры, здесь изменена команда вычисления промежуточной точки. Кроме того, указан новый критерий для оператора цикла. В частности, уточнение корня продолжается до тех пор, пока значение функции (которое в точке-решении уравнения должно равняться нулю) превышает параметр, задающий точность. Таким образом, во вновь создаваемой процедуре четвертый ее параметр задает точность не по аргументу, а по значению функции. Кроме того, добавлена команда первоначальной инициализации промежуточной точки диапазона. Для удобства она выбрана совпадающей с правой его границей.

Метод хорд

```
MKH(f,a,b,e) := | a1 ← a
                | b1 ← b
                | c ← a
                | while |f(c)| > e                  if f(a1)·f(b1) ≤ 0
                |   | c ← (a1·f(b1) − b1·f(a1)) / (f(b1) − f(a1))
                |   | b1 ← c  if f(a1)·f(c) ≤ 0
                |   | a1 ← c  otherwise
                | "Wrong range"  otherwise
```

⇦ Прооцедура поиска решения

```
F(z) := (z − 3)·(z + 1)·(z − 1)

MKH(sin,−1,1,0.00000001) = 0

MKH(F,−5,0,0.001) = −1

MKH(F,2.15,4.1,0.01) = 2.999

MKH(F,0,4,0.01) = "Wrong range"
```

⇦ Примеры вызова созданной процедуры

Рис. 7.6. Вычисление корня уравнения методом хорд

Задача 7.3. Метод Ньютона

Два других популярных метода решения уравнений — это метод Ньютона (касательных) и метод последовательных итераций.

При решении уравнения методом Ньютона, указывается не диапазон поиска решения, а начальное приближение для искомого корня. Понятно, что это начальное значение должно по возможности наиболее точно совпадать с решением уравнения. Сама же процедура поиска состоит в последовательном уточнении решения согласно формуле $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$, где через $f'(x)$ обозначена производная функции $f(x)$ (при условии, что решается уравнение $f(x) = 0$). Индекс у переменной указывает номер итерации. Этот метод достаточно эффективен, однако использоваться может далеко не всегда, т. е. не для всякой функции $f(x)$ и не для любого приближения начальной точки. Начальное значение корня по возможности должно быть таким, чтобы функция $f(x)$ и ее вторая производная имели одинаковые знаки. Вообще же для обеспечения сходимости метода нужно, чтобы отношение максимального

значения первой и второй производных на интервале поиска решения было больше половины длины этого интервала. В общем случае вопрос о возможности применения метода Ньютона — задача нетривиальная. Заинтересованный читатель может обратиться по этому вопросу к специальной литературе. Отметим только, что практическим критерием, позволяющим определиться с применимостью метода, является существенная плавность функции $f(x)$.

Если обратиться к графической интерпретации метода, то новая точка выбирается как пересечение с координатной осью прямой, касательной к графику функции $f(x)$ в точке x_n. Эта точка служит новым приближением для корня, через нее проводится касательная и определяется следующее приближение (рис. 7.7).

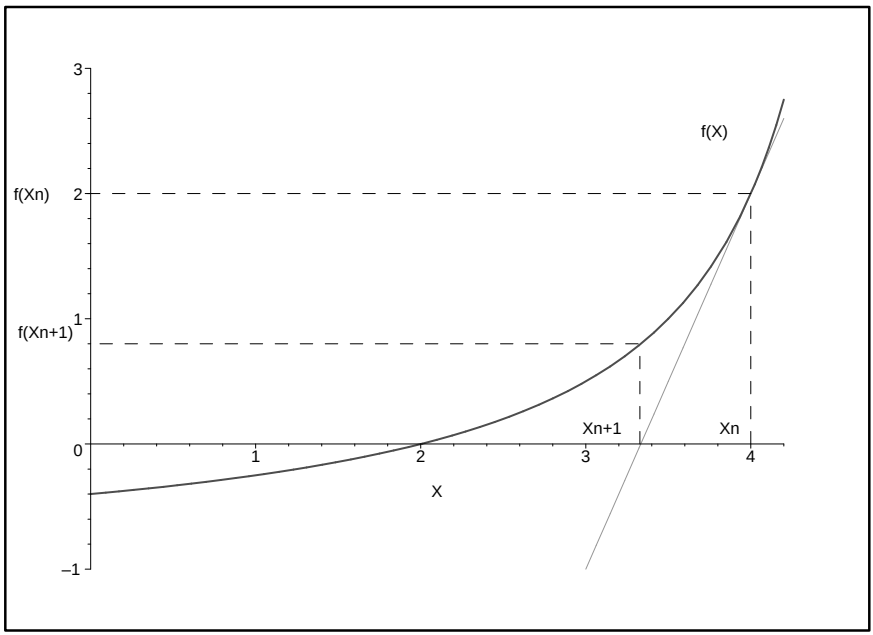


Рис. 7.7. Выбор точки по методу Ньютона

Код процедуры, реализующей метод Ньютона, приведен в документе, фрагмент которого показан на рис. 7.8.

Аргументы процедуры — функция, начальное приближение и число итераций. Вначале инициализируется локальная переменная со значением, указанным вторым аргументом процедуры. Далее следует оператор цикла, в котором последовательно выполняются две операции: вычисляется значение производной в точке и определяется новая точка. Число итераций задается третьим аргумен-

том процедуры. Вслед за описанием процедуры приведены примеры ее вызова. Не все они соответствуют критериям применимости метода, но, тем не менее, во всех случаях найдены решения уравнения. Правда, для полиномиальной функции, в зависимости от начального приближения, получаются разные корни. Связано это именно с тем, что некоторые начальные значения для решений лежат в областях, не удовлетворяющих критериям применимости метода, поэтому в результате последовательных итераций одно из приближений попадает в корректный диапазон значений и метод сходится.

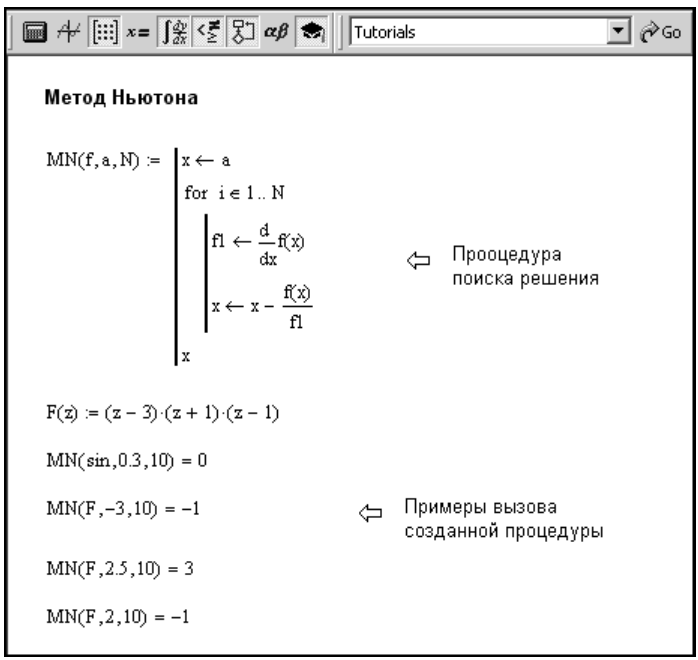


Рис. 7.8. Метод Ньютона

Задача 7.4. Метод последовательных приближений

Метод последовательных приближений применяется к уравнениям вида $x = \varphi(x)$. Значение x_{n+1} на n+1-м шаге определяется по значению x_n на n-м шаге по формуле $x_{n+1} = \varphi(x_n)$. Алгоритм реализации этого метода (рис. 7.9) достаточно прост.

Условие применимости метода — выполнение неравенства $|\varphi'(x)| < 1$ в некоторой окрестности искомого решения. Работа соответствующей процедуры проверена на примере решения уравнения $x = \sin(x)$. Несмотря на то, что начальное приближение достаточно неплохое, даже при очень большом числе

итераций точность вычисления корня оставляет желать лучшего. Как и ранее, проблема кроется в области, на которую попадает корень. Дело в том, что в точке решения, т. е. при $x = 0$, производная строго равна единице, что не соответствует критерию применимости метода. Если же данным методом решать, например, уравнение $x = 0.5\sin(x)$, проблема автоматически снимается и метод дает прекрасную сходимость (рис. 7.10).

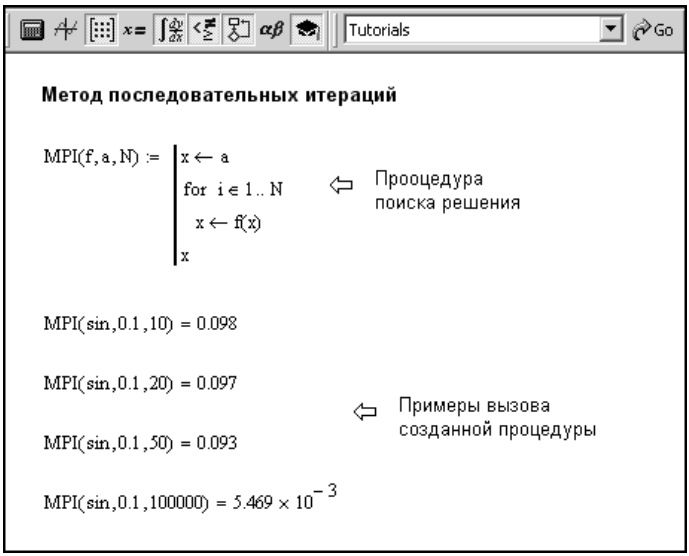


Рис. 7.9. Метод последовательных итераций

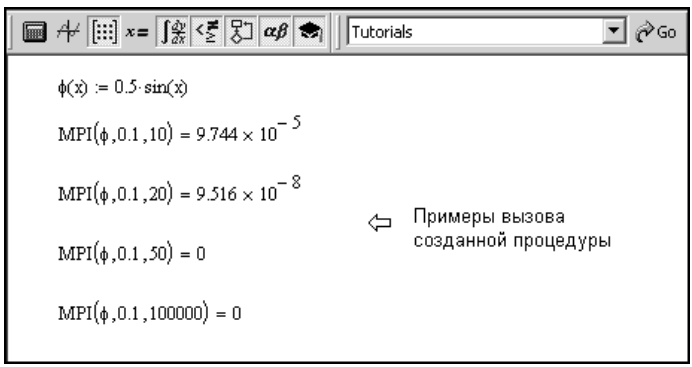


Рис. 7.10. Примеры решения уравнения методом последовательных итераций

Даже из этих простых примеров с достаточной очевидностью вытекает, что выбор метода решения уравнения — задача очень непростая. По возможности следует применять встроенные процедуры Mathcad. В этом случае обычно

проблем с подбором адекватного метода не возникает. Еще одна встроенная процедура, пригодная для решения алгебраических уравнений, — `Find()`.

Пример 7.2. Корни полиномов

Процедура `Find()` располагается в пределах специального блока, который начинается ключевым словом `Given` (водится с клавиатуры). После ключевого слова следует решаемое уравнение, а заканчивается блок вызовом процедуры `Find()` с аргументом — переменной, относительно которой решается уравнение. Перед блоком указывают начальное приближение для корня. Пример вызова процедуры `Find()` представлен на рис. 7.11.

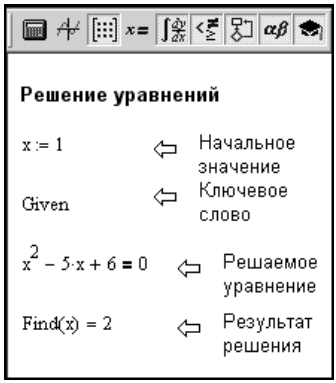


Рис. 7.11. Решение уравнения с помощью функции `Find()`

В качестве решаемого уравнения указан полином второй степени. Как видим, найден один из его корней. Для поиска корней полиномиальной функции в Mathcad предусмотрена специальная процедура `polyroots()`, аргументом которой указывают вектор коэффициентов полинома. Пример вызова процедуры показан на рис. 7.12.

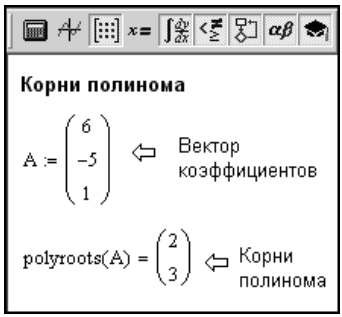


Рис. 7.12. Вычисление корней полинома

Вектор коэффициентов должен при этом быть определен как вектор-столбец. Коэффициенты указываются в порядке возрастания степени.

Системы уравнений

Упомянутая процедура `Find()` удобна и для решения систем нелинейных алгебраических уравнений. Разумеется, число аргументов у процедуры `Find()` теперь должно соответствовать числу уравнений в системе. Для каждой переменной перед началом блока `Given` указывают начальное значение.

Пример 7.3. Решение систем уравнений

Пример вызова этой процедуры для решения системы алгебраических уравнений показан в документе на рис. 7.13.

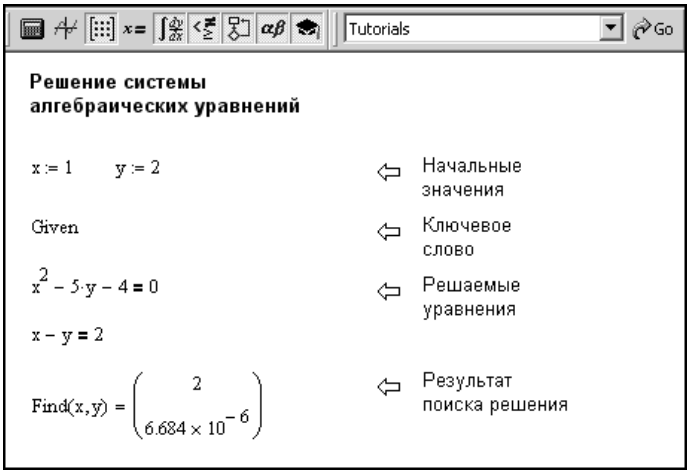


Рис. 7.13. Решение системы уравнений

Система может иметь несколько решений. В этом случае решение, которое будет найдено, во многом определяется начальным приближением. Такая ситуация имеет место и в рассматриваемом примере. На рис. 7.14 для той же системы уравнений найдено еще одно решение.

Это решение удалось найти, изменив начальное приближение для переменных, относительно которых решается система. Здесь четко прослеживается проблема, характерная для нелинейных систем: даже если некоторое решение найдено, нет гарантии, что оно единственное. Общих методов при этом не существует.

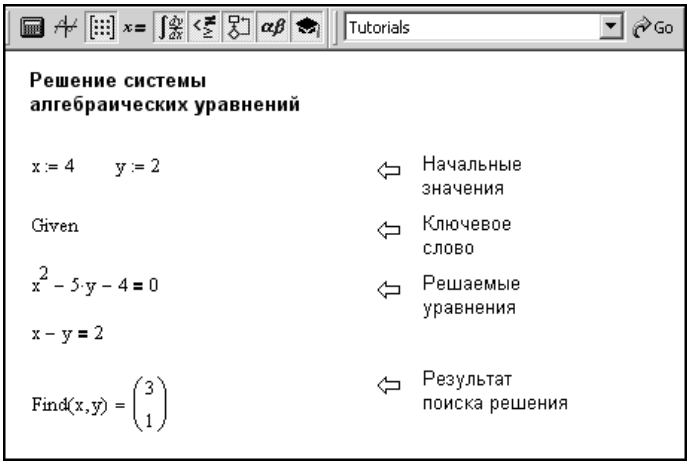


Рис. 7.14. Изменение начальных значений приводит к изменению результата решения системы уравнений

Если решается система линейных уравнений, то такой проблемы не возникает. В Mathcad на этот случай есть специальная процедура `lsolve()`, которая по матрице коэффициентов и вектору правых частей уравнений позволяет найти решение. Если в матричном виде решаемая система уравнений записана как $AX = B$, где A — квадратная матрица коэффициентов, а X и B — векторы неизвестных переменных и правых частей уравнений, соответственно, то решение может быть найдено с помощью команды `lsolve(A,B)`.

Пример 7.4. Система линейных уравнений

Пример вызова процедуры `lsolve` показан на рис. 7.15.

Однако сразу следует отметить, что для системы линейных уравнений решение легко находится и без специальных процедур. Действительно, вектор неизвестных переменных X может быть определен как произведение матрицы, обратной к матрице коэффициентов, на вектор правых частей уравнений, т. е. $X = A^{-1}B$. Именно этот прием проиллюстрирован на рис. 7.15 при вычислении решений системы уравнений без применения процедуры `lsolve()`.

Ранее отмечалось, что многие рассмотренные методы решения уравнений с минимальными изменениями пригодны и для решения систем. Кратко упомянем некоторые из них, оставив без обсуждения вопрос об условиях их применимости. Читатель при желании может без труда найти нужную информацию по этому поводу в справочной и научной литературе соответствующего профиля.

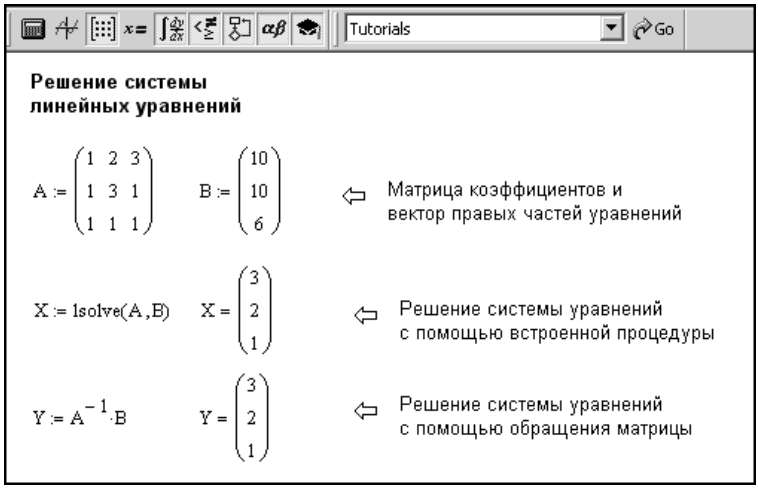


Рис. 7.15. Решение системы линейных уравнений

Предположим, что система N уравнений может быть представлена в виде $x_i = \varphi_i(x_1, x_2, ..., x_N)$, $i = 1, 2, ..., N$. Если ввести вектор неизвестных переменных $\vec{x} = (x_1, x_2, ..., x_N)^T$ и вектор-функцию $\vec{\varphi}(\vec{x}) = (\varphi_1(\vec{x}), \varphi_2(\vec{x}), ..., \varphi_N(\vec{x}))^T$ (символ T обозначает операцию транспонирования), то система формально может быть представлена в виде $\vec{x} = \vec{\varphi}(\vec{x})$. Применение метода последовательных итераций для решения такой системы подразумевает вычисление приближения $\vec{x}^{n+1}$ для вектора неизвестных переменных на n+1-м шаге по известным значениям этих переменных на n-м шаге по формуле $\vec{x}^{n+1} = \vec{\varphi}(\vec{x}^n)$. Эта схема может быть легко реализована с помощью программных методов Mathcad. Строго говоря, при этом даже не придется видоизменять код процедуры, созданной ранее для решения уравнений методом итераций.

Задача 7.5. Решение системы уравнений методом итераций

Пример решения системы представлен в документе на рис. 7.16.

При решении систем уравнений следует только корректно описать вектор-функцию, которая затем будет указываться в качестве первого аргумента процедуры. На рис. 7.16 решается система из двух уравнений: $x = 0.85 + \cos(y)$ и $y = -1.32 + \sin(x)$. Начальным приближением служат нулевые значения переменных x и y. Эти значения представлены в виде компонентов вектора. Вектор присвоен в качестве значения переменной a. Вектор-функция описывается следующим образом: после оператора присваивания указан вектор, компоненты которого выражаются через элементы

вектора, указанного аргументом функции. О том, что аргумент функции является вектором, свидетельствуют только индексы в правой части соотношения, инициализирующего вектор-функцию. Кстати, стоит напомнить, что по умолчанию индексирование элементов вектора начинается с нуля.

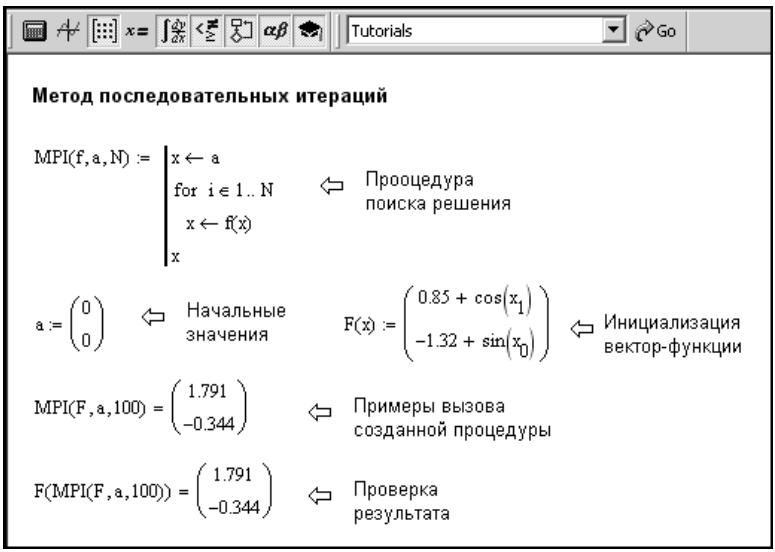


Рис. 7.16. Решение системы уравнений методом последовательных приближений

Для решения системы уравнений вызывается уже известная читателю процедура (для удобства ее код приведен на рис. 7.16) с тремя аргументами: описанном в рабочем документе вектора-функцией, вектором начальных значений и числом итераций. Кроме этого, проверяется результат поиска решения. Предположим, решение найдено верно. Тогда если полученный в результате вектор указать аргументом вектора-функции, должны получить тот же вектор. Для этого в рабочем документе вызывается вектор-функция. Ее аргументом указывается процедура поиска решения системы уравнений. Первый аргумент этой процедуры — вектор-функция. Как видно из рис. 7.16, результат вполне приемлем.

Задача 7.6. Решение системы уравнений методом Ньютона

Для решения систем уравнений часто применяется метод Ньютона. Как и в предыдущем случае, удобнее всего этот метод описать на основе векторного представления. Так, если решаемая система имеет вид $\vec{f}(\vec{x}) = 0$, при начальном значении $\vec{x} = \vec{a}$, то итерационное значение на n+1-м шаге $\vec{x}^{n+1}$ может быть получено по итерационному значению на n-м шаге по формуле

$\vec{x}^{n+1} = \vec{x}^n - A^{-1}(\vec{x})\vec{f}(\vec{x}^n)$, где $A^{-1}(\vec{x})$ есть матрица, обратная к матрице производных от вектора-функции $\vec{f}(\vec{x})$ по каждому из элементов вектора-аргумента функции. На рис. 7.17 приведен фрагмент документа, в котором описана процедура решения системы из двух уравнений. Процедура несколько примитивна, но, с другой стороны, достаточно наглядна.

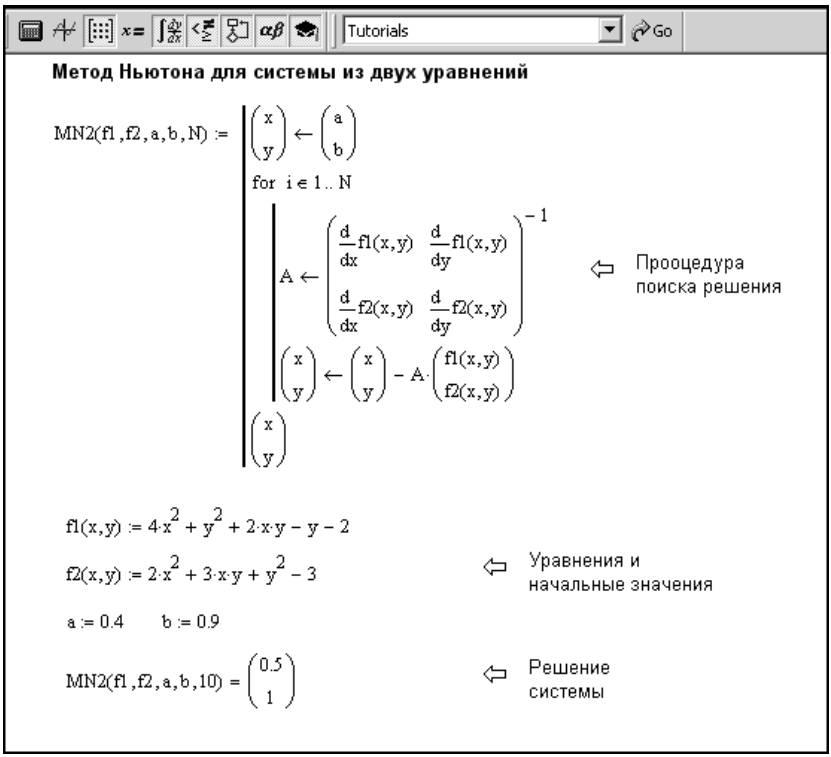


Рис. 7.17. Решение системы из двух уравнений методом Ньютона

У процедуры пять аргументов: две функции, определяющие систему уравнений, два начальных значения для каждой из переменных, относительно которых система решается, а также число итераций. Вводятся две локальные переменные (как компоненты вектора), и далее в рамках оператора цикла определяется матрица производных и уточняется текущее значение переменных. В нижней части рис. 7.17 показан пример вызова этой процедуры для решения системы из двух уравнений: $4x^2 + y^2 + 2xy - y - 2 = 0$ и $2x^2 + 3xy + y^2 - 3 = 0$.

Задача 7.7. Метод градиентного спуска

Еще один достаточно популярный способ решения системы уравнений — метод скорейшего (градиентного) спуска. Его суть сводится к следующему. Допустим, нужно решить систему из N уравнений $f_i(x_1, x_2, ..., x_N) = 0$, $i = 1, 2, ..., N$. Эта задача может быть сформулирована как поиск минимума функции $\Phi(x_1, x_2, ..., x_N) = \sum_{i=1}^{N} f_i^2(x_1, x_2, ..., x_N)$. Действительно, поскольку функция равна сумме квадратов действительных величин, то меньше нуля она быть не может. Нулевое же значение реализуется тогда и только тогда, когда каждое из слагаемых равно нулю. Если на n-м шаге значение переменных равно $x_i^{(n)}$, то на n+1-м шаге значение переменных определяется по формуле $x_i^{(n+1)} = x_i^{(n)} - \lambda_n \frac{\partial \Phi(x_1^{(n)}, x_2^{(n)}, ..., x_N^{(n)})}{\partial x_i}$. То же выражение в векторном виде может быть записано как $\vec{x}^{(n+1)} = \vec{x}^{(n)} - \lambda_n \vec{\nabla}\Phi(\vec{x}^{(n)})$, где обозначено $\vec{\nabla}\Phi(\vec{x}) = \left(\frac{\partial \Phi(x_1, x_2, ..., x_N)}{\partial x_1}, \frac{\partial \Phi(x_1, x_2, ..., x_N)}{\partial x_2}, ..., \frac{\partial \Phi(x_1, x_2, ..., x_N)}{\partial x_N} \right)$, т. е. градиент функции $\Phi(\vec{x})$. Параметр λ_n на каждом шаге выбирается из условия минимума функции $\psi_n(\lambda) = \Phi(\vec{x}^{(n)} - \lambda \vec{\nabla}\Phi(\vec{x}^{(n)}))$ (что фактически означает решение уравнения $\psi_n'(\lambda) = 0$). Данная функция определяет значение исходного минимизируемого выражения как зависимость от параметра λ. Это совсем нетривиальная задача, по уровню сложности сравнимая (а может и превосходящая) задачу по решению системы уравнений. Поэтому на практике обычно значение параметра λ вычисляют приближенно. В частности, можно использовать формулу $\lambda = \frac{1}{2} \frac{\left(A^T(\vec{x}^{(n)}) \vec{f}(\vec{x}^{(n)}) \right)^2}{\left(A(\vec{x}^{(n)}) A^T(\vec{x}^{(n)}) \vec{f}(\vec{x}^{(n)}) \right)^2}$, где $A(\vec{x})$ — матрица производных от вектора-функции $\vec{f}(\vec{x})$ по ее аргументам. Реализация метода градиентного спуска для системы из двух уравнений показана на рис. 7.18.

В теле оператора цикла, который составляет основу всей процедуры, определяется вектор-функция, задающая систему уравнений, а также матрица производных. Эти объекты необходимы далее для определения параметра λ. Поскольку в этом случае скалярно перемножаются векторы, то в числителе и знаменателе дроби вместо оператора возведения в квадрат указан оператор скалярного произведения векторов. Наконец, после вычисления λ можно рассчитать новое приближение для переменных, относительно которых

решается система уравнений. В нижней части документа методом градиентного спуска решается система уравнений, которая ранее решалась методом Ньютона. Как видим, результаты совпадают.

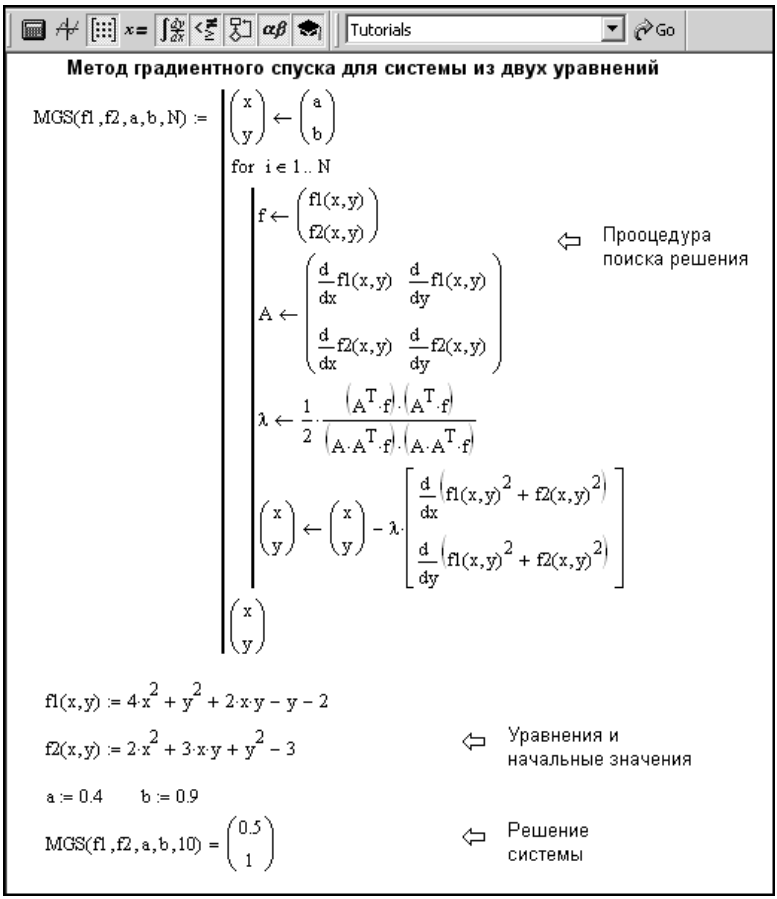


Рис. 7.18. Решение системы уравнений методом градиентного спуска

Интерполяция функций

Часто при проведении научных исследований, особенно связанных с обработкой эмпирических данных, приходится иметь дело с функциональными зависимостями, параметры которых неизвестны или информация о них неполная. Эту проблему можно решать несколькими путями. Достаточно популярный, распространенный и эффективный метод — интерполяция. В общем случае задача формулируется следующим образом. Представим ситуацию,

когда функциональная зависимость реализована в виде таблицы. Другими словами, значения неизвестной функции известны только в некоторых точках (узлах). В результате интерполяции строится такая функциональная зависимость, для которой значения в узлах совпадают со значениями неизвестной табулированной функции. Обычно выбирается функция определенного вида, которая помимо аргумента (аргументов) зависит еще и от некоторых параметров, рассчитываемых так, чтобы в узловых точках функция принимала определенные значения. Часто базовой интерполяционной функцией служит полиномиальная зависимость. В этом случае говорят об интерполяционных полиномах. Подгоночными параметрами являются коэффициенты полинома. Полином может быть один для всего диапазона, в котором лежат узловые точки, или диапазон разбивается на поддиапазоны, на каждом из которых полиномиальные зависимости разные. Помимо условия на узловые точки появляется еще и требование непрерывности функции и ее производных (до определенного порядка). Такая интерполяция называется интерполяцией сплайнами. Для выполнения интерполяции в Mathcad существуют встроенные процедуры, можно создавать и собственные программы. Остановимся подробнее на этих вопросах. Начнем со сплайн-интерполяции.

Интерполяция сплайнами

Для выполнения сплайн-интерполяции в Mathcad есть набор специальных встроенных утилит. Одна из них — `interp()`, у которой четыре аргумента. Три последних особых комментариев не требуют и их наличие вполне объяснимо. Это соответственно вектор узловых точек, вектор значений функции в узловых точках и переменная, для которой вычисляется интерполяционное значение. Первым аргументом указывается вектор вторых производных. Этот вектор существенно зависит от того, какими сплайнами выполняется интерполяция и вычисляется одной из процедур `cspline()`, `pspline()` или `lspline()`. Аргументами всех трех процедур указываются вектор узловых точек и вектор значений функции в узловых точках. Процедурой `cspline()` возвращается вектор вторых производных, соответствующих кубическим сплайнам (т. е. полиномы третьей степени) в граничных точках диапазона интерполяции. Чтобы в граничных точках сплайн-полиномы были второй степени, выбирают процедуру `pspline()`. Линейным полиномам на границе соответствует процедура `lspline()`.

Задача 7.8. Интерполяция сплайнами

Примеры применения перечисленных утилит для построения сплайн-интерполяции представлены на рис. 7.19.

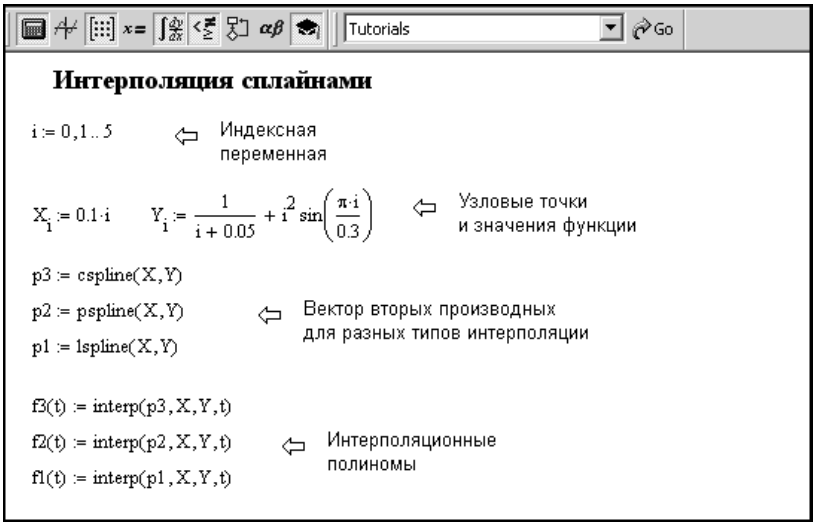


Рис. 7.19. Использование встроенных процедур Mathcad для выполнения сплайн-интерполяции

Сначала строятся базовые точки. Поскольку пример иллюстративный, то для создания вектора узловых точек и значений функции в них выбрана произвольная функция (с осцилляциями — для большей наглядности). Потом по этим базовым точкам строятся три сплайн-интерполяции — с кубическими, квадратичными и линейными сплайнами в граничных точках. Результат показан на рис. 7.20.

Как легко заметить, разница не очень существенна. Проблема связана с тем, какие условия выбирать на границе диапазона интерполирования. Дело в том, что условий сшивки сплайнов в узловых точках недостаточно для того, чтобы однозначно определить интерполяционную зависимость. Поэтому к ним добавляют еще и условия на границе, которые выбираются пользователем.

Нередко интерполяция должна быть линейной. Это означает, что точки просто соединяются прямыми линиями. Если речь идет о графическом представлении зависимостей, то соединения точек прямыми можно добиться путем настроек параметров графика. Если нужно вычислить значение функции в точке, которая не является узловой, лучше все же воспользоваться специальной процедурой. Для линейной интерполяции в Mathcad пригодна процедура `linterp()`. У нее три аргумента: вектор узловых точек, вектор значений функции в этих точках, а также точка, в которой вычисляется интерполяционное значение. На рис. 7.21 показан пример вызова этой процедуры.

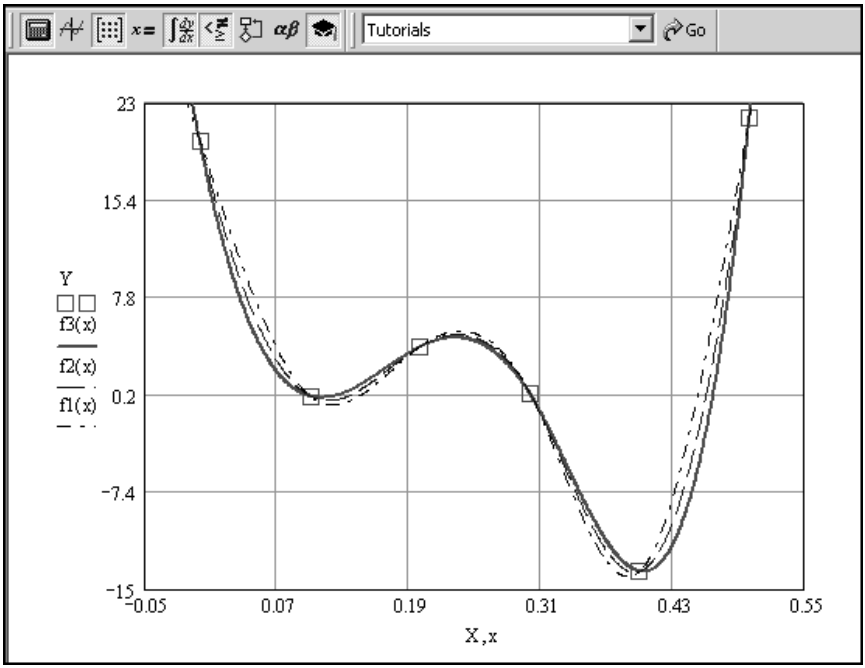


Рис. 7.20. Интерполяция различными сплайнами

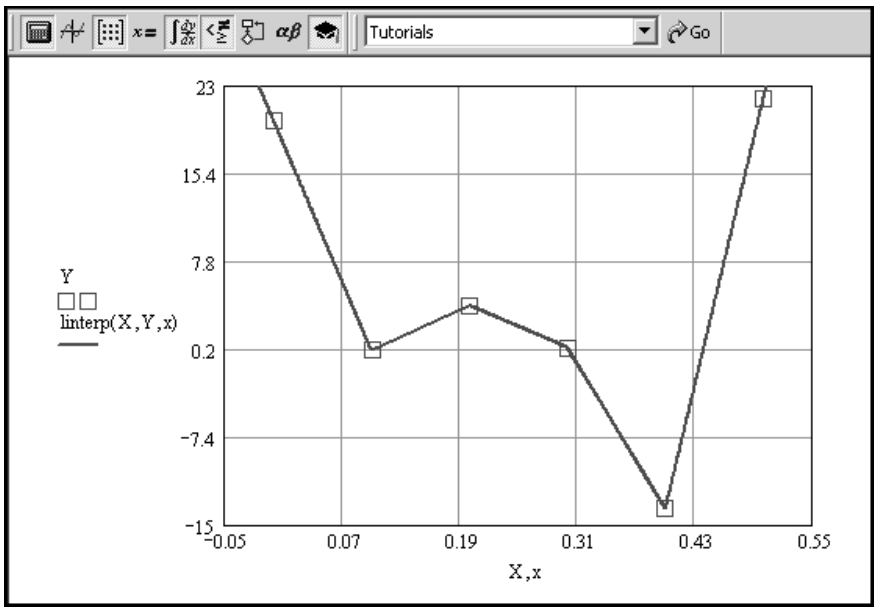


Рис. 7.21. Линейная интерполяция

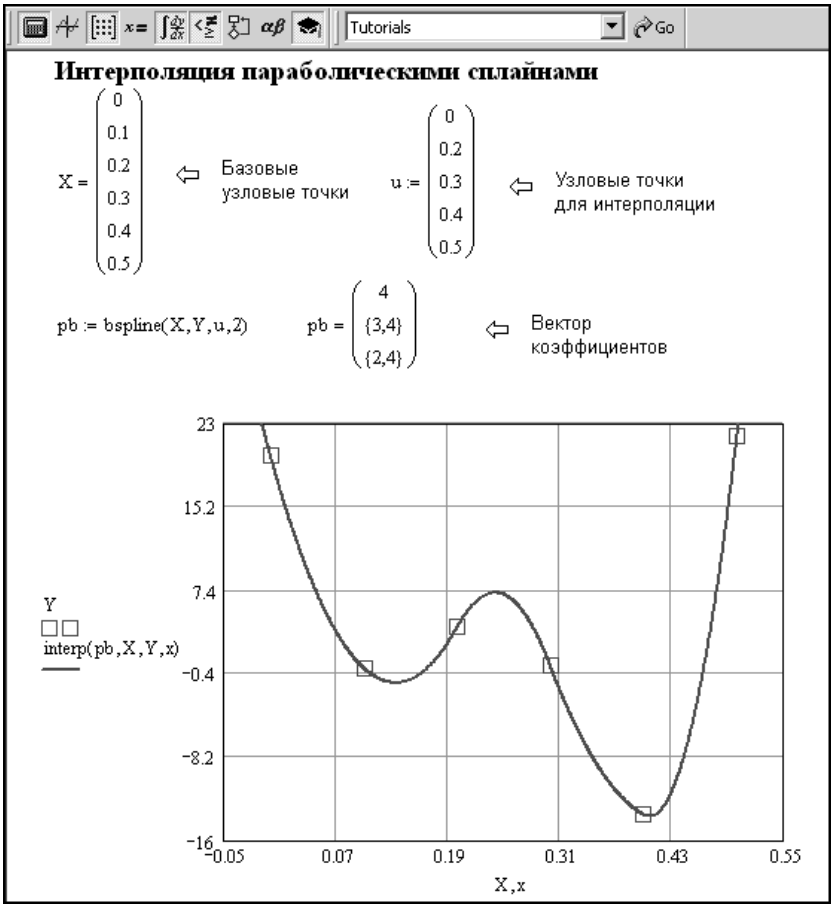


Рис. 7.22. Интерполяция параболическими сплайнами

В качестве исходных выбраны те же самые данные, что и в предыдущем примере. Здесь интерполяционная процедура указана непосредственно в области графика.

В Mathcad есть специальная процедура `bspline()` для выполнения интерполяции сплайн-полиномами, отличными от кубических. Она возвращает вектор коэффициентов, который затем указывают первым аргументом процедуры `interp()`. У процедуры `bspline()` четыре аргумента: два стандартных вектора узловых точек и значений функции в узловых точках, вектор базовых узлов, по которым строится интерполяционный полином, а также целое число в диапазоне от 1 до 3 включительно, определяющее степень интерполяционных полиномов. Третий аргумент (вектор узлов для выполнения интерполяции) должен содержать меньшее количество элементов, чем вектор

базовых узлов, указанный первым аргументом. Этих точек должно быть меньше на число, равное степени полиномов (четвертый аргумент процедуры) минус единица. Первый элемент этого вектора должен быть не больше первого элемента вектора базовых узловых точек, а последний элемент — не меньше последнего элемента вектора базовых узлов. Пример выполнения интерполяции параболическими сплайнами представлен на рис. 7.22.

В документе сначала определяется вектор с точками для выполнения интерполяции. Поскольку выбраны полиномы второй степени, то он должен содержать на один элемент меньше, чем вектор базовых узловых точек. Далее процедура создания интерполяционной зависимости достаточно стереотипна.

Следует заметить, что процедуры `cspline()`, `pspline()` или `lspline()` пригодны для выполнения двумерной интерполяции. Тогда аргументами функций указывают не векторы узловых точек и значений функции, а квадратные матрицы. Далее, как и в случае обычной (одномерной) интерполяции, вызывают процедуру `interp()`.

Интерполяционный полином Лагранжа

Популярна интерполяция численных данных одной общей функциональной зависимостью полиномиального вида. Существует несколько способов построения полинома. В соответствии с этим говорят об интерполяционном полиноме Ньютона или, например, Лагранжа. Однако и в том и в другом случае получаем один и тот же полином. Различны только процедуры его построения. Здесь рассмотрим способ построения интерполяционного полинома по методу Лагранжа.

Предположим, что в узловых точках $\{x_i\}$ функция должна принимать значения $\{y_i\}$, $i = 0,1,2,...,N$. Интерполяционный полином строится так, чтобы в узловых точках значения, вычисленные на его основе, совпадали с табличными значениями функции $\{y_i\}$. Легко показать, что степень такого полинома (для однозначного его определения) должна быть на единицу меньше числа узловых точек.

Если интерполяционное выражение выбрать в виде $L(x) = \sum_{i=0}^{N} y_i \varphi_i(x)$, где базисные функции $\varphi_i(x) = \prod_{k=0,k\neq i}^{N} (x - x_k) \Big/ \prod_{k=0,k\neq i}^{N} (x_i - x_k)$, то это будет именно тот полином, что нужен. Действительно, он имеет степень N (число узловых точек равно при этом $N+1$) и в узловых точках принимает правильные значения.

Исходя из единственности полинома, удовлетворяющего всем этим условиям, и следует с неизбежностью, что это искомый интерполяционный полином.

Задача 7.9. Интерполяционный полином Лагранжа

Реализация соответствующей процедуры в Mathcad показана в документе на рис. 7.23.

Интерполяционный полином Лагранжа

```
L(x,y,t) := | N ← rows(x) − 1
            | S ← 0
            | for i ∈ 0..N
            |   | s_i ← 1
            |   | for k ∈ 0..N
            |   |   s_i ← s_i · (t − x_k)/(x_i − x_k)  if k ≠ i
            |   | S ← S + y_i · s_i
            | S
```

⇦ Процедура вычисления полинома Лагранжа

$N := 10$

$i := 0,1..N$ ⇦ Индексная переменная

$X_i := 0.1 \cdot i \qquad Y_i := \frac{1}{i+0.05} + i\sin\left(\frac{\pi \cdot i}{0.3}\right)$ ⇦ Узловые точки и значения функции

X =

	0
0	0
1	0.1
2	0.2
3	0.3
4	0.4
5	0.5
6	0.6
7	0.7
8	0.8
9	0.9
10	1

Y =

	0
0	20
1	0.086
2	2.22
3	0.328
4	-3.217
5	4.528
6	0.165
7	-5.92
8	7.052
9	0.11
10	-8.561

$L(X,Y,0) = 20$

$L(X,Y,0.1) = 0.086$

$L(X,Y,0.4) = -3.217$ ⇦ Проверка значений

$L(X,Y,0.6) = 0.165$

$L(X,Y,0.8) = 7.052$

$L(X,Y,1) = -8.561$

Рис. 7.23. Процедура вычисления полинома Лагранжа

Процедура имеет три аргумента: векторы узловых точек и значений функции, а также значение точки, в которой вычисляется интерполяционное значение. Первой командой в теле процедуры вычисляется степень интерполяционного полинома, которая, как отмечалось, на единицу меньше числа узловых точек. После этого нулевым начальным значением инициализируется переменная, в которую будет записываться интерполяционная сумма. Основу процедуры составляют два вложенных оператора цикла. Индексная переменная первого

цикла определяет номер слагаемого в интерполяционной сумме (фактически, это номер узла и базисной функции). Индексная переменная внутреннего цикла является переменной произведения. Во внутреннем цикле также инициализируется локальная переменная, но с начальным значением 1. В эту переменную записывается значение базисной функции. Такие базисные функции, будучи умноженными на значение функции в соответствующем узле, суммируются (внешний цикл), в результате чего и получаем интерполяционный полином. В процессе перебора значений переменной внутреннего цикла локальная переменная умножается на разницу переменной, в которой вычисляется полином, и узловой точки, а также делится на разницу значений узловых точек с индексами, совпадающими со значениями индексных переменных. Эта операция выполняется только в том случае, если переменные внешнего и внутреннего циклов имеют различные значения. Далее изменяется локальная переменная для интерполяционной суммы: к ней добавляется вычисленное значение базисной функции, умноженное на значение функции в узловой точке.

В нижней части документа на рис. 7.23 генерируется последовательность узловых точек и значений функции в этих точках. Далее значения соответствующих векторов отображаются вместе со значениями интерполяционного полинома в узловых точках. Нет ничего удивительного в том, что эти величины совпадают со значениями функции в узловых точках. На рис. 7.24 показаны точки, по которым выполнялась интерполяция, а также отображен интерполяционный полином.

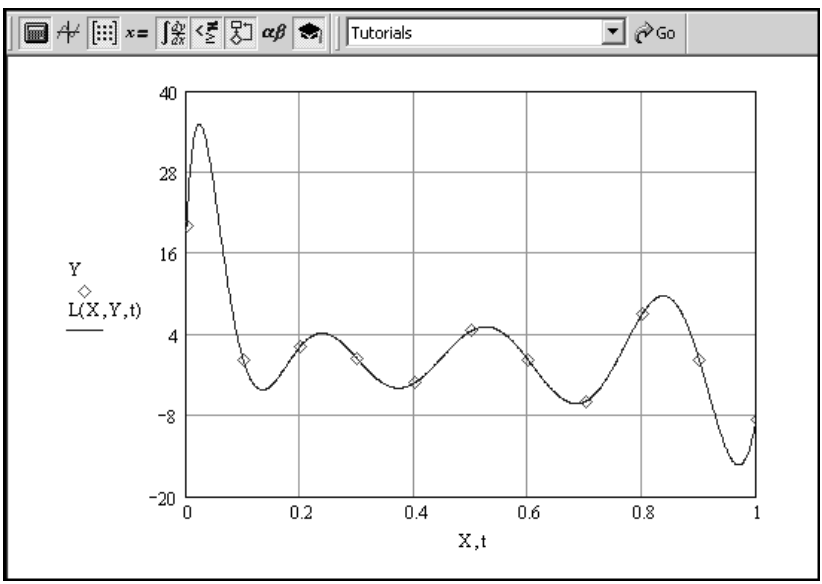


Рис. 7.24. Интерполяция Лагранжа

Хотя точки на графике разбросаны не очень сильно, на границах интерполяционный полином дает существенные осцилляции. Именно это заставляет искать другие методы интерполяции. С другой стороны, рассмотренная сплайн-интерполяция хотя и решает проблему с осцилляциями на границе, имеет тот недостаток, что интерполяционная функция является кусочно-гладкой. Если в дальнейшем придется вычислять производные от интерполяционной зависимости, то, начиная с некоторого порядка производных, последние окажутся разрывными.

Численное решение дифференциальных уравнений

Для решения дифференциальных уравнений в численном виде пользователям Mathcad предлагается ряд удобных процедур. Тем не менее, для успешного применения их на практике требуется иметь хотя бы минимальные познания в области численного решения дифференциальных уравнений. Как правило, процедуры для решения дифференциальных уравнений формально соответствуют основным математическим методам, широко известным специалистам в данной области. Другими словами, если читатель хотя бы немного ориентируется в математической сути базовых методов, назначение процедуры Mathcad ему становится понятным уже по названию, поскольку в нем, как правило, отражен исходный математический метод. С одной стороны это хорошо, поскольку позволяет специалистам выбирать именно то, что нужно — процесс решения практически полностью контролируется пользователем. С другой стороны, далеко не у каждого есть время и желание разбираться в особенностях того или иного алгоритма. Многие пользователи предпочитают получать практический результат с минимальными затратами сил, и во многом они правы. В этом смысле хочется выделить достаточно универсальную процедуру `Odesolve()`. В большинстве ситуаций дифференциальное уравнение или система уравнений могут быть решены с помощью этой процедуры, причем сам процесс решения достаточно прост.

Общий принцип работы с процедурой `Odesolve()` состоит в следующем. В рамках блока `Given` задается решаемое уравнение (или система уравнений) с начальными условиями. Заканчивается блок командой вызова процедуры `Odesolve()` с соответствующими аргументами. Таким образом, в численном виде определяется искомая функциональная зависимость (или зависимости).

Пример 7.5. Решение дифференциального уравнения

Простой пример вызова процедуры `Odesolve()` для решения неоднородного дифференциального уравнения второго порядка $x''(t) + x(t) = \sin(t)$ с нуле-

выми начальными условиями для функции и ее производной показан в документе на рис. 7.25.

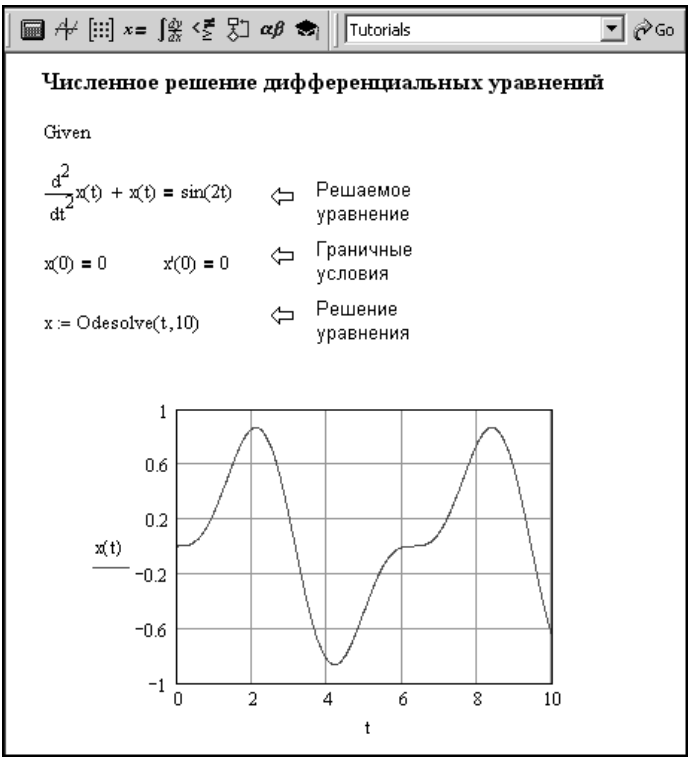


Рис. 7.25. Решение дифференциального уравнения с помощью процедуры `Odesolve()`

Отметим, что при вводе уравнения для начального значения производной символ дифференцирования в виде штриха вводился с помощью комбинации клавиш <Ctrl>+<F7>. Что касается непосредственно процедуры `Odesolve()`, то в данном случае у нее два аргумента. Первый — переменная, которая служит аргументом искомой функции. Второй аргумент определяет верхнюю границу (нижняя равна нулю) по аргументу искомой функции. Решение ищется в соответствующем диапазоне. Результат, возвращаемый процедурой `Odesolve()`, присваивается в качестве значения функции. Функция указывается без аргумента (фактически, аргумент указан при вызове процедуры `Odesolve()`) — только название.

Найденную в численном виде в результате решения дифференциального уравнения функцию можно протестировать по отдельным точкам (вызывая найденную функцию и указывая значение аргумента), но для наглядности это

лучше все же сделать графически. В нижней части рис. 7.25 приведен график, который строится стандартными методами, описанными ранее. При решении системы дифференциальных уравнений ситуация несколько сложнее.

Пример 7.6. Решение системы дифференциальных уравнений

Пример решения системы уравнений представлен на рис. 7.26. Системы решаются практически так же, как и отдельные уравнения. В рамках блока `Given` вводятся решаемые уравнения и начальные условия к ним. После этого вызывают процедуру `Odesolve()`. Отличие по сравнению с предыдущим случаем состоит в том, что теперь аргументов у процедуры не два, а три. Первый аргумент — вектор с названиями функций, относительно которых решается система уравнений (вторым аргументом функции указываем переменную, а третьим — верхнюю границу ее изменения).

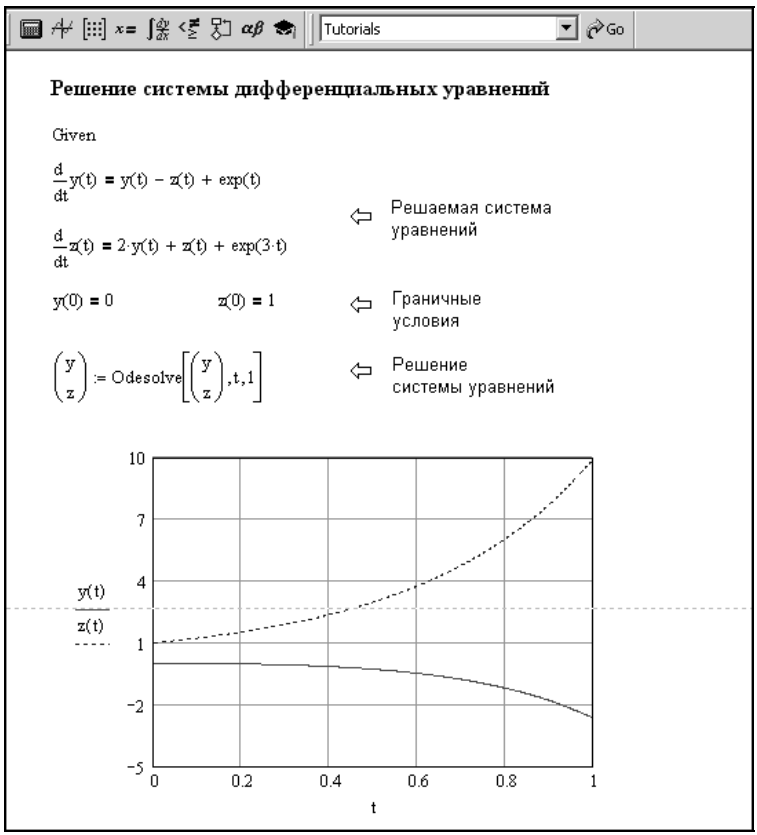


Рис. 7.26. Решение системы дифференциальных уравнений с помощью процедуры `Odesolve()`

В нижней части документа на рис. 7.26 представлены графики найденных функций.

Пример 7.7. Сведение уравнения к системе

Как известно, один из методов решения дифференциальных уравнений высокого порядка — сведение такого уравнения к системе уравнений первого порядка. Делается это достаточно просто: вместо производных порядка, выше первого, вводятся новые функции. Дополнительные уравнения в полученной системе уравнений получаются из тождеств, согласно которым вводились новые функции. В качестве иллюстрации рассмотрим пример 7.5 (решение отдельного неоднородного уравнения второго порядка), однако решать его будем сведением к системе двух уравнений. Документ показан на рис. 7.27.

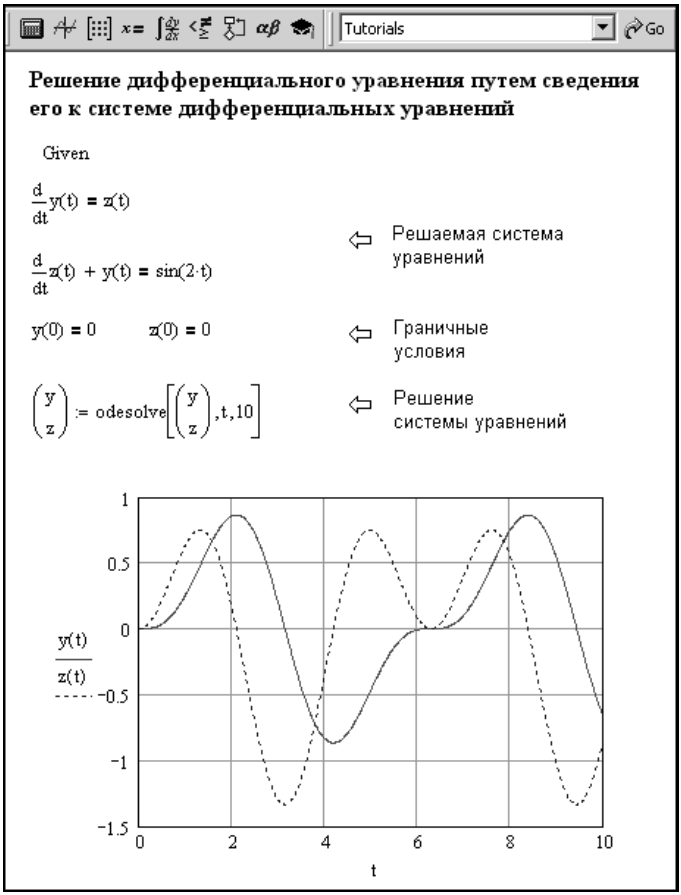


Рис. 7.27. Решение дифференциального уравнения путем сведения его к системе уравнений

Напомним, что исходное уравнение — $y''(t) + y(t) = \sin(t)$. Оно сводится к системе двух уравнений первого порядка: $y'(t) = z(t)$ и $z'(t) + y(t) = \sin(2t)$. Начальные условия для каждой из функций нулевые. Именно эта система и решалась в документе на рис. 7.27. Особенность решения, по сравнению с примером 7.5, состоит в том, что в результате получаем не только искомую функцию, но и ее производную (т. е. функцию $z(t)$). Часто это бывает крайне удобным.

Сделаем еще несколько замечаний относительно процедуры `Odesolve()`. Что касается ее аргументов, то последним из них можно указывать число шагов для поиска решения. Как отмечалось ранее, уравнения могут решаться различными математическими методами. По умолчанию в рамках процедуры `Odesolve()` решение ищется по методу Рунге-Кутта с переменным шагом. Можно задать адаптивный метод или метод решения жестких уравнений. Для этого нужно выделить процедуру `Odesolve()` и щелкнуть правой кнопкой мыши. В раскрывающемся списке есть несколько команд, позволяющих выбрать требуемый метод решения дифференциальных уравнений: команду `Adaptive` выбирают для адаптивного метода, а команда `Stiff` полезна при решении жестких уравнений и систем. Процедура выбора команды для метода решения показана на рис. 7.28.

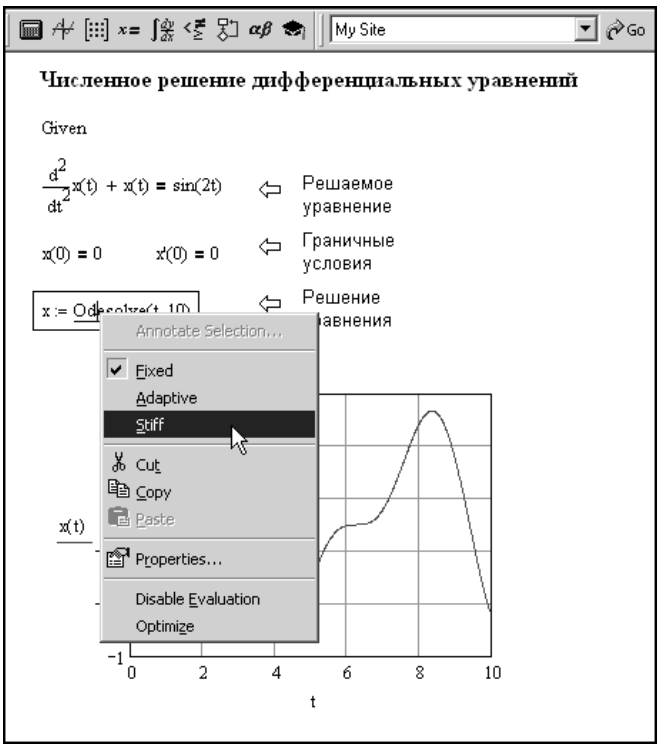


Рис. 7.28. Выбор метода решения уравнения

Хотя процедура `Odesolve()` весьма полезна и продуктивна, не следует ограничивать себя только ее рамками. В табл. 7.1 перечислены основные процедуры, применимые при численном решении дифференциальных уравнений, а также дана краткая справка по ним. Для удобства процедуры указаны вместе со своими аргументами.

Таблица 7.1. *Процедуры для численного решения дифференциальных уравнений*

Процедура	Описание
`bulstoer(y,x1,x2,e,F,k,s)`	Метод Булирша—Штера для вычисления значения функции (или функций) в конечной точке интервала поиска решения. Аргументы процедуры таковы: вектор `y` определяет начальные значения функций, `x1` и `x2` есть начальная и конечная точка интервала поиска решения, `e` определяет точность вычислений, вектор `F` задает правые части уравнений для первых производных функций (относительно которых решается уравнение), `k` — максимальное число внутренних точек для поиска решения, а `s` — наименьший допустимый промежуток между значениями для аппроксимации искомых функций
`Bulstoer(y,x1,x2,n,F)`	Метод Булирша—Штера. Вектор `y` задает начальные значения для системы дифференциальных уравнений, решение ищется на интервале от `x1` до `x2`, параметр `n` определяет число точек решения (за исключением начальной). `F` — вектор правых частей уравнений, определяющих зависимости первых производных для неизвестных функций, относительно которых решается система уравнений
`bvalfit(v1,v2,x1,x2,x3,F,l1,l2,t)`	Процедура определения краевых условий. `v1` и `v2` — векторы начальных приближений на границах, `x1`, `x2` и `x3` — граничные точки интервала поиска решения и точка сшивки решения, соответственно вектор `F` определяет функциональные зависимости для первых производных, аргументы `l1` и `l2` есть векторы-функции, определяющие значения на границах, аргумент `t` определяет тип сшивки решения

Таблица 7.1 *(продолжение)*

Процедура	Описание
`multigrid(M,N)`	Процедурой возвращается матрица решений для уравнения Пуассона для квадратной области с нулевыми значениями на границе. Матрица `M` определяет значения функции плотности в области, а `N` — число циклов прогонки для итерационной процедуры
`numol(x2,n1,t2,n2,n3,n4,p,f0,f1)`	Процедура решения одномерного уравнения в частных производных параболического типа. Аргументы функции: `x2` и `t2` есть конечные точки по координате и времени; `n1` и `n2` — число точек аппроксимации по каждой из переменных; `n3` и `n4` определяет для системы уравнений число дифференциальных и алгебраических уравнений; аргумент `p` является вектором-функцией, зависящей от координаты, времени, функции и ее производных по пространственной координате, вектор определяет правую часть уравнения (в левой стоит производная по времени); векторы-функции `f0` и `f1` определяют начальные и граничные условия соответственно
`Pdesolve(u,x,x2,t,t2,n1,n2)`	Процедура решения одномерных уравнений в частных производных гиперболического и параболического типа. Аргументы процедуры: вектор `u` названий функций в порядке появления их в блоке решения; пространственная переменная `x`; вектор граничных значений по пространственной переменной `x2`; временная переменная `t`; вектор граничных значений по временной переменной `t2`; необязательные аргументы `n1` и `n2` определяют количество точек дискретизации по пространственной и временной переменных
`radau(y,x1,x2,e,F,k,s)`	Вычисление значения функции (функций) в конечной точке интервала поиска решения методом RADAU5. Аргументы и их назначение такие же, как у процедуры `bulstoer()`

Таблица 7.1 *(продолжение)*

Процедура	Описание
`Radau(y,x1,x2,n,F)`	Вычисление значения функции (функций) на интервале значений методом RADAU5. Аргументы и их назначение такие же, как у процедуры `Bulstoer()`
`relax(A,B,C,D,E,M,U,r)`	По квадратным матрицам `A`, `B`, `C`, `D` и `E` коэффициентов дифференциального уравнения в частных производных второго порядка (коэффициенты при частных производных и функции), матрице значений функции источника `M`, матрице `U` граничных значений и начальных приближений вычисляется матрица решения. Аргумент `r` (значение в пределах от 0 до 1) есть спектральный радиус для итерационной процедуры Якоби и влияет на сходимость релаксационного алгоритма
`rkadapt(y,x1,x2,e,F,k,s)`	Вычисление значения функции (функций) в конечной точке интервала поиска решения методом Рунге—Кутта четвертого порядка с переменным шагом. Аргументы и их назначение такие же, как у процедуры `bulstoer()`
`Rkadapt(y,x1,x2,n,F)`	Метод Рунге—Кутта четвертого порядка с переменным шагом. Аргументы: вектор `y` задает начальные значения для системы дифференциальных уравнений; решение ищется на интервале от `x1` до `x2`; параметр `n` определяет число точек решения (за исключением начальной); `F` — вектор правых частей уравнений, определяющих зависимости первых производных для неизвестных функций
`rkfixed(y,x1,x2,n,F)`	Метод Рунге—Кутта четвертого порядка с фиксированным шагом. Аргументы: вектор `y` задает начальные значения для системы дифференциальных уравнений; решение ищется на интервале от `x1` до `x2`; параметр `n` определяет число точек решения (за исключением начальной); `F` — вектор правых частей уравнений, определяющих зависимости первых производных для неизвестных функций

Таблица 7.1 *(окончание)*

Процедура	Описание
`sbval(v1,x1,x2,F,l1,t)`	Процедура определения вектора начальных условий. Обозначения для аргументов формально такие же, как и для процедуры `bvalfit()`
`stiffb(y,x1,x2,e,F,J,k,s)`	Решение жесткого дифференциального уравнения в конечной точке диапазона с использованием метода Булирша—Штера. Через `J` обозначена матрица Якобиана дифференциального уравнения. Остальные обозначения такие же, как и для процедуры `bulstoer()`
`Stiffb(y,x1,x2,n,F,J)`	Решение жесткого дифференциального уравнения на диапазоне методом Булирша-Штера. Через `J` обозначена матрица Якобиана дифференциального уравнения. Остальные обозначения такие же, как и для процедуры `Bulstoer()`
`stiffr(y,x1,x2,e,F,J,k,s)`	Решение жесткого дифференциального уравнения в конечной точке диапазона методом Розенброка с переменным шагом. Через `J` обозначена матрица Якобиана дифференциального уравнения. Остальные обозначения такие же, как и для процедуры `bulstoer()`
`Stiffr(Y,x1,x2,n,F,J)`	Решение жесткого дифференциального уравнения на диапазоне методом Розенброка с переменным шагом. Через `J` обозначена матрица Якобиана дифференциального уравнения. Остальные обозначения такие же, как и для процедуры `Bulstoer()`

Таким образом, спектр встроенных утилит численного решения дифференциальных уравнений в Mathcad достаточно широк. Причем среди предлагаемых к услугам пользователя процедур есть и такие, что позволяют решать дифференциальные уравнения в частных производных. Правда, это относится к наиболее простым типам уравнений. Если уравнение достаточно сложное, приходится призывать на помощь не только возможности Mathcad, но и собственную изобретательность.

Статистический анализ

Задачи интерполяции, рассмотренные в предыдущем разделе, в некотором смысле относятся к статистическому анализу, а точнее, к задачам математической статистики, где по эмпирическим данным вычисляются оценочные значения для объективных характеристик систем и процессов. Существует несколько основных типов задач, с которыми приходится иметь дело в рамках математической статистики. Помимо упомянутой проблемы оценки неизвестных параметров по выборкам, можно также выделить проверку статистических гипотез, а также регрессионные задачи (оценка зависимостей между различными наборами данных).

Основные понятия математической статистики тесно связаны между собой и базируются на категориальном аппарате теории вероятностей. Во многом задачи теории вероятностей и математической статистики являются взаимообратными. Ведь, как известно, в рамках теории вероятностей по объективным характеристикам системы (в основном по функциям и законам распределения) делаются предположения о наблюдаемых на практике величинах и происходящих событиях. В этой главе будут рассмотрены задачи как математической статистики, так и теории вероятностей. К счастью, возможности Mathcad это позволяют. Среди встроенных функций приложения достаточно много предназначенных для статистического анализа, а также имеющих прямое или косвенное отношение к теории вероятностей.

Основные понятия теории вероятностей

Сразу оговоримся, что все теоретические сведения (особенно в части определений), приводимые далее, являются существенно упрощенными. Поэтому читателю, которого интересует строгая аксиоматика теории вероятностей и математической статистики, следует обратиться к специальной литературе, посвященной данному вопросу.

Одно из базовых понятий теории вероятностей — случайное событие. Это такое событие, которое может произойти, а может и не произойти. Совокупность всех случайных событий, с которыми предстоит иметь дело в процессе решения той или иной задачи, будем называть пространством событий. События, которые не содержат в себе других подсобытий, обычно называют элементарными. Любое событие, изучаемое в рамках теории вероятностей, рассматривается как состоящее из одного или нескольких элементарных событий. Над пространством событий определяют неотрицательную функцию, называемую вероятностью. Вероятность характеризует объективную возможность для события быть реализованным (т. е. произойти). Далеко не каждая функция, определенная над пространством событий, может называться вероятностью. Есть еще ряд дополнительных условий. К наиболее сущест-

венным и практически значимым можно отнести условие аддитивности. Формулируется оно следующим образом: если два события несовместимы (т. е. не могут произойти одновременно), то вероятность того, что произойдет хотя бы одно из них, равна сумме вероятностей этих событий. Кроме того, вероятность любого события не может превышать единицы. Единичная вероятность соответствует достоверному событию, т. е. такому, которое обязательно произойдет. Невозможным называется событие с нулевой вероятностью, т. е. такое, которое не может произойти ни при каких обстоятельствах.

Далее будут широко использоваться случайные величины. Случайной величиной называется численная функция от элементарного события. Практически любая задача, связанная со случайными величинами, подразумевает вычисление тех или иных их характеристик. Эти характеристики, в свою очередь, вычисляются на основе законов распределения случайной величины. Существует два практически важных случая: случайная величина может принимать значения из определенного интервала (непрерывная случайная величина) или она может принимать дискретные значения (дискретная случайная величина). В первом случае под законом распределения подразумевают функцию распределения (или плотность распределения). Для дискретной случайной величины закон распределения — набор возможных значений случайной величины и соответствующие им вероятности.

По определению функцией распределения $F(x)$ абсолютно непрерывной случайной величины ξ называется функция $F(x) = P(\xi < x)$, где $P(\xi < x)$ есть вероятность того, что случайная величина ξ меньше x. Плотностью распределения называется производная (если она существует) от функции распределения, т. е. $f(x) = F'(x)$.

Распределений существует достаточно много. Однако среди них можно выделить такие, что имеют особую важность и достаточно часто встречаются на практике. Для большинства этих распределений в Mathcad есть специальные встроенные функции, описывающие соответствующие функции распределения и плотности распределения. Некоторые примеры таких распределений приведены на рис. 7.29—7.32. Для определения различных распределений использованы встроенные функции Mathcad (они описываются несколько позже).

Пример 7.8. Основные распределения

На рис. 7.29 представлена функция и плотность нормального распределения.

Нормальное распределение с параметрами a и σ^2 характеризуется плотностью $f(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(x-a)^2}{2\sigma^2}\right\}$, которой соответствует функция распре-

деления $\Phi\left(\frac{x-a}{\sigma}\right)=\frac{1}{\sqrt{2\pi}}\int\limits_{-\infty}^{\frac{x-a}{\sigma}}\exp(-y^2/2)dy$. Плотность распределения (кривая Гаусса) имеет максимум в точке $x=a$. Функция распределения монотонно возрастает от нулевого значения в минус бесконечности до единицы в плюс бесконечности. Причем это общее свойство функций распределения — не только для нормального распределения.

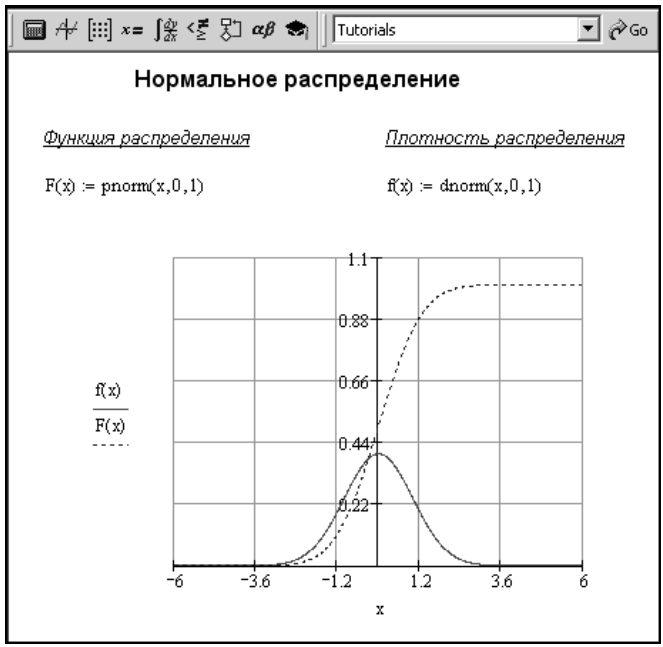


Рис. 7.29. Стандартное нормальное распределение

Показательное (или экспоненциальное) распределение с параметром λ характеризуется плотностью $f(x)=\lambda\exp(-\lambda x)$ для положительных аргументов и тождественно равно нулю для отрицательных. Функция распределения определяется равенством $F(x)=1-\exp(-\lambda x)$ для отрицательных аргументов. Обе функции представлены на рис. 7.30.

Для случайной величины, имеющей распределение Стьюдента с n степенями свободы, плотность распределения задается более сложной формулой: $f(x)=\frac{\Gamma((n+1)/2)}{\Gamma(n/2)\sqrt{\pi n}}\left(1+\frac{x^2}{n}\right)^{-\frac{n+1}{2}}$, где $\Gamma(z)$ — гамма-функция. Функция и плотность распределения для этого случая показаны на рис. 7.31.

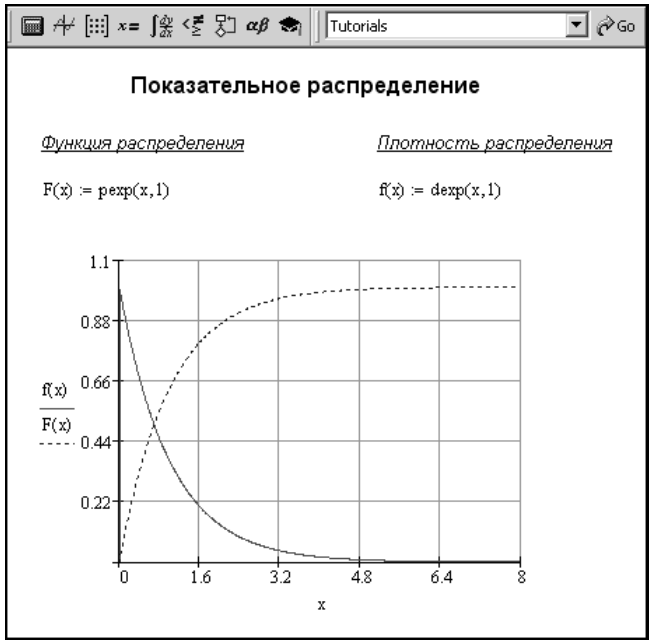


Рис. 7.30. Показательное распределение

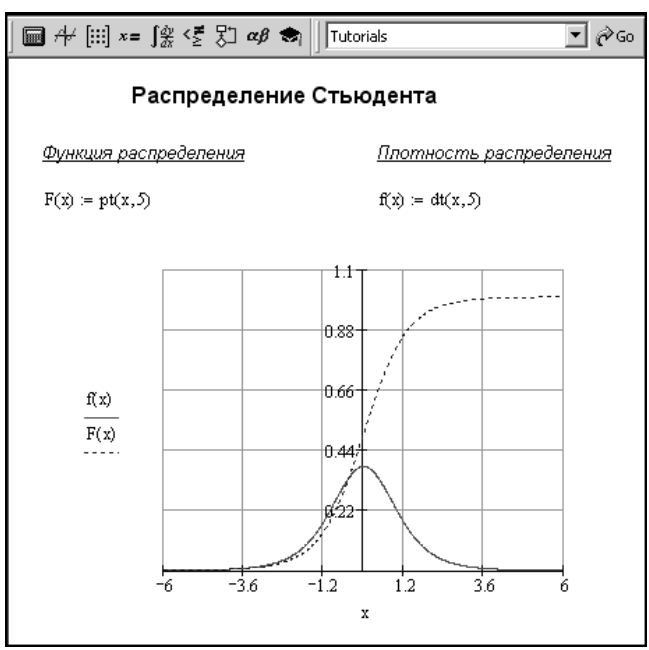


Рис. 7.31. Распределение Стьюдента

Если случайная величина характеризуется плотностью распределения вида $f(x) = \alpha\lambda x^{\alpha-1}\exp(-\lambda x^{\alpha})$ (α и λ — параметры распределения) для положительных значений x (и тождественно равна нулю для отрицательных x), говорят, что имеет место распределение Вейбулла. Функция распределения может быть вычислена по его плотности путем интегрирования, в данном случае получим $F(x) = 1 - \exp(-\lambda x^{\alpha})$. Функция (и плотность) распределения существенно (т. е. качественно) зависят от значения параметра α. Если он больше единицы, функции имеют вид, показанный на рис. 7.32.

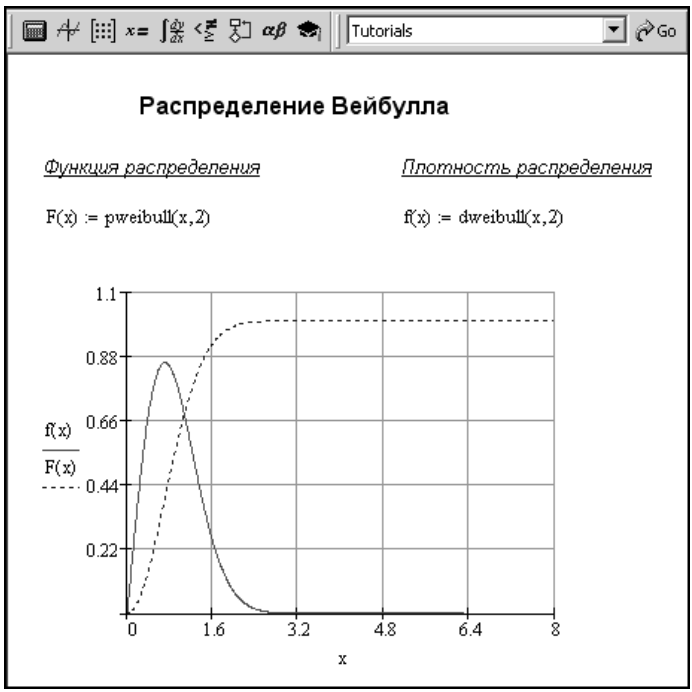


Рис. 7.32. Распределение Вейбулла

Как отмечалось, в Mathcad есть целый набор встроенных функций для основных распределений. В табл. 7.2 перечислены основные функции Mathcad для определения законов распределения случайных величин.

На основе перечисленных законов распределения можно генерировать случайные числа. Это очень важная задача при выполнении численного моделирования. Процедуры генерирования случайных чисел приведены в табл. 7.3. Все аргументы, за исключением первого, такие же, как у процедур для соответствующих распределений.

Таблица 7.2. *Функции законов распределения*

Закон распределения	Функция распределения	Плотность распределения
Бета-распределение. Плотность распределения дается выражением $f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1}$ (для значений аргумента `x` в интервале от 0 до 1)	`pbeta(x,a,b)`	`dbeta(x,a,b)`
Биномиальное распределение. В серии из `n` независимых опытов вероятность того, что будет ровно `k` успехов, дается выражением $P_n(\xi = k) = C_n^k p^k (1-p)^{n-k}$, где биномиальный коэффициент $C_n^k = \frac{n!}{k!(n-k)!}$, а `p` есть вероятность успеха в одном испытании. Функцией плотности распределения возвращается указанная вероятность. Функцией распределения возвращается вероятность того, что число успехов в серии опытов не превышает `k`: $P_n(\xi \le k) = \sum_{m=0}^{k} C_n^m p^m (1-p)^{n-m}$	`pbinom(k,n,p)`	`dbinom(k,n,p)`
Распределение Коши. Формально это распределение Стьюдента с одной степенью свободы. Плотность распределения дается выражением $f(x) = \frac{1}{\pi b\left(1+\left(\frac{x-a}{b}\right)^2\right)}$	`pcauchy(x,a,b)`	`dcauchy(x,a,b)`
Распределение хи-квадрат. Плотность распределения хи-квадрат с `n` степенями свободы равна $f(x) = \frac{x^{n/2-1}\exp(-x/2)}{2^{n/2}\Gamma(n/2)}$ (для положительных значений аргумента). Такое распределение имеет случайная величина, равная сумме квадратов независимых нормально распределенных случайных величин (с параметрами `(0,1)`)	`pchisq(x,n)`	`dchisq(x,n)`
Показательное (экспоненциальное) распределение. Плотность экспоненциального распределения с параметром `a` имеет вид $f(x) = a\exp(-ax)$	`pexp(x,a)`	`dexp(x,a)`

Таблица 7.2 *(продолжение)*

Закон распределения	**Функция распределения**	**Плотность распределения**
Распределение Фишера. Плотность распределения с `n` и `m` степенями свободы равна $f(x) = \frac{\Gamma((n+m)/2)}{\Gamma(n/2)\Gamma(m/2)} n^{m/2} m^{n/2} \frac{x^{n/2-1}}{(m+nx)^{(n+m)/2}}$. Это распределение характерно для случайных величин вида $\xi = \frac{\xi_n/n}{\xi_m/m}$, где ξ_n и ξ_m имеют хи-квадрат-распределение соответственно с `n` и `m` степенями свободы	`pF(x,n,m)`	`dF(x,n,m)`
Гамма-распределение. Плотность распределения (с параметром `n`) определяется выражением $f(x) = \frac{x^{n-1}\exp(-x)}{\Gamma(n)}$	`pgamma(x,n)`	`dgamma(x,n)`
Геометрическое распределение. Вероятность успеха в (`k+1`)-м опыте (все предыдущие `k` опытов неуспешны) при вероятности успеха одного опыта `p` определяется выражением $P(\xi = k+1) = p(1-p)^k$. Процедурой `pgeom()` возвращается вероятность $P = p\sum_{m=0}^{k}(1-p)^m = 1-(1-p)^{k+1}$ того, что в `k+1` испытаниях будет хотя бы один успех	`pgeom(k,p)`	`dgeom(k,p)`
Гипергеометрическое распределение. Иллюстрацией может быть ситуация, когда из урны, содержащей `N` шаров, среди которых `M` белых, вынимают `n` шаров. Вероятность того, что среди этих `n` шаров будет ровно `m` белых, определяется выражением $P_n(\xi = m) = \frac{C_M^m C_{N-M}^{n-m}}{C_N^n}$. Это значение возвращается процедурой `dhypergeom()`. Процедура `phypergeom()` позволяет найти вероятность того, что среди вытянутых шаров будет не менее `m` белых: $P_n(\xi \le m) = \sum_{k=0}^{m}\frac{C_M^k C_{N-M}^{n-k}}{C_N^n}$	`phypergeom (m,M,N-M,n)`	`dhypergeom (m,M,N-M,n)`

Таблица 7.2 *(продолжение)*

Закон распределения	Функция распределения	Плотность распределения
Нормальное логарифмическое распределение. Плотность распределения (с параметрами μ и σ) имеет вид $f(x) = \frac{\exp(-(\ln(x)-\mu)^2/2\sigma^2)}{\sqrt{2\pi}\sigma x}$	plnorm(x,μ,σ)	dlnorm(x,μ,σ)
Логистическое распределение. Плотность распределения с параметрами a и b дается выражением $f(x) = \frac{\exp(-(x-a)/b)}{b\left(1+\exp(-(x-a)/b)\right)^2}$	plogis(x,a,b)	dlogis(x,a,b)
Отрицательное биномиальное распределение. Закон распределения случайной величины определяется выражением для вероятности $P = C_{n+k-1}^{k} p^n (1-p)^k$	pnbinom(k,n,p)	dnbinom(k,n,p)
Нормальное распределение. Нормальное распределение (с параметрами a и σ) определяется плотностью распределения $f(x) = \frac{\exp(-(x-a)^2/2\sigma^2)}{\sqrt{2\pi}\sigma}$. Для функции нормального распределения с параметрами (0,1) может использоваться процедура cnorm()	pnorm(x,a,σ)	dnorm(x,a,σ)
Распределение Пуассона. Дискретное распределение (с параметром λ), характеризующееся законом распределения $P(\xi = k) = \frac{\lambda^k \exp(-\lambda)}{k!}$ (вероятность того, что случайная величина примет целочисленное значение k)	ppois(k,λ)	dpois(k,λ)
Распределение Стьюдента. Распределение Стьюдента с n степенями свободы характеризуется плотностью распределения $f(x) = \frac{\Gamma((n+1)/2)}{\sqrt{n\pi}\Gamma(n/2)}\left(1+\frac{x^2}{n}\right)^{-(n+1)/2}$. Такое распределение соответствует случайной величине $\xi = \xi_0/\sqrt{\xi_n/n}$, где ξ_0 имеет стандартное нормальное распределение (т. е. нормальное распределение с параметрами (0,1)), а случайная величина ξ_n имеет распределение хи-квадрат с n степенями свободы	pt(x,n)	dt(x,n)

Таблица 7.2 *(окончание)*

Закон распределения	Функция распределения	Плотность распределения
Равномерное распределение. Случайная величина, равномерно распределенная на интервале `(a,b)`, характеризуется плотностью распределения $f(x) = 1/(b-a)$ для аргумента, лежащего в этом диапазоне, и равная нулю в противном случае	`punif(x,a,b)`	`dunif(x,a,b)`
Распределение Вейбулла. Встроенные функции Mathcad (функции распределения и плотности распределения) основаны на выражении $ax^{a-1}\exp(-x^a)$	`pweibull(x,a)`	`dweibull(x,a)`

Таблица 7.3. *Процедуры для генерирования случайных чисел*

Распределение	Процедура
Вектор из `f` элементов, имеющих бета-распределение	`rbeta(f,a,b)`
Вектор из `f` элементов, имеющих биномиальное распределение	`rbinom(f,n,p)`
Вектор из `f` элементов, имеющих распределение Коши	`rcauchy(f,a,b)`
Вектор из `f` элементов, имеющих распределение хи-квадрат	`rchisq(f,n)`
Вектор из `f` элементов, имеющих экспоненциальное распределение	`rexp(f,a)`
Вектор из `f` элементов, имеющих распределение Фишера	`rF(f,n,m)`
Вектор из `f` элементов, имеющих гамма-распределение	`rgamma(f,n)`
Вектор из `f` элементов, имеющих геометрическое распределение	`rgeom(f,p)`
Вектор из `f` элементов, имеющих гипергеометрическое распределение	`rhypergeom(f,M,N-M,n)`
Вектор из `f` элементов, имеющих нормальное логарифмическое распределение	`rlnorm(f,μ,σ)`
Вектор из `f` элементов, имеющих логистическое распределение	`rlogis(f,a,b)`

Таблица 7.3 *(окончание)*

Распределение	Процедура
Вектор из f элементов, имеющих отрицательное биномиальное распределение	rnbinom(f,n,p)
Вектор из f элементов, имеющих нормальное распределение	rnorm(f,a,σ)
Вектор из f элементов, имеющих распределение Пуассона	rpois(f,λ)
Вектор из f элементов, имеющих распределение Стьюдента	rt(f,n)
Вектор из f элементов, имеющих равномерное распределение. Для генерирования случайного числа в диапазоне от 0 до x можно использовать команду rnd(x)	runif(f,a,b)
Вектор из f элементов, имеющих распределение Вейбулла	rweibull(f,a)

Наконец, может возникнуть обратная задача: по известной вероятности и типу распределения определить значение случайной величины (т. е. значение, вероятность реализации которого задана). В Mathcad имеется встроенный набор соответствующих обратных функций. Они перечислены в табл. 7.4. Первым аргументом у них указывается вероятность, а остальные совпадают с аргументами функций распределений.

Таблица 7.4. *Процедуры для определения случайных величин по известным вероятностям их реализации*

Распределение	Процедура
Бета-распределение	qbeta(P,a,b)
Биномиальное распределение	qbinom(P,n,p)
Распределение Коши	qcauchy(P,a,b)
Хи-квадра-распределение	qchisq(P,n)
Экспоненциальное распределение	qexp(P,a)
Распределение Фишера	qF(P,n,m)
Гамма-распределение	qgamma(P,n)
Геометрическое распределение	qgeom(P,p)
Гипергеометрическое распределение	qhypergeom(P,M,N-M,n)

Таблица 7.4 *(окончание)*

Распределение	Процедура
Нормальное логарифмическое распределение	`qlnorm(P,μ,σ)`
Логистическое распределение	`qlogis(P,a,b)`
Отрицательное биномиальное распределение	`qnbinom(P,n,p)`
Нормальное распределение	`qnorm(P,a,σ)`
Распределение Пуассона	`qpois(P,λ)`
Распределение Стьюдента	`qt(P,n)`
Равномерное распределение	`qunif(P,a,b)`
Распределение Вейбулла	`qweibull(P,a)`

Во всех процедурах в табл. 7.4 первым аргументом указывается, как отмечалось, вероятность `P`. Это значение вероятности, возвращаемое для данного распределения и искомого значения случайной величины процедурой, соответствующей кумулятивной функции распределения. Другими словами, возвращается значение, которое случайная величина, имеющая данное распределение, не превысит с вероятностью `P`.

Зная функцию или плотность распределения случайной величины, можно определить все ее численные характеристики, которые перечисляются далее.

Математическим ожиданием случайной величины ξ называется величина $M\xi = \int_{-\infty}^{\infty} xf(x)dx$ (если соответствующий интеграл сходится). Для дискретной случайной величины математическое ожидание определяется через сумму произведений ее значений на вероятности их реализации: $M\xi = \sum_{k=1}^{\infty} \xi_k P(\xi = \xi_k)$ (если ряд сходится абсолютно). Дисперсией случайной величины ξ называется величина $D\xi = M(\xi - M\xi)^2$. Дисперсия определяет степень отклонения случайной величины от математического ожидания.

Определение математического ожидания и дисперсии случайной величины относится к классу элементарных задач, которые легко и элегантно решаются в Mathcad. Обратимся к примерам.

Пример 7.9. Числовые характеристики случайных величин

В документе на рис. 7.33 приведены примеры вычисления математических ожиданий и дисперсий для случайных величин, имеющих дискретные распределения.

Tutorials Go

Числовые характеристики случайных величин

$n := 10 \quad p := 0.3 \quad \lambda := 4 \quad N := 1000$

Математическое ожидание: **Дисперсия:**

Биномиальное распределение

$$Mx := \sum_{k=0}^{n} k \cdot dbinom(k,n,p) \qquad Dx := \sum_{k=0}^{n} (k - Mx)^2 \cdot dbinom(k,n,p)$$

$Mx = 3 \qquad Dx = 2.1$

Геометрическое распределение

$$Mx := \sum_{k=0}^{N} k \cdot dgeom(k,p) \qquad Dx := \sum_{k=0}^{N} (k - Mx)^2 \cdot dgeom(k,p)$$

$Mx = 2.333 \qquad Dx = 7.778$

Распределение Пуассона

$$Mx := \sum_{k=0}^{N} k \cdot dpois(k,\lambda) \qquad Dx := \sum_{k=0}^{N} (k - Mx)^2 \cdot dpois(k,\lambda)$$

$Mx = 4 \qquad Dx = 4$

Рис. 7.33. Математические ожидания и дисперсии для дискретных распределений

Tutorials Go

Числовые характеристики случайных величин

$a := 1 \quad b := 3 \quad n := 10$

Математическое ожидание: **Дисперсия:**

Бета-распределение

$$Mx := \int_0^1 x \cdot dbeta(x,a,b)\, dx \qquad Dx := \int_0^1 (x - Mx)^2 \cdot dbeta(x,a,b)\, dx$$

$Mx = 0.25 \qquad Dx = 0.038$

Хи-квадрат распределение

$$Mx := \int_0^\infty x \cdot dchisq(x,n)\, dx \qquad Dx := \int_0^\infty (x - Mx)^2 \cdot dchisq(x,n)\, dx$$

$Mx = 10 \qquad Dx = 20$

Равномерное распределение

$$Mx := \int_0^1 x \cdot dunif(x,0,1)\, dx \qquad Dx := \int_0^1 (x - Mx)^2 \cdot dunif(x,0,1)\, dx$$

$Mx = 0.5 \qquad Dx = 0.083$

Рис. 7.34. Математические ожидания и дисперсии для непрерывных распределений

Как вычисляются математические ожидания и дисперсии для непрерывных случайных величин, можно составить представление по документу, представленному на рис. 7.34.

Класс задач, решаемых в рамках теории вероятностей, существенно выходит за рамки определения математических ожиданий и дисперсий случайных величин. Просто эти понятия наиболее близки к математической статистике, что будет проиллюстрировано в следующем подразделе. Далее рассмотрим задачу, решение которой реализовано в Mathcad методами, доступными только при использовании математических пакетов.

Задача 7.10. Функция распределения

Пусть имеется квадрат с единичной стороной. Внутри него случайным образом выбирается точка, которая служит вершиной прямоугольника. Нужно найти функцию распределения случайной величины, равную площади указанного прямоугольника.

Для решения этой задачи ее удобнее немного переформулировать. В частности, координаты выбираемой случайным образом точки можно интерпретировать как два случайных числа ξ_1 и ξ_2, равномерно распределенных в диапазоне значений от 0 до 1. Площадь прямоугольника равна, очевидно, произведению этих случайных чисел $\xi = \xi_1\xi_2$. Согласно определению, функция распределения случайной величины $F_\xi(x)$ есть вероятность того, что эта случайная величина меньше аргумента функции распределения, т. е. $F_\xi(x) = P(\xi < x)$. Из этого и будем исходить. В частности, воспользуемся методом Монте-Карло. Будем при фиксированном аргументе x генерировать пары случайных чисел и сравнивать их произведение со значением x. Число пар, для которых произведение меньше x, делится на общее число сгенерированных пар чисел. При большом числе общих сгенерированных пар данное отношение будет стремиться к искомой вероятности. Реализация процедуры вычисления функции распределения в Mathcad показана на рис. 7.35.

Процедура начинается с инициализации переменной, которая определяет число генерируемых пар чисел. Также вводится переменная (с начальным нулевым значением) для подсчета числа пар чисел, произведение которых меньше x. Далее в рамках оператора цикла генерируются случайные числа и выполняются все прочие необходимые действия. Они достаточно прозрачны и комментариев, думается, не требуют. Результат есть отношение двух локальных переменных, инициализированных в самом начале процедуры.

У этой задачи есть точное аналитическое решение, которое здесь приводить не будем. Отметим только, что искомая функция распределения $F_\xi(x) = x(1 - \ln(x))$ при $0 < x < 1$, нулю — при отрицательном аргументе и единице — при значениях аргумента, больших единицы. В соответствии

с этим в рабочем документе описана функция. Она нужна для сравнения результатов расчета функции распределения случайной величины различными методами. В нижней части документа на рис. 7.35 приведены два графика, построенные согласно точной функции и эмпирической, рассчитанной методом Монте-Карло. В последнем случае вычисления выполнялись на основе 10 000 пар случайных чисел для каждого значения аргумента. Как легко заметить, совпадение практически идеальное. Правда, из-за довольно большого числа шагов построение этих графиков может занять некоторое время. Работу процедуры можно ускорить, уменьшив число шагов. На рис. 7.36 функция распределения строится на основе генерирования 1000 пар случайных чисел. В этом случае прослеживается некоторый разброс точек эмпирической зависимости.

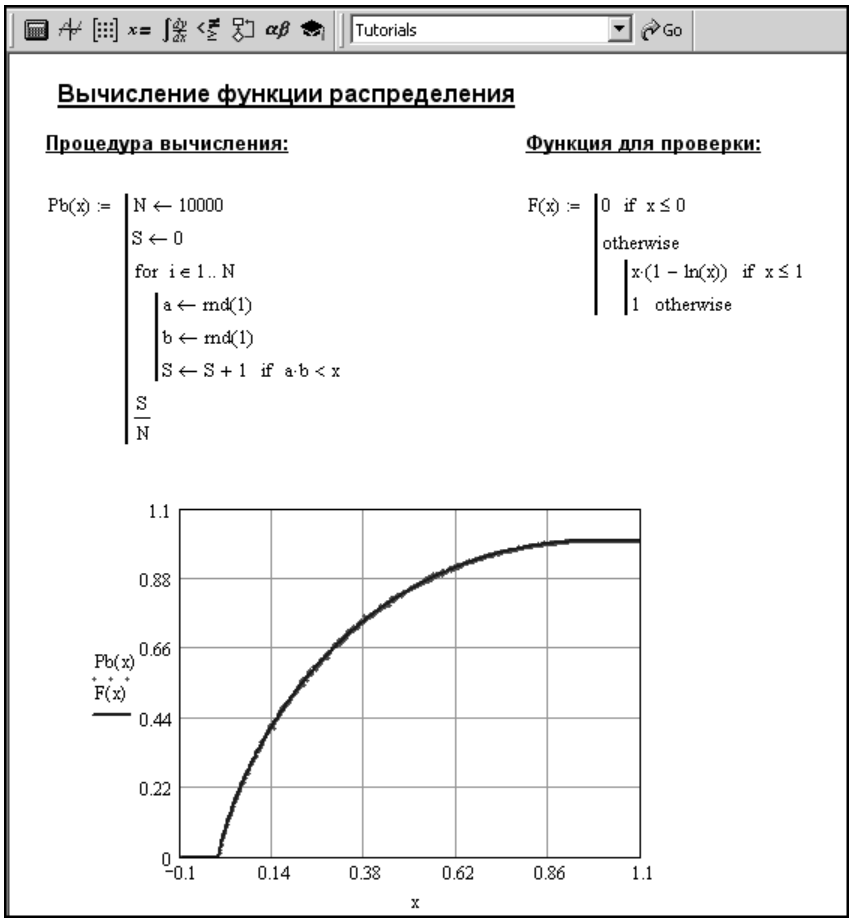


Рис. 7.35. Вычисление функции распределения (10 000 шагов)

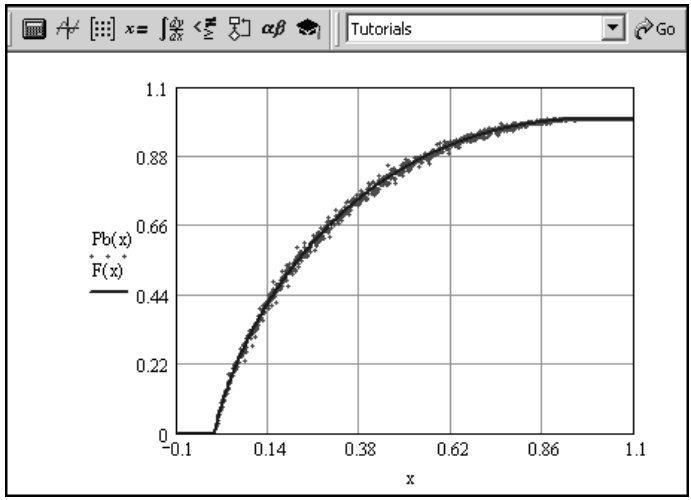


Рис. 7.36. Функция распределения на основе 1000 шагов

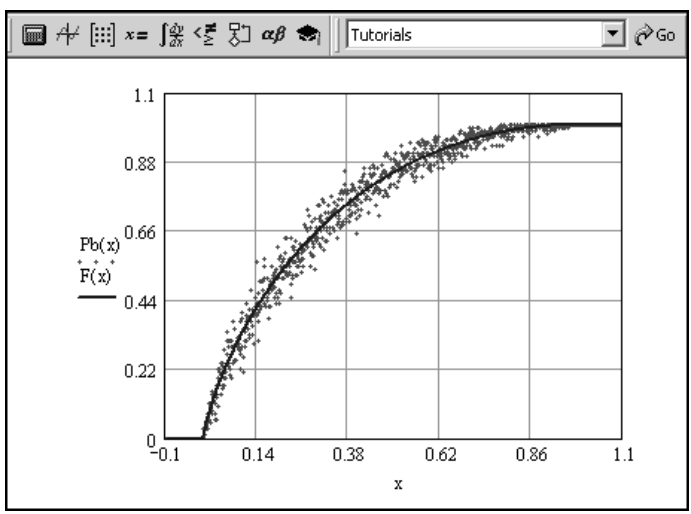


Рис. 7.37. Функция распределения на основе 100 шагов

Если число шагов уменьшить до 100, разброс становится очень существенным (рис. 7.37).

Поэтому к вопросу выбора минимального необходимого числа сгенерированных случайных чисел для получения достоверных результатов следует относиться крайне внимательно. В некотором смысле эта проблема перекликается с задачами математической статистики. Ведь объективность и информативность эмпирических данных — ключевой момент, который часто может быть

определен в рамках статистических подходов, особенно если в распоряжении исследователя имеется такое эффективное приложение, как Mathcad.

Элементарные задачи математической статистики

Спектр решаемых в рамках математической статистики задач весьма широк. Полное их описание не является целью этой книги. Поэтому здесь будут рассмотрены только самые простые задачи, причем исключительно в прикладном аспекте, с точки зрения их реализации в Mathcad. В основном это вычисление выборочных характеристик и корреляционный анализ данных. Приведем несколько определений.

Случайной выборкой называется случайный вектор, элементы которого независимы и одинаково распределены. Обычно под выборкой подразумевают результаты независимых измерений, которые проводятся в одинаковых условиях.

Эмпирическая функция распределения определяется как отношение числа элементов выборки, меньших аргумента функции, к общему числу элементов выборки. Фактически, это определение было использовано ранее при решении задачи о распределении случайной величины.

Если обозначить выборку как $(x_1, x_2, ..., x_n)$, то выборочным моментом порядка k называют величину $a_k = \frac{1}{n}\sum_{m=1}^{n} x_m^k$. Центральным выборочным моментом порядка k называется величина $b_k = \frac{1}{n}\sum_{m=1}^{n}(x_m - \overline{x})^k$, где через $\overline{x}$ обозначено выборочное среднее (выборочное математическое ожидание), совпадающее с выборочным моментом первого порядка: $\overline{x} = a_1 = \frac{1}{n}\sum_{m=1}^{n} x_m$.

Центральный выборочный момент второго порядка называется выборочной дисперсией.

Можно показать, что математическое ожидание от выборочного среднего равно математическому ожиданию элементов выборки (поскольку все элементы выборки одинаково распределены, то они имеют одинаковые числовые характеристики, в том числе и математические ожидания). Другими словами, если $Mx_m = a$, то $M\overline{x} = Mx_m = a$. Математическое ожидание от выборочной дисперсии совпадает с дисперсией элементов выборки с точностью до множителя $(n-1)/n$. Если обозначить $Dx_m = \sigma^2$, то $Mb_2 = \frac{n-1}{n}\sigma^2$.

Поэтому выборочную дисперсию называют смещенной оценкой для дисперсии

случайных величин — элементов выборки. Обычно рассматривают несмещенную оценку дисперсии $s^2 = \frac{n}{n-1} b_2 = \frac{1}{n-1} \sum_{m=1}^{n} (x_m - \overline{x})^2$. Для этой характеристики математическое ожидание совпадает с дисперсией элементов выборки. При большом объеме выборки (т. е. при большом числе элементов в ней) существенной разницы в этих оценках нет.

Для вычисления указанных характеристик в Mathcad есть специальные встроенные функции. Так, среднее значение может быть вычислено с помощью процедуры `mean()`. Ее аргументом указывают вектор выборки. Кроме этого иногда приходится вычислять среднее геометрическое и среднее гармоническое значения по элементам выборки. Среднее геометрическое вычисляется по формуле $\sqrt[n]{\prod_{m=1}^{n} x_m}$, а среднее гармоническое — $\left(\frac{1}{n} \sum_{m=1}^{n} \frac{1}{x_m} \right)^{-1}$.

В Mathcad среднее геометрическое вычисляется процедурой `gmean()`, а среднее гармоническое — `hmean()`. Центральное значение в выборке (такое значение, по отношению к которому число меньших и больших значений одинаково) вычисляют процедурой `median()`. Наиболее часто встречающееся значение легко найти с помощью процедуры `mode()`.

Само собой разумеется, что для вычисления выборочной дисперсии в Mathcad также есть встроенные функции. Смещенная дисперсия может быть рассчитана посредством процедуры `var()`, несмещенная — `Var()`. И в том, и в другом случае аргументом указывается набор данных, на основе которых проводятся вычисления. Стандартные отклонения (корень квадратный из дисперсии) также вычисляются без особых проблем: стандартное отклонение на основе смещенной дисперсии вычисляют процедурой `stdev()`, а на основе несмещенной дисперсии — с помощью `Stdev()`. Как и ранее, аргументы процедур — вектора выборок.

Пример 7.10. Выборочные статистические характеристики

Примеры вычисления выборочных характеристик можно найти в документе на рис. 7.38.

Все характеристики вычисляются на основе выборки, которая создана с помощью генератора случайных чисел. Данные в выборке имеют нормальное распределение со средним значением 3 и стандартным отклонением 0,5. В выборке 1000 элементов. Как видим, вычисленные выборочные характеристики достаточно близки к теоретическим.

Tutorials Go

Вычисление выборочных средних и дисперсий

Формирование выборки:

x := rnorm(1000,3,0.5)

x =

	0
0	2.883
1	3.09
2	3.221
3	3.342
4	2.405
5	3.07
6	2.854
7	2.8
8	3.016
9	3.195

Средние значения и дисперсии:

mean(x) = 2.996 ⇦ Среднее значение

gmean(x) = 2.954 ⇦ Среднее геометрическое

hmean(x) = 2.909 ⇦ Среднее гармоническое

median(x) = 2.991 ⇦ Центральное значение

var(x) = 0.239 ⇦ Смещенная дисперсия

Var(x) = 0.239 ⇦ Несмещенная дисперсия

stdev(x) = 0.489 ⇦ Смещенное отклонение

Stdev(x) = 0.489 ⇦ Несмещенное отклонение

Рис. 7.38. Вычисление выборочных характеристик

Tutorials Go

Данные для создания гистограмм

$$\text{hist}(3,x) = \begin{pmatrix} 86 \\ 820 \\ 94 \end{pmatrix} \qquad \text{histogram}(3,x) = \begin{pmatrix} 1.667 & 86 \\ 3 & 820 \\ 4.333 & 94 \end{pmatrix}$$

hist(15,x) =

	0
0	0
1	4
2	7
3	20
4	55
5	113
6	191
7	229
8	184
9	103
10	66
11	22
12	5
13	1
14	0

histogram(15,x) =

	0	1
0	1.133	0
1	1.4	4
2	1.667	7
3	1.933	20
4	2.2	55
5	2.467	113
6	2.733	191
7	3	229
8	3.267	184
9	3.533	103
10	3.8	66
11	4.067	22
12	4.333	5
13	4.6	1
14	4.867	0

Рис. 7.39. Использование процедур `hist()` и `histogram()`

Представление об эмпирической функции распределения можно составить по гистограммам. Последние создаются на основе векторов выборок с помощью процедур `hist()` и `histogram()`. Аргументы у обеих процедур одинаковы: первый определяет интервалы для создания гистограммы, а второй — это собственно выборка, на основе которой строится гистограмма. Что касается первого аргумента, то он может быть либо вектором конечных точек интервалов для группировки данных выборки, либо целым числом, задающим число интервалов. В последнем случае весь диапазон значений в выборке разбивается на равные интервалы.

Разница между процедурами `hist()` и `histogram()` состоит в том, что первой возвращается вектор частот, с которыми данные выборки попадают в базовые интервалы гистограммы. Второй процедурой возвращается матрица. Первый ее столбец содержит срединные точки базовых интервалов, а второй — частоты попадания данных в эти интервалы. На рис. 7.39 показаны простые примеры с процедурами `hist()` и `histogram()`.

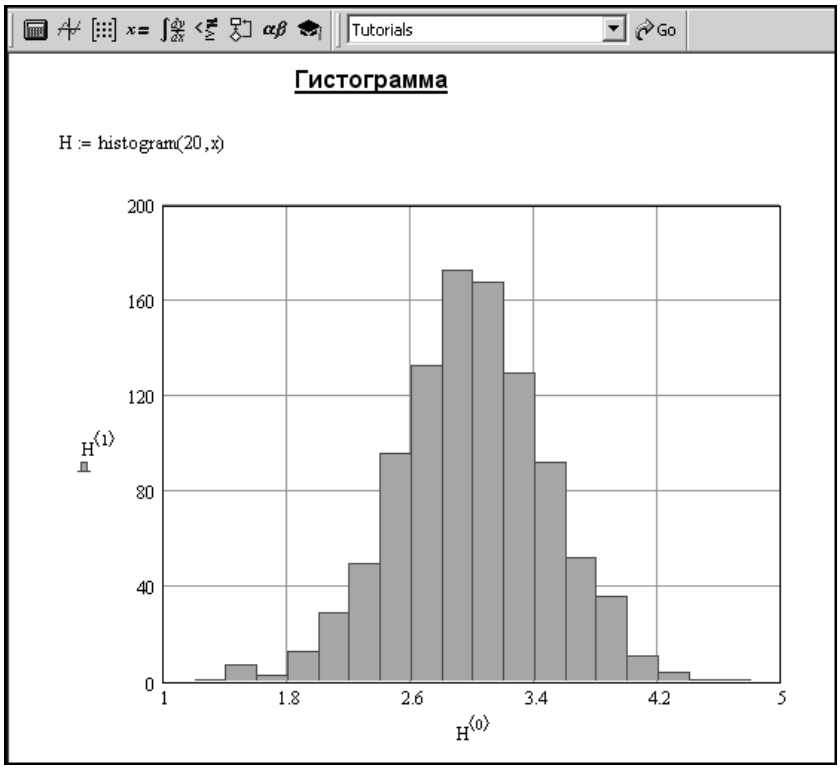


Рис. 7.40. Гистограмма

Наиболее простой способ создания гистограммы состоит в следующем. Сначала процедурой `histogram()` инициализируется объект (матрица), содержащая срединные точки интервалов гистограммы (первый столбец) и столбец частот. Далее у горизонтальной оси указывают столбец срединных точек интервалов (это столбец матрицы с нулевым индексом), а у вертикальной — столбец с частотами распределения данных по интервалам гистограммы (столбец с первым индексом). Индекс для столбца вводится с помощью пиктограммы панели **Matrix** или комбинацией клавиш <Ctrl>+<6>. Результат построения гистограммы по сформированной ранее выборке показан на рис. 7.40.

Сразу следует отметить, что число интервалов, по которым группируются данные, — весьма важный фактор, качественно определяющий вид гистограммы. На рис. 7.40 гистограмма строится на основе 20 интервалов. Если число интервалов увеличить до 60, гистограмма будет выглядеть так, как показано на рис. 7.41.

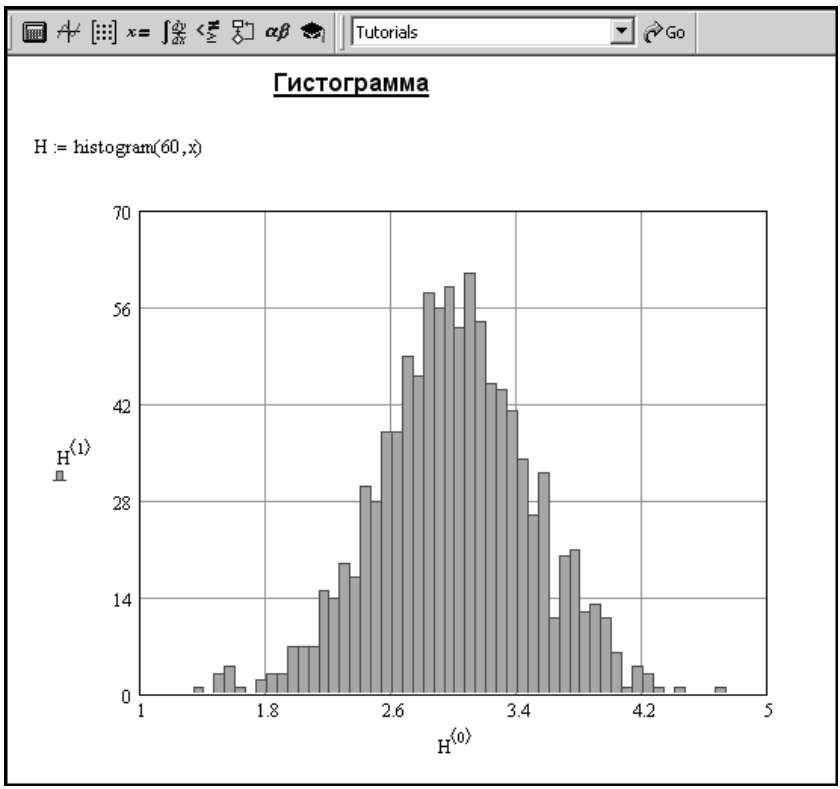


Рис. 7.41. При изменении числа интервалов гистограмма меняет вид

Важный класс задач математической статистики связан с исследованием наборов численных данных на предмет их зависимости или независимости. Обычно такой вывод делают на основе значения коэффициента корреляции, определение которого основано на таком понятии, как ковариация. Для двух случайных величин ξ и η ковариация определяется как $\mathrm{cov}(\xi,\eta) = M\{(\xi - M\xi)(\eta - M\eta)\}$. Для независимых случайных величин ковариация равна нулю. Коэффициент корреляции есть ковариация, нормированная на корень квадратный из произведения дисперсий, а именно $\rho(\xi,\eta) = \dfrac{\mathrm{cov}(\xi,\eta)}{\sqrt{D\xi D\eta}}$. Коэффициент корреляции не может по абсолютной величине превышать единицу. Как и ковариация, для независимых случайных величин коэффициент корреляции равен нулю. Однако из равенства нулю коэффициента корреляции независимость случайных величин в общем случае не следует. Нулевой коэффициент корреляции свидетельствует о независимости случайных величин, только если они имеют нормальное распределение.

Ранее приведено определение теоретического коэффициента корреляции (и, разумеется, ковариации). Выборочные ковариация и корреляция могут быть вычислены, если операцию определения математического ожидания заменить на вычисление выборочного среднего. В Mathcad для расчета ковариации имеется процедура `cvar()`. Коэффициент корреляции возвращается процедурой `corr()`. У обеих процедур по два аргумента — выборки, для которых вычисляются соответствующие характеристики.

Пример 7.11. Корреляционные характеристики

Примеры применения процедур `cvar()` и `corr()` показаны на рис. 7.42.

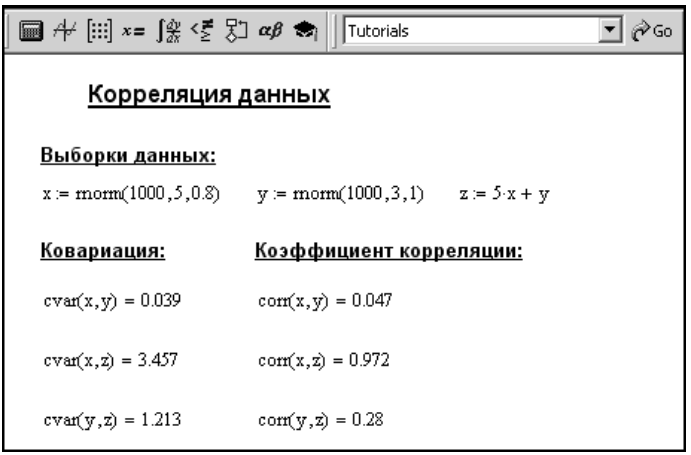


Рис. 7.42. Корреляция данных

К сожалению, по коэффициенту корреляции далеко не всегда можно сделать вывод о том, существует между данными зависимость или нет. Примеры на рис. 7.42 это подтверждают. В подобных случаях нужны другие методы.

Регрессионный анализ

Методы регрессионного анализа позволяют строить модели, описывающие взаимосвязь между наборами данных. Рассмотрим это более подробно. Предположим, имеется два набора зависимых данных, причем качественный вид зависимости также известен. Соответствующая зависимость дается некоторой функцией, в которую кроме аргумента входят неизвестные параметры, определяемые на основе наборов данных. От интерполяции такая задача принципиально отличается тем, что число неизвестных параметров существенно меньше базовых (узловых) точек функциональной зависимости, поэтому при фиксированном типе кривой провести ее так, чтобы она проходила через все узловые точки, в общем случае не удастся. Поэтому здесь необходим иной критерий. Обычно параметры определяются на основе метода наименьших квадратов. Главная идея метода состоит в том, что сумма квадратов разностей табличных значений функции и их оценок на основе регрессионной функции в узловых точках должна быть минимальной. Именно этим критерием руководствуются, создавая регрессионную модель.

Наиболее простая регрессионная модель — линейная. В этом случае зависимость дается соотношением вида $y = ax + b$, где через x обозначен независимый параметр, через y — зависимый, а параметры a и b нужно найти исходя из статистических данных для значений зависимого параметра $\{y_m\}$ в узловых точках $\{x_m\}$, $m = 1, 2, ..., n$. Определение этих параметров по методу наименьших квадратов подразумевает минимизацию выражения $\sum_{m=1}^{n} (y(x_m) - y_m)^2$, что дает $a = (\overline{xy} - \overline{x} \cdot \overline{y}) \big/ (\overline{x^2} - \overline{x}^2)$ и $b = \overline{y} - a\overline{x}$, где черта сверху обозначает усреднение. Эти же выражения можно записать несколько иначе. Поскольку выборочная ковариация случайных величин x и y равна $\mathrm{cov}(x, y) = \overline{xy} - \overline{x} \cdot \overline{y}$, а выборочная дисперсия величины x может быть записана как разность среднего квадрата случайной величины и квадрата ее среднего, т. е. $Dx = \overline{x^2} - \overline{x}^2$, то $a = \mathrm{cov}(x, y) \big/ Dx$. Однако все это не столь важно на практике, поскольку в Mathcad для выполнения всевозможных регрессий (в том числе и линейной регрессии) встроенных процедур больше чем достаточно.

Пример 7.12. Линейная регрессионная модель

Пример использования процедур линейной регрессии `line()`, `slope()` и `intercept()` представлен в документе на рис. 7.43.

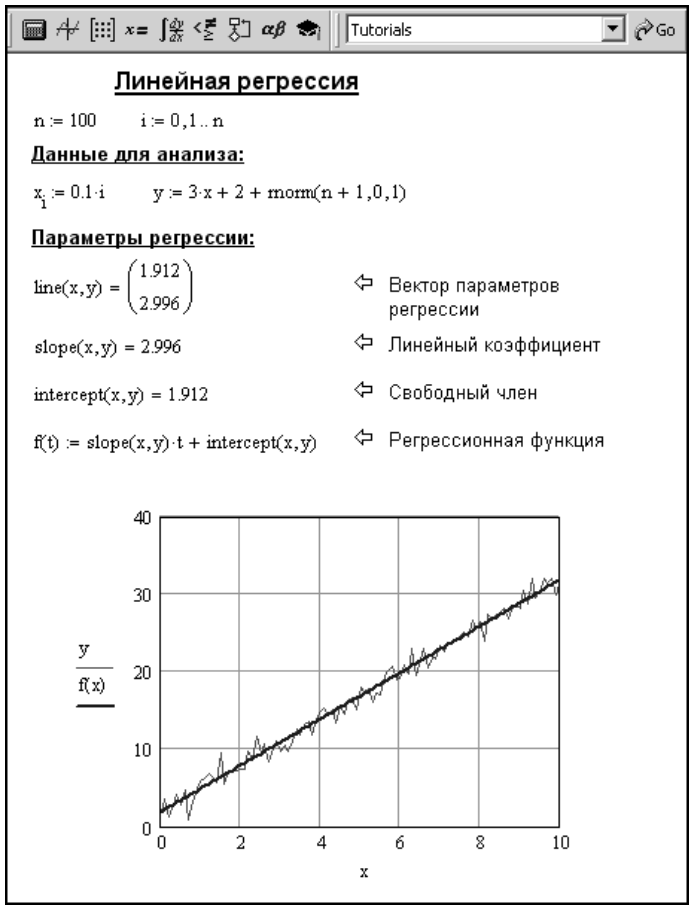


Рис. 7.43. Линейная регрессия

Аргументами процедур указывают векторы узловых точек и значений функции (зависимого параметра) в этих узловых точках. Процедура `line()` возвращает вектор параметров регрессии: свободный член b и линейный коэффициент a. Линейный коэффициент можно вычислить также с помощью `slope()`, а свободный член — с помощью `intercept()`. В примере на рис. 7.43 линейная регрессионная модель построена посредством именно этих процедур. Данные, на основе которых строилась модель, условны. Сначала сформирован вектор узловых точек, а затем создается вектор значений

зависимого параметра в этих узловых точках. Функциональная зависимость между параметрами линейная, однако в нее внесены случайные отклонения. Достигается последнее путем добавления случайных нормально распределенных чисел с нулевым средним значением и единичным стандартным отклонением. Найденные в процессе построения регрессионной модели параметры неплохо совпадают с коэффициентами функциональной зависимости.

Пример 7.13. Медианный метод

В Mathcad существует встроенная процедура `medfit()` для построения регрессионной модели медианным методом. Процедурой возвращается вектор коэффициентов регрессии. В примерах на рис. 7.44 регрессия выполнена различными методами, в том числе и с привлечением `medfit()`.

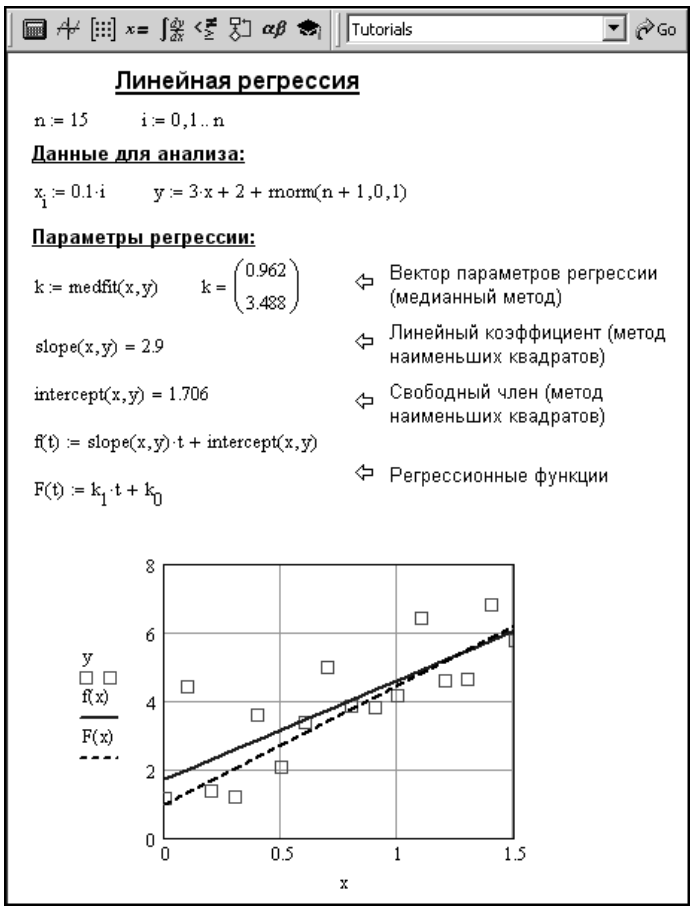


Рис. 7.44. Различные типы регрессии

Если узловых точек достаточно много, особой разницы в регрессионных моделях нет (имеются в виду линейные модели, полученные разными методами). На рис. 7.44 число точек специально выбрано незначительным, чтобы можно было увидеть различие между регрессионными прямыми.

Если вместо табличных статистических данных взять построенную регрессионную зависимость, то в узловых точках значения (теоретические, рассчитанные по регрессионной зависимости) зависимого параметра будут отличаться от табличных (экспериментальных) значений. Это так называемые погрешности регрессии. Общей характеристикой ошибки, связанной с использованием регрессионной зависимости, является величина, называемая ошибкой регрессии. Ошибка регрессии вычисляется по формуле $RE = \sqrt{\frac{1}{n-2}\sum_{m=1}^{n}(y_m - y(x_m))^2}$. В Mathcad для определения такого рода ошибки предназначена процедура `stderr()`. Ее аргументами указывают векторы узловых точек и значений зависимого параметра в узловых точках.

Далеко не всегда зависимость между данными носит линейный характер. Более общей, по сравнению с линейной моделью, является полиномиальная регрессия. В этом случае, как несложно догадаться, зависимость параметра y от параметра x ищется в виде полинома. Неизвестными, которые следует определить по статистическим данным, являются коэффициенты полинома. Для выполнения полиномиальной регрессии в Mathcad существуют процедуры `regress()` и `interp()`, действующие совместно. Результат, возвращаемый процедурой `regress()` (а это вектор коэффициентов), указывается первым аргументом процедуры `interp()`. Кроме этого аргумента, процедуре передаются векторы узловых точек, значений зависимого параметра в узловых точках, а также значение переменной полинома, для которой вычисляется регрессионное значение. У процедуры `regress()` три аргумента: векторы узловых точек и значений зависимого параметра в узловых точках, а также степень полинома регрессионной зависимости.

Пример 7.14. Полиномиальная регрессия

Примеры выполнения полиномиальной регрессии можно видеть на рис. 7.45.

В документе определена функция двух аргументов. Первый — точка, в которой вычисляется регрессионное значение, а второй — степень регрессионного полинома. Для сравнения кроме статистических данных (отдельные точки на графике) построено три полинома разной степени: первой, третьей и двадцатой. Очевидно, что степень полинома качественно влияет на характер зависимости. Может сложиться впечатление, что чем больше степень полинома, тем лучше, но это не всегда так.

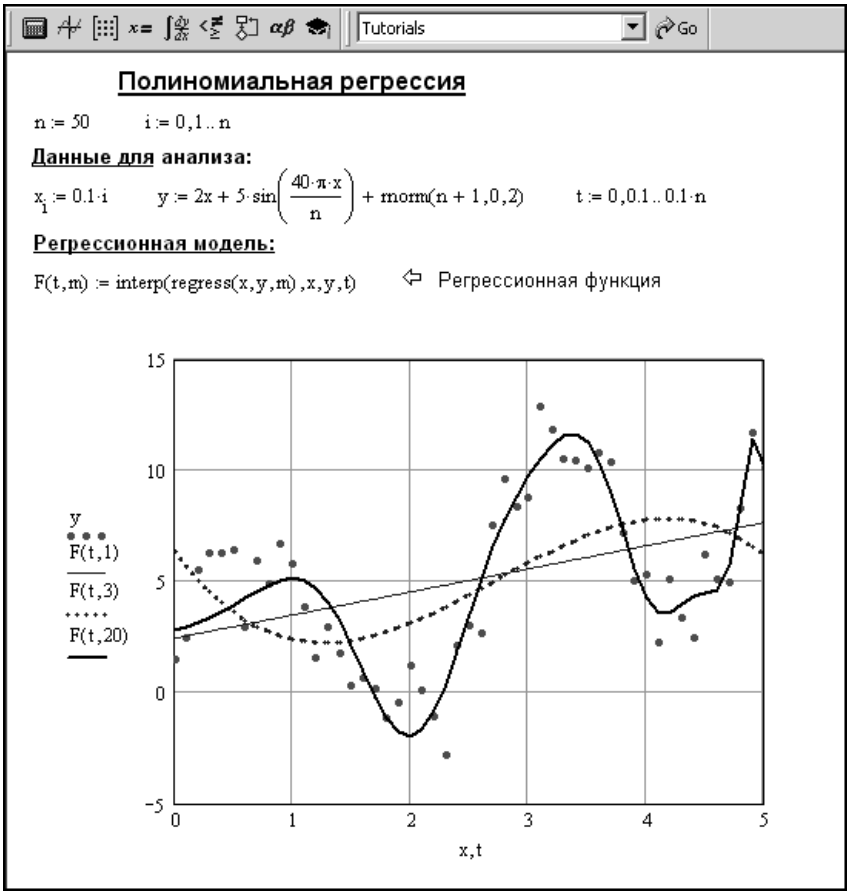


Рис. 7.45. Полиномиальная регрессия

Пример 7.15. Регрессия и интерполяция

Пример, позволяющий сравнить преимущества и недостатки регрессии и интерполяции, приведен в документе на рис. 7.46.

Здесь резко уменьшено число узловых точек, и по ним выполняется аппроксимация различными полиномами. Примечательно то, что если степень полинома на единицу меньше числа узловых точек, то аппроксимация фактически переходит в интерполяцию — полином будет проходить через все узловые точки. Однако в этом случае сталкиваемся с уже знакомой проблемой: на границах диапазона аппроксимирования могут наблюдаться существенные осцилляции. Полиномы более низких степеней хотя и не проходят через все узловые точки, зато более адекватно отображают характер функциональной зависимости.

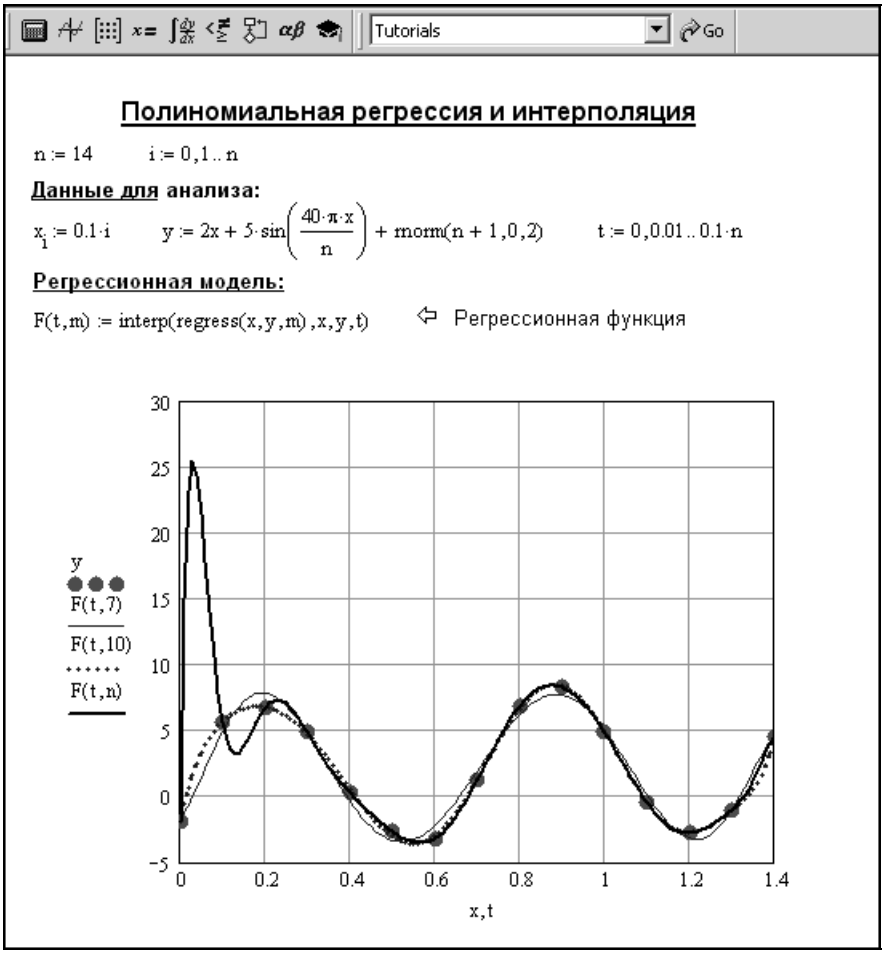


Рис. 7.46. Подобрав степень регрессионного полинома, можно выполнить интерполяцию

При выполнении аппроксимации зависимостей с помощью регрессионных моделей можно реализовать некое подобие интерполяции сплайнами. Так, существует возможность аппроксимировать зависимости полиномами второй степени. Область значений независимой переменной разбивается на подобласти, в пределах которых регрессионная зависимость дается полиномом второй степени. На границах подобластей выполняется сшивка таких полиномиальных зависимостей.

Чтобы выполнить указанную регрессию, в Mathcad вместо `regress()` нужно воспользоваться процедурой `loess()`, у которой три аргумента: вектор узловых точек, вектор значений зависимого параметра в узловых точках, а также параметр, определяющий размер подобластей.

Пример 7.16. Регрессия полиномами второй степени

Пример выполнения регрессии на основе описанного способа показан в документе на рис. 7.47.

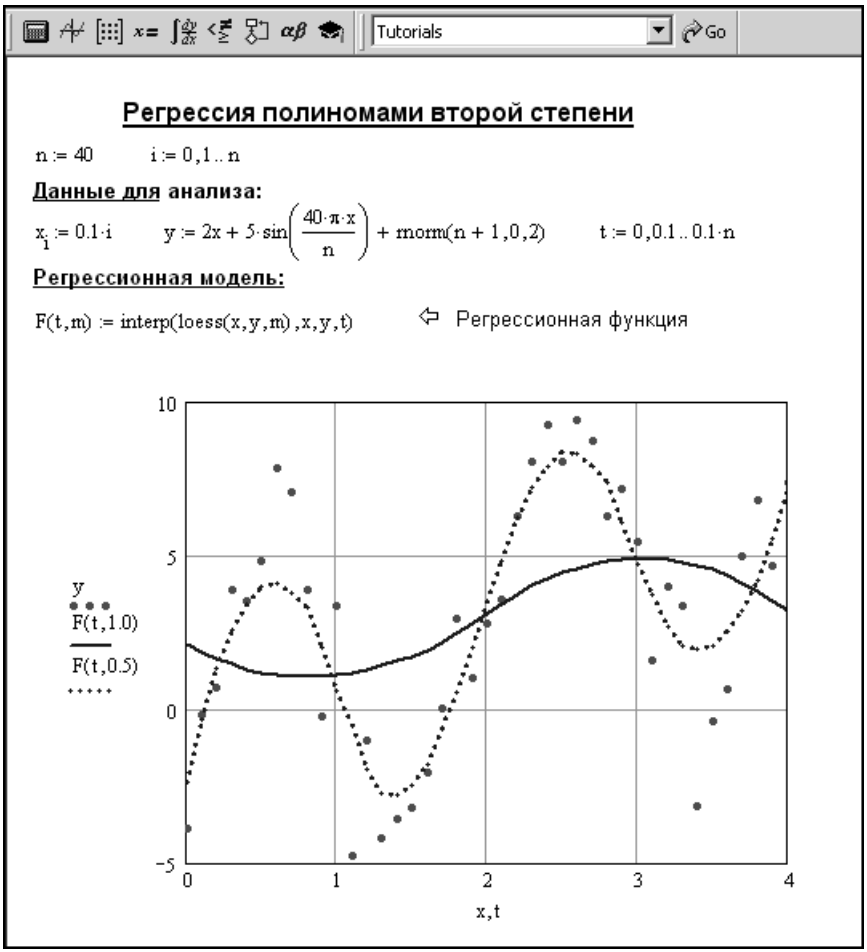


Рис. 7.47. Для регрессии использованы полиномы второй степени

Перечисленные процедуры выполнения полиномиальной регрессии могут использоваться для создания двухфакторных регрессионных моделей, в которых два независимых параметра. Делается это практически так же, как и в случае однофакторных моделей, только вместо вектора узловых точек указывают матрицу узлов (матрица состоит из двух столбцов, определяющих узловые точки по каждому из факторов).

Возможности Mathcad для построения различного типа регрессионных моделей достаточно широки. Среди встроенных утилит Mathcad можно найти процедуры на любой, как говорится, вкус. Перечень процедур, предназначенных для создания регрессионных зависимостей по статистическим данным, приведен в табл. 7.5.

Таблица 7.5. *Процедуры для создания регрессии*

Процедура	Описание
`expfit(x,y,g)`	Экспоненциальная регрессия. Регрессионная модель на основе выражения $y = a\exp(bx) + c$. В процессе построения модели находятся параметры a, b и c. Аргументами процедуры указывают вектор узловых точек `x`, вектор значений зависимого параметра в узловых точках `y`. Можно указать также третий параметр `g` — вектор с начальными оценочными значениями для параметров a, b и c
`logfit(x,y,g)`	Логарифмическая регрессия. Регрессионная модель на основе выражения $y = a\ln(x + b) + c$. В процессе построения модели находятся параметры a, b и c. Аргументами процедуры указывают вектор узловых точек `x`, вектор значений зависимого параметра в узловых точках `y`, третий параметр `g` — вектор с начальными оценочными значениями для параметров a, b и c
`lnfit(x,y)`	Упрощенная логарифмическая регрессия. Регрессионная модель на основе выражения $y = a\ln(x) + c$. В процессе построения модели находятся параметры a и c. Аргументами процедуры указывают векторы узловых точек `x` и значений зависимого параметра в узловых точках `y`
`lgsfit(x,y,g)`	Логистическая регрессия. Регрессионная модель на основе выражения $y = a/(1 + b\exp(-cx))$. Ищутся параметры a, b и c. Аргументы процедуры — векторы `x` узловых точек, `y` значений зависимого параметра в узловых точках, `g` начальных оценочных значений для параметров a, b и c
`pwrfit(x,y,g)`	Степенная регрессия. Регрессионная модель на основе выражения $y = ax^b + c$. Ищутся параметры a, b и c. Аргументы процедуры — векторы `x` узловых точек, `y` значений зависимого параметра в узловых точках, `g` начальных оценочных значений для параметров a, b и c

Таблица 7.5 *(окончание)*

Процедура	Описание
`sinfit(x,y,g)`	Тригонометрическая регрессия. Регрессионная модель на основе выражения $y = a\sin(x+b)+c$. Ищутся параметры a, b и c. Аргументы процедуры — векторы `x` узловых точек, `y` значений зависимого параметра в узловых точках, `g` начальных оценочных значений для параметров a, b и c
`linfit(x,y,F)`	Регрессионная модель строится в виде линейной комбинации функций, указанных в качестве элементов вектора `F` (третий аргумент процедуры). Первые два аргумента стандартные — векторы узловых точек и значений зависимого параметра в этих узловых точках
`genfit(x,y,g,F)`	Общая регрессионная модель. Строится на основе некоторой функции `f` (определяемой пользователем), зависящей от `x` и набора параметров, которые ищутся в процессе построения регрессионной модели. Параметры реализуются в виде элементов некоторого вектора `u`, т. е. модель строится на основе функции `f(x,u)`. Если модель содержит `n` параметров, то элементами вектора `F` (четвертый аргумент процедуры) должны быть функции от `x` и `u`, вектор должен состоять из `n+1` элемента, первым элементом является функция `f(x,u)`, а прочие его элементы — частные производные от этой функции по каждому из искомых параметров (они, напомним, являются элементами вектора `u`)

Все перечисленные в табл. 7.5 процедуры в качестве результата возвращают вектор параметров регрессионной модели. Далее рассмотрим некоторые примеры построения регрессионных моделей на основе указанных встроенных процедур Mathcad.

Пример 7.17. Тригонометрическая регрессия

На рис. 7.48 приведен фрагмент документа, в котором выполняется тригонометрическая (синусоидальная) регрессия. Сразу отметим, что все прочие регрессионные модели (с предопределенного типа регрессионными функциями) строятся так же.

Фактически, построение модели разбито на два этапа. На первом находятся параметры регрессионной зависимости. При этом важен выбор начального приближения для параметров модели. Это стандартная ситуация для случая,

когда тип регрессионной функции определен не очень удачно. В данном примере это сделано специально, чтобы продемонстрировать важность предварительного выбора типа функциональной зависимости (как лучше это делать — вопрос отдельный, выходящий за рамки этой книги). После того как параметры модели найдены, определяется сама регрессионная функция. График регрессионной кривой (и базовые точки, на основе которых строится регрессия) можно видеть в нижней части рис. 7.48. Как видим, аппроксимация оставляет желать лучшего. Причина, как только что было отмечено, заключается в неудачной базовой функции модели.

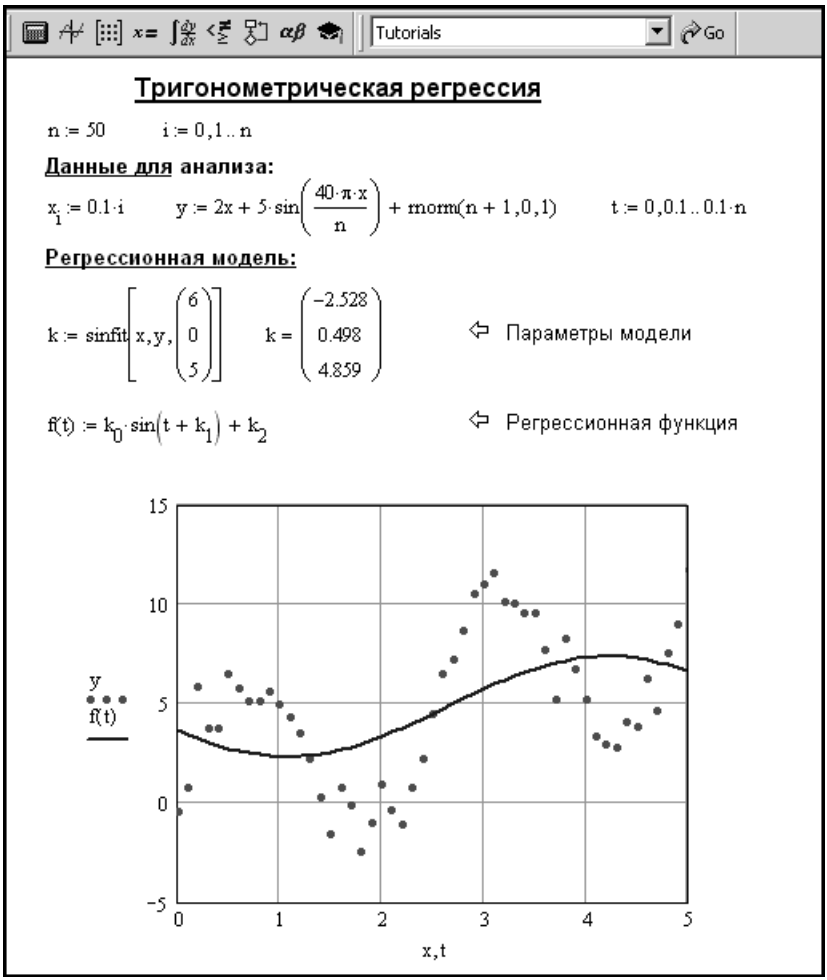


Рис. 7.48. Тригонометрическая регрессия

Пример 7.18. Регрессия на основе комбинации функций

Несколько иной подход продемонстрирован в документе на рис. 7.49. Там идентичный набор данных аппроксимируется в виде линейной комбинации двух функций: линейной по аргументу и тригонометрической. Выбор именно этих функций связан со способом генерирования узловых точек и значений зависимого параметра в них.

Рис. 7.49. Регрессия в виде линейной комбинации базисных функций

Зависимость формировалась именно на основе функций, которые и выбраны в качестве базисных при создании регрессионной модели. Разумеется, это не

совсем честно, поскольку в реальных ситуациях исследователь подобного рода информацией не обладает. Но поскольку пример иллюстративный, на такие тонкости особого внимания здесь обращать не будем.

Базисные функции в документе на рис. 7.49 инициализированы как элементы вектора, который указывается третьим аргументом процедуры вычисления параметров регрессионной модели `linfit()`. Это в последующем позволит, кроме прочего, определить регрессионную функцию в виде скалярного произведения двух векторов: вектора параметров регрессии и вектора базисных функций. Графики, приведенные в документе, свидетельствуют о том, что данный тип регрессии достаточно удачен (во всяком случае, по сравнению с предыдущим примером).

Пример 7.19. Регрессия общего вида

Базисная тригонометрическая функция (синус), использованная при создании регрессионной модели, имеет масштабированный аргумент. Коэффициент масштабирования достаточно нетривиален. Он был выбран, как отмечалось, исключительно на основе не очень легитимной информации о способе генерирования базисных точек. Усложним задачу и предположим, что этот коэффициент неизвестен. Тогда речь идет о создании регрессионной модели общего типа на основе процедуры `genfit()`. В ней три неизвестных параметра: коэффициенты при линейной функции, у синуса и масштабный множитель аргумента синуса. Документ с примером построения такого типа модели показан на рис. 7.50.

В первую очередь инициализируется регрессионная функция, которая определяется как функция двух переменных: непосредственно аргумента и вектора неизвестных параметров модели. Далее задается вектор-функция двух аргументов (тех же самых). У вектора-функции четыре элемента: регрессионная функция и ее частные производные по параметрам модели. Также определяется вектор начальных приближений для параметров регрессионной модели. Они по возможности должны быть достаточно близки к искомым оптимизационным параметрам. Далее вычисляется вектор параметров модели и строится регрессионная зависимость. Следует отметить, что параметры вычислены неплохо, если сравнить их с оригинальными параметрами функции, использованной для генерирования набора статистических данных.

В заключение раздела и главы остановимся на таком вопросе, как сглаживание данных. Эта проблема хотя напрямую и не относится к построению

регрессионных моделей, концептуально близка к ней. В общем случае задача может быть сформулирована следующим образом. Представим, что проводится серия некоторых измерений. Обычно реальные значения искажаются за счет случайных (несистематических) ошибок. Такие погрешности желательно отсеять. Один из способов — сглаживание, т. е. корректировка статистических данных на основе значений в соседних узлах. В Mathcad есть несколько процедур, позволяющих выполнять сглаживание данных (табл. 7.6).

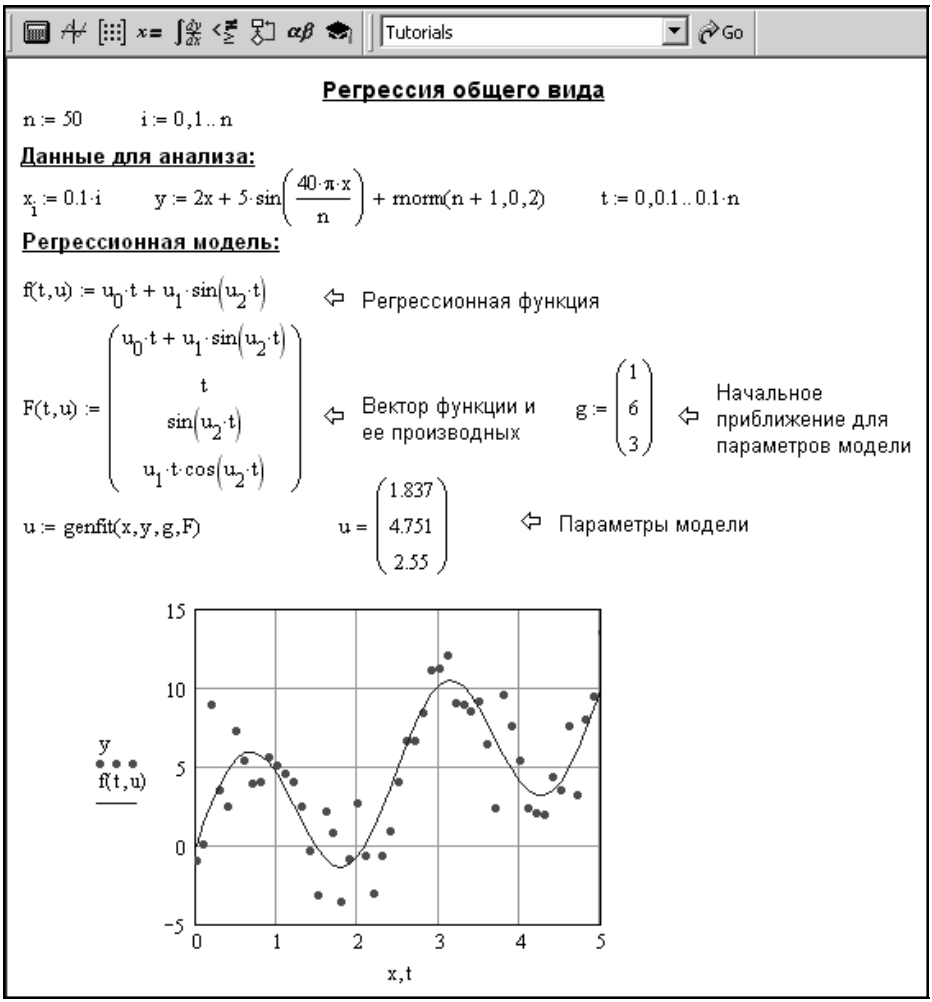


Рис. 7.50. Регрессия общего вида

Таблица 7.6. *Процедуры для сглаживания данных*

Процедура	Описание
`medsmooth(y,n)`	Набор данных (первый аргумент `y`) сглаживается по методу скользящей медианы. Второй аргумент `n` процедуры (нечетное число) определяет окно сглаживания
`ksmooth(x,y,b)`	Сглаживание данных по методу Гауссова распределения. Первые два аргумента процедуры — узловые точки и значения зависимого параметра в узловых точках, а третий аргумент задает ширину окна сглаживания
`supsmooth(x,y)`	Линейное сглаживание на основе метода наименьших квадратов. Аргументы — векторы значений узловых точек и зависимого параметра

Пример 7.20. Сглаживание данных

Пример сглаживания данных различными методами можно найти в документе на рис. 7.51.

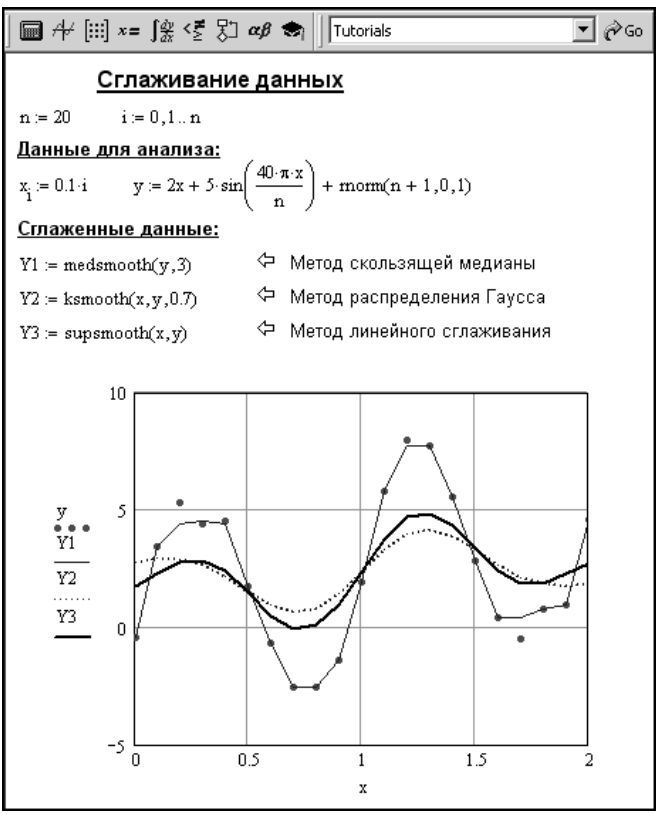


Рис. 7.51. Сглаживание данных

Сглаживание, выполненное различными методами, дает разные результаты. С другой стороны, выбирая параметры сглаживания (для процедур `medsmooth()` и `ksmooth()`), можно добиваться самых разных эффектов. Формально все это схоже с процедурой построения регрессионной кривой, однако следует понимать принципиальную разницу между двумя процессами. Если при построении регрессионной модели создается функциональная зависимость, которая не влияет на исходные данные, то в процессе сглаживания данных речь в общем случае идет об изменении всего базисного набора значений, причем изменения эти можно выполнять далеко неоднозначно.

Заключение

На сегодняшний день, к сожалению (или к счастью), нет такого приложения (и вряд ли оно когда-нибудь появится), которое могло бы решать абсолютно все задачи, что называется, в автоматическом режиме. Каким бы замечательным ни было приложение, есть задачи, с которыми справиться может только человек. Человеческой интуиции сложно что-либо противопоставить. Это следует осознавать и не требовать от Mathcad невозможного. В первую очередь замечание относится к новичкам, которые только начинают знакомство с математическими пакетами. Первое разочарование, как правило, наступает после осознания того факта, что для решения сложной задачи нередко приходится проделать подготовительную работу, объем которой сопоставим с усилиями на решение этой же задачи без помощи пакета. В качестве примера можно привести уравнения в частных производных. Ведь для их решения даже стандартными приемами требуется некоторая изобретательность. Естественным образом возникает вопрос: а могут ли быть полезны математические пакеты, и Mathcad в частности, в подобных ситуациях? Ответ положителен, и обусловлен он несколькими существенными причинами. Во-первых, всегда существует необходимость выполнять промежуточные вычисления, в которых специальный математический пакет будет существенным подспорьем. Во-вторых, как Mathcad, так и другие популярные пакеты позволяют представлять результат в весьма наглядном виде и его анализ становится делом несложным, а часто просто приятным. Уже этих двух обстоятельств достаточно для того, чтобы однозначно склониться в пользу поддержки и популяризации математических пакетов среди специалистов и исследователей.

В книге был рассмотрен ряд примеров и задач, на основе которых, собственно, и иллюстрировались возможности Mathcad. Они подбирались с тем расчетом, чтобы охватить все основные университетские математические курсы для студентов естественнонаучных специальностей. Выбор не случаен, поскольку это как раз тот класс задач, с которыми приходится сталкиваться на практике. Примеры, безусловно, иллюстративные. Кроме того, сам алгоритм решения определялся не столько соображениями математической оптимальности,

сколько желанием продемонстрировать возможности Mathcad, подчеркнуть наиболее важные моменты. Другими словами, внимание обращалось в первую очередь на способ получения результата, а уже во вторую — на простоту решения. Поэтому на практике, когда важен не способ решения, а результат, читатель может и должен проявить смекалку и инициативу, идти своим путем, а не проторенной дорожкой. Хотя, разумеется, нет ничего плохого в том, чтобы использовать уже имеющиеся заготовки. По крайней мере, это сэкономит время.

Примеры и задачи в книге отличаются по сложности и объему. Думается, это позволит читателю составить более полное представление о возможностях Mathcad при решении различных задач, уяснить для себя сильные и слабые стороны приложения. Ведь специальных математических пакетов существует на сегодня достаточно много. Поэтому для работы разумно выбрать тот из них, в котором более полно реализованы соответствующие алгоритмы. Эффективной также может оказаться работа сразу в нескольких пакетах. Хотя следует отметить, что Mathcad достаточно универсален и его возможностей вполне хватит для решения большинства практических задач. Это особенно справедливо, если речь идет об инженерных расчетах. Впрочем, область применения приложения простирается на физику, математику, экономику и социологию. Причем с каждой новой версией приложения области его применения расширяются.

Несколько замечаний относительно различных версий Mathcad. Несмотря на то, что книга писалась на основе Mathcad 13, примеры и задачи подбирались так, чтобы они работали и в более ранних версиях. В крайнем случае, можно внести минимальные изменения с учетом версии приложения. В целом же предложенные в книге алгоритмы и методы решения не привязаны к какой-то конкретной версии Mathcad, поэтому проблема совместимости сводится к формальной реализации алгоритмов в приложении.

Следует понимать, что в книге рассмотрены далеко не все аспекты, связанные с применением приложения Mathcad на практике. Выражаясь образно, за бортом остался ряд интересных тем, среди которых совместное использование Mathcad с другими приложениями, электронными таблицами, реализация в Mathcad документации, в том числе научной, особенности различных форматов, работа с электронными книгами и подключение дополнительных пакетов. Думается, читатель в случае необходимости найдет полезную для себя информацию из литературы, специально посвященной этим вопросам. Полезными также будут книги по другим аналогичным пакетам и системам. Вообще же, тематика, относящаяся к математическим пакетам, сейчас очень актуальна, появляется много новых интересных изданий и статей. Тем, кто работает в данной области, приходится постоянно пополнять свой багаж знаний и навыков. Это очень полезное и перспективное занятие, поэтому хочется пожелать читателю успехов на этом трудном, но захватывающе интересном пути.

Приложение

Описание компакт-диска

На компакт-диске можно найти файлы с примерами (их около 170) из книги. Для удобства все примеры размещены в отдельных папках в соответствии с теми главами, в которых примеры рассматривались. Кроме примеров в *главах 6* и *7* есть задачи. Деление на примеры и задачи достаточно условное. Разница, по большому счету, состоит только в способе именования файлов. Задачи и примеры нумеруются независимо друг от друга. Нумерация в каждой главе своя. Названия файлов на компакт-диске в точности совпадают с названиями примеров в книге. Файлы задач следует искать по номерам. Например, файл с названием Задача 7.8. Интерполяция сплайнами.xmcd соответствует восьмой задаче из *главы 7*, а файл Пример 3.5. Спираль Архимеда.xmcd — примеру с номером 5 из *главы 3*.

При работе с файлами, находящимися на компакт-диске, необходимо учитывать следующие замечания.

1. Примеры не привязаны к версии Mathcad 13, однако поскольку соответствующие файлы сохранялись именно в Mathcad 13, открывать их следует также в Mathcad 13.
2. В некоторых примерах, где использовался генератор случайный чисел, при открытии документа каждый раз можно наблюдать кривые и массивы данных, отличные от тех, что представлены в книге. Исключение составляют примеры, в которых генератор случайных чисел инициализируется в явном виде.
3. В примере 2.17 вместо команды присваивания значения переменной, определяющей рабочий каталог, указана команда определения значения соответствующей переменной.
4. В *главе 3*, помимо прочего, обсуждаются вопросы создания анимации. Как результат генерируются два файла AVI, которые также представлены на компакт-диске. Их можно просматривать независимо от файлов с соответствующими примерами.

5. Примеры 3.25 и 3.26 *главы 3* подразумевают чтение и запись внешних файлов. В силу очевидных причин в файлах примеров на компакт-диске ссылки на внешние файлы отличны от тех, что приведены и описаны в книге.

Содержание прилагаемого компакт-диска приведено в табл. П1.

Таблица П1. *Содержание компакт-диска*

Папка или файл	Число файлов	Описание
Chapter_2	55	Примеры из второй главы книги. Файлы Пример 2.1. … .xmcd — Пример 2.54. … .xmcd. Кроме этого в папке размещен файл data.txt, который записывается в результате открытия файла Пример 2.17. … .xmcd
Chapter_3	32	Примеры из третьей главы книги. Файлы Пример 3.1. … .xmcd — Пример 3.28. … .xmcd (28 файлов), 2 видеофайла Волна.avi и Мембрана.avi для примера 3.28, а также файлы Girl.jpg с изображением и Beauty.bmp (в который записывается изображение) для примеров 3.25 и 3.26
Chapter_4	19	Примеры из четвертой главы книги. Файлы Пример 4.1. … .xmcd — Пример 4.19. … .xmcd
Chapter_5	20	Примеры из пятой главы книги. Файлы Пример 5.1. … .xmcd — Пример 5.20. … .xmcd
Chapter_6	22	Примеры и задачи из шестой главы книги. Файлы с задачами Задача 6.1. … .xmcd — Задача 6.20. … .xmcd и файлы примеров Пример 6.1. … .xmcd и Пример 6.2. … .xmcd
Chapter_7	30	Примеры и задачи из седьмой главы книги. Файлы с задачами Задача 7.1. … .xmcd — Задача 7.10. … .xmcd и файлы примеров Пример 7.1. … .xmcd — Пример 7.20. … .xmcd
Readme.doc	1	Файл с описанием компакт-диска

Литература

1. Березин И. С., Жидков Н. П. Методы вычислений. В 2 т. — М.: Физматлит, 1962.
2. Бермант А. Ф., Араманович И. Г. Краткий курс математического анализа. — М.: Наука, 1973.
3. Бидасюк Ю. М. Самоучитель Mathsoft MathCAD 11. — М.: Диалектика, 2004.
4. Боярчук А. К., Ляшко И. И., Гай Я. Г., Головач Г. П. Справочное пособие по высшей математике. В 5 т. — М.: Едиториал УРСС, 2000.
5. Будак Б. М., Самарский А. А., Тихонов А. Н. Сборник задач по математической физике: Учебное пособие. — М.: Наука, 1980.
6. Воскобойников Ю., Очков В. Программирование и решение задач в пакете Mathcad. — Новосибирск: НГАСУ, 2002.
7. Гольдберг О. Переходные процессы в электрических машинах и аппаратах и вопросы их проектирования. — М.: Высшая школа, 2001.
8. Гречко Л. Г., Сугаков В. И., Томасевич О. Ф., Федорченко Ф. М. Сборник задач по теоретической физике. — М.: Высшая школа, 1984.
9. Гурский Д. Вычисления в Mathcad. — Мн.: Новое знание, 2003.
10. Демидович Б. П. Сборник задач и упражнений по математическому анализу. — М.: Наука, 1977.
11. Дьяконов В. Mathcad 2001: Специальный справочник. — СПб.: Питер, 2001.
12. Дьяконов В. Mathcad 2001: Учебный курс. — СПб.: Питер, 2001.
13. Дьяконов В. VisSim+Mathcad+MATLAB: Визуальное математическое моделирование. — М.: Солон-Пресс, 2004.
14. Дьяконов В. П. Mathcad 8-12 для всех. — М.: Солон-Пресс, 2005.
15. Дьяконов В. П. Энциклопедия Mathcad 2001i и Mathcad 11. — М.: Солон-Пресс, 2004.

16. Ивановский Р. Компьютерные технологии в науке: Практика применения систем Mathcad Pro. — М.: Высшая школа, 2003.

17. Каганов В. Компьютерные вычисления в средах Excel и Mathcad. — М.: Горячая линия — Телеком, 2003.

18. Кирьянов Д. Mathcad 12 в подлиннике. — СПб.: БХВ-Петербург, 2004.

19. Кирьянов Д. Самоучитель MathCAD 11. — СПб.: БХВ-Петербург, 2003.

20. Кирьянов Д. Самоучитель Mathcad 12. — СПб.: БХВ-Петербург, 2004.

21. Кирьянов Д. Самоучитель Mathcad 2001. — СПб.: БХВ-Петербург, 2001.

22. Кузнецов А., Мельников О., Новиков В., Черняк А. Математика для экономистов на базе Mathcad. — СПб.: БХВ-Петербург, 2003.

23. Ландау Л. Д., Лившиц Е. М. Механика / Под ред. Л. П. Питаевского. 5-е изд., стер. — М.: Физматлит, 2001.

24. Макаров Е. Инженерные расчеты в MathCAD: Учебный курс. — СПб.: Питер, 2003.

25. Очков В. Физические и экономические величины в Mathcad и Maple. — М.: Финансы и статистика, 2002.

26. Очков В. Ф. Mathcad 12 для студентов и инженеров. — СПб.: БХВ-Петербург, 2005.

27. Поршнев С. Компьютерное моделирование физических процессов с использованием пакета Mathcad: Учебное пособие. — М.: Горячая линия — Телеком, 2002.

28. Пугачев В. С. Теория вероятностей и математическая статистика. — М.: Наука, 1979.

29. Тарасевич Ю. Информационные технологии в математике. — М.: Солон-Пресс, 2003.

30. Тихонов А. Н., Самарский А. А. Уравнения математической физики: Учебное пособие. — М.: Наука, 1977.

31. Черняк А., Черняк Ж., Доманова Ю. Высшая математика на базе Mathcad. — СПб.: БХВ-Петербург, 2004.

32. Эльсгольц Л. Э. Дифференциальные уравнения и вариационное исчисление. Учеб. для физич. спец. ун-тов. — М.: Наука, 1969.

Предметный указатель

В

Векторизация 149
Вероятность 469
Выборочный момент 484

Г

Геликоид 226
Гистограмма 487
Группа 41

Д

Диаграмма 234
Дисперсия 479
Дробь 77, 356

И

Инициализация:
  глобальная 67
  локальная 67
Интеграл 344, 390
  двойной 401, 404
  криволинейный 392, 396
  тройной 407
Интерполяция 452
  Лагранжа 457
  сплайнами 453

К

Ковариация 489
Колонтитул 38
Константа 85, 290
Конус 228
Координаты:
  полярные 192
  сферические 225
Коэффициент
  корреляции 489

М

Математическое
  ожидание 479
Матрица:
  обратная 145, 358
  определитель 146
  транспонированная 144, 358
Меню:
  Edit 6, 11
  File 6, 8
  Format 6, 17
  Help 6, 23
  Insert 6, 14
  Symbolics 6, 20
  Tools 6, 19
  View 6, 12
  Window 6, 22

Метод:
градиентного спуска 451
дихотомии 437
итераций 443, 448
Ньютона 441, 449
хорд 439

О

Оператор:
логический 267, 304
пользователя 166
условный 269, 305
цикла 277, 280, 307

П

Панель:
инструментов 6, 25, 31
математическая 28
меню 6
ресурсов 7, 30
форматирования 6, 27
элементов управления 30
Параболоид 236
Предел 371
Преобразование:
Лапласа 361
Фурье 360
Произведение векторное 147
Производная 336, 377
в полярных координатах 384
высокого порядка 342, 388
неявной функции 380
параметрическая 382

Р

Распределение:
Вейбулла 473
нормальное 470
показательное 471
Стьюдента 471
Рекурсия 281
Ряд:
Тейлора 366
Фурье 412

С, Ф

Сглаживание
данных 501
Сложение чисел 64
Спираль 229
Архимеда 192
Среднее значение 485
Строка:
заголовка 6
состояния 7
Сумма ряда 370
Сценарий 289
Формат 71, 75

Ч, Э

Число:
восьмеричное 83
двоичное 82
комплексное 78
шестнадцатеричное 83
Эллипсоид 225