Дмитрий Кирьянов

bhv®

Mathcad 15/ Mathcad Prime 1.0

+ видеокурс
+ книга по вычислительной физике

- Работа в среде Mathcad 2001/2001i/ 11/12/13/14/15/Prime 1.0
- Аналитические преобразования и численные методы
- Дифференцирование и интегрирование
- Вычислительная линейная алгебра
- Нелинейные уравнения и оптимизация
- Статистика, обработка данных, спектральный анализ
- Дифференциальные уравнения

Материалы на www.bhv.ru

Наиболее полное руководство

В ПОДЛИННИКЕ®

Дмитрий Кирьянов

Mathcad 15/ Mathcad Prime 1.0

Санкт-Петербург
«БХВ-Петербург»
2012

УДК 681.3.06
ББК 32.973.26-018.2
К43

Кирьянов Д. В.

К43 Mathcad 15/Mathcad Prime 1.0. — СПб.: БХВ-Петербург, 2012. — 432 с.: ил. + Видеокурс — (В подлиннике)

ISBN 978-5-9775-0746-2

Книга посвящена методике решения задач высшей математики при помощи программы Mathcad. Приводятся примеры расчета типовых задач линейной алгебры, математического анализа, дифференциальных уравнений, статистики и обработки данных. Объясняется работа численных алгоритмов, заложенных во встроенных функциях и операторах системы Mathcad. Предлагаются неочевидные приемы решения актуальных задач современной вычислительной науки. Описывается интерфейс Mathcad и его основные составные части, предоставляется необходимая справочная информация. Подробно описана работа в среде Mathcad 2001/2001i/11/12/13/14/15. Рассматриваются новые возможности и отличительные черты новой версии Mathcad Prime 1.0. На FTP-сервере издательства находятся мультимедийный видеокурс по работе с Mathcad, листинги примеров, справочник по вычислительной математике и учебник по вычислительной физике с примерами, оформленными в виде электронной книги Mathcad.

Для широкого круга пользователей

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	*Алексей Семенов*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.08.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 34,83.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 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 978-5-9775-0746-2

© Кирьянов Д. В., 2011
© Оформление, издательство "БХВ-Петербург", 2011

Оглавление

Предисловие **11**

Глава 1. Основные сведения о Mathcad **15**
1.1. Знакомство с Mathcad 15
1.1.1. Назначение Mathcad 15
1.1.2. Версии Mathcad 17
1.1.3. Интерфейс пользователя 18
1.1.4. Панели инструментов 19
1.2. Основы вычислений в Mathcad 23
1.2.1. Операторы численного и символьного вывода 23
1.2.2. Математические выражения и встроенные функции 24
1.2.3. Переменные и оператор присваивания 27
1.2.4. Одновременное присваивание и вычисление результата 28
1.2.5. Функции пользователя 29
1.2.6. Основные типы данных 30
Действительные числа 30
Комплексные числа 31
Встроенные константы 32
Строковые переменные 33
1.2.7. Ранжированные переменные и матрицы 34
1.2.8. Размерные переменные 35
1.3. Ввод и редактирование формул 38
1.3.1. Элементы интерфейса редактора формул 38
1.3.2. Ввод формул 39
1.3.3. Перемещение линий ввода внутри формул 40
1.3.4. Изменение формул 41
Вставка оператора 41
Выделение части формулы 42
Удаление части формулы 43
О форматировании формул 43
1.3.5. Программирование 44
Принцип программирования в Mathcad 44
Добавление строк программного кода 45
Локальное присваивание (←) 47

1.4. Графики ... 47
1.4.1. Типы графиков ... 47
1.4.2. Создание графика ... 48
1.4.3. График двух векторов ... 49
1.4.4. График функции ... 49
1.4.5. Построение нескольких рядов данных ... 50
1.4.6. Форматирование графиков ... 52
1.4.7. Трехмерные графики ... 52
1.5. Отладка и комментирование программ ... 53
1.5.1. Сообщения об ошибках ... 53
1.5.2. Отладка программ ... 54
1.5.3. Обработка ошибок в программных модулях ... 57
1.5.4. Текст и комментарии ... 58
Текстовые блоки ... 58
Комментарии ... 59
1.6. Управление файлами документов ... 60
1.6.1. Создание нового документа ... 60
1.6.2. Открытие существующего документа ... 61
1.6.3. Сохранение документа ... 61
1.6.4. Управление вычислениями ... 63
1.7. О дополнительных элементах оформления документов ... 65
1.8. Ввод/вывод данных ... 69
1.8.1. Ввод/вывод в текстовые файлы ... 69
1.8.2. Ввод/вывод в файлы других типов ... 71
Графические файлы ... 71
Звуковые файлы ... 72
1.8.3. Универсальный импорт данных ... 73
Мастер импорта данных ... 73
Функция *READFILE* ... 75

Глава 2. Алгебраические вычисления ... 77
2.1. Операторы и функции ... 77
2.1.1. Арифметические операторы ... 78
2.1.2. Вычислительные операторы ... 78
2.1.3. Логические операторы ... 79
2.1.4. Матричные операторы ... 80
2.1.5. Операторы вычисления ... 81
2.1.6. Функции ... 81
2.2. Алгебраические преобразования ... 82
2.2.1. О способах символьных вычислений ... 82
2.2.2. Разложение выражений ... 84
Первый способ (разложение с помощью меню) ... 84
Второй способ (разложение с помощью оператора →) ... 85
2.2.3. Объединение выражений ... 86
2.2.4. Упрощение выражений ... 87
2.2.5. Разложение на множители ... 88
2.2.6. Разложение на элементарные функции ... 89
2.2.7. Разложение числа на простые числа ... 89
2.2.8. Разложение на простые дроби ... 90

2.2.9. Разложение на непрерывные дроби ... 90
2.2.10. Приведение подобных слагаемых ... 91
2.2.11. Вычисление коэффициентов полинома ... 92
2.2.12. Получение численного значения выражения ... 93
2.2.13. Явные вычисления ... 94
2.2.14. Вычисление рядов и произведений ... 95
2.2.15. Подстановка переменной ... 96
2.2.16. Вычисление предела ... 97
2.2.17. О специфике аналитических вычислений ... 97

Глава 3. Дифференцирование ... 99
3.1. Аналитическое дифференцирование ... 99
3.1.1. Аналитическое дифференцирование функции ... 99
3.1.2. Вычисление производной функции в точке ... 101
3.1.3. Определение функций пользователя через оператор дифференцирования ... 102
3.1.4. Дифференцирование при помощи меню ... 102
3.2. Численное дифференцирование ... 103
3.2.1. Дифференцирование в точке ... 103
3.2.2. Об алгоритме дифференцирования ... 104
3.3. Производные высших порядков ... 106
3.4. Частные производные ... 108
3.4.1. Частные производные ... 108
3.4.2. Примеры: градиент, дивергенция и ротор ... 110
3.4.3. Пример: якобиан ... 115
3.5. Разложение функции в ряд Тейлора ... 117
3.5.1. Разложение в ряд при помощи меню ... 117
3.5.2. Оператор разложения в ряд ... 118

Глава 4. Интегрирование ... 121
4.1. Определенный интеграл ... 121
4.1.1. Оператор интегрирования ... 121
4.1.2. О выборе алгоритма численного интегрирования ... 123
4.1.3. О традиционных алгоритмах интегрирования ... 125
4.1.4. Алгоритм Ромберга ... 128
4.2. Неопределенный интеграл ... 128
4.2.1. Символьное интегрирование ... 129
4.2.2. Интегрирование при помощи меню ... 129
4.3. Интегралы специального вида ... 130
4.3.1. Интегралы с бесконечными пределами ... 130
4.3.2. Расходящиеся интегралы ... 130
4.3.3. Интеграл с переменным пределом ... 131
4.3.4. Кратные интегралы ... 131
4.3.5. Пример: длина дуги кривой ... 132
4.4. Интеграл Фурье ... 133
4.4.1. Об интегральных преобразованиях функций ... 134
4.4.2. Аналитическое преобразование Фурье ... 134
4.4.3. Дискретное преобразование Фурье ... 135
4.4.4. Преобразование Фурье комплексных данных ... 137
4.4.5. Двумерное преобразование Фурье ... 139

4.4.6. Пример: артефакты дискретного Фурье-преобразования ... 139
Влияние конечности интервала выборки ... 139
Сдвиг ноль-линии ... 140
Маскировка частот ... 142
4.5. Другие интегральные преобразования ... 143
4.5.1. Преобразование Лапласа ... 143
4.5.2. Z-преобразование ... 144
4.5.3. Вейвлет-преобразование ... 145
Встроенная функция вейвлет-преобразования ... 145
Программирование других вейвлет-преобразований ... 147

Глава 5. Нелинейные алгебраические уравнения ... 149
5.1. Аналитическое решение уравнений ... 149
5.1.1. Вычислительный блок *Given* / *Find* ... 150
5.1.2. Одно уравнение ... 151
Использование блока *Given* / *Find* ... 151
Ключевое слово *solve* ... 154
5.1.3. Системы уравнений ... 154
5.1.4. Решение уравнений при помощи меню ... 155
5.2. Численное решение уравнений ... 156
5.2.1. Системы уравнений: функция *Find* ... 157
Вычислительный блок *Given / Find* ... 157
Одно уравнение с одним неизвестным ... 158
Системы уравнений ... 159
Системы уравнений и неравенств ... 161
5.2.2. Уравнение с одним неизвестным: функция *root* ... 161
5.2.3. Корни полинома: функция *polyroots* ... 164
5.2.4. Локализация корней ... 165
5.3. О численных методах ... 166
5.3.1. Метод секущих: функция *root* ... 167
5.3.2. Градиентные методы: функция *Find* ... 168
Принцип действия градиентных алгоритмов ... 168
Выбор градиентного алгоритма ... 169
Параметры градиентных алгоритмов ... 170
5.3.3. Метод продолжения по параметру ... 171

Глава 6. Оптимизация ... 177
6.1. Поиск экстремума функции ... 177
6.1.1. Локальный экстремум ... 178
6.1.2. Условный экстремум ... 180
6.1.3. Экстремум функции нескольких переменных ... 182
6.1.4. Пример: линейное программирование ... 182
6.1.5. Аналитическое решение задач на экстремум ... 185
6.2. Приближенное решение алгебраических уравнений ... 186
6.3. Пример: некорректные задачи ... 190

Глава 7. Линейная алгебра ... 195
7.1. Простейшие матричные операции ... 195
7.1.1. Транспонирование ... 195

7.1.2. Сложение и вычитание ... 196
7.1.3. Умножение ... 198
7.2. Векторная алгебра ... 199
7.2.1. Модуль вектора ... 199
7.2.2. Скалярное произведение ... 200
7.2.3. Векторное произведение ... 200
7.2.4. Векторизация массива ... 201
7.3. Вычисление определителей и обращение квадратных матриц ... 202
7.3.1. Определитель квадратной матрицы ... 202
7.3.2. Ранг матрицы ... 203
7.3.3. Обращение квадратной матрицы ... 204
7.3.4. Возведение квадратной матрицы в степень ... 204
7.3.5. Матричные нормы ... 205
7.3.6. Число обусловленности квадратной матрицы ... 206
7.4. Вспомогательные матричные функции ... 207
7.4.1. Автоматическая генерация матриц ... 207
Создание матриц на основе некоторой функции ... 207
Создание матриц для построения 3D-графиков ... 208
Создание диагональных матриц ... 210
7.4.2. Разбиение и слияние матриц ... 210
Выделение подматрицы ... 210
Слияние матриц ... 212
Специфические преобразования матриц ... 212
7.4.3. Сортировка элементов матриц ... 213
7.4.4. Вывод размера матрицы ... 214

Глава 8. Системы линейных уравнений ... 217
8.1. Хорошо обусловленные системы с квадратной матрицей ... 218
8.1.1. Вычислительный блок *Given* / *Find* ... 218
8.1.2. Функция *lsolve* ... 220
8.2. Произвольные системы линейных уравнений ... 221
8.2.1. Переопределенные системы ... 221
О постановке задач ... 221
Псевдорешение (метод наименьших квадратов) ... 223
8.2.2. Недоопределенные системы ... 227
О постановке задач ... 227
Нормальное псевдорешение ... 228
8.2.3. Вырожденные и плохо обусловленные системы ... 231
Вырожденные СЛАУ ... 231
Плохо обусловленные системы ... 233
Метод регуляризации ... 236
8.3. Матричные разложения ... 238
8.3.1. СЛАУ с треугольной матрицей ... 239
8.3.2. Разложение Холецкого ... 240
8.3.3. *LU*-разложение ... 241
8.3.4. *QR*-разложение ... 243
8.3.5. *SVD*-(сингулярное) разложение ... 246
8.4. Собственные векторы и собственные значения матриц ... 248

Глава 9. Обыкновенные дифференциальные уравнения: динамические системы ... 251
9.1. О постановке задач ... 251
9.1.1. Задачи Коши для ОДУ ... 251
9.1.2. Фазовый портрет динамической системы ... 254
9.2. Дифференциальное уравнение N-го порядка ... 257
9.3. Система N дифференциальных уравнений ... 259
9.3.1. Встроенные функции для решения систем ОДУ ... 259
9.3.2. Решение одного уравнения (N=1) ... 262
9.3.3. Решение систем ОДУ в одной заданной точке ... 263
9.3.4. О численных методах ... 265
Рекомендации по выбору численного алгоритма ... 265
Число шагов ... 266
Погрешность алгоритмов решения ОДУ в точке ... 268
9.4. Жесткие системы ОДУ ... 270
9.4.1. Что такое жесткие ОДУ? ... 271
9.4.2. Функции для решения жестких ОДУ ... 273
9.4.3. Пример: химическая кинетика ... 274
Попытка решения стандартными методами ... 274
Применение специфических алгоритмов ... 276
9.5. Дополнительные функции решения ОДУ ... 278
9.5.1. Новые алгоритмы решения жестких и нежестких ОДУ ... 278
9.5.2. Решение линейных ОДУ ... 280

Глава 10. Обыкновенные дифференциальные уравнения: краевые задачи ... 283
10.1. О постановке задач ... 283
10.2. Решение краевых задач средствами Mathcad ... 285
10.2.1. Алгоритм стрельбы ... 285
10.2.2. Двухточечные краевые задачи ... 286
10.2.3. Краевые задачи с условием во внутренней точке ... 289
10.3. Задачи на собственные значения для ОДУ ... 292
10.4. Разностные схемы для ОДУ ... 295
10.4.1. О разностном методе ... 295
10.4.2. Жесткие краевые задачи ... 297
10.5. Нелинейные краевые задачи ... 300
10.5.1. О постановке задачи ... 300
10.5.2. Метод стрельбы ... 301
10.5.3. Разностные схемы ... 302

Глава 11. Дифференциальные уравнения в частных производных ... 307
11.1. О постановке задач ... 308
11.1.1. Классификация уравнений в частных производных ... 308
11.1.2. Пример: уравнение диффузии тепла ... 308
Двумерное динамическое уравнение ... 308
Стационарное двумерное уравнение ... 309
Одномерное динамическое уравнение ... 310
Линейное и нелинейное уравнения ... 311
Обратное уравнение теплопроводности ... 312

11.2. Разностные схемы ... 313
11.2.1. Явная схема Эйлера ... 314
Построение разностной схемы ... 314
Линейное уравнение ... 315
Нелинейное уравнение ... 317
Устойчивость ... 318
11.2.2. Неявная схема Эйлера ... 320
Построение неявной разностной схемы ... 320
Алгоритм прогонки ... 322
11.2.3. О возможности решения многомерных уравнений ... 323
11.3. Встроенные функции для решения уравнений в частных производных ... 325
11.3.1. Параболические и гиперболические уравнения ... 325
Вычислительный блок *Given / pdesolve* ... 326
Пример: волновое уравнение ... 328
Встроенная функция *numol* ... 329
11.3.2. Эллиптические уравнения ... 331
Уравнение Пуассона с нулевыми граничными условиями ... 331
Уравнение Пуассона с произвольными граничными условиями ... 333
Разностная схема для решения уравнения Пуассона ... 334
Решение уравнения диффузии тепла при помощи функции *relax* ... 335

Глава 12. Статистика ... 339
12.1. Статистические распределения ... 339
12.1.1. Статистические функции ... 340
12.1.2. Пример: нормальное (Гауссово) распределение ... 342
12.2. Выборочные статистические характеристики ... 346
12.2.1. Гистограммы ... 346
Гистограммы с произвольными интервалами ... 346
Гистограммы с равными интервалами ... 348
12.2.2. Среднее и дисперсия ... 349
12.2.3. Примеры: выборочная оценка дисперсии и среднего нормальной случайной величины ... 351
Интервальная оценка дисперсии ... 351
Проверка статистических гипотез ... 352
12.2.4. Корреляция ... 353
12.2.5. Функции корреляционного анализа сигналов ... 354
12.2.6. Коэффициенты асимметрии и эксцесса ... 355
12.2.7. Статистические функции матричного аргумента ... 355
12.3. Методы Монте-Карло ... 357
12.3.1. Генерация псевдослучайных чисел ... 357
Параметры генераторов псевдослучайных чисел ... 358
12.3.2. Генерация коррелированных выборок ... 359
12.3.3. Моделирование случайного процесса ... 360
12.3.4. Новые функции генерации случайных чисел ... 362

Глава 13. Интерполяция, регрессия, фильтрация ... 365
13.1. Интерполяция ... 365
13.1.1. Линейная интерполяция ... 366
13.1.2. Кубическая сплайн-интерполяция ... 368

13.1.3. Полиномиальная сплайн-интерполяция ... 370
13.1.4. Сплайн-экстраполяция ... 372
13.1.5. Экстраполяция функцией предсказания ... 373
13.1.6. Многомерная интерполяция ... 375
13.2. Регрессия ... 377
13.2.1. Линейная регрессия ... 377
13.2.2. Полиномиальная регрессия ... 380
Полиномиальная регрессия ... 380
Регрессия отрезками полиномов ... 381
Двумерная полиномиальная регрессия ... 382
13.2.3. Другие типы регрессии ... 384
13.2.4. Регрессия общего вида ... 385
13.2.5. Многопараметрическая полиномиальная регрессия ... 386
13.3. Фильтрация ... 388
13.3.1. Встроенные функции для сглаживания: ВЧ-фильтр ... 389
13.3.2. Скользящее усреднение: ВЧ-фильтр ... 390
13.3.3. Устранение тренда: НЧ-фильтр ... 391
13.3.4. Полосовая фильтрация ... 392
13.3.5. Спектральная фильтрация ... 394

Глава 14. Знакомство с Mathcad Prime ... 397
14.1. Интерфейс Mathcad Prime ... 397
14.2. Конвертация документов Mathcad в формат Mathcad Prime ... 400
14.3. Примеры расчетов ... 402
14.3.1. Типы областей ... 402
14.3.2. Формулы ... 403
14.3.3. Блоки решения ... 405
14.3.4. Текст ... 407
14.3.5. Размерные переменные ... 409
14.3.6. Отладка ... 409
14.4. Графики ... 410

Приложение. Описание компакт-диска ... 415
П.1. Содержимое диска ... 415
П.2. Как пользоваться расчетами Mathcad ... 415
П.3. Как пользоваться видеокурсом ... 418
П.4. Как пользоваться учебником по вычислительной физике ... 418

Предметный указатель ... 423

Ньютон. Если вы повернёте вон тот выключатель возле двери, что произойдёт?

Инспектор. Зажжётся свет.

Ньютон. Другими словами, вы замкнёте электрическую цепь. Вы что-нибудь смыслите в электричестве, Рихард?

Инспектор. Я ведь не физик.

Ньютон. Я тоже мало что понимаю в этом (...), нынче любой остолоп может зажечь электрическую лампочку — или взорвать атомную бомбу. (...) Почему же вы, хотя ничего не понимаете в электричестве, решаетесь включать свет?

Фридрих Дюренматт "Физики"
(разговор полицейского и пациента в сумасшедшем доме)

Предисловие

Mathcad — необычная программа. Она относится к классу приложений, называемых PSE (*problem solution environment — программная среда для решения задач*). Это подразумевает, что ее работа не определяется однозначно действиями пользователя (как, например, в текстовых редакторах и т. п.), а является (в большей степени) результатом работы встроенных алгоритмов, недоступных взору исследователя. Введя в редакторе Mathcad выражение, даже довольно простое, например `df(x)/dx=`, и получив ответ, многие даже не задумываются о том, что для его вычисления проделывается довольно сложная работа. Результат этой работы заранее не предопределен и зависит от целого ряда факторов, не представленных непосредственно на рабочей области документа (свойств функции `f`, параметров численного алгоритма дифференцирования, значения системных констант и т. д.). Поэтому, проводя даже очень простые расчеты, вам придется иногда сталкиваться с неочевидным поведением программы, которое нельзя понять без ясного представления об основах работы соответствующих алгоритмов, встроенных в Mathcad.

Приложение Mathcad, принадлежащее теперь компании PTC, — самый популярный из компьютерных математических пакетов, остающийся, бесспорно, на протяжении многих последних лет лидером в своем классе математического и образовательного программного обеспечения (ПО). С его помощью можно решать самые разные математические задачи и оформлять результаты расчетов на высоком профессиональном уровне, и сейчас уже сложно представить современного ученого, не пользующегося Mathcad. При помощи этого пакета осуществляются не только простые и вспомогательные вычисления, но и довольно сложные расчеты и научные исследования, использующие комбинации самых разных численных алгоритмов и аналитических преобразований.

Суммируя сказанное, мне хочется отметить четыре основные цели, которые преследует эта книга (первые две из них являются главными).

- Привести примеры решения в Mathcad стандартных задач и, по возможности, объяснить действие наиболее важных встроенных алгоритмов.
- Предложить читателю не совсем очевидные приемы решения актуальных задач современной вычислительной науки (многие из которых разработаны автором в ходе научной и учебной деятельности), чтобы читатель мог использовать их в качестве идиом для своей работы.
- Обозначить отличительные черты различных версий, вплоть до последней, Mathcad 15. Бóльшая часть такой информации оформлена в виде примечаний, чтобы книгу могли свободно применять все пользователи Mathcad версий 2000—15.
- Дать обширную справочную информацию, которая поможет читателю (уже накопившему опыт) быстро и эффективно работать в среде Mathcad.

Книга может использоваться как самоучитель, позволяющий "с нуля" освоить самые главные возможности вычислительной системы Mathcad и научиться с ее помощью решать все основные задачи математики. Тем не менее ее главная цель — изложить материал, делая акцент на решении конкретных математических проблем, причем я старался начинать рассказ с краткого определения математических понятий и терминов, конечно, имея в виду, что читатель обладает базовыми математическими знаниями.

Хочется сделать еще несколько замечаний по строению книги. Все листинги автономны и работают вне каких-либо дополнительных модулей. В листингах умышленно, чтобы не загромождать их, нет текстовых полей, — они содержат только расчеты по формулам. Все комментарии к ним находятся в тексте. Почти все графики вынесены в рисунки, причем, если они являются продолжением листингов, это помечено в подрисуночной подписи. Я старался сделать так, чтобы книга была максимально полезной пользователям всех последних версий Mathcad: 2000—15 и Mathcad Prime 1.0. Все файлы на прилагаемом к книге компакт-диске, насколько это возможно, записывались в формате Mathcad 2001. Функциональности, появившиеся в "новых" версиях Mathcad, отмечены значком в виде номера версии в кружке, например, ⑫.

Поскольку последние версии Mathcad русифицированы, причем очень качественно, то перевод объектов интерфейса соответствует локализованной русской версии. Лишь в некоторых случаях я немного отступал от нее (например, чаще употреблял вместо названия оператора "определение" слово "присваивание"). Я старался, чтобы рисунки в тексте по возможности не содержали элементов управления (т. е. не были привязаны к языку), а там, где без этого было не обойтись, чаще использовал русскую версию.

Обозначенные символом ⓘ разделы содержат информацию, относящуюся к особенностям численных алгоритмов, а символом 🛠 — полезным советам, идиомам и программным решениям самого автора. Эти разделы при первом знакомстве

с Mathcad могут быть пропущены. Знак 🌡 означает, что в данном разделе рассказывается о конкретном примере из области вычислительной математики, физики, химии, биологии или экономики.

К книге прилагается компакт-диск, который является не просто ее дополнением, а настоящим мультимедийным продолжением. Архив компакт-диска можно скачать по ссылке **ftp://85.249.45.166/9785977507462.zip**. Ссылка также доступна на странице книги на сайте **www.bhv.ru**. На компакт-диске вы найдете видеокурс, который поможет быстро и просто научиться основным приемам работы с Mathcad. Кроме того, компакт-диск содержит почти все листинги и расчеты графиков из книги, за исключением тривиальных, а также дополнительные примеры, которые лишь упомянуты в книге. Пиктограмма 💿 в тексте указывает на то, что соответствующие расчеты вынесены из книги на прилагаемый к ней компакт-диск.

Обо всех перечисленных возможностях я попытался в доступной форме рассказать в этой книге. Дополнительную информацию читатель может получить в Интернете на сервере производителя Mathcad **http://www.mathcad.com**, на обучающем ресурсе автора на сервере ИПМ РАН им. М. В. Келдыша **http://www.keldysh.ru/comma**, а также из моей книги "Вычислительная физика" (Полибук Мультимедиа, 2006), посвященной численным методам, полную версию которой вы также найдете на компакт-диске.

ГЛАВА 1

Основные сведения о Mathcad

В данной главе рассмотрены базовые приемы работы с Mathcad 2001—15. Мы дадим самые основные сведения, касающиеся интерфейса и возможностей Mathcad, пользуясь тем, что он интуитивен и похож на другие программы Windows.

1.1. Знакомство с Mathcad

Mathcad является математическим редактором, позволяющим проводить разнообразные научные и инженерные расчеты, начиная от элементарной арифметики и заканчивая сложными реализациями численных методов. С точки зрения классификации программного обеспечения, пакет Mathcad — типичный представитель класса PSE-приложений. Пользователи Mathcad — это студенты, ученые, инженеры, разнообразные технические специалисты и все, кому приходится проводить математические расчеты. Благодаря простоте применения, наглядности математических действий, обширной библиотеке встроенных функций и численных методов, возможности символьных вычислений, а также превосходному аппарату представления результатов (графики самых разных типов, мощных средств подготовки печатных документов и Web-страниц) Mathcad стал наиболее популярным математическим приложением.

1.1.1. Назначение Mathcad

Что из себя представляет система Mathcad? Следует хорошо представлять себе, что в состав Mathcad входят несколько интегрированных между собой компонентов:

- мощный текстовый редактор, позволяющий вводить, редактировать и форматировать как текст, так и математические выражения;
- вычислительный процессор, умеющий проводить расчеты по введенным формулам, используя встроенные численные методы;
- символьный процессор, позволяющий проводить аналитические вычисления и являющийся, фактически, системой искусственного интеллекта;
- огромное хранилище справочной информации, как математической, так и инженерной, оформленной в качестве интерактивной электронной книги.

Отличительной чертой Mathcad от большинства других современных математических приложений является его построение по принципу WYSIWYG ("What You See Is What You Get" — "что вы видите, то и получите"). Поэтому он очень прост в использовании, в частности, из-за отсутствия необходимости сначала писать программу, реализующую те или иные математические расчеты, а потом запускать ее на исполнение. Вместо этого достаточно просто вводить математические выражения с помощью встроенного редактора формул, причем в виде, максимально приближенном к общепринятому, и тут же получать результат. Кроме того, можно изготовить на принтере печатную копию документа или создать страницу в Интернете именно в том виде, который этот документ имеет на экране компьютера при работе с Mathcad, либо можно включить документ в структуру электронной книги Mathcad.

Создатели Mathcad сделали все возможное, чтобы пользователь, не обладающий специальными знаниями в программировании (а таких большинство среди ученых и инженеров), мог в полной мере приобщиться к достижениям современной вычислительной науки и компьютерных технологий. Для эффективной работы с редактором Mathcad достаточно базовых навыков пользователя. С другой стороны, профессиональные программисты (к которым относит себя и автор этих строк) могут извлечь из Mathcad намного больше, создавая различные программные решения, существенно расширяющие возможности, непосредственно заложенные в Mathcad.

В соответствии с проблемами реальной жизни, математикам приходится решать одну или несколько из следующих задач:

- ввод на компьютере разнообразных математических выражений (для дальнейших расчетов или создания документов, презентаций, Web-страниц или электронных книг);
- проведение математических расчетов (как аналитических, так и при помощи численных методов);
- подготовка графиков с результатами расчетов;
- ввод исходных данных и вывод результатов в текстовые файлы или файлы с базами данных в других форматах;
- подготовка отчетов работы в виде печатных документов;
- подготовка Web-страниц и публикация результатов в Интернете;
- получение различной справочной информации из области математики.

Со всеми этими (а также некоторыми другими) задачами с успехом справляется Mathcad:

- математические выражения и текст вводятся с помощью формульного редактора Mathcad, который по возможностям и простоте использования не уступает, к примеру, редактору формул, встроенному в Microsoft Word;
- математические расчеты производятся немедленно, в соответствии с введенными формулами;
- графики различных типов (по выбору пользователя) с богатыми возможностями форматирования вставляются непосредственно в документы;

- возможен ввод и вывод данных в файлы различных форматов;
- документы могут быть распечатаны непосредственно в Mathcad в том виде, который пользователь видит на экране компьютера, или сохранены в формате RTF для последующего редактирования в более мощных текстовых редакторах (например, Microsoft Word);
- возможно полноценное сохранение документов Mathcad в формате RTF-документов, а также Web-страниц: HTML и (начиная с 12-й версии) XML;

> (12) ***Примечание***
>
> Начиная с 12-й версии, файлы Mathcad имеют формат XMCD, являющийся разновидностью текстовой XML-разметки.

- имеется опция объединения разрабатываемых вами документов в электронные книги, которые, с одной стороны, позволяют в удобном виде хранить математическую информацию, а с другой — являются полноценными Mathcad-программами, способными осуществлять расчеты;
- символьные вычисления позволяют осуществлять аналитические преобразования, а также мгновенно получать разнообразную справочную математическую информацию;
- справочная система, а также многочисленные дополнительные материалы, оформленные в виде электронных книг (Ресурсы Mathcad), помогают быстро отыскать нужную математическую информацию или пример тех или иных расчетов.

Таким образом, следует хорошо представлять себе, что в состав Mathcad входит несколько интегрированных между собой компонентов — это мощный текстовый редактор для ввода и правки, как текста, так и формул, вычислительный процессор — для проведения расчетов согласно введенным формулам, а также символьный процессор, являющийся, по сути, системой искусственного интеллекта. Сочетание этих компонентов создает удобную вычислительную среду для разнообразных математических расчетов и, одновременно, документирования результатов работы.

1.1.2. Версии Mathcad

На протяжении последнего десятка лет базовая версия Mathcad менялась очень мало, несмотря на почти ежегодное обновление: 2000, 2001, 2001i, 11, 12, 13, 14, 15. За это время разработчик Mathcad — компания Mathsoft — была приобретена фирмой PTC (Parametric Technology Corporation), которой было принято решение полностью переписать код Mathcad на современной платформе Microsoft .Net, назвав полученный в результате продукт новым именем — Mathcad Prime.

Поэтому по состоянию на 2011 год компания PTC поддерживала две версии Mathcad:

- Mathcad 15 — "каноническая" версия Mathcad (которой и будет посвящена почти вся наша книга);

- Mathcad Prime 1.0 — новая редакция Mathcad (*см. главу 14*), кардинально отличающаяся организацией интерфейса, обладающая почти тем же функционалом, что и Mathcad 15.

1.1.3. Интерфейс пользователя

После того как Mathcad установлен на компьютере и запущен на исполнение, появляется основное окно приложения, показанное на рис. 1.1. В нем, сверху вниз, располагаются заголовок окна, строка меню, панели инструментов (стандартная и форматирования) и *рабочий лист*, или *рабочая область*, документа (worksheet). Новый документ создается автоматически при запуске Mathcad. В самой нижней части окна находится строка состояния. Таким образом, интерфейс пользователя Mathcad сходен с другими приложениями Windows, и, помня о близости редактора Mathcad к обычным текстовым редакторам, вы интуитивно поймете назначение большинства кнопок на панелях инструментов.

> 13 ***Примечание***
>
> Начиная с Mathcad 13 и 14 разработчики предусмотрели специальную область окна Mathcad, называемую **Окно трассировки** (Trace Window), а также дополнительную панель инструментов **Отладка** (Debug) (см. рис. 1.2 ниже).

Помимо элементов управления, характерных для типичного текстового редактора, Mathcad снабжен дополнительными средствами для ввода и редактирования математических символов, одним из которых является панель инструментов **Математика** или **Математические** (Math) (рис. 1.1). С помощью этой, а также ряда вспомогательных наборных панелей удобно осуществлять ввод уравнений.

Перечислим составные элементы интерфейса пользователя Mathcad:

- верхнее меню или строка меню (menu bar);

> ***Примечание***
>
> Далее в книге, говоря о совершении того или иного действия с помощью меню, последовательность выбора пунктов меню будем приводить сокращенно, разделяя их косыми чертами. Например, пункт **Панели инструментов** (Toolbars) меню **Вид** (View) обозначается как **Вид / Панели инструментов** (View / Toolbars).

- панели инструментов (toolbars) **Стандартная** (Standard), **Форматирование** (Formatting), **Ресурсы** (Resources), **Отладка** (Debug) и **Элементы управления** (Controls);
- панель инструментов **Математические** (Math) и доступные через нее дополнительные математические панели инструментов;
- рабочая область (worksheet);
- вспомогательное **Окно трассировки** (Trace Window);
- строка состояния (status line или status bar);
- всплывающие или контекстные меню (pop-up menus или context menus);
- диалоговые окна или диалоги (dialogs);

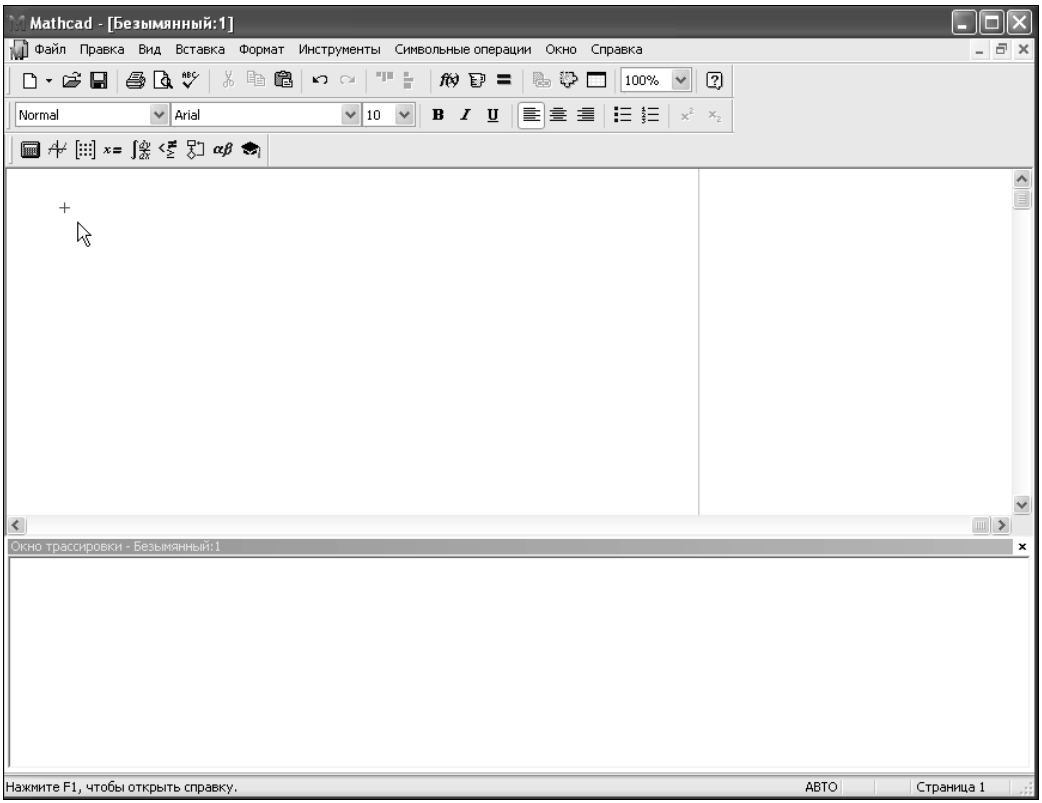


Рис. 1.1. Окно приложения Mathcad 15 с пустым документом

- окна **Ресурсов Mathcad** (Mathcad Resources) со встроенными примерами и дополнительной информацией.

Большинство команд можно выполнить как с помощью меню (верхнего или контекстного), так и панелей инструментов или клавиатуры.

1.1.4. Панели инструментов

Панели инструментов служат для быстрого (в один щелчок мыши) выполнения наиболее часто применяемых команд. Все действия, которые можно выполнить с помощью панелей инструментов, доступны и через верхнее меню. На рис. 1.2 изображено окно Mathcad с открытыми панелями инструментов (первые две из которых расположены непосредственно под строкой меню), а также дополнительными *математическими* (или *наборными*) панелями, о которых речь пойдет далее. Перечислим основные панели.

- **Стандартная** (Standard) — служит для выполнения большинства операций, таких как действия с файлами, редакторская правка, вставка объектов и доступ к справочным системам;
- **Форматирование** (Formatting) — для форматирования (изменения типа и размера шрифта, выравнивания и т. п.) текста и формул;

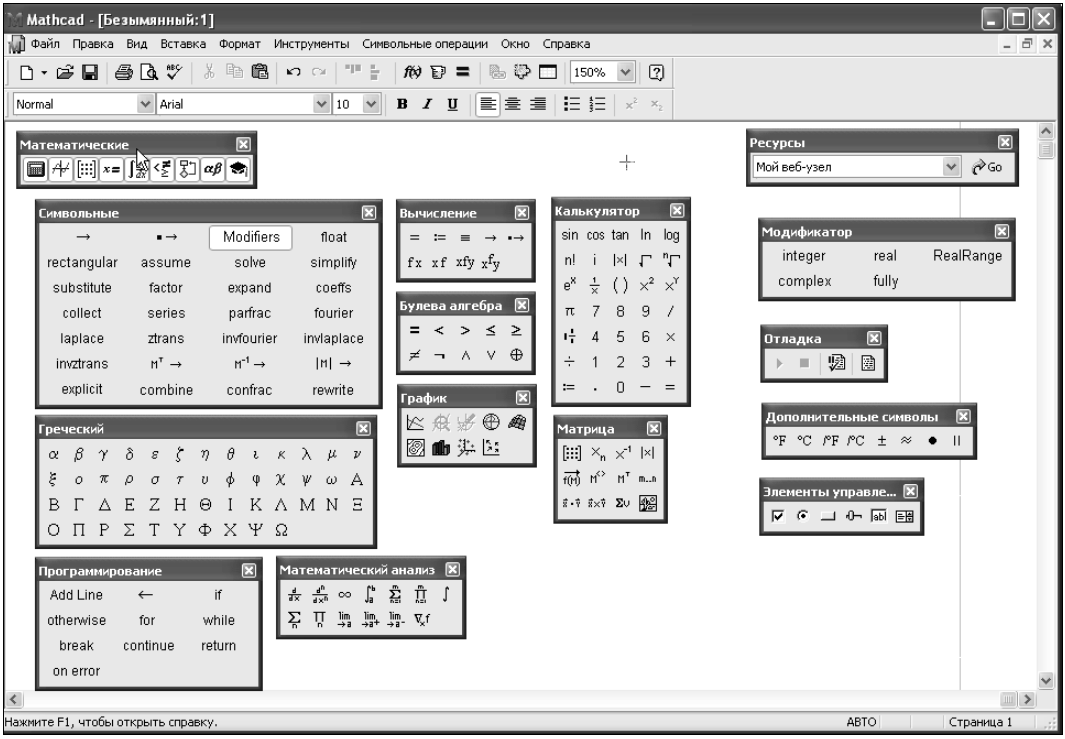


Рис. 1.2. Основные и математические (наборные) панели инструментов

- **Математические** (Math) — для вставки математических символов и операторов в документы;
- **Элементы управления** (Controls) — для вставки в документы стандартных элементов управления интерфейса пользователя (флажков проверки, полей ввода и т. п.);
- **Ресурсы** (Resources) — для быстрого вызова *ресурсов* Mathcad (примеров, учебников, электронных книг и т. п.);
- **Отладка** (Debug) — для управления отладкой Mathcad-программ.

13 ***Примечание***

Панель инструментов **Отладка** (Debug) появилась в версии Mathcad 13.

Группы кнопок на панелях инструментов разграничены по смыслу вертикальными линиями — *разделителями*. При наведении указателя мыши на любую из кнопок рядом с кнопкой появляется *всплывающая подсказка* — короткий текст, поясняющий назначение кнопки. Наряду со всплывающей подсказкой более развернутое объяснение готовящейся операции можно отыскать в строке состояния.

Панель **Математические** (Math) предназначена для вызова на экран еще девяти панелей (см. рис. 1.2), с помощью которых, собственно, и происходит вставка математических операций в документы. В прежних версиях Mathcad эти математиче-

ские панели инструментов назывались *палитрами* (palettes) или *наборными панелями*. Чтобы вызвать какую-либо из них, нужно нажать соответствующую кнопку на панели **Математические** (Math).

Перечислим назначение математических панелей:

- **Калькулятор** (Calculator) — служит для вставки основных математических операций, получила свое название из-за схожести набора кнопок с кнопками типичного калькулятора;
- **График** (Graph) — для вставки графиков;
- **Матрица** (Matrix) — для вставки матриц и матричных операторов;
- **Вычисление** (Evaluation) — для вставки операторов управления вычислениями;
- **Математический анализ** (Calculus) — для вставки операторов интегрирования, дифференцирования, суммирования и т. п.;
- **Булева алгебра** (Boolean) — для вставки логических (булевых) операторов;
- **Программирование** (Programming) — для программирования средствами Mathcad;
- **Греческие символы** (Greek) — для вставки греческих символов;
- **Символьные** (Symbolic) — для вставки символьных операторов;
- **Модификатор** (Modifier) — для вставки некоторых операторов (например, преобразования числа);
- **Дополнительные символы** (Custom Characters) — для вставки специальных символов (единиц измерения температуры и т. п.).

Примечание 1

Две последние (из перечисленных) панели используются довольно редко и потому вызываются не через панель **Математические** (Math), а посредством команды меню **Вид / Панели инструментов** (View / Toolbars).

13 ***Примечание 2***

Панель инструментов **Дополнительные символы** (Custom Characters) появилась в версии Mathcad 13 вместе с дополнительными размерностями температуры (в частности, градусов шкалы Цельсия и Фаренгейта).

При наведении указателя мыши на многие из кнопок математических панелей появляется всплывающая подсказка, содержащая еще и сочетание *горячих клавиш*, нажатие которых приведет к эквивалентному действию. Ввод действий с клавиатуры часто удобнее нажатия кнопок панелей инструментов, но требует большего опыта.

Вызвать любую панель на экран или скрыть ее можно с помощью пункта **Панели инструментов** (Toolbars) меню **Вид** (View), выбирая в открывающемся подменю имя нужной панели. Убрать любую панель с экрана можно еще и посредством контекстного меню, которое вызывается щелчком правой кнопки мыши в любом месте панели (например, на любой кнопке). В контекстном меню следует выбрать пункт

Скрыть (Hide). Кроме того, если панель *плавающая*, т. е. не прикреплена к основному окну (как, например, все панели на рис. 1.2), то ее можно отключить кнопкой закрытия.

Математические панели, в отличие от основных, можно вызвать или скрыть нажатием соответствующей кнопки панели **Математические** (Math) (см. рис. 1.2).

На некоторых рисунках этой главы (см., например, рис. 1.1) виден *курсор ввода* в виде небольшого крестика (на дисплее он имеет красный цвет). С его помощью отмечается незаполненное место в документе, куда в текущий момент можно вводить формулы или текст. Чтобы переместить курсор, достаточно щелкнуть указателем мыши в требуемом месте либо передвинуть его клавишами-стрелками. Если выполнить щелчок в области формулы или начать ввод выражения на пустом месте, вместо курсора появятся линии редактирования, отмечающие место в формуле или тексте, редактируемое в данный момент (см., например, рис. 1.6).

Если в какой-либо момент работы с Mathcad вам потребовалась помощь, выберите **Справка / Справка Mathcad (**Help / Mathcad Help) либо нажмите клавишу <F1>

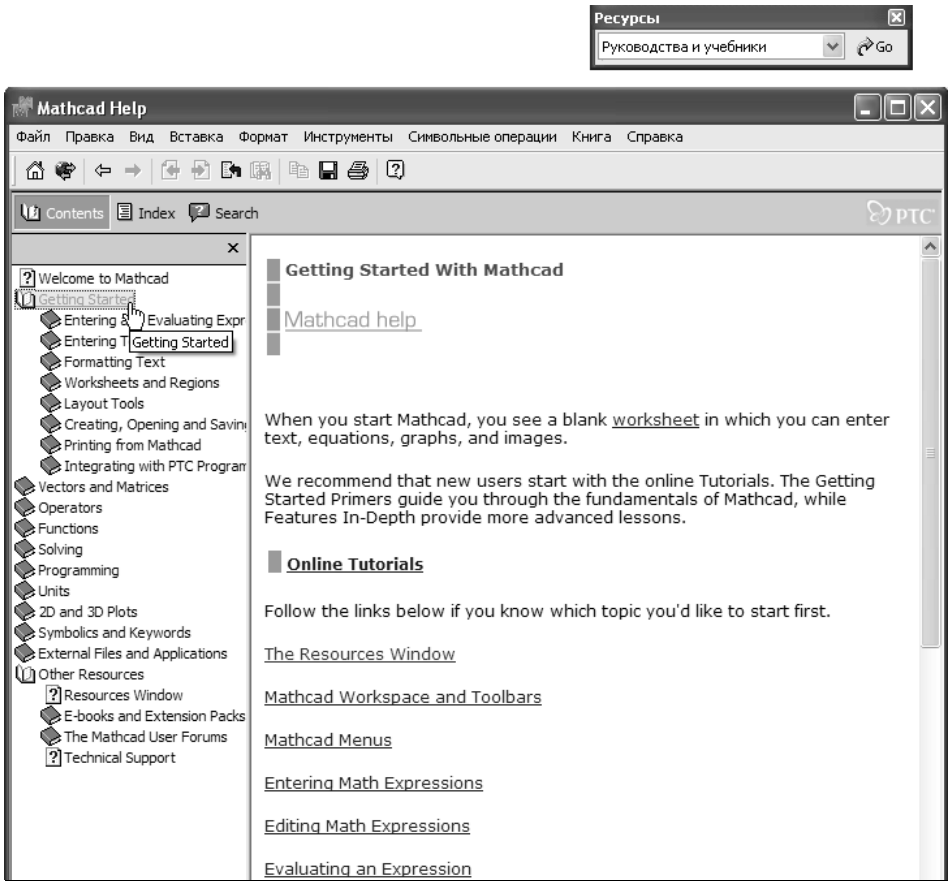


Рис. 1.3. Ресурсы Mathcad содержат большое количество справочной и учебной информации

или кнопку **Справка** (Help) со знаком вопроса на стандартной панели инструментов. Справка в Mathcad является контекстно-зависимой, т. е. ее содержание определяется тем, на каком месте документа она вызвана.

Помимо традиционной справочной системы, Mathcad содержит ряд источников дополнительной информации, в том числе онлайновой, а также примеры расчетов, которые оформлены в виде *электронных книг Mathcad*. Для их вызова удобно использовать панель инструментов **Ресурсы** (Resources) (рис. 1.3).

В заключение отметим, что как справочная система, так и Ресурсы Mathcad представляют собой не просто статьи и примеры с описанием его возможностей. Они могут быть названы полноправными учебными пособиями по нескольким курсам высшей математики (в случае Ресурсов, к тому же, еще и интерактивными). Там освещены и основные определения, и математический смысл многих операций, и алгоритмы численных методов. Причем, на взгляд автора, некоторые из тем объяснены лучше, чем где бы то ни было. Если вы в достаточной степени владеете английским, обязательно ознакомьтесь с Ресурсами Mathcad.

1.2. Основы вычислений в Mathcad

Продемонстрируем, как можно быстро начать работу с Mathcad, научиться вводить математические выражения и получать результаты расчетов.

Внимание!

Бóльшая часть содержания книги с одинаковым успехом применима к семи последним версиям Mathcad: 2001, 2001i, 11, 12, 13, 14 и 15. Если определенные опции касаются некоторых поздних версий, на это делается соответствующее указание.

1.2.1. Операторы численного и символьного вывода

Для того чтобы выполнить простые расчеты по формулам, проделайте следующее:

1. Определите место в документе, где должно появиться выражение, щелкнув мышью в соответствующей точке документа.
2. Введите левую часть выражения.
3. Введите знак численного равенства = (клавишей <=>) или символьного равенства → (сочетанием клавиш <Ctrl>+<.>). В первом случае будет рассчитано численное значение выражения, а во втором (если это возможно) — аналитическое.

Оставим пока разговор о более надежных способах ввода математических символов и приведем пример простейших расчетов. Для вычисления арккосинуса какого-нибудь числа, например `0`, достаточно ввести с клавиатуры выражение `acos(0)=` или `acos(0)`→. После того как будет нажата клавиша со знаком равенства (или введен знак символьных вычислений →), с правой стороны выражения, как по мановению волшебной палочки, появится результат (листинги 1.1 и 1.2 соответственно).

Листинг 1.1. Численный расчет простого выражения

```
acos(0) = 1.571
```

Листинг 1.2. Аналитический расчет простого выражения

$$\mathrm{acos}(0) \rightarrow \frac{1}{2}\cdot\pi$$

Примечание

Здесь и далее во всей книге в листинги вынесено содержание рабочей области документа Mathcad вместе с полученными результатами вычислений. Почти все листинги выглядят в окне Mathcad рассматриваемых версий совершенно одинаково. Исключение составляют листинги, содержащие новые возможности той или иной версии Mathcad (в этих случаях они снабжены специальной ремаркой). Все листинги (а также рисунки) книги помещены также и на прилагаемом к ней компакт-диске, причем, по возможности, в формате Mathcad 2001, что позволяет просматривать их с одинаковым успехом при помощи любой из версий программы (2001—15), которая установлена на вашем компьютере.

Важно заметить, что по умолчанию вычисления в документе производятся в режиме реального времени, т. е. как только пользователь вводит в формулу оператор численного или символьного равенства, Mathcad пытается вычислить это выражение (и все остальные формулы, находящиеся ниже по тексту). Иногда, в основном в случае сложных и долгих расчетов, бывает полезно остановить их нажатием клавиши <Esc>, а затем (в нужный момент) возобновить нажатием клавиши <F9> или командой **Инструменты / Вычислить / Вычислить сейчас** (Tools / Calculate / Calculate Now) (*см. разд. 1.6.4*).

Примечание

В очень старых версиях Mathcad (2001 и ниже) вместо меню **Инструменты** (Tools) использовался заголовок **Математические** (Math).

1.2.2. Математические выражения и встроенные функции

Описанным в предыдущем разделе образом можно проводить более сложные и громоздкие вычисления, пользуясь при этом всем арсеналом функций, которые заложены разработчиками в систему Mathcad и называются поэтому *встроенными функциями* (в отличие от *пользовательских* функций, конструируемых непосредственно при разработке Mathcad-программы). Легче всего вводить имена встроенных функций с клавиатуры, как в примере с вычислением арккосинуса, но, чтобы избежать возможных ошибок в их написании, лучше выбрать другой путь (тем более что многие из них весьма сложны и имеют несколько аргументов, так что сложно запомнить имена и параметры всех функций наизусть).

Чтобы ввести встроенную функцию в выражение:

1. Определите место в выражении, куда следует вставить функцию.
2. Нажмите кнопку с надписью ***f(x)*** на стандартной панели инструментов.

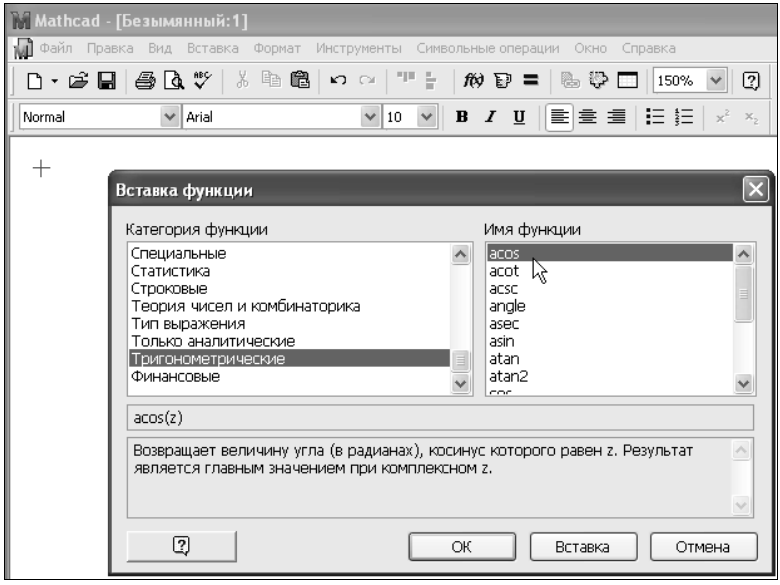


Рис. 1.4. Вставка встроенной функции (см. листинги 1.1 и 1.2)

3. В списке **Категория функции** (Function Category) появившегося диалогового окна **Вставка функции** (Insert Function) (рис. 1.4) выберите категорию, к которой принадлежит функция, — в нашем случае это категория **Тригонометрические** (Trigonometric).
4. В списке **Имя функции** (Function Name) выберите имя встроенной функции, под которым она фигурирует в Mathcad: в нашем примере — арккосинуса (**acos**). В случае затруднения с выбором ориентируйтесь на подсказку, появляющуюся при выборе функции в нижнем текстовом поле диалогового окна **Вставка функции** (Insert Function).
5. Нажмите кнопку **OK** — функция появится в документе.
6. Введите недостающие аргументы введенной функции (в нашем случае это число 0) в *местозаполнителе*, обозначаемом черным прямоугольником).

Результатом будет введение выражения из листинга 1.1, для получения значения которого осталось лишь ввести знак (численного или символьного) вывода (оба примера приведены на рис. 1.5).

> ***Примечание***
>
> Большинство численных методов, запрограммированных в Mathcad, реализовано в виде встроенных функций. Пролистайте на досуге списки в диалоговом окне **Вставка функции** (Insert Function), чтобы представлять себе, какие специальные функции и численные методы можно использовать в расчетах.

Не всякий символ можно ввести с клавиатуры. Например, неочевидно, как вставить в документ знак интеграла или дифференцирования. Для этого в Mathcad имеются специальные панели инструментов, очень похожие на средства формульного редактора Microsoft Word. Как мы уже говорили ранее, одна из них — панель инструмен-

тов **Математические** (Math), показанная на рис. 1.1. Она содержит инструменты для вставки в документы типично математических объектов (операторов, графиков, элементов программ и т. п.). Эта панель показана более крупным планом на рис. 1.5 уже на фоне редактируемого документа.

Рис. 1.5. Панель инструментов **Математические** (Math) служит для вызова на экран остальных наборных панелей

Панель содержит девять кнопок, нажатие каждой из которых приводит, в свою очередь, к появлению на экране еще одной панели инструментов. С помощью этих девяти дополнительных панелей можно вставлять в документы Mathcad разнообразные объекты. На рис. 1.5, как легко увидеть, на панели **Математические** (Math) в нажатом состоянии находится только одна кнопка (левая, на которую наведен указатель мыши). Поэтому на экране присутствует только одна математическая панель — **Калькулятор** (Calculator). Легко догадаться, какие объекты вставляются при нажатии кнопок на этой панели.

Примечание

Подробнее о назначении этих и других наборных панелей инструментов рассказано далее (*см. разд. 1.3 и 1.4*).

Большинство математических выражений можно ввести исключительно с помощью панели **Калькулятор** (Calculator), не пользуясь клавиатурой. Например, для расчета выражения `sin(1/2)` нужно сначала нажать кнопку **sin** (самую первую сверху), затем набрать выражение `1/2` в появившемся местозаполнителе внутри скобок. Для этого нажмите последовательно кнопки **1**, **/** и **2** на панели **Калькулятор** (Calculator) и затем, на ней же, кнопку =, чтобы получить ответ.

Как видите, вставлять в документы математические символы можно по-разному, как и во многих других приложениях Windows. В зависимости от опыта работы с Mathcad и привычек работы на компьютере пользователь может выбрать любой из них.

Совет

Если вы только начинаете осваивать редактор Mathcad, настоятельно рекомендую, где это только возможно, вводить формулы, пользуясь наборными панелями инструментов и описанной процедурой вставки функций с помощью диалогового окна **Вставка функции** (Insert Function). Это позволит избежать многих возможных ошибок.

1.2.3. Переменные и оператор присваивания

Описанные пока действия демонстрируют использование Mathcad в качестве обычного калькулятора с расширенным набором функций. Для математика же интерес представляет, как минимум, возможность задания переменных и операций с функциями пользователя. Для того чтобы присвоить некоторой переменной (например, переменной `x`) определенное значение, необходимо ввести выражение типа `x:=1`. Этот пример приведен в первой строке листинга 1.3, а в его второй строке осуществляется вычисление значения переменной `x` при помощи оператора численного вывода (знака равенства).

Как вы видите, *присваивание* (по-другому, *определение*) обозначается не знаком равенства, а специальным символом, чтобы подчеркнуть его отличие от операции численного вывода. Оператор присваивания вводится нажатием клавиши-двоеточия <:> либо при помощи панели **Калькулятор** (Calculator). Символ равенства "=" говорит о вычислении значения слева направо, а символ "**:=**" — о присваивании значения справа налево.

Примечание 1

Тем не менее, пользователю позволено изменить внешнюю форму оператора на более привычный для математика символ обычного равенства (что категорически не рекомендуется делать, поскольку сильно ухудшает восприятие Mathcad-программы). Для этого (рис. 1.6) следует вызвать нажатием правой кнопки мыши из области оператора присваивания контекстное меню и выбрать в нем **Отображать определение как / Знак равенства** (View Definition As / Equal). Кстати, подобным образом можно выбирать написание и некоторых других операторов, допускающих обозначение разными символами (например, оператора умножения).

Примечание 2

Если попытаться ввести знак численного вывода (обычного равенства) для переменной, впервые встречающейся в документе, он будет автоматически заменен символом присваивания.

Листинг 1.3. Присваивание значения переменной и его использование в расчетах

```
x := 1
x = 1
(x + 5)^2 = 36
```

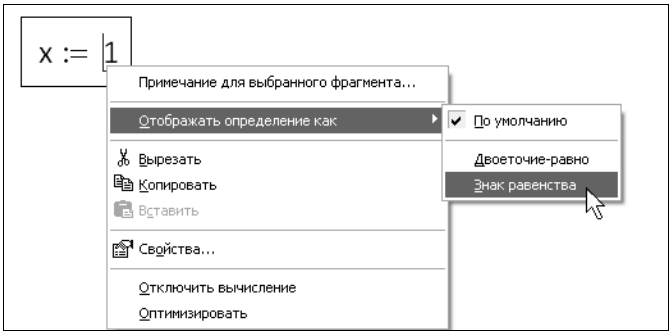


Рис. 1.6. Выбор вида оператора присваивания (см. листинг 1.3)

Для того чтобы вычислить значение выражения, содержащего некоторую переменную, следует просто ввести его, а затем применить оператор численного вывода (листинг 1.3, последняя строка). При этом необходимо, чтобы этой переменной ранее в документе было присвоено какое-либо значение.

> ***Примечание***
>
> Помимо оператора обычного присваивания, в Mathcad имеется еще один оператор *глобального присваивания* (≡). Если вставить его для задания значения переменной в любой части документа (например, в самом низу), то эта переменная автоматически будет определена в любой части документа.

В отличие от численных, для символьных вычислений задание значений всех переменных необязательно (листинг 1.4). Если некоторым переменным присвоены значения (как переменной `a` в листинге 1.4), то для получения результата используется это числовое значение. Если же переменной не было присвоено никакого значения (как переменной `k`), она воспринимается аналитически, просто как некоторое имя.

Символьные вычисления, позволяющие решить многие задачи аналитически, являются одной из самых впечатляющих возможностей Mathcad. Фактически Mathcad "знает" математику, по крайней мере, на уровне неплохого ученого. Умелое использование интеллекта символьного процессора Mathcad избавит вас от огромного количества рутинных вычислений, например, интегралов и производных. Обратите внимание на традиционную форму написания выражений (листинг 1.4), единственная особенность заключается в необходимости применения знака символьных вычислений → вместо знака равенства. Его, кстати, можно ввести в редакторе Mathcad с любой из панелей **Вычисление** (Evaluation) или **Символьные** (Symbolic), а символы интегрирования и дифференцирования — с панели **Математический анализ** (Calculus).

Листинг 1.4. Переменные в аналитических расчетах

$a := 3$

$$\frac{d}{dx}\sin\left(\frac{k\cdot x}{a^2}\right) \rightarrow \frac{1}{9}\cdot\cos\left(\frac{1}{9}\cdot k\cdot x\right)\cdot k$$

14 1.2.4. Одновременное присваивание и вычисление результата

Начиная с Mathcad 14, наконец, появилась функциональность, связанная с возможностью одновременного определения переменной или функции и численного вывода значения (листинг 1.5).

Листинг 1.5. Присваивание и вычисление результата в одной строке

$$x := \sin\left(\frac{\pi}{6}\right) + 2 = 2.5$$

$x = 2.5$

На самом деле, правильнее говорить не об одновременном присваивании и вычислении, а о присваивании переменной или функции значения некоторого выражения (того, что будет стоять после оператора численного вывода). Сравните новый тип присваивания с традиционным (листинг 1.6). В первом случае переменной `x` присваивается вычисленное значение $\sqrt{3} = 1.732$ (стоящее после знака равенства), о чем можно судить по второй строке листинга. Переменная `y`, напротив, определена непосредственно через $\sqrt{3}$, и ее значение, будучи выведено аналитически (последняя строка листинга), представляет собой именно $\sqrt{3}$, а не 1.732, как в первом случае.

Листинг 1.6. Присваивание переменной результата выражения и обычное присваивание

$x := \sqrt{3} = 1.732$

$x \rightarrow 1.7320508075688772$

$y := \sqrt{3}$

$y \rightarrow \sqrt{3}$

1.2.5. Функции пользователя

Подобно присваиванию числовых значений переменным, можно определить функции пользователя одного или нескольких аргументов (листинги 1.7 и 1.8). В листинге 1.7 определяется функция `f(x)`, а в листинге 1.8 — функция трех переменных `g(a,y,ϕ)`.

Листинг 1.7. Определение функции пользователя и расчет ее значений в точке

$f(x) := x^2 - 3 \cdot x - 2$

$f(0) = -2$

$f(10) = 68$

Листинг 1.8. Функция пользователя трех аргументов и ее вычисление в точке

$g(a, y, \phi) := a \cdot \sin(y + \phi)$

$g(1, 0, \pi) = 0$

График функции `f(x)` показан на рис. 1.7. Чтобы построить его, следует нажать на панели **График** (Graph) кнопку с нужным типом графика (на нее на рисунке наведен указатель мыши) и в появившейся заготовке графика определить значения, которые будут отложены по осям. В нашем случае потребовалось ввести `x` в местозаполнитель возле оси `X` и `f(x)` — возле оси `Y`.

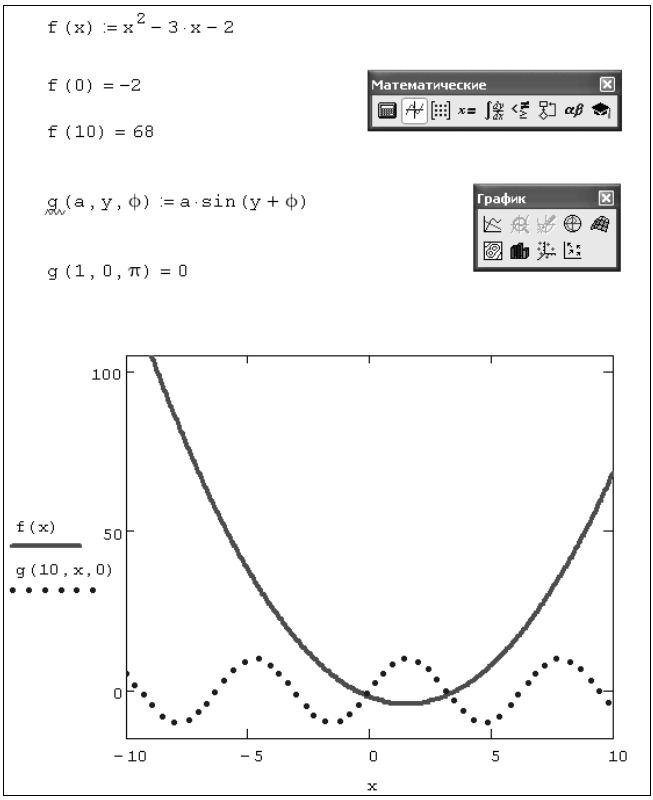


Рис. 1.7. Построение графика функции (продолжение листинга 1.7)

Примечание 1

Сравните содержание листинга 1.7 и рис. 1.7. Такой стиль подачи материала будет сохранен во всей книге. Листинги представляют собой фрагменты рабочих областей документа, которые работают без какого-либо дополнительного кода (если это не оговорено особо). Можно ввести содержание любого листинга в новый (пустой) документ, и он будет работать точно так же, как в книге. Чтобы не загромождать листинги, графики выведены в отдельные рисунки. В отличие от рис. 1.7, в следующих рисунках код листингов не дублируется, а если имеется ссылка на листинг в подрисуночной надписи, то это подразумевает, что данный график может быть вставлен в документ после упомянутого листинга.

Примечание 2

На том же графике на рис. 1.7 изображена и вторая кривая, представляющая собой двумерный график функции `g(10,x,0)`. Для того чтобы нарисовать и этот график, потребовалось ввести имя функции `g(10,x,0)` через запятую после `f(x)` возле оси `Y`.

1.2.6. Основные типы данных

Перечислим основные типы переменных, которые используются в Mathcad.

Действительные числа

Любое выражение, начинающееся с цифры, Mathcad интерпретирует как число. Поэтому для ввода числа просто начните его набирать на клавиатуре (листинг 1.9).

Примечание

Можно организовать ввод числа в других системах счисления: *двоичной* (binary), *восьмеричной* (octal) или *шестнадцатеричной* (hexadecimal) (листинг 1.10).

Листинг 1.9. Ввод действительных чисел

$a := 1000$ $b := 1.3474$

$c := 3124.1$ $d := 45.21 \cdot 10^{-5}$

Листинг 1.10. Ввод чисел в других системах исчисления

$a := 100010b$ $a = 34$

$b := 13o$ $c := 0f3h$

Комплексные числа

Большинство операций в среде Mathcad по умолчанию осуществляются над *комплексными числами*. Комплексное число является суммой действительного и *мнимого числа*, получающегося путем умножения любого действительного числа на *мнимую единицу* (imaginary unit) i. По определению полагается, что $i = \sqrt{-1}$ или $i^2=-1$.

Чтобы ввести мнимое число, например $3i$:

1. Введите действительный сомножитель (3).
2. Введите символ "i" или "j" непосредственно после него.

Примечание

Для ввода мнимой единицы надо нажать клавиши <1>, <i.>. Если просто ввести символ "i", то Mathcad интерпретирует его как переменную i. Кроме того, мнимая единица имеет вид $1i$, только когда соответствующая формула выделена. В противном случае мнимая единица отображается просто как i (рис. 1.8).

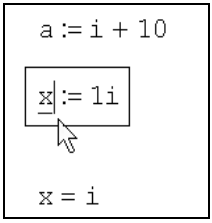


Рис. 1.8. Ввод мнимой единицы

Комплексное число можно ввести в виде обычной суммы действительной и мнимой частей или в виде любого выражения, содержащего мнимое число. Примеры ввода и вывода комплексных чисел иллюстрируются листингом 1.11. Для работы с комплексными числами имеется несколько простых функций и операторов, действие которых показано в листинге 1.12.

Примечание

Можно выводить мнимую единицу в результатах вычислений не как `i`, а как `j`. Для смены представления выберите нужное в списке **Мнимое значение** (Imaginary Value) диалогового окна **Формат результата** (Result Format), доступного по команде **Формат / Результат / Параметры отображения** (Format / Result / Display Options).

Листинг 1.11. Ввод и вывод комплексных чисел

$a := 2i + 0.5$ $\quad a = 0.5 + 2i$

$b := 1.77 \cdot e^{2i}$ $\quad b = -0.737 + 1.609i$

$c := 25j + 12$ $\quad c = 12 + 25i$

Листинг 1.12. Примеры простых вычислений с комплексными числами (продолжение листинга 1.11)

$Im(a) = 2$ $\quad Re(a) = 0.5$

$|a| = 2.062$ $\quad arg(a) = 1.326$

$arg(b) = 2$ $\quad |b| = 1.77$

Встроенные константы

Некоторые имена в Mathcad зарезервированы под системные переменные, которые называются *встроенными константами* (built-in constants). Встроенные константы делятся на два типа: *математические* (math constants), хранящие значения некоторых общеупотребительных специальных математических символов, и *системные* (system variables), определяющие работу большинства численных алгоритмов, реализованных в Mathcad.

Примечание

При желании можно изменить значение любой из перечисленных констант или использовать их в качестве переменных в расчетах. Разумеется, если присвоить константе новое значение, прежнее станет недоступным.

Математические константы по-разному интерпретируются при численных и символьных вычислениях. Вычислительный процессор просто воспринимает их как некоторые числа (листинг 1.13), а символьный распознает каждое из них, исходя из математического контекста, и способен выдавать математические константы в качестве результата. Перечислим математические константы:

- ∞ — символ бесконечности (вводится клавишами <Ctrl>+<Shift>+<z>);
- `e` — основание натурального логарифма (клавиша <e>);
- π — число "пи" (вводится клавишами <Ctrl>+<Shift>+<p>);
- `i, j` — мнимая единица (вводится клавишами <1>, <i> или <1>, <j>);
- `%` — символ процента, <%>, эквивалентный `0.01`.

Листинг 1.13. Значения математических констант

$\infty = 1 \times 10^{307}$

$e = 2.718$

$\pi = 3.142$

$i = i$

$j = i$

$\% = 0.01 \qquad 100 \cdot 25 \cdot \% = 25$

Системные переменные определяют работу численных методов, заложенных во встроенные функции. Их предустановленные значения перечислены в листинге 1.14 (в принципе, допускается их менять в любой части документа). Системные переменные:

- TOL — допуск сходимости (точность численных методов);
- CTOL — допуск ограничения (точность выполнения выражений, используемая в некоторых численных методах);
- ORIGIN — номер начального индекса в массивах и строковых переменных;
- PRNPRECISION — установка формата данных при выводе в файл;
- PRNCOLWIDTH — установка формата столбца при выводе в файл;
- CWD — строковое представление пути к текущей рабочей папке.

Листинг 1.14. Предустановленные значения системных переменных

$TOL = 1 \times 10^{-3} \qquad CTOL = 1 \times 10^{-3}$

$ORIGIN = 0$

Строковые переменные

Значением переменной или функции может быть не только число, но и строка, состоящая из любой последовательности символов, заключенной в кавычки (листинг 1.15). Для работы со строками в Mathcad имеется несколько встроенных функций.

> **12 *Примечание***
>
> Системная константа ORIGIN может теперь устанавливать не только номер начального индекса массивов, но и начало отсчета для соответствующих встроенных функций строкового (текстового) аргумента. Если вы хотите включить эту опцию, установите флажок проверки **Использовать ORIGIN для индексации строк** (Use ORIGIN for string indexing) на вкладке **Вычисления** (Calculations) диалогового окна **Параметры документа** (Worksheet options).

Листинг 1.15. Ввод и вывод строк

$s := \text{"Hello,"} \qquad s = \text{"Hello,"}$

$concat(s, \text{" world!"}) = \text{"Hello, world!"}$

1.2.7. Ранжированные переменные и матрицы

Массивами (arrays) называют упорядоченные последовательности чисел, или *элементов массива*. Доступ к любому элементу массива возможен по его *индексу*, т. е. номеру в последовательности чисел (в листинге 1.16: a — это массив, a_1 — его элемент). Применение массивов чрезвычайно эффективно в математических расчетах.

Листинг 1.16. Одномерный массив (вектор)

$$a := \begin{pmatrix} 14 \\ 1.4 \\ 4.7 \end{pmatrix}$$

$$a_0 = 14$$

$$a_1 = 1.4$$

$$a_2 = 4.7$$

В Mathcad условно выделяются два типа массивов:

- *векторы* (одноиндексные массивы, листинг 1.16), *матрицы* (двухиндексные, листинг 1.17) и *тензоры* (многоиндексные);
- *ранжированные переменные* (range variable, в русской версии они названы термином *"переменная — диапазон"*) — векторы, элементы которых определенным образом зависят от их индекса.

Листинг 1.17. Двумерный массив (матрица)

$$A := \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 4 & 5 & 6 \\ 7 & 8 & -9 \end{pmatrix}$$

$$A_{0,0} = 0.1$$

$$A_{2,0} = 7$$

Доступ ко всему массиву осуществляется обычным определением векторной переменной. Над элементами массива можно совершать действия как над обычными числами. Нужно только правильно задать соответствующий индекс или сочетание индексов массива. Например, чтобы получить доступ к нулевому элементу вектора a из листинга 1.16:

1. Введите имя переменной массива (a).
2. Нажмите кнопку **Индекс** (Subscript) со значком $\mathbf{x_n}$ на панели **Матрица** (Matrix) либо введите [.
3. В появившийся справа снизу от имени массива местозаполнитель введите желаемый индекс (0).

Если после этого ввести знак численного вывода, то справа от него появится значение нулевого элемента вектора, как показано во второй строке листинга 1.16.

Чтобы получить доступ к элементу многоиндексного массива (например, элементу $a_{1,0}$ матрицы `a` из листинга 1.17):

1. Введите имя переменной массива (`a`).
2. Перейдите к вводу нижнего индекса, введя `[`.
3. Введите в местозаполнитель индекса первый индекс (`2`), запятую (`,`) и в появившийся после запятой местозаполнитель введите второй индекс (`0`).

В результате будет получен доступ к элементу, как показано в последней строке листинга 1.17.

Примечание 1

В рассмотренных листингах нумерация индексов массивов начинается с нуля, иными словами, первый элемент массива имеет индекс `0`. Стартовый индекс массива задается системной переменной `ORIGIN`, которая по умолчанию равна нулю. Если вы привыкли нумеровать элементы векторов и матриц с единицы, присвойте этой переменной значение `1`. Обратите внимание, что в этом случае попытка выяснить значение нулевого элемента вектора приводит к ошибке, поскольку его значение не определено.

Примечание 2

Помимо доступа к отдельным элементам массива, имеется возможность совершать действия над его подмассивами (например, векторами-столбцами, образующими матрицу). Делается это с помощью оператора со значком **x<>** на панели **Матрица** (Matrix) (листинг 1.18). Символ "T" во второй строке листинга 1.18 обозначает операцию транспонирования матрицы.

Листинг 1.18. Доступ к подмассиву (продолжение листинга 1.15)

$$A^{\langle 0 \rangle} = \begin{pmatrix} 0.1 \\ 4 \\ 7 \end{pmatrix}$$

$$\left| A^{T} \right|^{\langle 2 \rangle^{T}} = \begin{pmatrix} 7 & 8 & -9 \end{pmatrix}$$

1.2.8. Размерные переменные

В Mathcad числовые переменные и функции могут обладать *размерностью*. Сделано это для упрощения инженерных и физических расчетов. В Mathcad встроено большое количество *единиц измерения*, с помощью которых и создаются размерные переменные.

Чтобы создать размерную переменную, определяющую, например, силу тока в `10 A`, введите выражение, присваивающее переменной `I` значение `10`: `I:=10`, затем символ умножения <*>, а потом букву "A". Поскольку все символы, обозначающие единицы измерения, зарезервированы и имеют предустановленные значения (свя-

занные с размерностью), то литера A будет распознана Mathcad как ампер (листинг 1.19, первая строка). Если ранее вы переопределили переменную A, присвоив ей какое-либо значение, то восприниматься как единица силы тока она уже не будет.

Листинг 1.19. Расчеты с размерными переменными

$$I := 10 \cdot A$$

$$U := 110 \cdot V$$

$$R := \frac{U}{I}$$

$$R = 11\,\Omega$$

Вставить единицу измерения можно и по-другому, не вручную, а при помощи средств Mathcad. Для этого выберите команду **Вставка / Единица измерения** (Insert / Unit) и нажмите либо кнопку с изображением мерного стакана на стандартной панели инструментов, либо клавиши <Ctrl>+<U>. Затем в списке **Единица измерения** (Unit) открывшегося диалогового окна **Вставка единицы измерений** (Insert Unit) выберите нужную единицу измерения **Ampere (A)** и нажмите кнопку **OK**. Если вы затрудняетесь с выбором конкретной единицы измерения, но знаете, какова размерность переменной (в нашем случае это электрический ток), то попробуйте выбрать ее в списке **Dimension** (Размерность) диалогового окна **Вставка единицы измерения** (Insert Unit). Тогда в списке **Единица измерения** (Unit) появятся допустимые для этой величины единицы измерений, из которых выбрать нужную будет легче (рис. 1.9).

Просмотреть вставку единиц измерения можно и без выхода из диалогового окна **Вставка единицы измерения** (Insert Unit), нажимая вместо кнопки **OK**

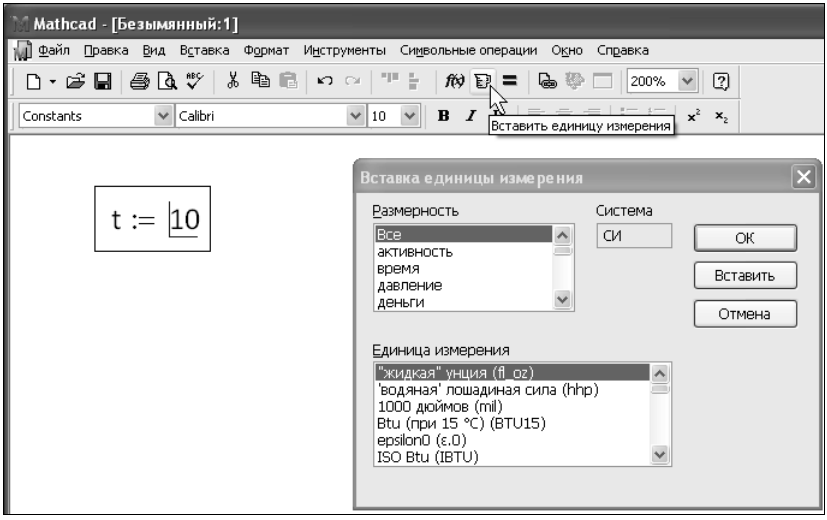


Рис. 1.9. Вставка единиц измерения размерной величины

кнопку **Вставить** (Insert). В этом случае вы увидите, что единица измерений появилась в нужном месте документа.

Примечание

Многие единицы измерения можно представлять в виде различных символов. Например, ампер — как `A` или `amp`, ом — как `ohm` или Ω и т. д.

Над размерными переменными можно производить любые разумные с физической точки зрения расчеты. Пример расчета сопротивления через отношение напряжения к току приведен в листинге 1.19. Работая с размерными переменными, приготовьтесь к тому, что Mathcad будет постоянно контролировать корректность расчетов. Например, нельзя складывать переменные разной размерности, в противном случае будет получено сообщение об ошибке. Тем не менее позволяется складывать, например, амперы с килоамперами и величины, размерность которых выражена в разных системах измерения (например, СИ и СГС).

Примечание 1

Можно включить автоматический перевод единиц измерения в более простые единицы, как это показано в листинге 1.19 (ответ автоматически переводится в омы). Для этого перейдите в диалоговое окно **Формат результата** (Result Format) на вкладку, посвященную размерностям, с помощью команды **Формат / Результат / Отображение единиц измерения** (Format / Result / Unit Display). Установите в ней флажок **Упрощать единицы измерения, когда это возможно** (Simplify units when possible).

⑮ ***Примечание 2***

Mathcad распознает тысячи различных единиц измерения и способен осуществлять, где это необходимо, их автоматическую конвертацию.

⑫ Единицу измерения в системе СИ любой размерной переменной можно вывести при помощи встроенной функции `SIUnitsOf` (листинг 1.20):

❒ `SIUnitsOf (a)` — возвращает единицу измерения переменной (в системе СИ):

- `A` — переменная.

Внимание!

В прежних версиях Mathcad эта функция имела другое название — `UnitsOf`.

Листинг 1.20. Вывод единицы измерения размерной величины в системе СИ

$$v := 40 \cdot \frac{km}{hr}$$

$$v = 11.111 \frac{m}{s}$$

$$SIUnitsOf(v) = 1 \frac{m}{s}$$

⑬ В Mathcad 13 появились новые единицы измерения температуры (а именно градусы Цельсия и Фаренгейта, а также их относительные значения). Осуществлять

ввод единиц измерения температуры следует при помощи наборной панели **Дополнительные символы** (Custom Characters) либо (еще лучше) посредством диалогового окна **Вставка единицы измерения** (Insert Unit). Применение температурных единиц измерения иллюстрируется листингом 1.21.

> ***Внимание!***
>
> Имейте в виду, что температура может измеряться как в абсолютных (Кельвинах), так и в относительных (градусах Фаренгейта и Цельсия) единицах. Чтобы избежать путаницы, связанной со сложением и умножением относительных температурных единиц, предусмотрены дополнительные размерности — изменение температур, обозначаемое символом Δ (листинг 1.21).

Листинг 1.21. Пример работы с температурными единицами

```
t := 10 °C
t = 283.15 K
t − 10 °C = 0 K
20 °F − 10 °F = 10 Δ°F
20 °C + 10 °C = 576.3 K
20 °C + 10·Δ°C = 303.15 K
```

1.3. Ввод и редактирование формул

Формульный редактор Mathcad позволяет быстро и эффективно вводить и изменять математические выражения. Тем не менее некоторые аспекты его применения не совсем интуитивны, что связано с необходимостью избежания ошибок при расчетах по этим формулам. Поэтому не пожалейте немного времени на знакомство с особенностями формульного редактора, впоследствии при реальной работе вы сэкономите гораздо больше.

1.3.1. Элементы интерфейса редактора формул

Перечислим элементы интерфейса редактора Mathcad:

- указатель мыши (mouse pointer) — играет обычную для приложений Windows роль, следуя за движениями мыши;
- курсор — обязательно находится внутри документа в одном из трех видов:
  - курсор ввода (crosshair) — крестик красного цвета, который отмечает пустое место в документе, куда можно вводить текст или формулу;
  - линии ввода (editing lines) — горизонтальная (underline) и вертикальная (insertion line) линии синего цвета, выделяющие в тексте или формуле определенную часть;
  - линия ввода текста (text insertion point) — красная вертикальная линия, аналог линий ввода для текстовых областей;

- местозаполнители (placeholders) — появляются внутри незавершенных формул в местах, которые должны быть заполнены символом или оператором:
  - местозаполнитель символа — черный прямоугольник;
  - местозаполнитель оператора — черная прямоугольная рамка.

Курсоры и местозаполнители, относящиеся к редактированию формул, представлены на рис. 1.10.

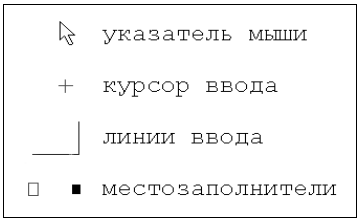


Рис. 1.10. Интерфейс редактирования

1.3.2. Ввод формул

Бóльшую часть окна Mathcad занимает *рабочая область* документа Mathcad, в которую пользователь вводит математические выражения, текстовые поля и элементы программирования. Ввести математическое выражение можно в любом пустом месте документа Mathcad. Для этого поместите курсор ввода в желаемое место документа, щелкнув в нем мышью, и просто начинайте вводить формулу, нажимая клавиши на клавиатуре. При этом в документе создается *математическая область* (math region), которая предназначена для хранения формул, интерпретируемых процессором Mathcad. Продемонстрируем последовательность действий на примере ввода выражения x^{5+x} (рис. 1.11):

1. Щелкните мышью, обозначив место ввода.
2. Нажмите клавишу <x> — в этом месте вместо курсора ввода появится область с формулой, содержащей один символ x, причем он будет выделен линиями ввода.
3. Введите оператор возведения в степень, нажав клавишу <^> либо выбрав кнопку возведения в степень на панели инструментов **Калькулятор** (Calculator) — в формуле появится местозаполнитель для введения значения степени, а линии ввода выделят этот местозаполнитель.
4. Последовательно введите остальные символы <5>, <+>, <x>.

Таким образом, поместить формулу в документ можно, просто начиная вводить символы, числа или операторы, например + или /. Во всех этих случаях на месте курсора ввода создается математическая область, иначе называемая *регионом*, с формулой и линиями ввода. В последнем случае, если пользователь начинает ввод формулы с оператора, в зависимости от его типа автоматически появляются и местозаполнители, без заполнения которых формула не будет восприниматься процессором Mathcad (рис. 1.12).

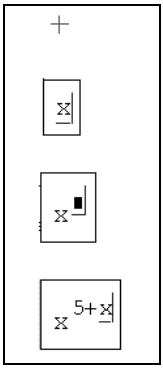


Рис. 1.11. Пример ввода формулы (коллаж)

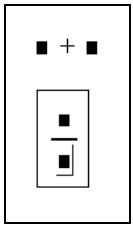


Рис. 1.12. Пример начала ввода операторов

1.3.3. Перемещение линий ввода внутри формул

Чтобы изменить формулу, щелкните на ней мышью, поместив таким образом в ее область линии ввода, и перейдите к месту, которое хотите исправить. Перемещайте линии ввода в пределах формулы одним из двух способов:

- щелкая в нужном месте мышью;
- нажимая на клавиатуре клавиши: стрелки, пробел и <Ins>:
  - клавиши со стрелками имеют естественное назначение, переводя линии ввода вверх, вниз, влево или вправо;
  - клавиша <Ins> переводит вертикальную линию ввода с одного конца горизонтальной линии ввода на противоположный;
  - пробел предназначен для выделения различных частей формулы.

Если раз за разом нажимать клавишу пробела в формуле, пример которой рассмотрен выше (см. рис. 1.11), то линии ввода будут циклически изменять свое положение, как это показано на рис. 1.13. Если в ситуации, показанной сверху на этом рисунке, нажать стрелку <←>, то линии ввода переместятся влево (рис. 1.14). При нажатии пробела теперь линии ввода будут попеременно выделять одну из двух частей формулы.

> ***Совет***
>
> Привыкнув к использованию пробела для перемещения внутри формул, можно существенно облегчить себе работу с Mathcad.

Таким образом, комбинация клавиш со стрелками и пробела позволяет легко перемещаться внутри формул. Накопив некоторый опыт, вы без труда освоите эту технику. Иногда поместить линии ввода в нужное место формулы с помощью указателя мыши непросто. Поэтому в Mathcad для этого лучше использовать клавиатуру.

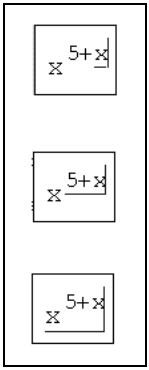


Рис. 1.13. Изменение положения линий ввода с помощью пробела (коллаж)

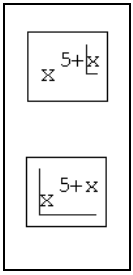


Рис. 1.14. Изменение положения линий ввода пробелом после сдвига стрелкой <←> (коллаж)

1.3.4. Изменение формул

Редактируйте формулы в Mathcad так, как подсказывают вам интуиция и опыт работы с другими текстовыми редакторами. Большинство операций правки формул реализованы естественным образом, однако некоторые из них несколько отличаются от общепринятых, что связано с особенностью Mathcad как вычислительной системы. Рассмотрим основные действия по изменению формул.

Вставка оператора

Операторы могут быть унарными (действующими на один операнд, как, например, оператор транспонирования матрицы или смены знака числа), так и бинарными (например + или /, действующими на два операнда). При вставке нового оператора в документ Mathcad определяет, сколько операндов ему требуется. Если в точке вставки оператора один или оба операнда отсутствуют, Mathcad автоматически помещает рядом с оператором один или два местозаполнителя.

> ***Внимание!***
>
> То выражение в формуле, которое выделено линиями ввода в момент вставки оператора, становится его первым операндом.

Последовательность вставки оператора в формулу такова:

1. Поместите линии ввода на часть формулы, которая должна стать первым операндом.
2. Введите оператор, нажав кнопку на панели инструментов или сочетание клавиш.

> ***Примечание***
>
> Для того чтобы вставить оператор не после, а перед частью формулы, выделенной линиями ввода, нажмите перед его вводом клавишу <Ins>, которая передвинет вертикальную линию ввода вперед. Это важно, в частности, для вставки оператора отрицания.

На рис. 1.15 показано несколько примеров вставки оператора сложения в разные части формулы, создание которой мы подробно разбирали ранее (см. рис. 1.11).

В левой колонке рис. 1.15 приведены возможные размещения линий ввода в формуле, а в правой — результат вставки оператора сложения (т. е. нажатия клавиши <+>). Как видно, Mathcad сам расставляет, если это необходимо, скобки, чтобы часть формулы, отмеченная линиями ввода, стала первым слагаемым.

Некоторые операторы Mathcad вставит в правильное место независимо от положения линий ввода. Таков, например, оператор численного вывода =, который по смыслу выдает значение всей формулы в виде числа (*см. разд. 1.2.1*).

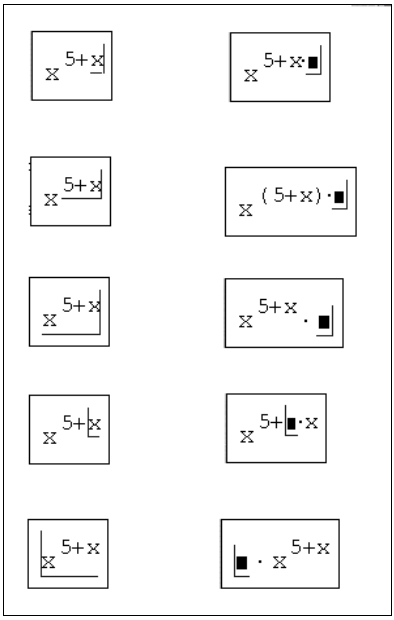


Рис. 1.15. Вставка оператора в разные части формулы (коллаж)

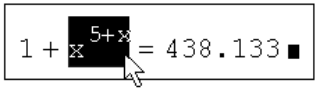


Рис. 1.16. Выделение части формулы

Выделение части формулы

Чтобы выделить часть формулы в некоторой математической области (рис. 1.16):

1. Поместите ее между линиями ввода, пользуясь, при необходимости, клавишами-стрелками и пробелом.
2. Поместите указатель мыши на вертикальную линию ввода, нажмите и удерживайте левую кнопку мыши.
3. Удерживая кнопку мыши, протащите указатель мыши вдоль горизонтальной линии ввода, при этом часть формулы будет выделяться обращением цвета.
4. Отпустите кнопку мыши, когда будет выделена нужная часть формулы.

Примечание

Часть формулы можно выделить и без помощи мыши, нажимая клавиши со стрелками при удерживаемой клавише <Shift>. В этом случае вместо перемещения линий ввода происходит выделение соответствующей части формулы. Многие пользователи находят работу с клавиатурой при выделении части математических областей более удобной.

Удаление части формулы

Чтобы удалить часть формулы:

1. Выделите ее.
2. Нажмите клавишу <Del>.
3. Кроме того, можно удалить часть формулы, помещая ее перед вертикальной линией ввода и нажимая клавишу <BackSpace>. В некоторых случаях, например при работе со сложными формулами, для достижения желаемого эффекта может потребоваться повторное нажатие <BackSpace>.

Примечание

Имеется еще один способ удаления части формулы: выделите ее нужную часть, затем нажмите комбинацию клавиш <Ctrl>+<X>, тем самым вырезая и помещая ее в буфер обмена. Этот способ удобен в случае, если требуется использовать фрагмент формулы в дальнейшем. Особенно удобно работать с буфером обмена при помощи контекстного меню (рис. 1.17)

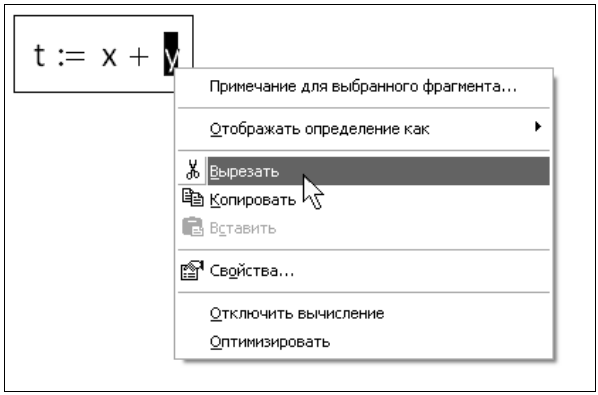


Рис. 1.17. Удаление части формулы в буфер обмена

О форматировании формул

Когда вы начинаете вводить формулу, формат шрифта и абзаца определяется в соответствии со *стилями*, выбранными по умолчанию и сохраненными в шаблоне документа.

Особенность форматирования формул заключается в том, что изменения шрифта при помощи панели инструментов **Форматирование** (Formatting), примененные к отдельному параметру в одной формуле, немедленно приводят к его изменению во всех формулах в документе (там, где этот параметр присутствует).

Изменить форматирование формулы (или абзаца текстового региона) целиком можно с помощью применения к нему стиля. Для этого выделите абзац или формулу, а затем выберите из списка стиль, который вы желаете применить к формуле или абзацу текста.

1.3.5. Программирование

Основными инструментами работы в Mathcad являются математические выражения, переменные и функции. Нередко записать формулу, использующую ту или иную внутреннюю логику (например, возвращение различных значений в зависимости от условий), в одну строку не удается. Назначение программных модулей как раз и заключается в определении выражений, переменных и функций в несколько строк, часто с применением специфических программных операторов.

Принцип программирования в Mathcad

При помощи элементов программирования можно определять переменные и функции (как показано в листинге 1.22).

Листинг 1.22. Функция условия, определенная с помощью программы

```
f(x) := | "negative"  if x < 0
        | "positive"  if x > 0
        | "zero"  otherwise
f(1) = "positive"
f(−1) = "negative"
f(0) = "zero"
```

Традиционное программирование, упрощенный вариант которого применен в Mathcad и осуществляется при помощи панели инструментов **Программирование** (Programming), имеет ряд существенных преимуществ, которые в ряде случаев делают документ более простым и читаемым:

- возможность применения циклов и условных операторов;
- простота создания функций и переменных, требующих несколько простых шагов;
- возможность создания функций, содержащих закрытый для остального документа код, включая преимущества использования локальных переменных и обработку исключительных ситуаций.

Как видно из листинга 1.22, программный модуль обозначается в Mathcad вертикальной чертой, справа от которой последовательно записываются операторы языка программирования. Чтобы начать создание программного модуля, следует (в случае листинга 1.22 после символа присваивания) нажать на панели **Программирование** (Programming) кнопку **Добавить строку программы** (Add Line). Затем, если приблизительно известно, сколько строк кода будет содержать программа, можно создать нужное количество линий повторными нажатиями кнопки **Добавить строку программы** (Add Line) (рис. 1.18).

В появившиеся местозаполнители введите желаемый программный код, используя программные операторы. В рассматриваемом примере в каждый местозаполнитель вводится строка, например, в средний — `"positive"` (рис. 1.19). Затем нажимается

кнопка **If** (Если) на панели **Программирование** (Programming) и в возникший местозаполнитель вводится выражение `x>0`. После того как программный модуль полностью определен, и ни один местозаполнитель не остался пустым, функция может использоваться обычным образом, как в численных, так и в символьных расчетах.

Внимание!

Не вводите с клавиатуры имена программных операторов. Для их вставки можно применять лишь сочетания клавиш, которые приведены в тексте всплывающей подсказки (рис. 1.18 и 1.19).

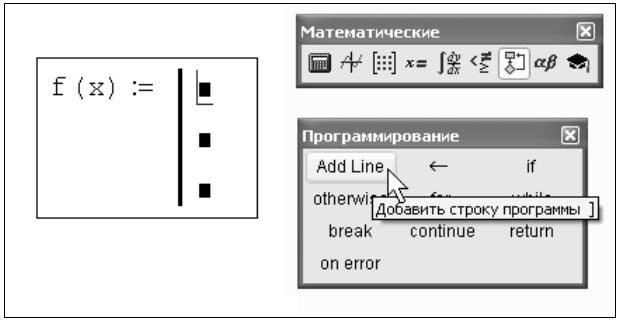


Рис. 1.18. Начало создания программного модуля

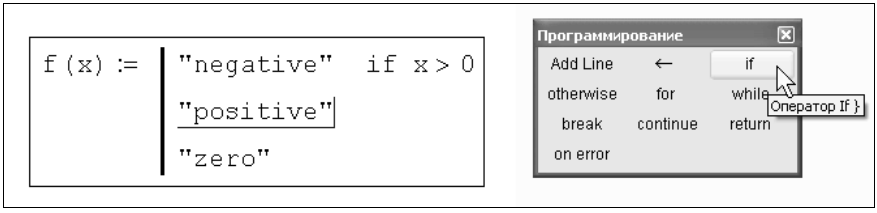


Рис. 1.19. Вставка программного оператора

Добавление строк программного кода

Вставить строку программного кода в уже созданную программу можно в любой момент с помощью той же самой кнопки **Добавить строку программы** (Add Line). Для этого следует предварительно поместить на нужное место внутри программного модуля линии ввода. Например, расположение линии ввода на строке, показанной на рис. 1.18, приведет к появлению новой линии с местозаполнителем перед этой строкой. Если передвинуть вертикальную линию ввода из начала строки (как это показано на рис. 1.20) в ее конец, то новая линия появится после строки. Если выделить строку не целиком, а лишь некоторую ее часть (рис. 1.20), то это повлияет на положение в программе новой строки кода. Результат нажатия кнопки **Добавить строку программы** (Add Line) показан на рис. 1.21.

Совет

Не забывайте, что для желаемого размещения линий ввода внутри формулы можно использовать не только мышь и клавиши со стрелками, но и пробел. С помощью последовательных нажатий пробела линии ввода "захватывают" разные части формулы.

```
f(x) := | "negative"  if x < 0
        | "positive"  if x > 0
        | "zero"  otherwise
```

Рис. 1.20. Положение линий ввода влияет на положение создаваемой строки программы

```
f(x) := | "negative"  if x < 0
        | if x > 0
        |   | "positive"
        |   |
        | "zero"  otherwise
```

Рис. 1.21. Результат вставки новой линии в программу (из положения рис. 1.20)

Зачем может потребоваться вставка новой линии в положение, показанное на рис. 1.21? Новая вертикальная черта с двумя линиями выделяет фрагмент программы, который относится к условию `x>0`, находящемуся в его заголовке. Пример возможного дальнейшего программирования показан в листинге 1.23.

Листинг 1.23. Пример усовершенствования программы

```
f(x) := | "negative"  if x < 0
        | if x > 0
        |   | "positive"
        |   | "big positive"  if x > 1000
        | "zero"  otherwise
f(1) = "positive"
f(10^5) = "big positive"
```

В режиме выполнения программы, а это происходит при любой попытке вычислить `f(x)`, выполняется последовательно каждая строка кода. Например, в предпоследней строке листинга 1.23 вычисляется `f(1)`. Рассмотрим работу каждой строки кода этого листинга.

1. Поскольку `x=1`, то условие `x<0` не выполнено, и в первой строке ничего не происходит.
2. Условие второй строки `x>0` выполнено, поэтому выполняются обе следующие строки, объединенные короткой вертикальной чертой в общий фрагмент.
3. Функции `f(x)` присваивается значение `f(x)="positive"`.
4. Условие `x>1000` не выполнено, поэтому значение `"big positive"` не присваивается `f(x)`, она так и остается равной строке `"positive"`.
5. Последняя строка не выполняется, т. к. одно из условий (`x>0`) оказалось истинным, и оператор `otherwise` (т. е., "иначе") не понадобился.

Таким образом, основной принцип создания программных модулей заключается в правильном расположении строк кода. Ориентироваться в их действии довольно легко, т. к. фрагменты кода одного уровня сгруппированы в программе с помощью вертикальных черт.

Локальное присваивание (←)

Язык программирования Mathcad не был бы эффективным, если бы не позволял создавать внутри программных модулей локальные переменные, которые "не видны" извне, из других частей документа. Присваивание в пределах программ, в отличие от документов Mathcad, производится с помощью оператора **Локальное определение** (Local Definition), который вставляется нажатием кнопки с изображением стрелки (←) на панели **Программирование** (Programming).

> ***Внимание!***
> Ни оператор присваивания `:=`, ни оператор численного вывода `=` в пределах программ применять не разрешается.

Локальное присваивание иллюстрируется листингом 1.24. Переменная `z` существует только внутри программы, выделенной вертикальной чертой. Из других мест документа получить ее значение невозможно. На этом же листинге вы видите пример использования оператора цикла `for`.

Листинг 1.24. Локальное присваивание в программе

```
x := | z ← 0
     | for i ∈ 0 .. 5
     |   z ← z + i
x = 15
```

1.4. Графики

Одним из наиболее впечатляющих достоинств Mathcad, несомненно, являются развитые возможности построения графиков.

1.4.1. Типы графиков

В Mathcad встроено несколько различных типов графиков, которые можно разбить на две большие группы.

- Двумерные графики:
  - X-Y (декартовый) график (X-Y Plot);
  - полярный график (Polar Plot).
- Трехмерные графики:
  - график трехмерной поверхности (Surface Plot);
  - график линий уровня (Contour Plot);
  - трехмерная гистограмма (3D Bar Plot);
  - трехмерное множество точек (3D Scatter Plot);
  - векторное поле (Vector Field Plot).

Деление графиков на типы несколько условно, так как, управляя установками многочисленных параметров, можно создавать комбинации типов графиков, а также новые типы (например, двумерная гистограмма распределения является разновидностью простого X-Y графика).

1.4.2. Создание графика

Все графики создаются совершенно одинаково, с помощью панели инструментов **График** (Graph), различия обусловлены отображаемыми данными.

Чтобы создать график, например, двумерный декартов:

1. Поместите курсор ввода в то место документа, куда требуется вставить график.
2. Если на экране нет панели **График** (Graph), вызовите ее нажатием кнопки с изображением графиков на панели **Math** (Математические).
3. Нажмите на панели **График** (Graph) кнопку **График X-Y** (X-Y Plot) для создания декартова графика (рис. 1.22) или другую кнопку для вставки графика иного типа.
4. В результате в обозначенном месте документа появится пустая область графика с одним или несколькими местозаполнителями (рис. 1.22, слева). Введите в местозаполнители имена переменных или функций, которые должны быть изображены на графике. В случае декартова графика это два местозаполнителя данных, откладываемых по осям X и Y.

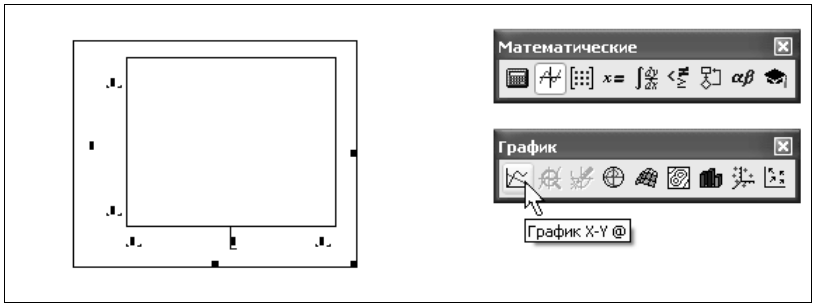


Рис. 1.22. Создание декартового графика при помощи панели **График** (Graph)

Если имена данных введены правильно, нужный график появится на экране. Созданный график можно изменить, в том числе меняя сами данные, форматируя его внешний вид или добавляя дополнительные элементы оформления.

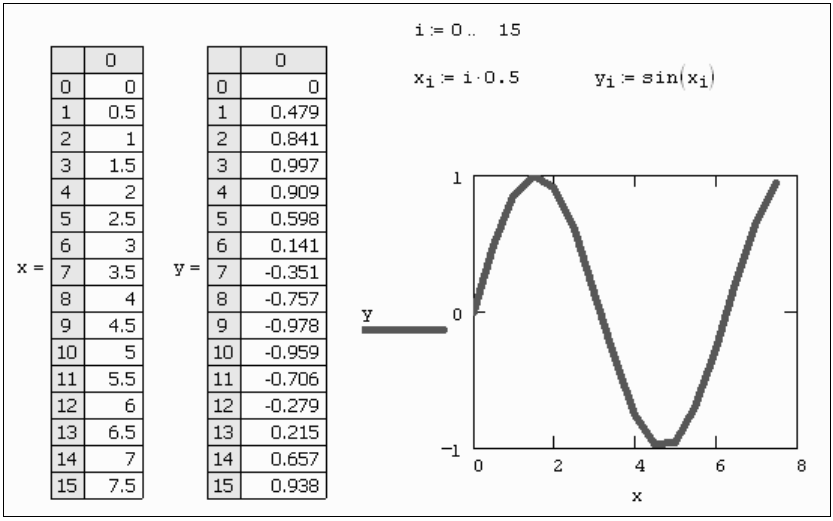
> ***Внимание!***
>
> Некорректное определение данных приводит вместо построения графика к выдаче сообщения об ошибке.

Возможности форматирования координатных осей графиков включают в себя управление их внешним видом, диапазоном, шкалой, нумерацией и отображением некоторых значений на осях при помощи маркеров. Когда график создается впервые, Mathcad выбирает представленный диапазон для обеих координатных осей автома-

тически. Чтобы изменить этот диапазон, выделите график и введите новое значение диапазона в каждом из четырех полей, появившихся вблизи осей.

1.4.3. График двух векторов

Самый простой и наглядный способ получить декартов график — это сформировать два вектора данных, которые будут отложены вдоль осей `X` и `Y`. Последовательность построения графика двух векторов `x` и `y` показана на рис. 1.23.



	0
0	0
1	0.5
2	1
3	1.5
4	2
5	2.5
6	3
7	3.5
8	4
9	4.5
10	5
11	5.5
12	6
13	6.5
14	7
15	7.5

	0
0	0
1	0.479
2	0.841
3	0.997
4	0.909
5	0.598
6	0.141
7	-0.351
8	-0.757
9	-0.978
10	-0.959
11	-0.706
12	-0.279
13	0.215
14	0.657
15	0.938

Рис. 1.23. X-Y график двух векторов

В этом случае в местозаполнители возле осей вводятся просто имена векторов. Также допускается откладывать по осям элементы векторов, т. е. вводить в местозаполнители возле осей имена x_i и y_i соответственно. В результате получается график, на котором отложены точки, соответствующие парам элементов векторов, соединенные отрезками прямых линий. Образованная ими ломаная называется *рядом данных*, или *кривой* (trace).

Примечание

Обратите внимание, что Mathcad автоматически определяет границы графика, исходя из диапазона значений элементов векторов.

Стоит отметить, что подобным образом легко создать и X-Y график столбцов или строк матрицы, применяя оператор выделения столбца и откладывая соответствующие выражения по осям графика (множество подобных примеров вы найдете на рисунках в последующих главах книги).

1.4.4. График функции

Нарисовать график любой скалярной функции `f(x)` можно двумя способами. Первый заключается в дискретизации значений функции, присвоении этих значений

вектору и прорисовке графика вектора. Собственно, так и были получены графики синуса на рис. 1.23. Второй, более простой способ, называемый *быстрым построением графика*, заключается во введении функции в один из местозаполнителей (например, у оси Y), а имени аргумента — в местозаполнитель у другой оси (рис. 1.24).

В результате Mathcad сам создает график функции в пределах некоторых предустановленных значений аргумента. Разумеется, впоследствии можно поменять диапазон значений аргумента, и график автоматически подстроится под него.

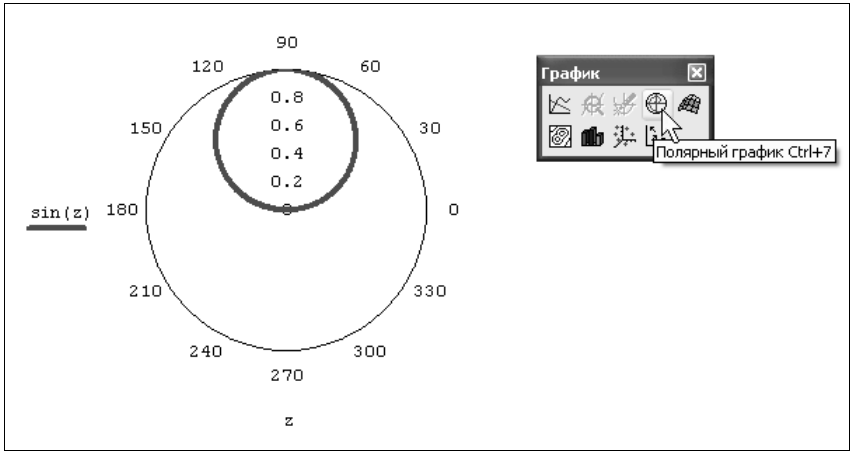


Рис. 1.24. Быстрое построение графика функции

Необходимо заметить, что если переменной аргумента функции было присвоено значение до построения в документе графика, то вместо быстрого построения графика будет нарисована зависимость функции с учетом этого значения.

1.4.5. Построение нескольких рядов данных

На одном графике может быть отложено до 16 различных зависимостей. Чтобы построить на графике еще одну кривую, необходимо выполнить следующие действия:

1. Поместите линии ввода таким образом, чтобы они целиком захватывали выражение, стоящее в надписи координатной оси Y (рис. 1.25).
2. Нажмите клавишу <,>.
3. В результате появится местозаполнитель, в который нужно ввести выражение для второй кривой.
4. Щелкните в любом месте вне этого выражения (на графике или вне его).

После этого вторая кривая будет отображена на графике. На рис. 1.25 уже нарисованы два ряда данных, а нажатие клавиши с запятой <,> приведет к появлению третьего местозаполнителя, с помощью которого можно определить третий ряд данных.

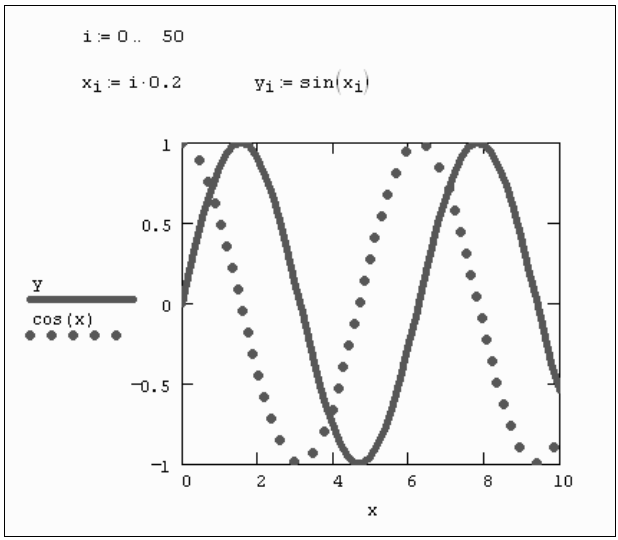


Рис. 1.25. Построение нескольких зависимостей на одном графике

Примечание

Чтобы убрать один или несколько рядов данных с графика, удалите клавишами <BackSpace> или <Del> соответствующие им надписи у координатных осей.

Описанным способом будет создано несколько зависимостей, относящихся к одному аргументу. Чтобы на одном и том же графике отложить функции разного аргумента, следует ввести имена этих аргументов через запятую возле оси x (как это показано на рис. 1.26).

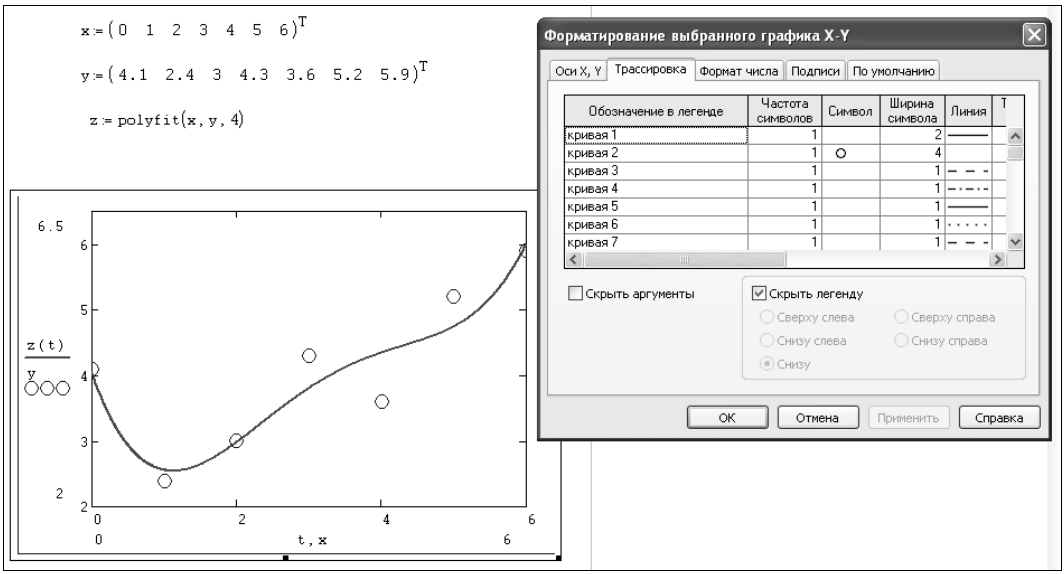


Рис. 1.26. Форматирование графика

1.4.6. Форматирование графиков

Форматирование рядов данных, диапазона координатных осей, выбор типа графика и установка прочих атрибутов осуществляется при помощи диалогового окна **Форматирование выбранного графика** (Formatting Currently Selected Plot). В частности, с помощью вкладки **Ряды данных** (Traces) легко установить комбинацию параметров линии и точек для каждого из рядов данных, представленных на графике. Пользователю требуется выделить в списке нужный ряд данных (его положение в списке соответствует положению метки зависимости у оси `Y`) и изменить в списках в середине диалогового окна желаемые установки (рис. 1.26).

1.4.7. Трехмерные графики

Коллекция трехмерных графиков — настоящее чудо, которое Mathcad дарит пользователю. За считанные секунды вы можете создать великолепную презентацию результатов своих расчетов. Рамки данной книги не позволяют описать технику их создания и форматирования подробно, поэтому мы ограничимся лишь вводными замечаниями и простыми примерами, которые помогут ориентироваться в дальнейшем материале. Для этого рассмотрим на простом примере функции `z(x,y)` и матрицы `Z` (они заданы в листингах 1.25 и 1.26 соответственно) технику построения трехмерных графиков различных типов.

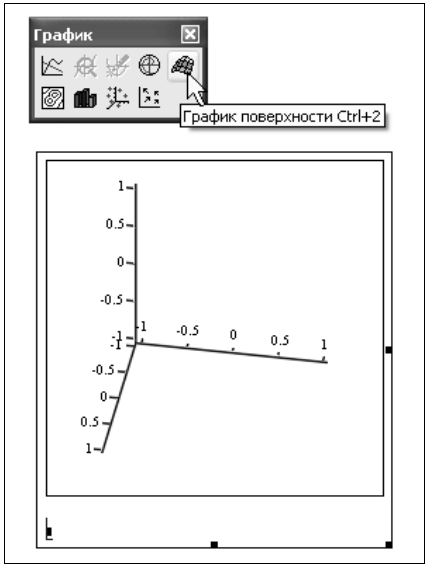


Рис. 1.27. Создание трехмерного графика

Чтобы создать трехмерный график, требуется нажать кнопку с изображением любого из типов трехмерных графиков на панели инструментов **График** (Graph). В результате появится пустая область графика с тремя осями (рис. 1.27) и единственным местозаполнителем в нижнем левом углу. В этот местозаполнитель следует ввести либо имя `z` функции `z(x,y)` двух переменных для *быстрого построения трехмерного графика* (рис. 1.28), либо имя матричной переменной `Z`, которая задаст распределение данных `Zx,y` на плоскости `XY` (рис. 1.29). Еще раз отметим, что для получения графиков (и этих, и последующих) не требуется никакого текста, кроме соответствующего листинга и введения имени соответствующей функции или матрицы в местозаполнитель.

Листинг 1.25. Функция для быстрого построения трехмерных графиков

$$z(x, y) := x^2 + y^2$$

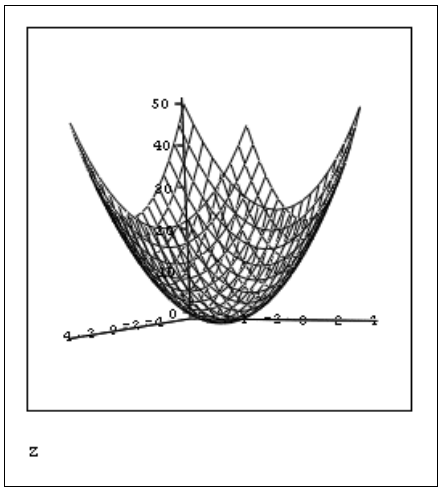


Рис. 1.28. Быстрое построение графика поверхности функции (продолжение листинга 1.25)

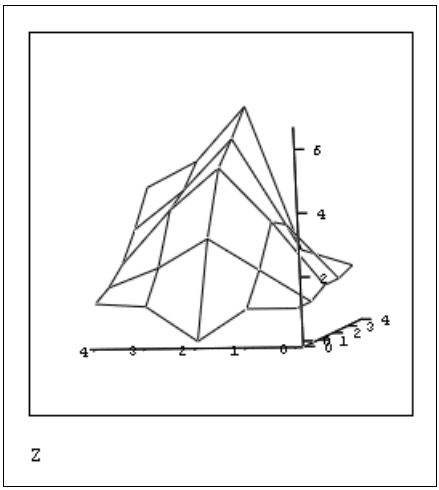


Рис. 1.29. График поверхности, заданный матрицей (продолжение листинга 1.26)

Листинг 1.26. Матрица для отображения на трехмерных графиках

$$Z := \begin{pmatrix} 1 & 1 & 0 & 1.1 & 1.2 \\ 1 & 2 & 3 & 2.1 & 1.5 \\ 1.3 & 3.3 & 5 & 3.7 & 2 \\ 1.3 & 3 & 5.7 & 4.1 & 2.9 \\ 1.5 & 2 & 6.5 & 4.8 & 4 \end{pmatrix}$$

Помимо трехмерных графиков поверхности, нажатие соответствующих кнопок на панели **График** (Graph) приводит к созданию графика линий уровня, трехмерной гистограммы, трехмерного распределения точек или векторного поля.

1.5. Отладка и комментирование программ

Программа Mathcad, несмотря на исключительную дружественность к пользователю и близость к обычным компьютерным редакторам, все-таки является полноценной системой программирования. Благодаря этому, разработчики предусмотрели ряд инструментов, которые служат для отладки расчетов в среде Mathcad, а также для их комментирования.

1.5.1. Сообщения об ошибках

Когда процессор Mathcad по тем или иным причинам не может вычислить выражение, он вместо ответа выдает сообщение об ошибке. Если курсор находится вне формулы с ошибкой, то в ней имя функции или переменной, которая вызвала ошибку, отмечается красным цветом (внизу на рис. 1.30).

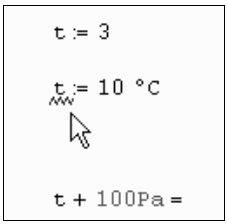


Рис. 1.30. Mathcad автоматически отыскивает ошибки в формулах

Рис. 1.31. Сообщение об ошибке

Помимо, собственно, сообщений об ошибках, которые приводят к аварийной остановке расчетов некоторых формул, последние версии Mathcad выдают пользователю дополнительные предупреждения о вероятных ошибках. Эти предупреждения выделяются при помощи подчеркивания волнистой зеленой линией и связаны, главным образом, с переопределением уже существующих переменных или функций, как встроенных, так и пользовательских (как в середине рис. 1.30).

При щелчке на формуле с ошибкой (либо подозрением на ошибку) происходит ее выделение, сопровождающееся всплывающей подсказкой с сообщением о типе ошибки (рис. 1.31).

Если некоторые выражения вызывают ошибку, они просто игнорируются, а следующие выражения в документе по-прежнему вычисляются. Конечно, если формулы, вызвавшие ошибку, влияют на значения нижеследующих формул, то они будут также интерпретированы как ошибочные. Поэтому, встречая в документе сообщения об ошибках, найдите сначала самое первое из них. Часто ее устранение позволяет избавиться и от последующих ошибок.

⑬ 1.5.2. Отладка программ

Начиная с версии 13, Mathcad стал обладать расширенными возможностями по отладке программ. Для отладки предназначены три новых элемента Mathcad, связанные между собой: встроенные функции `trace` и `pause`, окно **Окно трассировки** (Trace Window), а также панель инструментов **Отладка** (Debug).

Встроенные функции отладки позволяют организовать в документах Mathcad определенные точки, в которых можно наблюдать значения любых переменных и функций, а также прерывать и продолжать расчеты. Сами значения наблюдаемых переменных выводятся в текстовом виде в окно **Окно трассировки** (Trace Window), а управлять приостановкой и продолжением вычислений, а также опциями отладки, можно при помощи панели **Отладка** (Debug).

Панель **Отладка** (Debug) содержит следующие кнопки (рис. 1.32):

- **Возобновить** (Resume) — для продолжения процесса вычислений, прерванного функцией `pause`;
- **Прервать** (Interrupt) — для отключения дальнейших вычислений (прерванных благодаря работе функции `pause`);

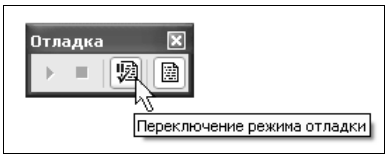


Рис. 1.32. Панель **Отладка** (Debug)

- **Переключение режима отладки** (Toggle Debugging) — для включения (или отключения) опции отладки (т. е. работы функций отладки);
- **Окно трассировки** (Trace Window) — для вызова на экран (или скрытия) окна отладки.

> ***Примечание***
> Вызвать панель **Отладка** (Debug) на экран можно посредством команды меню **Вид / Панели инструментов / Отладка** (View / Toolbars / Debug).

В Mathcad 13 появились следующие функции отладки:

- `trace ([format_string],x])` — функция, осуществляющая вывод наблюдаемых переменных в окно трассировки;
- `pause ([format_string],x])` — функция, осуществляющая приостановку процесса вычислений с одновременным выводом наблюдаемых переменных в окно трассировки, где:
  - `format_string` — строка, определяющая формат вывода (*см. пример ниже*);
  - `x` — наблюдаемая переменная (или список переменных через запятую).

Пример использования функции `trace` в документе приведен на рис. 1.33. Каждое обращение к функции `trace` приводит к появлению в окне **Окно трассировки** (Trace Window) соответствующего значения переменной, за которой осуществляется наблюдение.

> ***Внимание!***
> Помните о том, что опция отладки (т. е. прерывание вычислений и вывод наблюдаемых величин в окно трассировки) будет работать, только если кнопка **Переключение режима отладки** (Toggle Debugging) на панели **Отладка** (Debug) находится в нажатом состоянии. Возможно, потребуется пересчет документа, чтобы наблюдаемые переменные появились в окне трассировки.

Пример использования функции `trace` в программном блоке иллюстрируется рис. 1.34.

Организацию точек останова вычислений в теле документа при помощи функции `pause` иллюстрирует рис. 1.35. Отметим, что использование первого строкового аргумента функций `trace` и `pause` позволяет задать формат вывода и снабдить его соответствующими комментариями. Продолжить вычисления или прервать их можно нажатием кнопки **Возобновить** (Resume) или **Прервать** (Interrupt) соответственно на панели **Отладка** (Debug).

Содержимое окна **Окно трассировки** (Trace Window) можно скопировать в буфер обмена или попросту уничтожить (для того чтобы возобновить затем процесс от-

ладки "с чистого листа"). Для этого следует вызвать из области окна **Окно трассировки** (Trace Window) контекстное меню и выбрать в нем соответствующую команду (рис. 1.36):

- **Копировать** (Copy) — для копирования выделенного фрагмента в буфер обмена;
- **Выделить все** (Select all) — для выделения всего содержимого окна отладки;
- **Удалить все** (Clear all) — для очистки окна отладки.

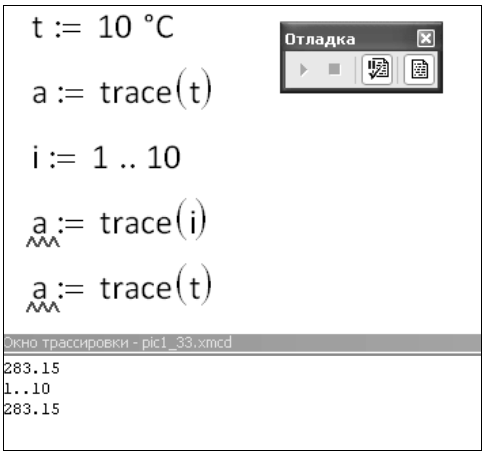


Рис. 1.33. Отладка документа Mathcad при помощи окна **Окно трассировки** (Trace Window)

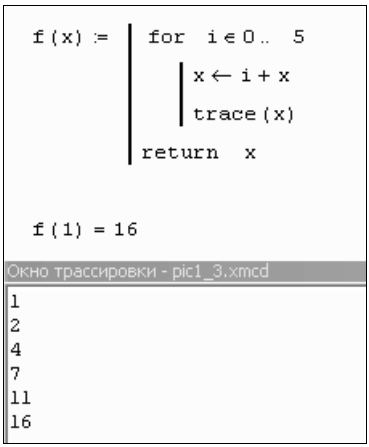


Рис. 1.34. Отслеживание значений переменных в программных блоках

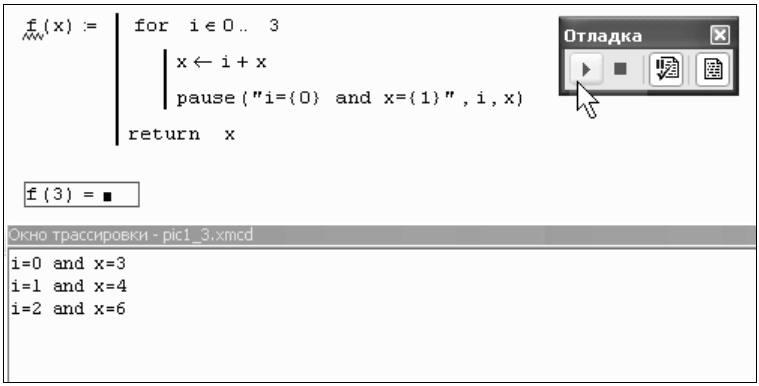


Рис. 1.35. Организация прерывания вычислений

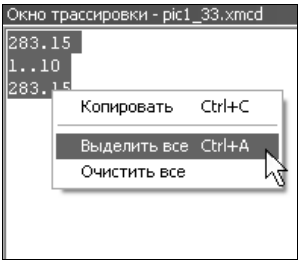


Рис. 1.36. Управление содержимым окна трассировки

1.5.3. Обработка ошибок в программных модулях

Программирование в Mathcad позволяет осуществлять дополнительную обработку ошибок. Если пользователь предполагает, что выполнение кода в каком-либо месте программного модуля способно вызвать ошибку (например, деление на ноль), то эту ошибку можно перехватить с помощью оператора `on error`. Чтобы вставить его в программу, надо поместить линии ввода в нужное положение и нажать кнопку с именем оператора `on error` на панели **Программирование** (Programming). В результате появится строка с двумя местозаполнителями и оператором `on error` посередине. В правом местозаполнителе следует ввести выражение, которое должно выполняться в данной строке программы. В левом — выражение, которое будет выполнено вместо правого выражения, если при выполнении последнего возникнет ошибка.

Приведем пример использования оператора `on error` (листинг 1.27) в программном модуле, который рассчитывает функцию обратного числа значению `n`. Если `n≠0`, то и присвоенное значение `z≠0`, поэтому в последней строке программы выполняется правое выражение расчета `1/z`. Так происходит при расчете `f(-2)`. Если попытаться вычислить `f(0)`, как в конце листинга, то выполнение программы, заложенной в `f(n)`, вызовет ошибку деления на ноль в последней строке программы. Соответственно, вместо выражения справа от оператора `on error` будет выполнено левое выражение, присваивающее функции `f(n)` строковое значение `"user error: can't divide by zero"` (пользовательская ошибка: деление на ноль невозможно). Конечно, этой строке можно присвоить и текст на русском языке.

Листинг 1.27. Перехват ошибки деления на ноль

```
f(n) := | z ← n
        |
        | "user error: can't divide by zero"   on error 1/z

f(-2) → -1/2

f(0) = "user error: can't divide by zero"
```

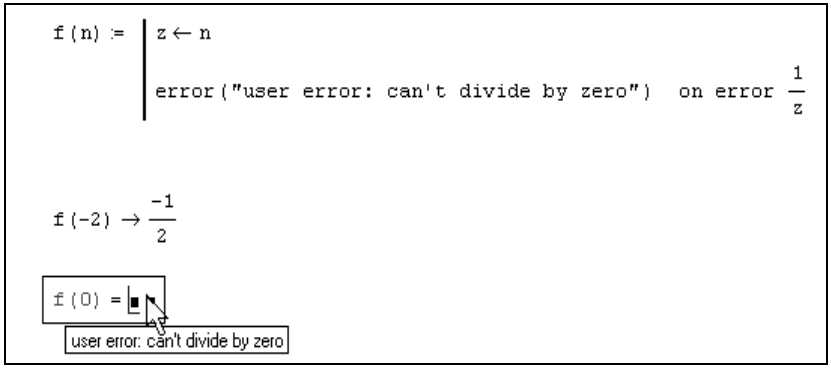


Рис. 1.37. Перехват ошибки деления на ноль

Оператор перехвата ошибок удобно применять в комбинации со встроенной функцией `error(S)`. Она приводит к генерации ошибки в обычной для Mathcad форме с сообщением `S`. Пример усовершенствования листинга 1.27 для такого стиля обработки ошибки деления на ноль показан на рис. 1.37. Обратите внимание, что сделанные изменения свелись к помещению текста сообщения об ошибке в аргумент функции `error`.

1.5.4. Текст и комментарии

Документы Mathcad могут содержать текстовые объекты, а также разного рода комментарии и примечания.

Текстовые блоки

Для того чтобы ввести текст непосредственно в рабочую область документа Mathcad, достаточно непосредственно перед началом ввода текста нажать клавишу <">. В результате, в месте расположения курсора ввода появится область с характерным выделением, обозначающая, что ее содержимое не будет восприниматься процессором Mathcad в качестве формул, а станет простым текстовым блоком (рис. 1.38, фразы "Рассмотрим квадратное уравнение" и "Детерминант").

Редактировать атрибуты текста в пределах блоков можно стандартными для текстовых редакторов средствами панели **Форматирование** (Formatting).

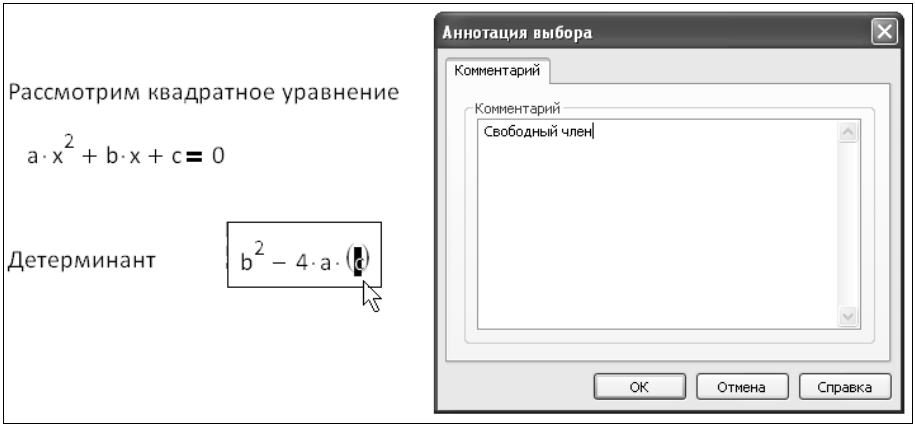


Рис. 1.38. Текст в рабочей области документа и комментарий к фрагменту формулы

14 ***Примечание***

В версии Mathcad 14, наконец, появилась полная поддержка шрифтов в кодировке Unicode. То есть теперь можно вставлять в документы русскоязычные текстовые блоки, комментарии и имена переменных и функций, не задумываясь о том, что это создаст опасность возникновения ошибок при переходе к новой версии Mathcad.

В тексте могут быть размечены *гиперссылки* — активные области в документах, которые выводят на экран какой-либо другой фрагмент активного документа, дру-

гой документ Mathcad или переводят на некоторый сайт в Интернете. Для вставки гиперссылки достаточно выделить нужный фрагмент текстовой или формульной области документа, выбрать в меню **Вставка** (Insert) пункт **Гиперссылка** (Hyperlink) и в диалоговом окне **Вставка гиперссылки** (Insert Hyperlink) в текстовом поле **Ссылка на файл или URL-адрес** (Link to file or URL) определить путь к документу или Web-странице.

⑫ Комментарии

Несколько нововведений, связанных с интерфейсом, помогут вам разрабатывать документы Mathcad с большим комфортом. Все они представляют собой дополнительные опции вставки в документы комментариев различного рода. Предусмотрено несколько типов комментариев, называемых *примечаниями* (annotation) и *метаданными* (metadata):

- Комментарии к файлу документа как единому целому, облегчающие его идентификацию как при работе в Mathcad, так и при поиске и отборе файлов средствами ОС Windows. Для добавления и редактирования комментариев ко всему документу выполните команду **Файл / Свойства** (File / Properties) и в открывшемся диалоговом окне установите стандартные свойства для данного файла (заголовок, автор, комментарии, ключевые слова).
- Примечания к отдельным выражениям, представляющие собой обычный текст небольшого размера. Для их добавления выделите выражение, вызовите контекстное меню и выберите в нем пункт **Аннотация выбора** (Annotate Selection). В открывшемся диалоге теперь можно ввести текст примечания. Части формул, которые снабжены примечаниями, выделяются (при установке линий ввода в пределы выражения) дополнительными скобками зеленого цвета (как показано на рис. 1.38 для выделенной переменной).

  ⑬ ***Примечание***

  Отключить выделение примечаний можно командой верхнего меню **Вид / Аннотации** (View / Annotations), а поменять назначенный им цвет — при помощи команды **Формат / Цвет / Аннотация** (Format / Color / Annotations).

- Комментарии (метаданные) к отдельным элементам формул (переменным, функциям, выражениям), позволяющие задать для них несколько параметров. Для их создания выделите желаемую часть формулы, вызовите контекстное меню, выберите в нем пункт **Свойства** (Properties) и перейдите в открывшемся диалоге на вкладку **Настройка** (Custom). При помощи группы раскрывающихся списков можно добавить параметры разного вида и установить для них определенное значение того или иного типа (текст, число, дату, да/нет).
- Ключевые слова в документе (для глоссария), разметка которых производится в том же диалоге **Свойства** (Properties) на вкладке **Сводка** (Summary).

1.6. Управление файлами документов

Расчеты Mathcad организуются стандартным для Windows-приложений образом: в виде *документов* Mathcad, изначально пустых, на которые можно добавлять формулы и текст. Каждый документ представляет собой независимую серию математических расчетов и сохраняется в отдельном файле. Документ является одновременно и листингом Mathcad-программы, и результатом исполнения этой программы, получающимся при ее выполнении, и отчетом, пригодным для распечатки на принтере или публикации в Web.

1.6.1. Создание нового документа

Для того чтобы создать новый пустой документ, уже работая в Mathcad, следует выполнить одно из трех эквивалентных действий:

- одновременное нажатие клавиш <Ctrl>+<N>;
- нажатие кнопки **Создать** (New) на панели инструментов;
- ввод команды верхнего меню **Файл / Создать** (File / New).

Кнопка **Создать** (New) на стандартной панели состоит из двух частей (рис. 1.39). При нажатии ее левой половины (значка в виде чистого листа), как и при наборе комбинации клавиш <Ctrl>+<N>, создается новый пустой документ. Если же нажать правую часть (маленькую стрелку), это приведет к появлению выпадающего списка *шаблонов* нового документа.

Для создания пустого документа необходимо выделить в списке выпадающего списка (если вы используете панель инструментов) или диалогового окна окно **Соз-**

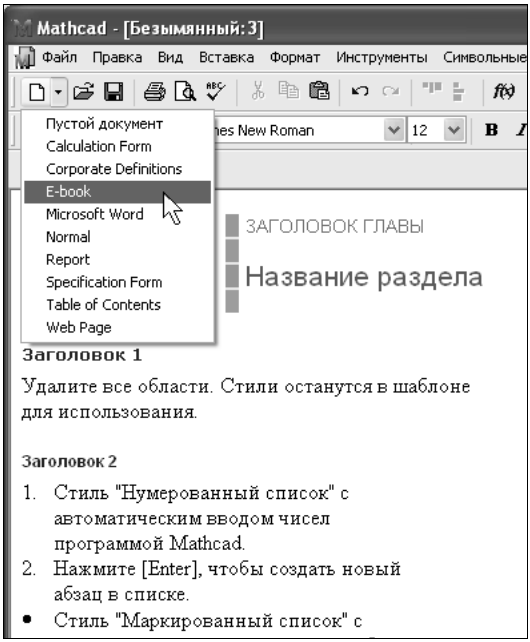


Рис. 1.39. Создание документа на основе шаблона

дать (New) (если вы действуете через меню) пункт **Пустой документ** (Blank Worksheet) и нажать кнопку **OK**. В результате в окне Mathcad появляется пустой документ с условным названием **Безымянный: 2** (Untitled: 2) или **Безымянный: 3** (Untitled: 3) и т. д., в зависимости от того, какой по счету новый документ создается.

Помимо начала работы с пустым документом, можно использовать один из предустановленных шаблонов расчетов, который наилучшим образом подходит для вашей задачи. *Шаблон* (template) — это заготовка нового документа с введенными формулами, графиками, текстом, включая разметку, форматирование, выбор по умолчанию режима вычислений и т. д. Помимо пустого шаблона, который предназначен для начала редактирования документа "с чистого листа", в Mathcad имеется несколько предустановленных шаблонов, список которых отображен на рис. 1.39. Для создания нового документа на основе одного из этих шаблонов достаточно просто выбрать в списке нужный вам шаблон и начать работу с появившимся документом, на котором уже будут изначально размещены характерные для того или иного шаблона элементы оформления.

1.6.2. Открытие существующего документа

Чтобы открыть существующий документ для редактирования, выполните команду **Файл / Открыть** (File / Open) или нажмите комбинацию клавиш <Ctrl>+<O> (или кнопку **Открыть** (Open) на стандартной панели инструментов). В диалоговом окне **Открыть** (Open) выберите файл и нажмите кнопку **OK**. Таким способом можно открывать документы не только с дисков вашего компьютера, но и из локальной сети и даже с удаленных серверов Интернета.

Открыть файл можно и в обозревателе Windows, щелкнув дважды на его имени с расширением xmcd или mcd.

1.6.3. Сохранение документа

Для того чтобы сохранить документ в формате Mathcad, выберите **Файл / Сохранить** (File / Save) либо нажмите комбинацию клавиш <Ctrl>+<S> или кнопку **Сохранить** (Save) на стандартной панели инструментов. Если созданный документ сохраняется впервые, на экран будет выведено диалоговое окно **Сохранить как** (Save As), в котором потребуется определить его имя (рис. 1.40).

Начиная с версии 12, изменен формат файлов, использующийся по умолчанию для хранения документов Mathcad. Теперь они сохраняются в формате XMCD, являющемся разновидностью текстовой XML-разметки (и следующим шагом по сравнению с форматом MathML, примененным в версии 2001). Преимущества этого нововведения практически не заметны для рядового пользователя, процедура сохранения для которого осталась практически неизменной. Применение XML-формата оправдано, во-первых, тем, что он становится общеупотребительным для целого ряда приложений и данных самого различного типа. Во-вторых, удобство XML-файлов заключается в возможности использовать для просмотра и манипуляций с Mathcad-

документами другие (предусматривающие это) приложения. Их можно также просматривать и редактировать "вручную", в любом текстовом редакторе. Наконец, XMCD-файлы более надежно конвертируются в файлы других математических форматов и HTML.

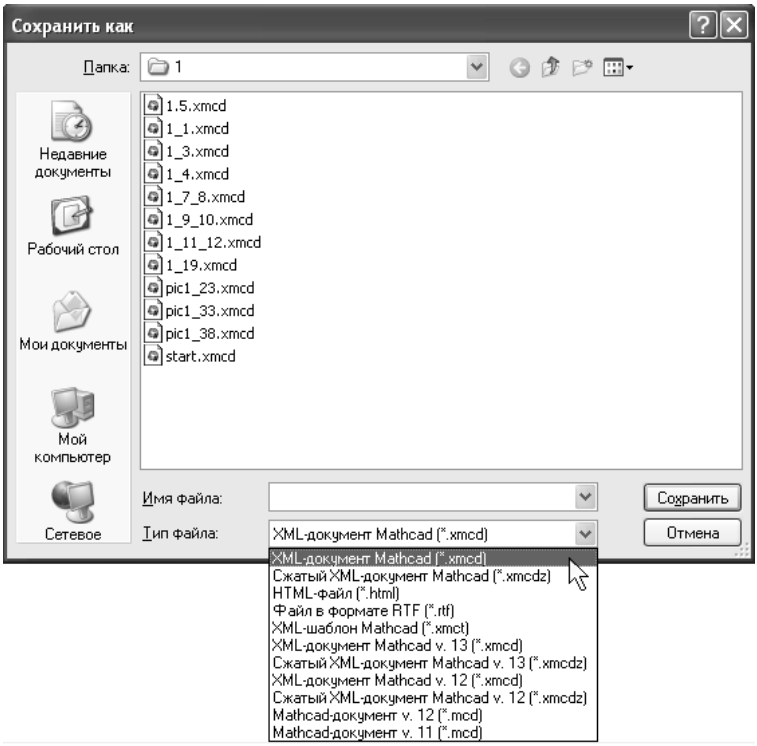


Рис. 1.40. Сохранение документа

Помимо формата XMCD поддерживаются форматы файлов MCD прежних версий Mathcad — от 11-й до 13-й, а также XML-формат со сжатием данных, называемый XMCDZ. В Mathcad 12 и выше сохранены опции экспорта в Web-страницы (HTML) и текстовые RTF-файлы, записывающие информацию о разметке документа и успешно читаемые такими редакторами, как Microsoft Word. Расчеты Mathcad, сохраненные в виде Web-страницы, можно просматривать в окне любого браузера как стандартный гипертекстовый документ.

Чтобы переименовать документ, сохраните его под другим именем с помощью команды **Файл / Сохранить как** (File / Save As) и появляющегося в результате диалогового окна **Сохранить как** (Save As). В этом случае файл со старым названием не изменяется.

На рис. 1.40 показан раскрывающийся список с возможными форматами сохраняемых файлов:

- **XML-документ Mathcad (*.xmcd)** — последний и наиболее современный формат на основе XML-разметки, используется по умолчанию;

- **Сжатый XML-документ Mathcad (*.xmcdz)** — сжатый XML-документ, т. е. тот же XML-формат, с дополнительной компрессией файла;
- **HTML-файл (*.html)** — формат Web-страницы, полностью сохраняющий информацию о расчетах и позволяющий впоследствии открывать их в Mathcad как обычные документы;
- **Файл в формате RTF (*.rtf)** — сохраняйте файлы в этом формате только для дальнейшего редактирования в текстовых редакторах с целью создания отчетов. В частности, сохранив документ в RTF-файле, можно загрузить его в Microsoft Word, OpenOffice.org или другой текстовый процессор, большинство из которых поддерживает этот формат;
- **XML-шаблон Mathcad (*.xmct)** — формат шаблона на основе XML-разметки;
- форматы прежних версий Mathcad. Если вы работаете одновременно с несколькими версиями Mathcad (например, решаете с другими разработчиками общую задачу), то сохраняйте файлы в наиболее раннем формате из тех, с которыми приходится иметь дело. Однако помните, что возможности прежних версий (в частности, наборы встроенных функций) ограничены по сравнению с более поздними версиями Mathcad, поэтому некоторые из них будут недоступны.

> 13 ***Примечание***
>
> Начиная с версии Mathcad 13, разработчики предусмотрели стандартную опцию *автоматического сохранения* (или, короче, *автосохранения*) файлов документов (в фоновом режиме). Если включить данную опцию на вкладке **Save** (Сохранение) диалогового окна **Параметры** (Preferences), которое можно вызвать одноименной командой меню **Инструменты** (Tools), то Mathcad будет автоматически сохранять в определенном месте на диске резервные копии документов. В случае возникновения нештатной ситуации (аварийного завершения работы программы) при последующем запуске Mathcad и обращении к несохраненным файлам будет выдано диалоговое окно с предложением вернуться к их автоматически сохраненным резервным копиям.

1.6.4. Управление вычислениями

Как мы уже говорили, документ Mathcad — это в полном смысле слова компьютерная программа, а сама система Mathcad — настоящая система программирования, правда, ориентированная на математика, а не на профессионального программиста. Большинство других сред программирования (знакомых читателю по реализации таких языков, как Си, Фортран и т. п.) разделяют редактирование кода программ и их выполнение, которое можно вызвать предназначенными для этого командами. В Mathcad и код программы, и результат их выполнения объединены в документе. Тем не менее функции редактирования формул и их расчеты разделены, и пользователь имеет возможность управлять всеми важнейшими опциями вычислений.

Вообще говоря, имеются два режима вычислений:

- *автоматический режим* (automatic mode) — все вычисления выполняются автоматически по мере ввода формул;

- *ручной режим* (manual mode) — старт вычислений каждой формулы или всего документа производится пользователем.

Режим вычислений можно выбрать с помощью команды **Инструменты / Вычислить / Автоматический расчет** (Tools / Calculate / Automatic Calculation), как показано на рис. 1.41. Если в этой строке меню установлен флажок проверки, значит, включен автоматический режим, если флажка нет, то редактируется документ в ручном режиме вычислений. Чтобы сменить режим, просто выберите этот пункт меню (например, нажав кнопку мыши в ситуации, показанной на рис. 1.41, включите ручной режим).

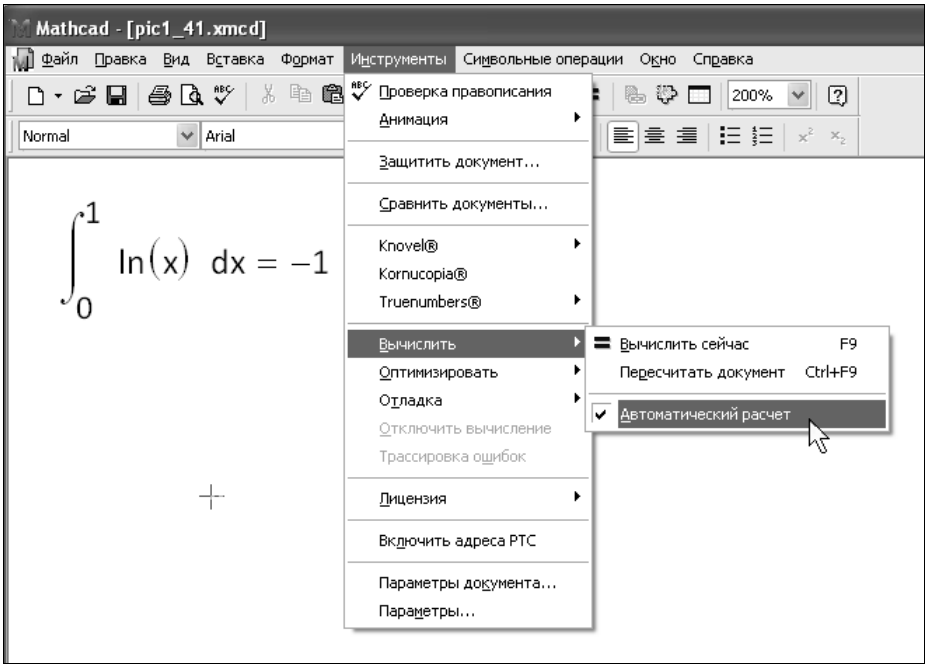


Рис. 1.41. Выбор режима вычислений

Mathcad осуществляет вычисления документа, как это принято в большинстве сред программирования: сверху вниз и слева направо. Чтобы прервать затянувшийся процесс вычислений, нажмите клавишу <Esc>. Появится диалоговое окно, в котором нужно подтвердить прерывание вычислений (нажать кнопку **OK**). В этом случае выражения, которые Mathcad не успел вычислить, будут помечены в документы красным цветом. Прерванные вычисления возобновляются нажатием клавиши <F9> или командой **Инструменты / Вычислить / Вычислить сейчас** (Tools / Calculate / Calculate Now).

Установки режимов вычислений для всего документа сведены на вкладке **Вычисления** (Calculations) диалогового окна **Параметры документа** (Worksheet Options), вызываемого с помощью команды **Инструменты / Параметры документа** (Tools / Worksheet Options). Опции сгруппированы на нескольких вкладках (рис. 1.42), а их набор изменялся от версии к версии Mathcad.

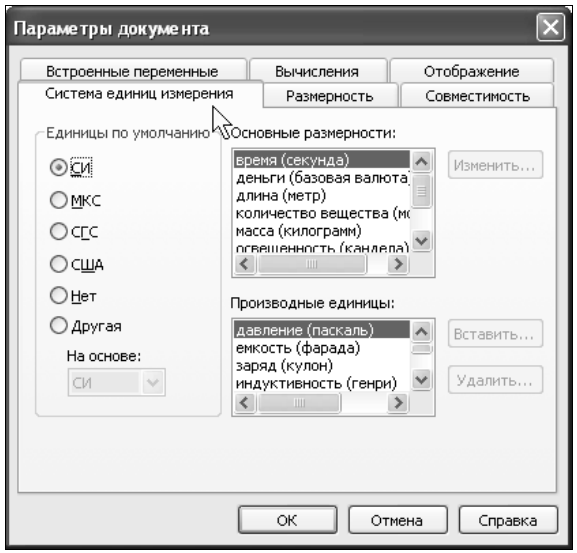


Рис. 1.42. Управление режимом вычислений документа

- **Система единиц измерения** (Unit System) — выбор системы, в которой отображаются размерные величины (СИ, СГС и т. д., в том числе пользовательская система);
- **Размерность** (Dimensions) — задание набора единиц измерения основных размерных величин;
- **Совместимость** (Compatibility) — включение опций работы математического процессора по обработке команд присваивания в стиле версий Mathcad 11 или 12;
- **Встроенные переменные** (Built-In Variables) — определение значений системных констант, таких как `ORIGIN`, `TOL` или `CTOL`;
- **Вычисления** (Calculation) — выбор опций режима вычислений (например, результата деления 0/0, предварительной проверки матриц на сингулярность и др.);
- **Отображение** (Display) — выбор установок по умолчанию для символов операций, допускающих разное отображение на экране (например, присваивания, численного и символьного вывода, умножения).

1.7. О дополнительных элементах оформления документов

Для оформления документов в Mathcad могут применяться следующие основные элементы (рис. 1.43, сверху вниз):

- текстовые области (text region);
- математические области или формулы (math region);

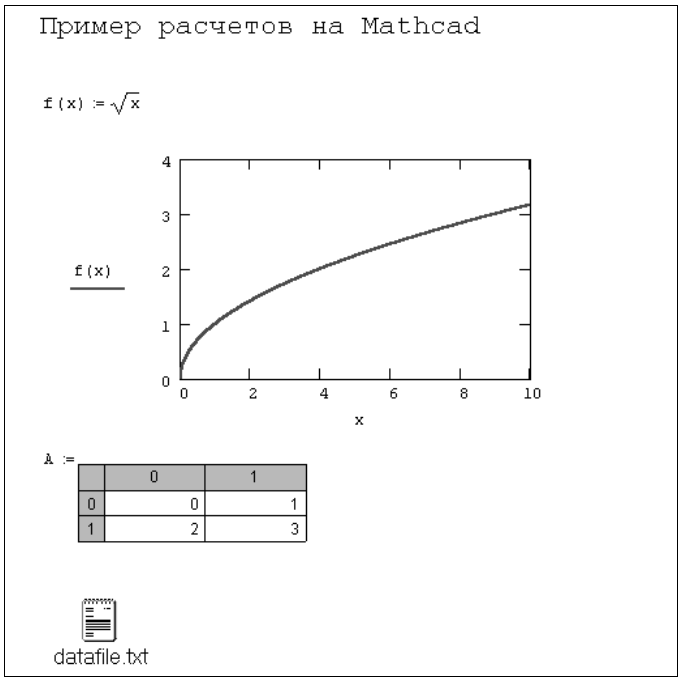


Рис. 1.43. Основные элементы оформления документов Mathcad

- графики или графические области (graphics region);
- компоненты других приложений (component);
- внедренные объекты (object).

За пределами границ областей находится пустая часть документа. Кроме перечисленных, часто бывает полезным применение следующих дополнительных элементов оформления:

- закрытые и выделенные области (locked and highlighted area);
- колонтитулы (header, footer);
- разметка документов — разрывы страниц (page break), стили (styles) и поля (margins);
- ссылки (references);
- гиперссылки (hyperlinks);
- рисунки (pictures).

Большинство перечисленных объектов можно добавить в документ при помощи соответствующих команд меню **Вставка** (Insert).

Приведем несколько неочевидных возможностей форматирования, которые специфичны для Mathcad.

- Команду **Формат / Разделить регионы** (Format / Separate Regions) удобно использовать, когда некоторые из объектов в документе (тест, формулы) перекрываются рисунками (pictures).

- Если вы по какой-либо причине считаете, что некоторые регионы должны в документе перекрываться, то порядком их наложения друг на друга можно управлять при помощи команд контекстного меню **На передний план** (Bring to Front) или **На задний план** (Send to Back).
- Чтобы выделить формулу или текст цветом, вызовите нажатием на ней правой кнопкой мыши контекстное меню и выберите в нем пункт **Свойства** (Properties) либо такой же пункт в меню **Формат** (Format). Установите в диалоговом окне **Properties** (Свойства) флажок **Выделить регион** (Highlight Region) и нажмите кнопку **OK** (рис. 1.44).

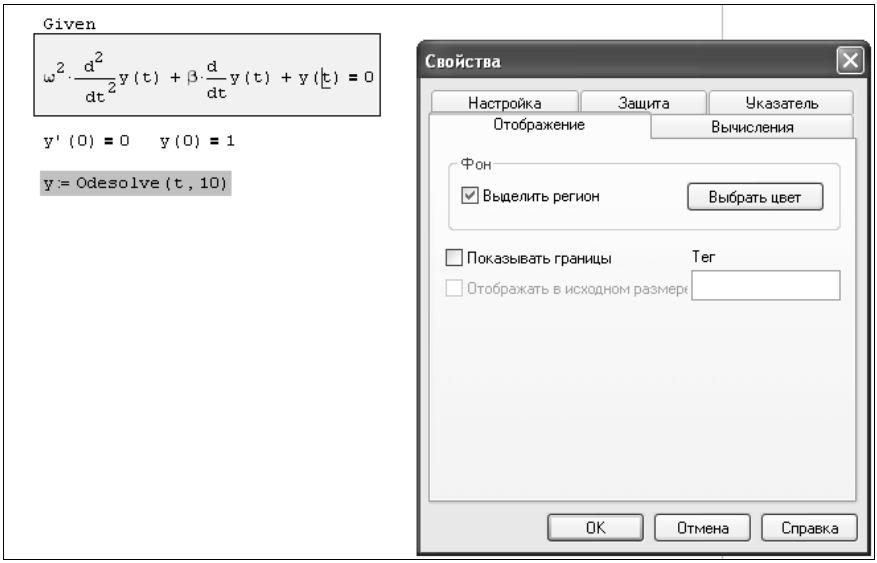


Рис. 1.44. Выделение формулы цветом

- Для того чтобы задать цвет фона всего документа, выберите команду **Формат / Цвет / Фон** (Format / Color / Background) и определите в палитре нужный цвет.
- Для включения обрамления установите флажок **Показывать границы** (Show Border) в том же самом диалоговом окне **Свойства** (Properties). Обрамление может применяться вместе с выделением цветом.
- Для вставки рисунка в документ нажмите кнопку **Рисунок** (Picture) на панели **Матрица** (Matrix) и в местозаполнитель появившейся области введите в кавычках имя файла. Если ввести в местозаполнитель имя какой-либо определенной ранее в документе матрицы, то созданный рисунок отразит строение этой матрицы (что очень эффективно для визуализации матриц большого размера, особенно разреженных).
- При выделении рисунка на экран автоматически будет вызвана панель инструментов **Панель инструментов** (Picture), которая позволяет его редактировать, применяя довольно развитые графические средства, например зеркальное отражение рисунка, увеличение его фрагмента и т. п. (рис. 1.45).

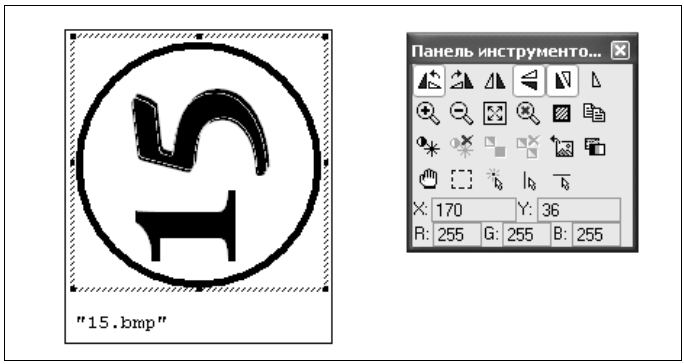


Рис. 1.45. Рисунок в документе

- Участки документа Mathcad можно объединять в *области* (area), применяемые для разграничения фрагментов документа по смыслу и для их временного скрытия (рис. 1.46). Чтобы создать новую зону в документе, достаточно поместить курсор ввода в желаемое место и выбрать в верхнем меню пункт **Вставка / Область** (Insert / Area).

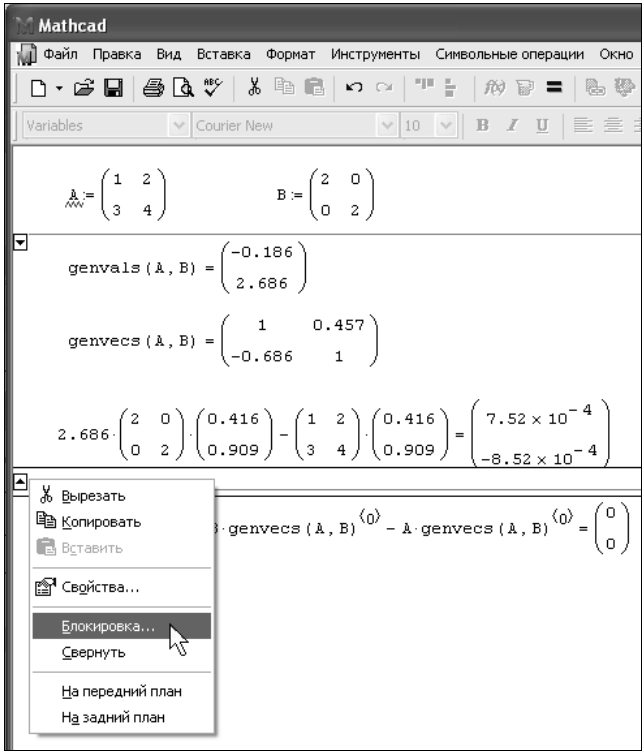


Рис. 1.46. Блокировка области документа

- Чтобы скрыть зону, дважды щелкните мышью на любой из линий, ее выделяющих. Альтернативный способ заключается в помещении курсора внутрь зоны и

выполнении команды **Формат / Область / Свернуть** (Format / Area / Collapse). Для удаления зоны из документа выделите щелчком мыши любую из горизонтальных линий и нажмите клавишу <Del>.

- Для запирания зоны (защиты ее содержимого от редактирования) выберите команду **Формат / Область / Блокировка** (Format / Area / Lock).
- Чтобы вставить в документ колонтитулы, выберите пункт **Колонтитулы** (Header and Footer) меню **View** (Вид). Откройте одну из вкладок (рис. 1.47): **Верхний колонтитул** (Header) или **Нижний колонтитул** (Footer) и вставьте текст колонтитула в одно из полей: **Слева** (Left), **В центре** (Center) или **Справа** (Right). Текст колонтитула можно комбинировать с дополнительной информацией (дата, номер страницы, имя файла и т. п.) при помощи кнопок с соответствующим значком.

Примечание

Колонтитулы влияют только на вид печатных копий документа. Просмотреть их можно, к примеру, в режиме предварительного просмотра, выбрав команду **Файл / Предварительный просмотр** (File / Print Preview).

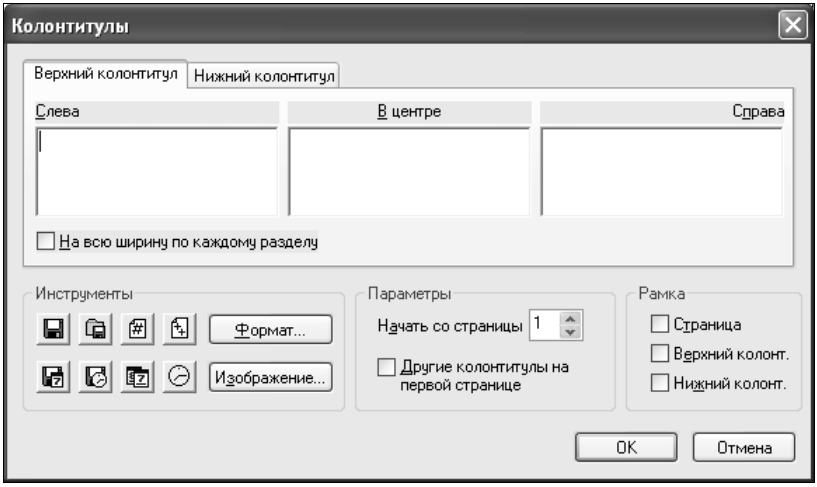


Рис. 1.47. Установка колонтитулов

1.8. Ввод/вывод данных

В большинстве случаев ввод внешних данных в документы Mathcad применяется чаще вывода, поскольку Mathcad имеет гораздо лучшие возможности представления результатов расчетов, чем многие пользовательские программы.

1.8.1. Ввод/вывод в текстовые файлы

Перечислим встроенные функции для работы с текстовыми файлами, которые имеются в Mathcad 2001—15.

(12) *Примечание*

В Mathcad 12 и выше имеется дополнительная универсальная встроенная функция `READFILE`, значительно облегчающая процесс импорта данных.

- `READPRN("file")` — чтение данных в матрицу из текстового файла;
- `WRITEPRN("file")` — запись данных в текстовый файл;
- `APPENDPRN("file")` — дозапись данных в существующий текстовый файл:
  - `file` — путь к файлу.

Примеры использования встроенных функций иллюстрируются листингами 1.28 и 1.29. Результат их действия можно понять, просмотрев получающиеся текстовые файлы, например, с помощью Блокнота Windows (рис. 1.48 и 1.49 соответственно).

Листинг 1.28. Запись матрицы в текстовый файл и чтение данных из него

$$Z := \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
WRITEPRN("datafile.prn") := Z
C := READPRN("datafile.prn")
```

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Листинг 1.29. Дозапись вектора `k` в существующий текстовый файл

```
k := (1  2  3  4  5)
APPENDPRN("datafile.prn") := k
```

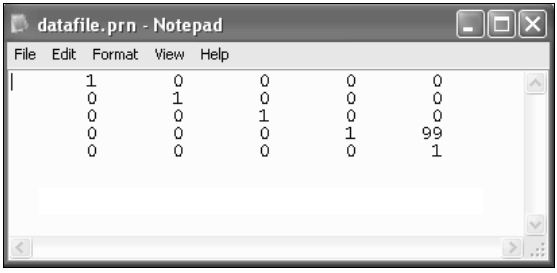


Рис. 1.48. Файл, созданный листингом 1.28

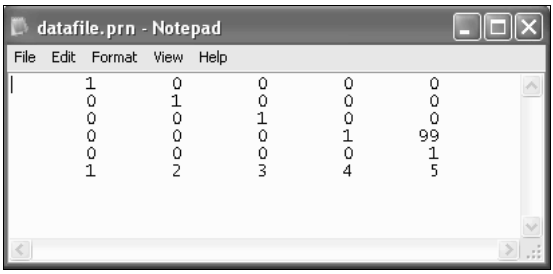


Рис. 1.49. Файл, созданный листингами 1.28 и 1.29

1.8.2. Ввод/вывод в файлы других типов

Подобно вводу/выводу в текстовые файлы можно организовать чтение и запись данных в графические и звуковые файлы.

Графические файлы

При записи и чтении числовой информации в файлы различных графических форматов данные отождествляются с интенсивностью того или иного цвета пиксела изображения, находящегося в файле. Перечислим основные встроенные функции, предназначенные для графического ввода/вывода:

- `READRGB("file")` — чтение цветного изображения;
- `READBMP("file")` — чтение изображения в оттенках серого;
- `WRITERGB("file")` — запись цветного изображения;
- `WRITEBMP("file")` — запись изображения в оттенках серого:
  - `file` — путь к файлу.

> ***Примечание***
> Имеется также большое количество функций специального доступа к графическим файлам, например, чтение интенсивности цветов в других цветовых моделях (яркость-насыщенность-оттенок), а также чтение только одного из основных цветов и т. п. Вы без труда найдете информацию об этих функциях в справочной системе Mathcad, а их применение полностью эквивалентно описанным встроенным функциям.

Действие функций доступа к графическим файлам иллюстрируется листингами 1.30—1.32. Заметим, что для создания изображения используется встроенная функция `identity`, создающая единичную матрицу. Изображение, созданное листингом 1.30, приведено на рис. 1.50.

Листинг 1.30. Запись матрицы `I` в графический файл

```
I := identity(100) · 100
I3,9 := 500
WRITEBMP("data.bmp") := I
```

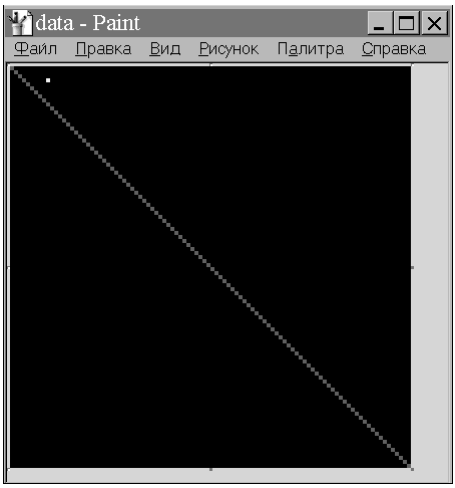


Рис. 1.50. Файл, созданный листингом 1.30

Листинг 1.31. Чтение из графического файла

```
C := READBMP("data.bmp")
```

Листинг 1.32. Запись в цветной графический файл

```
R := identity(100) · 100
G := identity(100)
B := identity(100)
WRITERGB("color.bmp") := augment(R, G, B)
```

Звуковые файлы

Для записи и считывания амплитуды акустических сигналов в звуковые файлы с расширением wav применяются следующие встроенные функции:

- `READWAV("file")` — чтение звукового файла в матрицу;
- `WRITEWAV("file",s,b)` — запись данных в звуковой файл;
- `GETWAVINFO("file")` — создает вектор из четырех элементов с информацией о звуковом файле:
  - `file` — путь к файлу;
  - `s` — скорость следования сэмплов, задаваемых матрицей;
  - `b` — разрешение звука в битах.

Использование этих встроенных функций позволяет организовать обработку звука.

⑫ 1.8.3. Универсальный импорт данных

Рассмотрим две более универсальных возможности для импорта данных из внешнего файла. Они связаны с появлением Мастера импорта данных, позволяющего осуществить импорт в нужном формате в диалоговом режиме с подсказками, а также новой встроенной функции `READFILE`, призванной унифицировать процесс импорта. Первый путь позволяет импортировать данные "вручную", проследив процесс считывания данных последовательно, шаг за шагом, а второй — автоматизировать его, не путаясь в других многочисленных функциях импорта.

Примечание

Оба способа подразумевают возможность импорта файлов данных самых разных форматов: текстовых с разнообразными символами-разделителями, а также файлы формата XLS (Microsoft Excel). В версии Mathcad 15 появилась поддержка Excel 2007.

Мастер импорта данных

Рассмотрим реализацию первой возможности на примере считывания данных из файла, представленного ранее (*см. разд. 1.8.1*):

1. Введите команду меню **Вставка / Компонент** (Insert / Component), а затем выберите в списке тип компонента **Мастер импорта данных** (Data Import Wizard). В результате появится окно Мастера, которое в пошаговом диалоговом режиме позволит осуществить считывание нужной информации (рис. 1.51).
2. Выберите в раскрывающемся списке **Формат файла** (File format) желаемый формат файла, из которого вы осуществляете импорт (рис. 1.51). Если вы затрудняетесь с его точной идентификацией, лучшим решением будет задание типа **Текст с разделителями** (Delimited Text), что позволит возложить распознавание типа данных и формат их записи на Mathcad.

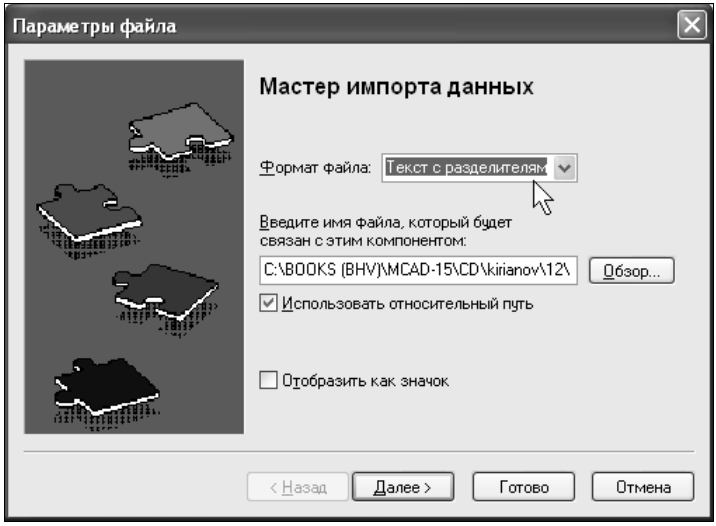


Рис. 1.51. Стартовая страница Мастера импорта данных

3. Нажмите кнопку **Обзор** (Browse) и отыщите в открывшемся диалоговом окне местоположение нужного вам файла.
4. Если вы уверены (например, основываясь на накопленном опыте работы с Мастером импорта), что данные будут считаны правильно, то можете сразу нажать кнопку **Готово** (Finish). Если вы хотите просмотреть и, при необходимости, изменить те или иные опции импорта, нажмите кнопку **Далее** (Next). В последнем случае в диалоговом режиме еще на двух страницах Мастера (подобных рис. 1.52, но с новыми параметрами) можно будет отредактировать многочисленные параметры импорта (такие как тип разделителя между данными, интервалы импорта и т. д.).
5. После нажатия кнопки **Готово** (Finish) в диалоге **Мастер импорта данных** (Data Import Wizard) и возвращения на рабочую область документа Mathcad вве-

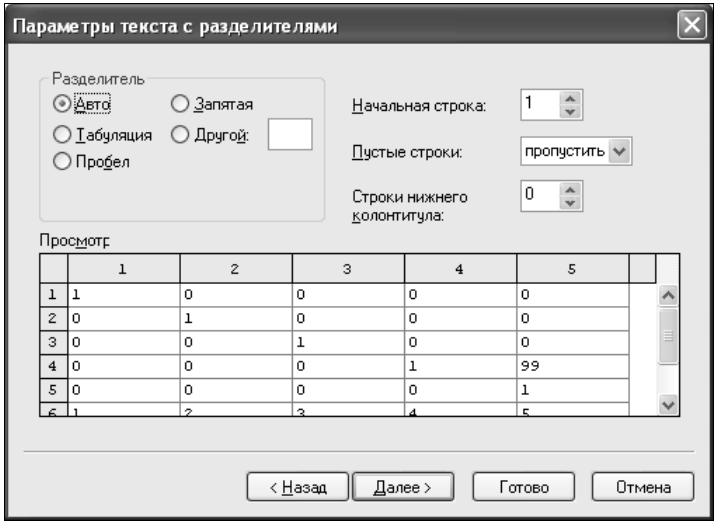
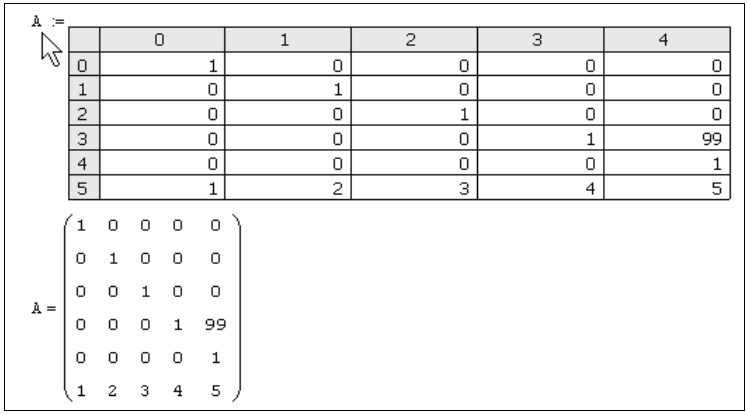


Рис. 1.52. Следующая страница Мастера импорта данных



A :=

	0	1	2	3	4
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	99
4	0	0	0	0	1
5	1	2	3	4	5

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Рис. 1.53. Результат импорта данных из файла

дите в местозаполнитель, появившийся слева от таблицы импортированных данных, желаемое имя переменной. В дальнейших расчетах ее можно будет использовать как обычную матрицу.

Итог работы Мастера показан на рис. 1.53. Его первая строка является результатом описанных шагов по считыванию данных в матрицу, а вторая строка показывает вывод этой матрицы в стандартной для Mathcad форме.

Функция *READFILE*

Функция `READFILE` облегчает процесс "программного" считывания данных из файла (листинг 1.33):

- `READFILE ("file","type",[colwidth,rows,cols,emptyfill])` — возвращает матрицу с элементами, считанными из внешнего файла данных:
  - `"file"` — название файла (включая путь к нему на диске);
  - `"type"` — тип файла (`"delimited"` или `"Excel"`);
  - `colwidth` — ширина столбца данных, считываемого из файла в случае выбора в качестве предыдущего параметра типа `"fixed"`, т. е. с фиксированной шириной данных;
  - `rows` — начальная строка импорта данных или двухкомпонентный вектор, задающий интервал импорта строк;
  - `cols` — начальный столбец импорта данных или двухкомпонентный вектор, задающий интервал импорта столбцов;
  - `emptyfill` — значение, которое будет использовано для замены отсутствующих данных (пустот в файле). Для него можно использовать значение Не-Число (NaN).

Листинг 1.33. Импорт данных при помощи универсальной функции `READFILE`

$$A := \mathrm{READFILE}(\text{"datafile.txt"}, \text{"delimited"})$$

$$B := \mathrm{READFILE}\left[\text{"datafile.txt"}, \text{"delimited"}, \begin{pmatrix} 4 \\ 6 \end{pmatrix}, \begin{pmatrix} 4 \\ 5 \end{pmatrix}\right]$$

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 99 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 99 \\ 0 & 1 \\ 4 & 5 \end{pmatrix}$$

ГЛАВА 2

Алгебраические вычисления

В данной главе рассматриваются простые вычисления, осуществляемые в Mathcad. Во-первых, приведено описание имеющихся встроенных операторов и функций (*см. разд. 2.1*), при помощи которых можно рассчитать значение алгебраических выражений, построить графики и т. п. Во-вторых, составлен обзор наиболее простых символьных операций, реализующих в Mathcad аналитические преобразования для решения типичных задач алгебры. Они проводятся без применения численных методов и, соответственно, без погрешностей вычислений (*см. разд. 2.2*).

2.1. Операторы и функции

Каждый оператор в Mathcad обозначает некоторое математическое действие в виде символа. В полном согласии с терминологией, принятой в математике, ряд действий (например, сложение, деление, транспонирование матрицы и т. п.) реализован в Mathcad в виде встроенных операторов, а другие действия (например, `sin`, `erf` и т. п.) — в виде встроенных функций. Каждый оператор действует на одно или два числа (переменную или функцию), которые называют *операндами*. Если в момент вставки оператора одного или обоих операндов не хватает, то недостающие операнды будут отображены в виде местозаполнителей. Символ любого оператора в нужное место документа вводится одним из двух основных способов:

- нажатием соответствующей клавиши (или сочетания клавиш) на клавиатуре;
- нажатием указателем мыши соответствующей кнопки на одной из математических панелей инструментов.

Напомним, что большинство математических панелей содержат сгруппированные по смыслу математические операторы, а вызвать эти панели на экран можно нажатием соответствующей кнопки на панели **Математические** (Math).

Выше мы рассмотрели особенности применения трех операторов: присваивания (*см. разд. 1.2.3*), численного и символьного вывода (*см. разд. 1.2.1*). Разберем в данном разделе более подробно действие этих и других операторов Mathcad, а также напомним, что более сложные действия организованы в Mathcad посредством *функций*.

2.1.1. Арифметические операторы

Операторы, обозначающие основные арифметические действия, вводятся с панели **Калькулятор** (Calculator), показанной на рис. 2.1:

- сложение и вычитание: + / –;
- умножение и деление: · / ÷ ;
- факториал: **!**;
- модуль числа: |**x**|;
- квадратный корень: $\sqrt{\ }$;
- корень *n*-й степени: $\sqrt[n]{\ }$;
- возведение x в степень y: $\mathbf{x^y}$;
- изменение приоритета: скобки;
- численный вывод: = (все листинги).

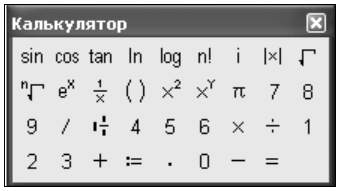


Рис. 2.1. Калькулятор

2.1.2. Вычислительные операторы

Вычислительные операторы вставляются в документы при помощи панели инструментов **Calculus** (Математический анализ) (рис. 2.2). При нажатии любой из кнопок в документе появляется символ соответствующего математического действия, снабженный несколькими местозаполнителями. Количество и расположение местозаполнителей определяется типом оператора и в точности соответствует их общепринятой математической записи. Например, при вставке оператора суммы (см. рис. 2.2) необходимо задать четыре величины: переменную, по которой надо произвести суммирование, нижний и верхний пределы, а также само выражение, которое будет стоять под знаком суммы. Для того чтобы вычислить неопределенный интеграл, следует заполнить два местозаполнителя: подынтегрального выражения и переменной интегрирования и т. д.

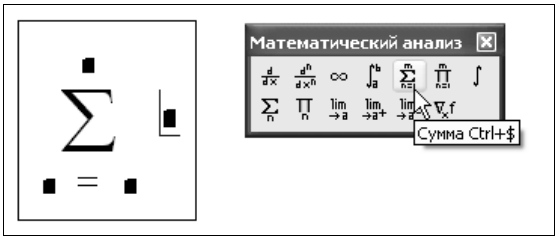


Рис. 2.2. Вставка оператора суммы

После ввода какого-либо вычислительного оператора имеется возможность вычислить его значение либо численно, нажатием клавиши <=>, либо аналитически, с помощью оператора символьного вывода.

Примечание

Дифференцированию и интегрированию, как более сложным операциям, посвящены отдельные главы этой книги (*см. главу 3 и 4 соответственно*). Суммирование и вычисление предела рассматривается ниже в этой главе (*см. соответственно разд. 2.2.14 и 2.2.16*).

2.1.3. Логические операторы

Результатом действия логических, или булевых, операторов являются только числа 1 (если логическое выражение, записанное с их помощью, истинно) или 0 (если логическое выражение ложно). Чтобы вычислить значение логического выражения, например `1=1` (рис. 2.3):

1. Вставьте с панели **Булева алгебра** (Boolean) соответствующий оператор **=**.
2. В появившиеся местозаполнители вставьте операнды (две единицы).
3. Нажмите клавишу <=>, чтобы получить ответ.

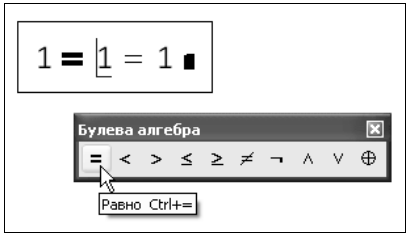


Рис. 2.3. Вставка логического оператора

Получается абсурдное на первый взгляд выражение `1=1=1`. Однако на самом деле все правильно. Слева от оператора вывода записано логическое выражение `1=1` (обратите внимание, что логический знак равенства выглядит по-другому, нежели обычный), которое является истинным. Поэтому значение данного выражения равно 1, что и показано справа от знака равенства.

Перечислим логические операторы:

- **Больше** (Greater Than) `x>y`;
- **Меньше** (Less Than) `x<y`;
- **Больше или равно** (Greater Than or Equal) `x≥y`;
- **Меньше или равно** (Less Than or Equal) `x≤y`;
- **Равно** (Equal) `x=y`;
- **Не равно** (Not Equal to) `x≠y`;
- **И** (And) `x∧y`;
- **Или** (Or) `x∨y`;
- **Исключающее или** (Exclusive or) `x⊕y`;
- **Отрицание** (Not) `¬x`.

Примечание

Операнды в логических выражениях могут быть любыми числами. Однако если оператор по смыслу применим только к `0` и `1`, то любое неравное нулю число по умолчанию принима-

ется равным 1. Но в результате все равно может получиться либо 0, либо 1. Например, ¬(-0.33)=0.

Примеры действия логических операторов приведены в листингах 2.1 и 2.2.

Примечание

Логические операторы чрезвычайно важны при записи в редакторе Mathcad подлежащих решению алгебраических уравнений и неравенств (*см. главу 5*).

Листинг 2.1. Операторы сравнения

$2 = 3 = 0$ $\quad 5 > 1 = 1$ $\quad 3 > 3 = 0$

$7 = 7 = 1$ $\quad 3 < \infty = 1$ $\quad 3 \geq 3 = 1$

$0 \neq 0 = 0$

Листинг 2.2. Булевы операторы

$1 \vee 0 = 1$ $\quad 1 \wedge 0 = 0$ $\quad 1 \oplus 0 = 1$ $\quad \neg 1 = 0$

$0 \vee 0 = 0$ $\quad 0 \wedge 0 = 0$ $\quad 0 \oplus 0 = 0$ $\quad \neg 0 = 1$

$1 \vee 1 = 1$ $\quad 1 \wedge 1 = 1$ $\quad 1 \oplus 1 = 0$

2.1.4. Матричные операторы

Матричные операторы предназначены для совершения различных действий над векторами и матрицами. Поскольку большинство из них реализует численные алгоритмы, о них будет подробно рассказано в разделах, посвященных линейной алгебре (*см. главу 7*). Остановимся в данном разделе лишь на вопросе вставки матриц в документы Mathcad.

Самый простой и наглядный способ создания вектора или матрицы заключается в следующем:

1. Нажмите кнопку **Матрица** (Matrix) или **Вектор** (Vector) на панели **Матрица** (Matrix), либо клавиши <Ctrl>+<M>, либо выберите пункт меню **Вставка / Матрица** (Insert / Matrix).
2. В диалоговом окне **Вставка матрицы** (Insert Matrix) (рис. 2.4) задайте целое число столбцов и строк матрицы, которую хотите создать. Например, для создания вектора 3×1 введите показанные на рис. 2.4 значения.
3. Нажмите кнопку **OK** или **Вставка** (Insert) — в результате в документ будет вставлена заготовка матрицы с определенным числом строк и столбцов (рис. 2.5).
4. Введите значения в местозаполнители элементов матрицы. Переходить от одного элемента матрицы к другому можно с помощью указателя мыши либо клавиш со стрелками.

Добавление в уже созданную матрицу строк или столбцов производится точно так же:

1. Выделите линиями ввода элемент матрицы, правее и ниже которого будет осуществлена вставка столбцов и (или) строк.

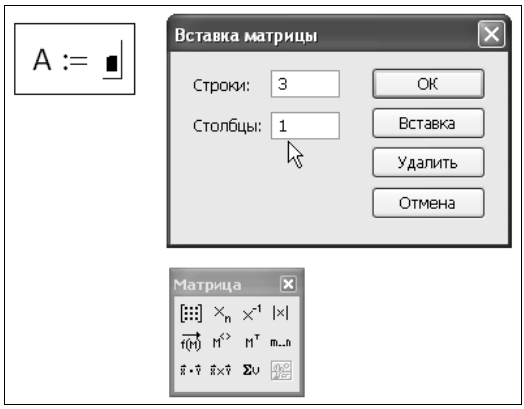


Рис. 2.4. Создание матрицы

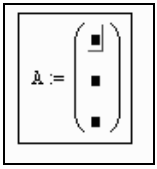


Рис. 2.5. Результат создания матрицы

2. Вставьте в него матрицу, как было описано выше. При этом допускается задание числа столбцов или строк равным нулю.
3. Заполните местозаполнители недостающих элементов матрицы.

Примечание

Определить новую матрицу можно и по-другому, например (впервые в документе), присвоив определенное значение ее некоторому элементу, например $A_{3,3}:=1$, либо импортировав в матрицу данные из внешнего файла.

2.1.5. Операторы вычисления

Вычислительные операторы сгруппированы на панели **Вычисление** (Evaluation) (*см. разд. 1.2.1*). Перечислим их еще раз (без дополнительных комментариев):

- *численный* вывод (Evaluate Numerically) = ;
- *символьный* (аналитический) вывод (Evaluate Symbolically) → ;
- *определение*, или присваивание (Definition) **:=** ;
- *глобальное определение* (Global Definition) ≡.

2.1.6. Функции

Mathcad содержит огромное количество встроенных функций. Некоторые из них просто рассчитывают определенное значение, а некоторые реализуют сложные численные алгоритмы.

Для вставки функции в документ можно как набрать ее имя и список аргументов вручную, так и использовать диалоговое окно **Вставка функции** (Insert Function), которое вызывается нажатием кнопки с надписью ***f(x)*** на стандартной панели инструментов (рис. 2.6).

Функции, в отличие от операторов, всегда записываются в формате некоторого уникального имени, после которого (в скобках) следует список аргументов (см. рис. 2.6). Весьма условно (согласно математической терминологии) функции

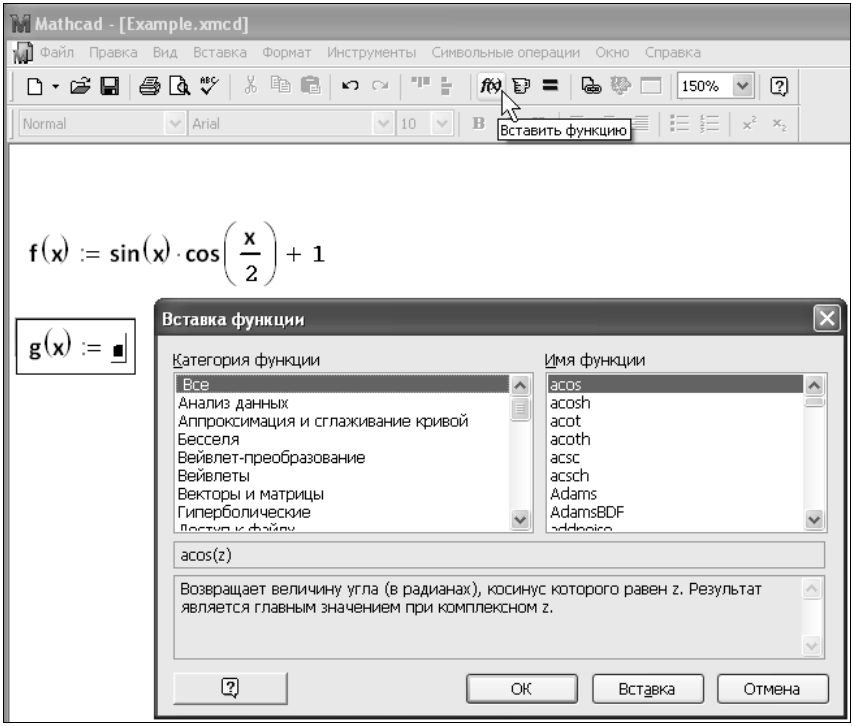


Рис. 2.6. Вставка функции в документ

Mathcad можно разделить на элементарные и специальные, добавив к ним еще один класс служебных функций, которые играют в расчетах вспомогательные роли.

2.2. Алгебраические преобразования

В этом разделе речь пойдет об алгебраических вычислениях, которые выполняются в Mathcad главным образом аналитически. Как ни странно, многие пользователи Mathcad не очень хорошо осведомлены об этих возможностях, тогда как они во многих ситуациях могут существенно сэкономить их время и силы по выполнению несложных, но рутинных преобразований.

2.2.1. О способах символьных вычислений

Символьные вычисления в Mathcad можно осуществлять в двух различных вариантах:

- с помощью команд меню;
- с помощью оператора символьного вывода →, ключевых слов символьного процессора и обычных формул (в справочной системе Mathcad этот способ называется *символьными вычислениями в реальном времени* — live symbolic evaluation).

Первый способ более удобен, когда требуется быстро получить какой-либо аналитический результат для однократного использования, не сохраняя сам ход вычис-

лений. Второй способ более нагляден, т. к. позволяет записывать выражения в традиционной математической форме и сохранять символьные вычисления в документах Mathcad. Кроме того, аналитические преобразования, проводимые через меню, касаются только одного, выделенного в данный момент, выражения.

Примечание

В символьных вычислениях допускается использование большинства встроенных функций Mathcad, конечно, за исключением тех, которые реализуют численные методы.

Символьный процессор Mathcad умеет выполнять основные алгебраические преобразования, такие как упрощение выражений, разложение их на множители, символьное суммирование и перемножение.

Для символьных вычислений при помощи команд предназначено главное меню **Символьные операции** (Symbolics), объединяющее математические операции, которые Mathcad умеет выполнять аналитически. Для реализации второго способа применяются все средства Mathcad, пригодные для численных вычислений, например, панели **Калькулятор** (Calculator), **Вычисление** (Evaluation) и т. д.), и специальная математическая панель инструментов, которую можно вызвать на экран нажатием кнопки **Панель символьных преобразований с ключевыми словами** (Symbolic Keyword Toolbar) на панели **Математические** (Math). На панели **Символьные** (Symbolic) находятся кнопки, соответствующие специфическим командам символьных преобразований (рис. 2.7). Например, таким как разложение выражения на множители, приведение подобных слагаемых и другим операциям, которые в Mathcad нельзя проводить численно и для которых, соответственно, не предусмотрены встроенные функции.

Помимо панели **Символьные** (Symbolic), для задания опций тех или иных аналитических преобразований применяется вспомогательная панель **Модификатор** (Modifier). С ее помощью, например, можно сузить круг преобразования, ограничившись действительными или комплексными числами.

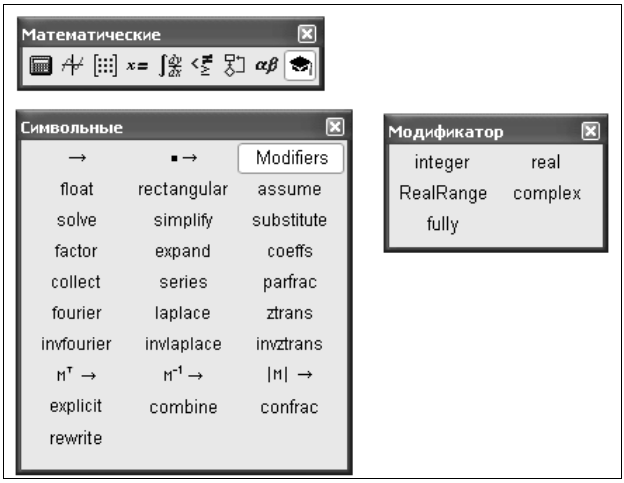


Рис. 2.7. Панели **Символьные** (Symbolic) и **Модификатор** (Modifier)

Примечание

Необходимо отметить, что приемы более сложных символьных вычислений описываются также в остальных главах данной книги, в рамках рассказа о решении конкретных вычислительных задач.

2.2.2. Разложение выражений

Рассмотрим оба типа символьных вычислений на простом примере разложения на сомножители выражения `cos(4·x)`. В ходе операции символьного разложения, или *расширения*, раскрываются все суммы и произведения, а сложные тригонометрические зависимости разлагаются с помощью тригонометрических тождеств. Разложение выражений производится путем выбора команды **Символьные операции / Разложить** (Symbolics / Expand) либо использованием вместе с оператором символьного вывода ключевого слова `expand`.

Первый способ (разложение с помощью меню)

1. Введите выражение `cos(4·x)`.
2. Выделите его целиком (рис. 2.8).
3. Выберите в главном меню пункты **Символьные операции / Развернуть** (Symbolics / Expand).

После этого результат разложения выражения появится чуть ниже в виде еще одной строки (см. рис. 2.8).

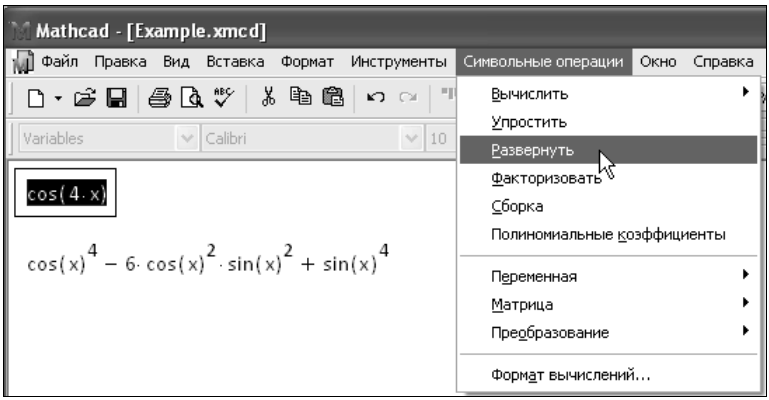


Рис. 2.8. Разложение выражения на множители при помощи команды меню **Символьные операции / Развернуть** (Symbolics / Expand)

Внимание!

Символьные операции с помощью меню возможны лишь над каким-либо объектом (выражением, его частью или отдельной переменной). Для того чтобы правильно осуществить желаемое аналитическое преобразование, предварительно необходимо выделить тот объект, к которому оно будет относиться. В данном случае преобразование было применено ко всему выражению `cos(4·x)`. Если же выделить часть формулы, как показано на рис. 2.9, то соответствующее преобразование будет отнесено только к выделенной части (нижняя строка на этом рисунке).

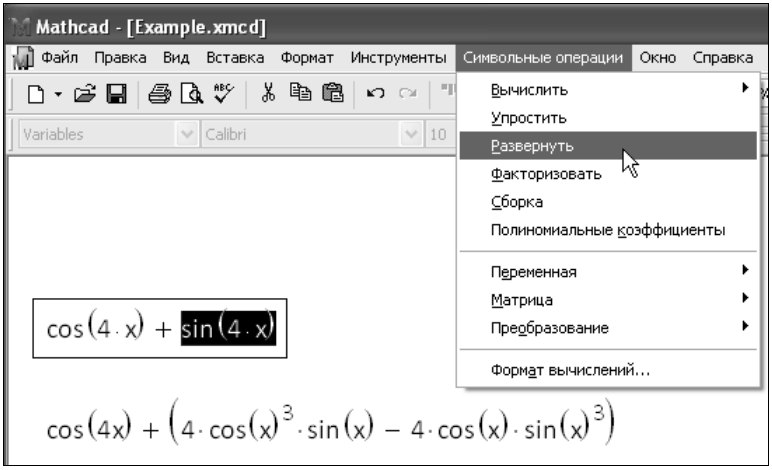


Рис. 2.9. Символьное разложение части выражения и его результат

Второй способ (разложение с помощью оператора →)

1. Введите выражение, например `cos(4·x)`.
2. Нажмите кнопку **Разложить** (Expand) на панели **Символьные** (Symbolic).
3. Если вы собираетесь разложить выражение по определенной переменной, то нажмите после ключевого слова `expand` запятую и введите в появившийся местозаполнитель имя переменной (например, `x`). Соответствующий пример приведен в листинге 2.3.
4. Нажмите клавишу <Enter> либо просто щелкните мышью за пределами выражения (рис. 2.10).

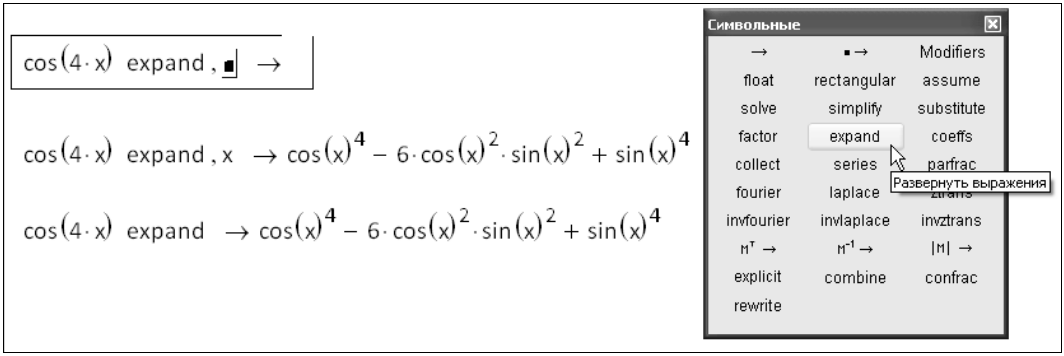


Рис. 2.10. Символьное разложение выражения

Примечание

Можно действовать и в другом порядке: сначала ввести ключевое слово, а уже затем впечатать в появившиеся местозаполнители выражение и переменную (рис. 2.11).

Оператор символьного вывода, как вы помните, можно ввести в редакторе Mathcad несколькими способами: нажатием кнопки → на любой из панелей **Вычисление**

(Evaluation) или **Символьные** (Symbolic) либо сочетанием клавиш <Ctrl>+<.>. Результат символьного разложения выражения показан на том же рис. 2.10.

Совет

Если вы можете выбрать способ символьных вычислений, рекомендую второй путь — с помощью оператора →, поскольку при этом в документе сохраняются действия пользователя. Наличие специального меню символьных вычислений — своего рода дань очень давним версиям Mathcad. В них аналитические преобразования были встроены не так гармонично и были доступны, главным образом, через меню.

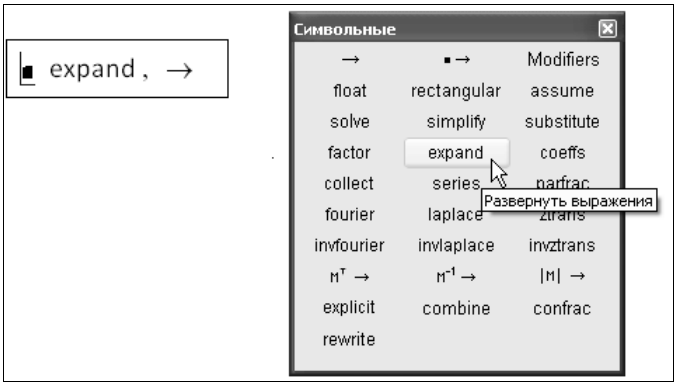


Рис. 2.11. Результат первоначального ввода ключевого слова

Не всякое выражение поддается аналитическим преобразованиям. Если это так (либо в силу того, что задача вовсе не имеет аналитического решения, либо она оказывается слишком сложной для символьного процессора Mathcad), то в качестве результата выводится само выражение (листинг 2.3, внизу).

Листинг 2.3. Символьные преобразования

$\cos(2\cdot x)$ expand, x $\rightarrow 2\cdot\cos(x)^2 - 1$

$\cos(x)$ expand, x $\rightarrow \cos(x)$

Примечание

Далее в этой главе, рассматривая символьные вычисления с помощью меню, будем иллюстрировать результаты рисунками, а символьные вычисления с применением оператора → приводить в виде листингов.

⑭ 2.2.3. Объединение выражений

Операция, обратная символьному разложению выражений, — это их объединение, которое осуществляется посредством ключевого слова `combine`. С его помощью можно объединить степени переменных, скомбинировать тригонометрические, логарифмические или экспоненциальные функции, как это показано в листинге 2.4.

Принцип комбинации указывается путем задания соответствующего модификатора, который отделяется от имени ключевого слова `combine` запятой.

Листинг 2.4. Объединение выражений

$z^{x}\cdot z^{2y}$ combine $\rightarrow z^{x+2\cdot y}$

$e^{x}\cdot e^{2y}$ combine , exp $\rightarrow e^{x+2\cdot y}$

$\cos(x)^{4}-6\cdot\cos(x)^{2}\cdot\sin(x)^{2}+\sin(x)^{4}$ combine , sincos $\rightarrow \cos(4\cdot x)$

$\ln(x)+4\ln(2)+7\ln\left(\frac{1}{2}\right)$ combine , ln $\rightarrow \ln\left(\frac{x}{8}\right)$

2.2.4. Упрощение выражений

Упрощение выражений — очень часто применяемая операция, также противоположная по смыслу операции разложения, рассмотренной в *разд. 2.2.2*. Символьный процессор Mathcad стремится так преобразовать выражение, чтобы оно приобрело более простую форму. При этом используются различные арифметические формулы, приведение подобных слагаемых, тригонометрические тождества, пересчет обратных функций и др. Чтобы упростить выражение с помощью меню (рис. 2.12):

1. Введите выражение.
2. Выделите выражение целиком или его часть, которую нужно упростить.
3. Выберите команду **Символьные операции / Упростить** (Symbolics / Simplify).

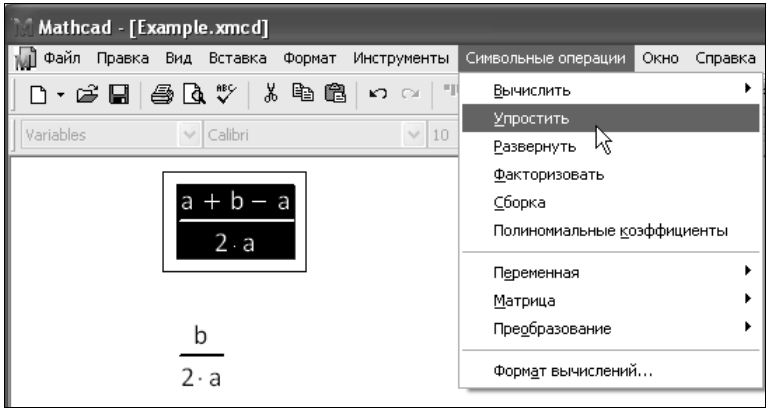


Рис. 2.12. Упрощение выражения

Для упрощения выражения при помощи оператора символьного вывода используйте ключевое слово `simplify` (листинг 2.5). Не забывайте, если некоторым переменным, входящим в выражение, ранее были присвоены некоторые значения, то они будут подставлены в него при выполнении символьного вывода (листинг 2.6).

Листинг 2.5. Упрощение выражения

$$\frac{a+b-a}{2\cdot a}\ \text{simplify}\ \rightarrow \frac{1}{2}\cdot\frac{b}{a}$$

$$\frac{a+b-a}{2\cdot a}\rightarrow \frac{1}{2}\cdot\frac{b}{a}$$

Листинг 2.6. Упрощение выражения с подстановкой значения переменных

$a := 5 \qquad b := 10$

$$\frac{a+b-a}{2\cdot a}\ \text{simplify}\ \rightarrow 1$$

$$\frac{a+b-a}{2\cdot a}\rightarrow 1$$

Упрощение выражений, содержащих числа, производится по-разному, в зависимости от наличия в числах десятичной точки. Если она есть, то выполняется непосредственное вычисление выражения (листинг 2.7).

Листинг 2.7. Упрощение выражения с числами

$$\sqrt{3.01}\ \text{simplify}\ \rightarrow 1.7349351572897472412$$

$$\text{acos}(0)\ \text{simplify}\ \rightarrow \frac{1}{2}\cdot\pi$$

2.2.5. Разложение на множители

Разложение выражений на простые множители производится при помощи команды **Символьные операции / Факторизовать** (Symbolics / Factor) (рис. 2.13) либо использованием вместе с оператором символьного вывода ключевого слова `factor` (листинг 2.8). Эта операция позволяет разложить полиномы на произведение более простых полиномов, а целые числа — на простые сомножители. Применяя команду меню, не забывайте перед ее вызовом выделить все выражение или его часть, которую планируете разложить на множители.

Листинг 2.8. Примеры разложения на множители

$$x^4-16\ \text{factor}\ \rightarrow (x-2)\cdot(x+2)\cdot\left(x^2+4\right)$$

$$28\ \text{factor}\ \rightarrow 2^2\cdot 7$$

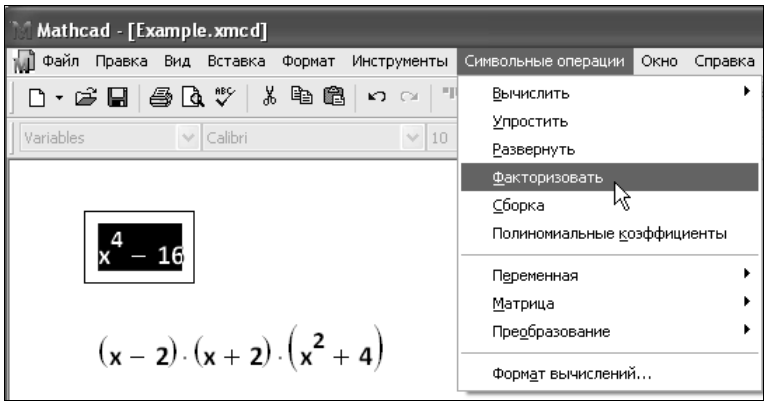


Рис. 2.13. Разложение выражения на множители

⑭ 2.2.6. Разложение на элементарные функции

В Mathcad 14 появилось ключевое слово `rewrite`, которое, будучи использовано с модификатором, позволяет аналитически выразить формулу через определенные в модификаторе элементарные функции (листинг 2.9).

Листинг 2.9. Пример разложения выражения на элементарные функции

$$\cos(x)\ \text{rewrite}, \exp \rightarrow \frac{e^{x\cdot i\cdot(-1)}}{2} + \frac{e^{x\cdot i}}{2}$$

$$\frac{e^{x\cdot i\cdot(-1)}}{2} + \frac{e^{x\cdot i}}{2}\ \text{rewrite}, \text{sincos} \rightarrow \cos(x)$$

⑭ 2.2.7. Разложение числа на простые числа

Начиная с версии Mathcad 14, при помощи символьного процессора несложно выяснить, является ли то или иное целое число простым (листинг 2.10), а также осуществить его разложение на простые числа (листинг 2.11).

Листинг 2.10. Тест простых чисел

```
IsPrime(191) → 1
IsPrime(15) → 0
IsPrime(7654) → 0
```

Листинг 2.11. Разложение целого числа на простые числа

```
7654 factor → 2·43·89
191 factor → 191
```

2.2.8. Разложение на простые дроби

Чтобы разложить сложную дробь на более простые дроби, следует либо выполнить команду **Символьные операции / Переменная / Преобразовать к дробно-рациональному виду** (Symbolics / Variable / Convert to Partial Fractions), либо указать ключевое слово `parfrac` (листинг 2.12). Применяя первый способ (меню), не забывайте перед выбором его команды выделить переменную, по которой будет производиться разложение, а если используется второй способ (с оператором символьного вывода), то имя переменной следует указать после ключевого слова `parfrac`.

Листинг 2.12. Разложение на элементарные дроби

$$\frac{11\cdot x^2+9\cdot x+1}{x^2-3\cdot x+2}\ \text{convert}\,,\text{parfrac}\,,x\ \rightarrow 11-\frac{21}{(x-1)}+\frac{63}{(x-2)}$$

⑭ 2.2.9. Разложение на непрерывные дроби

Способом, аналогичным представленному в предыдущем разделе, можно разложить число на непрерывные дроби, пользуясь ключевым словом `confrac`. Листинг 2.13 представляет пример такого разложения и раскрывает его смысл.

Листинг 2.13. Разложение выражения на непрерывные дроби

$$\frac{2+\sqrt{5}}{19}\ \text{confrac}\ \rightarrow \begin{pmatrix} 0 \\ 4 \\ 2 \\ 16 \\ 2 \\ 84 \\ 2 \end{pmatrix}$$

$$\frac{2+\sqrt{5}}{19}\ \text{confrac}\,,\text{fraction}\ \rightarrow \frac{1}{\left[4+\frac{1}{\left[2+\frac{1}{\left[16+\frac{1}{\left[2+\frac{1}{\left(84+\frac{1}{2}\right)}\right]}\right]}\right]}\right]}$$

2.2.10. Приведение подобных слагаемых

Чтобы привести подобные слагаемые полинома с помощью меню (рис. 2.14):

1. Введите выражение.
2. Выделите в выражении имя переменной, относительно которой надо привести подобные слагаемые (в примере на рис. 2.14 это переменная `y`).
3. Выберите команду **Символьные операции / Сборка** (Symbolics / Collect).

В результате появится строка с результатом приведения подобных слагаемых (нижняя строка на рис. 2.14).

Чтобы привести подобные слагаемые с помощью оператора символьного вывода (листинг 2.14):

1. Введите выражение.
2. Нажмите кнопку **Collect** на панели **Символьные** (Symbolic).
3. Введите в местозаполнитель после вставленного ключевого слова `collect` имя переменной, относительно которой требуется привести подобные слагаемые (в первой строке примера из листинга 2.14 это переменная `x`, во второй — `y`).
4. Введите оператор символьного вывода →.
5. Нажмите клавишу <Enter>.

Примечание

После ключевого слова `collect` допускается задание нескольких переменных через запятую. В этом случае, что иллюстрируется последней строкой листинга 2.14, приведение подобных слагаемых выполняется последовательно по всем переменным.

Листинг 2.14. Приведение подобных слагаемых по разным переменным

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ collect}, x \rightarrow \left(z-z^2\cdot y\right)\cdot x+2\cdot z\cdot y-5\cdot z^2\cdot y^2+z$$

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ collect}, y \rightarrow -5\cdot z^2\cdot y^2+\left(2\cdot z-z^2\cdot x\right)\cdot y+z\cdot x+z$$

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ collect}, x, y, z \rightarrow \left(z-z^2\cdot y\right)\cdot x+2\cdot z\cdot y-5\cdot z^2\cdot y^2+z$$

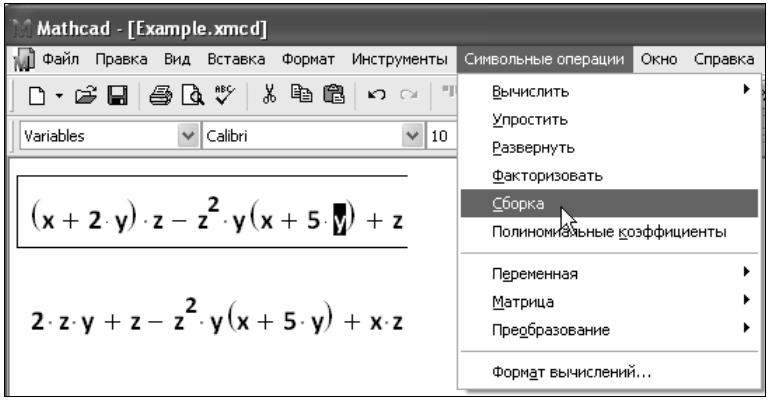


Рис. 2.14. Приведение подобных слагаемых

2.2.11. Вычисление коэффициентов полинома

Если выражение является полиномом относительно некоторой переменной x, заданным не в обычном виде $a_0+a_1x+a_2x^2+\ldots$, а как произведение других, более простых полиномов, то коэффициенты $a_0, a_1, a_2\ldots$ легко определяются символьным процессором Mathcad. Коэффициенты сами могут быть функциями (подчас довольно сложными) других переменных.

Чтобы вычислить полиномиальные коэффициенты в выражении при помощи меню:

1. Введите выражение.
2. Выделите в нем имя переменной или выражение, для которого требуется рассчитать полиномиальные коэффициенты.
3. Выполните команду **Символьные операции / Полиномиальные коэффициенты** (Symbolic / Polynomial Coefficients).

В результате под выражением появится вектор, состоящий из полиномиальных коэффициентов. Первым элементом вектора является свободный член a_0, вторым — a_1 и т. д.

> ***Примечание***
>
> Конкретная задача, требующая вычисления полиномиальных коэффициентов, приведена в разделе, посвященном численному отделению корней полинома (*см. разд. 5.2.3*).

Чтобы вычислить полиномиальные коэффициенты с помощью оператора символьного вывода:

1. Введите выражение.
2. Нажмите кнопку **Coeffs** на панели **Символьные** (Symbolic).
3. Введите в местозаполнитель после вставленного ключевого слова `coeffs` аргумент полинома.
4. Введите оператор символьного вывода **→**.
5. Нажмите клавишу <Enter>.

Примеры вычисления коэффициентов полинома приведены в листингах 2.15 и 2.16. Листинг 2.15 показывает расчет коэффициентов для разных аргументов. Листинг 2.16 демонстрирует возможность определения коэффициентов не только для отдельных переменных, но для более сложных выражений, входящих в рассматриваемую формулу в качестве составной части.

Листинг 2.15. Вычисление коэффициентов полинома

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ coeffs}, z \rightarrow \begin{pmatrix} 0 \\ x+2\cdot y+1 \\ -y\cdot x-5\cdot y^2 \end{pmatrix}$$

$$(x+2\cdot y)\cdot z-z^2\cdot y\cdot(x+5\cdot y)+z \text{ coeffs}, x \rightarrow \begin{pmatrix} 2\cdot z\cdot y-5\cdot z^2\cdot y^2+z \\ z-z^2\cdot y \end{pmatrix}$$

Листинг 2.16. Вычисление полиномиальных коэффициентов для простой переменной и выражения

$$(x-4)\cdot(x-7)\cdot x+99 \text{ coeffs}, x \rightarrow \begin{pmatrix} 99 \\ 28 \\ -11 \\ 1 \end{pmatrix}$$

$$(x-4)^3+(x-4)\cdot(x-7)\cdot x+99 \text{ coeffs}, x-4 \rightarrow \begin{pmatrix} 99 \\ x^2-7\cdot x \\ 0 \\ 1 \end{pmatrix}$$

2.2.12. Получение численного значения выражения

С помощью символьного процессора можно рассчитать численное значение выражения (действительное или комплексное). Иногда такой путь представляется более удобным, чем применение численного процессора (т. е. знака обычного равенства). Чтобы рассчитать значение некоторого выражения (рис. 2.15), выберите команду **Символьные операции / Вычислить / Аналитически** (Symbolics / Evaluate / Symbolically) либо пункт **Символьные операции / Вычислить / С плавающей точкой** (Symbolics / Evaluate / Floating Point). В последнем случае вам будет предложено с помощью диалога **Расчеты с плавающей точкой** (Floating Point Evaluation) задать точность вывода. В итоге применения данных команд Mathcad заменяет символьные результаты, где это возможно, значениями в виде чисел с плавающей точкой.

Еще один пункт меню **Символьные операции / Вычислить / В комплексных числах** (Symbolics / Evaluate / Complex) позволяет представить выражение в виде `a+b·i`.

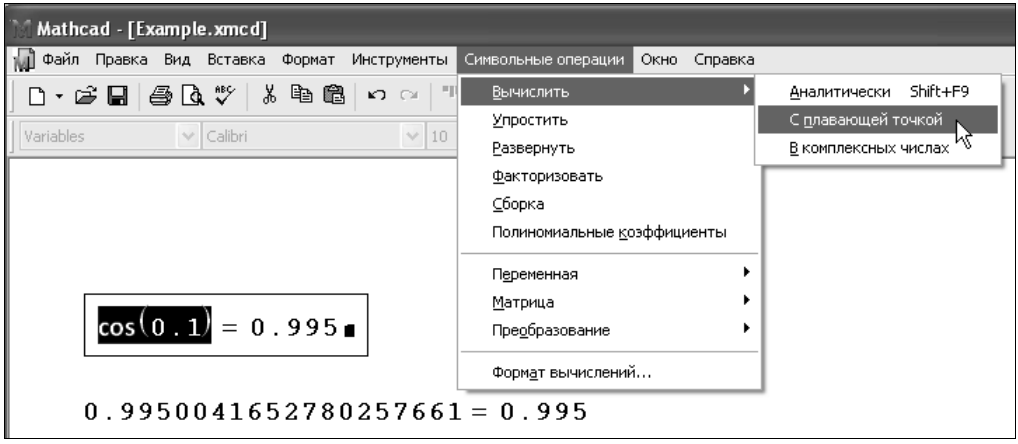


Рис. 2.15. Вычисление выражения с плавающей точкой

Аналогичные по действию ключевые слова `float` и `complex` можно использовать в документах, вводя их с панели **Символьные** (Symbolic). Ключевое слово `float` применяется вместе со значением точности вывода результата с плавающей точкой (листинг 2.17). С помощью слова `complex` можно преобразовывать выражения как в символьном виде, так и с учетом численных значений, если они были ранее присвоены переменным (несколько примеров приведено в листинге 2.18).

Листинг 2.17. Вычисление выражения с плавающей точкой

$x := 3 \qquad k := 2.4$

$\cos(k \cdot x) + 4 \cdot x^{2-k}$ float, 3 $\rightarrow 3.19$

$\cos(k \cdot x) + 4 \cdot x^{2-k}$ float, 10 $\rightarrow 3.185927374$

$\cos(k \cdot x) + 4 \cdot x^{2-k}$ float, 20 $\rightarrow 3.1859273744412716730$

Листинг 2.18. Комплексные преобразования выражений

e^{z+2i} complex $\rightarrow \exp(z) \cdot \cos(2) + i \cdot \exp(z) \cdot \sin(2)$

$4.2 \cdot 2i^{1.8-3i}$ complex $\rightarrow 1193.4523970930846183 + 1107.3477730509390980 \cdot i$

$x := i$

$4 \cdot x^3$ complex $\rightarrow -4 \cdot i$

$4 \cdot x^{3.1}$ complex $\rightarrow .62573786016092347604 - 3.9507533623805509048 \cdot i$

⑬ 2.2.13. Явные вычисления

Термином *явных* (explicit) символьных вычислений разработчики обозначили расчеты, в которых осуществляется простая подстановка в выражение численного значения той или иной переменной (без упрощения результата). Для организации явных вычислений предусмотрено специальное ключевое слово `explicit`, которое вводится с панели **Символьные** (Symbolic). Сама формула вводится в левый местозаполнитель, а переменные, числовые значения которых должны быть подставлены в нее явно и без упрощения, — в правый местозаполнитель.

Пример вычислений, включающих ключевое слово `explicit`, приведен в листинге 2.19.

Листинг 2.19. Явные вычисления

$x := 2 \cdot m \qquad y := 0.5 \cdot sec$

$z := 1.5 \cdot \frac{m}{sec}$

$\frac{x}{y} + z$ explicit, x $\rightarrow \frac{2 \cdot m}{y} + z$

$$\frac{x}{y} + z \text{ explicit}, x, y, z \rightarrow \frac{2 \cdot m}{0.5 \cdot sec} + 1.5 \cdot \frac{m}{sec}$$

$$\frac{x}{y} + z \text{ explicit}, z, y \rightarrow \frac{x}{0.5 \cdot sec} + 1.5 \cdot \frac{m}{sec}$$

2.2.14. Вычисление рядов и произведений

Чтобы вычислить аналитически конечную или бесконечную сумму или произведение:

1. Введите выражение, используя панель **Математический анализ** (Calculus) для вставки соответствующих символов суммирования или произведения. При необходимости введите в качестве предела ряда символ бесконечности (клавиши <Ctrl>+<Shift>+<Z>).
2. В зависимости от желаемого стиля символьных вычислений выберите команду **Символьные операции / Упростить** (Symbolics / Simplify) или введите оператор символьного вывода **→**.

Примеры численного и символьного вычисления рядов и произведений приведены в листингах 2.20 и 2.21.

Листинг 2.20. Символьные и численные расчеты рядов

$$\sum_{i=0}^{10} 2^i = 2.047 \times 10^3 \qquad \sum_{i=0}^{10} 2^i \rightarrow 2047$$

$$\sum_{i=0}^{\infty} a^i \rightarrow \frac{-1}{(a-1)}$$

$$\sum_{n=0}^{\infty} \frac{x^n}{2^n \cdot n!} \rightarrow \exp\left(\frac{1}{2} \cdot x\right) \qquad \sum_{n=0}^{\infty} \frac{1^n}{2^n \cdot n!} \rightarrow \exp\left(\frac{1}{2}\right) = 1.649$$

$$\sum_{n=0}^{100} \frac{1^n}{2^n \cdot n!} = 1.649$$

Листинг 2.21. Символьный расчет произведения

$$\prod_{n=1}^{\infty} \frac{1}{n^3 + 1} \rightarrow 0 \qquad \prod_{n=1}^{\infty} \sqrt{n} \rightarrow \infty$$

2.2.15. Подстановка переменной

Очень удобная возможность символьных вычислений — это операция подстановки значения переменной в выражение. При помощи меню подстановка производится следующим образом (рис. 2.16):

1. Выделите значение переменной, которое необходимо подставить в некоторое выражение. Значение переменной может быть любым выражением относительно любых переменных (на рис. 2.16 в качестве подстановки взята самая первая строка документа).
2. Скопируйте значение переменной в буфер обмена, например, нажатием клавиш <Ctrl>+<C> или кнопки **Копировать** (Copy) на панели инструментов **Стандартная** (Standard).
3. Выделите в выражении, в которое требуется подставить значение из буфера обмена, переменную, которая будет заменяться (во второй строке на рис. 2.16 выделена переменная `x`).
4. Выполните команду **Символьные операции / Переменная / Подставить** (Symbolics / Variable / Substitute).

Результат этих действий иллюстрируется нижней строкой в документе на рис. 2.16.

Для осуществления той же операции в совокупности с оператором символьного вывода используйте ключевое слово `substitute`, которое вставляется в документ одноименной кнопкой на панели **Символьные** (Symbolic). После ключевого слова `substitute` необходимо ввести в местозаполнители логическое выражение, показывающее, какую именно переменную какой формулой следует заменить (листинг 2.22).

Листинг 2.22. Подстановка значения переменной

$$\sin\left(k\cdot x^2+b\cdot x\right)\ \text{substitute},\ k=a\cdot x^2\ \rightarrow \sin\left(a\cdot x^4+b\cdot x\right)$$

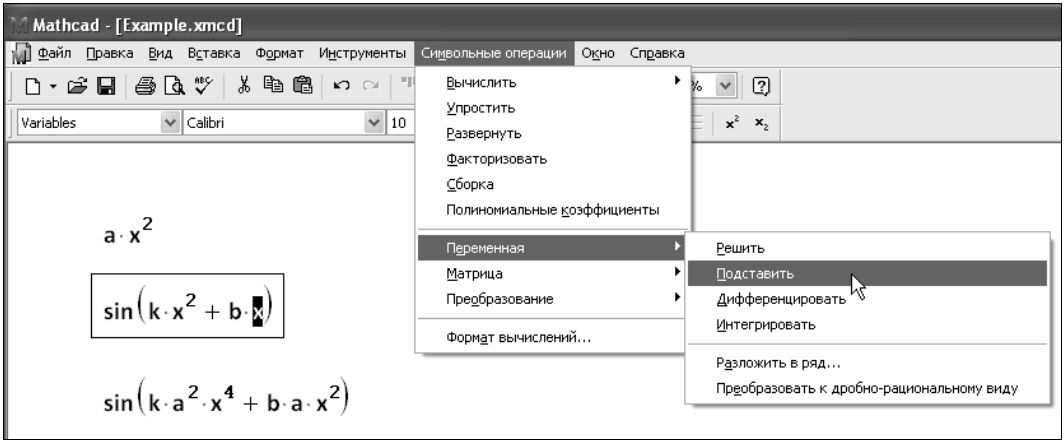


Рис. 2.16. Подстановка значения переменной

2.2.16. Вычисление предела

Наиболее ярким проявлением возможностей символьного процессора в Mathcad являются аналитические вычисления пределов, производных, интегралов и разложений в ряд, а также решение алгебраических уравнений. Все эти операции, при выполнении их посредством меню **Символьные операции** (Symbolics), находятся в его подменю **Переменная** (Variable). Соответственно, требуется предварительное выделение в выражении переменной, относительно которой будет совершаться операция. Для выделения переменной достаточно поместить ее между линиями ввода, но для большей наглядности лучше выделить ее черным цветом путем протаскивания указателя мыши через нужную часть выражения.

> ***Примечание***
> В отличие от других вычислительных операторов, пределы могут быть вычислены только аналитически.

Пределы (листинг 2.23):

- ❒ двусторонний;
- ❒ левый;
- ❒ правый.

Листинг 2.23. Операторы символьного вычисления пределов

$$\lim_{x \to \infty} \frac{1 + 3 \cdot x}{x} \to 3$$

$$\lim_{x \to 0^+} \frac{1}{x} \to \infty$$

$$\lim_{x \to 0^-} \frac{1}{x} \to -\infty$$

2.2.17. О специфике аналитических вычислений

Ранее в этой главе были разобраны основные приемы символьных вычислений в Mathcad. Они, как правило, были показаны на простых примерах, которые иллюстрировали ту или иную символьную операцию. Тем не менее при проведении разнообразных (и численных тоже) расчетов в Mathcad возможности символьного процессора можно использовать более эффективно. Отметим некоторые из них.

При проведении символьных вычислений с оператором символьного вывода функции пользователя и переменные, определенные ранее в документе Mathcad, воспринимаются символьным процессором корректно. Таким образом, имеется мощный аппарат включения символьных расчетов в программы пользователя. Примеры применения функции пользователя приведены в листингах 2.24 и 2.25. Сравните последние строчки этих листингов. Несмотря на их идентичность слева от знака

символьного вывода, результат получен различный. Это связано с тем, что в листинге 2.25 предварительно переменной `x` присвоено значение `4`. Поскольку значения переменных влияют на символьные вычисления, то результат учитывает подстановку вместо `x` числа `4`.

Листинг 2.24. Функция пользователя в символьных вычислениях

$$f(k,x) := \cos(k\cdot x) + 4\cdot x^{2-k}$$

$$f(k,x)\ \text{substitute},\ k=\sqrt{x} \rightarrow \cos\left(x^{\frac{3}{2}}\right) + 4\cdot x^{\left(2-x^{\frac{1}{2}}\right)}$$

$$f(k,x)\ \text{series},\ k,2 \rightarrow 1 + 4\cdot x^2 - 4\cdot x^2\cdot \ln(x)\cdot k$$

Листинг 2.25. Значения переменных влияют на результат символьных вычислений

$$f(k,x) := \cos(k\cdot x) + 4\cdot x^{2-k}$$

$$x := 4$$

$$f(k,x)\ \text{series},\ k,2 \rightarrow 65 - 64\cdot \ln(4)\cdot k$$

Напротив, при осуществлении символьных операций через меню **Символьные операции** (Symbolics) символьный процессор "не видит" ничего, кроме выражения, в пределах которого находятся линии ввода. Поэтому ни функции пользователя, ни предварительно определенные значения каких-либо переменных никак не влияют на вычисления.

Совет

Используйте меню **Символьные операции** (Symbolics), если требуется "сиюминутно" провести некоторые аналитические действия с выражением и получить ответ в общем виде, не учитывающем текущие значения переменных, входящих в выражение.

ГЛАВА 3

Дифференцирование

Операция дифференцирования реализована в Mathcad как в численной, так и в аналитической форме и обозначается при помощи традиционного оператора, т. е. соответствующими математическими символами (подобно сложению или умножению). Если расчеты выполняются с помощью вычислительного процессора, необходимо хорошо представлять себе особенности численного алгоритма, действие которого остается для пользователя "за кадром". С помощью Mathcad можно вычислять производные скалярных функций любого количества аргументов, причем как функции, так и аргументы могут быть и действительными, и комплексными.

3.1. Аналитическое дифференцирование

Вне всякого сомнения, вы по достоинству оцените возможности символьного процессора, позволяющего с легкостью выполнять рутинные вычисления производных громоздких функций. В отличие от всех других операций, символьное дифференцирование выполняется успешно для подавляющего большинства аналитически заданных функций. Думаю, что многие читатели, которым иногда приходится аналитически дифференцировать функции, как и автор этих строк, единожды попробовав сделать это в среде Mathcad, уже никогда не возьмут в руки карандаша для расчета производных "вручную".

3.1.1. Аналитическое дифференцирование функции

Для того чтобы аналитически найти производную функции `f(x)` в Mathcad:

1. Задайте функцию `f(x)`.
2. Введите оператор дифференцирования нажатием кнопки **Производная** (Derivative) на панели **Математический анализ** (Calculus) или введите с клавиатуры вопросительный знак <?>.
3. В появившихся местозаполнителях оператора дифференцирования (рис. 3.1) введите функцию, зависящую от аргумента `x`, т. е. `f(x)`, и имя самого аргумента `x`.

Рис. 3.1. Оператор дифференцирования

4. Введите оператор <→> символьного вычисления для получения ответа (листинг 3.1).

Листинг 3.1. Пример аналитического дифференцирования

$$f(x) := \sin(x) \cdot \ln(x)$$

$$\frac{d}{dx} f(x) \rightarrow \cos(x) \cdot \ln(x) + \frac{\sin(x)}{x}$$

Примечание 1

Помните о том, что в описанном применении оператора дифференцирования его результатом является *функция* той же переменной x. Пример визуализации операции дифференцирования с помощью графика приведен на рис. 3.2.

Примечание 2

Исходная функция может зависеть не только от аргумента x, но и от других аргументов, например f(x,y,z,t) и т. п. В этом случае дифференцирование производится точно так же, причем становится более понятной необходимость определения переменной дифференцирования (в нижнем местозаполнителе оператора дифференцирования). Расчеты производных по разным аргументам (в этом случае говорят о *частных производных*), разумеется, будут давать совершенно разные результаты (*см. разд. 3.4*).

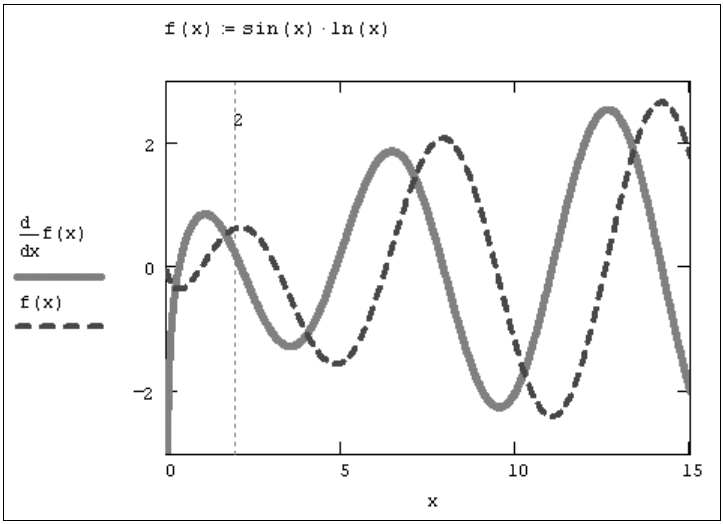


Рис. 3.2. График производной функции

3.1.2. Вычисление производной функции в точке

Для того чтобы рассчитать производную в точке, необходимо предварительно задать значение аргумента в этой точке (листинг 3.2, вторая строка). Результатом дифференцирования в этом случае будет число — значение производной в этой точке. Если результат удается отыскать аналитически, то он приводится в виде числового выражения, а для того чтобы получить его в форме числа, достаточно ввести после выданного выражения символ числового равенства <=> (последняя строка листинга 3.2).

Листинг 3.2. Аналитическое дифференцирование функции в точке

$$f(x) := \sin(x) \cdot \ln(x)$$

$$x := 2$$

$$\frac{d}{dx} f(x) \rightarrow \cos(2) \cdot \ln(2) + \frac{1}{2} \cdot \sin(2) = 0.166$$

Для того чтобы продифференцировать функцию, вовсе не обязательно предварительно присваивать ей какое-либо имя, как это сделано в листингах 3.1 и 3.2. Можно определить функцию непосредственно в операторе дифференцирования (это демонстрирует первая строка листинга 3.3).

Листинг 3.3. Правильное и неправильное использование оператора дифференцирования

$$\frac{d}{dx} \sin(x) \rightarrow \cos(x)$$

$$\frac{d}{dx} \sin(2) \rightarrow 0$$

Как вы заметили, оператор дифференцирования, в основном, соответствует его общепринятому математическому обозначению, и поэтому его легко использовать интуитивно. Однако в некоторых случаях при вводе оператора дифференцирования следует проявить осторожность. Рассмотрим один показательный пример, приведенный во второй строке листинга 3.3, который демонстрирует неправильное применение оператора дифференцирования для вычисления производной в точке. Вместо вычисления производной `sin(x)` при `x=2`, как этого можно было ожидать, получено нулевое значение. Это случилось из-за того, что аргумент функции `sin(x)` введен не в виде переменной `x`, а в виде числа. Поэтому Mathcad воспринимает последнюю строку как вычисление сначала значения синуса в точке `x=2`, а затем дифференцирование этого значения (т. е. константы) также в точке `x=2`, в соответствии с требованием первой строки листинга. Поэтому ответ, на самом деле, неудивителен — в какой точке ни дифференцируй константу, результатом будет ноль.

Примечание

То же самое касается и операции численного дифференцирования, т. е. применения оператора <=> вместо <→> (*см. листинг 3.3.2 на компакт-диске*).

3.1.3. Определение функций пользователя через оператор дифференцирования

Разумеется, оператор дифференцирования, как и любой другой, можно применять для определения собственных функций пользователя. В листинге 3.4 через производную от `f(x)` определяется еще одна пользовательская функция `g(x)`, затем при помощи оператора символьного вывода находится ее явный вид (предпоследняя строка листинга) и конкретное значение в точке `x=1` (последняя строка).

Листинг 3.4. Определение функции посредством оператора дифференцирования

$$f(x) := x^4 + 2x^3 - 7x^2 + 3x - 1$$

$$g(x) := \frac{d}{dx} f(x)$$

$$g(x) \rightarrow 4 \cdot x^3 + 6 \cdot x^2 - 14 \cdot x + 3$$

$$g(1) \rightarrow -1$$

3.1.4. Дифференцирование при помощи меню

Чтобы аналитически продифференцировать выражение по некоторой переменной, выделите в нем эту переменную и выберите команду **Символьные операции / Переменная / Дифференцировать** (Symbolics / Variable / Differentiate) (рис. 3.3).

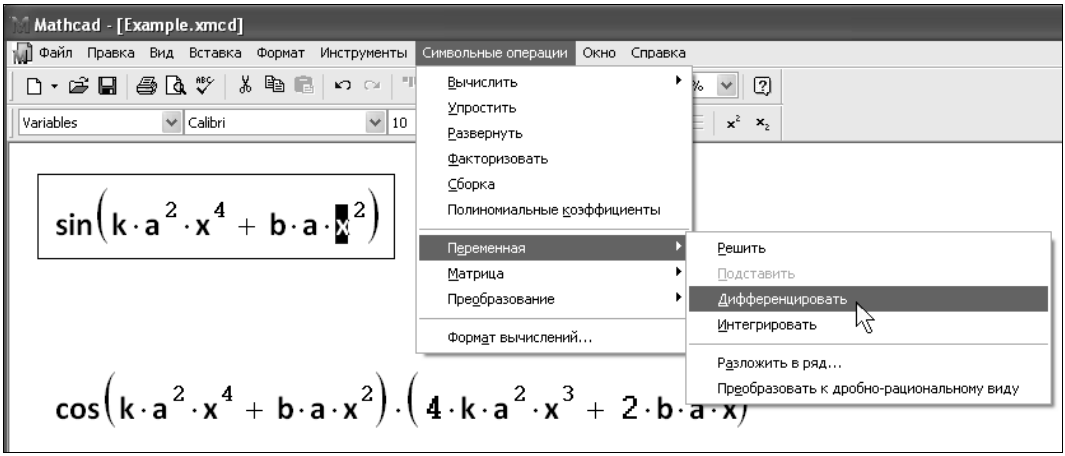


Рис. 3.3. Аналитическое дифференцирование по переменной

В результате в следующей строке за выражением появится значение ее производной. Для того чтобы найти вторую производную, повторно примените эту последовательность действий, но уже к полученному результату дифференцирования. Так же находятся и производные высших порядков.

3.2. Численное дифференцирование

Вычислительный процессор Mathcad обеспечивает превосходную точность численного дифференцирования.

3.2.1. Дифференцирование в точке

Для того чтобы численно продифференцировать функцию `f(x)` в некоторой точке, следует использовать оператор численного вывода (вместо символьного):

1. Определите точку `x`, в которой будет вычислена производная, например, `x:=1`.
2. Введите оператор дифференцирования и обычным образом введите имена функции и аргумента в местозаполнители (см. рис. 3.1).
3. Введите оператор = численного вывода результата.

Пример дифференцирования функции `f(x)=sin(x)·ln(x)` приведен в листинге 3.5.

Листинг 3.5. Численное дифференцирование функции в точке

```
f(x) := sin(x)·ln(x)
x := 0.1
d/dx f(x) = -1.293
```

Внимание!

Не забывайте предварительно определять точку, в которой производится численное дифференцирование, как это сделано во второй строке листинга 3.5. Иначе будет выдано сообщение об ошибке, показанное на рис. 3.4, означающее, что переменная или функция, входящая в выражение, ранее не определена. Между тем, символьное дифференцирование (*см. разд. 3.1*) не требует обязательного явного задания точки дифференцирования. В этом случае вместо значения производной (числа или числового выражения) будет выдана аналитическая зависимость (см. листинг 3.1).

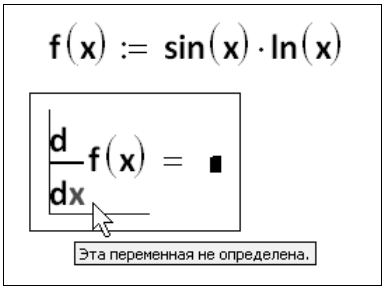


Рис. 3.4. Ошибка в применении оператора дифференцирования (не задан аргумент)

Примечание

Для ускорения и повышения точности численного дифференцирования функций, заданных аналитически, автоматически задействуется символьный процессор. Сначала предпринимается попытка отыскать производные, входящие в выражение, аналитически, а уже затем, если она оказывается неудачной, включается в работу численный метод.

3.2.2. Об алгоритме дифференцирования

Для численного дифференцирования Mathcad применяет довольно сложный алгоритм, вычисляющий производную с колоссальной точностью до 7—8-го знака после запятой. Погрешность дифференцирования не зависит от констант `TOL` или `CTOL`, в противоположность большинству остальных численных методов, а определяется непосредственно алгоритмом. Этот алгоритм (метод Риддера) описан во встроенной справочной системе Mathcad, доступной через меню **Справка** (Help). Мы не будем здесь его описывать, однако остановимся на важных аспектах численного определения производной функции `f(x)` на более простом примере. Несмотря на то, что простейшая разностная формула сильно отличается от метода Риддера, он все-таки поможет нам разобраться в некоторых вопросах, т. к. основан на базовом принципе численного дифференцирования, а именно на вычислении производной через значения функции `f(x)` в нескольких точках, расположенных на близком расстоянии друг от друга.

Исходя из определения производной функции, можно констатировать, что

$$\frac{df(x)}{dx} = \frac{f(x + \Delta) - f(x)}{\Delta} + o(\Delta). \qquad (3.1)$$

Основная проблема численного определения производной (как в этой простейшей формуле, так и в более сложных алгоритмах, в том числе Риддера) связана как раз с процедурой выбора значения Δ, которая является далеко не очевидной. На первый взгляд может показаться, что следует выбирать очень малые Δ, чтобы соблюсти желаемую точность, однако это не совсем так. Чтобы лучше разобраться в сути проблемы, используем Mathcad-программу, приведенную в листинге 3.6, которая рассчитывает (в зависимости от шага Δ) погрешность разностной формулы (3.1). График полученной зависимости изображен на рис. 3.5, причем для его обеих осей выбран логарифмический масштаб, а сама производная (ради примера), согласно листингу 3.6, считается в одной точке `x=1`.

Листинг 3.6. Расчет зависимости точности разностной формулы от шага Δ

$f(x) := \sin(x) \cdot \ln(x)$

$x := 1$

$i := 0 .. 20$

$\Delta_i := 10^{-i}$

$$\varepsilon_i := \left| \frac{d}{dx} f(x) - \frac{f(x + \Delta_i) - f(x)}{\Delta_i} \right|$$

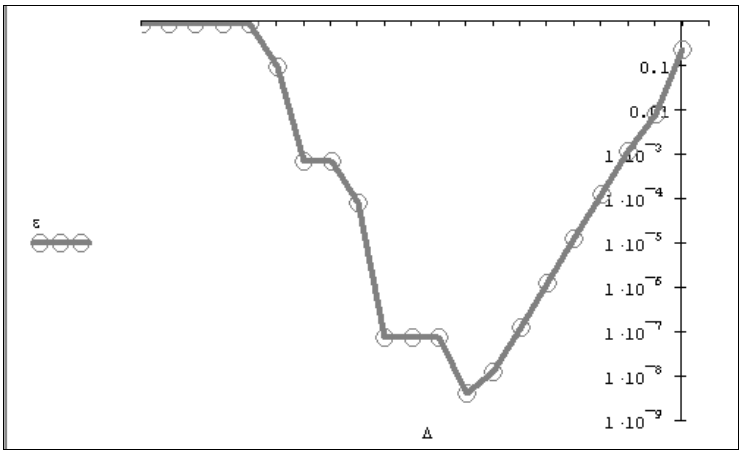


Рис. 3.5. График точности формулы (3.1) в зависимости от шага Δ (продолжение листинга 3.6)

Если увеличение ошибки на правом конце графика является совершенно очевидным, поскольку, согласно формуле (3.1), чем больше Δ, тем больше погрешность, то рост ошибки при очень малых Δ может, на первый взгляд, показаться неожиданным. Однако все дело в том, что, применяя разностную формулу, мы неявно полагали, что умеем точно вычислять значения функции `f(x)` в любой точке. Между тем, любые компьютерные вычисления сопряжены с неустранимыми погрешностями, в частности, обусловленными дискретным представлением чисел. Поэтому в реальности мы можем вычислить значение `f(x)` лишь с некоторой погрешностью δ, обусловленной (по крайней мере) заведомым округлением чисел при расчетах на компьютере.

В результате при очень малом шаге разностные формулы означают вычитание друг из друга близких чисел. В этом случае ошибки вычисления функции `f(x)` становятся доминирующими и приводят к существенному росту суммарной погрешности вычисления разностной производной. Отсюда как раз и следует тот вывод, что значение шага следует выбирать "не очень малым", иначе ошибки вычисления `f(x)` неминуемо сделают результат дифференцирования неправильным. Глядя на рис. 3.5, легко сообразить, что в данном случае следует выбирать промежуточные значения Δ, которые обеспечат минимальную (или почти минимальную) погрешность.

Следует подчеркнуть, что в зависимости от характера дифференцируемой функции диапазон приемлемых значений Δ будет различным. Поэтому в каждом конкретном случае требуется совершать дополнительные шаги, тестирующие верность выбора шага для численного дифференцирования. Такая процедура, кстати говоря, заложена в адаптивном алгоритме дифференцирования, примененном в Mathcad, что делает его весьма надежным для численного расчета производной.

С учетом сказанного ранее, с дифференцированием в Mathcad обычно не возникает сложных проблем. Исключение составляют функции, которые дифференцируются

в окрестности сингулярной точки; например, для функции `f(x)=1/x` это будут точки вблизи `x=0`. При попытке найти ее производную при `x=0` (рис. 3.6) будет выдано сообщение об одной из ошибок деления на ноль.

Рис. 3.6. Если производная функции в данной точке не существует, выдается сообщение об ошибке

3.3. Производные высших порядков

Mathcad позволяет численно определять производные высших порядков, от `0`-го до `5`-го включительно. Чтобы вычислить производную функции `f(x)` `N`-го порядка в точке `x`, нужно проделать те же самые действия, что и при взятии первой производной (*см. разд. 3.1 и 3.2*), за тем исключением, что вместо оператора производной необходимо применить оператор **n-я производная** (Nth Derivative). Этот оператор вводится с той же панели **Математический анализ** (Calculus) либо с клавиатуры нажатием клавиш <Ctrl>+<?> и содержит еще два дополнительных местозаполнителя (рис. 3.7), в которые следует поместить число `N`.

Рис. 3.7. Оператор производной высшего порядка

В полном соответствии с математическим смыслом оператора определение порядка производной в одном из местозаполнителей приводит к автоматическому появлению того же числа в другом из них.

Очевидно, что "производная" при `N=0` по определению равна самой функции, при `N=1` получается обычная первая производная. Листинг 3.7 демонстрирует численное и символьное вычисление второй производной функции в заданной точке. Обратите внимание, что, как и при вычислении обычной производной, необходимо перед оператором дифференцирования присвоить аргументу функции значение, для которого будет вычисляться производная. А вот для аналитического нахождения производных высших порядков при помощи оператора символьного вывода (*в полном соответствии с разд. 3.1*) вводить значения аргумента не следует (листинг 3.8).

Листинг 3.7. Пример вычисления второй производной функции в точке

$$f(x) := \frac{1}{x}$$

$$x := 3$$

$$\frac{d^2}{dx^2} f(x) = 0.074$$

$$\frac{d^2}{dx^2} f(x) \rightarrow \frac{2}{27}$$

Листинг 3.8. Пример аналитического поиска второй производной функции

$$f(x) := \frac{1}{x}$$

$$\frac{d^2}{dx^2} f(x) \rightarrow \frac{2}{x^3}$$

Примечание

Убедиться в том, что символьный процессор Mathcad в последней строке листинга 3.7 дает тот же результат, что и вычислительный процессор в предыдущей строке, можно, упростив его. Для этого следует выделить полученное последнее выражение и выбрать в меню **Символьные операции** (Symbolics) пункт **Упростить** (Simplify). После этого ниже появится еще одна строка с численным результатом выделенного выражения.

Повторимся, что численный метод предусматривает возможность вычисления производных до 5-го порядка, а символьный процессор умеет считать производные произвольного порядка (конечно, если аналитическое решение задачи в принципе существует). Сказанное иллюстрирует листинг 3.9, в котором аналитически вычисляется шестая производная функции, а попытка численного вывода результата того же выражения приводит к ошибке.

Листинг 3.9. Численное и символьное вычисление шестой производной

$$f(x) := \frac{1}{x}$$

$$\frac{d^6}{dx^6} f(x) \rightarrow \frac{720}{x^7}$$

$$x := 3$$

$$\frac{d^6}{dx^6} f(x) = \blacksquare$$

Чтобы вычислить производную порядка выше 5-го численно, можно последовательно применить несколько раз оператор N-й производной (листинг 3.10), подобно тому, как производится отыскание кратных интегралов (*см. разд. 4.3.4*). Однако следует помнить о том, что численное определение производных высших порядков производится тем же вычислительным методом Риддера, что и для первых производных. Поскольку, как уже было сказано, для первой производной этот метод обеспечивает точность до 7—8 значащих разрядов числа, при повышении порядка производной на каждую единицу точность падает примерно на один разряд.

Внимание!

Из сказанного ясно, что падение точности при численном расчете высших производных может быть очень существенно. В частности, если попытаться определить шестую производную функции 1/x, то в качестве результата будет выдан ноль, в то время как истинное значение девятой производной может быть найдено при помощи символьного процессора (листинг 3.10).

Листинг 3.10. Попытка численного поиска шестой производной функции в точке дает неправильный результат

$$f(x) := \frac{1}{x}$$

$$x := 0.1$$

$$\frac{d}{dx}\left(\frac{d^5}{dx^5} f(x)\right) = 0$$

$$\frac{d^6}{dx^6} f(x) \rightarrow 7.20 \cdot 10^9$$

3.4. Частные производные

С помощью обоих процессоров Mathcad можно вычислять производные функций не только одного, но и любого количества аргументов. Как известно, производные функции нескольких аргументов по одному из них называются *частными*. Чтобы вычислить частную производную, необходимо, как обычно, ввести оператор производной с панели **Математический анализ** (Calculus) и в соответствующем местозаполнителе напечатать имя переменной, по которой должно быть осуществлено дифференцирование.

3.4.1. Частные производные

Примеры отыскания частных производных функции двух переменных приведены в листингах 3.11 и 3.12. В первой строке обоих листингов определяется сама функция, а в последующих (символьным или численным образом) рассчитываются ее производные по обеим переменным — x и k. Чтобы определить частную производную в точке, необходимо предварительно задать значения всех аргументов, что и

сделано в следующих строках листинга 3.12. Обратите внимание, что для символьного поиска производной функции нет необходимости задавать значения всех ее аргументов (третья строка листинга 3.12), а вот для численного дифференцирования (последняя строка листинга) должны быть предварительно определены все аргументы функции, иначе вместо результата появится сообщение об ошибке.

Листинг 3.11. Аналитическое вычисление частных производных

$$f(x, k) := k \cdot \sin(x)$$

$$\frac{\partial}{\partial x} f(x, k) \rightarrow k \cdot \cos(x)$$

$$\frac{\partial}{\partial k} f(x, k) \rightarrow \sin(x)$$

Листинг 3.12. Символьное и численное вычисления частных производных в точке

$$f(x, k) := k \cdot \sin(x)$$

$$x := 10$$

$$\frac{\partial}{\partial x} f(x, k) \rightarrow k \cdot \cos(10)$$

$$k := 1$$

$$\frac{\partial}{\partial x} f(x, k) = -0.839$$

Частные производные высших порядков рассчитываются точно так же, как и обычные производные высших порядков (*см. разд. 3.3*). Листинг 3.13 иллюстрирует расчет вторых производных функции по переменным `x` и `y`, а также смешанной производной.

Листинг 3.13. Вычисление второй частной производной

$$f(x, y) := y^2 \cdot x^3 + y \cdot x^2$$

$$\frac{\partial^2}{\partial x^2} f(x, y) \rightarrow 6 \cdot y^2 \cdot x + 2 \cdot y$$

$$\frac{\partial^2}{\partial y^2} f(x, y) \rightarrow 2 \cdot x^3$$

$$\frac{\partial}{\partial x} \frac{\partial}{\partial y} f(x, y) \rightarrow 6 \cdot y \cdot x^2 + 2 \cdot x$$

Возможно, вы обратили внимание, что во всех трех листингах 3.11—3.13 оператор дифференцирования записан в традиционной форме частной производной (с округлыми символами дифференциала). Запись оператора не влияет на вычисления, а служит лишь более привычной формой представления расчетов.

Для того чтобы изменить вид оператора дифференцирования на представление частной производной, следует:

1. Вызвать контекстное меню из области оператора дифференцирования нажатием правой кнопки мыши.
2. Выбрать в контекстном меню верхний пункт **Отображать производную как** (View Derivative As).
3. В появившемся подменю (рис. 3.8) выбрать пункт **Частную производную** (Partial Derivative).

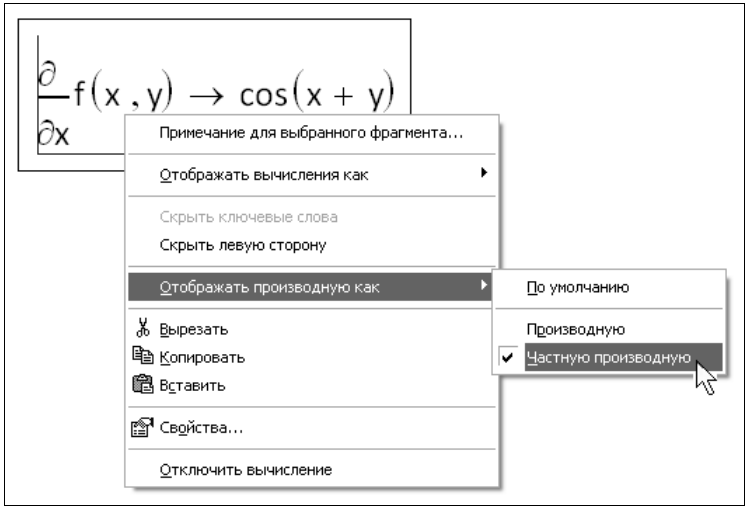


Рис. 3.8. Изменение вида оператора дифференцирования

Чтобы вернуть вид производной, принятый по умолчанию, выберите в подменю пункт **По умолчанию** (Default) либо, для представления ее в обычном виде, — пункт **Производную** (Derivative).

3.4.2. Примеры: градиент, дивергенция и ротор

Завершим разговор о частных производных несколькими примерами векторного анализа, которые нередко встречаются в вычислительной практике. Программная реализация первого из них, посвященная вычислению *градиента* функции двух переменных, приведена в листинге 3.14. В качестве примера взята функция `f(x,y)`, определяемая в первой строке листинга, график которой показан на рис. 3.9, в виде линий уровня. Как известно, градиент функции `f(x,y)` является векторной функцией тех же аргументов, что и `f(x,y)`, определенной через ее частные производные, согласно второй строке листинга 3.14. В его третьей строке производится аналити-

ческое вычисление градиента, а в оставшейся части листинга задаются ранжированные переменные и матрицы, необходимые для подготовки графика линий уровня самой функции и графика векторного поля ее градиента (рис. 3.10).

14 *Примечание*

В Mathcad 14 появился оператор вычисления градиента функции, который записывается следующим образом:

$$\nabla_{x,y} f(x,y) \rightarrow \begin{bmatrix} y + 0.24\cdot x \\ x + (-0.04)\cdot y^3 \end{bmatrix}$$

Таким образом, вычисления, приведенные в листинге 3.14, можно записать проще и нагляднее, заменив приведенной формулой вторую и третью строку листинга.

Листинг 3.14. Вычисление градиента функции двух переменных

$$f(x,y) := 0.12\cdot x^2 + x\cdot y - 0.01\cdot y^4$$

$$\mathrm{grad}(x,y) := \begin{pmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{pmatrix}$$

$$\mathrm{grad}(x,y) \rightarrow \begin{pmatrix} .24\cdot x + y \\ x - 4.\cdot 10^{-2}\cdot y^3 \end{pmatrix}$$

$N := 5$

$i := 0 .. 2\cdot N$ $\qquad$ $j := 0 .. 2\cdot N$

$F_{i,j} := f(i-N, j-N)$ $\qquad$ $V_{i,j} := \mathrm{grad}(i-N, j-N)$

$X_{i,j} := |V_{i,j}|_0$ $\qquad$ $Y_{i,j} := |V_{i,j}|_1$

Как можно убедиться, сравнив графики на рис. 3.9 и 3.10, математический смысл градиента состоит в задании в каждой точке `(x,y)` направления на плоскости, в котором функция `f(x,y)` растет наиболее быстро.

Абсолютное значение градиента (т. е. длина вектора в каждой точке) определяет локальную скорость изменения `f(x,y)`. Из сопоставления графиков ясно, что в центре показанной на них области `(x,y)` сама функция `f(x,y)` меняется медленно (соответственно, значения ее градиента являются малыми), а в углах — быстро (там значения градиента максимальны).

Очень важно заметить, что градиент является не скалярной, а векторной функцией переменных `x,y`, поскольку фактически представляет собой комбинацию двух функций, задающих соответствующие проекции (горизонтальную и вертикальную) вектора в каждой точке. До сих пор в данной главе мы рассматривали дифференцирование скалярных функций, однако в математике часто приходится иметь дело и

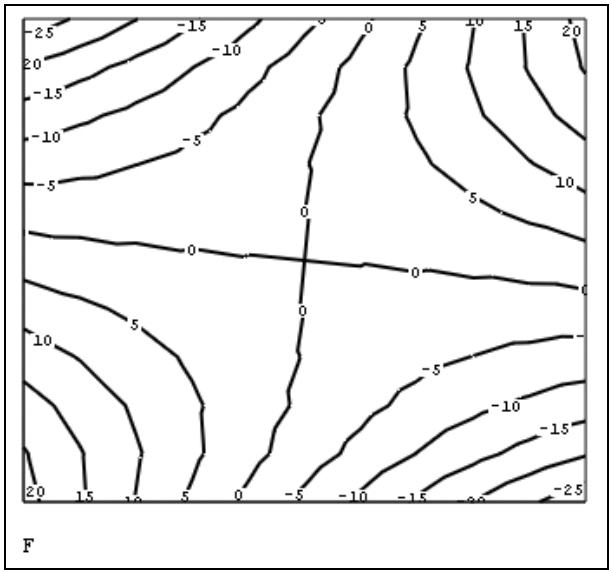


Рис. 3.9. Модельная функция двух переменных
(продолжение листинга 3.14)

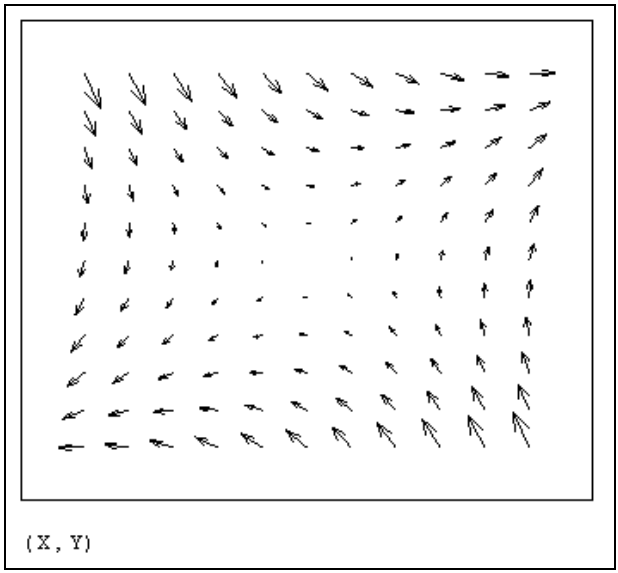


Рис. 3.10. Векторное поле градиента функции двух переменных
(продолжение листинга 3.14)

с вычислением производных векторных функций. Рассмотрим эти действия на примере операции поиска *дивергенции* (листинг 3.15 и рис. 3.11), применимой к *векторному полю*, т. е. векторной функции, зависящей от пространственных координат (на плоскости, как в нашем примере, или в трехмерном пространстве).

Листинг 3.15. Вычисление дивергенции векторной функции

$$f(x,y) := \begin{pmatrix} 0.24\cdot x + y \\ x - 4\cdot 10^{-2}\cdot y^{3} \end{pmatrix}$$

$$\mathrm{div}(x,y) := \frac{\partial}{\partial x}\left| f(x,y)_0 \right| + \frac{\partial}{\partial y}\left| f(x,y)_1 \right|$$

$$\mathrm{div}(x,y) \rightarrow .24 - \frac{3}{25}\cdot y^{2}$$

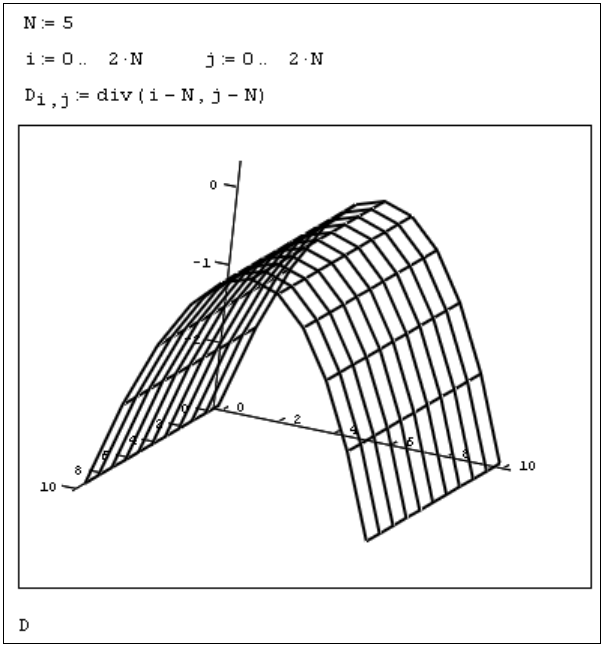


Рис. 3.11. График дивергенции векторной функции (продолжение листинга 3.15)

Если, как принято в математике, обозначить оператор взятия градиента символом ∇, то дивергенцию вектор-функции можно формально определить как скалярное произведение $\nabla\cdot$`f`, а еще одну распространенную операцию векторного анализа — *ротор* (или, по-другому, *вихрь* или *завихренность*) — как векторное произведение $\nabla\times$`f`. Рисунок 3.11 иллюстрирует пример векторной функции `f(x,y)` (определяемой в первой строке листинга) и вычисление ее дивергенции (которое производится аналитически в третьей строке). Обратите внимание, что в качестве исходной вектор-функции взят результат предыдущих расчетов, показанный (в форме векторного поля) на рис. 3.10. Строки кода в верхней части рис. 3.11 нужны для подготовки графика вычисленной дивергенции (в виде трехмерной поверхности и линий уровня, соответственно, сверху и снизу).

Точно такую же структуру имеют расчеты ротора той же векторной функции `f(x,y)` в листинге 3.16, причем определение операции взятия ротора приводится в его второй строке (как и в случае дивергенции для листинга 3.15). Читателю, знакомому с векторным анализом, предлагается догадаться самому, почему в рассматриваемом примере (листинги 3.14—3.16) ротор получается тождественно равным нулю (последняя строка листинга 3.16).

Листинг 3.16. Вычисление ротора векторной функции

$$f(x,y) := \begin{pmatrix} 0.24\cdot x + y \\ x - 4\cdot 10^{-2}\cdot y^{3} \end{pmatrix}$$

$$\mathrm{rot}(x,y) := \frac{\partial}{\partial x}\left| f(x,y)_1 \right| - \frac{\partial}{\partial y}\left| f(x,y)_0 \right|$$

$$\mathrm{rot}(x,y) \rightarrow 0$$

В заключение разговора о векторном анализе функций подчеркнем, что примеры в листингах 3.14—3.16 относились к функциям двух переменных, т. е. описывали двумерный случай. Еще два листинга — 3.17 и 3.18 — показывают, как действуют перечисленные операции векторного анализа в трехмерном (пространственном) случае.

Листинг 3.17. Градиент функции трех переменных

$$f(x,y,z) := z\cdot\sin(x\cdot y)$$

$$\mathrm{grad}(x,y,z) := \begin{pmatrix} \frac{\partial}{\partial x} f(x,y,z) \\ \frac{\partial}{\partial y} f(x,y,z) \\ \frac{\partial}{\partial z} f(x,y,z) \end{pmatrix} \qquad \mathrm{grad}(x,y,z) \rightarrow \begin{pmatrix} z\cdot\cos(x\cdot y)\cdot y \\ z\cdot\cos(x\cdot y)\cdot x \\ \sin(x\cdot y) \end{pmatrix}$$

Листинг 3.18. Дивергенция и ротор в трехмерном пространстве

$$f(x,y,z) := \begin{pmatrix} x\cdot y \\ z \\ x + 2z \end{pmatrix}$$

$$\mathrm{div}(x,y,z) := \frac{\partial}{\partial x}\left| f(x,y,z)_0 \right| + \frac{\partial}{\partial y}\left| f(x,y,z)_1 \right| + \frac{\partial}{\partial z}\left| f(x,y,z)_2 \right|$$

$$\mathrm{rot}(x,y,z) := \begin{bmatrix} \frac{\partial}{\partial y}\left|f(x,y,z)_2\right| - \frac{\partial}{\partial z}\left|f(x,y,z)_1\right| \\ \frac{\partial}{\partial z}\left|f(x,y,z)_0\right| - \frac{\partial}{\partial x}\left|f(x,y,z)_2\right| \\ \frac{\partial}{\partial x}\left|f(x,y,z)_1\right| - \frac{\partial}{\partial y}\left|f(x,y,z)_0\right| \end{bmatrix}$$

$$\mathrm{div}(x,y,z) \rightarrow y+2$$

$$\mathrm{rot}(x,y,z) \rightarrow \begin{pmatrix} -1 \\ -1 \\ -x \end{pmatrix}$$

3.4.3. Пример: якобиан

Еще одна задача, связанная с нахождением частных производных векторной функции, заключается в вычислении *якобиана* (или определителя *матрицы Якоби*) — матрицы, составленной из частных производных векторной функции по всем ее аргументам. Эта задача встречается в различных областях математики, например, применительно к жестким дифференциальным уравнениям (*см. разд. 9.4*). Приемы вычисления якобиана векторной функции `f(x)` векторного аргумента `x` демонстрируются в листинге 3.19. В нем для определения частных производных якобиана каждый `i`-й скалярный компонент $f(x)_i$ дифференцируется символьным процессором Mathcad.

Листинг 3.19. Вычисление якобиана векторной функции векторного аргумента

$$f(x) := \begin{bmatrix} x_0 \cdot x_1 \\ \left|x_1\right|^{x_0} \\ x_1 \cdot x_2 \end{bmatrix}$$

$$J(x,y,z) := \begin{bmatrix} \frac{\partial}{\partial x}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_0 & \frac{\partial}{\partial y}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_0 & \frac{\partial}{\partial z}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_0 \\ \frac{\partial}{\partial x}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_1 & \frac{\partial}{\partial y}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_1 & \frac{\partial}{\partial z}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_1 \\ \frac{\partial}{\partial x}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_2 & \frac{\partial}{\partial y}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_2 & \frac{\partial}{\partial z}\left[f\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)\right]_2 \end{bmatrix}$$

$$J(x,y,z) \rightarrow \begin{pmatrix} y & x & 0 \\ y^{x}\cdot\ln(y) & y^{x}\cdot\frac{x}{y} & 0 \\ 0 & z & y \end{pmatrix}$$

Тот же самый якобиан можно вычислить и несколько по-другому, если определить функцию не одного векторного, а трех скалярных аргументов `f(x,y,z)` (листинг 3.20). Не забывайте, что для численного определения якобиана необходимо сначала определить точку, в которой он будет рассчитываться, т. е. вектор `x` в терминах листинга 3.19, или все три переменных `x,y,z` в обозначениях листинга 3.20.

Листинг 3.20. Вычисление якобиана векторной функции трех скалярных аргументов

$$f(x,y,z) := \begin{pmatrix} x\cdot y \\ y^{x} \\ y\cdot z \end{pmatrix}$$

$$J(x,y,z) := \begin{pmatrix} \frac{\partial}{\partial x}f(x,y,z)_0 & \frac{\partial}{\partial y}f(x,y,z)_0 & \frac{\partial}{\partial z}f(x,y,z)_0 \\ \frac{\partial}{\partial x}f(x,y,z)_1 & \frac{\partial}{\partial y}f(x,y,z)_1 & \frac{\partial}{\partial z}f(x,y,z)_1 \\ \frac{\partial}{\partial x}f(x,y,z)_2 & \frac{\partial}{\partial y}f(x,y,z)_2 & \frac{\partial}{\partial z}f(x,y,z)_2 \end{pmatrix}$$

$$J(x,y,z) \rightarrow \begin{pmatrix} y & x & 0 \\ y^{x}\cdot\ln(y) & y^{x}\cdot\frac{x}{y} & 0 \\ 0 & z & y \end{pmatrix}$$

14 *Примечание*

В Mathcad 14 предусмотрена новая функция вычисления якобиана `Jacob` (листинг 3.21).

14 Листинг 3.21. Вычисление якобиана при помощи встроенной функции Mathcad 14

$$\mathrm{Jacob}\left[\begin{bmatrix} x_0\cdot x_1 \\ |x_1|^{x_0} \\ x_1\cdot x_2 \end{bmatrix}, x\right] \rightarrow \begin{bmatrix} x_1 & x_0 & 0 \\ \ln|x_1|\cdot|x_1|^{x_0} & x_0\cdot|x_1|^{x_0-1} & 0 \\ 0 & x_2 & x_1 \end{bmatrix}$$

$$\mathrm{Jacob}\left[\begin{pmatrix} x\cdot y \\ y^{x} \\ y\cdot z \end{pmatrix}, \begin{pmatrix} x \\ y \\ z \end{pmatrix}\right] \rightarrow \begin{pmatrix} y & x & 0 \\ y^{x}\cdot \ln(y) & x\cdot y^{x-1} & 0 \\ 0 & z & y \end{pmatrix}$$

3.5. Разложение функции в ряд Тейлора

Еще одна операция, тесно связанная с дифференцированием, представляет собой разложение функции в *ряд Тейлора* по любой переменной x в некоторой точке. Если эта точка $x=0$, то ряд называют также *рядом Маклорена*, и он представим в окрестности точки $x=0$ суммой вида $a_0+a_1x+a_2x^2+a_3x^3+\ldots$ Здесь a_i — некоторые коэффициенты, не зависящие от x, но, возможно, являющиеся функциями других переменных, от которых зависит исходная функция. Именно эти коэффициенты выражаются через производные функции. Если она имеет в точке $x=0$ особенность, то соответствующее разложение называют *рядом Лорана*.

При поиске разложения в ряд Тейлора нет необходимости явно рассчитывать все его коэффициенты, поскольку эта операция предусмотрена разработчиками Mathcad и выполняется при помощи символьного процессора. При этом можно использовать для ее осуществления как соответствующие встроенные функции, так и меню **Символьные операции** (Symbolics).

3.5.1. Разложение в ряд при помощи меню

Чтобы разложить некоторое выражение в ряд:

1. Введите выражение.
2. Выделите значение переменной, по которой требуется получить разложение в ряд.
3. Выполните команду **Символьные операции / Переменная / Разложить в ряд** (Symbolics / Variable / Expand to Series) (рис. 3.12).

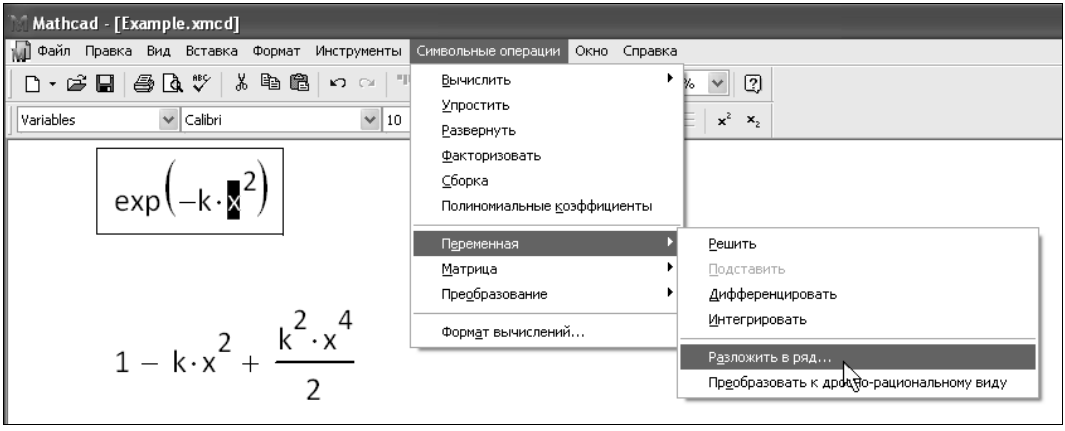


Рис. 3.12. Аналитическое разложение выражения в ряд по переменной x

4. В появившемся диалоговом окне **Разложение в ряд** (Expand to Series) введите желаемый **Порядок аппроксимации** (Order of Approximation) и нажмите кнопку **OK**.

Результат разложения появится под выражением (он показан на рис. 3.12, внизу).

Примечание

Не забывайте, что разложение строится только в точке `x=0`. Чтобы получить разложение в другой точке `x=a`, можно, к примеру, подставить вместо переменной `x` значение `x-a`.

3.5.2. Оператор разложения в ряд

Для разложения в ряд альтернативным способом, с помощью оператора символьного вывода, используйте ключевое слово `series`, вставляя его одноименной кнопкой панели **Символьные операции** (Symbolics). После ключевого слова `series`, через запятую, указывается имя переменной, по которой производится разложение, и порядок аппроксимации (листинги 3.22 и 3.23). Сравнить поведение исходной функции и нескольких ее разложений в ряд Тейлора (с разными порядками аппроксимации) можно по графику, приведенному на рис. 3.13.

Видно, что разложение в ряд хорошо работает в окрестности точки `x=0`, а по мере удаления от нее все сильнее и сильнее отличается от функции. Естественно, что чем выше порядок аппроксимации, тем ближе к исходной функции располагается соответствующее разложение Тейлора.

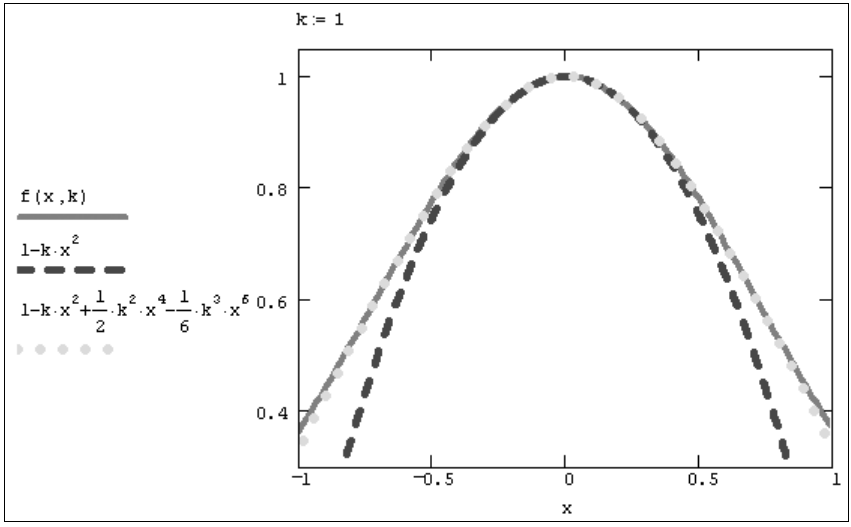


Рис. 3.13. График разложений функции в ряд в зависимости от порядка аппроксимации (продолжение листинга 3.21)

Листинг 3.22. Разложение функции в ряд с разным порядком аппроксимации

$f(x,k) := \exp\left(-k\cdot x^2\right)$

$f(x,k)\ \text{series},x,3 \rightarrow 1-k\cdot x^2$

$f(x,k)\ \text{series},x,5 \rightarrow 1-k\cdot x^2+\frac{1}{2}\cdot k^2\cdot x^4$

$f(x,k)\ \text{series},x,7 \rightarrow 1-k\cdot x^2+\frac{1}{2}\cdot k^2\cdot x^4-\frac{1}{6}\cdot k^3\cdot x^6$

Листинг 3.23. Разложение функции нескольких переменных в ряд по разным переменным

$f(x,k) := \exp\left(-k\cdot x^2\right)$

$f(x,k)\ \text{series},k,4 \rightarrow 1-k\cdot x^2+\frac{1}{2}\cdot k^2\cdot x^4-\frac{1}{6}\cdot k^3\cdot x^6$

$f(x,k)\ \text{series},x,4 \rightarrow 1-k\cdot x^2$

ГЛАВА 4

Интегрирование

С одной стороны, численное интегрирование — одна из самых простых, с вычислительной точки зрения, операций, с другой — аналитически проинтегрировать можно далеко не каждую функцию. Всегда помните об этом, когда вы сталкиваетесь с численным или аналитическим интегрированием.

4.1. Определенный интеграл

Интегрирование в Mathcad реализовано в виде вычислительного оператора. Допускается вычислять интегралы от скалярных функций в пределах интегрирования, которые также должны быть скалярными. Несмотря на то, что пределы интегрирования обязаны быть действительными, подынтегральная функция может иметь и комплексные значения, поэтому и значение интеграла может быть комплексным.

4.1.1. Оператор интегрирования

Интегрирование, как и дифференцирование, и множество других математических действий, устроено в Mathcad по принципу "как пишется, так и вводится". Чтобы вычислить определенный интеграл, следует напечатать его обычную математическую форму в документе. Делается это с помощью панели **Математический анализ** (Calculus) нажатием кнопки со значком интеграла или вводом с клавиатуры сочетания клавиш <Shift>+<7> (или символа "&", что то же самое). Появится символ интеграла с несколькими местозаполнителями (рис. 4.1), в которые нужно ввести

Рис. 4.1. Оператор интегрирования

нижний и верхний интервалы интегрирования, подынтегральную функцию и переменную интегрирования.

Примечание

Если пределы интегрирования имеют размерность, то она должна быть одной и той же для обоих пределов.

Чтобы получить результат интегрирования, следует ввести знак равенства или символьного равенства. В первом случае интегрирование будет проведено численным методом, во втором — в случае успеха будет найдено точное значение интеграла с помощью символьного процессора Mathcad. Эти два способа иллюстрирует листинг 4.1. Конечно, символьное интегрирование возможно только для сравнительно небольшого круга несложных подынтегральных функций.

Листинг 4.1. Численное и символьное вычисление определенного интеграла

$$\int_0^{\pi} \exp\left(-x^2\right)\, dx = 0.886$$

$$\int_0^{\pi} \exp\left(-x^2\right)\, dx \rightarrow \frac{1}{2}\cdot \mathrm{erf}(\pi)\cdot \pi^{\frac{1}{2}} = 0.886$$

Примечание 1

Можно вычислять интегралы с одним или обоими бесконечными пределами (листинг 4.2). Для этого на месте соответствующего предела введите символ бесконечности, воспользовавшись, например, той же самой панелью **Математический анализ** (Calculus). Чтобы ввести $-\infty$ (минус бесконечность), добавьте знак минус к символу бесконечности, как к обычному числу.

Листинг 4.2. Вычисление интеграла с бесконечными пределами

$$\int_{-\infty}^{\infty} \exp\left(-x^2\right)\, dx \rightarrow \pi^{\frac{1}{2}}$$

Примечание 2

Подынтегральная функция может зависеть от любого количества переменных. Именно для того, чтобы указать, по какой переменной Mathcad следует вычислять интеграл, и нужно вводить ее имя в соответствующий местозаполнитель. Помните, что для численного интегрирования по одной из переменных предварительно следует задать значение остальных переменных, от которых зависит подынтегральная функция и для которых вы намерены вычислить интеграл (листинг 4.3).

Листинг 4.3. Интегрирование функции двух переменных по разным переменным

$$\int_a^{b} \exp\left(-x\, z^2\right)\, dx \rightarrow \frac{-1}{z^2}\cdot \exp\left(-b\cdot z^2\right) + \frac{1}{z^2}\cdot \exp\left(-a\cdot z^2\right)$$

$$\int_a^b \exp\left(-x\,z^2\right)\,dz \rightarrow \frac{1}{2}\cdot \mathrm{erf}\left(b\cdot x^{\frac{1}{2}}\right)\cdot\frac{\pi^{\frac{1}{2}}}{x^{\frac{1}{2}}} - \frac{1}{2}\cdot \mathrm{erf}\left(a\cdot x^{\frac{1}{2}}\right)\cdot\frac{\pi^{\frac{1}{2}}}{x^{\frac{1}{2}}}$$

Примечание 3

Оператор интегрирования может использоваться точно так же, как и другие операторы: для определения функций, в циклах и при вычислении ранжированных переменных. Пример присваивания пользовательской функции `f(z)` значения определенного интеграла и вычисления нескольких ее значений приведен на рис. 4.2. На том же рисунке показано, как можно построить график результата интегрирования.

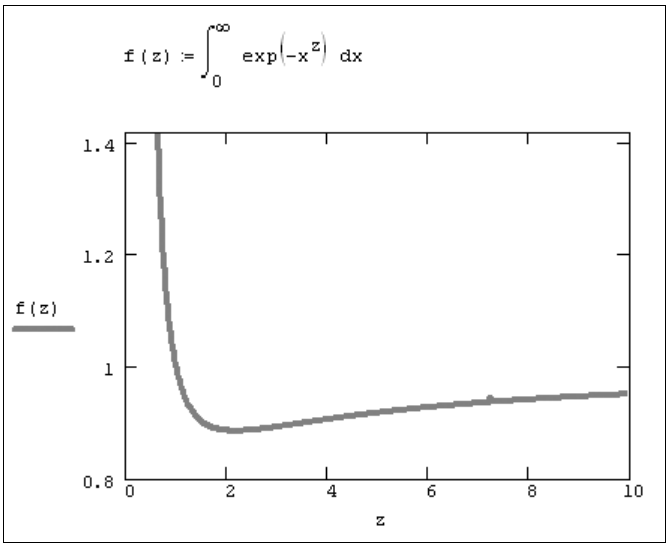


Рис. 4.2. Использование оператора интегрирования в функции пользователя

ⓘ 4.1.2. О выборе алгоритма численного интегрирования

Результат численного интегрирования — это не точное, а приближенное значение интеграла, определенное с погрешностью, которая зависит от встроенной константы `TOL`. Чем она меньше, тем с лучшей точностью будет найден интеграл, но и тем больше времени будет затрачено на расчеты. По умолчанию `TOL=0.001`. Для того чтобы ускорить вычисления, можно установить большее значение `TOL`.

Совет

Если скорость расчетов имеет для вас принципиальное значение, например, при многократном вычислении интеграла внутри цикла, проявите осторожность, выбирая значение точности. Обязательно поэкспериментируйте на тестовом примере с характерной для ваших расчетов подынтегральной функцией. Посмотрите, как уменьшение константы `TOL` сказывается на погрешности интегрирования, вычислив интеграл для разных ее значений и выбрав оптимальное, исходя из соотношения точность/скорость вычислений.

Отдавайте себе отчет в том, что при вводе в редакторе Mathcad оператора численного интегрирования вы фактически создаете самую настоящую программу. Например, программой является первая строка листинга с рис. 4.2, просто основная ее часть скрыта от вашего взора разработчиками Mathcad. В большинстве случаев об этом не приходится специально задумываться, можно полностью положиться на Mathcad. Но иногда может потребоваться умение управлять параметрами этой программы, как мы уже рассмотрели на примере выбора константы `TOL`. Кроме нее, пользователь имеет возможность выбирать сам алгоритм численного интегрирования. Для этого:

1. Щелкните правой кнопкой мыши в любом месте на левой части вычисляемого интеграла.
2. В появившемся контекстном меню выберите один из имеющихся в наличии численных алгоритмов, например **Метод Ромберга** (Romberg) (рис. 4.3).

Обратите внимание, что перед тем как один из алгоритмов выбран впервые, как показано на рис. 4.3, флажок проверки в контекстном меню установлен возле пункта **Автовыбор** (AutoSelect). Это означает, что алгоритм определяется Mathcad, исходя из анализа пределов интегрирования и особенностей подынтегральной функции. Как только один из алгоритмов выбран, этот флажок сбрасывается, а избранный алгоритм отмечается точкой.

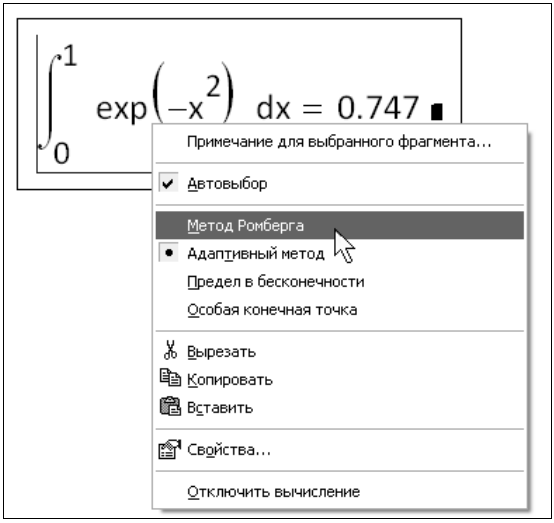


Рис. 4.3. Выбор алгоритма численного интегрирования производится при помощи контекстного меню

Разработчиками Mathcad запрограммированы четыре численных алгоритма интегрирования:

- **Метод Ромберга** (Romberg) — для большинства функций, не содержащих особенностей;
- **Адаптивный метод** (Adaptive) — для функций, быстро меняющихся на интервале интегрирования;

- **Предел в бесконечности** (Infinite Limit) — для интегралов с бесконечными пределами;
- **Особая конечная точка** (Singular Endpoint) — для интегралов с сингулярностью на конце (применяется модифицированный алгоритм Ромберга для функций, не определенных на одном или обоих концах интервала интегрирования).

Старайтесь все-таки оставить выбор численного метода за Mathcad, установив флажок **Автовыбор** (AutoSelect) в контекстном меню. Попробовать другой метод можно, например, чтобы сравнить результаты расчетов в специфических случаях, когда у вас закрадываются сомнения в их правильности.

Если подынтегральная функция "хорошая", т. е. не меняется на интервале интегрирования слишком быстро, не имеет особенностей и не обращается в бесконечность, то численное решение интеграла не принесет никаких неприятных сюрпризов.

4.1.3. О традиционных алгоритмах интегрирования

Прежде чем перейти к изложению метода численного интегрирования, реализованного в Mathcad, скажем несколько слов об основных принципах численного интегрирования. Исходя из геометрического смысла определенного интеграла функции `f(x)` как площади фигуры, образованной графиком этой функции и осью `X`, можно предложить самый простой способ интегрирования "хорошей" функции — применить формулу прямоугольников. С ее помощью площадь упомянутой искомой фигуры подсчитывается как сумма элементарных прямоугольников, множеством которых заменяется подынтегральная функция `f(x)`.

Иллюстрация метода прямоугольников приведена на рис. 4.4. Для подсчета интеграла `I` интервал интегрирования `[a,b]` разбивается на `N` отрезков. На каждом `i`-м отрезке `f(x)` заменяется прямоугольником с шириной `h` и высотой $f(x_i)$. Площадь каждого из этих элементарных прямоугольников составляет $h \cdot f(x_i)$, а их сумма `S` может считаться приближением к искомому интегралу `I`. Несложно показать, что при $N \to \infty$ множество элементарных прямоугольников стремится к искомой фигуре, образованной подынтегральной функцией, а значение $S \to I$, причем погрешность (отличие `S` от точного значения `I`) составляет $o(h^2)$.

Можно воспринимать смысл алгоритма прямоугольников в замене исходной подынтегральной функции другой, близкой к ней (в данном случае, кусочно-непрерывной) функцией, интеграл от которой легко подсчитать аналитически. Принцип более точных методов интегрирования как раз и состоит в интерполяции подынтегральной функции `f(x)` некоторой близкой зависимостью `y(x)` и расчете интеграла уже от этой функции. Важно, чтобы при этом, во-первых, интеграл от `y(x)` мог быть точно рассчитан аналитическими методами и, во-вторых, функция `f(x)` была бы по возможности ближе к `y(x)`, чтобы уменьшить погрешность.

Очевидно, что наиболее простой алгоритм заключается в интерполяции подынтегральной функции на каждом из `N` шагов интегрирования `f(x)` каким-либо полино-

мом y(x). Известно, что могут быть предложены различные пути построения интерполирующих полиномов, отличающихся, в частности, порядком. Например, *полиномы Лагранжа* строятся при интерполяции f(x) в n точках на каждом из N элементарных интервалов интегрирования. Семейство классических алгоритмов интегрирования в этом случае называется *методами Ньютона—Котеса*. Заметим, что при n=1 полиномом является прямая линия, и мы имеем метод трапеций; при n=2 интерполирующим полиномом на каждом шаге интегрирования будет квадратичная парабола, и мы получим *алгоритм Симпсона* и т. д.

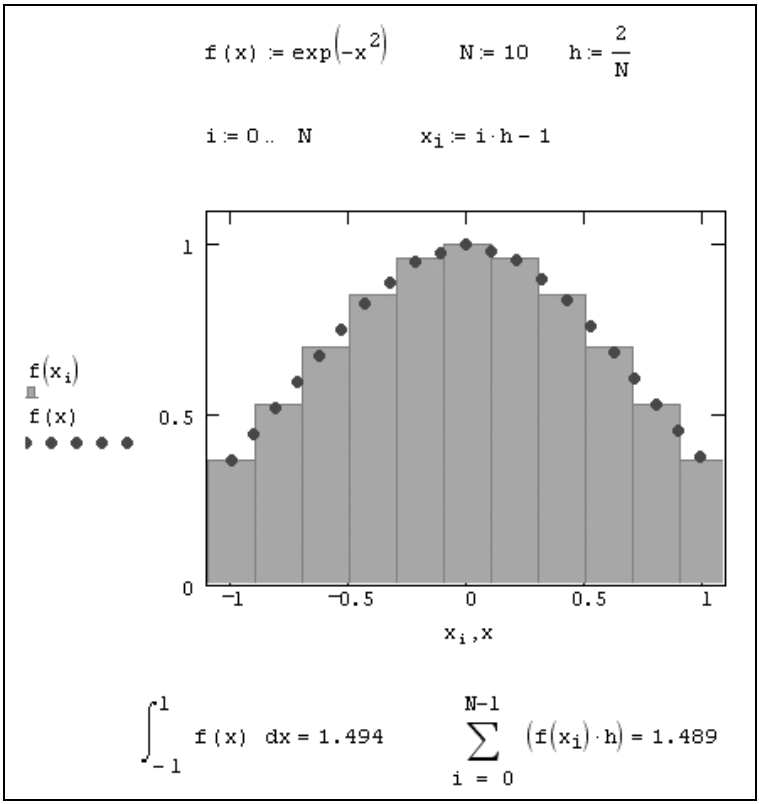


Рис. 4.4. Реализация алгоритма прямоугольников

Примечание

Дополнительные сведения об алгоритме полиномиальной интерполяции приведены в *главе 13.*

Недостатком перечисленных традиционных алгоритмов являются затруднения в количественной оценке погрешности. Аналитические формулы для погрешности включают, помимо множителя h^K, задающего, собственно, порядок аппроксимации метода, сомножитель, характеризующий величину производной (определенного высшего порядка) подынтегральной функции. Оценить ее значение при конкретных расчетах очень сложно и поэтому, соответственно, сложно вычислить суммарную

погрешность алгоритма. В то же время сведения о величине погрешности являются очень важными, и, чаще всего, желательно иметь их количественную оценку, чтобы оптимально выбрать число разбиений интервала интегрирования N.

Для апостериорной оценки погрешности можно, например, применять анализ зависимости S(N), подсчитанной для нескольких значений N (рис. 4.5). Зная, что, с одной стороны, S(N) изменяется по определенному степенному закону N^{-K}, а с другой — $S(N) \to I$ (к точному значению интеграла), можно довольно точно определить погрешность метода. На нижнем графике рис. 4.5 приведена зависимость погрешности от N (правда, в данном случае мы использовали для наглядности графика точное значение интеграла, которое в практических случаях, конечно, неизвестно). Именно

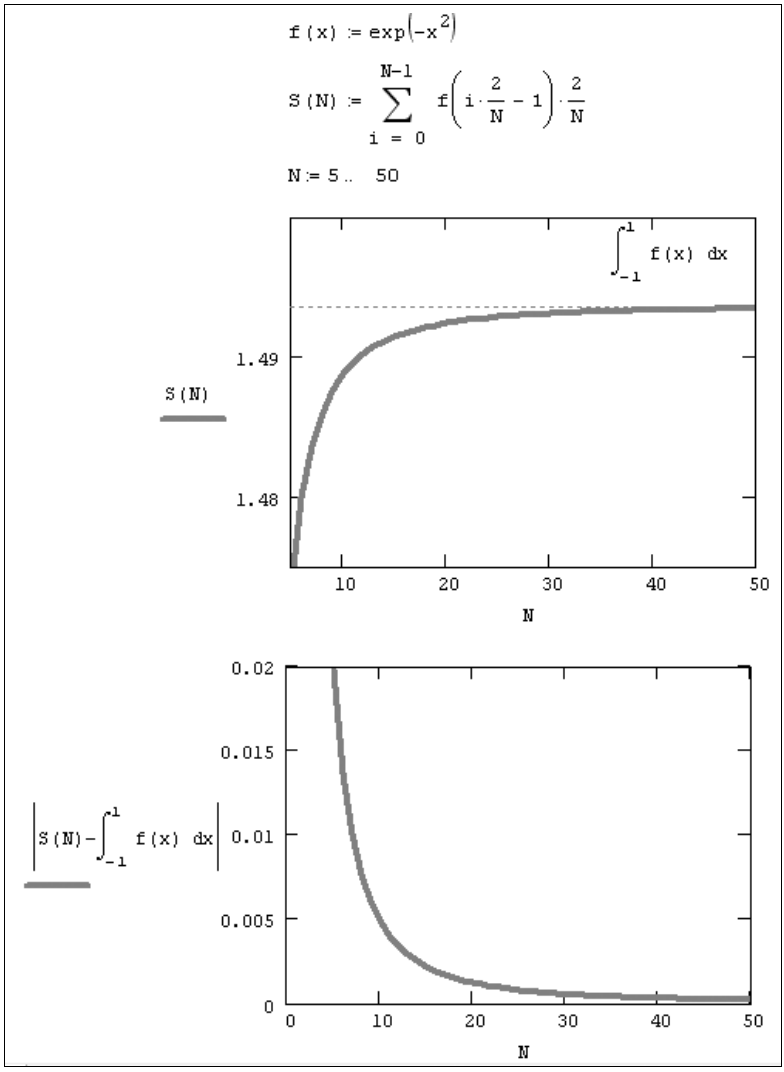


Рис. 4.5. Оценка погрешности алгоритма прямоугольников

с подобной процедурой и связан алгоритм расчета определенных интегралов, использованный в Mathcad, который будет представлен в следующем разделе.

ⓘ 4.1.4. Алгоритм Ромберга

После сделанных вводных замечаний приведем основные идеи итерационного *алгоритма Ромберга*, который применяется в системе Mathcad для выполнения операции численного интегрирования.

- Сначала строится несколько интерполирующих полиномов, которые заменяют на интервале интегрирования подынтегральную функцию `f(x)`. В качестве первой итерации полиномы вычисляются по 1, 2 и 4 интервалам. Например, как уже отмечалось выше, первый полином, построенный по 1 интервалу, — это просто прямая линия, проведенная через две граничные точки интервала интегрирования, второй — квадратичная парабола и т. д.
- Интеграл от каждого полинома с известными коэффициентами легко вычисляется аналитически. Таким образом, определяется последовательность интегралов от интерполирующих полиномов: $I_1, I_2, I_4, \ldots$ Например, по правилу трапеций $I_1=(b-a)\cdot(f(a)+f(b))/2$ и т. д.
- Из-за интерполяции по разному числу точек вычисленные интегралы $I_1, I_2, \ldots$ несколько отличаются друг от друга. Причем чем больше точек используется для интерполяции, тем интеграл от интерполяционного полинома ближе к искомому интегралу I, стремясь к нему в пределе бесконечного числа точек. Поэтому определенным образом осуществляется экстраполяция последовательности $I_1, I_2, I_4, \ldots$ до нулевой ширины элементарного интервала. Результат этой экстраполяции J принимается за приближение к вычисляемому интегралу.
- Осуществляется переход к новой итерации с помощью еще более частого разбиения интервала интегрирования, добавления нового члена последовательности интерполирующих полиномов и вычисления нового (N-го) приближения Ромберга J^N.
- Чем больше количество точек интерполяции, тем ближе очередное приближение Ромберга к вычисляемому интегралу и, соответственно, тем меньше оно отличается от приближения предыдущей итерации. Как только разница между двумя последними итерациями $|J^N-J^{N-1}|$ становится меньше погрешности TOL или меньше $TOL\cdot|J^N|$, итерации прерываются, и J^N появляется на экране в качестве результата интегрирования.

4.2. Неопределенный интеграл

Предыдущий раздел был посвящен проблеме поиска определенного интеграла, т. е. числового значения, равного площади фигуры, образованной графиком подынтегральной функции и осью x (см. рис. 4.4). Задача нахождения неопределенного интеграла намного сложнее, поскольку связана с поиском функции, производная от которой равна исходной подынтегральной функции. Решение этой задачи целиком возложено на символьный процессор Mathcad.

4.2.1. Символьное интегрирование

Для того чтобы аналитически проинтегрировать некоторую функцию, следует ввести с панели **Математический анализ** (Calculus) символ неопределенного интеграла, в появившемся в документе шаблоне заполнить местозаполнители и, наконец, ввести знак символьного равенства. В случае успеха по истечении некоторого времени расчетов справа от введенного выражения появится его аналитический результат (листинг 4.4). Если же функцию не удается проинтегрировать аналитически, введенное вами выражение будет просто продублировано (листинг 4.5).

Примечание

Помните, что при символьном интегрировании допускается использовать в выражениях, которые вы вводите, различные параметры. Если перед выражением вы нигде не определяли их значения, то (в случае успешных вычислений) символьный процессор Mathcad выдаст аналитическую зависимость результата интегрирования от этих параметров (как в листинге 4.4 от параметра a).

Листинг 4.4. Аналитическое вычисление неопределенного интеграла

$$\int \exp\left(-a\cdot x^2\right)\,dx \rightarrow \frac{1}{2}\cdot\frac{\pi^{\frac{1}{2}}}{a^{\frac{1}{2}}}\cdot \mathrm{erf}\left(a^{\frac{1}{2}}\cdot x\right)$$

Листинг 4.5. Аналитическое интегрирование невозможно

$$\int \exp\left(-a\cdot x^b\right)\,dx \rightarrow \int \exp\left(-a\cdot x^b\right)\,dx$$

4.2.2. Интегрирование при помощи меню

Для вычисления неопределенного интеграла от некоторого выражения по определенной переменной при помощи меню выделите в выражении переменную и вы-

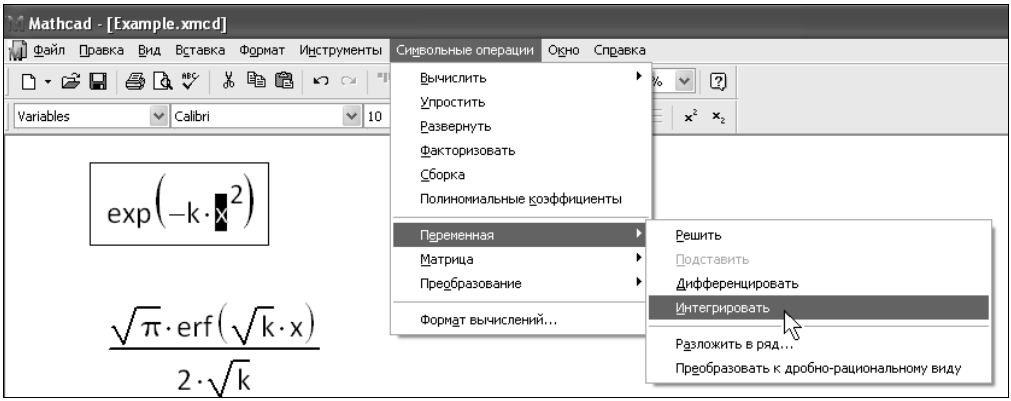


Рис. 4.6. Интегрирование выражения по переменной при помощи меню

полните команду **Символьные операции / Переменная / Интегрировать** (Symbolics / Variable / Integrate) (рис. 4.6).

Вычисленное аналитическое представление неопределенного интеграла появится ниже. При этом результат может содержать как встроенные в Mathcad функции, так и другие спецфункции, которые нельзя непосредственно рассчитать в Mathcad, но символьный процессор "умеет" выдавать их в качестве результата некоторых аналитических операций.

4.3. Интегралы специального вида

Завершим разговор о приемах интегрирования в среде Mathcad примерами вычислений в некоторых специальных случаях, которые довольно часто встречаются в самых разнообразных областях математики.

4.3.1. Интегралы с бесконечными пределами

Как мы уже говорили (*см. примечание 1 в разд. 4.1.1 и листинг 4.2*), для того чтобы вычислить определенный интеграл с одним или обоими бесконечными пределами, достаточно ввести, пользуясь панелью **Математический анализ** (Calculus), специально предусмотренный разработчиками символ бесконечности в нужные местозаполнители интервалов интегрирования.

4.3.2. Расходящиеся интегралы

Если интеграл расходится (равен бесконечности), то вычислительный процессор Mathcad может выдать сообщение об ошибке, выделив при этом оператор интегрирования, как обычно, красным цветом. Листинг 4.6 демонстрирует невозможность численного расчета интеграла $\int_0^\infty \frac{1}{\sqrt{x}}\,dx$ (нижняя строка листинга). Тем не менее символьный процессор справляется с этим интегралом, совершенно правильно находя его бесконечное значение (верхняя строка листинга 4.6).

Листинг 4.6. Вычисление расходящегося интеграла

$$\int_0^\infty \frac{1}{\sqrt{x}}\,dx \rightarrow \infty$$

$$\int_0^\infty \frac{1}{\sqrt{x}}\,dx = \blacksquare$$

При попытке численного решения задачи листинга 4.6 методом, отличным от алгоритма вычисления интегралов **Предел в бесконечности** (Infinite Limit), получится неверное решение (листинг 4.7). А именно, вместо бесконечности будет выдано

большое, но конечное число, немного не дотягивающее до численной бесконечности, являющейся для вычислительного процессора просто большим числом 10^{307}. Отметим, что Mathcad в режиме автоматического выбора алгоритма **Автовыбор** (AutoSelect) предлагает для интегралов с бесконечным пределом именно алгоритм **Предел в бесконечности** (Infinite Limit).

Листинг 4.7. Плохо выбранный численный алгоритм (в данном случае, адаптивный) неверно находит расходящийся интеграл

$$\int_0^{\infty} \frac{1}{\sqrt{x}}\, dx = 6.325 \times 10^{153}$$

4.3.3. Интеграл с переменным пределом

Символьный процессор предоставляет замечательные возможности аналитического вычисления интегралов, в том числе зависящих от параметров. Особую важность имеет вычисление интеграла *с переменным пределом* (верхним или нижним), для которого один из пределов интегрирования является переменной, отличной от переменной интегрирования (листинг 4.8).

Листинг 4.8. Аналитическое вычисление интеграла с переменным верхним пределом

$$\int_0^{z} \frac{1}{\sqrt{x}}\, dx \rightarrow 2 \cdot z^{\frac{1}{2}}$$

4.3.4. Кратные интегралы

Кратным называется интеграл функции многих переменных, берущийся по нескольким переменным.

Для того чтобы вычислить кратный интеграл:

1. Введите, как обычно, оператор интегрирования.
2. В соответствующих местозаполнителях введите имя первой переменной интегрирования и пределы интегрирования по этой переменной.
3. На месте ввода подынтегральной функции введите еще один оператор интегрирования (рис. 4.7).
4. Точно так же введите вторую переменную, пределы интегрирования и подынтегральную функцию (если интеграл двукратный) или следующий оператор интегрирования (если более чем двукратный) и т. д., пока выражение с многократным интегралом не будет введено окончательно.

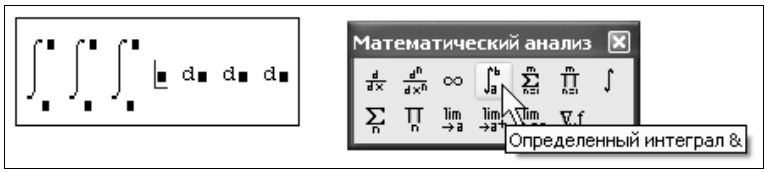


Рис. 4.7. Ввод нескольких операторов интегрирования для расчета кратного интеграла

Пример символьного и численного расчета двукратного интеграла в бесконечных пределах приведен в листинге 4.9. Обратите внимание, что символьный процессор "угадывает" точное значение интеграла π, а вычислительный определяет его приближенно и выдает в виде числа `3.142`.

Листинг 4.9. Символьное и численное вычисления кратного интеграла

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\left|x^2+y^2\right|}\,dx\,dy \rightarrow \pi$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-\left|x^2+y^2\right|}\,dx\,dy = 3.142$$

Внимание!

Аккуратнее вводите в редакторе Mathcad кратные интегралы, если они имеют различные пределы интегрирования по разным переменным. Не перепутайте пределы, относящиеся к разным переменным. Если вы имеете дело с такого рода задачами, обязательно разберитесь с листингом 4.10, в котором символьный процессор вычисляет двукратный интеграл. В первой строке пределы интегрирования `[a, b]` относятся к переменной `y`, а во второй строке — к переменной `x`.

Листинг 4.10. Символьное вычисление кратных интегралов

$$\int_{a}^{b}\int_{-1}^{1} x+y^3\,dx\,dy \rightarrow \frac{1}{2}\cdot b^4-\frac{1}{2}\cdot a^4$$

$$\int_{a}^{b}\int_{-1}^{1} x+y^3\,dy\,dx \rightarrow b^2-a^2$$

4.3.5. Пример: длина дуги кривой

В заключение приведем пример использования вычислительного процессора Mathcad для расчета длины участка кривой, задаваемой некоторой функцией `f(x)` в промежутке между двумя значениями ее аргумента `a` и `b` (рис. 4.8).

Решение данной простой задачи математического анализа приведено в листинге 4.11, причем формула, по которой рассчитывается длина дуги, приведена в третьей (последней) строке листинга. Обратите внимание, что для получения результата необходимо применить и операцию численного интегрирования и дифференцирования.

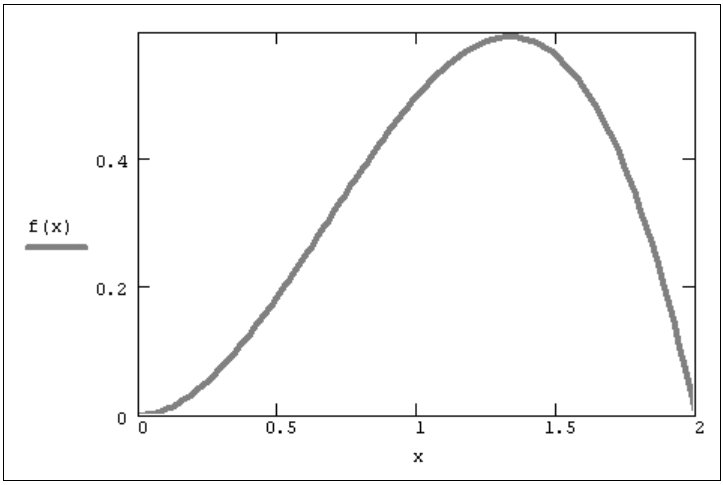


Рис. 4.8. График функции определяет дугу некоторой длины (продолжение листинга 4.11)

Листинг 4.11. Расчет длины дуги кривой

$$f(x) := x^2 - \frac{x^3}{2}$$

$a := 0 \qquad b := 2$

$$\int_a^b \sqrt{1 + \left(\frac{d}{dx} f(x)\right)^2}\, dx = 2.42$$

4.4. Интеграл Фурье

Обратимся теперь к характерным проблемам вычислительной математики, связанным с (аналитическим или численным) вычислением интегралов определенного вида. Благодаря обширному применению для вычисления таких интегралов разработаны специфические алгоритмы, несравненно более эффективные, чем традиционные (*см. разд. 4.1.4*), причем некоторые из них имеются в арсенале Mathcad в форме встроенных функций вычислительного процессора и соответствующих операций символьного процессора.

Наиболее широко распространенным интегральным преобразованием является преобразование Фурье, которое представляет функцию f(x) в виде интеграла по гармоническим функциям, называемого *интегралом Фурье*:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) \cdot \exp(-i\omega x)\, dx .$$

Функция F(ω) называется также *преобразованием Фурье*, или *Фурье-спектром* исходной функции f(x). Ее аргумент ω имеет смысл частоты соответствующей гармонической составляющей f(x). Важно отметить, что функция, выражающая преобразование Фурье, комплексна, даже если f(x) является действительной.

ⓘ 4.4.1. Об интегральных преобразованиях функций

Вообще говоря, интегральные преобразования по определению ставят в соответствие некоторой функции f(x) другую функцию от другого аргумента F(ω). Причем это соответствие f(x)→F(ω) задается интегральной зависимостью. Символьный процессор Mathcad позволяет осуществлять три вида интегральных преобразований функций — преобразование Фурье, Лапласа и Z-преобразование. Наряду с прямыми преобразованиями имеется возможность совершать любое из этих трех обратных преобразований, т. е. F(ω)→f(x).

Аналитически все интегральные преобразования выполняются аналогично символьному интегрированию (*см. разд. 4.2.2*). Для вычисления преобразования выражения выделяется переменная, по которой будет осуществляться преобразование, и затем выбирается соответствующий пункт меню. Преобразования с применением оператора символьного вывода используются с одним из соответствующих ключевых слов, вслед за которым требуется указать имя нужной переменной.

Приведем примеры символьного расчета каждого из трех интегральных преобразований, а также расскажем о численных методах Фурье- и вейвлет-преобразований.

4.4.2. Аналитическое преобразование Фурье

Аналитический расчет преобразования Фурье при помощи меню показан на рис. 4.9, для чего используется команда меню **Символьные операции / Преобразование / Фурье** (Symbolics / Transform / Fourier). В листинге 4.12 приведены два примера вычисления прямого преобразования Фурье с применением ключевого слова fourier и оператора символьного вывода →. Листинг 4.13 иллюстрирует применение обратного преобразования Фурье и последующее прямое преобразование полученного выражения, в результате которого получается исходная функция.

Внимание!

Помните о том, что результаты символьных преобразований могут включать спецфункции, которые не являются встроенными функциями Mathcad и поэтому не могут впоследствии использоваться в вычислениях. Их имена представляют собой не более, чем текстовое сообщение.

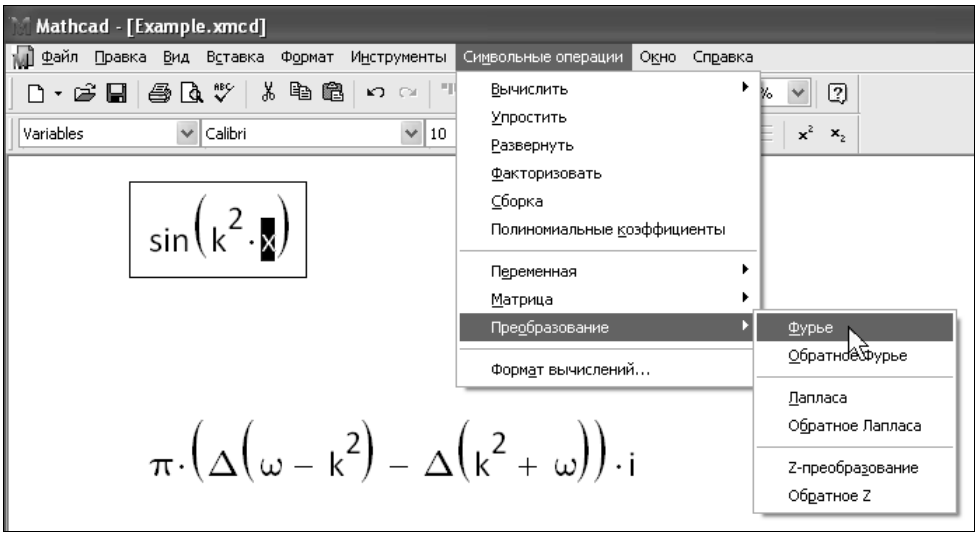


Рис. 4.9. Расчет Фурье-преобразования при помощи меню

Листинг 4.12. Примеры прямого преобразования Фурье

$$\cos\left(k^2\cdot x\right)\ \text{fourier}, x \rightarrow \pi\cdot\Delta\left(\omega - k^2\right) + \pi\cdot\Delta\left(\omega + k^2\right)$$

$$\cos\left(k^2\cdot x\right)\ \text{fourier}, k \rightarrow \left(\frac{\pi}{|x|}\right)^{\frac{1}{2}}\cdot\exp\left(\frac{-1}{4}\cdot\frac{\omega^2}{|x|}\right)$$

Листинг 4.13. Обратное и прямое преобразования Фурье

$$\frac{1}{\omega}\ \text{invfourier}, \omega \rightarrow \frac{1}{2}\cdot i\cdot\left(\Phi(t) - \Phi(-t)\right)$$

$$\frac{-1}{2}\cdot i\cdot\left(-\Phi(t) + \Phi(-t)\right)\ \text{fourier}, t \rightarrow \frac{1}{\omega}$$

4.4.3. Дискретное преобразование Фурье

В предыдущем разделе рассказывалось о возможностях символьного процессора Mathcad, позволяющего осуществить аналитическое преобразование Фурье функции, заданной формулой. Между тем огромный пласт задач вычислительной математики связан с расчетом интегралов Фурье для функций, либо заданных таблично (например, представляющих собой результаты какого-либо эксперимента), либо функций, проинтегрировать которые аналитически не удается. В данном случае вместо символьных преобразований приходится применять численные методы интегрирования, связанные с дискретизацией подынтегральной функции и называемые потому *дискретным Фурье-преобразованием*.

В численном процессоре Mathcad дискретное преобразование Фурье реализовано при помощи популярнейшего алгоритма *быстрого преобразования Фурье* (сокращенно БПФ). Этот алгоритм реализован в нескольких встроенных функциях Mathcad, различающихся только нормировками:

- `fft(y)` — вектор прямого преобразования Фурье;
- `FFT(y)` — вектор прямого преобразования Фурье в другой нормировке;
- `ifft(ω)` — вектор обратного преобразования Фурье;
- `IFFT(ω)` — вектор обратного преобразования Фурье в другой нормировке:
  - `Y` — вектор действительных данных, взятых через равные промежутки значений аргумента;
  - ω — вектор действительных данных Фурье-спектра, взятых через равные промежутки значений частоты.

Примечание 1

Аргумент прямого Фурье-преобразования, т. е. вектор `y`, должен иметь ровно 2^n элементов (`n` — целое число). Результатом является вектор с $1+2^{n-1}$ элементами. И наоборот, аргумент обратного Фурье-преобразования должен иметь $1+2^{n-1}$ элементов, а его результатом будет вектор из 2^n элементов. Если число данных не совпадает со степенью `2`, то необходимо дополнить недостающие элементы нулями.

15 Примечание 2

В Mathcad 15 появились две новые функции для расчета Фурье-преобразования — `dft` и `idft`. Их применение аналогично функциям `fft` и `ifft` соответственно, но работают они значительно быстрее и не требуют того, чтобы размер данных был кратным числу 2.

В листинге 4.14 показан пример расчета Фурье-спектра для модельной функции `f(x)`, представляющей собой сумму двух синусоид разной амплитуды (верхний график на рис. 4.10). Расчет проводится по `N=128` точкам, причем предполагается, что интервал дискретизации данных y_i равен `h`. В предпоследней строке листинга корректно определяются соответствующие значения частот Ω_i, а в последней применяется встроенная функция `FFT`. Полученный график Фурье-спектра показан на рис. 4.10 (снизу). Обратите внимание, что результаты расчета представляются в виде его модуля, поскольку сам спектр, как уже отмечалось, является комплексным. Очень полезно сравнить полученные амплитуды и местоположение пиков спектра с определением синусоид в начале листинга.

Листинг 4.14. Дискретное преобразование Фурье (алгоритм БПФ) модельного сигнала

$$f(x) := 0.5\cdot\sin(2\pi\cdot 0.1\cdot x) + 1\cdot\sin(2\pi\cdot 0.5\cdot x)$$

$$L := 100 \qquad N := 128 \qquad h := \frac{L}{N}$$

$$i := 0 .. N-1$$

$x_i := i \cdot h$ $\quad y_i := f(x_i)$ $\quad \Omega_i := \frac{i}{L}$

$F := FFT(y)$

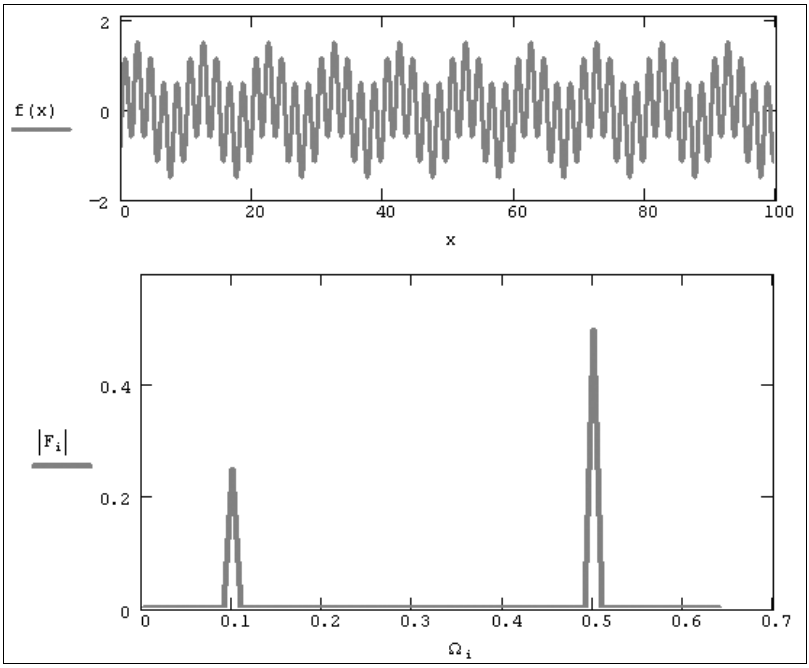


Рис. 4.10. Модельная функция и ее преобразование Фурье (продолжение листинга 4.14)

4.4.4. Преобразование Фурье комплексных данных

Алгоритм быстрого преобразования Фурье для комплексных данных встроен в соответствующие функции, в название которых входит литера "C":

- `cfft(y)` — вектор прямого комплексного преобразования Фурье;
- `CFFT(y)` — вектор прямого комплексного преобразования Фурье в другой нормировке;
- `icfft(y)` — вектор обратного комплексного преобразования Фурье;
- `ICFFT(ω)` — вектор обратного комплексного преобразования Фурье в другой нормировке:
  - `y` — вектор данных, взятых через равные промежутки значений аргумента;
  - ω — вектор данных Фурье-спектра, взятых через равные промежутки значений частоты.

Функции действительного преобразования Фурье используют тот факт, что в случае действительных данных спектр получается симметричным относительно нуля, и выводят только его половину (*см. разд. 4.4.3*).

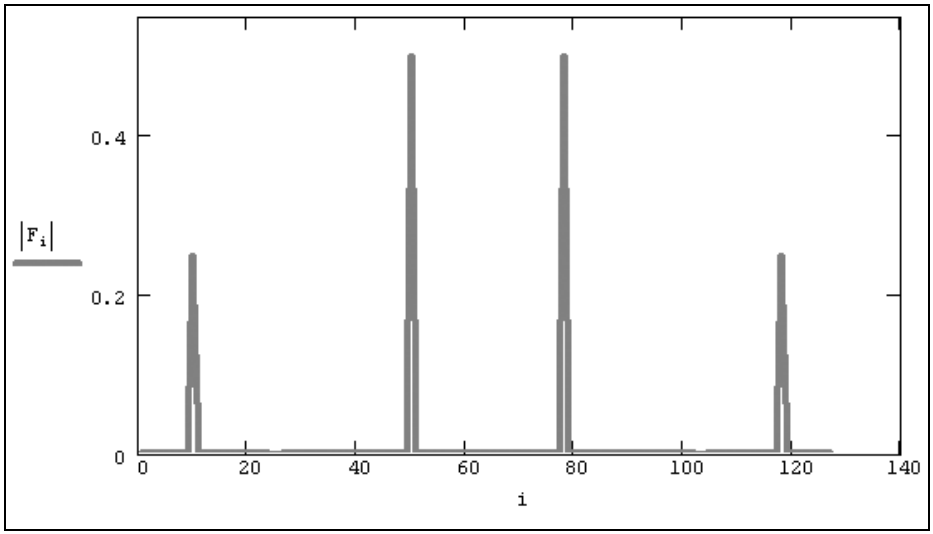


Рис. 4.11. Комплексное преобразование Фурье (продолжение листинга 4.14)

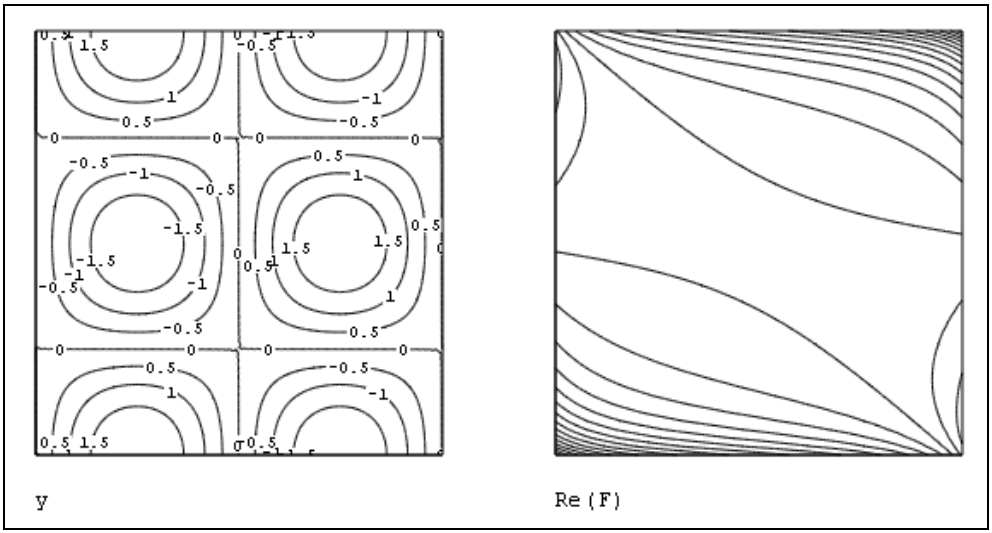


Рис. 4.12. Данные (слева) и их Фурье-спектр (справа) (продолжение листинга 4.15)

Поэтому, в частности, по 128 действительным данным получалось всего 65 точек спектра Фурье. Если к тем же данным применить функцию комплексного преобразования Фурье (рис. 4.11), то получится вектор из 128 элементов. Сравнивая рис. 4.10 и 4.11, можно уяснить соответствие между результатами действительного и комплексного Фурье-преобразования.

4.4.5. Двумерное преобразование Фурье

В Mathcad имеется возможность вычислять не только одномерное преобразование Фурье функции `f(x)`, но и двумерное преобразование функции двух переменных `f(x,y)`. Иными словами, допускается применять встроенные функции комплексного дискретного преобразования Фурье не только к одномерным, но и к двумерным массивам, т. е. матрицам. Соответствующий пример приведен в листинге 4.15 и на рис. 4.12 в виде графика поверхности исходных данных (верхний график) и рассчитанного двумерного Фурье-спектра (нижний график).

Листинг 4.15. Двумерное дискретное преобразование Фурье

$N := 64$

$i := 0 .. N-1 \qquad j := 0 .. N-1$

$y_{i,j} := \sin\left(\frac{i+j}{10}\right) + \sin\left(\frac{i-j}{10}\right)$

$F := \mathrm{CFFT}(y)$

$F := \mathrm{submatrix}(F, 7, N-7, 7, N-7)$

4.4.6. Пример: артефакты дискретного Фурье-преобразования

При численном нахождении преобразования Фурье следует очень внимательно относиться к таким важнейшим параметрам, как объем выборки (`xMAX`) и интервал дискретизации (Δ). Соотношение этих двух величин определяет диапазон частот (Ω_0, Ω_N), для которых возможно вычисление значений Фурье-спектра, называемых соответственно *граничной частотой* и *частотой Найквиста*. Граничная частота Ω_0 определяет нижнюю, а частота Найквиста Ω_N — верхнюю границу аргумента вычисленного спектра.

В этой связи хотелось бы обратить внимание на три типичные опасности, которые могут подстерегать неподготовленного исследователя при расчете дискретного Фурье-преобразования и быть для него весьма неожиданными.

Влияние конечности интервала выборки

Во-первых, следует обратить внимание на само определение преобразования Фурье как интеграла с бесконечными пределами. Его численное отыскание подразумевает принципиальную ограниченность интервала интегрирования (просто в силу конечности объема выборки). Поэтому самым очевидным несоответствием будет поиск, вообще говоря, другого интеграла, отличного от интеграла Фурье. Влияние конечности интервала выборки проявляется главным образом на искажении его низкочастотной области. В качестве примера приведем Фурье-спектр гармонической функции с частотой `0.015`. Для его расчета достаточно заменить в листинге 4.14

(*см. разд. 4.4.3*) первую строку на равенство `f(x):=sin(2·π·0.015·x)`. Соответствующий Фурье-спектр изображен на рис. 4.13 (сверху — в обычном, а снизу — в более крупном масштабе) и демонстрирует не совсем правильное поведение в низкочастотной области. Как видно, спектр содержит довольно широкий максимум вместо одиночного пика, как было в случае средних частот сигнала на рис. 4.10.

Примечание

Если быть точным, вместо спектра некоторой функции `f(x)` дискретное преобразование Фурье подразумевает вычисление спектра другой функции `f(x)·Φ(x)`, где `Φ(x)` — это функция-ступенька, равная единице в пределах расчетного интервала и нулю за его пределами. В частотной области это соответствует операции *свертки* означенных двух функций, что, конечно, искажает (неизвестный) точный спектр `f(x)`.

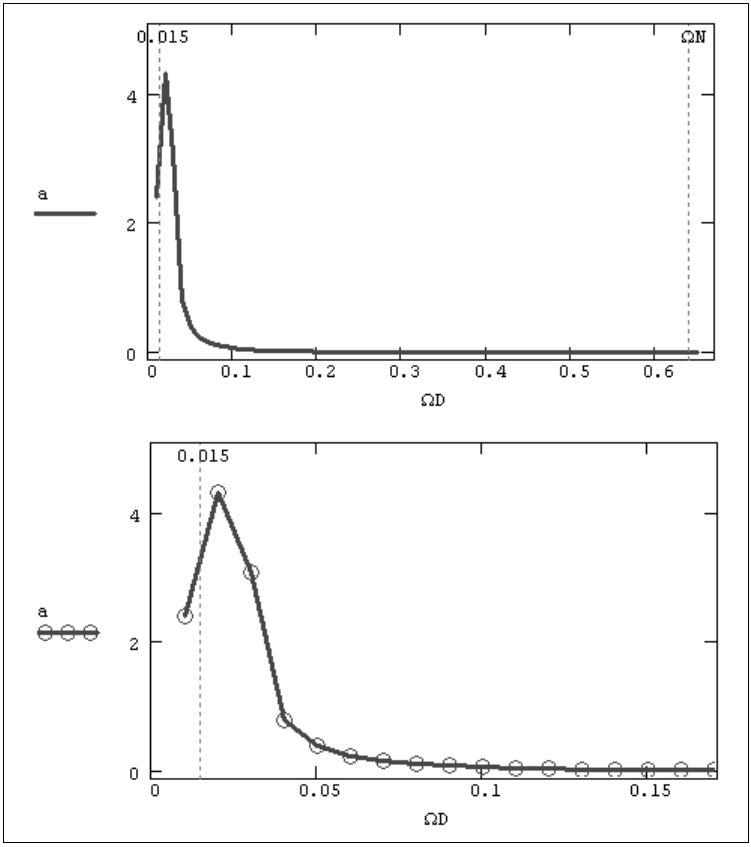


Рис. 4.13. Иллюстрация влияния конечности выборки на расчет низкочастотной части Фурье-спектра

Сдвиг ноль-линии

Еще одним, наиболее ярким, проявлением вредного влияния конечности интервала выборки может служить расчет Фурье-преобразования суммы гармонического сиг-

нала и константы (рис. 4.14). Для того чтобы получить данный рисунок, достаточно еще слегка модифицировать первую строку листинга 4.14 (*см. разд. 4.4.3*), касающуюся определения функции, добавив к ней `1: f(x):=sin(2·π·0.015·x)+1`.

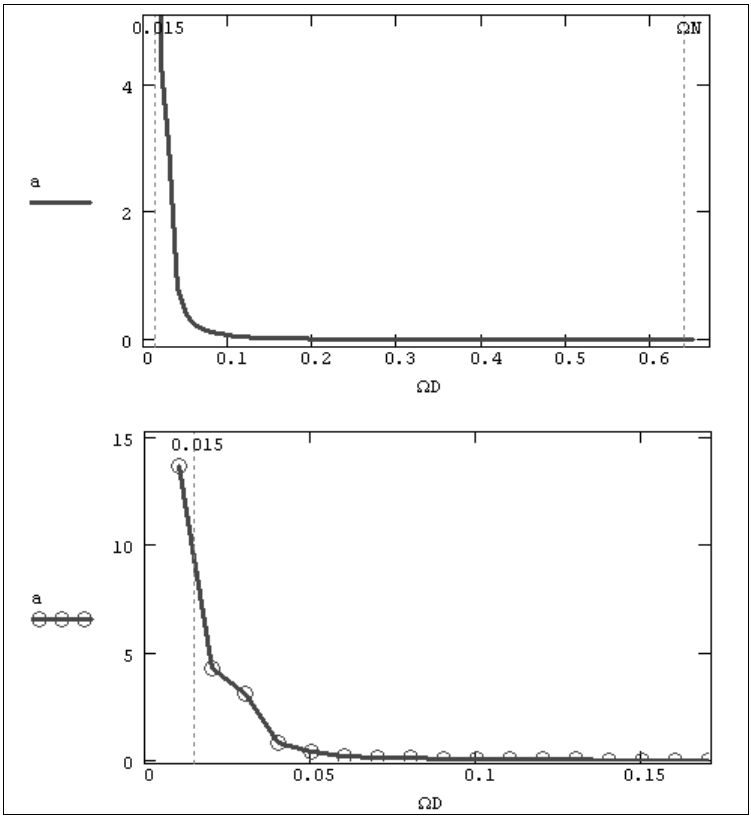


Рис. 4.14. Фурье-спектр суммы гармонического сигнала и константы (влияние конечности выборки)

Сравнивая рис. 4.13 и 4.14, несложно догадаться, почему так разительно изменился вид спектра в низкочастотной области. Пугающий рост спектра на левом крае частотного интервала объясняется совокупностью двух факторов: конечности выборки и добавлением к сигналу ненулевой постоянной составляющей (так называемым сдвигом *ноль-линии*). Сумма сигнала и константы определяет соответствующее влияние на вычисленный спектр, который также оказывается (просто в силу линейности операции интегрирования) суммой спектров сигнала и ступенчатой функцией (равной той самой константе внутри расчетного интервала и нулю за его пределами).

Избавиться от искажений, вызванных сдвигом ноль-линии, довольно просто. Достаточно (до Фурье-преобразования) вычислить среднее значение выборки и затем вычесть его из каждого элемента выборки. Если после этой операции вычислить Фурье-спектр, то он окажется примерно таким, как показано на рис. 4.13.

Маскировка частот

Еще один классический пример ошибочного расчета Фурье-спектра связан с возможным присутствием в сигнале гармоник с частотой, превышающей частоту Найквиста, в данном примере $\Omega_N=0.64$. Иллюстрация эффекта, называемого *"маскировкой частот"*, приведена на рис. 4.15, который содержит расчет спектров трех

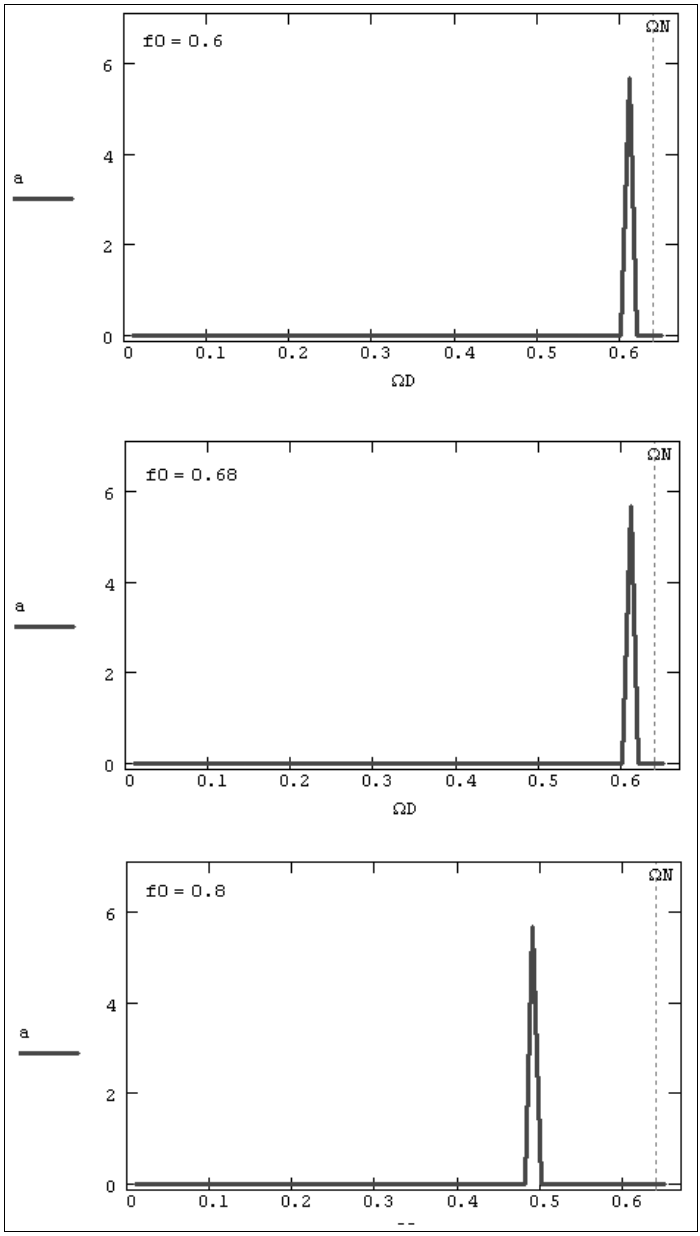


Рис. 4.15. Расчеты Фурье-спектров гармонических сигналов с разной частотой ("маскировка частот")

различных синусоидальных сигналов с разной частотой `f0`. Первый спектр сигнала с частотой, меньшей частоты Найквиста, вычислен верно, а вот два остальных спектра показывают, что в случае превышения частоты Найквиста в спектре начинают присутствовать "лишние" пики. Появление артефактов спектра связано с тем, что дискретных отсчетов начинает не хватать для того, чтобы прописать высокочастотные гармоники с достаточной информативностью.

4.5. Другие интегральные преобразования

Рассмотрим в завершение главы, посвященной интегрированию, еще три преобразования, которые часто применяются помимо интеграла Фурье. Отметим, что преобразование Лапласа и Z-преобразование встречаются в практических задачах вычислительной математики относительно редко, а вот вейвлет-преобразование, теория которого появилась относительно недавно, постепенно выходит на лидирующие позиции в проблемах обработки данных.

4.5.1. Преобразование Лапласа

Преобразованием Лапласа называют интеграл от `f(x)` следующего вида:

$$F(s) = \int_0^\infty f(x) \cdot \exp(-sx)\, dx.$$

Рассчитывается преобразование Лапласа совершенно аналогично Фурье-преобразованию (*см. разд. 4.4*). Примеры преобразования Лапласа приведены в листинге 4.16 и на рис. 4.16.

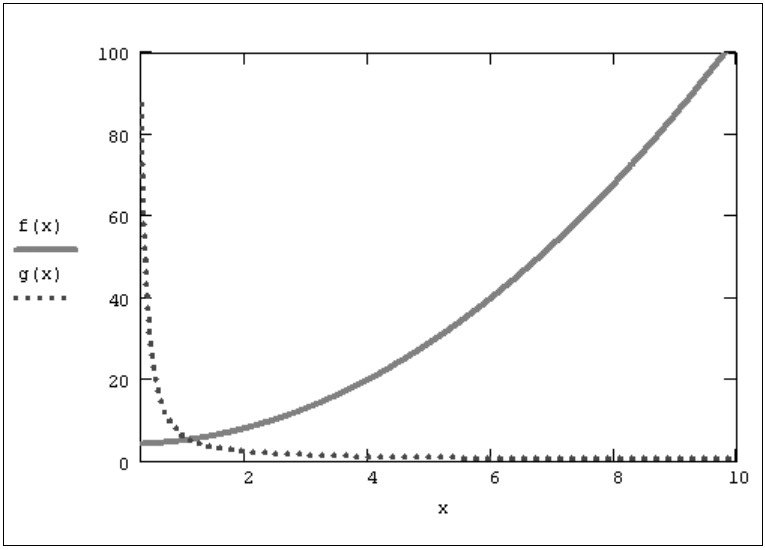


Рис. 4.16. Прямое и обратное преобразование Лапласа (продолжение листинга 4.16)

Листинг 4.16. Двумерное преобразование Лапласа

$f(x) := x^2 + 4$

$g(s) := f(x)\ \text{laplace}, x \rightarrow \frac{2}{s^3} + \frac{4}{s}$

$g(s)\ \text{invlaplace}, s \rightarrow t^2 + 4$

4.5.2. Z-преобразование

Z-преобразование функции `f(x)` определяется не интегралом, а бесконечной суммой следующего вида:

$$F(z) = \sum_{n=0}^{\infty} f(n) \cdot z^{-n}.$$

Пример Z-преобразования приведен в листинге 4.17, а его результаты — на рис. 4.17.

Листинг 4.17. Прямое и обратное Z-преобразования

$f(x) := x^2 + 4$

$g(z) := f(x)\ \text{ztrans}, x \rightarrow z \cdot \frac{\left|-7 \cdot z + 5 + 4 \cdot z^2\right|}{(z-1)^3}$

$g(z)\ \text{invztrans}, z \rightarrow 4 + n^2$

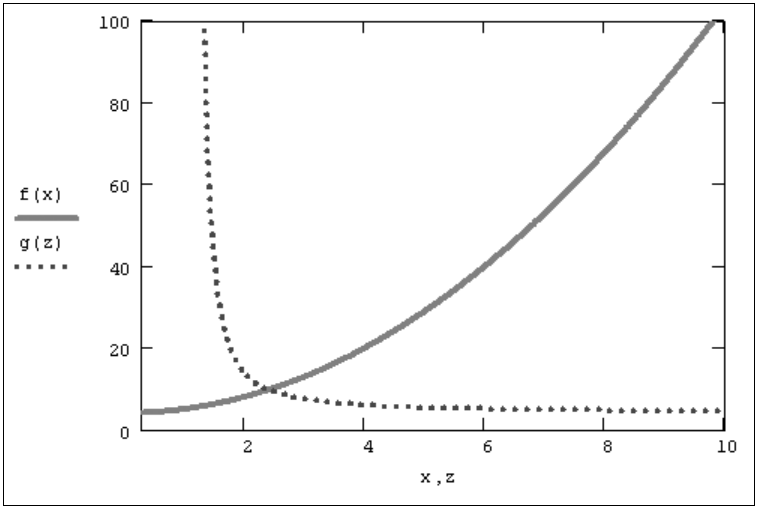


Рис. 4.17. Прямое и обратное Z-преобразования (продолжение листинга 4.17)

4.5.3. Вейвлет-преобразование

В последнее время возрос интерес к другим интегральным преобразованиям, в частности к *вейвлет-преобразованию* (или *дискретному волновому преобразованию*). Оно применяется главным образом для анализа нестационарных сигналов и для многих задач подобного рода оказывается более эффективным, чем преобразование Фурье. Основным отличием вейвлет-преобразования является разложение данных не по синусоидам (как для преобразования Фурье), а по другим функциям, называемым *вейвлетобразующими*. Вейвлетобразующие функции, в противоположность бесконечно осциллирующим синусоидам, локализованы в некоторой ограниченной области своего аргумента, а вдали от нее равны нулю или ничтожно малы. Пример такой функции, называемой "мексиканской шляпой", показан на рис. 4.18.

По своему математическому смыслу вейвлет-спектр имеет не один аргумент, а два. Помимо частоты, вторым аргументом `b` является место локализации вейвлетобразующей функции. Поэтому `b` имеет ту же размерность, что и `x`.

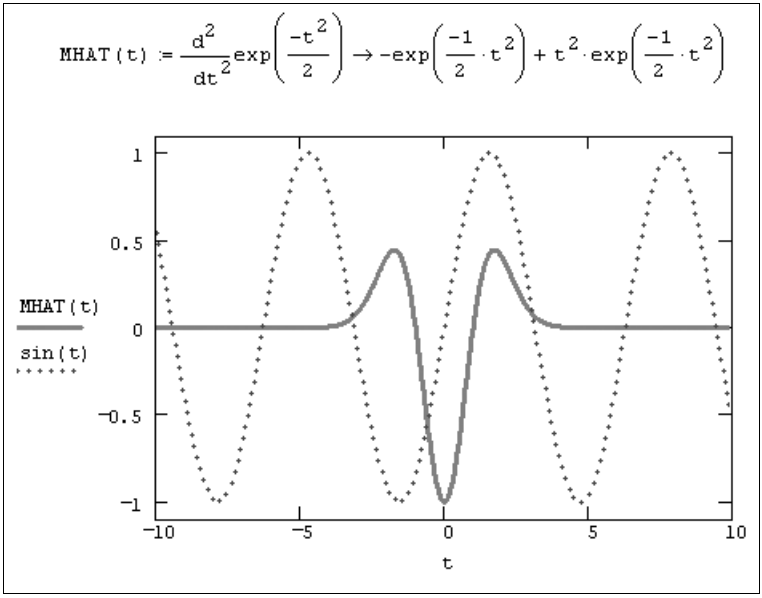


Рис. 4.18. Сравнение синусоиды и вейвлетобразующей функции

Встроенная функция вейвлет-преобразования

Mathcad имеет одну встроенную функцию для расчета вейвлет-преобразования на основе вейвлетобразующей функции Добеши:

- ❒ `wave(y)` — вектор прямого вейвлет-преобразования Добеши;
- ❒ `iwave(v)` — вектор обратного вейвлет-преобразования Добеши:
  - `y` — вектор данных, взятых через равные промежутки значений аргумента;
  - `v` — вектор данных вейвлет-спектра.

Аргумент функции вейвлет-преобразования, т. е. вектор y, должен так же, как и в преобразовании Фурье, иметь ровно 2^n элементов (n — целое число). Результатом функции wave является вектор, скомпонованный из нескольких коэффициентов c двухпараметрического вейвлет-спектра. Особенности использования функции wave иллюстрируются листингом 4.18, где в качестве модельной функции взята сумма двух синусоид, график которой был изображен на рис. 4.10. Результаты вычислений вейвлет-спектра Добеши представлены в виде трех семейств его коэффициентов на рис. 4.19.

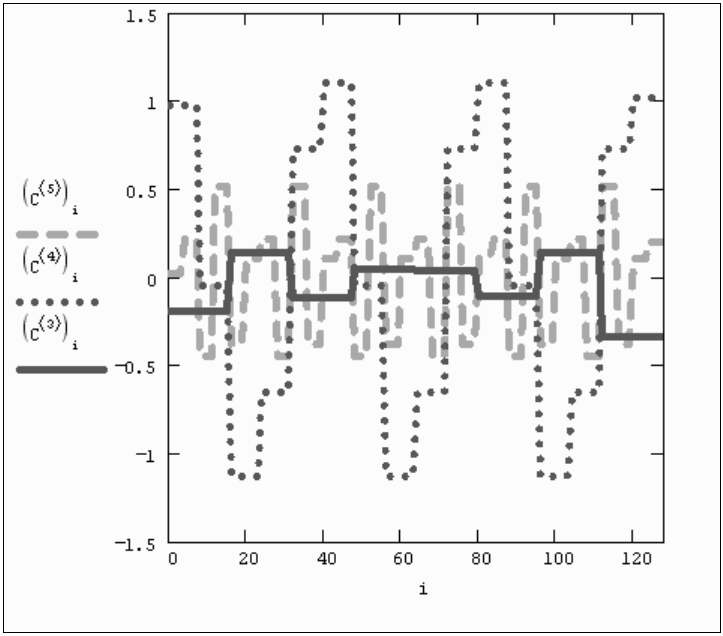


Рис. 4.19. Вейвлет-спектр Добеши модельного сигнала (продолжение листинга 4.18)

Листинг 4.18. Вычисление вейвлет-спектра Добеши модельного сигнала

```
f(t) := 0.5·sin(2π·0.1·t) + 1·sin(2π·0.5·t)
N := 128
i := 0 .. N − 1
Y_i := f(i)
W := wave(Y)
coeffs(level) := submatrix(W, 2^level, 2^(level+1) − 1, 0, 0)
Nlevels := ln(N)/ln(2) − 1
Nlevels = 6
```

$k := 1 .. \ Nlevels$

$C_{i,k} := coeffs(k)_{floor\left(\frac{i \cdot 2^k}{N}\right)}$

Программирование других вейвлет-преобразований

Наряду со встроенной функцией `wave`, Mathcad снабжен пакетом расширения для осуществления вейвлет-анализа. Пакет расширения содержит большое число дополнительных встроенных функций, имеющих отношение к вейвлет-преобразованиям. Обзор пакетов расширения выходит за рамки данной книги, поэтому ограничимся простым упоминанием об этой возможности.

Примечание

Дополнительную информацию об использовании данных встроенных функций можно найти в соответствующей электронной книге, которую можно открыть при помощи меню **Справка / Электронные книги / Вейвлет-анализ** (Help / E-Books / Wavelets extension pack).

Помимо встроенной функции вейвлет-спектра Добеши и возможностей пакета расширения, допускается непосредственное программирование алгоритмов пользователя для расчета вейвлет-спектров. Оно сводится к аккуратному численному расчету соответствующих семейств интегралов. Один из примеров такой программы приведен в листинге 4.19. Анализу подвергается функция, составленная из суммы двух синусов, а график двухпараметрического спектра `C(a,b)` выведен на рис. 4.20 в виде привычных для вейвлет-анализа линий уровня на плоскости `(a,b)`.

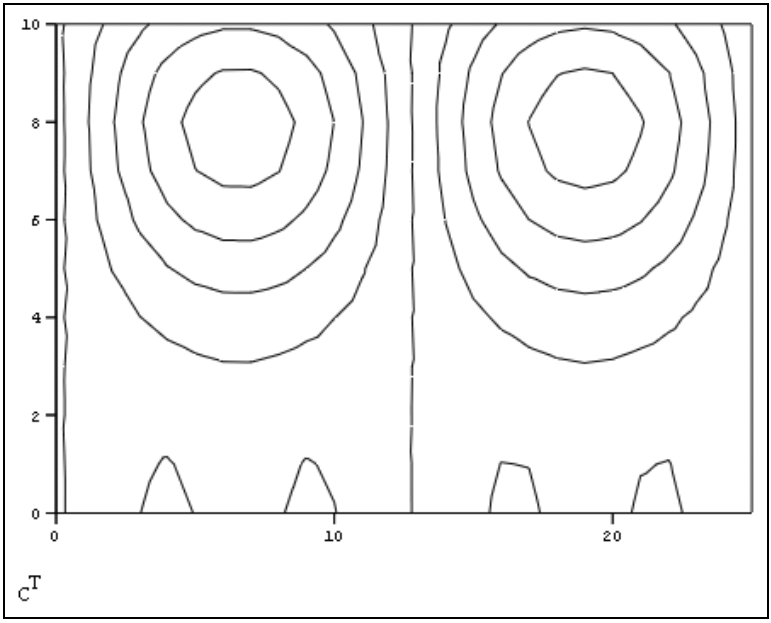


Рис. 4.20. Вейвлет-спектр модельного сигнала на основе "мексиканской шляпы" (продолжение листинга 4.19)

Примечание

Программа листинга очень проста, но исключительно далека от хорошей в смысле быстродействия. Каждый интеграл вычисляется независимо, без использования методов ускорения, типа применяемых в алгоритме БПФ. Однако простые приемы программирования вполне доступно раскрывают математический смысл вейвлет-преобразования.

Листинг 4.19. Вычисление вейвлет-спектра на основе "мексиканской шляпы"

$$f(t) := 1.0 \cdot \sin\left(2\pi \cdot 0.02 \cdot t\right) + 0.3 \cdot \sin\left(2\pi \cdot 0.1 \cdot t\right)$$

$$MHAT(t) := \frac{d^2}{dt^2} \exp\left(\frac{-t^2}{2}\right)$$

$$N := 256$$

$$W(a, b) := \int_{-N}^{N} f(t) \cdot MHAT\left(\frac{t-b}{a}\right) dt$$

$$b := 0, 1 .. \frac{N}{10}$$

$$i := 0 .. 10$$

$$a_i := \frac{(i+15)^4}{2 \times 10^4}$$

$$C_{i,b} := W\left(a_i, 2 \cdot b - \frac{N}{10}\right)$$

ГЛАВА 5

Нелинейные алгебраические уравнения

Огромное количество задач вычислительной математики связано с решением нелинейных алгебраических уравнений, а также систем таких уравнений. При этом необходимость решения нелинейных уравнений возникает зачастую на промежуточных шагах, при реализации фрагментов более сложных алгоритмов (к примеру, при расчетах дифференциальных уравнений при помощи разностных схем и т. п.).

Постановка задач выглядит следующим образом. Пусть имеется либо одно алгебраическое уравнение с неизвестным x:

$$f(x)=0,$$

(где $f(x)$ — некоторая функция), либо система из N алгебраических уравнений:

$$\begin{cases} f_1(x_1, \ldots, x_M) = 0, \\ \ldots \\ f_N(x_1, \ldots, x_M) = 0. \end{cases}$$

Требуется найти корни, т. е. все значения x (или, в случае системы все N-покомпонентные векторы x), которые переводят уравнение (или, соответственно, систему уравнений) в верное равенство (равенства).

Примечание

Решение систем линейных уравнений, у которых все функции имеют вид $f_i(x)=a_{i1}\cdot x_1+a_{i2}\cdot x_2+\ldots+a_{iN}\cdot x_N$, представляет собой отдельную задачу вычислительной линейной алгебры. Она рассматривается в *главе 8.*

5.1. Аналитическое решение уравнений

Относительно небольшое количество задач отыскания корней алгебраических уравнений можно решить аналитически, а на практике почти всегда приходится искать решение при помощи численных методов. Тем не менее мы начнем знакомство с принципами решения алгебраических уравнений в Mathcad именно с описания использования символьного процессора, что позволит, с одной стороны, осво-

ить соответствующий инструментарий Mathcad, а с другой — лучше понять специфику исследуемых задач.

5.1.1. Вычислительный блок *Given / Find*

Рассмотрим решение системы N нелинейных уравнений с M неизвестными:

$$\begin{cases} f_1(x_1, \ldots, x_M) = b_1; \\ \ldots \\ f_N(x_1, \ldots, x_M) = b_N. \end{cases} \tag{5.1}$$

Здесь $f_1(x_1,\ldots,x_M)=b_1, \ldots, f_N(x_1,\ldots,x_M)=b_N$ — некоторые скалярные выражения, зависящие от скалярных переменных $x_1, x_2, \ldots, x_M$ и, возможно, от еще каких-либо переменных. Уравнений может быть как больше, так и меньше числа переменных. Заметим, что систему (5.1) можно формально переписать в виде

$$f(x)=b, \tag{5.2}$$

где x — вектор, составленный из переменных $x_1, x_2, \ldots, x_M$, b — вектор, составленный из правых частей уравнений, а $f(x)$ — соответствующая векторная функция их левых частей.

Для решения систем в Mathcad применяется специальный *вычислительный блок* `Given/Find` (Дано/Найти), или, по-другому, блок решения, состоящий из трех частей, идущих последовательно друг за другом:

- `Given` — ключевое слово;
- система, записанная логическими операторами в виде равенств и, возможно, неравенств;
- `Find`$(x_1, \ldots, x_M)$ — встроенная функция для решения системы уравнений относительно переменных $x_1, \ldots, x_M$.

Вставлять логические операторы следует, пользуясь панелью инструментов **Булева алгебра** (Boolean). Если вы предпочитаете ввод с клавиатуры, помните, что логический знак равенства вводится сочетанием клавиш <Ctrl>+<=>. Значение функции `Find` представляет собой матрицу, составленную из всевозможных решений по каждой переменной, причем количество ее строк в точности равно числу аргументов `Find`. Структура матрицы решения станет сразу вам понятной, как только вы бросите взгляд на примеры, приведенные ниже в данном разделе.

Примечание

При решении уравнений в векторной форме в вычислительном блоке не рекомендуется, а, начиная с версии Mathcad 12, просто запрещено использование одних элементов вектора в качестве неизвестных, а других — в качестве параметров задачи. Соответствующие примеры, иллюстрирующие данное ограничение, вы найдете на компакт-диске.

5.1.2. Одно уравнение

Рассмотрим теперь на практике, каким образом можно решить одно нелинейное уравнение.

Использование блока *Given / Find*

Поясним применение вычислительного блока на примере решения одного (кубического) уравнения с одним неизвестным `x` (рис. 5.1):

$$3x^3+2x^2-7x=0. \tag{5.3}$$

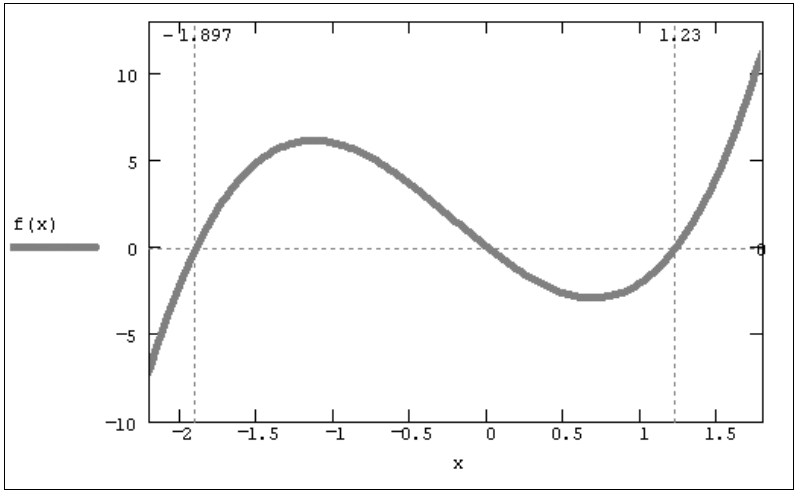


Рис. 5.1. График функции $f(x)=3x^3+2x^2-7x$

Листинг 5.1. Аналитическое решение кубических уравнений

```
Given
```

$$3x^3 + 2x^2 - 7x = 0$$

$$\text{Find}(x) \rightarrow \begin{pmatrix} 0 & \frac{-1}{3} + \frac{1}{3}\cdot\sqrt{22} & \frac{-1}{3} - \frac{1}{3}\cdot\sqrt{22} \end{pmatrix} = (0 \quad 1.23 \quad -1.89)^T$$

В листинге 5.1 вы видите все три последовательные строки вычислительного блока. Первая строка представляет собой обязательное ключевое слово `Given`, следующая строка является, собственно, записью уравнения (5.3), а в последней строке листинга включается в работу встроенная функция `Find`. Обратите внимание, что после имени функции `Find` находится оператор символьного вывода, справа от которого (по истечении необходимого времени работы символьного процессора) возникает аналитический результат решения уравнения. Существенно, что он является точным решением (записанным в данном случае в трансцендентном виде), а получить числовые значения корней можно, поставив после него символ численного равенства (как это сделано в последней строке листинга 5.1).

Как видно из листинга 5.1, уравнение имеет три различных корня, которые представляются справа от функции соответствующим трехкомпонентным вектором. Таким образом, решение предлагается пользователю в форме матрицы размера `1×3` (одна неизвестная переменная имеет три значения, каждое из которых обращает уравнение в тождество).

Пример, использованный в листинге 5.1, включает уравнение, записанное в традиционной форме равенства. Приведем решение того же самого уравнения, если оно представлено в несколько другой форме, подчеркивающей специфику задачи нахождения корней функции (листинг 5.2). Основное отличие листинга 5.2 от предыдущего связано с другой формой записи исследуемого уравнения через функцию пользователя `f(x)`. Иными словами, подчеркивается специфика поставленной задачи отыскания нулевых значений некоторой функции.

Важно заметить, что точное решение уравнения (непосредственно после оператора символьного вывода результата работы функции `Find`) обращает `f(x)` в тождественный ноль, а пересчитанные числовые значения корней (после знака обычного равенства) обеспечивают лишь ее приближенное равенство нулю (разумеется, это связано с ошибками округления).

Внимание!

Не забывайте о том, что вводить знаки равенства в уравнение в пределах вычислительного блока `Given/Find` следует при помощи панели **Булева алгебра** (Boolean).

Листинг 5.2. Аналитический поиск нулей функции `f(x)`

$$f(x) := 3x^3 + 2x^2 - 7x$$

Given

$$f(x) = 0$$

$$\text{Find}(x) \rightarrow \begin{pmatrix} 0 & \frac{-1}{3} + \frac{1}{3}\cdot\sqrt{22} & \frac{-1}{3} - \frac{1}{3}\cdot\sqrt{22} \end{pmatrix} = (0 \quad 1.23 \quad -1.897)$$

$$f(0) = 0$$

$$f\left(\frac{-1}{3} + \frac{1}{3}\cdot\sqrt{22}\right) = 0 \qquad f\left(\frac{-1}{3} - \frac{1}{3}\cdot\sqrt{22}\right) = 5.329 \times 10^{-15}$$

$$f(1.23) = -1.599 \times 10^{-3} \qquad f(-1.897) = -3.466 \times 10^{-3}$$

В заключение разговора о символьном решении уравнений с одним неизвестным приведем еще два показательных примера, связанных с нахождением нулей функции нескольких аргументов (т. е. зависящих, помимо, собственно, неизвестного, еще и от дополнительных параметров). Листинг 5.3 демонстрирует, как выглядит решение уравнения, включающего четыре различные переменные, по некоторым из этих переменных. Обратите внимание на последний из трех приведенных в листинге 5.3 примеров, иллюстрирующий результат решения уравнения относительно сразу всех входящих в него параметров.

Помните, что для корректной работы символьного процессора вовсе не обязательно задавать конкретные значения переменных, входящих в уравнение, а если такие значения (для некоторых либо всех параметров) определены, то это учитывается при выводе результата, что иллюстрируется листингом 5.4.

Листинг 5.3. Символьное решение уравнения относительно разных переменных

Given $a x^3 + b x^2 - c x = 0$

$$\text{Find}(x) \rightarrow \begin{bmatrix} 0 & \frac{1}{2\cdot a}\cdot\left[-b + \left(b^2 + 4\cdot a\cdot c\right)^{\frac{1}{2}}\right] & \frac{1}{2\cdot a}\cdot\left[-b - \left(b^2 + 4\cdot a\cdot c\right)^{\frac{1}{2}}\right] \end{bmatrix}$$

Given $a x^3 + b x^2 - c x = 0$

$$\text{Find}(a) \rightarrow \frac{-1}{x^2}\cdot(b\cdot x - c)$$

Given $a x^3 + b x^2 - c x = 0$

$$\text{Find}(a, b, c, x) \rightarrow \begin{pmatrix} a & a \\ b & b \\ c & a\cdot x^2 + b\cdot x \\ 0 & x \end{pmatrix}$$

Листинг 5.4. Символьное решение уравнения, зависящего от параметров, в случае предварительного задания их числовых значений

$a := 3$ $b := 2$

Given

$a x^3 + b x^2 - c x = 0$

$$\text{Find}(x) \rightarrow \begin{bmatrix} 0 & \frac{-1}{3} + \frac{1}{3}\cdot(1 + 3\cdot c)^{\frac{1}{2}} & \frac{-1}{3} - \frac{1}{3}\cdot(1 + 3\cdot c)^{\frac{1}{2}} \end{bmatrix}$$

Если решить уравнение аналитически не удается, то результатом применения оператора символьного вывода после функции `Find` будет либо тривиальное выражение типа `Find(x)→x`, либо сообщение об ошибке. При этом следует помнить, что символьный процессор Mathcad "умеет" находить не только действительные, но и комплексные корни уравнений. Например, при решении уравнения $x(x^2+1)=0$, имеющего три очевидных корня — одного действительного (равного нулю) и двух чисто мнимых (±i, где i — мнимая единица) в качестве решения будет выдан вектор из трех элементов `(0,i,- i)`.

Ключевое слово *solve*

Для аналитического решения нелинейного алгебраического уравнения предусмотрена и иная запись, связанная с использованием ключевого слова `solve` (листинг 5.5). Имя переменной, по которой необходимо решить уравнение, указывается после запятой, которую следует ввести сразу за ключевым словом `solve`.

Листинг 5.5. Решение уравнения при помощи ключевого слова `solve`

$$(y\cdot x-1)\cdot x=1 \text{ solve}, x \rightarrow \begin{pmatrix} \dfrac{\sqrt{4\cdot y+1}+1}{2\cdot y} \\ -\dfrac{\sqrt{4\cdot y+1}-1}{2\cdot y} \end{pmatrix}$$

Отметим также, что в Mathcad 14 появилась новая функциональность аналитического решения уравнений. А именно новое ключевое слово `fully`, если его ввести через запятую после ключевого слова `solve` и имени переменной, даст возможность получить все решения уравнения (листинг 5.6). Сравните результат решения, представленный в листингах 5.5 и 5.6 (соответственно, без и с применением ключевого слова `fully`).

(14) Листинг 5.6. Аналитический поиск всех корней уравнения

$$(y\cdot x-1)\cdot x=1 \text{ solve}, x, \text{fully} \rightarrow \left| \begin{array}{ll} \begin{pmatrix} -\dfrac{\sqrt{4\cdot y+1}-1}{2\cdot y} \\ \dfrac{\sqrt{4\cdot y+1}+1}{2\cdot y} \end{pmatrix} & \text{if } y\neq 0 \\ -1 \quad \text{if } y=0 & \end{array} \right.$$

5.1.3. Системы уравнений

Символьное решение системы алгебраических уравнений отличается от описанного случая одного уравнения только количеством соотношений, задаваемых после ключевого слова `Given`. Соответственно, число неизвестных также может быть любым, причем необязательно равным числу уравнений. Если система уравнений имеет не обособленные решения, а целые семейства решений, то соответствующие результаты выдаются символьным процессором Mathcad в виде выражений, формально зависящих от одной из переменных как от параметра (пример решения одного уравнения с тремя неизвестными был приведен ранее в последних двух строках листинга 5.3).

Решение системы двух нелинейных уравнений иллюстрирует листинг 5.7. Нахождение символьным процессором его обоих корней визуализируется на графике,

приведенном на рис. 5.2. На нем каждое из уравнений показывается в виде зависимости `y(x)`: первое — сплошной кривой, а второе — пунктиром. Поскольку первое уравнение является квадратичным, то оно определяет на плоскости `XY` параболу, и поскольку второе уравнение линейное, то оно соответствует на графике прямой линии. Очевидно, что две точки пересечения кривых соответствуют одновременному выполнению обоих уравнений, т. е. их координаты равны искомым действительным корням системы.

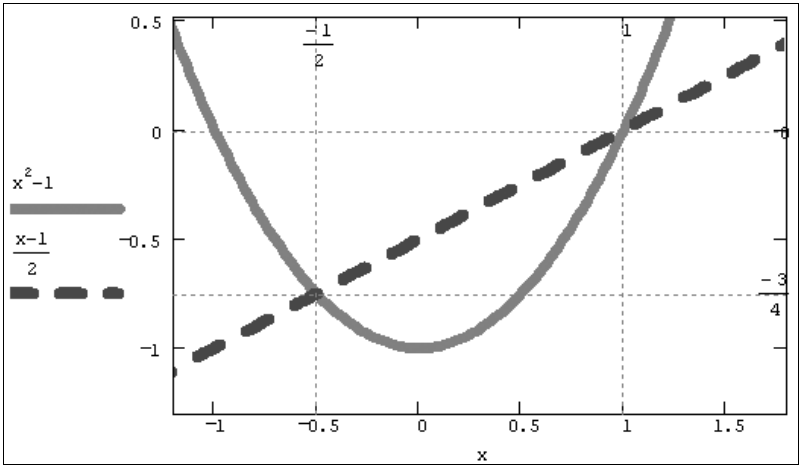


Рис. 5.2. Графическая интерпретация решения системы двух уравнений (см. листинг 5.7)

Листинг 5.7. Символьное решение системы двух уравнений

```
Given
−x^2 + y + 1 = 0
x = 2y + 1
Find(x, y) → ( 1   −1/2 )
             ( 0   −3/4 )
```

5.1.4. Решение уравнений при помощи меню

Аналитическое решение алгебраического уравнения можно получить и при помощи меню. Данный способ удобно применять, когда уравнение записано в форме математического выражения (зависящего, возможно, от различных переменных) и требуется вычислить аналитически значение одной переменной, при котором выражение обращается в ноль.

Для этого:

1. Введите выражение.
2. Выделите переменную, относительно которой будет решаться уравнение, приравнивающее выражение нулю.
3. Выберите в меню **Символьные операции** (Symbolics) пункт **Переменная / Решить** (Variable / Solve) (рис. 5.3).

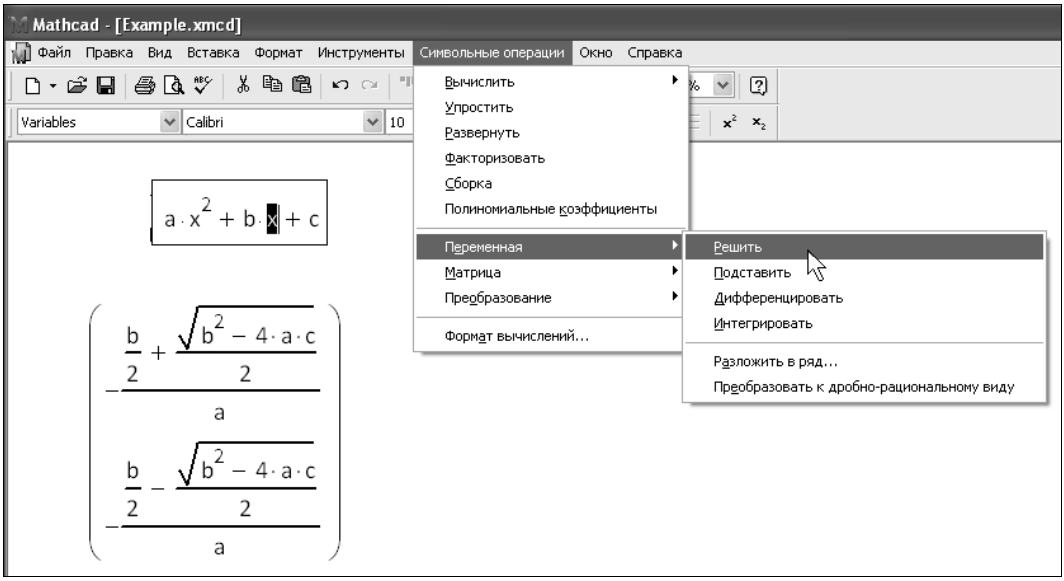


Рис. 5.3. Символьное решение уравнения

В итоге после выражения появится строка с результатом решения уравнения (т. е. значениями переменной, которые обращают исходное выражение в ноль). Следует иметь в виду, что решение уравнения производится именно по той переменной, которую пользователь выделяет перед вводом команды меню. Разумеется, если выделять различные переменные, то в итоге будут выдаваться совершенно разные решения.

> ***Внимание!***
>
> Помните о том, что символьные вычисления, проводимые при помощи меню, не являются "живыми", т. е. не будут меняться впоследствии, если вы будете редактировать исходное выражение.

5.2. Численное решение уравнений

Как правило, отыскание корней алгебраического уравнения (или системы уравнений) численными методами связано с двумя задачами:

- локализация корней, т. е. определение их существования в принципе, а также исследование их количества и примерного расположения;

- собственно отыскание корней с заданной погрешностью (в Mathcad ее роль играет системная константа TOL).

Последнее означает, что надо найти значения x_0, при которых $f(x_0)$ отличается от нуля не более чем на TOL. Почти все встроенные функции системы Mathcad, предназначенные для решения нелинейных алгебраических уравнений (в том числе функция Find применительно к численному процессору), нацелены на решение второй задачи, т. е. предполагают, что корни уже приблизительно локализованы. Иными словами, перед тем, как непосредственно приступать к нахождению корней уравнения, необходимо (хотя бы приблизительно) представлять себе, где они находятся. Имея в виду данную оговорку, мы перейдем к рассмотрению второй из задач, а проблему предварительной локализации корней рассмотрим в конце данного раздела.

5.2.1. Системы уравнений: функция *Find*

Рассмотрим решение системы N нелинейных уравнений с M неизвестными:

$$\begin{cases} f_1(x_1, \ldots, x_M) = 0; \\ \ldots \\ f_N(x_1, \ldots, x_M) = 0. \end{cases} \tag{5.4}$$

Здесь $f_1(x_1, \ldots, x_M), \ldots, f_N(x_1, \ldots, x_M)$ — некоторые скалярные функции от скалярных переменных $x_1, x_2, \ldots, x_M$ и, возможно, от еще каких-либо переменных. Уравнений может быть как больше, так и меньше числа переменных.

Заметим, что систему (5.4) можно формально переписать в виде

$$f(x)=0, \tag{5.5}$$

где x — вектор, составленный из переменных $x_1, x_2, \ldots, x_M$, а f(x) — соответствующая векторная функция.

Вычислительный блок *Given / Find*

Для численного решения систем уравнений применяется тот же самый *вычислительный блок*, что и для символьных вычислений (*см. разд. 5.1.1*). Повторимся, что он состоит из ключевого слова Given, самой системы уравнений, записанной при помощи логических операторов панели **Булева алгебра** (Boolean), а также встроенной функции Find. Find$(x_1, \ldots, x_M)$ — встроенная функция для решения системы алгебраических уравнений и неравенств относительно переменных $x_1, \ldots, x_M$. Значение функции Find представляет собой вектор, составленный из решений по каждой переменной.

Примечание 1

Встроенная функция Find использует в качестве численного алгоритма один из градиентных методов (*см. разд. 5.3*). Этот факт налагает некоторые ограничения на уравнения системы, которые должны быть достаточно гладкими функциями своих аргументов.

Примечание 2

При численных расчетах может быть найден только один из корней уравнения, в отличие от символьных вычислений, которые позволяют определить все имеющиеся корни.

Применение численного нахождения корней отличается от символьного двумя обстоятельствами:

- вместо оператора символьного вывода после функции `Find` следует использовать оператор численного вывода (знак равенства);
- перед вычислительным блоком `Given/Find` должны быть заданы *начальные значения* (guess value) для всех неизвестных, т. е. всем переменным $x_1, \ldots, x_M$, относительно которых решается уравнение, следует предварительно присвоить некоторые численные значения, с которых и будет начинаться поиск корня. Таким образом, присвоение начального значения требует априорной информации о примерном местонахождении корня и связано с проблемой локализации корней, упомянутой в начале *разд. 5.2*.

Примечание

Помните также о том, что, в отличие от символьного, для численного процессора необходимо задать числовые значения всех параметров, входящих в уравнение.

Одно уравнение с одним неизвестным

Рассмотрим в качестве примера (листинг 5.8) одно уравнение с одним неизвестным, которое уже решалось нами аналитически (см. листинги 5.1 и 5.2). Уравнение имеет три корня, как видно из графика, приведенного на рис. 5.4. Обратите внимание, что перед ключевым словом `Given` переменной `x` присваивается некоторое значение `x=1`. В остальном применение функции `Find` для решения уравнения не отличается от символьных расчетов.

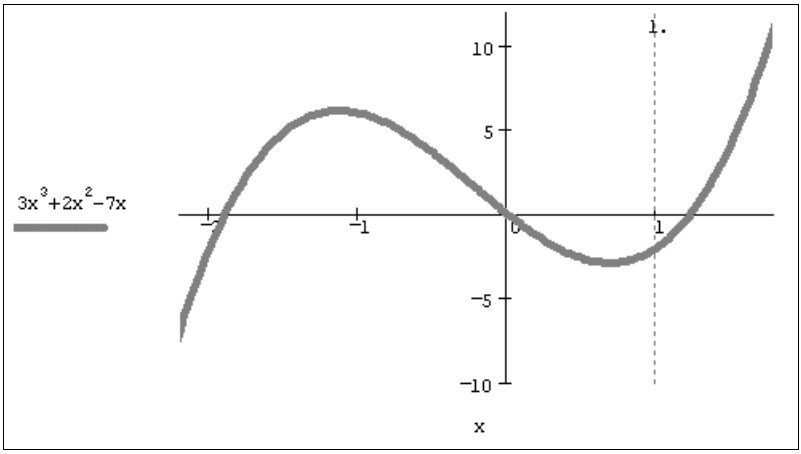


Рис. 5.4. Графическая иллюстрация решения кубического уравнения

Листинг 5.8. Численное решение кубического уравнения с начальным значением x=1

```
x := 1
Given
3x^3 + 2x^2 - 7x = 0
Find(x) = 1.23
```

Как уже отмечалось выше, результатом численного решения алгебраического уравнения является один его корень. Для того чтобы отыскать остальные корни, необходимо повторно решить уравнение, взяв для переменной `x` другие начальные значения. Например, если присвоить ей в начале листинга значение `x=-1`, то численным процессором будет выдан в качестве результата другой корень `x=0` (листинг 5.9). Такая работа программы Mathcad связана с особенностями применяемых численных алгоритмов (*см. разд. 5.3*). Для численного определения всех корней уравнения следует применять специальные приемы, например, сканирование по неизвестным (*см. разд. 5.2.4*).

Листинг 5.9. Численное решение кубического уравнения с начальным значением x=-1

```
x := -1
Given
3x^3 + 2x^2 - 7x = 0
Find(x) = 0
```

Системы уравнений

Приведем еще один пример численного решения алгебраических уравнений, обратившись на этот раз к системе двух уравнений, которая также уже исследовалась нами при помощи символьного процессора. Система имеет два решения, показанные графически на рис. 5.2 и найденные аналитически в листинге 5.7 (*см. разд. 5.1.3*). Листинг 5.10, демонстрирующий численное решение рассматриваемой системы, начинается с присвоения неизвестным начальных значений `x=10,y=10`. После этого следует ключевое слово `Given` и два логических оператора, выражающих рассматриваемую систему уравнений. В результате (последняя строка листинга) Mathcad находит один из корней `x=1,y=0`, причем первый элемент вектора решения есть первый аргумент функции `Find`, а второй элемент — ее второй аргумент. Поскольку решение производится численным методом, оно выдается с некоторой погрешностью, не превышающей встроенной константы `CTOL`. Если задать в первой строке листинга другие начальные значения, расположенные ближе к другому корню, например, `x=0,y=0`, то найден в итоге будет другой корень `x=-0.5,y=-0.75`.

Примечание 1

На самом деле в вычислительном блоке используются обе системные константы Mathcad, связанные с заданием погрешности: `TOL` и `CTOL`. Константа `CTOL` ограничивает невязку, т. е. задает точность выполнения уравнений, введенных после ключевого слова `Given`. Например, если `CTOL=0.001`, то уравнение `x=10` будет считаться выполненным и при `x=10.001`, и при `x=9.999`. Другая константа `TOL` определяет условие прекращения итераций численным алгоритмом (*см. разд. 5.4*). Значение `CTOL` может быть задано пользователем так же, как и `TOL`, например, `CTOL:=0.01`. По умолчанию принято, что `CTOL=TOL=0.001`, но вы по желанию можете переопределить их.

Примечание 2

Часто бывает очень полезным проверить точность решения уравнений "вручную", подставив найденные вычислительным процессором корни в исходные уравнения и оценив значение их невязок.

Листинг 5.10. Численное решение системы алгебраических уравнений

$x := 10 \qquad y := 10$

Given

$-x^2 + y + 1 = 0$

$x = 2y + 1$

$$\text{Find}(x, y) = \begin{pmatrix} 1 \\ -1.133 \times 10^{-5} \end{pmatrix}$$

Если предпринять попытку решить несовместную систему, в частности в рассматриваемом примере добавить еще одно уравнение, то Mathcad выдаст сообщение об ошибке, гласящее, что ни одного решения не найдено, и предложение попробовать поменять начальные значения или значение погрешности.

Особую осторожность следует соблюдать при решении систем с числом неизвестных бóльшим, чем число уравнений. Например, можно удалить одно из двух уравнений из рассмотренной нами задачи, попытавшись решить оставшееся единственное уравнение с двумя неизвестными `x` и `y` (листинг 5.11). В такой постановке задача имеет бесконечное множество корней: для любого `x` и, соответственно, $y=x^2-1$ условие, определяющее единственное уравнение, выполнено. Однако даже если корней бесконечно много, численный метод будет производить расчеты только до тех пор, пока логические выражения в вычислительном блоке не будут выполнены (конечно, в пределах погрешности). После этого итерации будут остановлены и выдано решение. В результате будет найдена всего одна пара значений `(x,y)`, обнаруженная первой, как это показано в последней строке листинга 5.11.

Примечание

Для того чтобы найти все решения рассматриваемой задачи, можно обратиться к возможностям символьного процессора Mathcad. Достаточно в последней строке листинга заменить знак равенства на оператор символьного вывода, и в качестве ответа будет выдано семейство решений `x` и x^2-1.

Листинг 5.11. Численное решение уравнения, имеющего бесконечное множество корней, приводит к одному из них

$x := 10 \qquad y := 10$

Given

$-x^2 + y + 1 = 0$

$$\text{Find}(x, y) = \begin{pmatrix} 3.377 \\ 10.402 \end{pmatrix}$$

Системы уравнений и неравенств

Пока мы рассматривали примеры систем уравнений, число которых было таким же, как и число неизвестных, что встречается наиболее часто. Но число уравнений и неизвестных может и не совпадать. Более того, в вычислительный блок можно добавить дополнительные условия в виде неравенств. Например, введение ограничения на поиск только отрицательных значений `x` в рассмотренный выше листинг 5.10 приведет к нахождению другого решения, как это показано в листинге 5.12.

Обратите внимание, что, несмотря на те же начальные значения, что и в листинге 5.10, в листинге 5.12 мы получили другой корень системы уравнений. Это произошло именно благодаря введению дополнительного неравенства, которое определено в блоке `Given` в предпоследней строке листинга 5.12.

Листинг 5.12. Численное решение системы алгебраических уравнений и неравенств

$x := 10 \qquad y := 10$

Given

$-x^2 + y + 1 = 0$

$x = 2y + 1$

$x < 0$

$$\text{Find}(x, y) = \begin{pmatrix} -0.5 \\ -0.75 \end{pmatrix}$$

5.2.2. Уравнение с одним неизвестным: функция *root*

Для решения уравнения с одним неизвестным в Mathcad, помимо вычислительного блока `Given/Find`, предусмотрена встроенная функция `root`, которая, в зависимости от типа задачи, может включать либо два, либо четыре аргумента и, соответственно, использовать разные алгоритмы поиска корней:

- `root(f(x),x)`;
- `root(f(x),x,a,b)`:
  - `f(x)` — скалярная функция, определяющая уравнение `f(x)=0`;
  - `x` — имя скалярной переменной, относительно которой решается уравнение;
  - `a,b` — границы интервала, внутри которого происходит поиск корня.

Первый тип функции `root`, аналогично встроенной функции `Find`, требует дополнительного задания начального значения переменной `x`, для чего нужно просто перед применением функции `root` присвоить `x` некоторое число. Таким образом, присвоение начального значения требует априорной информации о примерной локализации корня, т. к. поиск корня будет производиться вблизи этого числа. Пример работы функции `root` объясняется листингом 5.13.

Листинг 5.13. Два варианта решения уравнения методом секущих

$$\mathrm{root}\left(e^{x}-1, x, -1, 1\right) = 0$$

$$x := 1$$

$$\mathrm{root}\left(e^{x}-1, x\right) = 2.21 \times 10^{-5}$$

Как вы можете убедиться (первая строка листинга 5.13), для решения уравнения при помощи функции `root(f(x),x,a,b)` не требуется задавать начального приближения, а достаточно указать интервал `[a,b]`. Поиск корня будет осуществлен в промежутке между `a` и `b` альтернативным численным методом (Риддера или Брента). Когда функция `root` имеет четыре аргумента, следует помнить о двух ее особенностях. Во-первых, внутри интервала не должно находиться более одного корня, иначе будет найден один из них, заранее неизвестно, какой именно. Во-вторых, значения `f(a)` и `f(b)` должны иметь разный знак, иначе будет выдано сообщение об ошибке.

В чем же отличие встроенной функции `Find` от функции `root`? Оно состоит в том, что для решения одних и тех же задач используются различные численные алгоритмы (градиентные и метод секущих соответственно). В примерах уравнений с одним неизвестным, которые мы рассматривали до сего момента, выбор метода не влиял на окончательный результат, поскольку фигурировавшие в них функции были "хорошими", т. е. достаточно гладкими для поиска корня одним из градиентных методов, требующих, как известно, вычисления производных. Между тем бывают ситуации, когда применение того или иного метода имеет решающее значение.

Приведем пример простой функции `f(x)`, корни которой удается отыскать только при помощи функции `root` (листинг 5.14). Она определена в первой строке этого листинга, а ее корень вычислен во второй строке. Из графика, представленного на рис. 5.5, видно, что `f(x)` имеет особенность в окрестности своего корня, являясь в ней разрывной. В завершающей части листинга 5.14 предпринимается попытка отыскать нулевое значение `f(x)` посредством вычислительного блока `Given/Find`, которая оказывается неудачной.

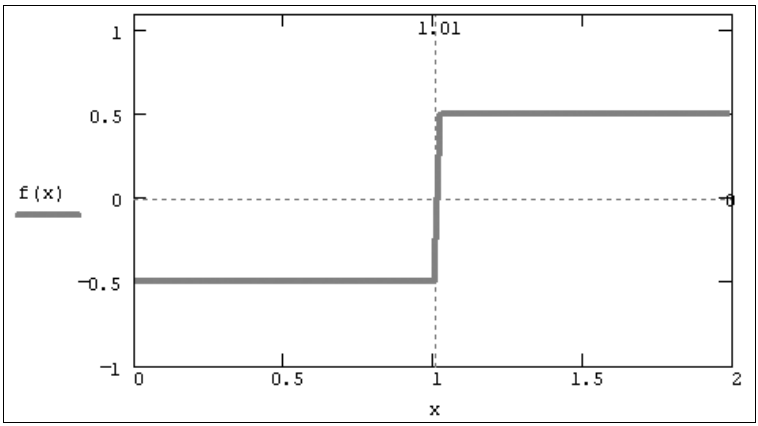


Рис. 5.5. Модельная функция `f(x)` (продолжение листинга 5.14)

Листинг 5.14. Пример уравнения, которое удается решить только методом секущих

```
f(x) := | Φ(x − 1.01) − 0.5   if x ≠ 1.01
        | 0  otherwise
root(f(x), x, 0, 2) = 1.01
x := 1
Given
f(x) = 0
Find(x) = ■
```

Остается добавить, что `f(x)` может быть функцией не только `x`, а любого количества аргументов. Именно поэтому в самой функции `root` необходимо определить, относительно какого из аргументов следует решить уравнение. Эта возможность проиллюстрирована листингом 5.15 на примере функции двух переменных $f(x,y)=x^2-y^2+1$. В нем сначала решается уравнение `f(x,0)=0` относительно переменной `x`, а потом — другое уравнение `f(0,y)=0` относительно переменной `y`, причем, благодаря удачному подбору начальных значений, вычисляются все корни данного квадратичного уравнения.

Таким образом, в обоих случаях один из аргументов функции `f(x)` воспринимается как неизвестное, а другой — как параметр. Не забывайте при численном решении уравнений относительно одной из переменных предварительно определить значения остальных переменных. Иначе попытка вычислить уравнения приведет к появлению ошибки **Эта переменная не определена** (This variable or function is not defined above).

Примечание

Для того чтобы отыскать зависимость корней уравнения, вычисленных по одной переменной, от других переменных, разработаны специальные эффективные алгоритмы. Об одной из возможностей читайте в *разд. 5.3.3.*

Листинг 5.15. Поиск корней уравнения, зависящего от двух переменных

$f(x, y) := x^2 - y^2 + 1$

$x := 1 \qquad root(f(x, 0), x) = -i$

$x := -1 \qquad root(f(x, 0), x) = i$

$y := 0 \qquad root(f(0, y), y) = 1$

$y := -100 \qquad root(f(0, y), y) = -1$

5.2.3. Корни полинома: функция *polyroots*

Если функция `f(x)` является *полиномом*, то все его корни можно определить, используя встроенную функцию:

❒ `polyroots(v)`

- `v` — вектор, составленный из коэффициентов полинома.

Поскольку полином `N`-й степени имеет ровно `N` корней (некоторые из них могут быть кратными), вектор `v` должен состоять из `N+1` элемента. В основе встроенной функции `polyroots` лежат специфические численные алгоритмы, а результатом ее действия является вектор, составленный из `N` корней рассматриваемого полинома. При этом нет надобности вводить какое-либо начальное приближение, как для функции `root`. Пример поиска корней полинома четвертой степени иллюстрируется листингом 5.16.

Коэффициенты рассматриваемого в примере полинома

$$f(x)=(x-3)\cdot(x-1)^3=x^4-6x^3+12x^2-10x+3$$

записаны в виде вектора в первой строке листинга. Первым в векторе должен идти свободный член полинома, вторым — коэффициент при x^1 и т. д. Соответственно, последним `N+1` элементом вектора должен быть коэффициент при старшей степени x^N.

> ***Совет***
>
> Иногда исходный полином имеется не в развернутом виде, а, например, как произведение нескольких полиномов. В этом случае определить все его коэффициенты можно, выделив его и выбрав в меню **Символьные операции** (Symbolics) пункт **Развернуть** (Expand). В результате символьный процессор Mathcad сам преобразует полином в нужную форму; пользователю надо будет только корректно ввести ее в аргументы функции `polyroots`.

Листинг 5.16. Вычисление корней полинома

$v := (3 \quad -10 \quad 12 \quad -6 \quad 1)^T$

$$polyroots(v) = \begin{pmatrix} 0.992 \\ 1.004 + 7.177i \times 10^{-3} \\ 1.004 - 7.177i \times 10^{-3} \\ 3 \end{pmatrix}$$

Обратим внимание на результат применения функции `polyroots`, заметив, что численный метод вместо двух из трех действительных единичных корней (иными словами, кратного корня `1`) выдает два мнимых числа. Однако малая мнимая часть этих корней находится в пределах погрешности, определяемой константой `TOL`, и не должна вводить пользователей в заблуждение. Просто нужно помнить, что корни полинома могут быть комплексными, и ошибка вычислений может сказываться как на действительной, так и на комплексной части искомого корня.

Для функции `polyroots` можно выбрать один из двух численных методов — метод полиномов Лаггера (он установлен по умолчанию) или метод парной матрицы.

Для смены метода:

1. Вызовите контекстное меню, щелкнув правой кнопкой мыши на слове `polyroots`.
2. В верхней части контекстного меню выберите либо пункт **Метод Лаггера** (LaGuerre), либо **Парная матрица** (Companion Matrix).
3. Щелкните правой кнопкой мыши вне действия функции `polyroots` — если включен режим автоматических вычислений, будет произведен пересчет корней полинома в соответствии с вновь выбранным методом.

Для того чтобы оставить за Mathcad выбор метода решения, установите флажок **Автовыбор** (AutoSelect), выбрав одноименный пункт в том же самом контекстном меню.

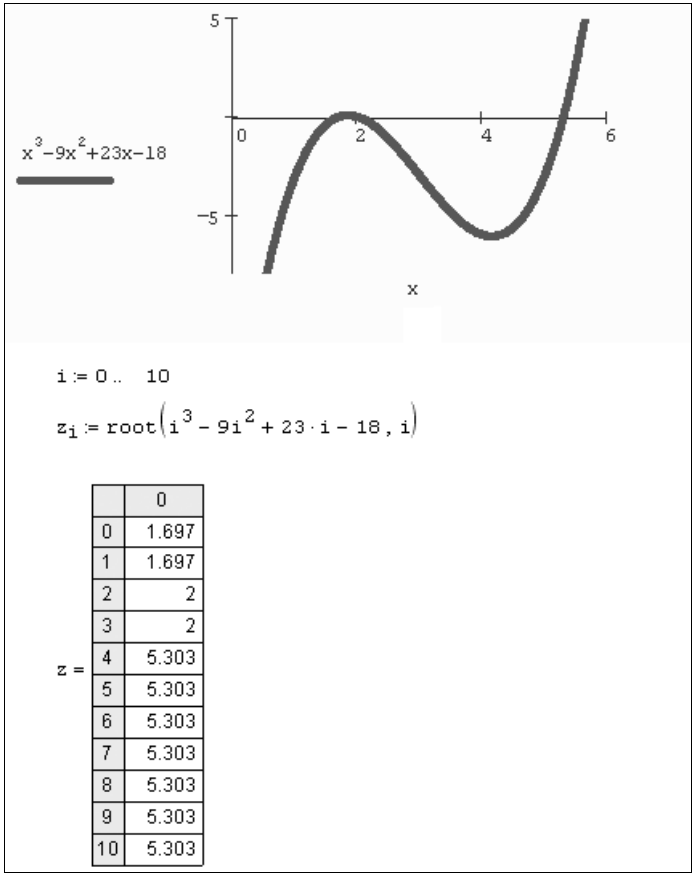
5.2.4. Локализация корней

Чтобы решить задачу предварительной (грубой) локализации корней, в самых простых случаях можно использовать графическое представление `f(x)` (см. рис. 5.1, 5.2 и 5.4). Понятно, что в случае многомерных систем такой способ практически неприменим. Если требуется исследовать определенную область определения переменных уравнения на наличие корней, определив их примерное положение, то обычно применяют весьма расточительный способ, называемый *сканированием*. Оно состоит в последовательном поиске корня, начиная из множества пробных точек, покрывающих расчетную область.

Обычно (вне Mathcad) сканирование организуют следующим образом. Область определения функции разбивается на элементарные области (в случае функции двух переменных чаще всего прямоугольные, в случае трех переменных — кубические и т. д.). Из центра каждой элементарной области запускается численный метод поиска корня, и в случае выхода итераций за ее пределы расчеты прерываются, а в противном случае происходит нахождение корня. Гораздо менее надежной (но зато более экономной) альтернативой является простое вычисление и сравнение между собой невязок системы уравнений в центральных точках элементарных областей. На тех участках области определения, где норма невязки невелика, вероятность локализации корня больше, и именно из локальных минимумов нормы невязки можно запускать градиентный метод для уточнения корня.

Пример организации упрощенного варианта сканирования по одной переменной приведен на рис. 5.6. График функции, корни которой подлежат определению, по-

казан в его верхней части. Затем осуществляется решение уравнения при помощи функции `root` для нескольких последовательно расположенных узлов. Результат выдается в последней строке листинга в виде таблицы, из которой видно, что на рассматриваемом интервале уравнение имеет три корня.



	0
0	1.697
1	1.697
2	2
3	2
4	5.303
5	5.303
6	5.303
7	5.303
8	5.303
9	5.303
10	5.303

Рис. 5.6. Сканирование по переменной `x`

Примечание

Конечно, гарантии, что все существующие корни будут найдены, особенно в многомерных случаях, чаще всего нет. Всегда существует вероятность "просмотреть" корень, расположенный между узлами сканирования.

ⓘ 5.3. О численных методах

Уделим теперь небольшое внимание численным алгоритмам, которые используются в работе встроенных функций `Find` и `root`, чтобы читатель имел возможность применять их для решения алгебраических уравнений более осмысленно.

ⓘ 5.3.1. Метод секущих: функция *root*

Итерационный алгоритм, реализованный в функции `root`, который называется *методом секущих*, состоит в следующем (рис. 5.7):

1. Начальное приближение принимается за `0`-е приближение к корню: `x0=x`.
2. Выбирается шаг `h=TOL·x` и определяется первое приближение к корню `x1=x0+h`. Если `x=0`, то принимается `h=TOL`.
3. Через эти две точки проводится секущая — прямая линия, которая пересекает ось `X` в некоторой точке `x2`. Эта точка принимается за второе приближение.
4. Новая секущая проводится через первую и вторую точки, тем самым определяя третье приближение, и т. д.
5. Если на каком-либо шаге оказывается, что уравнение выполнено, т. е. `|f(x)|<TOL`, то итерационный процесс прерывается, и `x` выдается в качестве решения.

Результат, показанный на рис. 5.7, получен для погрешности вычислений, которой в целях иллюстративности предварительно присвоено значение `TOL=0.5`. Поэтому для поиска корня с такой невысокой точностью оказалось достаточно одной итерации. В вычислениях, приведенных в листингах 5.13—5.15 (*см. разд. 5.2.2*), погрешность `TOL=0.001` была установлена по умолчанию, и решение, выданное численным методом, лежало намного ближе к истинному положению корня. Иными словами, чем меньше константа `TOL`, тем ближе к нулю будет значение `f(x)` в найденном корне, но тем больше времени будет затрачено вычислительным процессором Mathcad на его поиск.

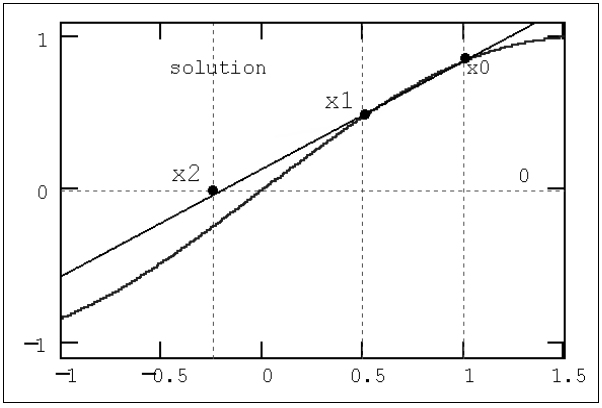


Рис. 5.7. Иллюстрация метода секущих

Если уравнение неразрешимо, то при попытке найти его корень будет выдано сообщение об ошибке. Кроме того, к ошибке или выдаче неправильного корня может привести и попытка применить метод секущих в области локального максимума или минимума `f(x)`. В этом случае секущая может иметь направление, близкое к горизонтальному, выводя точку следующего приближения далеко от предполагае-

мого положения корня. Для решения таких уравнений лучше применять другую встроенную функцию `Minerr` (*см. разд. 6.2*). Аналогичные проблемы могут возникнуть, если начальное приближение выбрано слишком далеко от настоящего решения или `f(x)` имеет особенности типа бесконечности.

ⓘ 5.3.2. Градиентные методы: функция *Find*

Если вы решаете "хорошие" уравнения, как все те, которые были приведены в предыдущих разделах, то, вообще говоря, можете никогда не задумываться, как именно `Find` ищет их корни. Однако даже в этом случае полезно представлять, что происходит "за кадром", т. е. какие действия совершаются в промежутке между введением необходимых условий после ключевого слова `Given` и получением результата после применения функции `Find`. Это важно хотя бы с позиций выбора начальных значений переменных перед вычислительным блоком. Рассмотрим в данном разделе некоторые особенности численных методов и возможности установки их различных параметров, которые предоставляет `Find`.

Принцип действия градиентных алгоритмов

Во встроенной функции `Find` реализовано несколько градиентных численных алгоритмов, один из которых может выбрать либо программа Mathcad, либо сам пользователь. Покажем их основную идею на примере уравнения с одним неизвестным `f(x)=0` для функции $f(x)=x^2+5x+2$, график которой показан на рис. 5.8. Принцип градиентных методов состоит в последовательных приближениях к истинному решению уравнения, которые вычисляются с помощью производной от `f(x)`. Приведем наиболее простую форму алгоритма, называемого методом Ньютона:

1. За нулевую итерацию принимается введенное пользователем начальное значение `x0=x`.
2. В точке `x0` методом конечных разностей вычисляется производная `f'(x0)`.

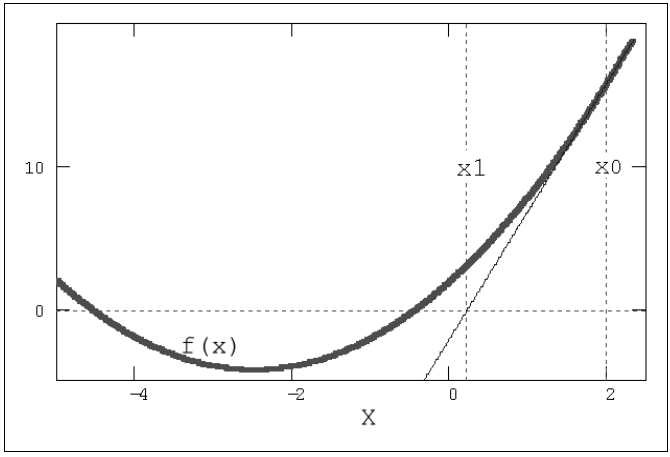


Рис. 5.8. Иллюстрация метода Ньютона

3. Пользуясь разложением Тейлора, можно заменить `f(x)` в окрестности `x0` касательной — прямой линией `f(x)≈f(x0)+f'(x0)·(x-x0)`.
4. Определяется точка `x1`, в которой прямая пересекает ось `X` (см. рис. 5.8).
5. Если `f(x1)<TOL`, то итерации прерываются, и значение `x1` выдается в качестве решения. В противном случае `x1` принимается за новую итерацию, и цикл повторяется: строится касательная к `f(x)` в точке `x1`, определяется `x2` — точка ее пересечения с осью `X` и т. д.

Модификация алгоритма Ньютона для решения системы нескольких уравнений заключается в линеаризации соответствующих функций многих переменных, т. е. аппроксимации их линейной зависимостью с помощью частных производных.

Например, для нулевой итерации в случае системы двух уравнений используются выражения типа:

$$f_1(x, y) \approx f_1(x0, y0) + \frac{\partial f_1(x0, y0)}{\partial x}(x - x0) + \frac{\partial f_1(x0, y0)}{\partial y}(y - y0);$$

$$f_2(x, y) \approx f_2(x0, y0) + \frac{\partial f_2(x0, y0)}{\partial x}(x - x0) + \frac{\partial f_2(x0, y0)}{\partial y}(y - y0).$$

Чтобы отыскать точку, соответствующую каждой новой итерации, требуется приравнять оба равенства нулю, т. е. решить на каждом шаге полученную систему линейных уравнений.

Выбор градиентного алгоритма

Как уже отмечалось, Mathcad предлагает три различных варианта градиентных методов. Чтобы поменять численный метод:

1. Щелкните правой кнопкой мыши на названии функции `Find`.
2. Наведите указатель мыши на пункт **Нелинейный** (Nonlinear) в контекстном меню.
3. В появившемся подменю (рис. 5.9) выберите один из трех методов: **Метод сопряженных градиентов** (Conjugate Gradient), **Квази-Ньютон** (Quasi-Newton) или **Метод Левенберга-Марквардта** (Levenberg-Marquardt).

Чтобы вернуть автоматический выбор типа численного метода, в контекстном меню надо выбрать пункт **Автовыбор** (AutoSelect). Если установлена опция автоматического выбора (о чем свидетельствует флажок), то текущий тип численного метода можно узнать, вызвав то же самое подменю и посмотрев, который из них отмечен точкой. Два последних метода являются квазиньютоновскими, основная идея которых была рассмотрена выше. Первый из них, метод сопряженных градиентов, является двухшаговым — для поиска очередной итерации он использует как текущую, так и предыдущую итерации. Алгоритм Левенберга обстоятельно описан в справочной системе Mathcad, а подробную информацию о методах Ньютона и сопряженных градиентов можно найти в большинстве книг по численным методам.

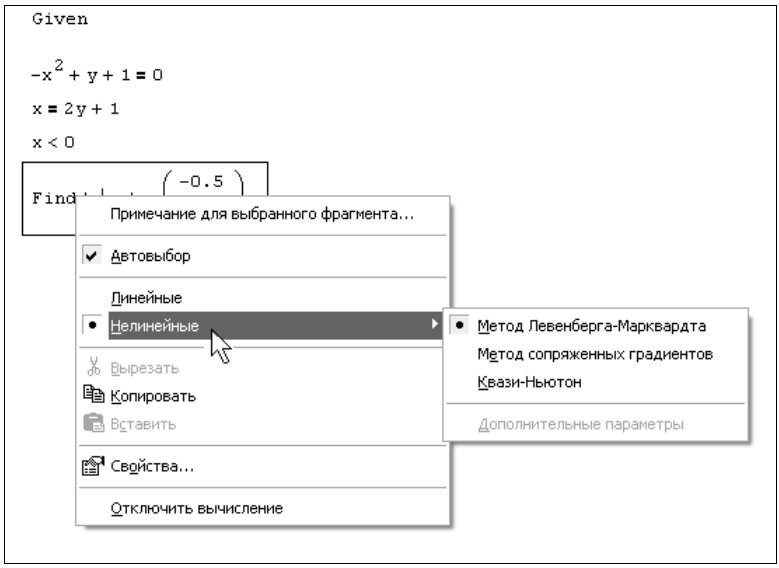


Рис. 5.9. Выбор численного алгоритма

Параметры градиентных алгоритмов

Помимо выбора самих методов имеется возможность устанавливать их некоторые параметры. Для этого нужно вызвать с помощью того же контекстного меню диалоговое окно **Дополнительные параметры** (Advanced Options), выбрав в контекстном меню пункты **Нелинейный / Дополнительные параметры** (Nonlinear / Advanced options). В этом диалоговом окне (рис. 5.10) имеется пять групп переключателей по два в каждой.

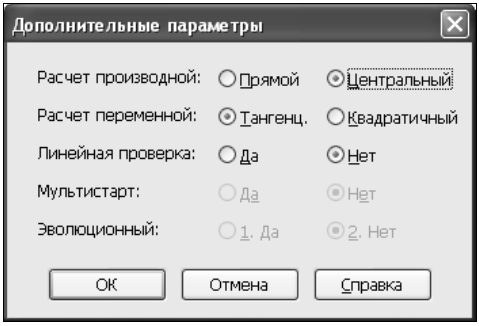


Рис. 5.10. Диалоговое окно **Дополнительные параметры**

В первой строке — **Аппроксимация производной** (Derivative estimation) — определяется метод вычисления производной **Вперед** (Forward) или **Центральная** (Central). Они соответствуют аппроксимации производной либо правой (двухточечная схема "вперед"), либо центральной (трехточечная симметричная схема) конечной разностью.

Примечание

Обратите внимание, что вычисление производной в градиентных численных методах решения уравнений производится более экономичным способом, нежели при численном дифференцировании (*см. главу 3*).

Во второй строке — **Аппроксимация переменных** (Variable estimation) — можно определить тип аппроксимации рядом Тейлора. Для рассмотренного нами в этом разделе случая аппроксимации касательной прямой линией выберите переключатель **Касательная** (Tangent), для более точной квадратичной аппроксимации (параболой) выберите **Квадратичная** (Quadratic). Следующая группа переключателей — **Проверка линейности** (Linear variable check) — позволяет в специфических задачах сэкономить время вычислений. Если вы уверены, что нелинейности всех функций, входящих в уравнение, мало сказываются на значениях всех их частных производных, то установите переключатель **Да** (Yes). В этом случае производные будут приняты равными константам и не будут вычисляться на каждом шаге.

Примечание

В русифицированной версии (рис. 5.10) перевод перечисленных терминов немного отличается от приведенного выше, который, на мой взгляд, лучше отражает смысл параметров.

Пара переключателей **Сканирование** (Multistart) задает опцию поиска глобального или локального минимума или максимума. Если выставлен переключатель **Да** (Yes), Mathcad будет пытаться найти наиболее глубокий экстремум из области, близкой к начальному приближению. Эта опция предназначена, в основном, для настройки (тех же самых градиентных) алгоритмов поиска экстремума, а не для решения алгебраических уравнений (*см. главу 6*).

Наконец, последний переключатель — **Эволюционный алгоритм** (Evolutionary), если установить его в положение **Yes** (Да), позволяет использовать модификацию численного метода для решения уравнений, определяемых не обязательно гладкими функциями. Как мы убедились в этом разделе, все градиентные методы, реализованные в функции `Find`, требуют многократного вычисления производных. Если вы работаете с достаточно гладкими функциями, то градиентные методы обеспечивают быстрый и надежный поиск корня. Для поиска корня недостаточно гладких функций одной переменной, следует либо выставить данную опцию функции `Find`, либо использовать метод секущих (функцию `root`). Помните, что правильный выбор численного метода и его параметров может помочь при решении нестандартной задачи, которая при стандартных установках может и не поддаваться решению.

5.3.3. Метод продолжения по параметру

Решение "хороших" нелинейных уравнений и систем типа тех, которые были рассмотрены в предыдущих разделах этой главы, представляет собой несложную, с вычислительной точки зрения, задачу. В реальных инженерных и научных расчетах очень распространена более сложная проблема: решение не одного уравнения (или системы), а целой серии уравнений, зависящих от некоторого параметра (или нескольких параметров). Для таких задач существуют очень эффективные методы, которые называются *методами продолжения*. Эти методы непосредственно не

встроены в Mathcad, но могут быть легко запрограммированы с помощью уже рассмотренных нами средств. Будем далее говорить об одном уравнении, имея в виду, что всегда возможно обобщение результатов на случай системы уравнений.

Пусть имеется уравнение `f(a,x)=0`, зависящее не только от неизвестного `x`, но и от параметра `a`. Требуется определить зависимость его корня `x` от параметра `a`, т. е. `x(a)`. Простой пример такой задачи был приведен в листинге 5.3 (*см. разд. 5.1.2*). Тогда нам повезло, и решение в общем виде было найдено с помощью символьных вычислений. Рассмотрим еще один, чуть более сложный, пример алгебраического уравнения, зависящего от параметра `a` следующим образом: $\ln(a\cdot x^2)=x$ (рис. 5.11).

Листинг 5.17. Попытка отыскания зависимости `x(a)` решения уравнения $\ln(a\cdot x^2)=x$

$i := 0 .. 30$

$a_i := 2 + i$

$x := 1$

$y_i := \mathrm{root}\left(\ln\left(a_i \cdot x^2\right) - x, x\right)$

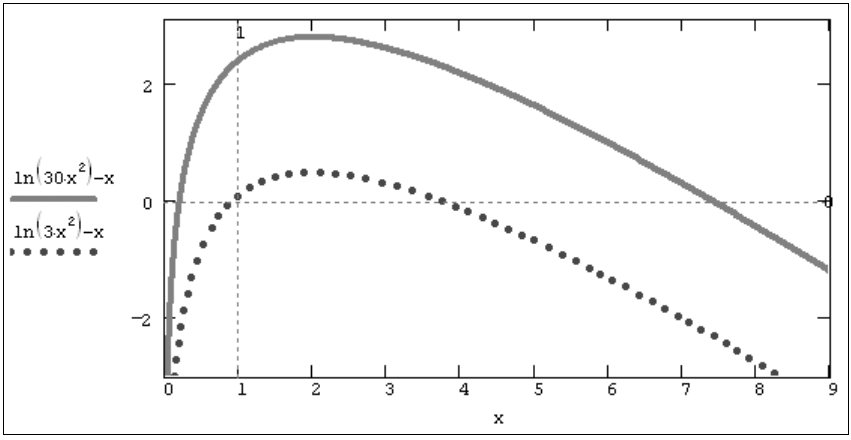


Рис. 5.11. График функции $\ln(a\cdot x2)-x$ (для $a=3$ и $a=30$)

Решим данное уравнение методом секущих, применяя для этого встроенную функцию `root`. Самый простой, но далеко не лучший, способ иллюстрируется листингом 5.17. Начинается листинг с вывода графика функции $\ln(a\cdot x^2)-x$, корни которой нам предстоит исследовать (ради определенности, для положительных значений `x`). Глядя на график, сразу можно сказать, что на рассматриваемом интервале уравнение будет иметь два решения (для каждого значения параметра `a`, бóльших некоторого порогового значения, ниже которого, видимо, уравнение вовсе не имеет корня).

Для того чтобы получить зависимость решения уравнения от параметра `a`, в следующих строках листинга создается ранжированная переменная `i`, с помощью которой определяется вектор значений параметра a_i. Его элементы пробегают значе-

ния от 3 до 33 с шагом 1 (эти числа взяты ради примера, вы можете поэкспериментировать с другими значениями и убедиться в том, что для значений параметра ниже порога $a \cong 3$ решение уравнения отсутствует).

Последняя строка листинга присваивает элементам еще одного вектора y вычисленные с помощью функции root значения корней уравнения для каждого a_i. Но для того чтобы функция root заработала, необходимо предварительно задать начальное приближение к решению, что сделано в предыдущей строке. Ключевой момент метода, примененного в листинге 5.17, заключается в том, что одно и то же начальное значение x=1 использовано для решения уравнения при всех a_i.

Результат расчетов y_i показан на рис. 5.12. Обратите внимание, что по мере увеличения a кривая корней уравнения сначала плавно идет по одному (нижнему) семейству решений, а потом (в районе $a \approx 11$) явно срывается, "перепрыгивая" на другое семейство. С вычислительной точки зрения такая ситуация чаще всего крайне неблагоприятна, поскольку хотелось бы отыскать непрерывное семейство решений. Скачки зависимости y(a) могут вводить пользователя в заблуждение, вовсе скрывая от него существование нижнего семейства решений при $a>11$.

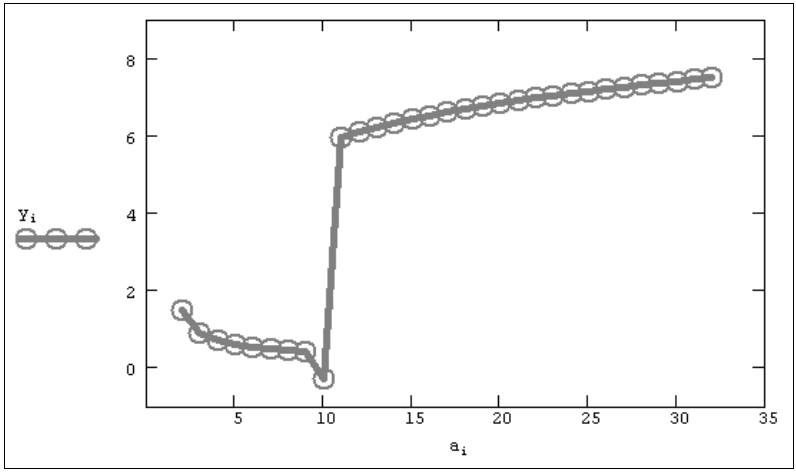


Рис. 5.12. Попытка отыскания зависимости x(a) решения уравнения $\ln(a \cdot x^2)=x$ (продолжение листинга 5.17)

Почему же происходят эти скачки с одного семейства решений на другое? Конечно, причина кроется в выборе начального значения для вычисления каждого из корней. Линия начальных значений x=1 обозначена на графике функции (рис. 5.11) в виде пунктирной вертикальной прямой. Для $a_0=3$, и вообще для нескольких первых a_i, начальное значение x=1 находится ближе всего к нижнему семейству решений. Поэтому неудивительно, что численный метод находит именно эти корни. В правой части графика к линии начальных значений ближе второе (верхнее) семейство решений, к ним-то и приводит численный метод.

Приведенные соображения диктуют очень простой рецепт избавления от скачков и нахождения одного из семейств непрерывных решений. Для этого требуется при

поиске каждого (i+1)-го корня взять начальное значение, по возможности близкое к отыскиваемому семейству. Неплохим вариантом будет выбор приближения в виде предыдущего i-го корня, который был найден для прошлого значения параметра a_i. Возможный вариант воплощения этого метода, называемого *продолжением по параметру*, приведен в листинге 5.18. В нем функция root применена внутри функции пользователя f(x0,a), определенной в самом начале листинга с помощью средств программирования. Назначение функции f(x0,a) заключается в том, что она выдает значение корня для заданного значения параметра a и начального приближения к решению x0. В остальном смысл листинга повторяет предыдущий, за исключением того, что осуществляется поиск сразу обоих семейств решений y и z, причем для каждого сначала явно задается начальное значение только для точки a_0. Для всех последующих точек, как следует из последней строки листинга, взято начальное значение, равное предыдущему корню.

Листинг 5.18. Поиск зависимости x(a) решения уравнения $\ln(a \cdot x^2)=x$ методом продолжения

```
f(x0, a) := | x ← x0
            | root(ln(a·x) − x, x)
i := 0 .. 30
a_i := 3 + i
y_0 := f(10, a_0)          z_0 := f(0.1, a_0)
y_{i+1} := f(y_i, a_i)     z_{i+1} := f(z_i, a_i)
```

Результаты вычислений, приведенные в виде двух графиков на рис. 5.13, разительно отличаются от предыдущего. Как видно, столь малое изменение идеологии применения численного метода привело к определению непрерывного семейства корней. Отметим, что получить результат рис. 5.12 (без продолжения по параметру) в терминах введенной нами в листинге 5.18 функции f(x0,a) можно, изменив ее первый аргумент на константу: f(1,a_i).

Примечание

С помощью метода продолжения можно решать и соответствующие задачи оптимизации, зависящие от параметра. Идеология в этом случае остается точно такой же, но вместо функций решения нелинейных уравнений root или Find вам следует применить одну из функций поиска экстремума Minerr, Maximize или Minimize (*см. разд. 6.1 и 6.2*).

Мы привели основную идею и один из возможных способов реализации метода продолжения по параметру. Безусловно, вы можете предложить иные, как математические, так и программистские решения этой проблемы. В частности, для выбора очередного начального приближения к корню можно использовать результат экстраполяции уже найденной зависимости x(a), придумать более сложные алгоритмы для ветвящихся семейств решений и т. д.

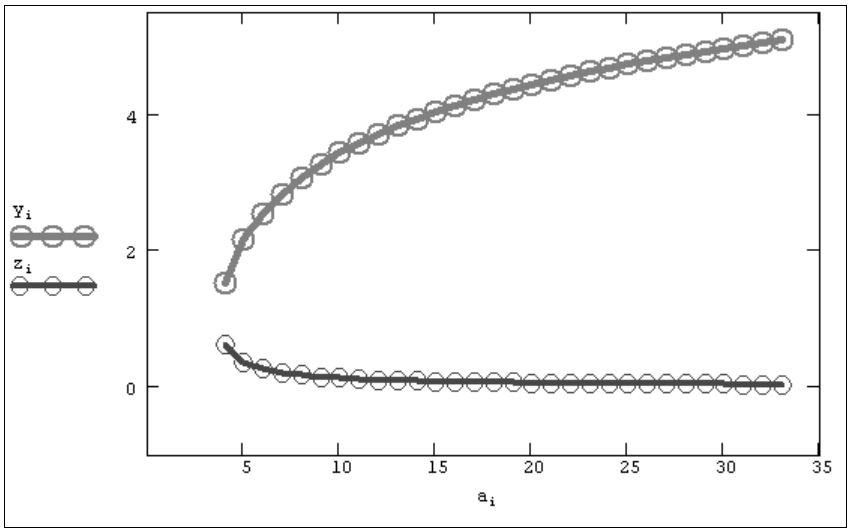


Рис. 5.13. Поиск зависимости `x(a)` решения уравнения `ln(a·x²)=x` методом продолжения (продолжение листинга 5.18)

ГЛАВА 6

Оптимизация

В этой главе рассматриваются задачи на поиск экстремума функций и близкие к ним задачи приближенного решения алгебраических нелинейных уравнений и систем. Задачи поиска *экстремума* функции означают нахождение ее *максимума* (наибольшего значения) или *минимума* (наименьшего значения) в некоторой области определения ее аргументов. С вычислительной точки зрения две задачи являются практически одинаковыми, так как, например, задача поиска максимума `f(x)` тождественна проблеме отыскания минимума `-f(x)`. Поэтому ниже будем часто называть задачу поиска экстремума функции задачей минимизации.

Общая проблема поиска экстремума функции включает в себя задачи нахождения *локального* и *глобального* минимума. Последние называют еще *задачами оптимизации*, и решить их, как правило, намного труднее, поскольку они подразумевают локализацию всех минимумов `f(x)` и выбор из них наименьшего. (На рис. 6.1 показаны два локальных минимума функции, из которых левый является глобальным.) Ограничения значений аргументов, задающих область определения `f(x)`, как и прочие дополнительные условия, могут быть определены в виде системы неравенств и (или) уравнений. В таком случае говорят о задаче на *условный экстремум*.

Численные методы, применяемые для минимизации, сходны с методами решения нелинейных уравнений, и поэтому материал этой главы близок по содержанию к предыдущей.

Примечание

Решение задач минимизации в Mathcad реализовано только при помощи численного алгоритма. Таким образом, непосредственное символьное нахождение минимума невозможно. Однако аналитический поиск экстремума функции несложно запрограммировать, опираясь на соответствующие сведения математического анализа (*см. разд. 6.1.5*).

6.1. Поиск экстремума функции

Для численного решения задач поиска локального максимума и минимума в Mathcad имеются встроенные функции `Minerr`, `Minimize` и `Maximize`. Принцип их

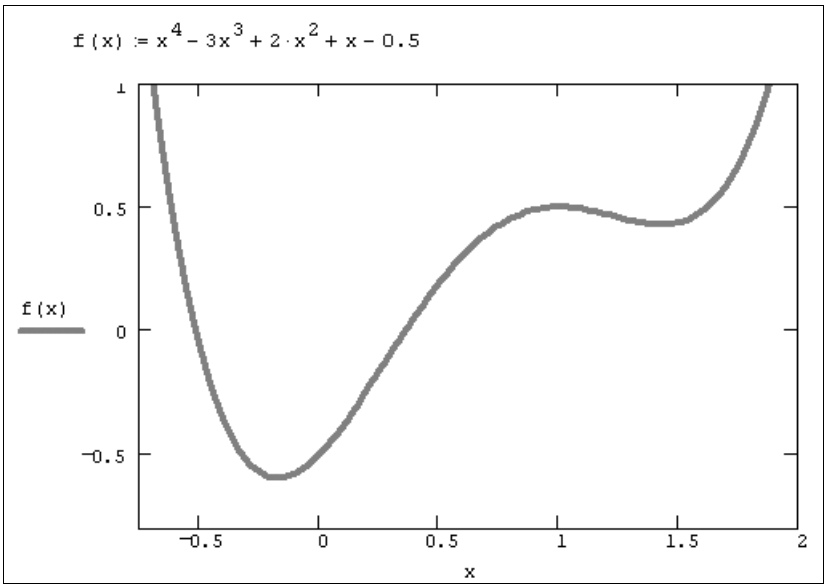


Рис. 6.1. К пояснению задач поиска локального и глобального экстремума

действия очень близок к принципу расчетов, заложенных во встроенной функции `Find`, предназначенной для решения алгебраических уравнений (*см. главу 5*). В частности, все встроенные функции минимизации используют те же градиентные численные методы, что и функция `Find`, поэтому допускается "вручную" выбирать численный алгоритм минимизации из уже рассмотренных нами численных методов (*см. разд. 5.3.2*). Кроме того, как и в случае решения уравнений, применение градиентного алгоритма, во-первых, требует задания некоторого начального приближения к точке минимума и, во-вторых, позволяет отыскать лишь один (т. е. локальный) из минимумов функции.

Таким образом, как и в случае решения уравнений (*см. разд. 5.2.4*), чтобы найти глобальный максимум (или минимум), требуется сначала *просканировать* с некоторым шагом рассматриваемую область и вычислить все локальные значения и потом выбрать из них наибольший (наименьший). Другим вариантом будет простое сканирование с вычислением значений функции, позволяющее выделить из нее подобласть наибольших (наименьших) значений функции и осуществить поиск глобального экстремума, уже находясь в его окрестности. Последний путь таит в себе некоторую опасность уйти в зону другого локального экстремума, но часто может быть предпочтительнее из соображений экономии времени.

6.1.1. Локальный экстремум

Для поиска локальных экстремумов имеются две встроенные функции, которые могут применяться как в пределах вычислительного блока, так и автономно:

- `Minimize(f,`x_1`,...,`x_M`)` — вектор значений аргументов, при которых функция `f` достигает минимума;

- `Maximize(f,x₁,...,xₘ)` — вектор значений аргументов, при которых функция `f` достигает максимума:
  - `f(x₁,...,xₘ,...)` — функция;
  - $x_1, \ldots, x_M$ — аргументы, по которым производится минимизация (максимизация).

  Примечание

  Вычислительный блок (ключевое слово `Given` со следующими после него логическими выражениями) обычно используется в задачах на условный экстремум (*см. следующий раздел*).

В качестве примера рассмотрим задачу численного поиска экстремумов полинома четвертой степени `f(x)`, график которого был приведен на рис. 6.1. Как известно, парабола четвертой степени имеет три точки экстремума, и все они видны на рис. 6.1.

Всем аргументам функции `f` предварительно следует присвоить некоторые значения, причем для тех переменных, по которым производится минимизация, они будут восприниматься как начальные приближения. Примеры вычисления локальных экстремумов функции одной переменной показаны в листингах 6.1 и 6.2. Поскольку никаких дополнительных условий в них не вводится, поиск экстремумов выполняется для любых значений `x` от $-\infty$ до ∞.

Листинг 6.1. Поиск минимума функции одной переменной (для трех начальных значений x)

$f(x) := x^4 - 3x^3 + 2 \cdot x^2 + x - 0.5$

$x := 0$

$\text{Minimize}(f, x) = -0.175$

$x := 100$

$\text{Minimize}(f, x) = 1.425$

$x := 1.426$

$\text{Minimize}(f, x) = -0.175$

Листинг 6.2. Поиск максимума функции одной переменной

$f(x) := x^4 - 3x^3 + 2 \cdot x^2 + x - 0.5$

$x := 5$

$\text{Maximize}(f, x) = 1$

$x := 10$

$\text{Maximize}(f, x) = \blacksquare$

Как видно из листингов, существенное влияние на результат оказывает выбор начального приближения, в зависимости от чего в качестве ответа выдаются различ-

ные локальные экстремумы. Очень полезно сопоставить результаты минимизации (листинг 6.1) с графиком функции `f(x)` (см. рис. 6.1). Как видно, функция `Minimize` очень уверенно находит глубокий минимум `x=-0.75`. А вот на второй (плохо выраженный) минимум можно набрести лишь случайно, выбирая определенные начальные значения `x`. В последнем из трех примеров демонстрируется, что если взять начальное приближение `x` даже в непосредственной близости от этого локального минимума, численный метод все равно "сваливается" в первый, более глубокий минимум `f(x)`. Попробуйте повторить расчеты, выбирая различные начальные значения, чтобы в этом убедиться.

В листинге 6.2 показаны аналогичные свойства функции `Maximize`. Если начальное приближение выбрать удачно, то итерационный процесс алгоритма сойдется к максимуму функции, а вот если выбрать его вдали от него, на участке `f(x)`, где неограниченно возрастает (при $x \to \pm\infty$), численный метод вообще не справится с задачей, выдавая сообщение об ошибке. Это происходит, поскольку начальное приближение `x=-10` выбрано далеко от области локального максимума, и поиск решения уходит в сторону увеличения `f(x)`, т. е. расходится при $x \to \infty$.

ⓘ ⑪ *Примечание*

Помните о возможности выбора численного алгоритма минимизации, который осуществляется при помощи контекстного меню (рис. 6.2). Не забывайте также, что, начиная с версии Mathcad 11, имеется возможность управлять параметром **Сканирование** (Multistart), при помощи которого можно попытаться организовать поиск глобального экстремума (*см. разд. 5.3.2*). Однако не слишком полагайтесь на эту опцию и, если перед вами стоит задача поиска глобального экстремума, постарайтесь организовать сканирование вручную.

Рис. 6.2. Выбор численного метода минимизации

6.1.2. Условный экстремум

В задачах на условный экстремум встроенные функции минимизации и максимизации должны быть включены в вычислительный блок, т. е. им должно предшествовать ключевое слово `Given`. В промежутке между `Given` и функцией поиска экстремума

с помощью булевых операторов записываются логические выражения (неравенства, уравнения), задающие ограничения на значения аргументов минимизируемой функции. Листинги 6.3 и 6.4 содержат примеры поиска условного экстремума на различных интервалах, определенных неравенствами. У вас не должны возникнуть сложности с записью условий в Mathcad, а вот разобраться в скрытой от глаз вычислительной стороне будет полезно. С этой точки зрения поучительно сравнить результаты работы приведенных листингов с листингами 6.1 и 6.2 соответственно.

Листинг 6.3. Поиск условного минимума

$f(x) := x^4 - 3x^3 + 2 \cdot x^2 + x - 0.5$

$x := 0$

Given

$1 < x < \infty$

$\text{Minimize}(f, x) = 1.425$

Листинг 6.4. Поиск условного максимума

$f(x) := x^4 - 3x^3 + 2 \cdot x^2 + x - 0.5$

$x := 1$

Given

$2 < x < 10$

$\text{Maximize}(f, x) = 10$

Как видно из листинга 6.3, если ограничить значения `x` интервалом, расположенным в окрестности правого локального максимума, с поиском которого мы встретили большие сложности при решении задачи на безусловный экстремум, то этот максимум будет без труда найден численным методом. Следует помнить также об одной особенности, иллюстрируемой листингом 6.4. А именно (в случае максимизации), если на границе интервала `f(x)` достигает большего значения, нежели на локальном максимуме внутри интервала, то в качестве решения, скорее всего, будет выдано наибольшее значение (т. е. граница интервала).

Конечно, если на рассматриваемом интервале `x` расположено несколько локальных максимумов, ответ станет еще менее предсказуемым, поскольку будет напрямую зависеть от выбранного начального приближения. Не забывайте о важности его правильного выбора и в случае задач на условный экстремум. В частности, если вместо начального значения `x=-1` задать `x=1`, то в качестве максимума будет найдена правая граница интервала: `Maximize(f,x)=2`, что неверно, поскольку максимальное значение достигается функцией `f(x)` на левой границе интервала при `x=-3`. Выбор начального приближения `x=-4` решает задачу правильно, выдавая в качестве результата `Maximize(f,x)=-5`.

6.1.3. Экстремум функции нескольких переменных

Вычисление экстремума функции многих переменных не несет принципиальных особенностей по сравнению с функциями одной переменной. Поэтому ограничимся примером нахождения максимума и минимума функции, показанной в виде графиков трехмерной поверхности и линий уровня (листинг 6.5). Привлечем внимание читателя только к тому, как с помощью неравенств, введенных логическими операторами, задается область на плоскости `(X,Y)`.

Листинг 6.5. Экстремум функции двух переменных

$$f(x,y) := 2\cdot(x-5.07)^2 + (y-10.03)^2 - 0.2\cdot(x-5.07)^3$$

$$x := 1$$

$$y := 1$$

Given

$$0 < x < 15$$

$$0 < y < 20$$

$$\text{Minimize}(f,x,y) = \begin{pmatrix} 5.07 \\ 10.03 \end{pmatrix}$$

Дополнительные условия могут быть заданы и равенствами. Например, определение после ключевого слова `Given` уравнения `x+y=10` приводит к такому решению задачи на условный экстремум:

$$\text{Minimize}(f,x,y) = \begin{pmatrix} 3.589 \\ 6.411 \end{pmatrix}$$

Как нетрудно сообразить, новое дополнительное условие означает, что численный метод ищет минимальное значение функции `f(x,y)` не во всей прямоугольной области `(X,Y)`, а лишь вдоль отрезка прямой, показанного на рис. 6.3.

Примечание

Поиск минимума можно организовать и с помощью функции `Minerr`. Для этого в листинге 6.5 надо поменять имя функции `Minimize` на `Minerr`, а после ключевого слова `Given` добавить выражение, приравнивающее функции `f(x,y)` значение, заведомо меньшее минимального, например, `f(x,y)=0`.

6.1.4. Пример: линейное программирование

Задачи поиска условного экстремума функции многих переменных часто встречаются в экономических расчетах для минимизации издержек, финансовых рисков, максимизации прибыли и т. п. Целый класс экономических задач оптимизации описывается системами линейных уравнений и неравенств. Они называются задачами *линейного программирования*. Приведем характерный пример так называемой *транспортной задачи*, которая решает одну из проблем оптимальной организации доставки товара потребителям с точки зрения экономии транспортных средств (листинг 6.6).

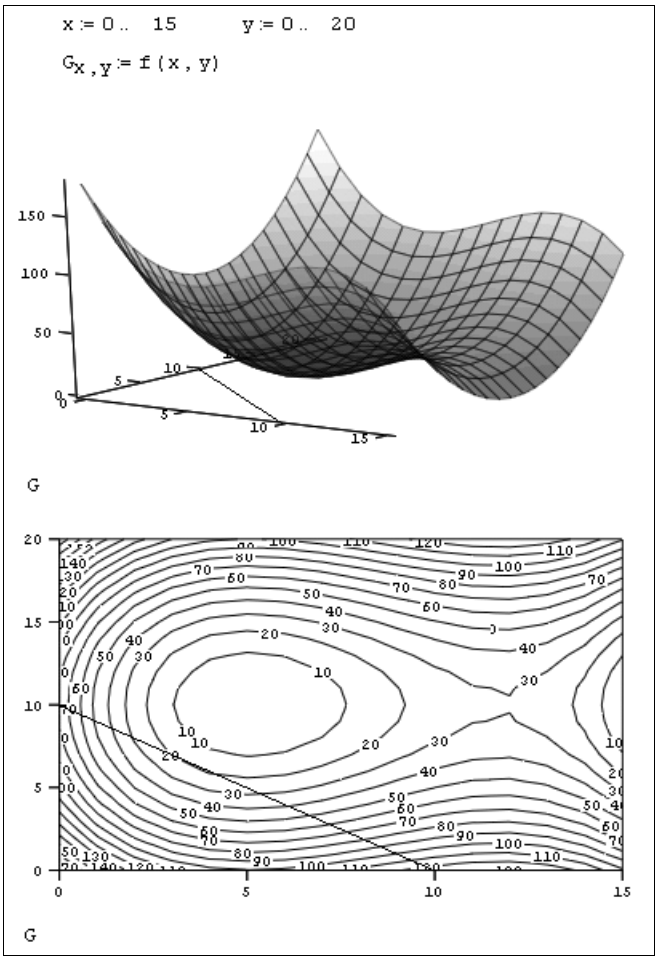


Рис. 6.3. Иллюстрация задачи на условный экстремум функции двух переменных: график функции `f(x,y)` и отрезок прямой `x+y=10` (продолжение листинга 6.5)

Листинг 6.6. Решение задачи линейного программирования

$$a := \begin{pmatrix} 145 \\ 210 \\ 160 \end{pmatrix} \qquad b := \begin{pmatrix} 237 \\ 278 \end{pmatrix}$$

$$\sum a = 515 \qquad \sum b = 515$$

$M := \mathrm{rows}(a) \qquad N := \mathrm{rows}(b)$

$M = 3 \qquad N = 2$

$$c := \begin{pmatrix} 11.5 & 7 & 12 \\ 6.2 & 10 & 9.0 \end{pmatrix}$$

$$f(x) := \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} c_{i,j} \cdot x_{i,j}$$

$$x_{N-1,M-1} := 0$$

Given

$$x_{0,0} + x_{0,1} + x_{0,2} = b_0 \quad x_{0,0} + x_{1,0} = a_0 \quad x_{0,0} \ge 0 \quad x_{1,0} \ge 0$$

$$x_{1,0} + x_{1,1} + x_{1,2} = b_1 \quad x_{0,1} + x_{1,1} = a_1 \quad x_{0,1} \ge 0 \quad x_{1,1} \ge 0$$

$$x_{0,2} + x_{1,2} = a_2 \quad x_{0,2} \ge 0 \quad x_{1,2} \ge 0$$

$$sol := \text{Minimize}(f, x)$$

$$sol = \begin{pmatrix} 0 & 210 & 27 \\ 145 & 0 & 133 \end{pmatrix} \qquad f(sol) = 3.89 \times 10^3$$

Модель типичной транспортной задачи следующая. Пусть имеется N предприятий-производителей, выпустивших продукцию в количестве $b_0, \ldots, b_{N-1}$ тонн. Эту продукцию требуется доставить M потребителям в количестве $a_0, \ldots, a_{M-1}$ тонн каждому. Численное определение векторов a и b находится в первой строке листинга. Сумма всех заказов потребителей a_i равна сумме произведенной продукции, т. е. всех b_i (проверка равенства во второй строке). Если известна стоимость перевозки тонны продукции от i-го производителя к j-му потребителю c_{ij}, то решение задачи задает оптимальное распределение соответствующего товаропотока x_{ij} с точки зрения минимизации суммы транспортных расходов. Матрица c и минимизируемая функция $f(x)$ матричного аргумента x находятся в середине листинга 6.6.

Условия, выражающие неотрицательность товаропотока, и равенства, задающие сумму произведенной каждым предприятием продукции и сумму заказов каждого потребителя, находятся после ключевого слова `Given`. Решение, присвоенное матричной переменной `sol`, выведено в последней строке листинга вместе с соответствующей суммой затрат. В строке, предшествующей ключевому слову `Given`, определяются нулевые начальные значения для x простым созданием нулевого элемента матрицы $x_{N-1,M-1}$.

Примечание 1

Если взять другие начальные значения для x, решение, скорее всего, будет другим! Возможно, вы сумеете отыскать другой локальный минимум, который еще больше минимизирует транспортные затраты. Это еще раз доказывает, что задачи на глобальный минимум, к классу которых относится линейное программирование, требуют аккуратного отношения в смысле выбора начальных значений. Часто ничего другого не остается, кроме сканирования всей области начальных значений, чтобы из множества локальных минимумов выбрать наиболее глубокий.

Примечание 2

Не забывайте о том, что в вычислительном блоке в качестве неизвестных следует использовать все компоненты вектора x, как это сделано в предпоследней строке листинга 6.6 (*см. разд. 5.1.1*). Иными словами, нельзя задавать некоторые из компонент x в виде параметров.

ПРИМЕЧАНИЕ 3

Еще раз напомним, что в новых версиях Mathcad (начиная с 11-й) появилась возможность программного управления глобальной минимизацией. Для этого служит параметр численного алгоритма **Сканирование** (Multistart), который следует установить в положение **Да** (Yes) для поиска глобального экстремума (*см. разд. 5.3.2*). Любопытно, что если при решении нашей задачи поменять только эту опцию функции `Minimize`, то численный метод даст сбой, и вместо решения появится сообщение об ошибке. Однако если вспомнить, что наша транспортная задача — линейная, и выставить в контекстном меню, вызываемом на имени функции `Minimize`, флажок **Линейный** (Linear), задающий линейный вариант алгоритма, то глобальное решение будет найдено.

6.1.5. Аналитическое решение задач на экстремум

Несмотря на то, что, как уже говорилось, разработчиками Mathcad символьное решение задач оптимизации не предусмотрено, пользователь все-таки имеет возможность аналитического исследования экстремумов функций, опираясь на базовые сведения математического анализа. Следует лишь вспомнить о том, что (при выполнении соответствующих условий на непрерывность и гладкость функции) точки экстремума `f(x)` характеризуются тем, что в них производная этой функции проходит через нулевое значение. Тип экстремума (максимум или минимум) определяется знаком второй производной в этой точке.

Таким образом, имея в виду данные правила, не представляет особого труда организовать аналитическое решение задачи на экстремум, центральным моментом которого будет решение алгебраического уравнения `f'(x)=0`. Сразу стоит подчеркнуть, что можно использовать гибрид символьных и аналитических расчетов, когда, например, производная `f'(x)` считается аналитически, а уравнение `f'(x)=0` (если символьное решение получить не удается) — численно. В этом случае во всей красе может проявиться мощь Mathcad, предоставляющего пользователю богатый арсенал как аналитических, так и численных методов.

Приведем несложный пример реализации аналитического поиска экстремумов той же функции (полинома 4-й степени), которая использовалась нами при демонстрации возможностей численных методов (*см. разд. 6.1.1*). В листинге 6.7, в первой строке, приводится определение `f(x)`, а затем при помощи символьного процессора осуществляется отыскание корней нелинейного уравнения `f'(x)=0`. Результатом решения являются все три точки экстремума (последняя строка листинга). На рис. 6.4 показан график функций `f(x)` и `f'(x)`, и легко убедиться, что экстремумам исходной функции соответствуют нули ее производной.

Листинг 6.7. Аналитический поиск экстремумов функции одной переменной

$$f(x) := x^4 - 3x^3 + 2\cdot x^2 + x - 0.5$$

Given

$$\frac{d}{dx} f(x) = 0$$

$$\text{Find}(x) \rightarrow \left(1 \quad \frac{5}{8} + \frac{1}{8}\cdot\sqrt{41} \quad \frac{5}{8} - \frac{1}{8}\cdot\sqrt{41}\right) = (1 \quad 1.425 \quad -0.175)$$

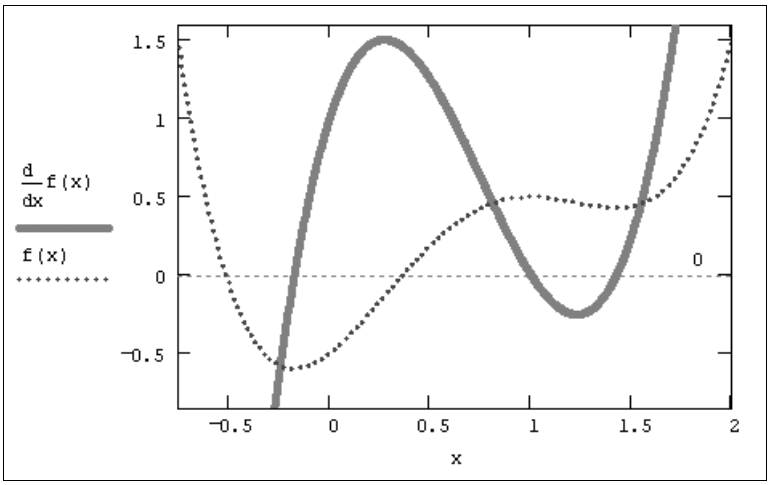


Рис. 6.4. Функция `f(x)` и ее производная (продолжение листинга 6.7)

Чтобы завершить анализ `f(x)` на экстремумы, необходимо определить, какие из найденных точек являются точками минимума, а какие — максимума. Для этого (листинг 6.8) следует просто рассчитать значения второй производной в результатах решения уравнения и определить ее знак.

Листинг 6.8. Анализ типа точек экстремума (продолжение листинга 6.7)

$$x := 1 \qquad \mathrm{sign}\left(\frac{d^2}{dx^2} f(x)\right) \cdot \left|f(x)\right| = -0.5$$

$$x := \frac{5}{8} + \frac{1}{8} \cdot \sqrt{41} \qquad \mathrm{sign}\left(\frac{d^2}{dx^2} f(x)\right) \cdot \left|f(x)\right| = 0.429$$

$$x := \frac{5}{8} - \frac{1}{8} \cdot \sqrt{41} \qquad \mathrm{sign}\left(\frac{d^2}{dx^2} f(x)\right) \cdot \left|f(x)\right| = 0.597$$

ⓘ 6.2. Приближенное решение алгебраических уравнений

Градиентные численные методы решения задач отделения корней уравнений и поиска экстремума функций очень близки. Поэтому, в частности, пользователь может тем же самым образом, с помощью контекстного меню, выбирать конкретный метод приближенного решения для функций `Minimize` и `Maximize`. Применительно к нелинейным уравнениям основная идея градиентных алгоритмов была приведена в *разд. 5.3.2.* Основным отличием в случае задач минимизации является критерий правильности решения (прекращения итераций): если при решении уравнений кри-

терием служит близость нулю невязки системы уравнений, то при минимизации критерий заключается в схождении итераций к минимальным значениям исследуемой функции.

Близость рассмотренных задач связана также с тем, что иногда приходится заменять проблему решения нелинейных уравнений задачей поиска экстремума функции многих переменных. Например, когда невозможно найти решение с помощью функции `Find`, можно попытаться потребовать вместо точного выполнения уравнений условий минимизировать их невязку. Чтобы не нужно было реализовывать данную схему вручную (вместо встроенной функции `Find` использовать соответствующую постановку задачи для функции `Minimize`), разработчики Mathcad предусмотрели дополнительную встроенную функцию `Minerr`. Она применяется аналогично функции `Find` (*см. разд. 5.1.1*), в частности, имеет тот же самый набор параметров и должна находиться в пределах вычислительного блока:

1. $x_1:=C_1$ ... $x_M:=C_M$ — начальные значения для неизвестных.
2. `Given` — ключевое слово.
3. Система алгебраических уравнений и неравенств, записанная логическими операторами.
4. `Minerr(`x_1`,...,`x_M`)` — приближенное решение системы относительно переменных $x_1, \ldots, x_M$, минимизирующее невязку системы уравнений.

> **Примечание 1**
>
> В функции `Minerr` реализованы те же самые алгоритмы, что и в функции `Find`, иным является только условие завершения работы численного метода (как в случае функций `Minimize` и `Maximize`).

> **Примечание 2**
>
> Согласно своему математическому смыслу, функция `Minerr` может применяться для построения регрессии серии данных по закону, заданному пользователем (*см. главу 13*).

Пример использования функции `Minerr` показан в листинге 6.9. Как видно, достаточно заменить в вычислительном блоке имя функции на `Minerr`, чтобы вместо точного (конечно, с поправкой на погрешность порядка `TOL`) получить приближенное решение уравнения, заданного после ключевого слова `Given`.

Листинг 6.9. Приближенное численное решение уравнения, имеющего корень (x=0, y=0)

$x := 1$ $\qquad$ $y := 1$

$k := 10^6$

Given

$k \cdot x^2 + y^2 = 0$

$\text{MinErr}(x, y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Листинг 6.9 демонстрирует приближенное решение уравнения $k \cdot x^2+y^2=0$, которое при любом значении коэффициента k имеет единственный точный корень $(x=0, y=0)$. Тем не менее при попытке решить его функцией `Find` для больших k, порядка принятых в листинге (рис. 6.5), происходит генерация ошибки "No solution was found" (Решение не найдено). Это связано с иным поведением функции $f(x,y)=k \cdot x^2+y^2$ вблизи ее корня по сравнению с функциями, поиск корней которых был разобран в предыдущей главе (см. рис. 5.1).

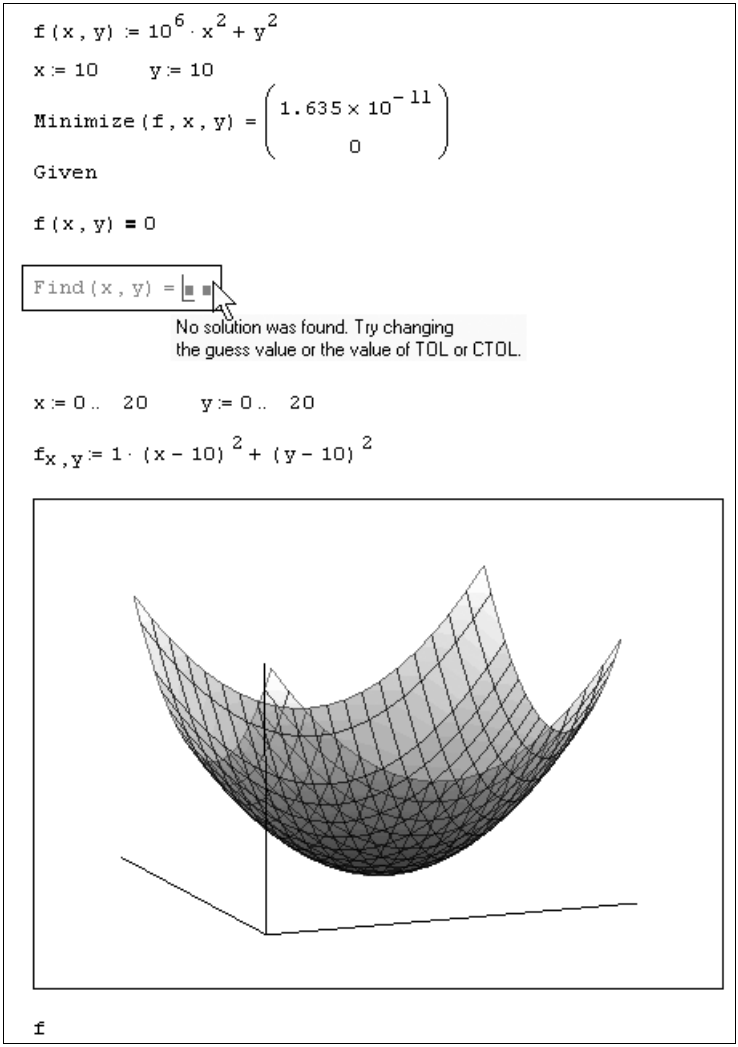


Рис. 6.5. Поиск минимума и попытка нахождения корня функции $f(x,y)=k \cdot x^2+y^2$

В отличие от них, $f(x,y)$ не пересекает плоскость $f(x,y)=0$, а лишь касается ее (рис. 6.5) в точке $(x=0, y=0)$. Поэтому и найти корень изложенными в предыдущем разделе градиентными методами намного сложнее, поскольку вблизи корня произ-

водные f(x,y) близки к нулю, и итерации, строящиеся по градиентному принципу, могут уводить предполагаемое решение далеко от корня.

Ситуация еще более ухудшается, если наряду с корнем типа касания (как на рис. 6.5) имеются (возможно, весьма удаленные) корни типа пересечения. Тогда попытка решить уравнение или систему уравнений с помощью функции Find может приводить к нахождению корня второго типа, даже если начальное приближение было взято очень близко к первому. Поэтому, если вы предполагаете, что система уравнений имеет корень типа касания, намного предпочтительнее использовать функцию Minerr, тем более что всегда есть возможность проконтролировать невязку уравнений простой подстановкой полученного решения в исходную систему уравнений.

Пока мы рассматривали пример нахождения существующего решения уравнения. Приведем теперь пример нахождения функцией Minerr приближенного решения, не имеющего корней уравнения (листинг 6.10), а также несовместной системы уравнений и неравенств (листинг 6.11). Решение, выдаваемое функцией Minerr, минимизирует невязку данной системы. Как видно из листингов, в качестве результата выдаются значения переменных, наилучшим образом удовлетворяющие уравнению и неравенствам внутри вычислительного блока.

Внимание!

Полученное в листинге 6.11 решение не удовлетворяет неравенствам, составляющим задачу. Это и неудивительно, поскольку точного решения системы нет, и в качестве ответа Mathcad выдает значения аргументов, минимизирующих норму общей невязки (не отдавая предпочтения ни уравнению, ни неравенствам).

Листинг 6.10. Приближенное решение уравнения $x^2+y^2+1=0$

$x := 10 \qquad y := 10$

Given

$x^2 + y^2 + 1 = 0$

$\mathrm{MinErr}(x, y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

Листинг 6.11. Приближенное решение несовместной системы уравнений и неравенств

$x := 10 \qquad y := 10$

Given

$x^2 + y^2 + 1 = 0$

$x > 1$

$y > 2$

$\mathrm{MinErr}(x, y) = \begin{pmatrix} 0.125 \\ 2 \end{pmatrix}$

Внимательный читатель может обнаружить, что решение, выдаваемое функцией `Minerr` в рассматриваемом примере, не является единственным, поскольку множество пар значений `(x,y)` в равной степени минимизирует невязку данной системы уравнений и неравенств. Поэтому для различных начальных значений будут получаться разные решения, подобно тому, как разные решения выдаются функцией `Find` в случае бесконечного множества корней (*см. разд. 5.2.4*). Еще более опасен случай, когда имеются всего несколько локальных минимумов функции невязки. Тогда неудачно выбранное начальное приближение приведет к выдаче именно этого локального минимума, несмотря на то, что другой (глобальный) минимум невязки может удовлетворять системе гораздо лучше.

6.3. Пример: некорректные задачи

Еще один широко распространенный круг задач на решение систем уравнений, называемых *обратными*, наиболее типичен для современной экспериментальной физики. Значительную часть обратных задач относят к классу некорректно поставленных (или просто *некорректных*), которые, благодаря своей принципиальной неустойчивости, требуют специального подхода. Как правило, общим принципом решения некорректных задач является их регуляризация, заключающаяся в их сведении к задаче минимизации.

Рассмотрим сначала типичную постановку обратных задач, а затем обратимся к обсуждению корректности их постановки. Строго говоря, обратные задачи связаны, как правило, не с отысканием значения корня уравнения (или системы) `f(x)=0`, а с вычислением неизвестной функции `y(x)`, описываемой уравнением

$$A[y(x)]=b(x). \tag{6.1}$$

Здесь `A[y]` — некоторый *функционал* (т. е. функция от функции), а `b(x)` — известная функция (правая часть уравнения).

Рассмотрим типичную постановку обратных задач на примере так называемой инструментальной задачи. Пусть имеется некоторый сигнал `y(x)`, который подвергается измерению на приборе, условно обозначаемом `S`. Физику-исследователю доступно измерение данного сигнала, которое находится на выходе прибора (дисплее, табло и т. п.). Обозначим это измерение `b(x)`. Поскольку прибор вносит в сигнал, во-первых, искажения (например, в устройствах типа спектрометров часто измеряются некоторые интегральные характеристики сигнала) и, во-вторых, шумовую компоненту. Формально данную физическую модель можно записать равенством (6.1), в котором оператором `S` обозначается *аппаратная функция*, т. е. действие прибора на сигнал `y(x)`.

Листинг 6.12. Пример моделирования прямой задачи, выражающей линейную схему измерений

$y(x) := \cos(x)$

$k := 20 \qquad \sigma := 0.5 \qquad A(x) := \exp\left(-k\cdot x^2\right)$

$$S(x) := \int_0^{10} A\left(\left|x-\chi\right|\right)\cdot y(\chi)\ d\chi + \sigma\cdot\left(\mathrm{rnd}(1) - \frac{1}{2}\right)$$

$x := 0, 0.1 .. 10$

$b(x) := S(x)$

Согласно изложенной модели, измерения `b(x)` могут довольно сильно отличаться от исходного сигнала `y(x)`, что иллюстрируется простым примером листинга 6.12 и рис. 6.6. В первой строке листинга выбирается модельный сигнал `y(x)`, а во второй и третьей — определяется оператор `S[y(x)]`. Важно отметить, что использованная структура оператора — интегральная зависимость от сигнала `y(x)` в сумме с шумовой компонентой — наиболее типична для инструментальных обратных задач. Читателю будет очень полезно "поиграть" с параметрами задачи `k` и σ (эффективной шириной спектральной характеристики прибора и интенсивностью шума соответственно), чтобы "почувствовать" специфику модели измерений.

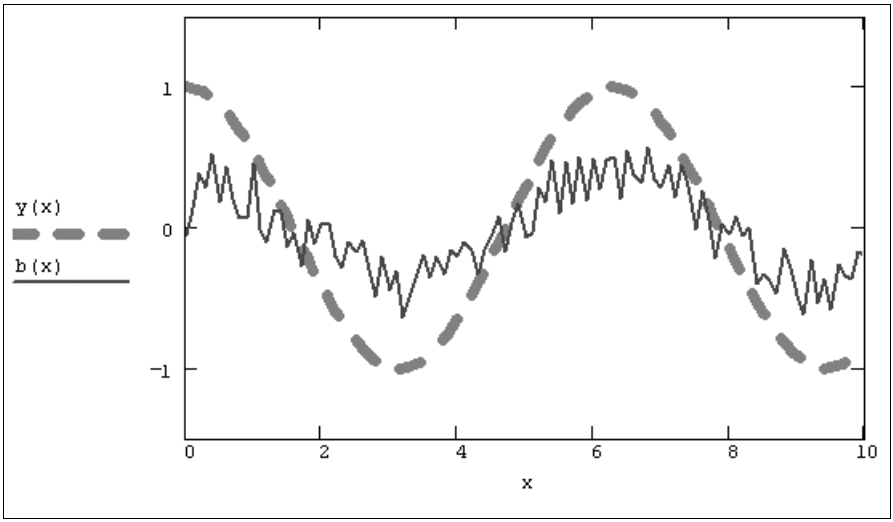


Рис. 6.6. Исходный сигнал и показания прибора (продолжение листинга 6.12)

Примечание

Модель измерений, представленная в третьей строке листинга 6.13, описывает, с математической точки зрения, типичное *интегральное уравнение*, в которое неизвестная функция `y(x)` входит в виде части подынтегральной функции. Класс обратных задач чаще всего (но не всегда) соответствует как раз интегральным уравнениям.

Из сказанного выше понятно, что в практических приложениях весьма актуальна задача восстановления сигнала `y(x)` по показаниям прибора `b(x)` (при наличии определенной дополнительной информации о физике измерения, т. е. об операторе `S`, выражающем действие прибора). Таким образом, в отличие от прямой задачи, выражающейся равенством (6.1), обратной задачей является восстановление функции `y(x)` по известной `b(x)`.

Обычно сигнал (и, соответственно, его измерение) может зависеть от времени и/или пространственных координат. Эту зависимость мы обозначили как зависимость от x. При использовании численных методов непрерывные зависимости от x дискретизируются, заменяясь соответствующими векторами. Таким образом, задача отыскания неизвестной функции $y(x)$ заменяется задачей поиска неизвестного вектора Y и может быть записана в матричном виде

$$S(Y)=B, \tag{6.2}$$

где вектор y неизвестен, а оператор S (в линейном случае $S(Y)=A\cdot Y$, где A — матрица) и вектор правых частей уравнений B известны. Таким образом, в дискретном случае обратная задача для рассматриваемой математической модели состоит в решении системы (в общем случае, нелинейных) уравнений (6.2). Построенная описанным способом дискретная задача представлена в листинге 6.13. Важно отметить, что, поскольку исходная задача линейна, дискретный вариант ее оператора записывается в виде матрицы A, а сама задача сводится к системе линейных алгебраических уравнений $A\cdot Y=B$.

Примечание

Рекомендуем читателю обратиться к соответствующему файлу Mathcad на прилагаемом к книге компакт-диске и сравнить графики сигнала и измерений, соответствующие непрерывной (интегральной) и дискретной (матричной) модели. Будет очень полезным осуществить расчеты для разных значений параметров, в том числе количества точек дискретизации N, а также разобраться, какова структура матрицы A.

Листинг 6.13. Дискретная форма прямой задачи линейной модели измерений (продолжение листинга 6.12)

$N := 100$ $\qquad h := \frac{10}{N}$

$i := 0 .. N$ $\qquad j := 0 .. N$

$X_i := i \cdot h$ $\qquad A_{i,j} := A\left(\left|X_i - X_j\right|\right) \cdot h$

$Y_i := y\left(X_i\right)$

$B_i := b\left(X_i\right)$

При решении обратных задач важную роль играет их устойчивость. Задача устойчива, если малым флуктуациям правых частей, т. е. вектора B, соответствуют малые флуктуации решения Y. Иными словами, *устойчивость по правой части* состоит в требовании, чтобы решения близких задач $A\cdot Y=B$ и $A\cdot Y=B+\delta B$ мало отличались друг от друга. Если задача изначально является неустойчивой, то решать ее нет смысла, поскольку погрешности алгоритмов, накапливающиеся в ходе решения численными методами, неизбежно приведут к тому, что будет найдено неверное решение.

Как правило (в том числе в нашем примере), обратные задачи характеризуются наличием шумов, что может быть символически отражено равенством (если опять-таки предположить, что шум σ входит в схему измерений линейно):

$$A\cdot Y+\sigma=B. \tag{6.3}$$

Наличие шума коренным образом меняет идеологию решения обратных задач. Если сама задача является устойчивой, то существование шума может эту устойчивость нарушать. Попросту говоря, различные (даже очень сильно отличающиеся) сигналы `Y1` и `Y2` могут, будучи искажены шумом, приводить к очень похожим измерениям `B1~B2`. Поэтому встает вопрос, можно ли извлечь из измерений полезную информацию о сигнале, если наличие шума делает задачу неустойчивой? Такие задачи называются задачами, *поставленными некорректно*, и для их решения развиты специальные методы, основанные на привлечении дополнительной априорной информации о решении `Y`, которые будут рассмотрены ниже. Следует также отметить, что класс некорректных задач шире класса обратных (*один из примеров вы найдете в конце разд. 11.1.2*).

Одним из наиболее простых методов решения некорректных обратных задач является концепция поиска их *квазирешения*. Рассмотрим обратную задачу `A·Y=B`, где неизвестный вектор `Y` подлежит определению, а оператор `A` (в линейном случае, матрица) и вектор правых частей уравнений `B` известны. Подчеркнем, что задача может быть и нелинейной, т. е. оператор `A` может описывать сложную зависимость.

Примечание

Проблемы, возникающие при попытке непосредственного обращения матрицы `A`, связаны с ее плохой обусловленностью. Подход к подобным задачам может быть не обязательно таким, как рассказывается далее (*см. главу 8*).

Основная идея квазирешения состоит в параметризации неизвестного вектора `Y`, исходя из физических соображений постановки задачи. То есть на основе некоторой имеющейся априорной информации следует заранее задать модельный вид `Y~Y0`, зависящий от ряда параметров `r1,r2,...` В результате пространство поиска решений значительно сужается — вместо отыскания всех компонент вектора `Y` требуется лишь найти значения модельных параметров, решающих (в определенном смысле) задачу.

Квазирешение `Y0` находится из решения задачи на минимум:

$$\texttt{r=arg min\{|A·Y0(r)-B|\}}, \tag{6.4}$$

где минимизация проводится по вектору `s` параметров модельной зависимости `Y0(r)`. Следует подчеркнуть, что задача поиска квазирешения является задачей на глобальный экстремум, что важно с позиций выбора вычислительного метода (как уже отмечалось, наиболее популярны градиентные методы поиска минимума в комбинации со сканированием для достижения глобальной минимизации — алгоритмами сплошного или случайного поиска).

Пример отыскания квазирешения обсуждавшейся в предыдущих разделах задачи приведен в листинге 6.14 и на рис. 6.7 (данные о результатах модельных измерений `B` и матрице `A` взяты из предыдущего листинга). Понятно, что при сведении некорректной задачи к проблеме отыскания квазирешения решающее значение принадлежит правильно выбранной параметризации неизвестного вектора `Y`.

Листинг 6.14. Квазирешение некорректной задачи (продолжение листингов 6.12 и 6.13)

$Y0(x, c, k, \phi) := c \cdot \sin(k \cdot x + \phi)$

$f(c, k, \phi) := |A \cdot Y0(X, c, k, \phi) - B|$

$c := 1 \qquad k := 0.5 \qquad \phi := 0$

$r := \text{Minimize}(f, c, k, \phi)$

$$r = \begin{pmatrix} -0.981 \\ 0.996 \\ -1.556 \end{pmatrix}$$

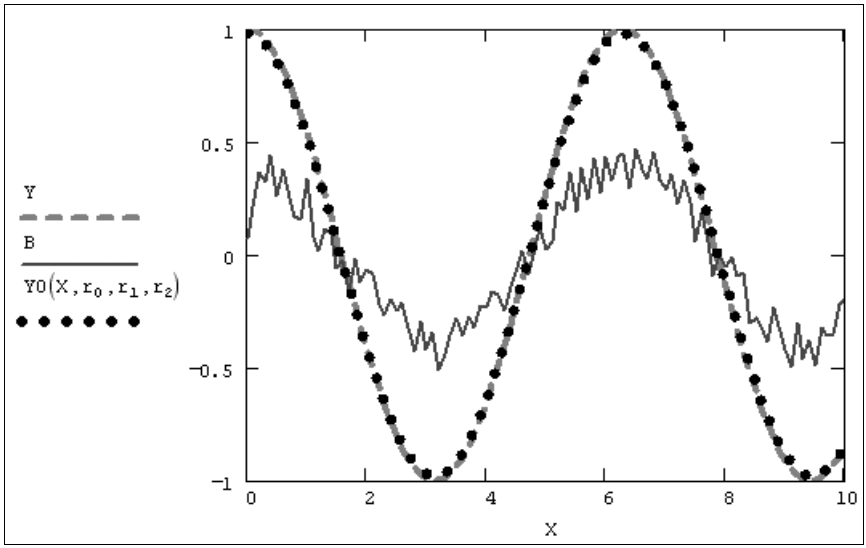


Рис. 6.7. Исходный сигнал Y, измерения B и квазирешение Y0 (продолжение листинга 6.14)

ГЛАВА 7

Линейная алгебра

Задачи линейной алгебры, решаемые в Mathcad, можно условно разделить на два класса. Первый — это простейшие матричные операции, которые сводятся к определенным арифметическим действиям над элементами матрицы. Они реализованы в виде операторов (*см. разд. 7.1 и 7.2*) и нескольких специфических функций, предназначенных для создания, объединения, сортировки, получения основных свойств матриц и т. п. (*см. разд. 7.4*). Второй класс — это более сложные действия, которые реализуют алгоритмы вычислительной линейной алгебры, такие как вычисление определителей и обращение матриц (*см. разд. 7.3*), вычисление собственных векторов и собственных значений, решение систем линейных алгебраических уравнений и различные матричные разложения (*см. главу 8*).

7.1. Простейшие матричные операции

Простейшие операции матричной алгебры реализованы в Mathcad в виде операторов, причем их запись максимально приближена к математическому значению. Каждый оператор выражается соответствующим символом. Некоторые операции применимы только к квадратным матрицам N×N, некоторые допускаются только для векторов (например, скалярное произведение), а другие, несмотря на одинаковое написание, по-разному действуют на векторы и матрицы.

7.1.1. Транспонирование

Транспонированием называют операцию, переводящую матрицу размерности M×N в матрицу размерности N×M, делая столбцы исходной матрицы строками, а строки — столбцами. Несколько примеров транспонирования приведены в листинге 7.1. Ввод символа транспонирования (transpose) осуществляется с помощью панели инструментов **Матрица** (Matrix) или нажатием клавиш <Ctrl>+<1>. Не забывайте, что для вставки символа транспонирования матрица должна находиться между линиями ввода.

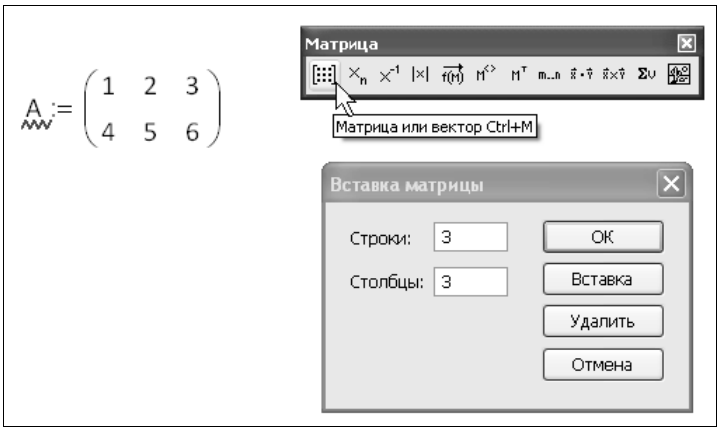


Рис. 7.1. Ввод матриц и основные операции над ними осуществляются при помощи панели **Матрица**

Примечание

Все матричные и векторные операторы, о которых пойдет речь, допустимо использовать как в численных, так и в символьных расчетах. Мощь символьных операций заключается в возможности проводить их не только над конкретными числами, но и над переменными. Смело используйте символьный процессор в качестве мощного математического справочника, когда вы хотите вспомнить какое-либо определение из области линейной алгебры. Именно по этой причине мы прибегаем к примерам символьных расчетов в большинстве иллюстраций к данному разделу.

Листинг 7.1. Транспонирование векторов и матриц

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

$$\begin{pmatrix} 1.01 & 2.02 & 3.03 & 4.04 \end{pmatrix}^T = \begin{pmatrix} 1.01 \\ 2.02 \\ 3.03 \\ 4.04 \end{pmatrix}$$

$$\begin{pmatrix} x & y & z \end{pmatrix}^T \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

7.1.2. Сложение и вычитание

В Mathcad можно как складывать матрицы, так и вычитать их друг из друга. Для этих операторов применяются стандартные символы "+" или "–" соответственно. Матрицы должны иметь одинаковую размерность, иначе будет выдано сообщение

об ошибке. Каждый элемент суммы двух матриц равен сумме соответствующих элементов матриц-слагаемых (листинг 7.2).

Результат унарной операции смены знака матрицы эквивалентен смене знака всех ее элементов. Для того чтобы изменить знак матрицы, достаточно ввести перед ней знак минуса, как перед обычным числом (нижняя строка листинга 7.2).

Листинг 7.2. Сложение, вычитание и смена знака матриц

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} + \begin{pmatrix} u & v & w \\ x & y & z \end{pmatrix} \rightarrow \begin{pmatrix} a+u & b+v & c+w \\ d+x & f+y & g+z \end{pmatrix}$$

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} - \begin{pmatrix} u & v & w \\ x & y & z \end{pmatrix} \rightarrow \begin{pmatrix} a-u & b-v & c-w \\ d-x & f-y & g-z \end{pmatrix}$$

$$-\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} -1 & -2 & -3 \\ -4 & -5 & -6 \end{pmatrix}$$

Кроме сложения матриц Mathcad поддерживает операцию сложения матрицы со скаляром (листинг 7.3). Каждый элемент результирующей матрицы равен сумме соответствующего элемента исходной матрицы и скалярной величины.

Листинг 7.3. Сложение матрицы со скалярной величиной

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} + k \rightarrow \begin{pmatrix} a+k & b+k & c+k \\ d+k & f+k & g+k \end{pmatrix}$$

Иногда бывает нужно вычислить сумму всех элементов вектора или матрицы. Для этого существует вспомогательный оператор (листинг 7.4, первая и вторая строки соответственно), задаваемый кнопкой **Сумма вектора** (Vector Sum) на панели **Матрица** (Matrix) или сочетанием клавиш <Ctrl>+<4>. Этот оператор чаще оказывается полезным не в матричной алгебре, а при организации циклов с индексированными переменными.

В том же листинге 7.4 (снизу) показано применение операции суммирования диагональных элементов квадратной матрицы. Эту сумму называют *следом* (trace) матрицы. Данная операция организована в виде встроенной функции `tr`:

- `tr(A)` — след квадратной матрицы `A`.

Листинг 7.4. Суммирование элементов вектора и вычисление следа матрицы

$$\sum \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \rightarrow a+b+c+d$$

$$\mathrm{tr}\left(\begin{pmatrix} a & b & c \\ d & f & g \\ x & y & z \end{pmatrix}\right) \rightarrow a + f + z$$

7.1.3. Умножение

При умножении следует помнить, что матрицу размерности M×N допустимо умножать только на матрицу размерности N×P (P может быть любым). В результате получается матрица размерности M×P. Чтобы ввести символ умножения, нужно нажать клавишу со звездочкой <*> Умножение матриц обозначается по умолчанию точкой, как показано в листинге 7.5.

Листинг 7.5. Перемножение матриц

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} \cdot \begin{pmatrix} u & x \\ v & y \\ w & z \end{pmatrix} \rightarrow \begin{pmatrix} a\cdot u + b\cdot v + c\cdot w & a\cdot x + b\cdot y + c\cdot z \\ d\cdot u + f\cdot v + g\cdot w & d\cdot x + f\cdot y + g\cdot z \end{pmatrix}$$

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} \cdot \begin{pmatrix} u & v & w \\ x & y & z \end{pmatrix} \rightarrow$$

Обратите внимание (нижняя строка листинга 7.5), что попытка перемножить матрицы A и B несоответствующего (одинакового 2×3) размера оказалась безрезультатной: после введенного знака равенства находится пустой местозаполнитель, а само выражение в редакторе Mathcad выделяется красным цветом. При установке курсора на это выражение появляется сообщение о несовпадении числа строк первой матрицы с числом столбцов второй матрицы.

Еще один пример, относящийся к умножению вектора на матрицу-строку и, наоборот, строки на вектор, приведен в листинге 7.6.

> ***Внимание!***
> Оператор умножения действует на два вектора по-другому (*см. разд. 7.2.2*).

Листинг 7.6. Умножение вектора и строки

$$\begin{pmatrix} a & b & c & d \end{pmatrix} \cdot \begin{pmatrix} s \\ x \\ y \\ z \end{pmatrix} \rightarrow a\cdot s + b\cdot x + c\cdot y + d\cdot z$$

$$\begin{pmatrix} s \\ x \\ y \\ z \end{pmatrix} \cdot \begin{pmatrix} a & b & c & d \end{pmatrix} \rightarrow \begin{pmatrix} a\cdot s & s\cdot b & s\cdot c & s\cdot d \\ a\cdot x & b\cdot x & x\cdot c & d\cdot x \\ y\cdot a & b\cdot y & c\cdot y & y\cdot d \\ z\cdot a & z\cdot b & c\cdot z & d\cdot z \end{pmatrix}$$

Аналогично сложению матриц со скаляром определяется умножение и деление матрицы на скалярную величину (листинг 7.7). Символ умножения вводится так же, как и в случае умножения двух матриц. На скаляр можно умножать матрицу любой размерности.

Листинг 7.7. Умножение матрицы на скалярную величину

$$\begin{pmatrix} a & b & c \\ d & f & g \end{pmatrix} \cdot k \rightarrow \begin{pmatrix} a \cdot k & b \cdot k & c \cdot k \\ d \cdot k & f \cdot k & g \cdot k \end{pmatrix}$$

7.2. Векторная алгебра

Векторы являются частным случаем матриц размерности N×1, поэтому для них справедливы все те операции, что и для матриц, если ограничения особо не оговорены. Вместе с тем для векторов в линейной алгебре предусмотрен целый ряд специфических операций, и все они реализованы в системе Mathcad.

> ***Внимание!***
>
> Непосредственное проведение многих векторных операций над матрицами-строками, т. е. матрицами 1×N, невозможно; для того чтобы превратить строку в вектор, ее нужно предварительно транспонировать.

7.2.1. Модуль вектора

Модуль вектора (vector magnitude) по определению равен квадратному корню из суммы квадратов его элементов (листинг 7.8).

> ***Внимание!***
>
> Не путайте модуль вектора и определитель матрицы, который обозначается тем же символом (*см. разд. 7.3.1*). Это характерный пример оператора, действующего по-разному на векторы и квадратные матрицы.

Листинг 7.8. Модуль вектора

$$\left|\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right| \rightarrow \left[|x|^2 + |y|^2 + |z|^2 \right]^{\frac{1}{2}}$$

$$\left|\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}\right| = 3.742$$

7.2.2. Скалярное произведение

Скалярное произведение векторов (vector inner product) определяется как скаляр, равный сумме попарных произведений соответствующих элементов. Векторы должны иметь одинаковую размерность, скалярное произведение имеет ту же размерность. Скалярное произведение двух векторов u и v равно $u \cdot v = |u| \cdot |v| \cdot \cos\theta$, где θ — угол между векторами. Если векторы ортогональны, то их скалярное произведение равно нулю. Обозначается скалярное произведение тем же символом, что и умножение (листинг 7.9).

> ***Внимание!***
> Для обозначения скалярного произведения пользователю позволяется выбирать представление оператора умножения при помощи контекстного меню (подобно умножению скалярных величин). Однако никогда не применяйте для обозначения скалярного произведения символ $\times$, который является общеупотребительным символом векторного произведения (*см. разд. 7.2.3*).

Листинг 7.9. Скалярное произведение векторов

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow a \cdot x + b \cdot y + c \cdot z$$

С осторожностью перемножайте несколько (более двух) векторов. По-разному расставленные скобки полностью изменяют результат умножения. Примеры такого умножения приведены в листинге 7.10.

Листинг 7.10. Скалярное произведение векторов, умноженное на третий вектор

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \cdot \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 224 \\ 256 \\ 288 \end{pmatrix}$$

$$\left[\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}\right] \cdot \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 224 \\ 256 \\ 288 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \cdot \left[\begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \cdot \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix}\right] = \begin{pmatrix} 122 \\ 244 \\ 366 \end{pmatrix}$$

7.2.3. Векторное произведение

Векторное произведение (cross product) двух векторов u и v с углом θ между ними равно вектору с модулем $|u| \cdot |v| \cdot \sin\theta$, направленным перпендикулярно плоскости

векторов u и v. Обозначают векторное произведение символом ×, который можно ввести нажатием кнопки **Векторное произведение** (Cross Product) в панели **Матрица** (Matrix) или сочетанием клавиш <Ctrl>+<8>. Пример приведен в листинге 7.11.

Листинг 7.11. Векторное произведение двух векторов

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} b \cdot z - c \cdot y \\ c \cdot x - a \cdot z \\ a \cdot y - b \cdot x \end{pmatrix}$$

7.2.4. Векторизация массива

Векторная алгебра Mathcad включает несколько необычный оператор, который называется *оператором векторизации* (vectorize operator). Этот оператор предназначен, как правило, для работы с массивами. Он позволяет провести однотипную операцию над всеми элементами массива (т. е. матрицы или вектора), упрощая тем самым программирование циклов. Например, иногда требуется умножить каждый элемент одного вектора на соответствующий элемент другого вектора, чтобы в результате также получился вектор. Непосредственно такой операции в Mathcad нет, но ее легко осуществить с помощью векторизации (листинг 7.12). Для этого:

1. Введите векторное выражение, как показано во второй строчке листинга 7.12 (обратите внимание, что в таком виде символ умножения обозначает оператор скалярного произведения векторов).
2. Переместите курсор таким образом, чтобы линии ввода выделяли все выражение, которое требуется подвергнуть векторизации.
3. Введите оператор векторизации, нажав кнопку **Векторизовать** (Vectorize) на панели **Матрица** (Matrix) (листинг 7.12) или сочетание клавиш <Ctrl>+<->.
4. Введите <=>, чтобы получить результат.

Листинг 7.12. Использование оператора векторизации для перемножения элементов вектора

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\overrightarrow{\left[v \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \right]} = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix} \qquad v \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = 14$$

14 *Примечание*

В Mathcad 14 и 15 векторизация может осуществляться не только в численных, но и в символьных (аналитических) расчетах.

Большинство неспецифических функций Mathcad не требуют векторизации для проведения одной и той же операции над всеми элементами вектора. Например, аргументом тригонометрических функций по определению является скаляр. Если попытаться вычислить экспоненту от векторной величины, Mathcad осуществит векторизацию по умолчанию, возведя в степень e каждый элемент и выдав в качестве результата соответствующий вектор (листинг 7.13).

Листинг 7.13. Векторизация аргумента необязательна для большинства встроенных функций Mathcad

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\exp(v) = \begin{pmatrix} 2.718 \\ 7.389 \\ 20.086 \end{pmatrix} \qquad \exp\left|\vec{v}\right| = \begin{pmatrix} 2.718 \\ 7.389 \\ 20.086 \end{pmatrix}$$

7.3. Вычисление определителей и обращение квадратных матриц

Рассмотрим еще несколько исключительно важных действий линейной алгебры, связанных с понятием определителя матрицы. Несмотря на то, что некоторые из них реализованы в Mathcad также в виде операторов, они требуют (при проведении расчетов по численным алгоритмам) несравненно больше внимания, нежели операторы, упомянутые в двух предыдущих разделах.

7.3.1. Определитель квадратной матрицы

Определитель (Determinant) матрицы обозначается стандартным математическим символом. Чтобы ввести оператор нахождения определителя матрицы, можно нажать кнопку **Определитель** (Determinant) на панели инструментов **Матрица** (Matrix) (листинг 7.14) или набрать на клавиатуре <|> (нажав клавиши <Shift>+<\>). В результате любого из этих действий появляется местозаполнитель, в который следует поместить матрицу. Чтобы вычислить определитель уже введенной матрицы:

1. Переместите курсор в документе таким образом, чтобы поместить матрицу между линиями ввода (напоминаем, что линии ввода — это вертикальный и горизонтальный отрезки синего цвета, образующие уголок, указывающий на текущую область редактирования).

2. Введите оператор нахождения определителя матрицы.
3. Введите знак равенства (либо символьного вывода), чтобы вычислить определитель (численно или аналитически соответственно, как это показано в листинге 7.14).

> **(12) *Внимание!***
>
> Не путайте операторы вычисления определителя квадратной матрицы и длины вектора. Начиная с Mathcad 12, введен принудительный контроль действий пользователя при вводе этих операторов во избежание путаницы (т. к. один и тот же символ используется для этих двух операций). При попытке вычислить определитель матрицы с помощью оператора $|A|$, введенного с панели **Калькулятор** (Calculator), а не **Матрица** (Matrix), будет выдано сообщение об ошибке, а результат вычисления детерминанта появится только после того, как пользователь вызовет контекстное меню и подтвердит в нем, что он собирается вычислить именно определитель матрицы. То же самое касается и длины вектора, если попытаться ввести его не с панели **Калькулятор** (Calculator), а с панели **Матрица** (Matrix).

Листинг 7.14. Вычисление определителя квадратной матрицы

$$\left|\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ 5 & 4 & 3.00001 \end{pmatrix}\right| = 16$$

$$\left|\begin{pmatrix} 1 & 0 & -1 \\ 0 & 2 & 0 \\ 5 & 4 & 3.00001 \end{pmatrix}\right| \rightarrow 16.00002$$

7.3.2. Ранг матрицы

Рангом (rank) матрицы называют наибольшее натуральное число `k`, для которого существует не равный нулю определитель `k`-го порядка подматрицы, составленной из любого пересечения `k` столбцов и `k` строк матрицы.

Для вычисления ранга в Mathcad предназначена функция `rank` (листинг 7.15).

- `rank(A)` — ранг матрицы:
  - `A` — матрица.

Листинг 7.15. Вычисление ранга матрицы

$$\left|\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right| = 0 \qquad \mathrm{rank}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right) = 2$$

$$\left|\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}\right| = 27 \qquad \mathrm{rank}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}\right) = 3$$

7.3.3. Обращение квадратной матрицы

Поиск *обратной матрицы* возможен, если матрица квадратная и ее определитель не равен нулю. Произведение исходной матрицы на обратную по определению является единичной матрицей. Для ввода оператора поиска обратной матрицы нажмите кнопку **Обращение** (Inverse) на панели инструментов **Матрица** (Matrix). В листинге 7.16 приведен пример поиска обратной матрицы и последующая проверка правильности ее вычисления.

Листинг 7.16. Вычисление обратной матрицы

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

7.3.4. Возведение квадратной матрицы в степень

К квадратным матрицам можно формально применять операцию возведения в степень `n`. Для этого `n` должно быть целым числом. Результат данной операции приведен в табл. 7.1. Ввести оператор возведения матрицы `M` в степень `n` можно точно так же, как и для скалярной величины: нажав кнопку **Возведение в степень** (Raise to Power) на панели **Калькулятор** (Calculator) или нажав клавишу <^>. После появления местозаполнителя в него следует ввести значение степени `n`.

Таблица 7.1. Правила возведения матрицы в степень

n	$\mathbf{M^n}$
0	Единичная матрица размерности матрицы `M`
1	Сама матрица `M`
−1	M^{-1} — матрица, обратная `M`

Таблица 7.1 *(окончание)*

n	M^n
2, 3, ...	M·M, (M·M)·M, ...
−2, −3, ...	$M^{-1}\cdot M^{-1}$, $(M^{-1}\cdot M^{-1})\cdot M^{-1}$, ...

Примеры возведения матрицы в степень приведены в листинге 7.17.

Листинг 7.17. Возведение квадратной матрицы в целую степень

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}^{-2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.063 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}^{0} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}^{1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}^{2} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 16 \end{pmatrix}$$

7.3.5. Матричные нормы

В линейной алгебре используются различные векторные и матричные *нормы* (norm), которые ставят в соответствие матрице некоторую скалярную числовую характеристику. Норма матрицы отражает порядок величины матричных элементов. В разных специфических задачах линейной алгебры применяются различные виды норм. Mathcad имеет четыре встроенных функции для расчета разных норм квадратных матриц:

- `norm1(A)` — норма в пространстве L1;
- `norm2(A)` — норма в пространстве L2;

- `norme(A)` — евклидова норма (euclidean norm);
- `normi(A)` — max-норма, или ∞-норма (infinity norm):
  - `A` — квадратная матрица.

Примеры расчета различных норм двух матриц `A` и `B` с различающимися на два порядка элементами иллюстрирует листинг 7.18.

> ***Совет***
>
> В большинстве задач неважно, какую норму использовать. Как видно, в обычных случаях разные нормы дают примерно одинаковые значения, хорошо отражая порядок величины матричных элементов. Конкретные формулы, определяющие нормы, заинтересованный читатель отыщет в справочниках по линейной алгебре или в Ресурсах Mathcad.

Листинг 7.18. Вычисление различных норм матриц

$$A := \begin{pmatrix} 1 & 2 & 3 \\ -4 & 5 & -6 \\ -7 & 8 & 9 \end{pmatrix} \qquad B := \begin{pmatrix} 100 & 200 & 300 \\ -400 & 500 & -600 \\ -700 & 800 & 900 \end{pmatrix}$$

$$\mathrm{norm1}(A) = 18 \qquad \mathrm{norm1}(B) = 1.8 \times 10^3$$

$$\mathrm{norm2}(A) = 14.213 \qquad \mathrm{norm2}(B) = 1.421 \times 10^3$$

$$\mathrm{normi}(A) = 24 \qquad \mathrm{normi}(B) = 2.4 \times 10^3$$

$$\mathrm{norme}(A) = 16.882 \qquad \mathrm{norme}(B) = 1.688 \times 10^3$$

$$\max(A) = 9 \qquad \max(B) = 900$$

7.3.6. Число обусловленности квадратной матрицы

Еще одной важной характеристикой матрицы является ее *число обусловленности* (condition number). Число обусловленности является мерой чувствительности системы линейных уравнений `A·x=b`, определяемой матрицей `A`, к погрешностям задания вектора `b` правых частей уравнений (*см. главу 8*). Чем больше число обусловленности, тем сильнее это воздействие и тем более неустойчив процесс нахождения решения линейной системы. Число обусловленности связано с нормой матрицы и вычисляется по-разному для каждой из норм:

- `cond1(A)` — число обусловленности в норме L1;
- `cond2(A)` — число обусловленности в норме L2;
- `conde(A)` — число обусловленности в евклидовой норме;
- `condi(A)` — число обусловленности в ∞-норме:
  - `A` — квадратная матрица.

Расчет чисел обусловленности для двух матриц `A` и `B` показан в листинге 7.19. Обратите внимание, что первая из матриц является *хорошо обусловленной*, а вторая — *плохо обусловленной* (два ее последних столбца очень близки между собой, с точ-

ностью до множителя 2). Вторая строка листинга дает формальное определение числа обусловленности как произведения норм исходной и обратной матриц. В других нормах определение точно такое же.

> ***Примечание***
> Как нетрудно понять, матрицы A и B из листинга 7.18 (*см. предыдущий раздел*) обладают одинаковыми числами обусловленности, т. к. B=100·A, и, следовательно, обе матрицы определяют одну и ту же систему уравнений.

Листинг 7.19. Вычисление чисел обусловленности матриц (в различных нормах)

$$A := \begin{pmatrix} 1 & 2 & 3 \\ -4 & 5 & -6 \\ -7 & 8 & 9 \end{pmatrix} \qquad B := \begin{pmatrix} 1 & 2 & 4 \\ -4 & 5 & 10 \\ -7 & 8 & 16.1 \end{pmatrix}$$

$$\mathrm{cond1}(A) = 12.14 \qquad \mathrm{cond1}(B) = 1.535 \times 10^3$$

$$\mathrm{cond2}(A) = 6.951 \qquad \mathrm{cond2}(B) = 1.017 \times 10^3$$

$$\mathrm{condi}(A) = 11.721 \qquad \mathrm{condi}(B) = 1.811 \times 10^3$$

$$\mathrm{conde}(A) = 8.556 \qquad \mathrm{conde}(B) = 1.024 \times 10^3$$

7.4. Вспомогательные матричные функции

Перечислим основные встроенные функции, предназначенные для облегчения работы с векторами и матрицами. Они нужны для создания матриц, слияния и выделения части матриц, получения основных свойств матриц и т. п.

7.4.1. Автоматическая генерация матриц

Самым наглядным способом создания матрицы или вектора является применение первой кнопки панели инструментов **Матрица** (Matrix). Однако в большинстве случаев, в частности при программировании сложных проектов, удобнее бывает создавать массивы с помощью встроенных функций.

Создание матриц на основе некоторой функции

Наиболее удобный прием автоматизации создания матриц заключается в предварительном определении функции f(i,j), аргументом которой должны быть индексы элементов матрицы:

- ❒ matrix(M,N,f) — создание матрицы размера M×N, каждый i,j элемент которой есть f(i,j) (листинг 7.20):
  - M — количество строк матрицы;
  - N — количество столбцов матрицы;
  - f(i,j) — функция.

Листинг 7.20. Создание матрицы на основе функции пользователя

$$f(i,j) := i^2 + j^2$$

$$\text{matrix}(3,4,f) = \begin{pmatrix} 0 & 1 & 4 & 9 \\ 1 & 2 & 5 & 10 \\ 4 & 5 & 8 & 13 \end{pmatrix}$$

Создание матриц для построения 3D-графиков

Для создания матриц имеются еще две специфические функции, применяемые, в основном, для быстрого и эффектного представления каких-либо зависимостей в виде трехмерных графиков (типа поверхности или пространственной кривой). Все их аргументы, кроме первого (имени функции), необязательны. Рассмотрим первую из этих встроенных функций.

- `CreateSpace(F(или f1,f2,f3),t0,t1,tgrid,fmap)` — создание вложенного массива, представляющего `x`-, `y`- и `z`-координаты параметрической пространственной кривой, заданной функцией `F`:
  - `F(t)` — векторная функция из трех элементов, заданная параметрически относительно единственного аргумента `t`;
  - `f1(t),f2(t),f3(t)` — скалярные функции;
  - `t0` — нижний предел `t` (по умолчанию `-5`);
  - `t1` — верхний предел `t` (по умолчанию `5`);
  - `tgrid` — число точек сетки по переменной `t` (по умолчанию `20`);
  - `fmap` — векторная функция от трех аргументов, задающая преобразование координат.

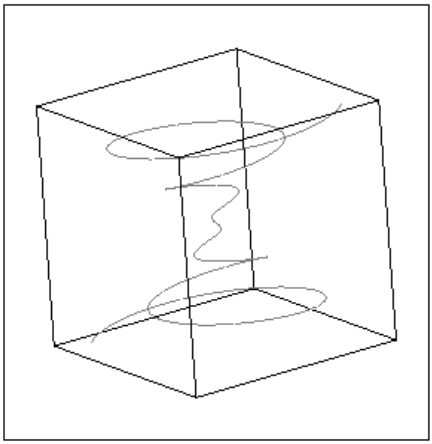
Пример использования функции `CreateSpace` показан на рис. 7.2. Заметьте, для построения графика кривой не потребовалось никакого дополнительного кода, кроме определения параметрической зависимости в вектор-функции `F`!

Функция создания матрицы для графика трехмерной поверхности устроена совершенно аналогично, за тем исключением, что для определения поверхности требуется не одна, а две переменных. Пример ее использования иллюстрирует рис. 7.3.

- `CreateMesh(F(или g, или f1,f2,f3),s0,s1,t0,t1,sgrid,tgrid,fmap)` — создание вложенного массива, представляющего `x`-, `y`- и `z`-координаты параметрической поверхности, заданной функцией `F`:
  - `F(s,t)` — векторная функция из трех элементов, заданная параметрически относительно двух аргументов `s` и `t`;
  - `g(s,t)` — скалярная функция;
  - `f1(s,t),f2(s,t),f3(s,t)` — скалярные функции;
  - `s0,t0` — нижние пределы аргументов `s,t` (по умолчанию `-5`);

$$f(t) := \begin{pmatrix} t \cdot \cos(t) \\ t^2 \cdot \sin(t) \\ t \end{pmatrix}$$

F := CreateSpace(f, -15, 15, 1000)



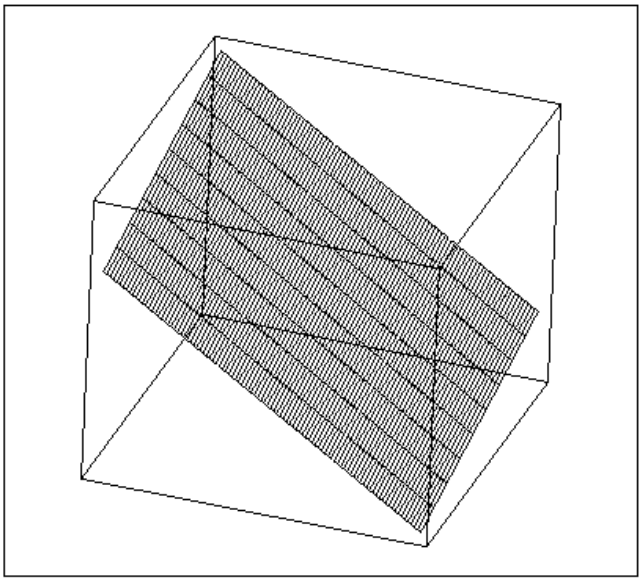
F

F = ((3,1))

Рис. 7.2. Использование функции `CreateSpace` для построения графика трехмерной кривой

f(x, y) := 1 − x − 3·y

F := CreateMesh(f, 10, 100)



F

Рис. 7.3. Использование функции `CreateMesh` для построения графика трехмерной поверхности

- `s1,t1` — верхние пределы аргументов `s,t` (по умолчанию `5`);
- `sgrid,tgrid` — число точек сетки по переменным `s` и `t` (по умолчанию `20`);
- `fmap` — векторная функция из трех элементов от трех аргументов, задающая преобразование координат.

Результатом обеих рассмотренных функций, `CreateMesh` и `CreateSpace`, является соответствующий вложенный массив, служащий в Mathcad для представления тензора. Каждая матрица из числа трех вложенных матриц, образующих вложенный массив данных, определяет `x`-, `y`- и `z`-координаты точек поверхности или кривой.

Создание диагональных матриц

В Mathcad легко создать матрицы, имеющие определенное простое строение, с помощью одной из встроенных функций. Примеры использования этих функций приведены в листинге 7.21:

- ❒ `identity(N)` — единичная матрица размера `N×N`;
- ❒ `diag(v)` — диагональная матрица, на диагонали которой находятся элементы вектора `v`:
  - `N` — целое число;
  - `v` — вектор.

Листинг 7.21. Создание единичной и диагональной матрицы заданной размерности

$$\mathrm{identity}(3) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \qquad \mathrm{diag}(v) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

7.4.2. Разбиение и слияние матриц

Из матрицы или вектора можно выделить либо подматрицу, либо вектор-столбец, либо отдельный элемент. И обратно, можно "склеить" несколько матриц в одну.

Выделение подматрицы

Часть матрицы выделяется одним из следующих способов (листинг 7.22):

- ❒ для выделения одного элемента предназначен оператор нижнего индекса. Оператор вводится нажатием кнопки **Индекс** (Subscript) со значком $\mathbf{x_n}$ на панели **Матрица** (Matrix) либо нажатием клавиши <[> (вторая строка листинга 7.22);

- для выделения из матрицы столбца примените оператор выделения столбца нажатием кнопки **Столбец матрицы** (Matrix Column), с изображением угловых скобок <> на панели **Матрица** (Matrix), либо сочетанием клавиш <Ctrl>+<6> (третья строка листинга 7.22). Этот оператор называют еще, по аналогии с предыдущим, оператором *верхнего индекса*;
- чтобы выделить из матрицы строку, примените тот же оператор <> к транспонированной матрице (конец листинга 7.22);
- для выделения подматрицы используйте встроенную функцию `submatrix(A,ir,jr,ic,jc)`, возвращающую часть матрицы `A`, находящуюся между строками `ir,jr` и столбцами `ic,jc` включительно (листинг 7.23).

Листинг 7.22. Доступ к отдельным элементам, столбцам и строкам матрицы

$$A := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

$A_{1,0} = 5$ $\qquad$ $A_{2,1} = 10$

$$A^{\langle 0 \rangle} = \begin{pmatrix} 1 \\ 5 \\ 9 \end{pmatrix} \qquad A^{\langle 3 \rangle} = \begin{pmatrix} 4 \\ 8 \\ 12 \end{pmatrix}$$

$$\left| A^T \right|^{\langle 0 \rangle^T} = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}$$

$$\left| A^T \right|^{\langle 1 \rangle^T} = \begin{pmatrix} 5 & 6 & 7 & 8 \end{pmatrix}$$

Примечание 1

Выделить из матрицы один столбец или строку можно и с помощью функции `submatrix` (листинг 7.23, нижняя строка).

Примечание 2

Те же операции применимы к матрицам-векторам и матрицам-строкам. Следует помнить только, что размер их составляет `N×1` и `1×N` соответственно.

Листинг 7.23. Выделение подматрицы

$$A := \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

$$\mathrm{submatrix}(A, 0, 1, 0, 2) = \begin{pmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \end{pmatrix}$$

$$\mathrm{submatrix}(A, 0, 0, 0, 2) = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

Слияние матриц

Для того чтобы составить из двух или более матриц одну, в Mathcad предусмотрена пара матричных функций (листинг 7.24):

- `augment(A,B,C,...)` — матрица, сформированная слиянием матриц-аргументов слева направо;
- `stack(A,B,C,...)` — матрица, сформированная слиянием матриц-аргументов сверху вниз:
  - `A,B,C,...` — векторы или матрицы соответствующего размера.

Листинг 7.24. Примеры слияния матриц

$$A := \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \qquad B := \begin{pmatrix} 4 & 4 \\ 4 & 4 \end{pmatrix}$$

$$\mathrm{augment}(A, B) = \begin{pmatrix} 2 & 2 & 4 & 4 \\ 2 & 2 & 4 & 4 \end{pmatrix}$$

$$\mathrm{stack}(A, B) = \begin{pmatrix} 2 & 2 \\ 2 & 2 \\ 4 & 4 \\ 4 & 4 \end{pmatrix}$$

$$\mathrm{augment}(A, B, A, B) = \begin{pmatrix} 2 & 2 & 4 & 4 & 2 & 2 & 4 & 4 \\ 2 & 2 & 4 & 4 & 2 & 2 & 4 & 4 \end{pmatrix}$$

Специфические преобразования матриц

Еще две встроенных функции Mathcad позволяют создавать матрицы на основе некоторой имеющейся матрицы (листинг 7.25):

- `geninv(A)` — создание матрицы, обратной (слева) прямоугольной матрице `A`;
- `rref(A)` — преобразование матрицы или вектора `A` в ступенчатый вид:
  - `A` — матрица, составленная из действительных чисел.

Примечание

Размер `N×M` матрицы `A` для функции `geninv` должен быть таким, чтобы `N≥M`.

Листинг 7.25. Создание матриц на основе другой матрицы

$$A := \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix}$$

$$\text{geninv}(A) = \begin{pmatrix} -0.55 & -0.225 & 0.1 & 0.425 \\ 0.25 & 0.125 & 0 & -0.125 \end{pmatrix}$$

$$\text{geninv}(A) \cdot A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\text{rref}(A) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad \text{rref}(\text{geninv}(A)) = \begin{pmatrix} 1 & 0 & -1 & -2 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

7.4.3. Сортировка элементов матриц

Часто бывает нужно переставить элементы матрицы или вектора, расположив их в определенной строке или столбце в порядке возрастания или убывания. Для этого имеются несколько встроенных функций, которые позволяют гибко управлять сортировкой матриц:

- `sort(v)` — сортировка элементов вектора в порядке возрастания (листинг 7.26, верхняя строка);
- `reverse(v)` — перестановка элементов вектора в обратном порядке (листинг 7.26, нижняя строка);
- `csort(A,i)` — сортировка строк матрицы выстраиванием элементов `i`-го столбца в порядке возрастания (листинг 7.27, верхняя строка);
- `rsort(A,i)` — сортировка столбцов матрицы выстраиванием элементов `i`-й строки в порядке возрастания (листинг 7.27, нижняя строка):
  - `v` — вектор;
  - `A` — матрица;
  - `i` — индекс строки или столбца.

Примечание

Если элементы матриц или векторов комплексные, то сортировка ведется по действительной части, а мнимая часть игнорируется.

Листинг 7.26. Сортировка вектора

$$v := \begin{pmatrix} 3 \\ 1 \\ 4 \\ 2 \end{pmatrix} \qquad \text{sort}(v) = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$$\mathtt{reverse(v)} = \begin{pmatrix} 2 \\ 4 \\ 1 \\ 3 \end{pmatrix} \qquad \mathtt{reverse\ (sort\ (v))} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix}$$

Листинг 7.27. Сортировка матриц по строке и столбцу

$$\mathtt{A} := \begin{pmatrix} 1 & 7 \\ 3 & 6 \\ 4 & 5 \\ 2 & 8 \end{pmatrix}$$

$$\mathtt{csort(A,0)} = \begin{pmatrix} 1 & 7 \\ 2 & 8 \\ 3 & 6 \\ 4 & 5 \end{pmatrix} \qquad \mathtt{csort(A,1)} = \begin{pmatrix} 4 & 5 \\ 3 & 6 \\ 1 & 7 \\ 2 & 8 \end{pmatrix}$$

$$\mathtt{rsort(A,0)} = \begin{pmatrix} 1 & 7 \\ 3 & 6 \\ 4 & 5 \\ 2 & 8 \end{pmatrix} \qquad \mathtt{rsort(A,1)} = \begin{pmatrix} 1 & 7 \\ 3 & 6 \\ 4 & 5 \\ 2 & 8 \end{pmatrix}$$

7.4.4. Вывод размера матрицы

Для получения сведений о характеристиках матриц или векторов предусмотрены следующие встроенные функции (листинги 7.28 и 7.29 соответственно):

- `rows(A)` — число строк;
- `cols(A)` — число столбцов;
- `length(v)` — число элементов вектора;
- `last(v)` — индекс последнего элемента вектора:
  - `A` — матрица или вектор;
  - `v` — вектор.

Примечание

Если матричные индексы нумеруются с `1`, т. е. системная константа `ORIGIN` равна не `0` (по умолчанию), а `1`, то число элементов вектора и индекс его последнего элемента совпадают.

Листинг 7.28. Размер матриц

$$A := \begin{pmatrix} 1 & 7 \\ 3 & 6 \\ 4 & 5 \\ 2 & 8 \end{pmatrix}$$

$\text{rows}(A) = 4$ $\qquad$ $\text{cols}(A) = 2$

Листинг 7.29. Размер векторов

$$v := \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}$$

$w := (1 \quad 2 \quad 3 \quad 4)$

$\text{last}(v) = 3$ $\qquad$ $\text{last}\left(w^T\right) = 3$

$\text{length}(v) = 4$ $\qquad$ $\text{length}\left(w^T\right) = 4$

$\text{rows}(v) = 4$ $\qquad$ $\text{rows}(w) = 1$

$\text{cols}(v) = 1$ $\qquad$ $\text{cols}(w) = 4$

ГЛАВА 8

Системы линейных уравнений

Одной из центральных проблем вычислительной линейной алгебры является решение систем линейных уравнений (*см. разд. 8.1 и 8.2*), отыскание собственных векторов и собственных значений (*см. разд. 8.4*), а также различные матричные разложения (*см. разд. 8.3*). Все они будут рассмотрены в данной главе, являющейся, фактически, продолжением предыдущей (которая была посвящена простейшим матричным вычислениям).

> ***Примечание***
>
> К системам линейных уравнений сводится множество, если не сказать большинство, задач вычислительной математики. Один из таких примеров подробно рассмотрен в *разд. 10.4.*

Задачу решения *систем линейных алгебраических уравнений* (СЛАУ), т. е. систем N уравнений вида

$$a_{i1}\cdot x_1+a_{i2}\cdot x_2+\ldots+a_{iN}\cdot x_N=b_i \tag{8.1}$$

можно записать в эквивалентной матричной форме:

$$A\cdot x=b, \tag{8.2}$$

где A — матрица коэффициентов СЛАУ размерности $N\times N$; x — вектор неизвестных; b — вектор правых частей уравнений. Из общего курса линейной алгебры известно, что такая СЛАУ имеет единственное решение, если матрица A является *невырожденной* или, по-другому, *несингулярной*, т. е. ее определитель не равен нулю. Самый простой способ решения почти всякой несингулярной системы — использование алгоритма Гаусса, реализованного во встроенной функции `lsolve` (*см. разд. 8.1.2*).

> ***Примечание***
>
> На практике, при решении *обратных задач* (inverse problems), часто приходится сталкиваться не только со СЛАУ с квадратной матрицей A, но и с прямоугольной матрицей размера $M\times N$, т. е. системами, в которых число неизвестных не равно числу уравнений (как больше, так и меньше него). Такие системы требуют специфического подхода, который будет рассмотрен в *разд. 8.2 и 8.3*.

8.1. Хорошо обусловленные системы с квадратной матрицей

С вычислительной точки зрения, решение СЛАУ с квадратной матрицей `A` не представляет трудностей, если размерность `A` не очень велика. С большой матрицей проблем также не возникает, если она не очень плохо обусловлена (конечно, надо учитывать, что объем вычислений возрастает с увеличением размерности матрицы). В данном разделе будут рассмотрены именно такие системы, решение которых реализовано в Mathcad в двух вариантах:

- вычислительный блок `Given/Find` (приближенный итерационный алгоритм);
- встроенная функция `lsolve` (точный алгоритм Гаусса).

При этом СЛАУ можно решать как в более наглядной форме (8.1), так и в более компактной матричной форме (8.2). Для первого способа следует использовать блок `Given/Find` (он может также применяться в случае систем уравнений и неравенств, а также при решении переопределенных СЛАУ), а для второго — как вычислительный блок, так и встроенную функцию `lsolve`.

8.1.1. Вычислительный блок *Given / Find*

Для того чтобы численным методом решить СЛАУ при помощи вычислительного блока (он был подробно описан в *главе 5*), следует после ключевого слова `Given` выписать ее, пользуясь логическими операторами. Рассмотрим в качестве примера систему трех уравнений, приведенную в листинге 8.1 (после ключевого слова `Given`). Неизвестными являются три компонента вектора `x`, поэтому именно эта векторная переменная является аргументом встроенной функции `Find(x)`, решающей систему в последней строке листинга. Очень важно, что при использовании вычислительного блока `Given/Find` всем неизвестным требуется присвоить начальные значения, как это сделано в первой строке листинга 8.1.

Примечание

Если матрица СЛАУ является невырожденной (точнее, если ее число обусловленности не слишком велико), то известно, что численное решение системы уравнений единственно. Поэтому начальные значения могут быть произвольными, т. к. результат работы численного метода все равно сойдется к точному решению.

Листинг 8.1. Решение СЛАУ с помощью вычислительного блока

$x_0 := 0 \quad x_1 := 0 \quad x_2 := 0$

Given

$1 \cdot x_0 + 2 \cdot x_1 - 3 \cdot x_2 = 10$

$4 \cdot x_0 + 5 \cdot x_1 + 6 \cdot x_2 = 20$

$7 \cdot x_0 - 8 \cdot x_1 - 9 \cdot x_2 = 30$

$$\text{Find}(x) = \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix}$$

Листинг 8.1 демонстрирует запись каждого уравнения системы (в промежутке между `Given` и `Find`), что очень неудобно, когда система содержит большое число уравнений. В последнем случае намного лучше применить матричную запись СЛАУ, как это показано в листинге 8.2. Первая строка листинга представляет собой определение матрицы СЛАУ `A` и вектора правых частей `b`, а в остальном работа блока `Given/Find` полностью идентична предыдущему листингу.

Листинг 8.2. Решение СЛАУ, записанной в матричной форме

$$A := \begin{pmatrix} 1 & 2 & -3 \\ 4 & 5 & 6 \\ 7 & -8 & -9 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Given

$A \cdot x = b$

$$\text{Find}(x) = \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix}$$

Проверка правильности решения СЛАУ прямой подстановкой, причем в матричной форме, приведена в листинге 8.3. Обратите внимание на матрицу в первой строке листинга, представляющую рассматриваемую систему уравнений. Во второй строке листинга 8.3 производится вычисление нормы невязки, характеризующей точность полученного решения СЛАУ.

Примечание

Такая большая невязка может вызвать совершенно обоснованное удивление читателя. На самом деле точность решения гораздо выше (в данном примере $\sim 10^{-15}$), а полученное значение невязки $\sim 10^{-3}$ объясняется соответствующим представлением вещественных чисел результата на экране (по умолчанию с точностью до 3-го знака).

Листинг 8.3. Проверка правильности решения СЛАУ

$$\begin{pmatrix} 1 & 2 & -3 \\ 4 & 5 & 6 \\ 7 & -8 & -9 \end{pmatrix} \cdot \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix} = \begin{pmatrix} 10 \\ 19.999 \\ 30.006 \end{pmatrix}$$

$$\left\| \begin{pmatrix} 10 \\ 19.999 \\ 30.006 \end{pmatrix} - \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \right\| = 6.083 \times 10^{-3}$$

8.1.2. Функция *lsolve*

Альтернативой способу решения СЛАУ, введенному в предыдущем разделе, является применение встроенной функции `lsolve`. Для этого система уравнений должна быть записана в матричной форме `A·x=b`:

- `lsolve(A,b)` — вектор решения системы линейных алгебраических уравнений:
  - `A` — матрица коэффициентов системы;
  - `b` — вектор правых частей.

13 *Примечание 1*

Начиная с версии Mathcad 13, встроенная функция `lsolve` может использоваться не только для решения линейных систем с квадратной матрицей, но также и систем с прямоугольной матрицей, т. е. как недоопределенных, так и переопределенных систем уравнений (*см. разд. 8.2*).

i *Примечание 2*

В функции `lsolve` запрограммирован численный метод LU-разложения (*см. разд. 8.3.3*), основанный на *алгоритме последовательных исключений Гаусса*. Он состоит в преобразовании матрицы `A` линейной системы к треугольному виду, т. е. к форме, когда все элементы ниже главной диагонали матрицы являются нулевыми (*см. разд. 8.3.1*). Точнее, исходная СЛАУ `Ax=b` заменяется эквивалентной системой с другой матрицей A^* и другим вектором правых частей b^*, но имеющей то же решение, что и исходная система. Очень важно заметить, что результат, выдаваемый методом Гаусса, является *точным* (конечно, с поправкой на неизбежно присутствующие ошибки численного округления, которые, в случае хорошо обусловленной матрицы `A`, являются ничтожными). Таким образом, в противоположность применению вычислительного блока `Given/Find` (в основе которого лежит приближенный итерационный алгоритм), функция `lsolve` не нуждается в присвоении начальных значений вектору `x`.

Применение функции `lsolve` показано в листинге 8.4. При этом матрица `A` может быть определена любым из способов, необязательно явно, как во всех примерах этого раздела. В последней строке листинга осуществляется вычисление нормы невязки, которая оказывается равной нулю. Заметим, что встроенную функцию `lsolve` допускается применять и при символьном решении СЛАУ (листинг 8.5). В последнем случае в уравнениях допускается использовать параметры (т. е. имена переменных, которым не присвоены никакие значения).

Листинг 8.4. Численное решение СЛАУ

$$A := \begin{pmatrix} 1 & 2 & -3 \\ 4 & 5 & 6 \\ 7 & -8 & -9 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix}$$

$$\mathrm{lsolve}(A,b) = \begin{pmatrix} 4.667 \\ 1.333 \\ -0.889 \end{pmatrix}$$

$$\left|A \cdot \mathrm{lsolve}(A,b) - b\right| = 0$$

Листинг 8.5. Символьное решение СЛАУ (продолжение листинга 8.4)

$$\mathrm{lsolve}(A,b) \rightarrow \begin{pmatrix} \frac{14}{3} \\ \frac{4}{3} \\ \frac{-8}{9} \end{pmatrix}$$

8.2. Произвольные системы линейных уравнений

Классические задачи решения СЛАУ, рассмотренные в предыдущем разделе, предполагают равное количество уравнений и неизвестных, т. е. квадратную матрицу A. Именно для таких систем доказано, что решение существует и единственно, если определитель матрицы $|A| \neq 0$. Рассмотрим теперь задачи, в которых матрица A не является квадратной либо плохо обусловлена. Разберемся в этом разделе лишь с основными свойствами и идеологией подхода к таким системам, имея в виду, что наиболее эффективными способами их решения являются матричные разложения (*см. разд. 8.3*).

8.2.1. Переопределенные системы

Начнем разговор о "плохих" СЛАУ с переопределенными условиями, в которых число уравнений больше числа неизвестных.

О постановке задач

Чаще всего СЛАУ с прямоугольной матрицей размера M×N (при M>N, т. е. при числе уравнений большем числа неизвестных) вовсе не имеет решения, т. е. является *несовместной*, или *переопределенной*. Листинг 8.6 демонстрирует, что несовместные системы не могут быть решены посредством вычислительного блока Given/Find.

13 *Внимание!*

При помощи встроенной функции lsolve (вторая строка листинга) можно решать системы с прямоугольной матрицей, только начиная с версии Mathcad 13. В более ранних версиях попытка применить ее для расчета решения таких СЛАУ закончится неудачей, т. е. вместо результата во второй строке листинга будет выведено сообщение об ошибке.

Листинг 8.6. Попытка решения несовместных СЛАУ

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 51 \\ 109 \\ 172 \end{pmatrix}$$

$$\mathrm{lsolve}(A, b) = \begin{pmatrix} 10.333 \\ 19.917 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Given

$A \cdot x = b$

Find (x) = ■

Конечно, в редких случаях система с прямоугольной матрицей может оказаться совместной (если выбран соответствующий вектор b), как это показано в листинге 8.7.

Листинг 8.7. Пример нахождения решения СЛАУ с прямоугольной матрицей

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 50 \\ 110 \\ 170 \end{pmatrix}$$

$$\mathrm{lsolve}(A, b) = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Given $\quad A \cdot x = b$

$$\mathrm{Find}(x) = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 20 \end{pmatrix} = \begin{pmatrix} 50 \\ 110 \\ 170 \end{pmatrix}$$

На практике (особенно в последнее время) задачи отыскания решения переопределенных СЛАУ встречаются довольно часто. Начиная с версии Mathcad 13, их решение не представляет труда, т. к. оно унифицировано с решением "хороших" СЛАУ. Тем не менее, очевидно, что в большинстве случаев "настоящего" решения (такого, которое бы обращало уравнения в тождества) не существует (а в случае недоопределенных уравнений "настоящих" решений, наоборот, бесконечно много). Поэтому посвятим оставшуюся часть *разд. 8.2*, главным образом, пояснению во-

просов математической постановки задач на решение СЛАУ с прямоугольной матрицей.

> **13 *Примечание***
>
> Пользователи Mathcad 13—15 могут расценивать содержание *разд. 8.2* как объяснение алгоритмов работы функции `lsolve`, в то время как пользователи прежних версий найдут в нем рецепты решения недоопределенных и переопределенных СЛАУ без применения `lsolve`.

Псевдорешение (метод наименьших квадратов)

Напрашивающийся сам собой путь решения нашего простого примера является хорошей иллюстрацией подхода к общей проблеме переопределенных СЛАУ. А именно, вместо точного решения системы уравнений следует организовать поиск такого вектора `x`, который будет наилучшим образом удовлетворять всем уравнениям, т. е. минимизировать их *невязку* (расхождение между вектором `A·x` и вектором правой части СЛАУ `b`). Поскольку невязка `A·x-b` является векторной величиной, то, исходя из практических соображений, минимизации надо подвергать ее норму (т. е. скаляр) `|A·x-b|`. Этот подход позволит, с одной стороны, получить разумное, с физической точки зрения, решение задачи, а с другой — использовать полезную информацию, заключенную во всех уравнениях.

Таким образом, при интерпретации переопределенных СЛАУ принято искать не точное решение (которого, как уже отмечалось, при данной постановке задачи просто нет), а *псевдорешение* — вектор, минимизирующий норму невязки системы уравнений. Таким образом, задача решения линейной системы уравнений заменяется задачей отыскания глобального минимума функции `f(x)=|A·x-b|`. Повторимся еще раз, что, в полном соответствии с обозначениями, принятыми в Mathcad, `x` — это неизвестный вектор, а символ модуля (две вертикальные черты) означает операцию вычисления нормы (евклидовой длины вектора). Поскольку эта минимизируемая норма зависит от суммы квадратов компонент неизвестного вектора, то процедура поиска псевдорешения является не чем иным, как реализацией *метода наименьших квадратов* (МНК).

График функции двух переменных $f(x_0, x_1)$ показан в виде трехмерной поверхности на рис. 8.1, а несколько его сечений в окрестности минимума — на рис. 8.2. Нетрудно сообразить, почему структура функции оказывается именно такой, если вспомнить, что $f(x)^2$ является квадратичной формой относительно неизвестных x_0 и x_1. Надо отметить, что если бы наша система имела большее число уравнений, то вычислительная задача соответствующим образом усложнилась бы, т. к. минимизацию следовало бы проводить не по двум, а по большему числу переменных.

Обращаясь к практической стороне дела, следует вспомнить, что для решения задач минимизации невязки системы уравнений (*см. главу 6*) в Mathcad предусмотрены две встроенные функции — `Minerr` и `Minimize`. Применение этих альтернативных средств иллюстрируется листингами 8.8 и 8.9 соответственно. В первом случае используется ключевое слово `Given` для ввода СЛАУ в матричной форме, а во вто-

ром — в явном виде определяется функция f(x), подлежащая минимизации. Обе встроенные функции, как вы помните, применяют итерационные численные алгоритмы и поэтому требуют ввода начальных значений для всех неизвестных, т. е. нулевой итерации (вторая строка обоих листингов).

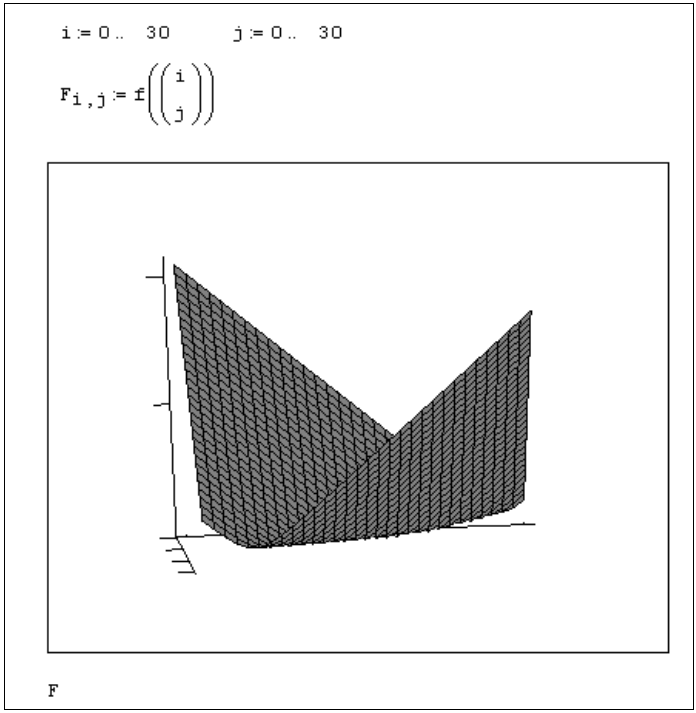


Рис. 8.1. График минимизируемой функции невязки f(x) = |A·x–b|

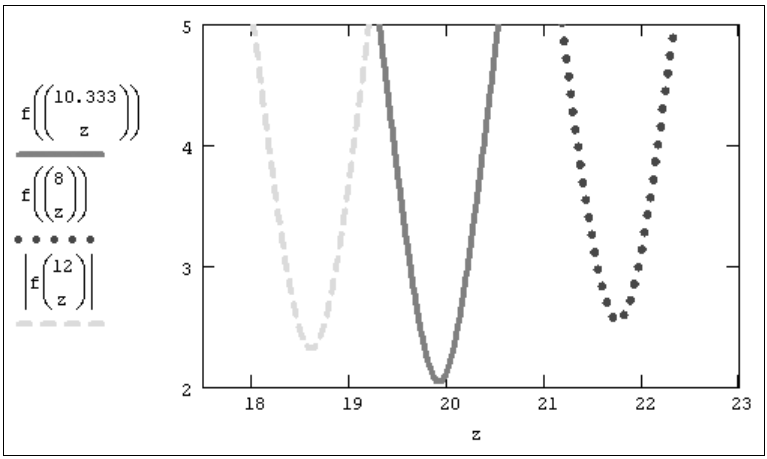


Рис. 8.2. Сечения графика невязки f(x) для трех фиксированных значений x_0

Листинг 8.8. Поиск псевдорешения при помощи функции `Minerr`

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 51 \\ 109 \\ 172 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Given

$A \cdot x = b$

$$\mathrm{MinErr}(x) = \begin{pmatrix} 12.364 \\ 17.979 \end{pmatrix}$$

Листинг 8.9. Поиск псевдорешения при помощи функции `Minimize`

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 51 \\ 109 \\ 172 \end{pmatrix}$$

$$x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$f(x) := |A \cdot x - b|$

$$\mathrm{Minimize}(f, x) = \begin{pmatrix} 10.333 \\ 19.917 \end{pmatrix}$$

$$\mathrm{lsolve}(A, b) = \begin{pmatrix} 10.333 \\ 19.917 \end{pmatrix}$$

На первый взгляд, удобнее использовать функцию `Minerr`, поскольку для нее текст Mathcad-программы (листинг 8.8) выглядит более приближенным к изначальной постановке задачи (а именно, найти приближенное решение заданной системы уравнений). Между тем, выбор функции `Minerr` таит в себе, по крайней мере, две опасности.

Во-первых, сравнивая результаты, можно с недоумением обнаружить, что они абсолютно различны, причем более похож на правильный ответ, выдаваемый функцией `Minimize` (об этом можно судить, опираясь на соображение ожидаемой близости псевдорешения к решению системы из листинга 8.7, от которого мы отошли лишь немного). Разгадка заключается в особенностях применения численного алгоритма, заложенного в функцию `Minerr`. По умолчанию решение линейной системы уравнений производится при помощи линейного алгоритма. Чтобы сменить его на нелинейный, необходимо нажатием правой кнопки мыши вызвать из области имени функции контекстное меню и сменить тип метода **Линейные** (Linear) на один из нелинейных методов **Нелинейные** (Nonlinear) (рис. 8.3). Однако даже в

этом случае, как показывает опыт, наиболее правильный ответ получается только для одного из трех доступных нелинейных численных алгоритмов.

Во-вторых, приближенное решение СЛАУ посредством вычислительного блока `Given/Minerr` невозможно при наличии дополнительных условий, выражающих вспомогательную априорную информацию о постановке задачи. В результате исходная задача сводится к условной минимизации функции `f(x)`, что в Mathcad-программе соответствует появлению дополнительных неравенств после ключевого слова `Given` (*см. пример ниже, решенный в листинге 8.10*). Однако все уравнения и неравенства, предшествующие функции `Minerr`, будут восприниматься численным процессором Mathcad одинаково, т. е. не в качестве жестких условий, а как выражения, которые должны выполняться лишь с некоторой точностью. В результате найденное решение может вовсе не отвечать условию положительной определенности (этот вопрос обсуждался в *разд. 6.2*, когда речь шла о задачах оптимизации).

Рис. 8.3. Изменение численного алгоритма для функции `Minerr`

В свете перечисленных проблем применение функции `Minimize` для поиска псевдорешения СЛАУ выглядит более обоснованным. К слову заметим, что выбор любого из трех нелинейных численных алгоритмов минимизации приводит к одинаковым результатам (листинг 8.9). Функция `Minimize` годится и для отыскания псевдорешения с дополнительными условиями, выражающими имеющуюся априорную информацию. Соответствующий пример иллюстрирует листинг 8.10, в котором осуществляется поиск псевдорешения СЛАУ, немного отличающейся от рассмотренной выше. Это отличие приводит к тому, что при использовании той же самой методики получается отрицательное решение (третья строка листинга). При добавлении в задачу дополнительных условий результат минимизации оказывается другим (последняя строка листинга 8.10).

Примечание

Несколько иной подход к переопределенным системам, для которых имеется дополнительная информация другого типа (а именно априорная оценка неизвестного вектора `x`), будет изложен в следующем разделе.

Листинг 8.10. Поиск псевдорешения при наличии априорной информации

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \qquad b := \begin{pmatrix} 62 \\ 109 \\ 172 \end{pmatrix}$$

$$f(x) := |A \cdot x - b| \qquad x := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\text{Minimize}(f, x) = \begin{pmatrix} -4.333 \\ 31.833 \end{pmatrix}$$

Given

$x_0 > 0.1$

$x_1 > 0.1$

$$\text{Minimize}(f, x) = \begin{pmatrix} 0.1 \\ 28.35 \end{pmatrix}$$

8.2.2. Недоопределенные системы

Альтернативным рассмотренному в предыдущем разделе классу СЛАУ с прямоугольной матрицей размера M×N (при M<N) является случай, когда количество уравнений меньше количества неизвестных. Как несложно сообразить, такие системы либо имеют бесконечное число решений, либо не имеют решения вовсе. Решение несовместной системы можно искать в смысле минимизации нормы невязки по методу наименьших квадратов (*см. разд. 8.2.1 и 8.2.3*). Ниже в этом разделе будут разобраны задачи, имеющие бесконечное множество решений.

О постановке задач

Пример аналитического решения системы двух уравнений с тремя неизвестными при помощи символьного процессора приведен в листинге 8.11.

Листинг 8.11. Аналитический поиск семейства решений недоопределенной СЛАУ

Given

$1x + 3y + 5z = 10$

$2x + 4y + 6z = 10$

$$\text{Find}(x, y) \rightarrow \begin{pmatrix} z - 5 \\ -2 \cdot z + 5 \end{pmatrix}$$

Приведенную простую систему двух уравнений нам удалось без труда решить аналитически, однако для решения недоопределенных систем, состоящих из большого числа уравнений, необходимо уметь использовать численные алгоритмы.

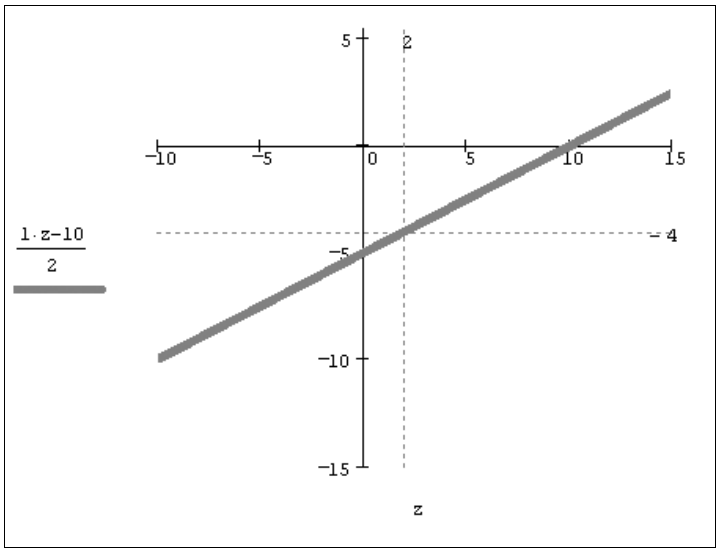


Рис. 8.4. График всех решений уравнения $x_0-2x_1=10$

Рассмотрим в качестве еще одного модельного примера единственное уравнение с двумя неизвестными $x_0-2x_1=10$. Хорошей визуализацией, подчеркивающей специфику данной задачи, будет график геометрического места его решений на плоскости (x_0,x_1) (рис. 8.4). Заметим, что на этом и на следующем рисунке мы использовали обозначение (для корректного построения графика) $x_0=z$. Очевидно, что решений уравнения бесконечно много, и все они находятся на прямой линии $x_1=(x_0-10)/2$.

Как и в предыдущем примере (листинг 8.11), нам удалось решить задачу аналитически, и опять-таки число решений получилось бесконечным. Для того чтобы получить возможность осмысленного численного решения подобных (уже не настолько тривиальных) задач, необходимо выделить из бесконечного множества решений одно, оправданное с математической точки зрения, и предложить алгоритм его поиска. Эта проблема решается посредством привлечения понятия *нормального псевдорешения*.

Нормальное псевдорешение

Способ выбора одного решения из бесконечного множества, изображенного на рис. 8.4, подсказывает, по аналогии с переопределенными СЛАУ (*см. разд. 8.2.1*), сам физический смысл задачи, которую можно интерпретировать как M измерений с N неизвестными ($M<N$). Для того чтобы получить разумное единственное решение задачи, необходимо "доопределить" ее, добавив некоторые априорные соображения о значении неизвестного вектора x.

Если априорной информации о примерной величине вектора x нет, единственным образом решить СЛАУ невозможно. Однако если о неизвестном векторе хоть что-то можно сказать, данная информация позволит доопределить систему уравнений

и получить решение, учитывающее как систему, так и априорную информацию. Иными словами, следует ввести в задачу определенные ожидания о величине вектора x. Математически, не теряя общности, можно полагать ожидаемое значение вектора x нулевым, поскольку перейти от любого x к нуль-вектору можно простым линейным преобразованием переменных, которое изменит только вектор правой части b.

Таким образом, вполне логично объявить решением недоопределенной СЛАУ такое из решений, которое ближе всего находится к нулевому вектору, т. е. обладает минимальной нормой |x|~min. Это решение называют *нормальным псевдорешением* СЛАУ, и искать его следует, минимизируя норму вектора x на предварительно полученном семействе решений СЛАУ. Иными словами, решение недоопределенной СЛАУ сводится к условной минимизации функции |x| (рис. 8.5). Геометрический смысл нормального псевдорешения (в рассматриваемом случае одного уравнения с двумя неизвестными) очевиден: это точка, лежащая на пересечении прямой семейства всех решений и перпендикуляра к этой прямой, восстановленного из начала координат. На рис. 8.4 нормальное псевдорешение выделено пунктирными линиями.

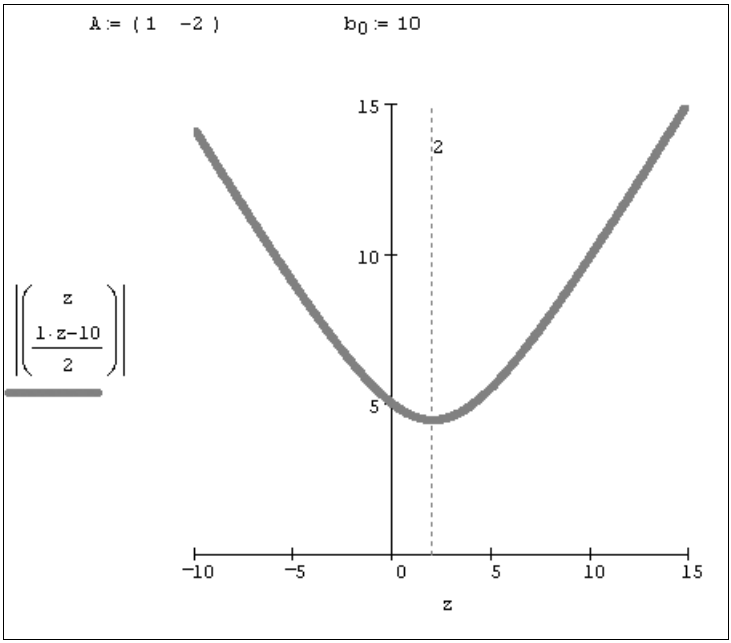


Рис. 8.5. График функции $f(x_0)=|x|$ при условии, что $x_0-2x_1=10$

Принимая во внимание введенную технику решения недоопределенных СЛАУ, можно предложить для этих задач простой и понятный алгоритм, опирающийся на встроенную функцию Minimize. В листинге 8.12 решена задача численного отыскания псевдорешения рассматриваемого единственного уравнения (рис. 8.5), а в листинге 8.13 — решение недоопределенной системы двух уравнений с тремя неизвестными, которая исследовалась аналитически в листинге 8.11 (*см. предыдущий*

раздел). Как уже отмечалось, смысл обоих листингов заключается в минимизации нормы искомого вектора при условии, что выполнена система равенств A·x=b. Графики функции f(x)=|x|, при условии выполнения СЛАУ из листингов 8.12 и 8.13, изображены на рис. 8.5 и 8.6 соответственно.

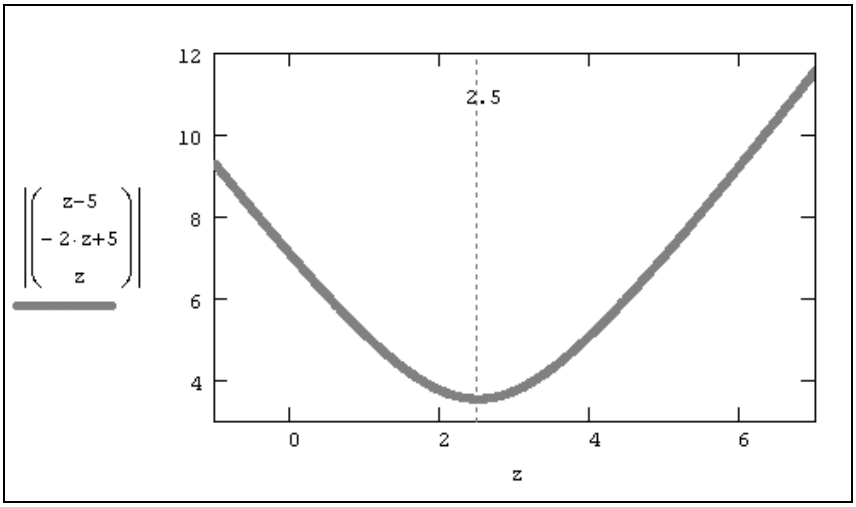


Рис. 8.6. График функции $f(x_2)=|x|$
при условии выполнения СЛАУ из листинга 8.13

Листинг 8.12. Поиск нормального псевдорешения уравнения $x_0-2x_1=10$

$A := (1 \quad -2)$

$b_0 := 10$

$\text{lsolve}(A, b) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}$

$f(x) := |x|$

$x := (0 \quad 0)^T$

Given

$A \cdot x = b$

$\text{Minimize}(f, x) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}$

Листинг 8.13. Поиск нормального псевдорешения недоопределенной СЛАУ

$A := \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$ $\qquad$ $b := \begin{pmatrix} 10 \\ 10 \end{pmatrix}$

$$\text{lsolve}(A,b) = \begin{pmatrix} -2.5 \\ -2.887\times 10^{-15} \\ 2.5 \end{pmatrix}$$

$$f(x) := |x|$$

$$x := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Given

$$A\cdot x = b$$

$$\text{Minimize}(f,x) = \begin{pmatrix} -2.5 \\ 5.101\times 10^{-7} \\ 2.5 \end{pmatrix}$$

Примечание

Если недоопределенная СЛАУ не имеет бесконечного множества решений, а является несовместной, то способ, предложенный в листинге 8.13, использовать нельзя, т. к. условие заведомо невыполнимо. Подход к решению таких задач заключается в поиске компромиссного решения, минимизирующего как норму невязки, так и норму решения (*см. разд. 8.2.3*).

8.2.3. Вырожденные и плохо обусловленные системы

Вернемся вновь к СЛАУ $A\cdot x=b$ с квадратной матрицей A размера $M\times M$, которая, в отличие от рассмотренного выше "хорошего" случая (*см. разд. 8.1*), требует специального подхода. Обратим внимание на два похожих типа СЛАУ:

- вырожденная система (с нулевым определителем $|A|=0$);
- плохо обусловленная система (определитель A не равен нулю, но число обусловленности очень велико).

Несмотря на то, что эти типы систем уравнений существенно отличаются друг от друга (для первого решение отсутствует, а для второго существует и единственно), с практической точки зрения вычислителя, между ними много общего.

Вырожденные СЛАУ

Вырожденная система — это система, описываемая матрицей с нулевым определителем $|A|=0$ (*сингулярной* матрицей). Поскольку некоторые уравнения, входящие в такую систему, представляются линейной комбинацией других уравнений, то фактически сама система является недоопределенной. Несложно сообразить, что, в зависимости от конкретного вида вектора правой части b, существует либо бесконечное множество решений, либо не существует ни одного.

Первый из вариантов сводится к построению нормального псевдорешения (т. е. выбора из бесконечного множества решений такого, которое наиболее близко к определенному, например нулевому, вектору). Данный случай был подробно рассмотрен в *разд. 8.2.2* (см. листинги 8.11—8.13).

Рассмотрим второй случай, когда СЛАУ `A·x=b` с сингулярной квадратной матрицей `A` не имеет ни одного решения. Пример такой задачи (для системы двух уравнений) проиллюстрирован на рис. 8.7, в верхней части которого вводятся матрица `A` и вектор `b`, а также предпринимается (неудачная, поскольку матрица `A` сингулярная) попытка решить систему при помощи функции `lsolve`. График, занимающий основную часть рисунка, показывает, что два уравнения, задающие систему, определяют на плоскости (x_0, x_1) две параллельные прямые. Прямые не пересекаются ни в одной точке координатной плоскости, и решения системы, соответственно, не существует.

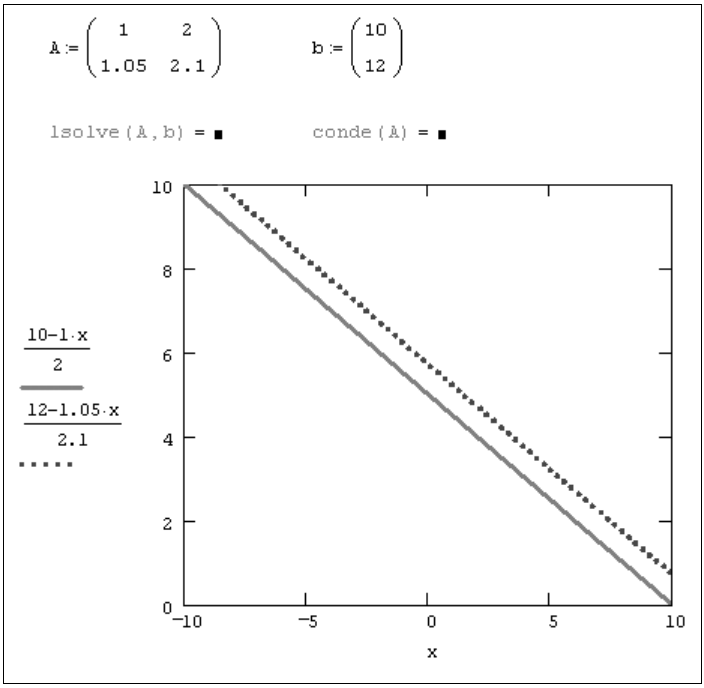


Рис. 8.7. Графическое представление несовместной системы двух уравнений с сингулярной матрицей

Примечание

Во-первых, заметим, что СЛАУ, заданная несингулярной квадратной матрицей размера 2×2, определяет на плоскости пару пересекающихся прямых (см. рис. 8.9). Во-вторых, стоит сказать, что, если бы система была совместной, то геометрическим изображением уравнений были бы две совпадающие прямые, описывающие бесконечное число решений.

Несложно догадаться, что в рассматриваемом сингулярном случае псевдорешений системы, минимизирующих невязку `|A·x−b|`, будет бесконечно много, и лежать они

будут на третьей прямой, параллельной двум показанным на рис. 8.7 и расположенной в середине между ними. Это иллюстрирует рис. 8.8, на котором представлено несколько сечений функции `f(x)=|A·x−b|`, которые говорят о наличии семейства минимумов одинаковой глубины. Если попытаться использовать для их отыскания встроенную функцию `Minimize`, ее численный метод будет всегда отыскивать какую-либо одну точку упомянутой прямой (в зависимости от начальных условий). Поэтому для определения единственного решения следует выбрать из всего множества псевдорешений то, которое обладает наименьшей нормой. Можно пытаться оформить данную многомерную задачу минимизации в Mathcad при помощи комбинаций встроенных функций `Minimize`, однако более эффективным способом будет использование регуляризации (см. ниже) или ортогональных матричных разложений (*см. разд. 8.3*).

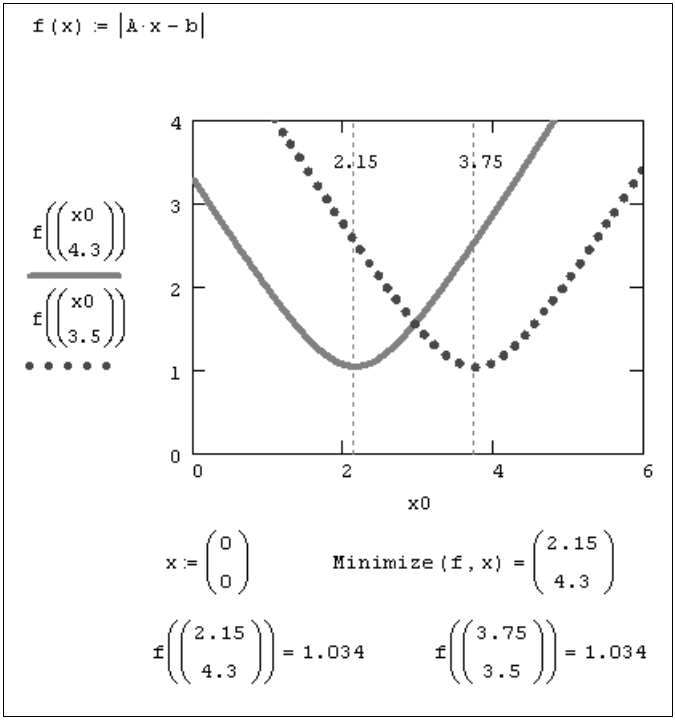


Рис. 8.8. График сечений функции невязки `f(x)=|A·x−b|`

Плохо обусловленные системы

Плохо обусловленная система — это система, у которой определитель `A` не равен нулю, но число обусловленности $|A^{-1}|\cdot|A|$ очень велико. Несмотря на то, что плохо обусловленные системы имеют единственное решение, на практике искать это решение чаще всего не имеет смысла. Рассмотрим свойства плохо обусловленных СЛАУ на двух конкретных примерах (листинг 8.14).

Листинг 8.14. Решение двух близких плохо обусловленных СЛАУ

$$\mathrm{lsolve}\left[\begin{pmatrix} 1 & 2 \\ 3 & 6.01 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}\right] = \begin{pmatrix} 803 \\ -400 \end{pmatrix}$$

$$\mathrm{lsolve}\left[\begin{pmatrix} 1 & 2.01 \\ 3 & 6 \end{pmatrix}, \begin{pmatrix} 3 \\ 5 \end{pmatrix}\right] = \begin{pmatrix} -265 \\ 133.333 \end{pmatrix}$$

Каждая строка листинга 8.14 содержит решение двух очень близких плохо обусловленных СЛАУ (с одинаковой правой частью `b` и мало отличающимися матрицами `A`). Несмотря на близость, точные решения этих систем оказываются очень далекими друг от друга. Надо заметить, что для системы двух уравнений точное решение получить легко, однако при решении СЛАУ большой размерности незначительные ошибки округления, неминуемо накапливаемые при расчетах (в том числе и "точным" алгоритмом Гаусса), приводят к огромным погрешностям результата. Возникает вопрос: имеет ли смысл искать численное решение, если заранее известно, что, в силу неустойчивости самой задачи, оно может оказаться совершенно неправильным?

Еще одно соображение, которое вынуждает искать специальные методы решения плохо обусловленных СЛАУ (даже приведенной в качестве примера в листинге 8.14 системы двух уравнений), связано с их физической интерпретацией как результатов эксперимента. Если изначально известно, что входные данные получены с некоторой погрешностью, то решать плохо обусловленные системы не имеет вовсе никакого смысла, поскольку малые ошибки модели (матрицы `A` и вектора `b`) приводят к большим ошибкам решения. Задачи, обладающие такими свойствами, называются *некорректными*.

Чтобы лучше понять причину некорректности, полезно сравнить графическую интерпретацию хорошо (рис. 8.9) и плохо (рис. 8.10) обусловленной системы двух уравнений. Решение системы визуализируется точкой пересечения двух прямых линий, изображающих каждое из уравнений.

Из рис. 8.10 видно, что прямые, соответствующие плохо обусловленной СЛАУ, располагаются в непосредственной близости друг от друга (почти параллельны). В связи с этим малые ошибки в расположении каждой из прямых могут привести к значительным погрешностям локализации точки их пересечения (решения СЛАУ) в противоположность случаю хорошо обусловленной системы, когда малые погрешности в наклоне прямых мало повлияют на место точки их пересечения (рис. 8.9).

Примечание

Плохая обусловленность матрицы типична и при реконструкции экспериментальных данных, задаваемых переопределенными (несовместными) СЛАУ (например, в задачах томографии). Именно такой случай приведен в качестве примера в следующем разделе (см. листинг 8.16).

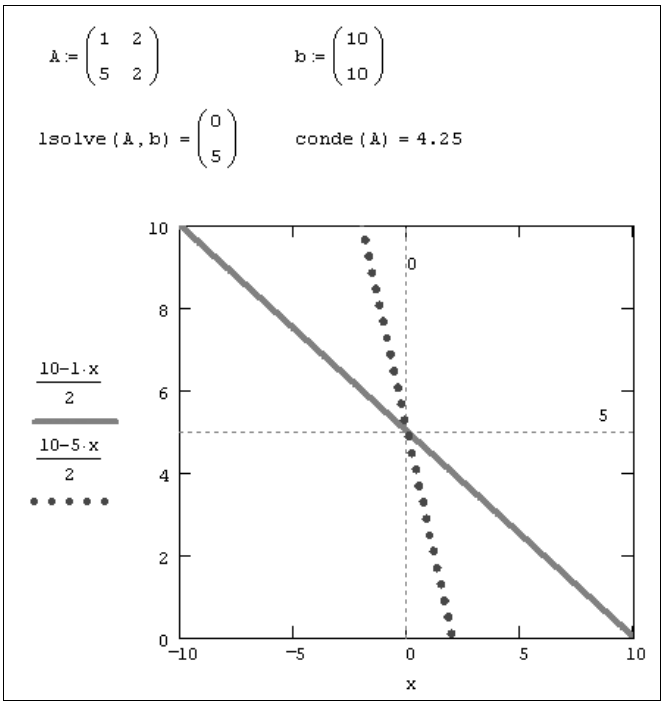


Рис. 8.9. График хорошо обусловленной системы двух уравнений

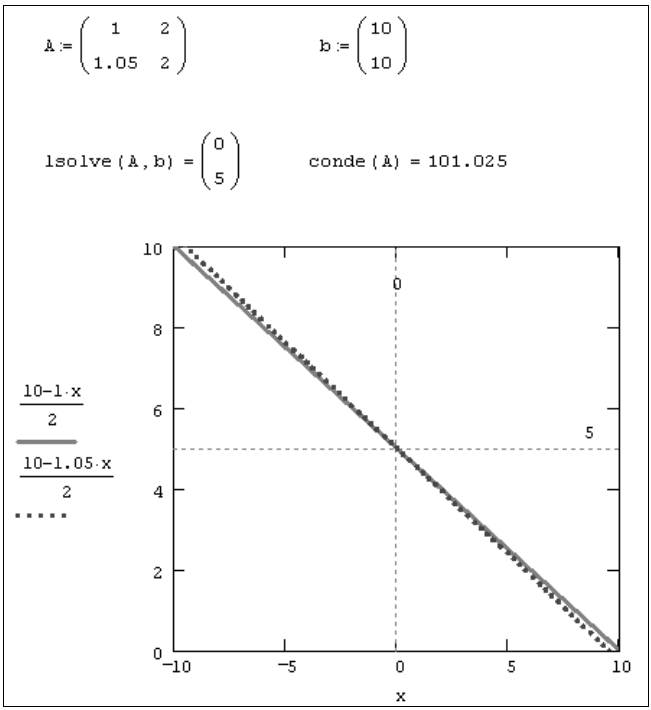


Рис. 8.10. График плохо обусловленной системы двух уравнений

Метод регуляризации

Для решения некорректных задач, в частности вырожденных и плохо обусловленных СЛАУ, разработан очень эффективный прием, называемый *регуляризацией*. В его основе лежит учет дополнительной априорной информации о структуре решения (векторе априорной оценки $x0$), которая очень часто присутствует в практических случаях. В связи с тем, что регуляризация была подробно рассмотрена в *разд. 6.3.3*, напомним лишь, что задачу решения СЛАУ $A \cdot x = b$ можно заменить задачей отыскания минимума *функционала Тихонова*:

$$\Omega(x,\lambda) = |A \cdot x - b|^2 + \lambda \cdot |x - x0|^2. \tag{8.3}$$

Здесь λ — малый положительный *параметр регуляризации*, который необходимо подобрать некоторым оптимальным способом. Легко показать, что задачу минимизации функционала Тихонова можно, в свою очередь, свести к решению другой СЛАУ:

$$(A^T \cdot A + \lambda \cdot I) \cdot x = A^T \cdot B + \lambda \cdot x0, \tag{8.4}$$

которая при $\lambda \to 0$ переходит в исходную плохо обусловленную систему, а при больших λ, будучи хорошо обусловленной, имеет решение $x0$. Очевидно, оптимальным будет некоторое промежуточное значение λ, устанавливающее определенный компромисс между приемлемой обусловленностью и близостью к исходной задаче. Отметим, что регуляризационный подход сводит некорректную задачу к условно-корректной (по Тихонову) задаче отыскания решения системы (8.4), которое, в силу линейности задачи, является единственным и устойчивым.

Приведем, без излишних комментариев, регуляризованное решение вырожденной системы, которая была представлена на рис. 8.8. Листинг 8.15 демонстрирует отыскание решения задачи (8.4), а полученная зависимость невязки и самого решения от параметра регуляризации λ показана на рис. 8.11 и 8.12 соответственно. Важно подчеркнуть, что решения исходной системы и, следовательно, системы (8.4) при $\lambda = 0$ не существует.

Листинг 8.15. Регуляризация вырожденной СЛАУ

$$A := \begin{pmatrix} 1 & 2 \\ 1.05 & 2.1 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 10 \end{pmatrix} \qquad x0 := \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$g(\lambda) := \mathrm{lsolve}\left(A^T \cdot A + \lambda \cdot \mathrm{identity}(2),\ A^T \cdot b + \lambda \cdot x0\right)$$

Заключительным шагом регуляризации является выбор оптимального λ. Имеется, по крайней мере, два соображения, исходя из которых можно выбрать параметр регуляризации, если опираться на зависимость от него невязки. В рассматриваемом примере используется критерий определения λ, опирающийся на подбор нормы невязки, равной априорной оценке погрешностей задания входных данных: матрицы A и вектора b, т. е. величине $|\delta A| + |\delta b|$. Например, можно выбрать норму невязки и, соответственно, параметр λ и решение $x(\lambda)$, которые отмечены на рис. 8.11 и 8.12 пунктирами.

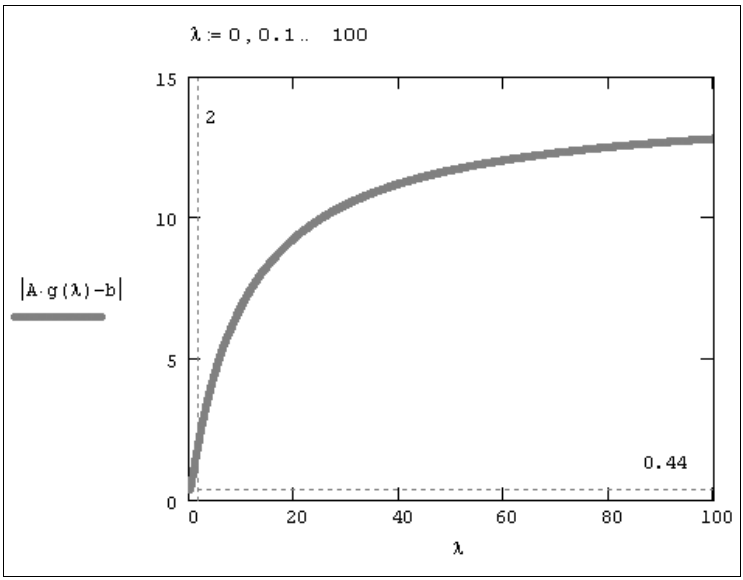


Рис. 8.11. Зависимость невязки регуляризованного решения вырожденной СЛАУ от параметра λ (продолжение листинга 8.15)

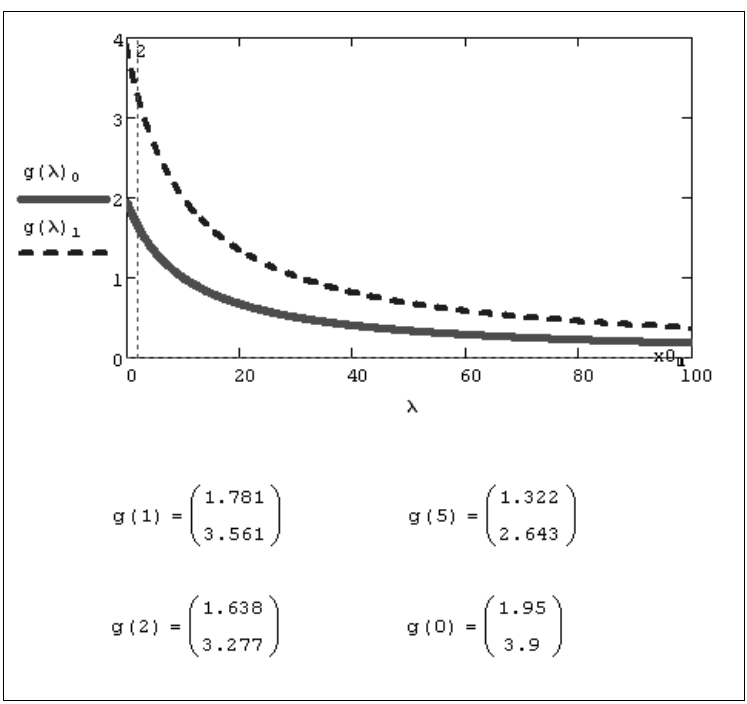


Рис. 8.12. Регуляризованное решение в зависимости от λ (продолжение листинга 8.15)

Примечание 1

Другим вариантом выбора λ, не требующим никаких априорных соображений относительно погрешностей модели, является так называемый *квазиоптимальный* метод, рассмотренный в *разд. 6.3*.

Примечание 2

Полезно убедиться в том, что формула (8.4) в случае линейной задачи дает тот же результат, что и общая формула (8.3). Для этого достаточно изменить в листинге 8.15 последнюю строку, выражающую решение СЛАУ (8.4), на код, реализующий минимизацию численным методом, как это показано в листинге 8.16. Расчеты (которые требуют значительно большего компьютерного времени) дают тот же самый результат, что был приведен на рис. 8.11 и 8.12.

Примечание 3

Попробуйте в расчетах листингов 8.15 и 8.16 взять иную, например, более реалистичную, априорную оценку решения (вместо использованного в них нулевого вектора `x0`) и посмотреть, как это повлияет на результат.

Примечание 4

Любопытно также применить вместо формулы (8.3) в качестве функционала Тихонова другую зависимость: $\Omega(x,\lambda)=|A\cdot x-b|+\lambda\cdot|x-x0|$. Соответствующий пример расчетов вы найдете на компакт-диске, прилагаемом к данной книге.

Листинг 8.16. Регуляризация СЛАУ при помощи алгоритма минимизации λ (продолжение листинга 8.15)

$$f(x,\lambda) := |A\cdot x-b|^2+\lambda\cdot|x-x0|^2$$

$$x := \begin{pmatrix}0\\0\end{pmatrix} \qquad g(\lambda) := \text{Minimize}(f,x)$$

8.3. Матричные разложения

Современная вычислительная линейная алгебра — бурно развивающаяся наука. Главная проблема линейной алгебры — это решение систем линейных уравнений. В настоящее время разработано множество методов, упрощающих эти задачи, которые, в частности, зависят от структуры матрицы СЛАУ. Большинство методов основано на представлении матрицы в виде произведения других матриц специального вида, или *матричных разложениях* (или, по-другому, *факторизациях*). Как правило, после определенного разложения матрицы задача решения СЛАУ существенно упрощается. Кроме того, прибегая к определенным разложениям матриц, можно получить универсальный вид решения как для "хороших" (*см. разд. 8.1*), так и для "плохих" (*см. разд. 8.2*) СЛАУ.

В Mathcad имеется несколько встроенных функций, реализующих алгоритмы наиболее популярных матричных разложений; и мы рассмотрим их вместе с соответствующими типовыми задачами решения СЛАУ. Однако вначале сделаем некоторые важные замечания о решении систем с треугольной матрицей.

8.3.1. СЛАУ с треугольной матрицей

Начнем разговор о решении СЛАУ посредством матричных разложений с простого, но исключительно важного частного случая, а именно систем с *треугольной матрицей*, т. е. такой матрицей, по одну из сторон от диагонали которой находятся одни нули. В листинге 8.17 приведен пример верхнетреугольной `L` матрицы (первая строка листинга), а также ряд формул, по которым за `N` операций рассчитывается решение соответствующей линейной системы. Из этого примера ясно, что сведение задачи общего вида к СЛАУ с треугольной матрицей практически решает эту задачу, поскольку гарантировано получение результата за минимальное число операций.

Примечание 1

Подчеркнем, что мы нарочно оформили расчет решения треугольной СЛАУ в виде пользовательской функции `trg(A,b)`, для чего нам пришлось применить элементы программирования. Мы так поступили, чтобы иметь впоследствии возможность применять ее в качестве подпрограммы решения СЛАУ посредством различных матричных разложений, включая листинг 8.17 в последующие листинги. Более того, несложно модифицировать листинг 8.17 для решения нижнетреугольных систем, определив, таким образом, еще одну функцию `ltrg(A,b)` (вы найдете соответствующий листинг на компакт-диске, прилагаемом к данной книге).

Примечание 2

Алгоритм решения СЛАУ с треугольной матрицей называют иногда *прямым ходом* решения СЛАУ (часто подразумевая, что до него выполнены определенные преобразования, именуемые *обратным ходом*) либо просто *подстановкой*. В частности, решение еще одного часто встречающегося типа систем с трехдиагональной матрицей сводится к прямому и обратному ходу алгоритма, называемого *прогонкой* (*см. разд. 11.2.2*).

Листинг 8.17. Решение СЛАУ с треугольной матрицей (прямой ход)

$$A := \begin{pmatrix} 1 & 2 & -3 \\ 0 & 5 & 6 \\ 0 & 0 & -3 \end{pmatrix} \qquad b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix}$$

$$\mathrm{trg}(A,b) := \left| \begin{array}{l} N \leftarrow \mathrm{cols}(A) \\ x_{N-1} \leftarrow \dfrac{b_{N-1}}{A_{N-1,N-1}} \\ \text{for } i \in N-2 .. 0 \\ \quad x_i \leftarrow \dfrac{1}{A_{i,i}} \cdot \left(b_i - \displaystyle\sum_{j=i+1}^{N-1} A_{i,j} \cdot x_j \right) \\ x \end{array} \right.$$

$$x := \mathrm{trg}(A,b)$$

$$x = \begin{pmatrix} -52 \\ 16 \\ -10 \end{pmatrix} \qquad A \cdot x - b = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

8.3.2. Разложение Холецкого

Разложением Холецкого *симметричной* (т. е. содержащей одинаковые элементы на местах, расположенных симметрично относительно главной диагонали) матрицы `A` является представление вида $A=L \cdot L^T$, где `L` — треугольная матрица. Алгоритм Холецкого реализован во встроенной функции `cholesky`:

- `cholesky(A)` — разложение Холецкого:
  - `A` — квадратная, положительно определенная симметричная матрица.

Пример разложения Холецкого приведен в листинге 8.18. Обратите внимание, что в результате получается верхняя треугольная матрица (нули сверху от диагонали), а транспонированная матрица является нижней треугольной. В последней строке листинга приведена проверка правильности найденного разложения.

> ***Примечание***
> Исходя из математического вида разложения Холецкого, матрицу `L` иногда называют *квадратным корнем* матрицы `A`.

Листинг 8.18. Разложение Холецкого

$$A := \begin{pmatrix} 5 & 2 & 3 \\ 2 & 6 & 1 \\ 3 & 1 & 7 \end{pmatrix}$$

$L := \text{cholesky}(A)$

$$L = \begin{pmatrix} 2.236 & 0 & 0 \\ 0.894 & 2.28 & 0 \\ 1.342 & -0.088 & 2.279 \end{pmatrix} \qquad L^T = \begin{pmatrix} 2.236 & 0.894 & 1.342 \\ 0 & 2.28 & -0.088 \\ 0 & 0 & 2.279 \end{pmatrix}$$

$$L \cdot L^T - A = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Решение СЛАУ, если разложение Холецкого для него известно, основано на замене исходной системы $A \cdot x=b$ другой системой $L \cdot y=b$ (где $y=L^T \cdot x$), что иллюстрируется листингом 8.19. В первой строке листинга задается вектор правой части `b` и вычисляется стандартным методом Гаусса решение системы. В оставшейся части листинга СЛАУ решается при помощи разложения Холецкого, проведенного в листинге 8.18. После нахождения (простой подстановкой, т. к. матрица `L` — треугольная) вектора `y`, опять-таки подстановкой в уравнение $y=L^T \cdot x$ (т. к. L^T — тоже треугольная матрица), отыскивается вектор `x`.

Внимание!

В листинге 8.19 использованы пользовательские функции решения треугольных СЛАУ, описанные в предыдущем разделе. Если вы набираете листинг "от руки", их можно заменить универсальной встроенной функцией `lsolve`.

Листинг 8.19. Решение СЛАУ при помощи разложения Холецкого (продолжение листингов 8.17 и 8.18)

$$b := \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \qquad \mathrm{lsolve}(A, b) = \begin{pmatrix} -2.148 \\ 3.259 \\ 4.741 \end{pmatrix}$$

$$y := \mathrm{ltrg}(L, b)$$

$$x := \mathrm{trg}\left(L^T, y\right)$$

$$x = \begin{pmatrix} -2.148 \\ 3.259 \\ 4.741 \end{pmatrix}$$

8.3.3. *LU*-разложение

`LU`-разложением матрицы `A`, или *треугольным* разложением, называется матричное разложение вида `P·A=L·U`, где `L` и `U` — нижняя и верхняя треугольные матрицы соответственно, а `P` — (диагональная) матрица перестановок. `P`, `A`, `L`, `U` — квадратные матрицы одного порядка:

- `lu(A)` — `LU`-разложение матрицы:
  - `A` — квадратная матрица.

Результатом работы встроенной функции `LU`-разложения является матрица, составленная из матриц `L` и `U` соответственно. Чтобы выделить сами матрицы `LU`-разложения, а именно `P`, `L`, `U`, необходимо применить функцию выделения подматрицы `submatrix` (листинг 8.20).

Листинг 8.20. `LU`-разложение

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{pmatrix}$$

$$B := \mathrm{lu}(A)$$

$$B = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 7 & 8 & 10 \\ 1 & 0 & 0 & 0.143 & 1 & 0 & 0 & 0.857 & 1.571 \\ 0 & 1 & 0 & 0.571 & 0.5 & 1 & 0 & 0 & -0.5 \end{pmatrix}$$

$$P := \text{submatrix}\left(B, 0, \text{rows}(B) - 1, 0, \frac{\text{cols}(B) - 1}{3}\right)$$

$$L := \text{submatrix}\left(B, 0, \text{rows}(B) - 1, \frac{\text{cols}(B) + 1}{3}, \frac{\text{cols}(B) - 1}{3} 2\right)$$

$$U := \text{submatrix}\left(B, 0, \text{rows}(B) - 1, \frac{\text{cols}(B) + 1}{3} 2, \text{cols}(B) - 1\right)$$

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 0.143 & 1 & 0 \\ 0.571 & 0.5 & 1 \end{pmatrix} \quad U = \begin{pmatrix} 7 & 8 & 10 \\ 0 & 0.857 & 1.571 \\ 0 & 0 & -0.5 \end{pmatrix}$$

$$\left|P \cdot A - L \cdot U\right| = 0$$

Треугольное разложение матрицы системы линейных уравнений производится при ее решении численным методом Гаусса. Фактически именно алгоритм LU-разложения заложен во встроенной функции lsolve, применяемой для "точного" решения хорошо обусловленных систем (*см. разд. 8.1.2*). Таким образом, посредством LU-разложения можно решать СЛАУ с хорошо обусловленной матрицей A.

Почему метод LU-разложения является мощным средством решения СЛАУ и особенно важен при обработке соответствующих экспериментальных данных? Чтобы понять это, заменим исходную систему A·x=b эквивалентной системой P·L·U·x=P·b, а ее, в свою очередь, парой других систем: L·y=P·b и U·x=y. Несложно сообразить, что обе системы решаются прямой подстановкой, т. к. матрицы P·L и U — треугольные. Сначала из первой СЛАУ определяется промежуточный вектор y, а затем (из второй системы) — искомый вектор x.

Главное преимущество метода LU-разложения заключается в том, что явный вид вектора правой части b при решении СЛАУ используется только на заключительном этапе (в формулах прямого хода), а наиболее трудоемкие операции по вычислению самих матриц L и U вовсе не требуют знания вектора b. Таким образом, если решается серия СЛАУ с одной и той же матрицей A, но разными правыми частями b (что весьма характерно в обратных задачах интерпретации эксперимента), очень выгодно единожды вычислить LU-разложение матрицы A, а уже затем быстрой подстановкой решить каждую из конкретных систем. Сказанное иллюстрирует листинг 8.21, в котором решается две СЛАУ с матрицей из предыдущего листинга и различными векторами правых частей b.

Листинг 8.21. Решение СЛАУ при помощи LU-разложения (продолжение листинга 8.20)

$$b := \begin{pmatrix} 6 \\ 12 \\ 24 \end{pmatrix} \qquad \text{lsolve}(A, b) = \begin{pmatrix} 4 \\ -8 \\ 6 \end{pmatrix}$$

$$y := \text{ltrg}(L, P \cdot b)$$

$$x := \text{trg}(U, y)$$

$$x = \begin{pmatrix} 4 \\ -8 \\ 6 \end{pmatrix}$$

8.3.4. *QR*-разложение

Среди матричных разложений особую роль играют ортогональные преобразования, обладающие свойством сохранения нормы вектора. Напомним из курса линейной алгебры, что матрица Q называется *ортогональной*, если $Q^T \cdot Q = I$, где I — единичная матрица. Свойство сохранения нормы вектора при ортогональных преобразованиях, т. е. $|Q \cdot x| = |x|$, дает рецепт поиска псевдорешения вырожденных СЛАУ, а именно замену исходной задачи минимизации невязки с "плохой" матрицей $|A \cdot x - b|$ задачей $|Q^T \cdot (A \cdot x - b)|$, в которой матрица уже будет "хорошей" благодаря специальному построению матрицы Q. Таким образом, ортогональные разложения используются при решении любых систем (в том числе с прямоугольной матрицей A, причем как переопределенных, так и недоопределенных).

Одним из важнейших вариантов ортогональных разложений некоторой матрицы A является *QR-разложение* вида $A = Q \cdot R$, где Q — ортогональная матрица, а R — верхняя треугольная матрица:

❒ qr(A) — QR-разложение:

- A — вектор или матрица любого размера.

Результатом действия функции qr(A) является матрица, составленная из матриц Q и R соответственно (подобно рассмотренному в предыдущем разделе LU-разложению). Выделить матрицы QR-разложения несложно при помощи встроенной функции выделения подматрицы submatrix (листинг 8.22).

Листинг 8.22. QR-разложение сингулярной матрицы

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$$B := qr(A)$$

$$B = \begin{pmatrix} 0.123 & -0.905 & 0.408 & 8.124 & 9.601 & 11.078 \\ 0.492 & -0.302 & -0.816 & 0 & -0.905 & -1.809 \\ 0.862 & 0.302 & 0.408 & 0 & 0 & 0 \end{pmatrix}$$

$$Q := submatrix\left(B, 0, rows(B) - 1, 0, \frac{cols(B) - 1}{2}\right)$$

$$R := submatrix\left(B, 0, rows(B) - 1, \frac{cols(B) + 1}{2}, cols(B) - 1\right)$$

$$Q = \begin{pmatrix} 0.123 & -0.905 & 0.408 \\ 0.492 & -0.302 & -0.816 \\ 0.862 & 0.302 & 0.408 \end{pmatrix} \qquad R = \begin{pmatrix} 8.124 & 9.601 & 11.078 \\ 0 & -0.905 & -1.809 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\text{norme}(A - Q \cdot R) = 4.311 \times 10^{-15}$$

Если бы исходная СЛАУ $A \cdot x = b$ не была вырожденной, то можно было сразу записать: $Q^T \cdot Q \cdot R \cdot x = Q^T \cdot b$, откуда следует (благодаря ортогональности матрицы Q): $R \cdot x = Q^T \cdot b$. Так как матрица R — треугольная, решение данной системы получается по формулам прямого хода. Если использовать нашу пользовательскую функцию решения треугольной системы (*см. разд. 8.3.1*), то результат исходной СЛАУ запишется в виде одной строки кода: $\text{trg}(R, Q^T \cdot b)$ (соответствующий пример решения невырожденной СЛАУ вы найдете на компакт-диске, прилагаемом к данной книге).

Обратимся теперь к проблеме решения СЛАУ с сингулярной квадратной $N \times N$ или прямоугольной $N \times M$ матрицей A. Если ее ранг равен k (он может быть меньше N и M, как в листинге 8.22, где $N=M=3$, а $k=2$), то получающаяся треугольная матрица R имеет следующую структуру:

$$R = \begin{pmatrix} R1 & R2 \\ 0 & 0 \end{pmatrix},$$

где $R1$ — верхняя треугольная матрица, а $R2$ — просто некоторая матрица, а нули обозначают, в общем случае, нулевые матрицы соответствующих размеров.

Если система вырожденная, то она имеет бесконечное множество псевдорешений (векторов, минимизирующих норму невязки). При помощи QR-разложения можно сразу выписать одно из них (правда, не обладающее минимальной нормой). В нашем примере последняя строка матрицы R содержит одни нули, поэтому последняя компонента вектора псевдорешения x может быть абсолютно любой. Если положить $x_2=0$, то остальные компоненты x определятся из треугольной СЛАУ $R1 \cdot x = Q^T \cdot b$, как это проиллюстрировано листингом 8.23.

Листинг 8.23. Поиск одного из псевдорешений вырожденной СЛАУ посредством QR-разложения (продолжение листинга 8.22)

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad b := \begin{pmatrix} 6 \\ 12 \\ 24 \end{pmatrix}$$

$R1 := \text{submatrix}(R, 0, 1, 0, 1)$

$R2 := \text{submatrix}(R, 0, 1, 2, 2)$

$b1 := \text{submatrix}\left(Q^T \cdot b, 0, 1, 0, 0\right)$

$$R1 = \begin{pmatrix} 8.124 & 9.601 \\ 0 & -0.905 \end{pmatrix} \qquad R2 = \begin{pmatrix} 11.078 \\ -1.809 \end{pmatrix} \qquad b1 = \begin{pmatrix} 27.326 \\ -1.809 \end{pmatrix}$$

$x := \text{stack}\left(\text{trg}\left(R1, Q^T \cdot b\right), 0\right)$

$$x = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} \qquad |A \cdot x - b| = 2.449 \qquad |x| = 2.236$$

Для того чтобы выбрать из всего множества псевдорешений (минимизирующих невязку исходной СЛАУ) нормальное псевдорешение (т. е. обладающее минимальной нормой), необходимо решить соответствующую задачу минимизации (*см. разд. 8.2.3*). Если построено QR-разложение, сделать это намного легче. Если произвольную компоненту решения обозначить переменной y: x_2=y, то она определится из соответствующей задачи оптимизации (листинг 8.24, 1—3 строки), а остальные составляющие самого решения x — из треугольной СЛАУ $R1 \cdot x = Q^T \cdot b - R2 \cdot y$. В последней строке листинга выводится полученное нормальное псевдорешение, а также его норма и соответствующая норма невязки (которые полезно сравнить с результатом прошлого листинга). Вспомогательный рис. 8.13 помогает понять структуру минимизируемой функции из листинга 8.24.

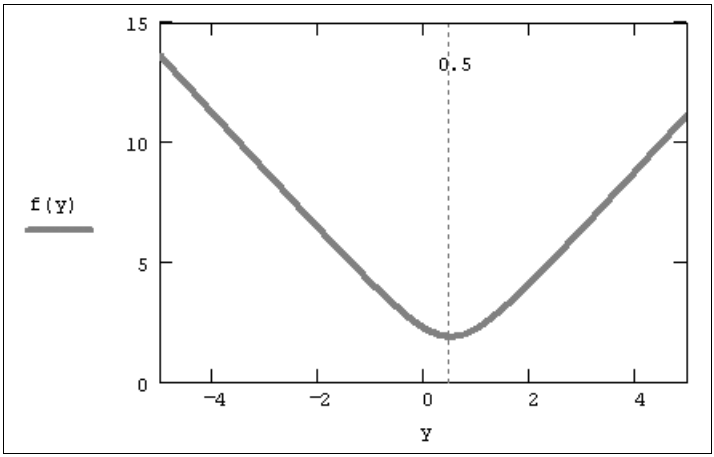


Рис. 8.13. Норма псевдорешения в зависимости от y
(продолжение листинга 8.24)

Листинг 8.24. Нормальное псевдорешение вырожденной СЛАУ (продолжение листингов 8.22 и 8.23)

$f(y) := \left|\text{stack}\left(\text{trg}\left(R1, b1 - R2 \cdot y\right), y\right)\right|$

$y := 0$

$y := \text{minimize}(f, y) \qquad y = 0.5$

$x := \text{stack}\left(\text{trg}\left(R1, b1 - R2 \cdot y\right), y\right)$

$$x = \begin{pmatrix} 1.5 \\ 1 \\ 0.5 \end{pmatrix} \qquad |A \cdot x - b| = 2.449 \qquad |x| = 1.871$$

Резюмируя практические аспекты применения QR-разложения, надо отметить, что алгоритмы решения СЛАУ на его основе практически одинаковы как для хорошо обусловленных, так и для сингулярных систем.

Примечание

На прилагаемом к книге компакт-диске вы найдете дополнительные примеры построения QR-разложения как для квадратной, так и прямоугольной матрицы A.

8.3.5. *SVD*-(сингулярное) разложение

Наиболее эффективными (но и ресурсоемкими) средствами решения произвольных СЛАУ (с матрицей A размера N×M) по методу наименьших квадратов являются так называемые полные ортогональные разложения, имеющие, по определению, вид $A=U\cdot K\cdot V^T$.

Здесь U и V — ортогональные матрицы размером N×N и M×M соответственно, а K — матрица размера N×M, имеющая следующую структуру:

$$K = \begin{pmatrix} W & 0 \\ 0 & 0 \end{pmatrix}, \tag{8.5}$$

причем W — матрица размера k×k, где k — ранг исходной матрицы A.

Самое известное из ортогональных, *SVD*- (singular value decomposition) или *сингулярное* разложение — разложение вида $A=U\cdot S\cdot V^T$, где S — диагональная матрица, состоящая из нулей и расположенных на диагонали *сингулярных чисел* матрицы A. Расчет сингулярного разложения в Mathcad осуществляется при помощи пары встроенных функций:

- svds(A) — вектор, состоящий из сингулярных чисел;
- svd(A) — сингулярное разложение:
  - A — действительная матрица.

Пример поиска сингулярного разложения сингулярной матрицы приведен в листинге 8.25, причем его последняя строка представляет собой проверку правильности найденного разложения. Подчеркнем, что вычисленные сингулярные числа находятся на главной диагонали средней матрицы разложения, а ее остальные элементы, по определению, равны нулю. Сравнивая матрицы из листинга 8.25, вы без труда разберетесь, каким образом следует выделять искомые матрицы сингулярного разложения из результата, поставляемого функцией svd.

Листинг 8.25. Сингулярное разложение сингулярной матрицы

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

$B := \mathrm{svd}(A)$

$$B = \begin{pmatrix} -0.215 & -0.887 & 0.408 \\ -0.521 & -0.25 & -0.816 \\ -0.826 & 0.388 & 0.408 \\ -0.48 & 0.777 & 0.408 \\ -0.572 & 0.076 & -0.816 \\ -0.665 & -0.625 & 0.408 \end{pmatrix} \qquad \mathrm{svds}(A) = \begin{pmatrix} 16.848 \\ 1.068 \\ 0 \end{pmatrix}$$

$$U := \mathrm{submatrix}\left(B, 0, \frac{\mathrm{rows}(B) - 1}{2}, 0, \mathrm{cols}(B) - 1\right)$$

$$V := \mathrm{submatrix}\left(B, \frac{\mathrm{rows}(B) + 1}{2}, \mathrm{rows}(B) - 1, 0, \mathrm{cols}(B) - 1\right)$$

$S := \mathrm{diag}(\mathrm{svds}(A))$

$$U = \begin{pmatrix} -0.215 & -0.887 & 0.408 \\ -0.521 & -0.25 & -0.816 \\ -0.826 & 0.388 & 0.408 \end{pmatrix} \qquad V = \begin{pmatrix} -0.48 & 0.777 & 0.408 \\ -0.572 & 0.076 & -0.816 \\ -0.665 & -0.625 & 0.408 \end{pmatrix}$$

$$S = \begin{pmatrix} 16.848 & 0 & 0 \\ 0 & 1.068 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\mathrm{norme}\left|U \cdot S \cdot V^T - A\right| = 2.878 \times 10^{-15}$$

Венцом современных алгоритмов решения произвольных СЛАУ является SVD-разложение. Прежде чем прокомментировать соответствующий алгоритм, приведенный в листинге 8.26, обратим внимание на матрицу S, которая имеет блочную структуру. Согласно формуле (8.5), из матрицы S можно выделить подматрицу W с ненулевыми диагональными элементами. Дальнейшая последовательность действий по построению нормального псевдорешения выглядит так:

1. Нахождение единственного решения вспомогательной СЛАУ: $W \cdot y = U^T \cdot b$ (первые два элемента вектора y во второй строке листинга 8.26, которая учитывает диагональность матрицы S).
2. Дополнение вектора нулевыми элементами до размера искомого вектора x.
3. Вычисление x простым умножением $x = V \cdot y$ (третья строка листинга).

Полученный результат (искомый вектор x и промежуточный y) выведен в конце листинга 8.26. Как вы можете убедиться, он совпадает с ответом, полученным в листинге 8.24 (*см. предыдущий раздел*) при помощи QR-разложения.

Листинг 8.26. Решение вырожденной СЛАУ при помощи сингулярного разложения (продолжение листинга 8.25)

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \qquad b := \begin{pmatrix} 6 \\ 12 \\ 24 \end{pmatrix}$$

$$y := \begin{bmatrix} \dfrac{\left|U^T \cdot b\right|_0}{S_{0,0}} \\ \dfrac{\left|U^T \cdot b\right|_1}{S_{1,1}} \\ 0 \end{bmatrix}$$

$x := V \cdot y$

$$x = \begin{pmatrix} 1.5 \\ 1 \\ 0.5 \end{pmatrix} \qquad y = \begin{pmatrix} -1.624 \\ 0.928 \\ 0 \end{pmatrix}$$

8.4. Собственные векторы и собственные значения матриц

Завершим главу, посвященную решению СЛАУ, еще одной задачей вычислительной линейной алгебры — задачей отыскания собственных векторов `x` и собственных значений λ матрицы `A`, т. е. решения матричного уравнения `A·x=λ·x`. Такое уравнение имеет решения в виде собственных значений `λ1,λ2,...` и соответствующих им собственных векторов `x1,x2,...` Для решения таких задач на собственные векторы и собственные значения в Mathcad встроено несколько функций, реализующих довольно сложные вычислительные алгоритмы:

- `eigenvals(A)` — вычисляет вектор, элементами которого являются собственные значения матрицы `A`;
- `eigenvecs(A)` — вычисляет матрицу, содержащую нормированные собственные векторы, соответствующие собственным значениям матрицы `A`. `n`-й столбец вычисляемой матрицы соответствует собственному вектору `n`-го собственного значения, вычисляемого `eigenvals`;
- `eigenvec(A,λ)` — вычисляет собственный вектор для матрицы `A` и заданного собственного значения λ:
  - `A` — квадратная матрица.

Применение этих функций иллюстрирует листинг 8.27. В его конце приведена проверка правильности нахождения первого из собственных векторов и собственных значений, причем подстановка результата в выражение `A·x=λ·x` проведена дважды — сначала на числовых значениях `x` и λ, а потом путем перемножения соответствующих матричных компонентов.

Листинг 8.27. Собственные векторы и собственные значения матрицы

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

$$\text{eigenvals}(A) = \begin{pmatrix} -0.372 \\ 5.372 \end{pmatrix}$$

$$\text{eigenvecs}(A) = \begin{pmatrix} 0.825 & 0.416 \\ -0.566 & 0.909 \end{pmatrix}$$

$$\text{eigenvec}(A, -0.372) = \begin{pmatrix} 0.825 \\ -0.566 \end{pmatrix}$$

$$-0.372 \cdot \begin{pmatrix} 0.825 \\ -0.566 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0.825 \\ -0.566 \end{pmatrix} = \begin{pmatrix} 10 \times 10^{-5} \\ -4.48 \times 10^{-4} \end{pmatrix}$$

$$\text{eigenvals}(A)_0 \cdot \text{eigenvecs}(A)^{\langle 0 \rangle} - A \cdot \text{eigenvecs}(A)^{\langle 0 \rangle} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Помимо рассмотренной проблемы поиска собственных векторов и значений иногда рассматривают более общую задачу, называемую задачей на *обобщенные собственные значения*: $A \cdot x = \lambda \cdot B \cdot x$. В ее формулировке помимо матрицы `A` присутствует еще одна квадратная матрица `B`. Для задачи на обобщенные собственные значения имеются еще две встроенные функции, действие которых аналогично рассмотренным (листинг 8.28):

- `genvals(A,B)` — вычисляет вектор `v` собственных значений, каждый из которых удовлетворяет задаче на обобщенные собственные значения;
- `genvecs(A,B)` — вычисляет матрицу, содержащую нормированные собственные векторы, соответствующие собственным значениям в векторе `v`, который вычисляется с помощью `genvals`. В этой матрице `i`-й столбец является собственным вектором `x`, удовлетворяющим задаче на обобщенные собственные значения:
  - `A,B` — квадратные матрицы.

Листинг 8.28. Поиск обобщенных собственных векторов и собственных значений

$$A := \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad B := \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

$$\text{genvals}(A, B) = \begin{pmatrix} 2.686 \\ -0.186 \end{pmatrix}$$

$$\text{genvecs}(A, B) = \begin{pmatrix} 0.416 & 0.825 \\ 0.909 & -0.566 \end{pmatrix}$$

$$2.686 \cdot \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \cdot \begin{pmatrix} 0.416 \\ 0.909 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 0.416 \\ 0.909 \end{pmatrix} = \begin{pmatrix} 7.52 \times 10^{-4} \\ -8.52 \times 10^{-4} \end{pmatrix} \blacksquare$$

$$\text{genvals}(A, B)_0 \cdot B \cdot \text{genvecs}(A, B)^{\langle 0 \rangle} - A \cdot \text{genvecs}(A, B)^{\langle 0 \rangle} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

ГЛАВА 9

Обыкновенные дифференциальные уравнения: динамические системы

В этой главе рассматриваются численные методы решений задач с начальными условиями (называемых *задачами Коши*) для *обыкновенных дифференциальных уравнений* (далее используется сокращение *ОДУ*). Такие задачи требуют нахождения функции (или нескольких функций) одной переменной, если, во-первых, определено дифференциальное уравнение (или система уравнений), содержащее производную функции, и, во-вторых, необходимое количество дополнительных условий, задающих значение функции в некоторой начальной точке.

Решение задач Коши для ОДУ — давно и детально разработанная технология. С "хорошими" ОДУ вообще никаких вычислительных проблем обычно не возникает (чаще всего они решаются при помощи алгоритма Рунге—Кутты), а для ОДУ особого типа, называемых жесткими, необходимо применять специальные методы. Все эти возможности заложены в Mathcad, причем пользователю позволено выбирать конкретный алгоритм решения ОДУ.

9.1. О постановке задач

Начнем разговор об обыкновенных дифференциальных уравнениях с формулировки типичных задач, сопровождая вопросы их постановки конкретными примерами. При этом мы будем, несколько забегая вперед, приводить их решение, не обсуждая особенностей его отыскания средствами Mathcad.

9.1.1. Задачи Коши для ОДУ

Дифференциальные уравнения — это уравнения, в которых неизвестными являются не переменные (т. е. числа), а функции одной или нескольких переменных. Эти уравнения (или системы) включают соотношения между искомыми функциями и их производными. Если в уравнения входят производные только по одной переменной, то они называются *обыкновенными дифференциальными.* В противном случае говорят об *уравнениях в частных производных* (*см. главу 11*). Таким образом,

решить (иногда говорят *проинтегрировать*) дифференциальное уравнение — значит, определить неизвестную функцию на определенном интервале изменения ее переменных.

ОДУ с неизвестной функцией `y(t)`, в которое входят производные этой функции вплоть до $y^{(N)}(t)$, называется *ОДУ N-го порядка*. В частности, уравнение первого порядка может по определению содержать помимо самой искомой функции `y(t)` только ее первую производную `y'(t)`, второго порядка — `y'(t)` и `y''(t)` и т. д. В подавляющем большинстве практических случаев дифференциальное уравнение можно записать в *стандартной форме* (*форме Коши*): `y'(t)=f(y(t),t)`. Уравнение второго порядка может содержать, помимо самой функции, ее первую и вторую производные и т. д. Листинг 9.1 демонстрирует решение простого ОДУ второго порядка, описывающего модель затухающего гармонического осциллятора. Само уравнение приведено во второй строке листинга, после задания параметров модели, а вычисленный результат, т. е. искомая функция `y(t)`, показан на рис. 9.1.

Примечание

Модель гармонического осциллятора описывает, в частности, колебания маятника: `y(t)` описывает изменения угла его отклонения от вертикали, `y'(t)` — угловую скорость маятника, `y''(t)` — ускорение, а начальные условия, соответственно, начальное отклонение маятника `y(0)=1.0` и начальную скорость `y'(0)=0`. Важно отметить, что модель является линейной, т. е. неизвестная функция и ее производные входят в уравнение в первой степени.

Листинг 9.1. Решение задачи Коши для ОДУ второго порядка (модель затухающего гармонического осциллятора)

$\omega := 0.5 \qquad \beta := 0.2$

Given $\qquad \omega^2 \cdot \frac{d^2}{dt^2} y(t) + \beta \cdot \frac{d}{dt} y(t) + y(t) = 0$

$y'(0) = 0$

$y(0) = 1.0$

$y := \mathrm{Odesolve}(t, 10)$

Примечание

В последних версиях Mathcad имеется возможность выбрать численный алгоритм, применяемый функцией `Odesolve` для решения ОДУ. Для этого следует щелкнуть на имени функции `Odesolve` правой кнопкой мыши (*см. разд. 9.5*).

Как показывает листинг 9.1, помимо самого уравнения потребовалось определить два начальных условия (третья и четвертая строки листинга) — начальные значения `y(t)` и `y'(t)` при `t=0`. Вообще говоря, ОДУ (или система ОДУ) имеет единственное решение, если помимо уравнения определенным образом заданы *начальные* или *граничные* условия. В соответствующих курсах высшей математики доказываются теоремы о существовании и единственности решения в зависимости от тех или иных условий.

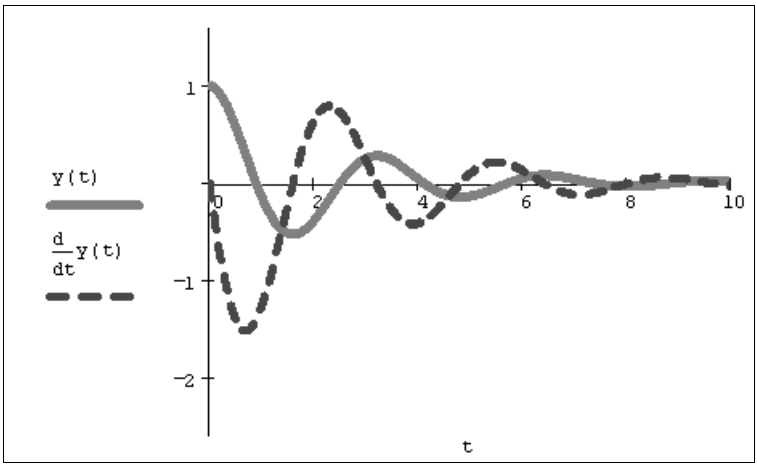


Рис. 9.1. Решение уравнения $\omega^2 \cdot y'' + \beta \cdot y' + y = 0$ (продолжение листинга 9.1)

Имеются два типа задач, которые возможно решать с помощью Mathcad:

- *задачи Коши* — для которых определены начальные условия на искомые функции, т. е. заданы значения этих функций в начальной точке интервала интегрирования уравнения (именно им и посвящена данная глава);
- *краевые задачи* — для которых заданы определенные соотношения сразу на обеих границах интервала (они рассматриваются в *главе 10*).

Примечание

На компакт-диске, прилагаемом к данной книге, вы найдете несколько примеров решения классических динамических систем (таких, как модели Лоренца, Ван-дер-Поля, Рикитаке и др.).

Сказанное относится как к отдельным дифференциальным уравнениям (*см. разд. 9.2*), так и к системам ОДУ (*см. разд. 9.3*). Важно подчеркнуть, что Mathcad умеет решать только такие системы дифференциальных уравнений, которые могут быть представлены в стандартной форме (форме Коши). Для неизвестных функций $y_0(t), y_1(t), \ldots, y_{N-1}(t)$ система ОДУ должна быть записана в форме:

$$\begin{gathered} y_0'(t)=f_0(y_0(t),y_1(t),\ldots,y_{N-1}(t),t); \\ y_1'(t)=f_1(y_0(t),y_1(t),\ldots,y_{N-1}(t),t); \\ \ldots \\ y_{N-1}'(t)=f_{N-1}(y_0(t),y_1(t),\ldots,y_{N-1}(t),t), \end{gathered} \tag{9.1}$$

что эквивалентно следующему векторному представлению:

$$Y'(t)=F(Y(t),t). \tag{9.2}$$

Здесь `Y` и `Y'` — соответствующие неизвестные векторные функции переменной `t` размера `N×1`, а `F` — векторная функция того же размера и количества переменных `(N+1)` (`N` компонент вектора и, возможно, `t`). Именно векторное представление (9.2) используется для ввода системы ОДУ в среде Mathcad.

Для того чтобы определить задачу Коши для системы из N ОДУ первого порядка, следует определить еще ровно N начальных условий, задающих значение каждой из функций $y_i(t0)$ в начальной точке интегрирования системы t0. В векторной форме они могут быть записаны в виде

$$Y(t0)=B, \tag{9.3}$$

где B — вектор начальных условий размера N×1, составленный из $y_i(t0)$. Обратите внимание на необходимость векторной записи как самого уравнения, так и начального условия. В случае одного ОДУ первого порядка соответствующие векторы имеют только один элемент, а в случае системы N>1 уравнений — N.

Стандартные процедуры Mathcad применимы для систем ОДУ первого порядка, записанных в форме (9.2)—(9.3). Тем не менее если в систему входят и уравнения высших порядков, то ее можно свести к системе большего числа уравнений первого порядка. Рассмотрим в качестве примера уравнение второго порядка модели осциллятора $\omega^2 \cdot y''+\beta \cdot y'+y=0$, решенное в листинге 9.1. Если ввести обозначение $y_0(t)\equiv y(t)$, $y_1(t)\equiv y'(t)$, то уравнение сведется к эквивалентной системе:

$$y_0(t)=y_1(t);$$

$$\omega^2 \cdot y_1'+\beta \cdot y'+y=0,$$

форма которой удовлетворяет форме (9.2), и уравнение может быть решено средствами Mathcad, предназначенными для систем ОДУ. Именно эта система решена ниже в листинге 9.3 (*см. разд. 9.3.1*). Отметим, что на рис. 9.1 показаны как зависимость y(t), так и $y_1(t)=y'(t)$.

9.1.2. Фазовый портрет динамической системы

Модели, основанные на задачах Коши для ОДУ, часто называют *динамическими системами*, подчеркивая, что, как правило, они содержат производные по времени t и описывают динамику некоторых параметров. Проблемы, связанные с динамическими системами, на самом деле весьма разнообразны и зачастую не сводятся к простому интегрированию ОДУ. Некоторые из них мы обозначим в данном разделе, отметив, что для изучения динамических систем центральным моментом является анализ фазовых портретов, т. е. решений, получающихся при выборе всевозможных начальных условий.

Решение ОДУ часто удобнее изображать не в виде графика $y_0(t), y_1(t), \ldots$, как на рис. 9.1, а в *фазовом пространстве*, по каждой из осей которого откладываются значения каждой из найденных функций. При таком построении графика аргумент t будет присутствовать на нем лишь параметрически. В рассматриваемом случае двух ОДУ (мы свели к ним в предыдущем разделе дифференциальное уравнение осциллятора второго порядка) фазовое пространство является координатной плоскостью, а решение представляет собой кривую, или, по-другому, *траекторию*, выходящую из точки, координаты которой равны начальным условиям (рис. 9.2). В общем случае, если система состоит из N ОДУ, то фазовое пространство является

N-мерным. При N>3 наглядность теряется, и для визуализации фазового пространства приходится строить его различные проекции или прибегать к другим специальным приемам (например, отображению Пуанкаре).

Как правило, решение задач Коши для ОДУ и их систем — задача хорошо разработанная и с вычислительной точки зрения довольно простая. На практике чаще встречаются другие, более сложные задачи, в частности исследование поведения динамической системы в зависимости от начальных условий. При этом в большинстве случаев бывает необходимым изучить только асимптотическое решение ОДУ, т. е. $y(t\to\infty)$, называемое *аттрактором*. Очень наглядным образом можно визуализировать такую информацию на фазовой плоскости, во многом благодаря тому, что существует всего несколько типов аттракторов, и для них можно построить четкую классификацию.

С одной стороны, каждое решение будет выходить из точки, координаты которой являются начальными условиями, но, оказывается, для большинства ОДУ целые семейства траекторий будут заканчиваться в одних и тех же аттракторах (стационарных точках или предельных циклах). Множество решений, вычисленное для всевозможных начальных условий, образует *фазовый портрет* динамической системы. С вычислительной точки зрения задача исследования фазового портрета часто сводится к обычному *сканированию* семейств решений ОДУ при разных начальных условиях.

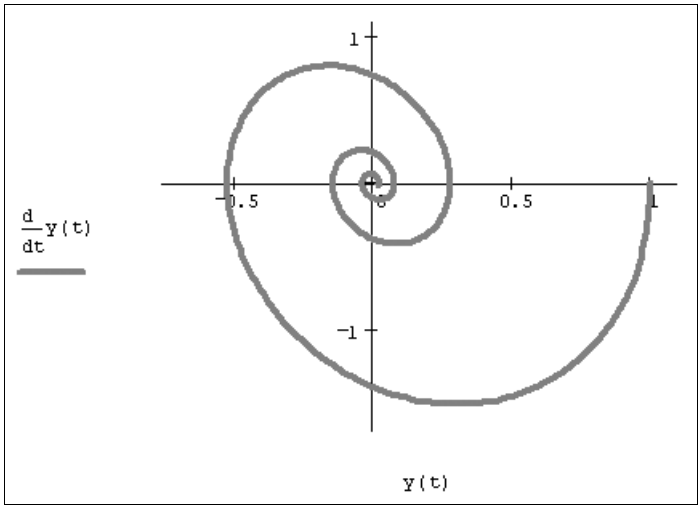


Рис. 9.2. Решение уравнения $\omega^2\cdot y''+\beta\cdot y'+y=0$ на фазовой плоскости (продолжение листинга 9.1)

Примечание

Для рассматриваемого примера модели гармонического осциллятора имеется единственная стационарная точка (аттрактор), на которую "накручивается" решение, из каких бы начальных условий оно ни выходило. В теории динамических систем аттрактор такого типа называется *фокусом*.

Дальнейшее усложнение задач анализа фазовых портретов связано с их зависимостью от параметров, входящих в систему ОДУ. В частности, при плавном изменении параметра модели может меняться расположение аттракторов на фазовой плоскости, а также могут возникать новые аттракторы и прекращать свое существование старые. В первом случае, при отсутствии особенностей, будет происходить простое перемещение аттракторов по фазовой плоскости (без изменения их типов и количества), а во втором — фазовый портрет динамической системы будет коренным образом перестраиваться. Критическое сочетание параметров, при которых фазовый портрет системы качественно меняется, называется в теории динамических систем точкой *бифуркации*.

Поясним понятие бифуркации на примере той же модели осциллятора, которая зависит от двух параметров (ω и β). При $\beta>0$ существует единственная стационарная точка типа фокуса (см. рис. 9.2), которая в точке бифуркации $\beta=0$ вырождается в аттрактор типа *центр*, характеризующийся тем, что решения ОДУ представляют собой циклы, совершаемые вокруг этой точки с амплитудой, которая существенно зависит от начальных условий (рис. 9.3). Для надежного исследования фазового портрета практически всегда необходимо решить систему ОДУ большое количество раз с самыми разными начальными условиями (и, возможно, с разным набором параметров модели), чтобы посмотреть, к каким аттракторам сходятся различные траектории.

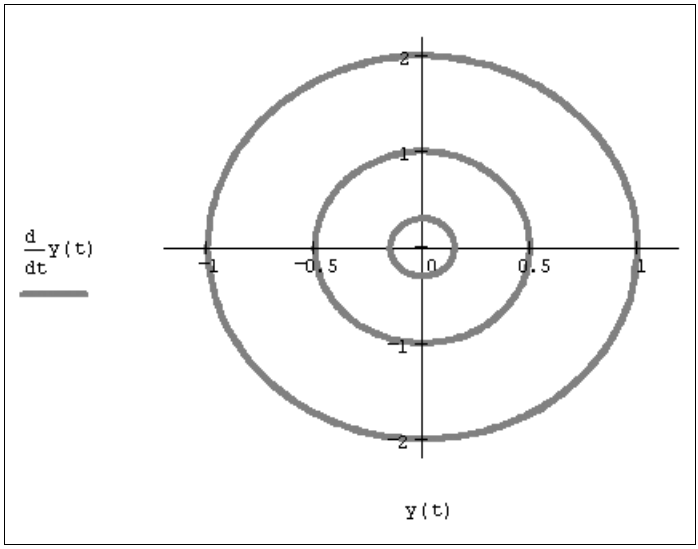


Рис. 9.3. Решение уравнения $\omega^2 \cdot y''+y=0$ для различных начальных условий (коллаж графиков)

Резюмируя содержание вводного раздела главы, перечислим еще раз типичные постановки задач, характерные для динамических систем:

- решение одной задачи Коши для ОДУ;
- исследование фазового портрета (поиск аттракторов);

- отыскание зависимости положения аттракторов в фазовом пространстве от параметров модели и фиксация бифуркационных значений параметров.

В дальнейших разделах этой главы при рассказе о возможностях Mathcad мы будем в первую очередь описывать решение первой (базовой) задачи, для которой предусмотрен целый арсенал средств. А именно: вычислительный блок для решения ОДУ (*см. разд. 9.2*), несколько встроенных функций для решения систем ОДУ (*см. разд. 9.3*), в том числе *жестких*, которые не поддаются решению стандартными методами (*см. разд. 9.4*).

9.2. Дифференциальное уравнение *N*-го порядка

Для решения ОДУ порядка `N≥1` в Mathcad предусмотрены две возможности:

- блок решения (вычислительный блок) `Given/Odesolve` — в этом случае решение имеет вид функции от `t`;
- встроенные функции решения систем ОДУ, причем уравнения высших порядков необходимо предварительно свести к эквивалентной системе уравнений первого порядка, как об этом рассказано в *разд. 9.1.1*, — в этом случае решение имеет формат вектора.

В данном разделе рассматривается первый из вариантов, а второй будет описан в следующем, применительно к общему случаю системы `N` уравнений. Заметим лишь, что первый путь предпочтительнее из соображений наглядности представления задачи и результатов, а второй дает пользователю больше рычагов воздействия на параметры численного метода.

Вычислительный блок для решения ОДУ, реализующий численный метод Рунге—Кутты, состоит из трех частей:

- `Given` — ключевое слово;
- ОДУ и начальные условия в формате `y(t0)=b`, записанные с помощью логических операторов, которые должны набираться на панели инструментов **Булева алгебра** (Boolean);
- `Odesolve(t,t1)` — встроенная функция для решения ОДУ относительно переменной `t` на интервале `(t0,t1)`, причем `t0<t1`.

Примечание 1

На запись ОДУ в пределах вычислительного блока накладывается несколько ограничений. Во-первых, ОДУ должно быть линейно относительно старшей производной, т. е. фактически должно быть поставлено в стандартной форме. Во-вторых, начальные условия должны иметь форму `y(t0)=b` или $y^{(N)}(t0)=b$, а не более сложную (как, например, встречающаяся в некоторых математических приложениях форма `y(t0)+y'(t0)=b`).

Примечание 2

Допустимо, и даже часто предпочтительнее, задание функции `Odesolve(t,t1,step)` с тремя параметрами, где `step` — внутренний параметр численного метода, определяю-

щий количество шагов, в которых метод Рунге—Кутты будет рассчитывать решение дифференциального уравнения. Чем больше `step`, тем с лучшей точностью будет получен результат, но тем больше времени будет затрачено на его поиск. Помните, что подбором этого параметра можно заметно (в несколько раз) ускорить расчеты без существенного ухудшения их точности.

Пример решения задачи Коши для ОДУ второго порядка, являющегося усложнением модели из листинга 9.1 на нелинейный случай (с параметром квадратичной нелинейности γ), приведен в листинге 9.2, а результат — на рис. 9.4. Учтите, что символ производной допускается вводить как средствами панели **Математический анализ** (Calculus), как это сделано в листинге 9.1, так и в виде штриха, набрав его с помощью сочетания клавиш <Ctrl>+<F7> (как в листинге 9.2). Выбирайте тот или иной способ представления производной из соображений наглядности представления результатов — на ход расчетов он не влияет.

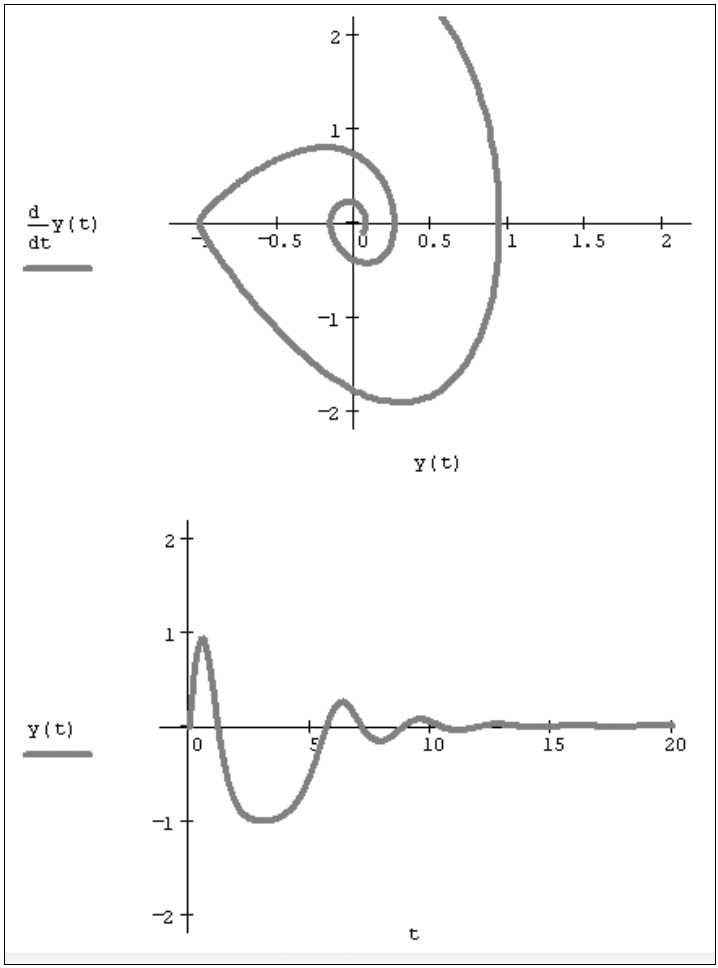


Рис. 9.4. Решение уравнения $\omega^2 \cdot y'' + \beta \cdot y' + y + \gamma \cdot y^2 = 0$
(продолжение листинга 9.2)

Еще раз подчеркнем, что результатом применения блока `Given/Odesolve` является функция `y(t)`, определенная на промежутке `(t0,t1)`. Следует воспользоваться обычными средствами Mathcad, чтобы построить ее график или получить значение функции в какой-либо точке указанного интервала, например: `y(10)=0.048`.

Листинг 9.2. Решение задачи Коши для ОДУ второго порядка (модель нелинейного осциллятора)

$\omega := 0.5 \quad \beta := 0.2 \quad \gamma := 0.95$

Given

$\omega^2 \cdot y''(t) + \beta \cdot y'(t) + y(t) + \gamma \cdot y(t)^2 = 0$

$y(0) = 0 \qquad y'(0) = 3$

$y := \mathrm{Odesolve}(t, 20)$

Пользователь имеет возможность выбирать между двумя модификациями численного метода Рунге—Кутты. Для смены метода необходимо нажатием правой кнопки мыши на области функции `Odesolve` вызвать контекстное меню и выбрать в нем один из трех пунктов: **С фиксированным шагом** (Fixed), **Адаптивный** (Adaptive) или **Для жестких ОДУ** (Stiff).

9.3. Система *N* дифференциальных уравнений

При помощи Mathcad можно решать системы `N≥1` ОДУ первого порядка, если они записаны в стандартной форме (Коши) в виде векторного соотношения: `Y'(t)=F(Y(t),t)` (*см. разд. 9.1.1*).

9.3.1. Встроенные функции для решения систем ОДУ

В Mathcad имеется несколько встроенных функций, которые позволяют решать задачу Коши различными численными методами. Для "хороших" нежестких систем ОДУ применяются следующие функции:

- `rkfixed(y0,t0,t1,M,D)` — метод Рунге—Кутты с фиксированным шагом;
- `Rkadapt(y0,t0,t1,M,D)` — метод Рунге—Кутты с переменным шагом;
- `Bulstoer(y0,t0,t1,M,D)` — метод Булирша—Штера;
  - `y0` — вектор начальных значений в точке `t0` размера `N×1`;
  - `t0` — начальная точка расчета;
  - `t1` — конечная точка расчета;
  - `M` — число шагов, на которых численный метод находит решение;

- D — векторная функция размера N×1 двух аргументов — скалярного t и векторного y. При этом y — искомая векторная функция аргумента t того же размера N×1.

Примечание 1

Соблюдайте регистр первой буквы рассматриваемых функций, поскольку это влияет на выбор алгоритма счета, в отличие от многих других встроенных функций Mathcad, например, Find≡find (*см. разд. 9.3.2*).

(14) Примечание 2

Несколько новых функций для решения жестких и нежестких ОДУ, основанных на более современных численных алгоритмах, появились в Mathcad 14 (*см. разд. 9.5*).

Каждая из приведенных функций выдает решение в виде матрицы размера (M+1)×(N+1). В ее левом столбце находятся значения аргумента t, делящие интервал на равномерные шаги, а в остальных N столбцах — значения искомых функций $y_0(t), y_1(t), \ldots, y_{N-1}(t)$, рассчитанные для этих значений аргумента (рис. 9.5). Поскольку всего точек (помимо начальной) M, то строк в матрице решения будет всего M+1.

В листинге 9.3 приведен пример решения той же самой системы ОДУ осциллятора с затуханием (*см. разд. 9.1 и 9.2*) при помощи первой из функций rkfixed. Результат расчетов представлен на рис. 9.5 как содержимое матрицы и на рис. 9.6 в виде графика. Точки, в которых получено решение, отмечены на графике рис. 9.6 кружками. Чтобы использовать другой численный алгоритм, достаточно поменять имя функции rkfixed в последней строке листинга на другое (на практике как раз более эффективны функции Rkadapt и Bulstoer).

Листинг 9.3. Решение системы двух ОДУ (модель осциллятора)

$\omega := 0.5 \qquad \beta := 0.2$

$$y0 := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$M := 50$

$$D(t, y) := \begin{pmatrix} y_1 \\ -\omega \cdot y_0 - \beta \cdot y_1 \end{pmatrix}$$

$u := \mathrm{rkfixed}(y0, 0, 40, M, D)$

Первая строка листинга представляет задание параметров модели, вторая — начального условия задачи Коши, а в третьей строке листинга определено число шагов, на которых рассчитывается решение. Самая важная — это предпоследняя строка листинга, в которой, собственно, определяется система ОДУ. Последняя строка присваивает матричной переменной u результат действия функции rkfixed. Решение системы ОДУ будет осуществлено на промежутке (0,40).

Сравните рассматриваемую систему (*см. разд. 9.1.1*), записанную в стандартной форме с формальной ее записью в Mathcad, чтобы не делать впоследствии ошибок. Во-первых, функция `D`, входящая в число параметров встроенных функций для решения ОДУ, должна быть функцией обязательно двух аргументов. Во-вторых, второй ее аргумент должен быть вектором того же размера, что и сама функция `D`. В-третьих, точно такой же размер должен быть и у вектора начальных значений `y0`. Не забывайте, что векторную функцию `D(t,y)` следует определять через компоненты вектора `y` с помощью кнопки **Индекс** (Subscript) с наборной панели **Матрица** (Matrix) или нажатием клавиши <[>.

u =

	0	1	2
0	0	1	0
1	0.8	0.852	-0.35
2	1.6	0.482	-0.548
3	2.4	0.027	-0.559
4	3.2	-0.368	-0.408
5	4	-0.599	-0.162
6	4.8	-0.624	0.095
7	5.6	-0.466	0.286
8	6.4	-0.197	0.367
9	7.2	0.089	0.33
10	8	0.307	0.204
11	8.8	0.404	0.038
12	9.6	0.371	-0.114
13	10.4	0.236	-0.211
14	11.2	0.054	-0.233
15	12	-0.118	-0.185

Рис. 9.5. Результат, выдаваемый встроенной функцией в качестве решения системы ОДУ (продолжение листинга 9.3)

Матрица, представляющая решение, показана на рис. 9.5. Размер полученной матрицы будет равен `(M+1)×(N+1)`, т. е. `51×3`. Просмотреть все компоненты матрицы `u`, которые не помещаются на экране, можно с помощью вертикальной полосы прокрутки. Как нетрудно сообразить, на этом рисунке отмечены выделением расчетные значения искомых векторов на 10-м шаге. Это соответствует, с математической точки зрения, найденным значениям $y_0(8.0)=0.307$ и $y_1(8.0)=0.204$. Для вычисления элементов решения в последней точке интервала используйте вывод элемента $u_{M,1}$. Для построения графика решения надо отложить соответствующие компоненты матрицы решения по координатным осям: значения аргумента $u^{<0>}$ — вдоль оси `X`, а $u^{<1>}$ и $u^{<2>}$ — вдоль оси `Y` (рис. 9.6).

Внимание!

Обратите внимание на некоторое разночтение в обозначении индексов вектора начальных условий и матрицы решения. В ее первом столбце собраны значения нулевой компоненты искомого вектора, во втором столбце — первой компоненты и т. д.

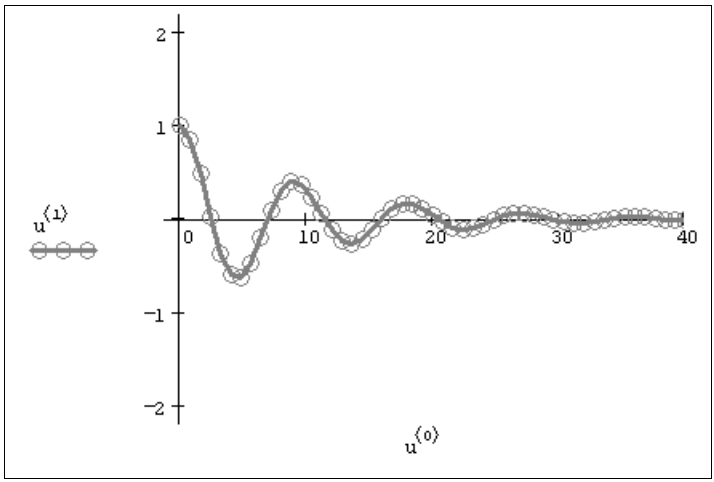


Рис. 9.6. График решения системы ОДУ осциллятора (продолжение листинга 9.3)

9.3.2. Решение одного уравнения (*N*=1)

Метод решения ОДУ при помощи встроенных функций `rkfixed`, `Rkadapt` или `Bulstoer` (в противоположность вычислительному блоку `Given/Odesolve`) сохранился с прежних версий Mathcad (до 2000-й). В большинстве случаев лучше использовать вычислительный блок `Given/Odesolve`, который выигрывает в простоте и в наглядности, однако иногда предпочтительнее решать ОДУ первого порядка с помощью второго способа, например, при следующих обстоятельствах:

- вы работаете одновременно с более старыми (до 2001-й включительно) версиями Mathcad и хотите, чтобы ваши документы воспринимались каждой из них корректно;
- одно ОДУ решается в контексте решения более сложных задач, в которые входят системы дифференциальных уравнений (для которых вычислительный блок неприменим) — в этом случае может потребоваться единый стиль программирования;
- ответ предпочтительнее получить в виде вектора, а не функции;
- вы привыкли к записи ОДУ в старых версиях Mathcad, у вас много документов, созданных с их помощью, и т. п.

Поскольку решение вторым способом одного ОДУ не отличается от решения систем ОДУ (*см. предыдущий раздел*), приведем пример его использования (листинг 9.4) практически без комментариев. Отметим лишь в случае одного ОДУ, что как само уравнение, так и начальное условие можно задавать не в векторной, а в скалярной форме. Результат выдается в виде матрицы размерности `M×2`, которая состоит из двух столбцов: в одном находятся значения аргумента `t` (от `t0` до `t1` включительно), а в другом соответствующие значения искомой функции `y(t)`.

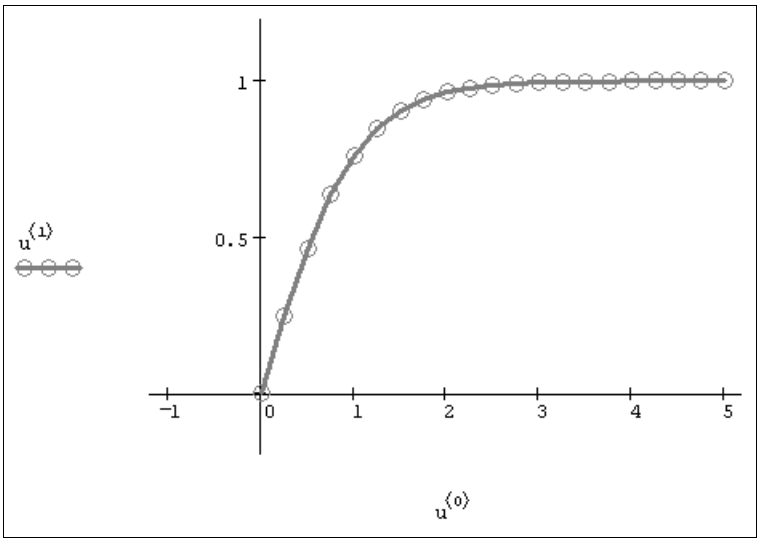


Рис. 9.7. Решение уравнения $y'=1-y^2$
(продолжение листинга 9.4)

Построение графика (рис. 9.7) осуществляется так же, как и в рассмотренном в предыдущем разделе случае N уравнений, при помощи выделения столбцов из матрицы решения посредством оператора <>.

Листинг 9.4. Решение задачи Коши для ОДУ первого порядка

```
y0 := 0
M := 20
D(t, y) := 1 − y^2
u := rkfixed(y0, 0, 5, M, D)
```

9.3.3. Решение систем ОДУ в одной заданной точке

Зачастую при решении дифференциальных уравнений требуется определить значения искомых функций не на всем интервале (t0,t1), а только в одной его последней точке. Например, весьма распространены задачи поиска аттракторов динамических систем. Известно, что для широкого класса ОДУ одна и та же система при разных (или даже любых, как рассмотренный выше пример осциллятора с затуханием) начальных условиях при $t\to\infty$ приходит в одну и ту же точку (аттрактор). Поэтому часто нужно определить именно эту точку.

Такая задача требует меньше ресурсов компьютера, чем решение системы ОДУ на всем интервале, поэтому в Mathcad имеются модификации встроенных функций Rkadapt и Bulstoer. Они имеют несколько другой набор параметров и работают значительно быстрее своих аналогов:

- `rkadapt(y0,t0,t1,acc,D,k,s)` — метод Рунге—Кутты с переменным шагом;
- `bulstoer(y0,t0,t1,acc,D,k,s)` — метод Булирша—Штера:
  - `y0` — вектор начальных значений в точке `t0`;
  - `t0`,`t1` — начальная и конечная точки расчета;
  - `acc` — погрешность вычисления (чем она меньше, тем с лучшей точностью будет найдено решение; рекомендуется выбирать значения погрешности в районе `0.001`);
  - `D` — векторная функция, задающая систему ОДУ;
  - `k` — максимальное число шагов, на которых численный метод будет находить решение;
  - `s` — минимально допустимая величина шага.

Как легко заметить, вместо числа шагов на интервале интегрирования ОДУ в этих функциях необходимо задать точность расчета численным методом значения функций в последней точке. В этом смысле параметр `acc` похож на константу `TOL`, которая влияет на большинство встроенных численных алгоритмов Mathcad. Количество шагов и их расположение определяются численным методом автоматически, чтобы обеспечить эту точность. Два последних параметра нужны для того, чтобы пользователь мог искусственно повлиять на разбиение интервала на шаги. Параметр `k` служит для того, чтобы шагов не было чрезмерно много, причем нельзя сделать `k>1000`. Параметр `s` — для того, чтобы ни один шаг не был слишком малым для появления больших погрешностей при разностной аппроксимации дифференциальных уравнений внутри алгоритма. Эти параметры следует задавать явно, исходя из свойств конкретной системы ОДУ. Как правило, проведя ряд тестовых расчетов, можно подобрать их оптимальный набор для каждого конкретного случая.

Пример использования функции `bulstoer` для той же модели линейного осциллятора приведен в листинге 9.5. В его первых двух строках, как обычно, определяется система уравнений и начальные условия; в следующей строке матрице `u` присваивается решение, полученное с помощью `bulstoer`. Структура этой матрицы в точности такая же, как и в случае решения системы ОДУ посредством уже рассмотренных нами ранее встроенных функций (*см. разд. 9.3.1*). Однако в данном случае нам интересна только последняя точка интервала. Поскольку сделанное численным методом количество шагов, т. е. размер матрицы `u`, заранее неизвестно, то его необходимо предварительно определить. Это сделано в четвертой строке листинга, присваивающей это число переменной `M`, в этой же строке оно выведено на экран.

В предпоследней строке листинга осуществлен вывод решения системы ОДУ на конце интервала, т. е. в точке `t=50` в виде вектора. В последней строке для примера еще раз выводится искомое значение первой функции из системы ОДУ (сравните его с соответствующим местом вектора из предыдущей строки). Чтобы попробовать альтернативный численный метод, достаточно в листинге 9.5 заменить имя функции `bulstoer` на `rkadapt`.

Листинг 9.5. Поиск аттрактора системы двух ОДУ модели осциллятора

$$y0 := \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$M := 50$$

$$D(t, y) := \begin{pmatrix} y_1 \\ -0.5 \cdot y_0 - 0.2 \cdot y_1 \end{pmatrix}$$

$$u := \mathrm{bulstoer}\left(y0, 0, 300, 10^{-5}, D, M, 0.01\right)$$

$$N := \mathrm{rows}(u) - 1$$

$$\left(u^T\right)^{\langle N \rangle} = \begin{pmatrix} 300 \\ -7.645 \times 10^{-14} \\ -3.128 \times 10^{-14} \end{pmatrix}$$

Внимание!

Функции `bulstoer` и `rkadapt` (те, что пишутся со строчной буквы) не предназначены для нахождения решения в промежуточных точках интервала, хотя они и выдают их в матрице-результате.

9.3.4. О численных методах

В завершение раздела сделаем несколько важных замечаний относительно выбора численного алгоритма решения ОДУ и задания его параметров. Они не претендуют на общность, но, надеемся, будут весьма полезны читателю, особенно в случае возникновения проблем.

Рекомендации по выбору численного алгоритма

Все численные методы решения ОДУ основаны на аппроксимации дифференциальных уравнений разностными аналогами. В зависимости от конкретной формы аппроксимации получаются алгоритмы различной точности и быстродействия. В Mathcad использован наиболее популярный алгоритм Рунге—Кутты четвертого порядка, описанный в большинстве книг по методам вычислений. Он обеспечивает малую погрешность для широкого класса систем ОДУ за исключением жестких систем. Поэтому в большинстве случаев стоит применять функцию `rkfixed`. Если по различным причинам время расчетов становится критичным или точность неудовлетворительна, стоит попробовать вместо `rkfixed` другие функции, тем более что сделать это очень просто благодаря одинаковому набору параметров. Для этого нужно только поменять имя функции в программе.

Функция `Rkadapt` может быть полезна в случае, когда известно, что решение на рассматриваемом интервале меняется слабо либо существуют участки медленных и быстрых его изменений. Метод Рунге—Кутты с переменным шагом разбивает интервал не на равномерные шаги, а более оптимальным способом. Там, где решение

меняется слабо, шаги выбираются более редкими, а в областях его сильных изменений — частыми. В результате для достижения одинаковой точности требуется меньшее число шагов, чем для `rkfixed`. Метод Булирша—Штера `Bulstoer` часто оказывается более эффективным для поиска гладких решений.

Имея в виду сделанные замечания, приведем краткую сводку алгоритмов решения задач Коши для ОДУ, отмечая, какие из встроенных функций следует использовать в конкретных случаях.

- Если вы пользуетесь Mathcad 14, применяйте универсальную функцию `AdamsBDF`, которая способна автоматически выбирать жесткий или нежесткий численный алгоритм.
- Для решения единственного уравнения (любого порядка) используйте вычислительный блок `Given/Odesolve`. Помните о том, что в последних версиях Mathcad допускается выбирать для этого блока численный алгоритм (в том числе жесткий) из контекстного меню.
- Для стандартных нежестких систем используйте алгоритм Булирша—Штера `Bulstoer`.
- Для систем с участками быстро и медленно меняющихся решений используйте адаптивный алгоритм Рунге—Кутты `Rkadapt`.
- В учебных целях и для решения несложных задач можно использовать алгоритм Рунге—Кутты с фиксированным шагом `Rkfixed`.
- Для получения решения в одной конечной точке интервала используйте (в зависимости от перечисленных классов задач) одну из встроенных функций с именем, начинающимся со строчной буквы.
- Для жестких систем (*см. разд. 9.4*) используйте функции `Radau` `Stiffb` или `Stiffr`, причем, если вы затрудняетесь с вводом якобиана, применяйте первую из них.

Мы не будем в этой книге приводить описание численных алгоритмов, поскольку им посвящено огромное количество книг, и читатель, несомненно, найдет соответствующую литературу с подробным объяснением их работы. Тем не менее разберем несколько важных моментов, касающихся выбора алгоритма и его параметров.

ⓘ Число шагов

Все встроенные функции требуют задания для численного метода количества шагов, разбивающих интервал интегрирования ОДУ. Это очень важный параметр, непосредственно влияющий на погрешность результата и скорость расчетов. При решении систем ОДУ многие проблемы могут быть устранены при помощи простой попытки увеличить число шагов численного метода. В частности, сделайте так при неожиданном возникновении ошибки "Found a number with a magnitude greater than 10^307" (Найдено число, превышающее значение 10^{307}). Данная ошибка не обязательно говорит о том, что решение в действительности расходится, а возникает из-за недостатка шагов для корректной работы численного алгоритма.

В качестве иллюстрации приведем результаты простых расчетов (листинг 9.6), в которых определяется пользовательская функция `u(M)`, представляющая решение ОДУ в зависимости от числа шагов `M`. На рис. 9.8 показано несколько графиков решения системы ОДУ затухающего линейного осциллятора для нескольких значений `M`.

Как можно заметить, несмотря, в целом, на правильное решение ОДУ в узлах (оно показано кружками и квадратами), профиль решения при малых `M` получается не-

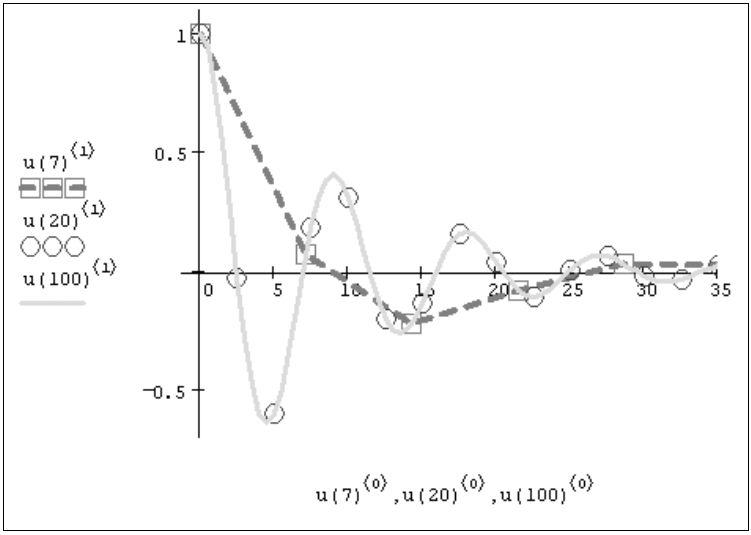


Рис. 9.8. Решение системы ОДУ в зависимости от числа узлов `M` (продолжение листинга 9.6)

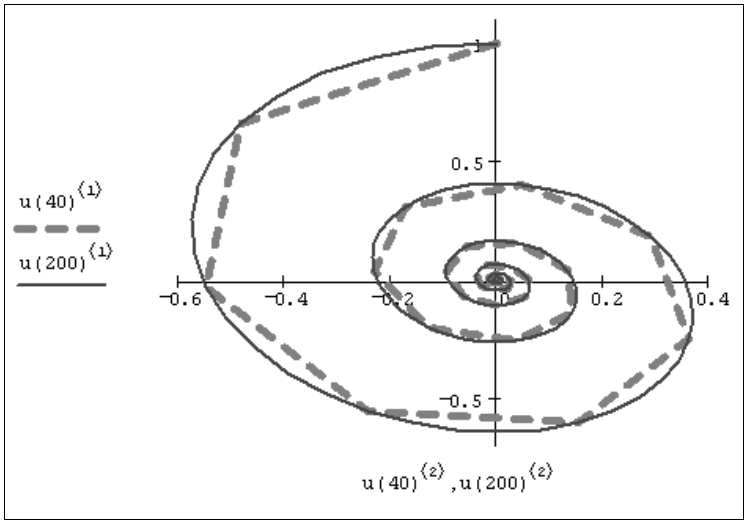


Рис. 9.9. Фазовый портрет решения системы ОДУ при `M=40` и `M=200` (продолжение листинга 9.6)

верным. Еще лучше это видно на рис. 9.9, представляющем фазовый портрет системы с разным числом шагов. Видно, что для `M=200` решение получается более гладким (конечно, при этом увеличивается и время расчетов).

Листинг 9.6. Решение системы ОДУ как функция числа шагов

$$D(t,y) := \begin{bmatrix} y_1 \\ -0.5\cdot|y_0| - 0.2\cdot y_1 \end{bmatrix}$$

$$u(M) := \mathrm{Bulstoer}\left[\begin{pmatrix}1\\0\end{pmatrix}, 0, 50, M, D\right]$$

Погрешность алгоритмов решения ОДУ в точке

Обратимся теперь к функциям `bulstoer` и `rkadapt` (тем, что пишутся со строчной буквы), которые предназначены для нахождения решения в определенной точке интервала интегрирования системы ОДУ. В первую очередь, подчеркнем еще раз, что их нельзя применять для получения решения в промежуточных точках интервала, несмотря на их вывод в матрице-результате. На рис. 9.10 показан фазовый портрет системы ОДУ динамической модели осциллятора, полученный посредством функции `bulstoer` в листинге 9.5 (*см. разд. 9.3.3*) и с помощью `rkadapt` (при соответствующей замене третьей строки листинга 9.5). Видно, что, несмотря на высокую точность, равную 10^{-5}, и верный результат на конце интервала, график, полученный при помощи функции `bulstoer` (рис. 9.10, *а*), мало напоминает правильный фазовый портрет (см. рис. 9.9), начиная быть приемлемым только при предельно допустимом для обсуждаемых функций значении $acc=10^{-16}$. В данном случае конкретной системы ОДУ функция `rkadapt` дает качественно верное решение (рис. 9.10, *б*), однако его необходимая точность обеспечивается только в последней точке временного интервала.

В заключение остановимся на влиянии выбора параметра `acc` на расчеты. Как правило, разные численные методы работают несколько по-разному. Алгоритм Рунге—Кутты дает результат тем ближе к истинному, чем меньше выбирается параметр `acc`, а Булирша—Штера часто демонстрирует менее естественную зависимость $y(\varepsilon)$: даже при относительно больших ε реальная погрешность остается хорошей (намного лучше метода Рунге—Кутты).

Поэтому для экономии времени расчетов в функции `bulstoer` можно выбирать и большие `acc`.

Чтобы обеспечить заданную точность, данные алгоритмы могут изменять как количество шагов, разбивающих интервал `(t0,t1)`, так и их расположение вдоль интервала. Чтобы выяснить, на сколько шагов разбивался интервал при расчетах, воспользуемся простой программой, представленной в листинге 9.7. В ней определены пользовательские функции $R(\varepsilon)$ и $B(\varepsilon)$, вычисляющие для конкретной задачи (методов Рунге—Кутты и Булирша—Штера соответственно) зависимость числа шагов (т. е. числа строк получающейся матрицы) от параметра $\varepsilon=acc$. Графики

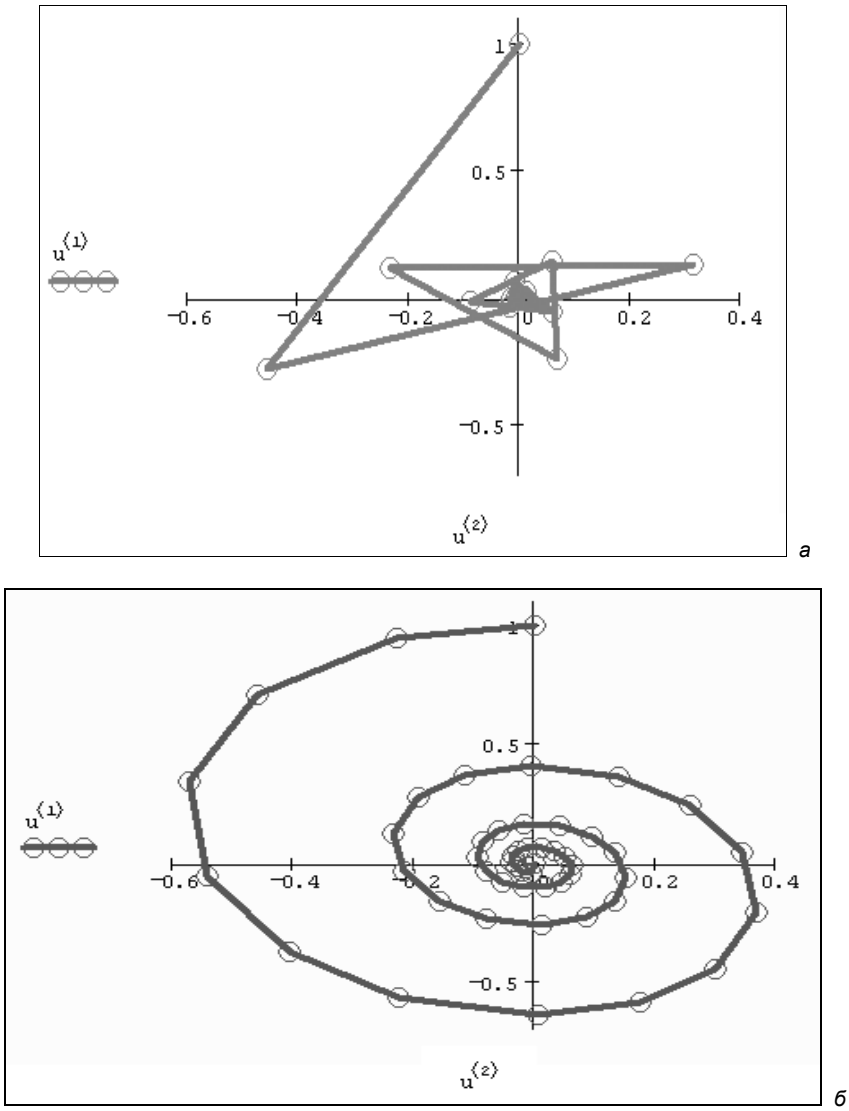


Рис. 9.10. Фазовый портрет, полученный `bulstoer` (*а*) и `rkadapt` (*б*) (продолжение листинга 9.5)

функций `R(ε)` и `B(ε)` показаны на рис. 9.11. Как видно, методу Рунге—Кутты при увеличении требуемой точности требуется все возрастающее количество шагов (и, соответственно, времени на расчеты), в противоположность методу Булирша—Штера, демонстрирующему бóльшую экономичность.

Примечание

Максимальное количество шагов алгоритма ограничивается еще одним параметром `k` встроенных функций `bulstoer` и `rkadapt` (*см. разд. 9.3.3*).

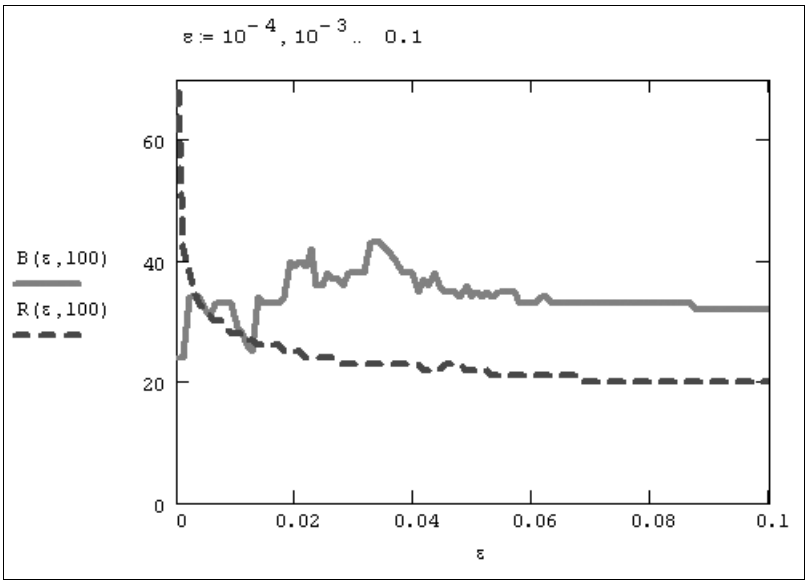


Рис. 9.11. Зависимость числа шагов от параметра `acc` численных методов (продолжение листинга 9.7)

Листинг 9.7. Число шагов адаптивного численного алгоритма в зависимости от погрешности `acc`

$$D(t,y) := \begin{pmatrix} y_1 \\ -y_0 - 0.1 \cdot y_1 \end{pmatrix}$$

$$k := 300$$

$$B(\varepsilon) := \mathrm{rows}\left[\mathrm{bulstoer}\left[\begin{pmatrix} 0.1 \\ 0 \end{pmatrix}, 0, 50, \varepsilon, D, k, 0.001\right]\right]$$

$$R(\varepsilon) := \mathrm{rows}\left[\mathrm{rkadapt}\left[\begin{pmatrix} 0.1 \\ 0 \end{pmatrix}, 0, 50, \varepsilon, D, k, 0.001\right]\right]$$

Таким образом, проводя тестовые расчеты для различных задач и подбирая наилучший набор параметров, можно существенно сэкономить ресурсы компьютера. Конечно, проводить подобный анализ стоит в случаях, когда время расчетов для вас критично.

9.4. Жесткие системы ОДУ

До сих пор мы имели дело с "хорошими" уравнениями, которые надежно решались численными методами Рунге—Кутты. Однако имеется класс так называемых *жестких* (stiff) систем ОДУ, для которых стандартные методы практически неприменимы, поскольку их решение требует исключительно малого значения шага чис-

ленного метода. Некоторые из специальных алгоритмов, разработанных для этих систем, реализованы в Mathcad.

9.4.1. Что такое жесткие ОДУ?

Исторически интерес к жестким системам возник в середине XX в. при изучении уравнений химической кинетики с одновременным присутствием очень медленно и очень быстро протекающих химических реакций (оно будет рассмотрено в *разд. 9.4.3*). Тогда неожиданно оказалось, что считавшиеся исключительно надежными методы Рунге—Кутты стали давать сбой при расчете этих задач.

Строгого общепринятого математического определения жестких ОДУ нет. В рамках этой книги будем считать, что жесткие системы — это те уравнения, решение которых получить намного проще с помощью определенных неявных методов, чем посредством явных методов (типа тех, что были рассмотрены в предыдущих разделах). Начнем разговор о жестких ОДУ с примера нежесткого уравнения (листинг 9.8), решение которого удается получить численным методом Рунге—Кутты (рис. 9.12).

Листинг 9.8. Решение нежесткого ОДУ

```
Given
d/dt y(t) = -10·(y(t) - cos(t))
y(0) = 1
y := Odesolve(t, 1)
```

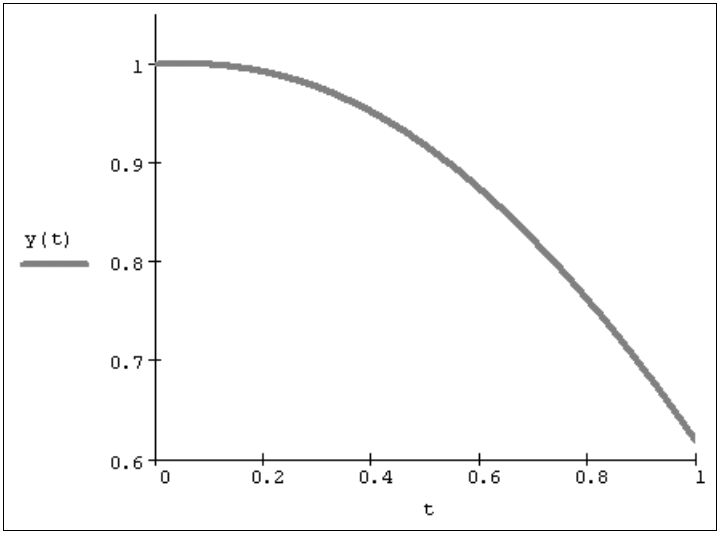


Рис. 9.12. Решение нежесткого ОДУ методом Рунге—Кутты (продолжение листинга 9.8)

Обратите внимание на значение коэффициента `-10` во второй строке листинга. Если изменить его на `-50`, то решение меняется до неузнаваемости (рис. 9.13). Вас, несомненно, должен насторожить результат, выданный Mathcad. Характерная "разболтка" решения говорит о неустойчивости алгоритма. Первое, что можно сделать, — увеличить количество шагов в методе Рунге—Кутты. Для этого достаточно добавить третий параметр `step` в функцию `Odesolve(t,1,step)`. После нескольких экспериментов можно подобрать такое значение `step`, которое будет обеспечивать устойчивость решения. Читатель может самостоятельно убедиться, что при `step>50` "разболтка" пропадает, и решение принимает вид графика, показанного далее на рис. 9.14.

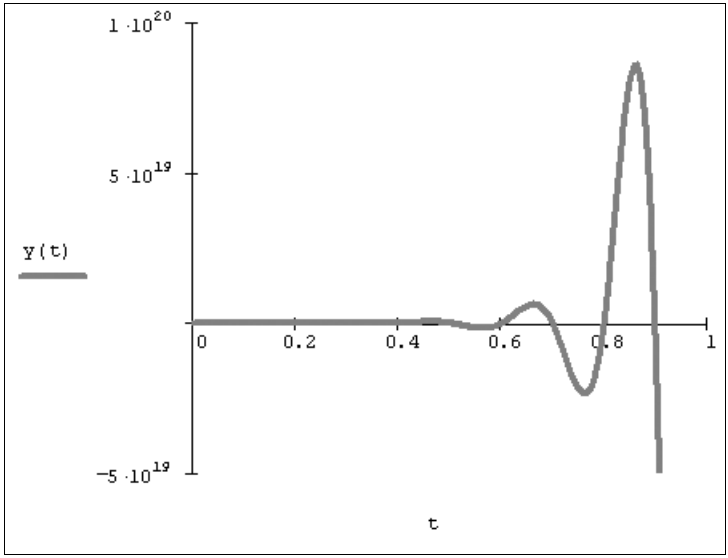


Рис. 9.13. Неверное решение более жесткого ОДУ методом Рунге—Кутты (листинг 9.8 с коэффициентом `−50`)

Таким образом, во-первых, мы выяснили, что одни и те же уравнения с разными параметрами могут быть как жесткими, так и нежесткими. Во-вторых, чем жестче уравнение, тем больше шагов в обычных численных методах требуется для его устойчивого решения. С классическим примером ОДУ из листинга 9.8 все получилось хорошо, т. к. оно было не очень жестким, и небольшое увеличение числа шагов разрешило все проблемы. Для решения обычными методами более жестких уравнений требуются миллионы, миллиарды и даже большее число шагов.

Примечание

В последних версиях (Mathcad 15) функция `Odesolve` успешно справляется с решением этой задачи и без ручной регулировки параметра `step`.

9.4.2. Функции для решения жестких ОДУ

Решение жестких систем дифференциальных уравнений можно осуществить только с помощью встроенных функций, аналогичных по действию семейству рассмотренных выше функций для обычных ОДУ:

- `Radau(y0,t0,t1,M,F)` — алгоритм RADAUS для жестких систем ОДУ;
- `Stiffb(y0,t0,t1,M,F,J)` — алгоритм Булирша—Штера для жестких систем ОДУ;
- `Stiffr(y0,t0,t1,M,F,J)` — алгоритм Розенброка для жестких систем ОДУ:
  - `y0` — вектор начальных значений в точке `t0`;
  - `t0`,`t1` — начальная и конечная точки расчета;
  - `M` — число шагов численного метода;
  - `F` — векторная функция `F(t,y)` размера `1×N`, задающая систему ОДУ;
  - `J` — матричная функция `J(t,y)` размера `(N+1)×N`, составленная из вектора производных функции `F(t,y)` по `t` (правый столбец) и ее якобиана (`N` левых столбцов).

Как вы можете заметить, для двух последних функций серьезным отличием от функций, решающих нежесткие системы, является добавление к стандартному набору параметров дополнительной матричной функции, задающей якобиан системы ОДУ. Решение выдается в виде матрицы, по форме идентичной аналогичным функциям решения нежестких задач Коши.

Решение жесткой задачи из предыдущего раздела при помощи функции `Radau` приведено в листинге 9.9. Результат показан в виде графика на рис. 9.14 вместе с графиком решения менее жесткой задачи (для которого применялся листинг 9.8). Как вы видите, хватило всего пяти точек разбиения интервала интегрирования жесткого ОДУ, чтобы метод с ним справился. Специфика применения других встроенных функций, требующих дополнительного задания якобиана, будет рассмотрена в следующем разделе на примере уравнения химической кинетики.

Листинг 9.9. Решение жесткого ОДУ алгоритмом RADAUS

```
F(t, y) := −100 · (y − cos(t))
y0 := 1        M := 5
u := Radau(y0, 0, 1, M, F)
```

В заключение приведем соответствующие встроенные функции, которые применяются для решения жестких систем ОДУ не на всем интервале, а только в одной заданной точке `t1`.

- `radau(y0,t0,t1,acc,F,k,s)` — метод RADAUS.
- `stiffb(y0,t0,t1,acc,F,J,k,s)` — метод Булирша—Штера.
- `stiffr(y0,t0,t1,acc,F,J,k,s)` — метод Розенброка.

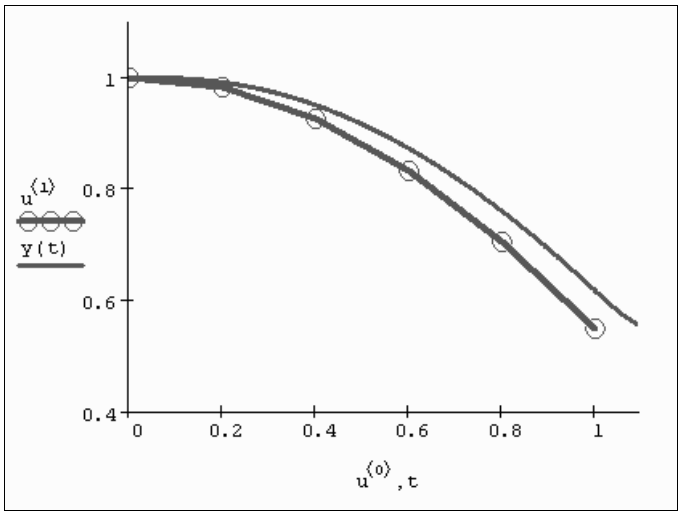


Рис. 9.14. Решение жесткого ОДУ методом RADAUS
(продолжение листингов 9.8 и 9.9)

Имена данных функций пишутся со строчной буквы, а их действие и набор параметров полностью аналогичны рассмотренным нами ранее для функций, относящихся к решению в заданной точке нежестких систем (*см. разд. 9.3.2*). Отличие заключается в специфике применяемого алгоритма и необходимости задания матричной функции якобиана `J(t,y)` (для двух последних функций).

9.4.3. Пример: химическая кинетика

Рассмотрим классическую модель химической кинетики (Робертсон, 1966 г.), которая как нельзя лучше передает смысл понятия жесткости ОДУ.

Попытка решения стандартными методами

Рассмотрим составную схему химического взаимодействия трех веществ. Пусть вещество `"0"` медленно превращается в `"1"`: `"0"`→`"1"` (со скоростью `0.1`), вещество `"1"` при каталитическом воздействии самого себя превращается очень быстро в вещество `"2"`: `"1"+"1"`→`"2"+"1"` (10^3). И наконец, подобным образом (но со средней скоростью) реагируют вещества `"2"` и `"1"`: `"1"+"2"`→`"0"+"2"` (10^2). Система ОДУ, описывающая динамику концентрации реагентов, с попыткой решения методом Рунге—Кутты, приведена в символической форме в первой строке листинга 9.10.

Листинг 9.10. Жесткая система ОДУ химической кинетики

$$F(t,y) := \begin{pmatrix} -0.1\cdot y_0 + 10^2\cdot y_1\cdot y_2 \\ 0.1\cdot y_0 - 10^2\cdot y_1\cdot y_2 - 10^3\cdot y_1 \\ 10^3\cdot y_1 \end{pmatrix}$$

$$y0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

```
D:= rkfixed(y0, 0, 50, 20000, F)
```

Бросается в глаза сильно различающийся порядок коэффициентов при разных слагаемых. Именно степень этого различия чаще всего и определяет жесткость системы ОДУ. В качестве соответствующей характеристики выбирают матрицу Якоби (якобиан) векторной функции `F(t,y)`, т. е. функциональную матрицу, составленную из производных `F(t,y)` (*см. разд. 3.4.3*). Чем сильнее вырождена матрица Якоби, тем жестче система уравнений. В приведенном примере определитель якобиана и вовсе равен нулю при любых значениях y_0, y_1 и y_2 (листинг 9.11, вторая строка). В первой строке листинга 9.11 приведено напоминание способа вычисления якобиана средствами Mathcad на примере определения элементов его первой строки.

Листинг 9.11. Якобиан рассматриваемой системы ОДУ химической кинетики

$$\frac{\partial}{\partial x}F\left[t, \begin{pmatrix} x \\ y_1 \\ y_2 \end{pmatrix}\right]_0 \rightarrow -.1 \quad \frac{\partial}{\partial x}F\left[t, \begin{pmatrix} y_0 \\ x \\ y_2 \end{pmatrix}\right]_0 \rightarrow 100\cdot y_2 \quad \frac{\partial}{\partial x}F\left[t, \begin{pmatrix} y_0 \\ y_1 \\ x \end{pmatrix}\right]_0 \rightarrow 100\cdot y_1$$

$$\left|\begin{pmatrix} -0.1 & 10^2\cdot y_2 & 10^2\cdot y_1 \\ 0.1 & -10^2\cdot y_2 - 10^3 & -10^2\cdot y_1 \\ 0 & 10^3 & 0 \end{pmatrix}\right| \rightarrow 0$$

Для примера, приведенного в листинге 9.10, стандартным методом Рунге—Кутты все-таки удается найти решение (оно показано на рис. 9.15). Однако для этого требуется очень большое число шагов, `M=20000`, что делает расчеты очень медленными. При меньшем числе шагов численному алгоритму не удается найти решение. В процессе работы алгоритма оно расходится, и Mathcad вместо результата выдает ошибку о превышении предельно большого числа.

Еще один факт, на который стоит обратить внимание, — это различие в порядке величины получающегося решения. Как видно из рис. 9.15, концентрация первого реагента y_1 существенно (в тысячи раз) превышает концентрацию остальных. Это свойство также очень характерно для жестких систем.

Примечание

В принципе можно было бы снизить жесткость системы "вручную", применяя масштабирование. Для этого нужно искусственно уменьшить искомую функцию y_1, к примеру, в тысячу раз, разделив все слагаемые в системе ОДУ, содержащие y_1, на `1000`. После масштабирования для решения полученной системы методом Рунге—Кутты будет достаточно взять всего `M=20` шагов. Соответствующий пример вы найдете на компакт-диске, прилагаемом к данной книге.

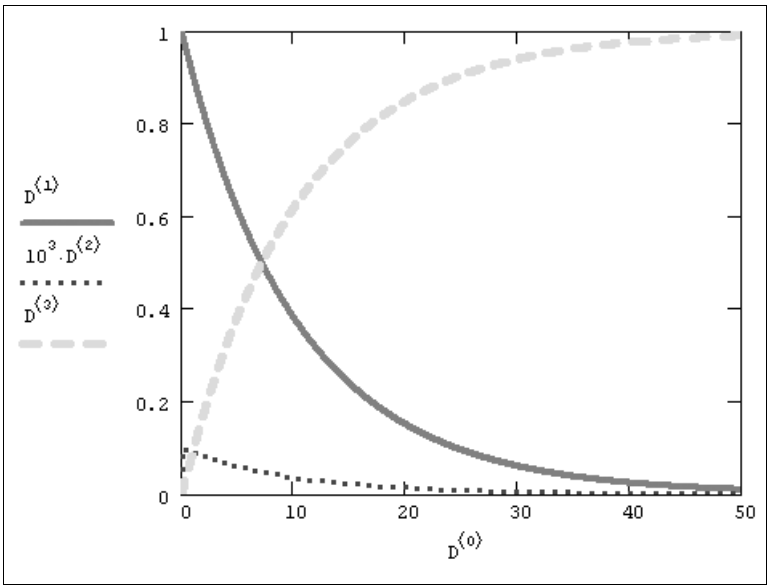


Рис. 9.15. Решение жесткой системы ОДУ химической кинетики методом Рунге—Кутты (продолжение листинга 9.10)

Применение специфических алгоритмов

Покажем действие этих алгоритмов на том же примере жесткой системы ОДУ химической кинетики (листинг 9.12). Обратите внимание, как следует представлять в данном случае якобиан, сравнив задание матричной функции в предпоследней строке листинга 9.12 с заданием якобиана из листинга 9.11.

Листинг 9.12. Решение жесткой системы ОДУ химической кинетики

$$y0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$F(t, y) := \begin{pmatrix} -0.1 \cdot y_0 + 10^2 \cdot y_1 \cdot y_2 \\ 0.1 \cdot y_0 - 10^2 \cdot y_1 \cdot y_2 - 10^3 \cdot y_1 \\ 10^3 \cdot y_1 \end{pmatrix}$$

$$J(t, y) := \begin{pmatrix} 0 & -0.1 & 10^2 \cdot y_2 & 10^2 \cdot y_1 \\ 0 & 0.1 & -10^2 \cdot y_2 - 10^3 & -10^2 \cdot y_1 \\ 0 & 0 & 10^3 & 0 \end{pmatrix}$$

```
D := Stiffb(y0, 0, 50, 20, F, J)
```

Расчеты показывают, что для получения того же результата (который был изображен на рис. 9.15) оказалось достаточно в тысячу раз меньшего количества шагов численного алгоритма, чем для стандартного метода Рунге—Кутты! Примерно во столько же раз требуется меньше компьютерного времени на проведение расчетов. Стоит ли говорить, что, если вы имеете дело с жесткими (в той или иной степени) системами, применение описанных специальных алгоритмов просто необходимо.

Важно заметить, что до сих пор мы имели дело с примером не очень жесткой системы. Попробуйте вместо скоростей упомянутых химических реакций `0.1`, 10^3 и 10^2 взять другие числа, например, `0.05`, 10^4 и 10^7 соответственно. Заметим, что такое соотношение скоростей часто встречается в прикладных задачах химической кинетики и определяет куда более жесткую систему ОДУ. Ее уже никак не удается решить стандартными методами, поскольку число шагов численного метода должно быть просто гигантским. А между тем алгоритмы для жестких ОДУ справляются с этой задачей с легкостью (рис. 9.16), причем практически при тех же значениях шага, что были взяты в листинге 9.12. Обратите внимание, что порядки величины решений для концентраций различных веществ на рис. 9.16 различаются еще сильнее, чем в предыдущем (менее жестком) примере.

Для решения очень жестких систем особенно подходит функция `Radau`, которая соблазнительна еще и тем, что избавляет от необходимости предварительного расчета якобиана. В случае очень жесткой задачи химической кинетики результат, выдаваемый функцией `Radau`, практически тот же, что и в случае использования алгоритма Розенброка (рис. 9.16). Любопытно, что решение менее жесткой системы (из предыдущего листинга) при помощи функции `Radau` удается получить только для первой половины интервала, примерно до `t1=20`. Для больших интервалов она дает сбой, выводя вместо решения сообщение об ошибке.

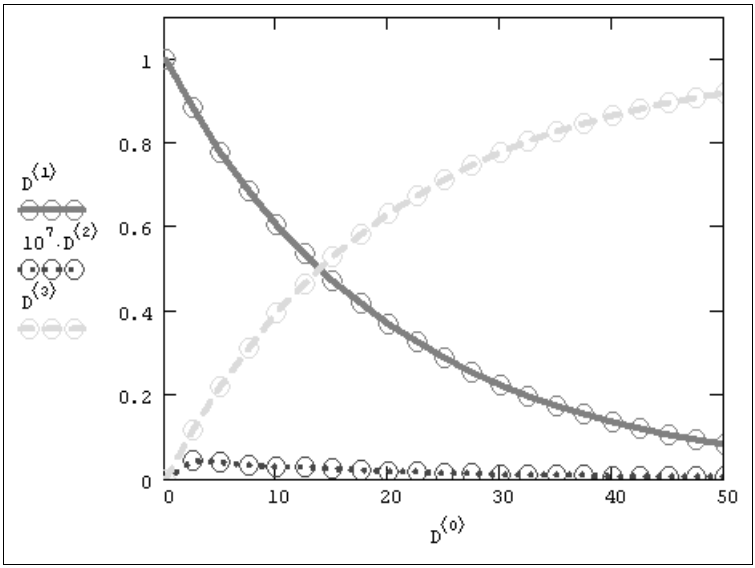


Рис. 9.16. Решение более жесткой системы ОДУ химической кинетики методом Розенброка

⑭ 9.5. Дополнительные функции решения ОДУ

В Mathcad 14 появилось несколько новых встроенных функций для решения как жестких, так и нежестких ОДУ, основанных на более современных численных алгоритмах.

9.5.1. Новые алгоритмы решения жестких и нежестких ОДУ

Перечислим новые встроенные функции, предназначенные для решения систем дифференциальных уравнений, а также упомянем об изменениях, добавленных в существующие функции. Учитывая, что почти все функции предусматривают как обязательные, так и опциональные аргументы, заключим такие необязательные аргументы в квадратные скобки.

- `Radau(y0,t0,t1,M,F,[J,A,acc])` — модификация алгоритма RADAU (для жестких систем ОДУ и для систем дифференциальных и алгебраических уравнений);
- `Adams(y0,t0,t1,M,F,[acc])` — алгоритм Адамса (применяется для нежестких систем ОДУ);
- `BDF(y0,t0,t1,M,F,[J,acc])` — алгоритм BDF (применяется для жестких систем ОДУ);
- `AdamsBDF(y0,t0,t1,M,F,[J,acc])` — универсальный гибридный метод Адамса-BDF, применяемый для любых (как жестких, так и нежестких) систем ОДУ (численный алгоритм сам решает, насколько жесткой является система, и какую из разновидностей метода следует использовать):
  - `y0` — вектор начальных значений в точке `t0`;
  - `t0`,`t1` — начальная и конечная точки расчета;
  - `M` — число шагов численного метода;
  - `F` — векторная функция `F(t,y)` размера `1×N`, задающая систему ОДУ;
  - `J` — Якобиан (матричная функция Якоби) `J(t,y)` размера `N×N`, составленная из векторов частных производных векторной функции `F(t,y)`.
  - `acc` — погрешность вычисления;
  - `A` — матрица, составленная из действительных чисел, которая задает связь между неизвестными в форме матричного равенства `A•y=F(t,y)`.

Внимание!

Структура матрицы `J`, применяемой для перечисленных функций, т. е. якобиана размера `N×N`, иная, нежели для других встроенных функций решения ОДУ, где она представляет собой комбинированную матрицу размера `(N+1)×N` (*см. разд. 9.4.2*).

Примечание

В Mathcad 14 именно гибридный алгоритм Адамса-BDF используется по умолчанию в работе функции `Odesolve`.

Пример использования новых функций приведен в листингах 9.13 и 9.14. Обратите внимание на то, что листинг 9.13 представляет автоматизированный подсчет матрицы Якоби при помощи новой встроенной функции `Jacob`.

Листинг 9.13. Решение ОДУ химической кинетики методом BDF

$$F(t,y) := \begin{pmatrix} -0.1\cdot y_0 + 10^2\cdot y_1\cdot y_2 \\ 0.1\cdot y_0 - 10^2\cdot y_1\cdot y_2 - 10^3\cdot y_1 \\ 10^3\cdot y_1 \end{pmatrix}$$

$$y0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$J(t,y) := \mathrm{Jacob}(F(t,y),y)$$

$$J(t,y) \rightarrow \begin{bmatrix} -0.1 & 100\cdot y_2 & 100\cdot y_1 \\ 0.1 & -100\cdot y_2 - 1000 & (-100)\cdot y_1 \\ 0 & 1000 & 0 \end{bmatrix}$$

$$D := \mathrm{BDF}(y0, 0, 50, 20, F, J)$$

Листинг 9.14. Решение системы ОДУ и алгебраического уравнения

$$F(t,y) := \begin{pmatrix} -0.1\cdot y_0 + 10^2\cdot y_1\cdot y_2 \\ 0.1\cdot y_0 - 10^2\cdot y_1\cdot y_2 - 10^3\cdot y_1 \\ y_0 + y_1 + y_2 - 1 \end{pmatrix}$$

$$y0 := \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$A := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$J(t,y) := \mathrm{Jacob}(F(t,y),y)$$

$$J(t,y) \rightarrow \begin{bmatrix} -0.1 & 100\cdot y_2 & 100\cdot y_1 \\ 0.1 & -100\cdot y_2 - 1000 & (-100)\cdot y_1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$D := \mathrm{Radau}(y0, 0, 50, 200, F, J, A)$$

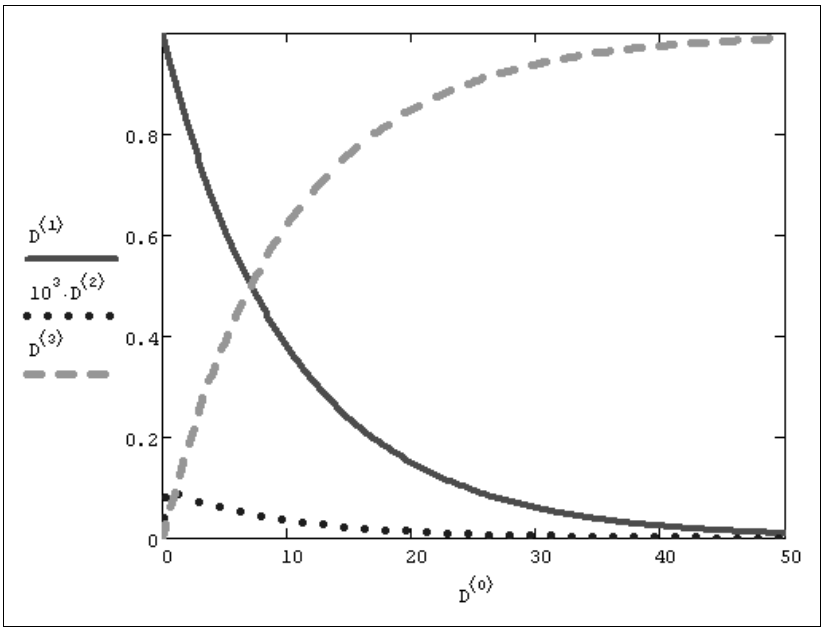


Рис. 9.17. График решения гибридной системы
(продолжение листинга 9.14)

В листинге 9.14 показано решение гибридной системы, составленной из двух дифференциальных и одного алгебраического уравнения вида

$$\frac{d}{dt}y_0 = -.1\cdot y_0 + 100\cdot y_1\cdot y_2$$

$$\frac{d}{dt}y_1 = .1\cdot y_0 - 100\cdot y_1\cdot y_2 - 10^3\cdot y_1$$

$$0 = y_0 + y_1 + y_2 - 1$$

посредством модернизированной функции `Radau`. Результат ее применения показан в виде графика на рис. 9.17.

9.5.2. Решение линейных ОДУ

Разработчики Mathcad 14 предусмотрели дополнительную встроенную функцию `statespace` решения линейных ОДУ определенного вида `y′(t)=A(t)·y(t)+ +B(t)·u(t)`.

❒ `statespace(y0,t0,t1,M,A,B,u)` — функция решения систем линейных ОДУ:

- `y0` — вектор начальных значений в точке `t0`;
- `t0`,`t1` — начальная и конечная точки расчета;
- `M` — число шагов численного метода;
- `acc` — погрешность вычисления;

- `A(t)` — матричная функция размера `N×N`;
- `B(t)` — векторная функция размера `N×k`;
- `u(t)` — функция скалярная, если `k=1`, или векторная размера `k×1`.

Пример использования функции `statespace` иллюстрируется листингом 9.15 и рис. 9.18.

Листинг 9.15. Решение системы линейных ОДУ

$$A(t) := \begin{pmatrix} 0 & 1 \\ -1 & -2 \end{pmatrix} \qquad B(t) := \begin{pmatrix} -3 \\ .1 \end{pmatrix}$$

$$y0 := \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$u(t) := \sin(t)$$

$$D := \mathrm{statespace}(y0, 0, 20, 500, A, B, u)$$

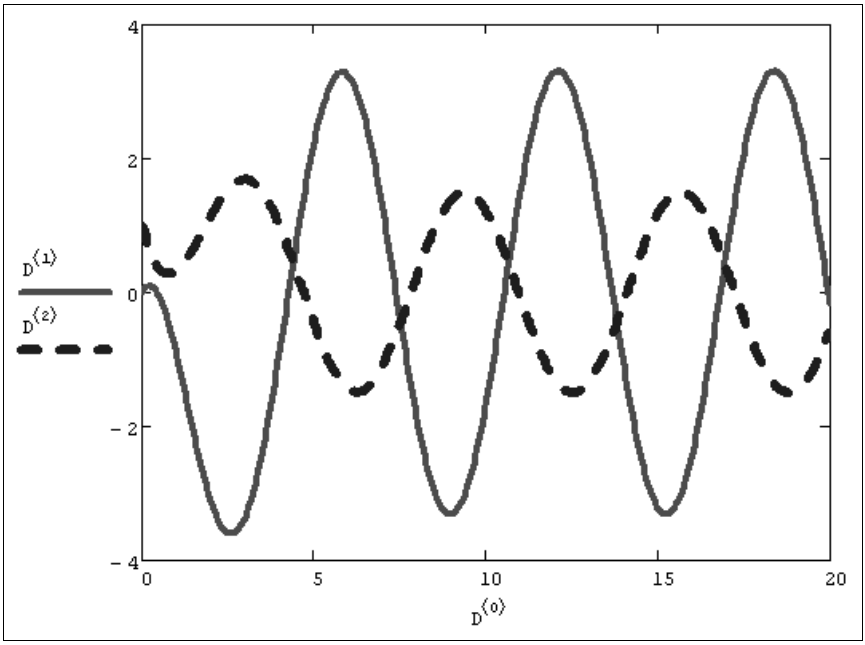


Рис. 9.18. Решение системы линейных ОДУ
(продолжение листинга 9.15)

ГЛАВА 10

Обыкновенные дифференциальные уравнения: краевые задачи

В этой главе рассматриваются краевые задачи для обыкновенных дифференциальных уравнений (ОДУ). Средства Mathcad, реализующие алгоритм стрельбы (*см. разд. 10.2*), позволяют решать краевые задачи для систем ОДУ, в которых часть граничных условий поставлена в начальной точке интервала, а остальная часть — в его конечной точке. Также возможно решать уравнения с граничными условиями, поставленными в некоторой внутренней точке.

Краевые задачи во множестве практических приложений часто зависят от некоторого числового параметра. При этом решение существует не для всех его значений, а лишь для счетного их числа. Такие задачи называют задачами на собственные значения (*см. разд. 10.3*).

Несмотря на то, что, в отличие от задач Коши для ОДУ, в Mathcad не предусмотрены встроенные функции для решения жестких краевых задач, с ними все-таки можно справиться, применив программирование разностных схем, подходящих для решения задач этого класса (*см. разд. 10.4*). О подходе к решению нелинейных краевых задач написано в конце главы (*см. разд. 10.5*).

10.1. О постановке задач

Постановка краевых задач для ОДУ отличается от задач Коши, рассмотренных в *главе 9*, тем, что граничные условия для них ставятся не в одной начальной точке, а на обеих границах расчетного интервала. Если имеется система N обыкновенных дифференциальных уравнений первого порядка, то часть из N условий может быть поставлена на одной границе интервала, а оставшиеся условия — на противоположной границе.

Примечание

Дифференциальные уравнения высших порядков можно свести к эквивалентной системе ОДУ первого порядка (*см. главу 9*).

Чтобы лучше понять, что из себя представляют краевые задачи, рассмотрим их постановочную часть на конкретном физическом примере модели взаимодействия встречных световых пучков. Предположим, что надо определить распределение интенсивности оптического излучения в пространстве между источником (лазером) и зеркалом, заполненном некоторой средой (рис. 10.1). Будем считать, что от зеркала отражается R-я часть падающего излучения (т. е. его коэффициент отражения равен R), а среда как поглощает излучение с коэффициентом ослабления a(x), так и рассеивает его. Причем коэффициент рассеяния назад равен r(x). В этом случае закон изменения интенсивности $y_0(x)$ излучения, распространяющегося вправо, и интенсивности $y_1(x)$ излучения, распространяющейся влево, определяется системой двух ОДУ первого порядка:

$$\begin{aligned} \frac{dy_0(x)}{dx} &= -a(x)\cdot y_0(x) + r(x)\cdot y_1(x); \\ \frac{dy_1(x)}{dx} &= a(x)\cdot y_1(x) - r(x)\cdot y_0(x)\cdot \end{aligned} \tag{10.1}$$

Для правильной постановки задачи требуется, помимо уравнений, задать такое же количество граничных условий. Одно из них будет выражать известную интенсивность излучения I0, падающего с левой границы x=0, а второе — закон отражения на его правой границе x=1:

$$\begin{aligned} &y_0(0) = I0; \\ &y_1(1) = R\times y_0(1). \end{aligned} \tag{10.2}$$

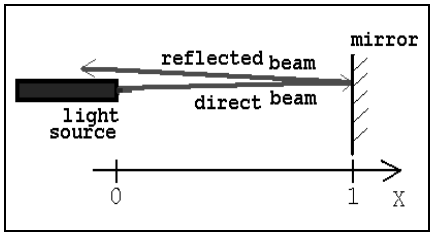


Рис. 10.1. Модель краевой задачи

Полученную задачу называют *краевой* (boundary value problem), поскольку условия поставлены не на одной, а на обеих границах интервала (0,1). И в связи с этим их не решить методами, описанными в предыдущей главе и предназначенными для задач с начальными условиями. Далее для показа возможностей Mathcad будем использовать этот пример с R=1 и конкретным видом a(x)=const=1 и r(x)=const=0.1, описывающим случай *изотропного* (не зависящего от координаты x) рассеяния.

Примечание 1

Модель, представленная на рис. 10.1, привела к краевой задаче для системы *линейных* ОДУ. Она имеет аналитическое решение в виде комбинации экспонент. Более сложные, *нелинейные*, задачи возможно решить только численно. Нетрудно сообразить, что модель станет нелинейной, если сделать коэффициенты ослабления и рассеяния зависящими от интенсивности излучения. Физически это будет соответствовать изменению оптических свойств среды под действием мощного излучения.

Примечание 2

Модель встречных световых пучков привела нас к системе уравнений (10.1), в которые входят производные только по одной переменной x. Если бы мы стали рассматривать более сложные эффекты рассеяния в стороны (а не только вперед и назад), то в уравнениях появились бы частные производные по другим пространственным переменным y и z. В этом случае получилась бы краевая задача для уравнений в частных производных, решение которой во много раз сложнее ОДУ.

10.2. Решение краевых задач средствами Mathcad

Для решения краевых задач в Mathcad реализован наиболее популярный алгоритм, называемый методом *стрельбы* или *пристрелки* (shooting method). Он, по сути, сводит решение краевой задачи к решению серии задач Коши с различными начальными условиями. Рассмотрим сначала основной принцип алгоритма стрельбы на примере модели световых пучков (см. рис. 10.1), а встроенные функции, реализующие этот алгоритм, приведем в следующем разделе.

10.2.1. Алгоритм стрельбы

Суть метода стрельбы заключается в пробном задании недостающих граничных условий на левой границе интервала и решении затем полученной задачи Коши хорошо известными методами (*см. главу 9*). В нашем примере не хватает начального условия для $y_1(0)$, поэтому сначала зададим ему произвольное значение, например, $y_1(0)=10$. Конечно, такой выбор не совсем случаен, поскольку из физических соображений ясно, что, во-первых, интенсивность излучения — величина заведомо положительная, и, во-вторых, отраженное излучение должно быть намного меньше падающего. Решение задачи Коши с помощью функции rkfixed приведено в листинге 10.1, причем в его последней строке выведены расчетные значения y_0 и y_1 на правой крайней точке интервала x=1. Разумеется, они не совпадают, т. е. правое краевое условие не выполнено, из чего следует, что полученный результат не является решением поставленной краевой задачи.

Листинг 10.1. Решение пробной задачи Коши для модели (10.1)

$r(x) := .1 \quad a(x) := 1$

$R := 1$

$$D(x, y) := \begin{pmatrix} -a(x)\cdot y_0 + r(x)\cdot y_1 \\ a(x)\cdot y_1 - r(x)\cdot y_0 \end{pmatrix}$$

$I0 := 100$

$$y := \begin{pmatrix} I0 \\ 10 \end{pmatrix}$$

$M := 10$

$S := \mathrm{rkfixed}(y, 0, 1, M, D)$

$S_{M,1} = 37.558 \qquad S_{M,2} = 15.372$

График полученных решений показан на рис. 10.2 (слева). Из него также видно, что взятое наугад второе начальное условие не обеспечило выполнение граничного условия при $x=1$. В целях лучшего выполнения этого граничного условия следует взять большее значение $y_1(0)$, например, $y_1(0)=15$, и вновь решить задачу Коши. Соответствующий результат показан на том же рис. 10.2 (в центре). Граничное условие выполняется с лучшей точностью, но опять-таки оказалось недостаточным. Для еще одного значения $y_1(0)=20$ получается решение, показанное на рис. 10.2 (справа). Из сравнения двух правых графиков легко заключить, что недостающее начальное условие больше 15, но меньше 20. Продолжая подобным образом "пристрелку" по недостающему начальному условию, возможно отыскать правильное решение краевой задачи.

В этом и состоит принцип алгоритма стрельбы. Выбирая пробные начальные условия (проводя пристрелку) и решая соответствующую серию задач Коши, можно найти то решение системы ОДУ, которое (с заданной точностью) удовлетворит граничному условию (или, в общем случае, условиям) на другой границе расчетного интервала.

Конечно, описанный алгоритм несложно запрограммировать самому, оформив его как решение системы заданных алгоритмически уравнений, выражающих граничные условия на второй границе, относительно неизвестных пристрелочных начальных условий. Но делать этого нет необходимости, поскольку он оформлен в Mathcad в виде встроенных функций.

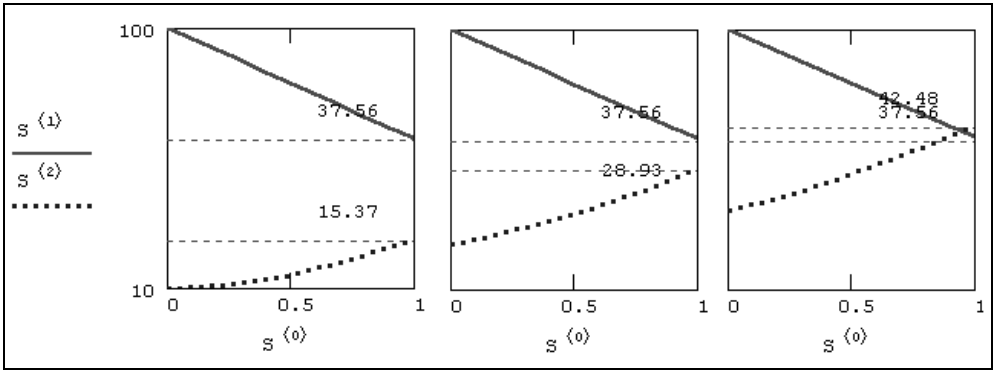


Рис. 10.2. Иллюстрация метода стрельбы (продолжение листинга 10.1)

10.2.2. Двухточечные краевые задачи

Решение краевых задач для систем ОДУ методом стрельбы в Mathcad достигается применением двух встроенных функций. Первая предназначена для двухточечных задач с краевыми условиями, заданными на концах интервала:

- `sbval(z,x0,x1,D,load,score)` — поиск вектора недостающих `L` начальных условий для двухточечной краевой задачи для системы `N` ОДУ:
  - `z` — вектор размера `L×1`, присваивающий недостающим начальным условиям (на левой границе интервала) начальные значения;
  - `x0` — левая граница расчетного интервала;
  - `x1` — правая граница расчетного интервала;
  - `load(x0,z)` — векторная функция размера `N×1` левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента `z`;
  - `score(x1,y)` — векторная функция размера `L×1`, выражающая `L` правых граничных условий для векторной функции `y` в точке `x1`;
  - `D(x,y)` — векторная функция, описывающая систему `N` ОДУ, размера `N×1` и двух аргументов — скалярного `x` и векторного `y`. При этом `y` — это неизвестная векторная функция аргумента `x` того же размера `N×1`.

Внимание!

Решение краевых задач в Mathcad, в отличие от большинства остальных операций, реализовано не совсем очевидным образом. В частности, помните, что число элементов векторов `D` и `load` равно количеству уравнений `N`, а векторов `z`, `score` и результата действия функции `sbval` — количеству правых граничных условий `L`. Соответственно, левых граничных условий в задаче должно быть `(N-L)`.

Как видно, функция `sbval` предназначена не для поиска собственно решения, т. е. неизвестных функций $y_i(x)$, а для определения недостающих начальных условий в первой точке интервала, т. е. $y_i(x0)$. Чтобы вычислить $y_i(x)$ на всем интервале, требуется дополнительно решить задачу Коши.

Разберем особенности использования функции `sbval` на конкретном примере (листинг 10.2), описанном выше (*см. разд. 10.1.1*). Краевая задача состоит из системы двух уравнений (`N=2`), одного левого (`L=1`) и одного правого (`N-L=2-1=1`) граничного условия.

Листинг 10.2. Решение краевой задачи

$r(x) := .1 \qquad a(x) := 1$

$R := 1 \qquad I0 := 100$

$$D(x,y) := \begin{pmatrix} -a(x)\cdot y_0 + r(x)\cdot y_1 \\ a(x)\cdot y_1 - r(x)\cdot y_0 \end{pmatrix}$$

$z_0 := 10$

$$load(x0,z) := \begin{pmatrix} 100 \\ z_0 \end{pmatrix}$$

$score(x1,y) := R\cdot y_0 - y_1$

$I1 := sbval(z,0,1,D,load,score)$

```
I1 = ( 18.555 )
```

$$S := \mathrm{rkfixed}\left[\begin{pmatrix} I0 \\ I1_0 \end{pmatrix}, 0, 1, 10, D\right]$$

Первые три строки листинга задают необходимые параметры задачи и саму систему ОДУ. В четвертой строке определяется вектор `z`. Поскольку правое граничное условие всего одно, то недостающее начальное условие тоже одно, соответственно, и вектор `z` имеет только один элемент z_0. Ему необходимо присвоить начальное значение (мы приняли z_0=10, как в листинге 10.1), чтобы запустить алгоритм стрельбы (*см. разд. 10.1.2*).

> ***Примечание***
>
> Начальное значение фактически является параметром численного метода и поэтому может решающим образом повлиять на решение краевой задачи.

В следующей строке листинга векторной функции `load(x,z)` присваиваются левые граничные условия. Эта функция аналогична векторной переменной, определяющей начальные условия для встроенных функций, решающих задачи Коши. Отличие заключается в записи недостающих условий. Вместо конкретных чисел на соответствующих местах пишутся имена искомых элементов вектора `z`. В нашем случае вместо второго начального условия стоит аргумент z_0 функции `load`. Первый аргумент функции `load` — это точка, в которой ставится левое граничное условие. Ее конкретное значение определяется непосредственно в списке аргументов функции `sbval`.

Следующая строка листинга определяет правое граничное условие, для введения которого используется функция `score(x,y)`. Оно записывается точно так же, как система уравнений в функции `D`. Аргумент `x` функции `score` аналогичен функции `load` и нужен для тех случаев, когда граничное условие явно зависит от координаты `x`. Вектор `score` должен состоять из такого же числа элементов, что и вектор `z`.

Реализованный в функции `sbval` алгоритм стрельбы ищет недостающие начальные условия таким образом, чтобы решение полученной задачи Коши делало функцию `score(x,y)` как можно ближе к нулю. Как видно из листинга, результат применения `sbval` для интервала `(0,1)` присваивается векторной переменной `I1`. Этот вектор похож на вектор `z`, только в нем содержатся искомые начальные условия вместо приближенных начальных значений, заданных в `z`. Вектор `I1` содержит, как и `z`, всего один элемент $I1_0$. С его помощью можно определить решение краевой задачи `y(x)` (последняя строка листинга). Тем самым функция `sbval` сводит решение краевых задач к задачам Коши. График решения краевой задачи показан на рис. 10.3.

На рис. 10.4 показано решение той же самой краевой задачи, но с другим правым граничным условием, соответствующим `R=0`, т. е. без зеркала на правой границе. В этом случае слабый обратный пучок света образуется исключительно за счет обратного рассеяния излучения от лазера. Конечно, многие из читателей уже обратили внимание, что реальная физическая среда не может создавать такого большого рассеяния назад. Иными словами, более реальны значения `r(x)<<a(x)`.

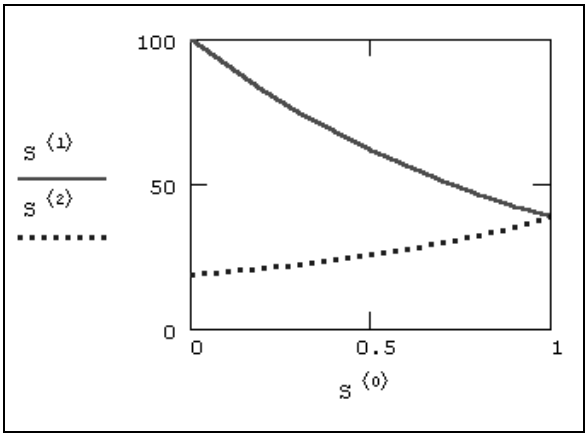


Рис. 10.3. Решение краевой задачи для `R=1` (продолжение листинга 10.2)

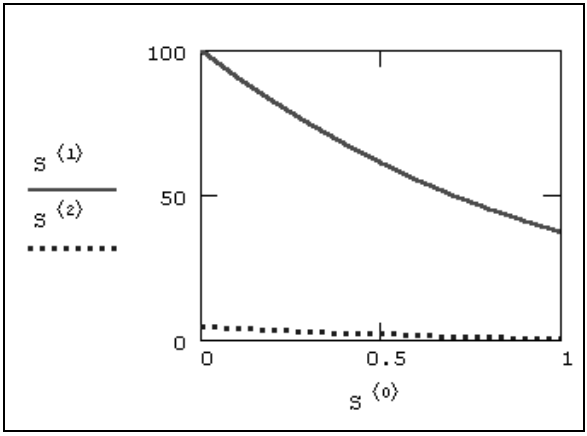


Рис. 10.4. Решение краевой задачи для `R=0`
(продолжение листинга 10.2 при `R=0`)

Однако когда коэффициенты в системе ОДУ при разных y_i очень сильно (на порядки) различаются, система ОДУ становится жесткой, и функция `sbval` не может найти решения, выдавая вместо него сообщение об ошибке.

Внимание!

Метод стрельбы не годится для решения жестких краевых задач. Поэтому алгоритмы решения жестких ОДУ в Mathcad приходится программировать самому (*см. разд. 10.4*).

10.2.3. Краевые задачи с условием во внутренней точке

Иногда дифференциальные уравнения определяются с граничными условиями не только на концах интервала, но и с дополнительным условием в некоторой промежуточной точке расчетного интервала. Чаще всего такие задачи содержат данные

о негладких в некоторой внутренней точке интервала решениях. Для них имеется встроенная функция `bvalfit`, также реализующая алгоритм стрельбы.

- `bvalfit(z1,z2,x0,x1,xf,D,load1,load2,score)` — поиск вектора недостающих граничных условий для краевой задачи с дополнительным условием в промежуточной точке для системы `N` ОДУ.
  - `z1` — вектор, присваивающий недостающим начальным условиям на левой границе интервала начальные значения.
  - `z2` — вектор того же размера, присваивающий недостающим начальным условиям на правой границе интервала начальные значения.
  - `x0` — левая граница расчетного интервала.
  - `x1` — правая граница расчетного интервала.
  - `xf` — точка внутри интервала.
  - `D(x,y)` — векторная функция, описывающая систему `N` ОДУ размера `N×1` и двух аргументов — скалярного `x` и векторного `y`. При этом `y` — это неизвестная векторная функция аргумента `x` того же размера `N×1`.
  - `load1(x0,z)` — векторная функция размера `N×1` левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента `z`.
  - `load2(x1,z)` — векторная функция размера `N×1` левых граничных условий, причем недостающие начальные условия поименовываются соответствующими компонентами векторного аргумента `z`.
  - `score(xf,y)` — векторная функция размера `N×1`, выражающая внутреннее условие для векторной функции `y` в точке `xf`.

Обычно функция `bvalfit` применяется для задач, в которых производная решения имеет разрыв во внутренней точке `xf`. Некоторые из таких задач не удается решить обычным методом пристрелки, поэтому приходится вести пристрелку сразу из обеих граничных точек. В этом случае дополнительное внутреннее условие в точке `xf` является просто условием сшивки в ней левого и правого решений. Поэтому для данной постановки задачи оно записывается в форме `score(xf,y):=y`.

Рассмотрим действие функции `bvalfit` на знакомом примере модели взаимодействия пучков света (см. рис. 10.1), предположив, что в промежутке между `xf=0.5` и `x1=1.0` находится другая, оптически более плотная среда с другим коэффициентом ослабления излучения `a(x)=3`. Соответствующая краевая задача решена в листинге 10.3, причем разрывный показатель ослабления определяется в его второй строке.

Листинг 10.3. Краевая задача с граничным условием в промежуточной точке

```
I0 := 100    r(x) := .1    R := 1
a(x) := | 1  if x < 0.5
        | 3  otherwise
```

$$D(x, y) := \begin{pmatrix} -a(x) \cdot y_0 + r(x) \cdot y_1 \\ a(x) \cdot y_1 - r(x) \cdot y_0 \end{pmatrix}$$

$$z1_0 := 20$$

$$load1(x0, z1) := \begin{pmatrix} 100 \\ z1_0 \end{pmatrix}$$

$$z2_0 := 30$$

$$load2(x1, z2) := \begin{pmatrix} z2_0 \\ R \cdot z2_0 \end{pmatrix}$$

$$score(xf, y) := y$$

$$I1 := bvalfit(z1, z2, 0, 1, 0.5, D, load1, load2, score)^T$$

$$I1 = \begin{pmatrix} 5.618 \\ 13.801 \end{pmatrix}$$

Система уравнений и левое краевое условие вводится так же, как и в предыдущем листинге для функции `sbval`. Обратите внимание, что таким же образом записано и правое краевое условие. Для того чтобы ввести условие отражения на правой границе, пришлось определить еще один неизвестный пристрелочный параметр $z2_0$. Строка листинга, в которой определена функция `score`, задает условие стрельбы — сшивку двух решений в точке `xf`. В самой последней строке листинга выдан ответ — определенные численным методом значения обоих пристрелочных параметров, которые объединены в вектор `I1` (мы применили в предпоследней строке операцию транспонирования, чтобы результат получился в форме вектора, а не матрицы-строки).

Для корректного построения графика решения лучше составить его из двух частей — решения задачи Коши на интервале `(x0,xf)` и другой задачи Коши для интервала `(xf,x1)`. Реализация этого способа приведена в листинге 10.4, который является продолжением листинга 10.3. В последней строке листинга 10.4 выведено значение второй искомой функции на правой границе интервала. Всегда полезно проконтролировать, что оно совпадает с соответствующим пристрелочным параметром (выведенным в последней строке листинга 10.3).

Листинг 10.4. Решение краевой задачи (продолжение листинга 10.3)

$$M := 10$$

$$S0 := rkfixed\left[\begin{pmatrix} I0 \\ I1_0 \end{pmatrix}, 0, 0.5, M, D\right]$$

$$S1 := rkfixed\left[\begin{pmatrix} S0_{M,1} \\ S0_{M,2} \end{pmatrix}, 0.5, 1, M, D\right]$$

$$S1_{M,2} = 13.801$$

Решение краевой задачи приведено на рис. 10.5. С физической точки зрения естественно, что интенсивность света уменьшается быстрее по мере распространения в более плотной среде в правой половине расчетного интервала. В средней точке `xf=0.5`, как и ожидалось, производные обоих решений имеют разрыв.

Примечание

Еще один пример решения краевых задач с разрывными коэффициентами ОДУ приведен в справочной системе Mathcad.

Ради справедливости необходимо заметить, что разобранную краевую задачу легко решить и с помощью функции `sbval`, заменив в листинге 10.2 зависимость `a(x)` на третью строку листинга 10.3. В этом случае листинг 10.2 даст в точности тот же ответ, что показан на рис. 10.5. Однако в определенных случаях (в том числе из соображений быстроты расчетов) удобнее использовать функцию `bvalfit`, т. е. вести пристрелку с обеих границ интервала.

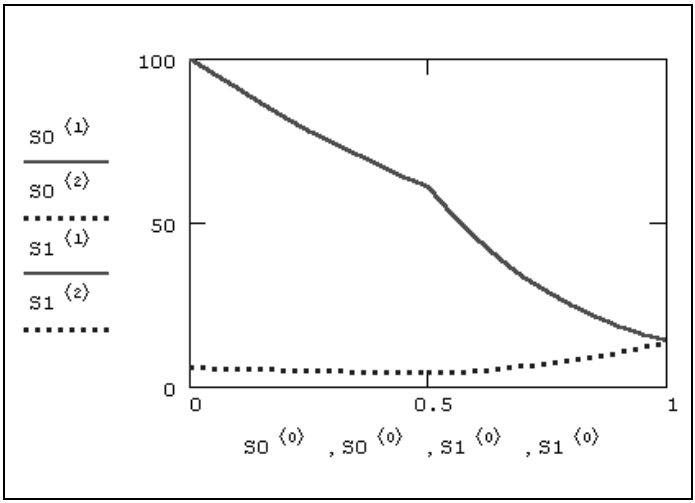


Рис. 10.5. Решение краевой задачи с разрывом в `xf=0.5` (продолжение листингов 10.3 и 10.4)

Совет

Если вы имеете дело с подобными уравнениями, попробуйте сначала решить их как обычную краевую задачу с помощью более надежной и легкой в применении функции `sbval`.

10.3. Задачи на собственные значения для ОДУ

Задачи на собственные значения — это краевые задачи для системы ОДУ, в которой правые части зависят от одного или нескольких параметров λ. Значения этих параметров неизвестны, а решение краевой задачи существует только при определенных λ_k, которые называются *собственными значениями* (eigenvalues) задачи.

Решения, соответствующие этим λ_k, называют *собственными функциями* (eigenfunctions) задачи. Правильная постановка таких задач требует формулировки количества граничных условий, равного сумме числа уравнений и числа собственных значений. Физическими примерами задач на собственные значения являются, например, уравнение колебаний струны, уравнение Шредингера в квантовой механике, уравнения волн в резонаторах и многие другие.

С вычислительной точки зрения, задачи на собственные значения очень похожи на рассмотренные выше краевые задачи. В частности, для многих из них также применим метод стрельбы (*см. разд. 10.1.2*). Отличие заключается в пристрелке не только по недостающим левым граничным условиям, но еще и по искомым собственным значениям. В Mathcad для решения задач на собственные значения используются те же функции `sbval` и `bvalfit`. В их первый аргумент, т. е. вектор, присваивающий начальные значения недостающим начальным условиям, следует включить и начальное приближение для собственного значения.

Рассмотрим методику решения на конкретном примере определения собственных упругих колебаний струны. Профиль колебаний струны `y(x)` описывается линейным дифференциальным уравнением второго порядка

$$\frac{d}{dx}\left[p(x)\cdot\frac{dy(x)}{dx}\right]+\lambda\cdot q(x)\cdot y(x)=0, \qquad (10.3)$$

где `p(x)` и `q(x)` — жесткость и плотность, которые, вообще говоря, могут меняться вдоль струны. Если струна закреплена на обоих концах, то граничные условия задаются в виде `y(0)=y(1)=0`. Сформулированная задача является частным случаем *задачи Штурма—Лиувилля*. Поскольку решается система двух ОДУ, содержащая одно собственное значение λ, то по идее задача требует задания трех (2+1) условий. Однако, как легко убедиться, уравнение колебаний струны — линейное и однородное, поэтому в любом случае решение `y(x)` будет определено с точностью до множителя. Это означает, что производную решения можно задать произвольно, например, `y'(0)=1`, что и будет третьим условием. Тогда краевую задачу можно решать как задачу Коши, а пристрелку вести только по одному параметру — собственному значению.

Процедура поиска первого собственного значения представлена в листинге 10.5.

Листинг 10.5. Решение задачи о собственных колебаниях струны

$p(x) := 1 \quad q(x) := 1 \quad p'(x) := 0$

$a := 0 \quad b := 1$

$\lambda_0 := 0.5$

$$D(x,y) := \begin{bmatrix} y_1 \\ -\dfrac{1}{p(x)}\cdot\left(p'(x)\cdot y_1 + y_2\cdot q(x)\cdot y_0\right) \\ 0 \end{bmatrix}$$

$$\mathrm{load}(a, \lambda) := \begin{pmatrix} 0 \\ 1 \\ \lambda_0 \end{pmatrix}$$

$$\mathrm{score}(b, y) := y_0$$

$$\Lambda := \mathrm{sbval}(\lambda, a, b, D, \mathrm{load}, \mathrm{score})$$

$$\Lambda = (\ 9.87\ ) \qquad 1^2 \cdot \pi^2 = 9.87$$

В первых двух строках листинга определяются функции, входящие в задачу, в том числе `p'(x):=0`, и границы расчетного интервала `(0,1)`. В третьей строке дается начальное приближение к собственному значению λ_0, в четвертой вводится система ОДУ. Обратите внимание, что она состоит не из двух, а из трех уравнений. Первые два из них определяют эквивалентную (10.3) систему ОДУ первого порядка, а третье необходимо для задания собственного значения в виде еще одного компонента y_2 искомого вектора `y`. Поскольку по определению собственное значение постоянно при всех `x`, то его производная должна быть приравнена нулю, что отражено в последнем уравнении. Важно также, что во втором из уравнений собственное значение записано как y_2, поскольку является одним из неизвестных.

В следующих двух строках листинга задается левое граничное условие, включающее и недостающее условие на собственное значение для третьего уравнения, и правое граничное условие $y_0=0$. В предпоследней строке листинга обычным образом применяется функция `sbval`, а в последней выводится результат ее работы вместе с известным аналитически собственным значением $n^2 \cdot \pi^2$. Как легко убедиться, мы нашли первое собственное значение для `n=1`, а чтобы найти другие собственные значения, необходимо задать другие начальные приближения к ним (в третьей строке листинга 10.5). Например, выбор $\lambda_0=50$ приводит ко второму собственному значению $2^2 \cdot \pi^2$, а $\lambda_0=100$ — к третьему $3^2 \cdot \pi^2$.

Чтобы построить график соответствующей собственной функции, надо добавить в листинг строку, программирующую решение задачи Коши, например, та-

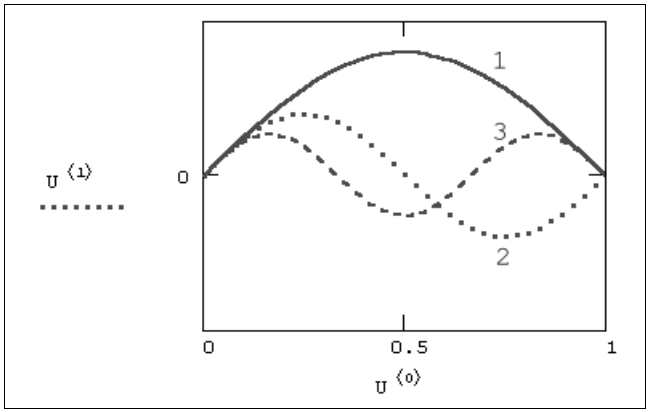


Рис. 10.6. Первые три собственные функции задачи колебаний струны (коллаж трех графиков, рассчитанных листингом 10.5 для разных $\lambda 0$)

кую: `U:=rkfixed(load(a,Λ),a,b,100,D)`. Полученные кривые показаны на рис. 10.6 в виде коллажа трех графиков, рассчитанных для трех собственных значений.

🛠 10.4. Разностные схемы для ОДУ

Многие краевые задачи не поддаются решению методом стрельбы. Однако в Mathcad 12 других встроенных алгоритмов нет. Тем не менее это не означает, что по-другому решать краевые задачи невозможно, ведь другие численные алгоритмы несложно запрограммировать самому пользователю. Рассмотрим возможную реализацию наглядного метода, называемого разностным, которым можно решать краевые задачи как для ОДУ, так и для дифференциальных уравнений в частных производных.

ⓘ 🛠 10.4.1. О разностном методе

Разберем идею разностного метода решения краевых задач на примере взаимодействия световых пучков (см. рис. 10.1), переобозначив в системе (10.1) интенсивность излучения вправо на Y, а интенсивность излучения влево на y (просто в целях удобства, чтобы не писать в дальнейшем индекс). Суть метода заключается в покрытии расчетного интервала сеткой из N точек. Тем самым определяются $(N-1)$ шагов (рис. 10.7). Затем надо заменить дифференциальные уравнения исходной краевой задачи аппроксимирующими их уравнениями в конечных разностях, выписав соответствующие разностные уравнения для каждого i-го шага. В нашем случае достаточно просто заменить первые производные из (10.1) их разностными аналогами (такой метод называется еще *методом Эйлера*):

$$\begin{aligned} \frac{Y_{i+1} - Y_i}{\Delta} &= -a(x_i)\cdot Y_i + r(x_i)\cdot y_i\,; \\ \frac{y_{i+1} - y_i}{\Delta} &= a(x_i)\cdot y_i - r(x_i)\cdot Y_i\,. \end{aligned} \tag{10.4}$$

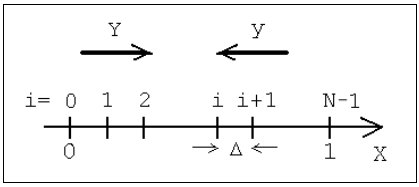


Рис. 10.7. Сетка, покрывающая расчетный интервал

Примечание

Существует множество способов аппроксимации дифференциальных уравнений разностными. От выбора конкретного варианта зависит не только простота, быстрота и удобство вычислений, но и сама возможность получения правильного ответа.

Получилась система (по числу шагов) $2\cdot(N-1)$ разностных линейных алгебраических уравнений с $2\cdot N$ неизвестными Y_i и y_i. Для того чтобы она имела единственное решение, надо дополнить число уравнений до $2\cdot N$.

Это можно сделать, записав в разностном виде оба граничных условия:

$$Y_0=I0,\ y_N=R\cdot Y_N. \tag{10.5}$$

Сформированная полная система алгебраических уравнений называется *разностной схемой*, аппроксимирующей исходную краевую задачу. Обратите внимание, что правые части разностных уравнений системы (10.4) на каждом шаге записаны для левой границы шага. Такие разностные схемы называют *явными*, т. к. все значения Y_{i+1} и y_{i+1} находятся в левой части уравнений. Полученную явную разностную схему легко записать в матричной форме

$$A\cdot z=B, \tag{10.6}$$

где z — неизвестный вектор, получающийся объединением векторов Y и y. Решив систему (10.6), мы получим решение краевой задачи.

> ***Примечание***
>
> На самом деле все несколько сложнее, поскольку, вообще говоря, необходимо еще доказать, что, во-первых, разностная схема действительно аппроксимирует дифференциальные уравнения и, во-вторых, при $N\to\infty$ разностное решение действительно сходится к дифференциальному.

Процесс решения системы разностных уравнений называют также *реализацией* разностной схемы. Программа, которая решает рассматриваемую краевую задачу разностным методом, приведена в листинге 10.6.

Листинг 10.6. Реализация явной разностной схемы

```
a(x) := 1    r(x) := 0.01    I0 := 100    R := 1

N := 5                Δ := 1/(N − 1)

i := 1 .. N − 1

A_{i,i} := a(i·Δ − Δ)·Δ − 1      A_{i,i+1} := 1      A_{i,i+N} := −Δ·r(i·Δ − Δ)

i := N + 1 .. 2·N − 1

A_{i,i} := −a(i·Δ − Δ)·Δ − 1     A_{i,i+1} := 1      A_{i,i−N} := Δ·r(i·Δ − Δ)

A_{0,1} := 1

A_{N,N·2} := 1      A_{N,N} := −R

i := 1 .. 2·N − 1

B_i := 0

B_0 := I0

A := submatrix(A, 0, 2·N − 1, 1, 2·N)

U := lsolve(A, B)

U_N = 13.453          U_{N−1} = 31.78
```

Дадим минимальные комментарии, надеясь, что заинтересовавшийся читатель с карандашом в руках разберется в порядке индексов и соответствии матричных элементов, а возможно, составит и более удачную программу.

В первой строке листинга определяются функции и константы, входящие в модель, во второй задается число точек сетки `N=5` и ее равномерный шаг. Следующие две строки определяют матричные коэффициенты, аппроксимирующие уравнения для Y_i, а пятая и шестая — для y_i. Седьмая и восьмая строки листинга задают соответственно левое и правое граничное условие, а строки с девятой по одиннадцатую — правые части системы (10.6). В следующей строке завершается построение матрицы `A` вырезанием из нее левого нулевого столбца. В предпоследней строке листинга применена встроенная функция `lsolve` для решения системы (10.6), а в последней выведены рассчитанные ею неизвестные граничные значения. Графики решения приведены на рис. 10.8, причем первые `N` элементов итогового вектора есть вычисленное излучение вперед, а последние `N` элементов — излучение назад.

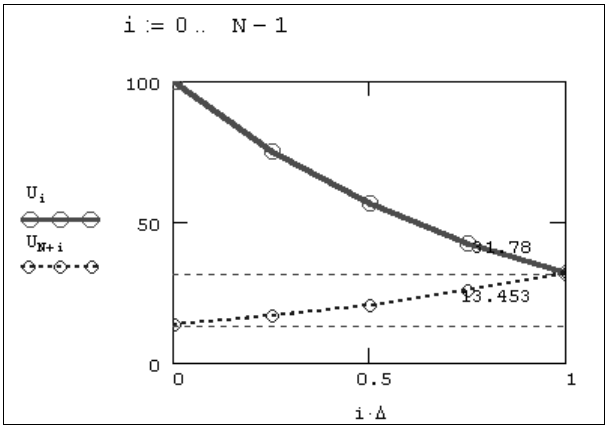


Рис. 10.8. Решение краевой задачи разностным методом (продолжение листинга 10.6)

Как мы увидели, реализация в Mathcad разностных схем вполне возможна и не слишком трудоемка — предложенная программа состоит всего из двух десятков математических выражений. Конечно, для их написания требуется и время, и часто кропотливые расчеты, но, собственно, в этом и состоит работа математика. Кстати говоря, при небольшом числе шагов расчеты по разностным схемам не требуют существенного времени (программа, приведенная в листинге 10.6, работает быстрее, чем метод стрельбы, встроенный в функцию `sbval`). Существуют, кроме того, весьма очевидные для многих читателей пути ускорения расчетов, связанные с применением более подходящих методов решения систем линейных уравнений с разреженной матрицей.

10.4.2. Жесткие краевые задачи

Один из случаев, когда применение разностных схем может быть очень полезным, связан с решением жестких краевых задач (подробнее о жестких ОДУ читайте в *главе 11*). В частности, рассматриваемая задача о встречных световых пучках становится жесткой при увеличении коэффициента ослабления `a(x)` в несколько де-

сятков раз. Например, при попытке решить ее с `a(x):=100` с помощью листинга 10.2 вместо ответа выдается сообщение об ошибке "Can't converge to a solution. Encountered too many integration steps" (Не сходится к решению. Слишком много шагов интегрирования). Это и неудивительно, поскольку жесткие системы характерны тем, что требуют исключительно малого значения шага в стандартных алгоритмах.

Для жестких задач неприменимы и явные разностные схемы, о которых рассказывалось в предыдущем разделе (*см. разд. 10.4.1*). Результат расчетов по программе листинга 10.6, например с `a(x):=20` (рис. 10.9), дает характерную для неустойчивых разностных схем "разболтку" — колебания нарастающей амплитуды, не имеющие ничего общего с реальным решением.

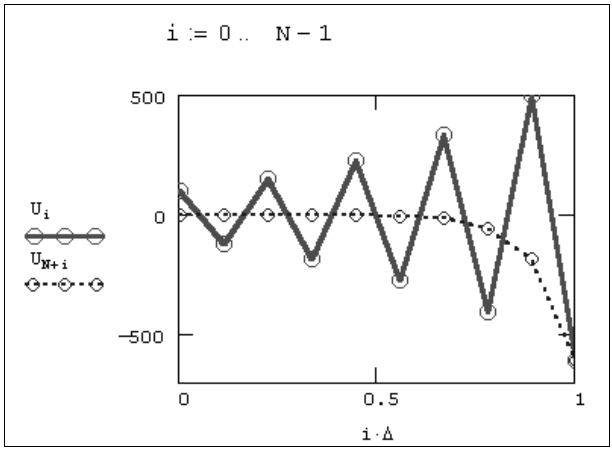


Рис. 10.9. Неверное решение жесткой краевой задачи по неустойчивой явной разностной схеме

Выходом из положения будет использование *неявных* разностных схем. Применительно к нашей задаче достаточно заменить правые части уравнений (10.1) значениями не на левой, а на правой границе каждого шага:

$$\begin{aligned} \frac{Y_{i+1} - Y_i}{\Delta} &= -a(x_{i+1}) \cdot Y_{i+1} + r(x_{i+1}) \cdot y_{i+1}\,; \\ \frac{y_{i+1} - y_i}{\Delta} &= a(x_{i+1}) \cdot y_{i+1} - r(x_{i+1}) \cdot Y_{i+1}\,. \end{aligned} \qquad (10.7)$$

Граничные условия, конечно, можно оставить в том же виде (10.5). Поскольку мы имеем дело с линейными дифференциальными уравнениями, то и схему (10.7) легко будет записать в виде матричного равенства (10.6), перегруппировывая соответствующим образом выражение (10.7) и приводя подобные слагаемые. Разумеется, полученная матрица `A` будет иной, нежели матрица `A` для явной схемы (10.4). Поэтому и решение (реализация неявной схемы) может отличаться от изображенного на рис. 10.9 результата расчетов по явной схеме. Программа, составленная для решения системы (10.7), приведена в листинге 10.7.

Листинг 10.7. Реализация неявной разностной схемы для жесткой краевой задачи

```
a(x) := 20    r(x) := 0.01    I0 := 100    R := 1

N := 20        Δ := 1/(N − 1)

i := 1 .. N − 1

A_{i,i} := 0 − 1    A_{i,i+N+1} := −Δ·r(i·Δ − Δ)    A_{i,i+1} := a(i·Δ − Δ)·Δ + 1

i := N + 1 .. 2·N − 1

A_{i,i} := 0 − 1    A_{i,i+1} := 1 − a(i·Δ − Δ)·Δ    A_{i,i−N+1} := Δ·r(i·Δ − Δ)

A_{0,1} := 1

A_{N,N·2} := 1    A_{N,N} := −R

i := 1 .. 2·N − 1

B_i := 0

B_0 := I0

A := submatrix(A, 0, 2·N − 1, 1, 2·N)

U := lsolve(A, B)

U_N = 0.017          U_{N−1} = 1.523 × 10^−6
```

Не будем специально останавливаться на обсуждении листинга 10.7, поскольку он почти в точности повторяет предыдущий листинг. Отличие заключается лишь в формировании матрицы A другим способом, согласно неявной схеме. Решение, показанное на рис. 10.10, демонстрирует, что произошло "небольшое чудо": "разболтка" исчезла, а распределение интенсивностей стало физически предсказуемым. Обратите внимание, что (из-за взятого нами слишком большого коэффициента ослабления излучения) отраженный пучок света имеет очень маленькую интенсивность, и ее пришлось построить на графике с увеличением в тысячу раз.

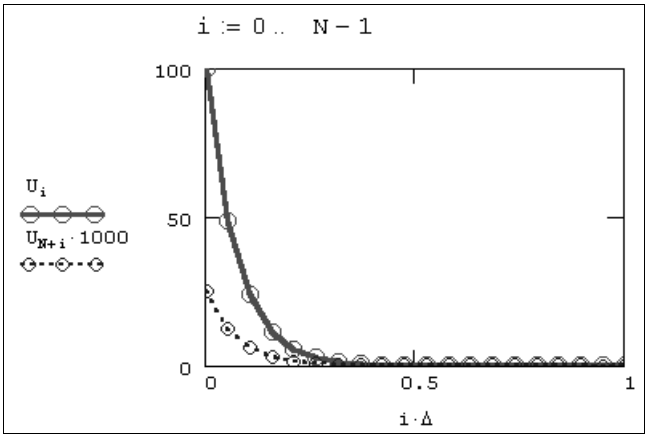


Рис. 10.10. Решение краевой задачи по неявной разностной схеме (продолжение листинга 10.7)

🛠 10.5. Нелинейные краевые задачи

Все примеры, приведенные пока в этом разделе, связаны с краевыми задачами для линейных ОДУ. Между тем на практике часто приходится исследовать именно нелинейные задачи, которые несравненно сложнее с вычислительной точки зрения. Рассмотрим оба подхода к решению нелинейных задач (алгоритм стрельбы и разностный метод), не забывая о том, что нелинейные ОДУ также могут быть в той или иной степени жесткими.

🌡 10.5.1. О постановке задачи

На примере решения нелинейной задачи взаимодействия световых пучков при помощи алгоритма стрельбы разберемся с физикой явления, которую описывает нелинейность. Рассмотрим простейшее усложнение модели (10.1) (*см. разд. 10.1*), добавив в уравнения малые слагаемые, зависящие квадратично от интенсивности света. Чтобы быть ближе к физической сущности явления, мы введем эту зависимость в формулу коэффициента поглощения среды `a(x)`, сделав `a(x)` из (10.1) функцией не только координаты, но и суммарной интенсивности:

$$a(x,Y,y)=a0(x)-\varepsilon(x)\cdot(Y+y), \tag{10.8}$$

где `a0(x)` — постоянная (не зависящая от интенсивности света) составляющая коэффициента поглощения, а `ε(x)` — малый коэффициент пропорциональности. В этом случае модель (10.1) перепишется в виде:

$$\begin{aligned} \frac{dY(x)}{dx} &= -a0(x)\cdot Y(x) + r(x)\cdot y(x) + \varepsilon(x)\cdot\left(y^2(x) + Y(x)\cdot y(x)\right); \\ \frac{dy(x)}{dx} &= a0(x)\cdot y(x) - r(x)\cdot Y(x) - \varepsilon(x)\cdot\left(y^2(x) + Y(x)\cdot y(x)\right). \end{aligned} \tag{10.9}$$

Таким образом, физическая подоплека квадратичной нелинейности задачи (10.9) связана с зависимостью коэффициента поглощения среды от `Y+y`. Иными словами, чем мощнее свет в некоторой точке среды, тем меньше локальный коэффициент поглощения среды в этой точке.

Механизм этого явления будет хорошо понятен, если предположить, что средой является воздух, насыщенный парами жидкости (попросту говоря, туман). Чем сильнее в некоторой точке туман, тем больше локальное поглощение. Если свет, проходящий через туман, имеет малую интенсивность, то он не оказывает на среду никакого (или почти никакого) воздействия. Однако если пучок света будет очень мощным, за счет его поглощения пары воды будут разогреваться и, следовательно, испаряться. По мере испарения капель воды туман станет рассеиваться, и среда просветлеет (т. е. ее коэффициент поглощения уменьшится). Таким образом, чем мощнее свет, тем активнее пойдет процесс испарения, тем существеннее уменьшится коэффициент поглощения. Именно такой эффект, совершенно понятный с физической точки зрения, и описывает формула (10.8) и, соответственно, модель (10.9).

Подчеркнем еще раз и физический смысл коэффициентов `a0(x)` и `ε(x)`. Первый описывает постоянный (не зависящий от интенсивности света) коэффициент по-

глощения среды, а второй — его зависимость от эффекта разогрева среды светом. Чем больше коэффициент `a0(x)`, тем более жесткой является краевая задача; а чем больше `ε(x)`, тем сильнее ее нелинейность. В предельном случае `ε(x)→0` задача (10.9) переходит в задачу (10.1), становясь линейной.

10.5.2. Метод стрельбы

Встроенная функция `sbval`, реализующая в Mathcad алгоритм стрельбы (*см. разд. 10.2*), может справляться и с нелинейными задачами. Приведем пример решения краевой задачи (10.9) (листинг 10.8 и рис. 10.11) с теми же граничными условиями, что были поставлены для модели (10.1). Интерес представляет решение `y`, описывающее рост интенсивности отраженного пучка по мере его распространения справа налево (обратите внимание, что на рис. 10.11 оно отложено с десятикратным увеличением).

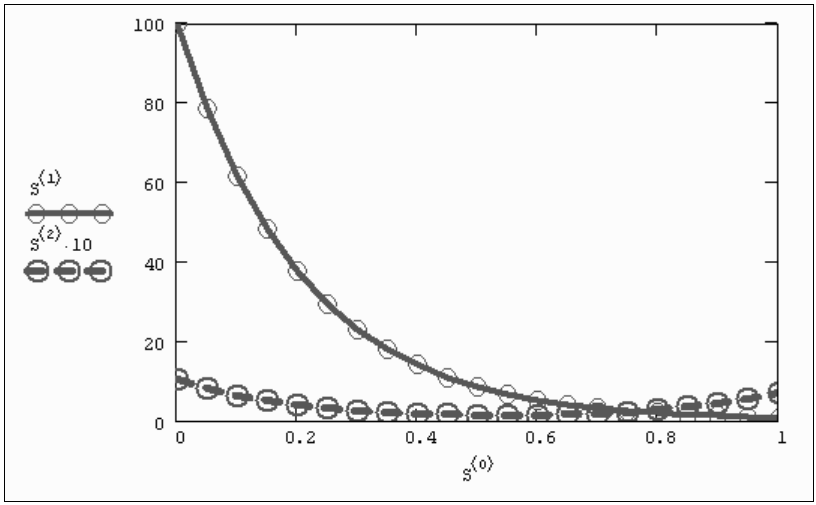


Рис. 10.11. Решение нелинейной краевой задачи (продолжение листинга 10.8)

Листинг 10.8. Решение нелинейной краевой задачи методом стрельбы

$I0 := 100$ $\qquad r(x) := .1$

$\varepsilon(x) := 0.002$ $\qquad a(x) := 5$

$R := 1$

$$D(x, y) := \begin{bmatrix} -a(x) \cdot y_0 + r(x) \cdot y_1 + \varepsilon(x) \cdot |y_0 + y_1| \cdot y_0 \\ a(x) \cdot y_1 - r(x) \cdot y_0 - \varepsilon(x) \cdot |y_0 + y_1| \cdot y_1 \end{bmatrix}$$

$z_0 := 10$

$$load(x0, z) := \begin{pmatrix} 100 \\ z_0 \end{pmatrix}$$

score (x1 , y) := $R \cdot y_0 - y_1$

I1 := sbval (z , 0 , 1 , D , load , score)

I1 = (1.033)

$$S := \mathrm{rkfixed}\left[\begin{pmatrix} I0 \\ I1_0 \end{pmatrix}, 0, 1, 20, D\right]$$

Следует помнить, что чем сильнее нелинейность и чем жестче краевая задача, тем более сильные требования предъявляются к начальному значению алгоритма, ввод которого осуществляется посредством функции `load`. Попробуйте повторить расчеты листинга 10.8 с иным сочетанием параметров `a0(x)` и `ε(x)`, и вы увидите, что с немного более жесткими задачами алгоритм стрельбы уже не справляется.

Примечание

На компакт-диске, прилагаемом к книге, вы найдете еще более интересное решение для другой задачи, с нелинейностью типа $(y+Y)^2$.

10.5.3. Разностные схемы

Решение краевых задач при помощи разностных схем связано с необходимостью разработки собственного алгоритма для каждой конкретной задачи. Как вы помните (*см. разд. 10.4*), в случае линейных уравнений в результате построения разностной схемы система алгебраических сеточных уравнений также получалась линейной. Это автоматически означало, что она имеет единственное решение, которое в большинстве случаев могло быть найдено стандартными численными методами. В ситуации, когда исходные дифференциальные уравнения нелинейны, система разностных уравнений тоже является нелинейной, и их решение существенно усложняется, хотя бы потому, что оно не является единственным. Поэтому подход к построению разностных схем нелинейных уравнений должен быть специфическим, но наградой за него станет решение задач, с которыми не справляется алгоритм стрельбы (например, жестких).

Подробная разработка алгоритмов и соответствующих программ Mathcad выходит далеко за пределы данной книги, поэтому мы конспективно представим один из приемов решения нелинейных краевых задач, сводящийся к их *линеаризации*. В общих чертах подход заключается в следующем. Предположим, что некоторое приближение (обозначим его `Y(x)` и `y(x)`) к решению нелинейной задачи (10.9) нам известно, и можно считать, что $Y \to Y+Z$ и $y \to y+z$, где `Z` и `z` — близкие к нулю функции `x`. Тогда, пользуясь их малостью (по сравнению с `Y0` и `y0`), можно разложить нелинейные слагаемые в уравнениях (10.9) в ряд Тейлора по `Z` и `z`. Получим:

$$\begin{aligned} Y'+Z' &= -aY + ry + \varepsilon(Y^2 + Yy) - aZ + rz + 2\varepsilon Y(Z + z); \\ y'+z' &= ay - rY - \varepsilon(y^2 + Yy) + az - rZ - 2\varepsilon y(Z + z). \end{aligned} \qquad (10.10)$$

Теперь, поскольку `Y(x)` и `y(x)` являются приближением к решению исходной задачи, то можно считать, что они (приблизительно) удовлетворяют и уравнению, и

граничным условиям. Тогда, вычитая (10.9) из (10.10), получим краевую задачу для `Z(x)` и `z(x)`:

$$\begin{aligned} Z' &= -aZ + rz + 2\varepsilon Y(Z + z); \\ z' &= az - rZ - 2\varepsilon y(Z + z); \\ Z(0) &= z(0) = Z(1) = z(1) = 0. \end{aligned} \tag{10.11}$$

Это и есть та самая новая задача для "маленьких" функций `Z(x)` и `z(x)`, которую надо решить, и которая, благодаря малости `Z` и `z`, является линейной. Вся беда в том, что мы не знаем "больших" функций `Y(x)` и `y(x)`, а они, увы, входят в задачу (10.11). Тем не менее рецепт получения этих функций напрашивается сам собой: если нелинейность исходных ОДУ не слишком сильная, то в качестве `Y` и `y` можно взять решение линейной краевой задачи, т. е. задачи (10.9) с `ε(x)=0` (*см. разд. 10.4.1*).

Сказанное иллюстрируют листинги 10.9 и 10.10, первый из которых решает линейную краевую задачу (являясь, фактически, небольшой модификацией листинга 10.7 из *разд. 10.4.2*), а второй решает линеаризованную задачу (10.11), учитывая результат листинга 10.9. В листинге 10.9 матрица `D` и вектор правых частей `B` являются разностной аппроксимацией ОДУ (ее первые `N` строк аппроксимируют первое уравнение, а оставшиеся `N` строк — второе). Такой же смысл и точно такую же структуру имеют матрица `C` и вектор правых частей `G` для второй (линеаризованной) задачи (10.11). Для решения сеточных уравнений `Dy=B` и `Cz=G` используется (конечно, весьма неэкономично) встроенная функция `lsolve`, реализующая алгоритм Гаусса.

Важно привлечь внимание читателя к последним строкам листинга 10.9. В них осуществляется интерполяция полученного решения системы сеточных уравнений для того, чтобы в нелинейной задаче (в листинге 10.10) можно было использовать непрерывные "большие" функции из линейной задачи. В последней строке листинга 10.10 осуществляется сложение "больших" и "маленьких" функций (результатов листингов 10.9 и 10.10) для получения полного решения нелинейной задачи (10.9), которое изображено на рис. 10.12.

Не будем давать дополнительных комментариев к Mathcad-программам, надеясь, что читатель, заинтересовавшийся нелинейным примером со световыми пучками и эффектом разогрева светом среды, сам разберется в листингах, тем более что техника разработки разностных схем была нами детально разобрана раньше (*см. разд. 10.4*).

Листинг 10.9. Решение линейной (приближенной) краевой задачи

```
I0 := 100          R := 1          ε(x) := 0.001
a(x) := 5          r(x) := .1
                   Δ := 1/(N − 1)
N := 20
```

$D_{0,1} := 1$ $\quad D_{N,N\cdot 2} := 1$ $\quad D_{N,N} := -R$

$i := 1 .. \; N-1$

$D_{i,i} := a\left(i\cdot\Delta - \Delta\right)\cdot\frac{\Delta}{2} - 1$ $\quad D_{i,i+1} := a\left(i\cdot\Delta - \Delta\right)\cdot\frac{\Delta}{2} + 1$

$D_{i,i+N+1} := \frac{-\Delta}{2}\cdot r\left(i\cdot\Delta - \Delta\right)$ $\quad D_{i,i+N} := \frac{-\Delta}{2}\cdot r\left(i\cdot\Delta - \Delta\right)$

$i := N+1 .. \; 2\cdot N-1$

$D_{i,i} := -a\left(i\cdot\Delta - \Delta\right)\cdot\frac{\Delta}{2} - 1$ $\quad D_{i,i+1} := 1 - a\left(i\cdot\Delta - \Delta\right)\cdot\frac{\Delta}{2}$

$D_{i,i-N} := \frac{\Delta}{2}\cdot r\left(i\cdot\Delta - \Delta\right)$ $\quad D_{i,i-N+1} := \frac{\Delta}{2}\cdot r\left(i\cdot\Delta - \Delta\right)$

$B_0 := I0$

$i := 1 .. \; 2\cdot N-1$

$B_i := 0$

$D := \mathrm{submatrix}(D, 0, 2\cdot N-1, 1, 2\cdot N)$

$I := \mathrm{lsolve}(D, B)$

$i := 0 .. \; N-1$

$Z1_i := i\cdot\Delta$ $\quad Z2_i := I_i$

$Z3_i := I_{i+N}$ $\quad Z4_i := I_i + I_{i+N}$

$KS1 := \mathrm{cspline}(Z1, Z2)$ $\quad I1(z) := \mathrm{interp}(KS1, Z1, Z2, z)$

$KS2 := \mathrm{cspline}(Z1, Z3)$ $\quad I2(z) := \mathrm{interp}(KS2, Z1, Z3, z)$

$KS3 := \mathrm{cspline}(Z1, Z4)$ $\quad S(z) := \mathrm{interp}(KS3, Z1, Z4, z)$

$DI1(x) := \frac{d}{dx} I1(x)$ $\quad DI2(x) := \frac{d}{dx} I2(x)$

Листинг 10.10. Решение линеаризованной задачи (продолжение листинга 10.9)

$C_{0,1} := 1$ $\quad C_{N,N\cdot 2} := 1$ $\quad C_{N,N} := -R$

$G_0 := 0$ $\quad G_N := R\cdot I1(1) - I2(1)$

$i := 1 .. \; N-1$

$C_{i,i} := 0 - 1$

$C_{i,i+1} := a\left(i\cdot\Delta\right)\cdot\Delta - 2\cdot\varepsilon\left(i\cdot\Delta\right)\cdot S\left(i\cdot\Delta\right)\cdot\Delta + 1$

$C_{i,i+N+1} := -\Delta\cdot r\left(i\cdot\Delta\right) - 2\cdot\varepsilon\left(i\cdot\Delta\right)\cdot S\left(i\cdot\Delta\right)\cdot\Delta$

$G_i := \Delta\cdot\left(-1\cdot a\left(i\cdot\Delta\right)\cdot I1\left(i\cdot\Delta\right) + 1\cdot r\left(i\cdot\Delta\right)\cdot I2\left(i\cdot\Delta\right) + \varepsilon\left(i\cdot\Delta\right)\cdot S\left(i\cdot\Delta\right)^2\right) +$

$\quad + -1\cdot\Delta\cdot DI1\left(i\cdot\Delta\right)$

$i := N+1 .. \; 2\cdot N-1$

$C_{i,i} := 0 - 1$

$C_{i,i+1} := 1 - a(i\cdot\Delta)\cdot\Delta + 2\cdot\varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta - N\cdot\Delta)\cdot\Delta$

$C_{i,i-N+1} := \Delta\cdot r(i\cdot\Delta) + 2\cdot\varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta - N\cdot\Delta)\cdot\Delta$

$G_i := \Delta\cdot\left(a(i\cdot\Delta)\cdot I2(i\cdot\Delta - N\cdot\Delta)\cdot 1 - 1\cdot r(i\cdot\Delta)\cdot I1(i\cdot\Delta - N\cdot\Delta) - \right.$

$\left. - \varepsilon(i\cdot\Delta)\cdot S(i\cdot\Delta - N\cdot\Delta)^2\right) + -1\cdot\Delta\cdot DI2(i\cdot\Delta - N\cdot\Delta)$

$C := \mathrm{submatrix}(C, 0, 2\cdot N - 1, 1, 2\cdot N)$

$JK := \mathrm{lsolve}(C, G)$

$i := 0 .. N - 1$

$J_i := JK_i + I1(i\cdot\Delta)$

$J_{N+i} := JK_{N+i}\cdot 1 + I2(i\cdot\Delta)$

Последний важный момент, который следует обозначить, связан с решением задач, обладающих значительной нелинейностью. Решение, приведенное в листингах 10.9, 10.10 и на рис. 10.12, согласно самой постановке, должно не сильно отличаться от решения линейной краевой задачи, поскольку функции `Z(x)` и `z(x)` малы по сравнению с `Y(x)` и `y(x)`. Если же нелинейность сильная, то решение может значительно отличаться от `Y` и `y`, и линеаризация (10.11) будет просто неправильной. В этом случае следует слегка усложнить алгоритм решения нелинейной краевой задачи.

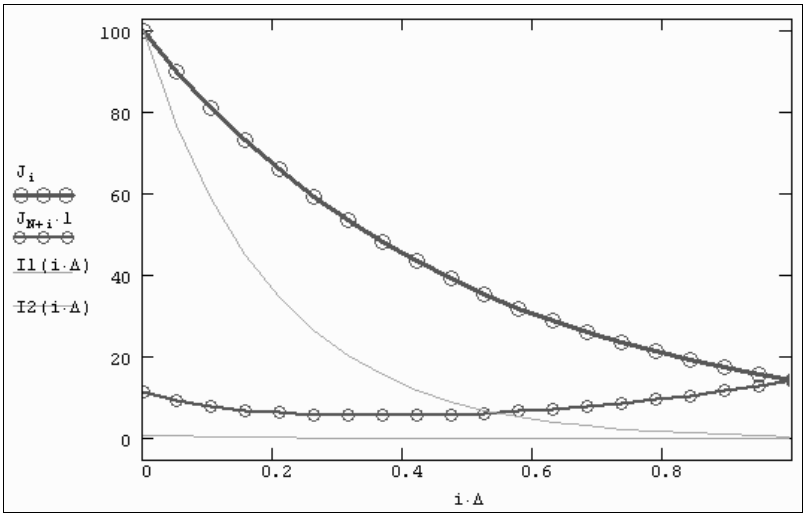


Рис. 10.12. Решение нелинейной краевой задачи разностным методом (продолжение листингов 10.9 и 10.10)

Обозначим полученное в результате решение, как и в листинге 10.10, вектором `J(ε)`, подчеркивая тем самым его зависимость от параметра нелинейности. Очевидно, что `J(0)` есть решение линейной задачи. Для того чтобы решить задачу с сильной нелинейностью, т. е. довольно большим ε=ε1, можно организовать продолжение по ε как

по параметру. Иными словами, используя в качестве начального приближения `J(0)`, можно решить задачу для другого, малого $\varepsilon=\Delta\varepsilon$, получив `J(Δε)`, затем, взяв это `J(Δε)` в качестве приближенного решения, получить `J(2Δε)` и т. д. малыми шагами добраться до желаемого `ε1`.

Примечание

Упрощенную реализацию этого алгоритма вы найдете на компакт-диске, прилагаемом к книге. Она связана с выводом во внешний файл данных результата задачи из листинга 10.10 и считыванием из него же этих данных в качестве входной информации для следующей итерации. В качестве нулевой итерации используется решение линейной задачи, выводимое предварительно в файл из усовершенствованного листинга 10.9.

ГЛАВА 11

Дифференциальные уравнения в частных производных

Дифференциальные уравнения в частных производных представляют собой одну из наиболее сложных и одновременно интересных задач вычислительной математики. Эти уравнения характеризуются тем, что для их решения не существует единого универсального алгоритма, и большинство задач требует своего собственного особого подхода. Уравнениями в частных производных описывается множество разнообразных физических явлений, и с их помощью можно с успехом моделировать самые сложные явления и процессы (диффузия, гидродинамика, квантовая механика, экология и т. д.).

Дифференциальные уравнения в частных производных требуют нахождения функции не одной, как для ОДУ, а нескольких переменных, например, `f(x,y)` или `f(x,t)`. Постановка задач (*см. разд. 11.1*) включает в себя само уравнение (или систему уравнений), содержащее производные неизвестной функции по различным переменным (частные производные), а также определенное количество краевых условий на границах расчетной области.

Несмотря на то, что Mathcad обладает довольно ограниченными возможностями по отношению к уравнениям в частных производных, в нем имеется несколько встроенных функций (*см. разд. 11.3*). Решать уравнения в частных производных можно и путем непосредственного программирования пользовательских алгоритмов (*см. разд. 11.2*). Автор совершенно сознательно сначала рассматривает численные методы для решения уравнений в частных производных, а уже затем описывает предназначенные для этого встроенные функции, чтобы читатель ясно осознавал, каким образом Mathcad производит расчеты. "Слепое" использование встроенных функций для решения уравнений в частных производных не всегда бывает успешным, и ответственность за верный выбор их параметров часто ложится на пользователя, которому необходимо четко представлять основные принципы функционирования численных алгоритмов, примененных во встроенных функциях.

11.1. О постановке задач

Остановимся сначала на общей классификации уравнений в частных производных, принятой в математике, а затем более детально поговорим о постановочной части соответствующих задач на примере различных вариаций уравнения диффузии тепла, которое допускает наглядную физическую интерпретацию.

11.1.1. Классификация уравнений в частных производных

Постановка задач для уравнений в частных производных включает определение самого уравнения (или системы нескольких уравнений), а также необходимого количества краевых условий (число и характер задания которых определяются спецификой уравнения). По своему названию уравнения должны содержать частные производные неизвестной функции `u` (или нескольких функций, если уравнений несколько) по различным аргументам, например пространственной переменной `x` и времени `t`. Соответственно, для решения задачи требуется вычислить функцию нескольких переменных, например, `u(x,t)` в некоторой области определения аргументов $0 \le x \le L$ и $0 \le t \le T$. Граничные условия определяются как заданные временные зависимости функции `u`, или производных этой функции, на границах расчетной области `0` и `L`, а начальные — как заданная `u(x,0)`.

Сами уравнения в частных производных (несколько условно) можно разделить на три основных типа:

- *параболические* — содержащие первую производную по одной переменной и вторую — по другой, причем все эти производные входят в уравнение с одинаковым знаком;
- *гиперболические* — содержащие первую производную по одной переменной и вторую — по другой, входящие в уравнение с разными знаками;
- *эллиптические* — содержащие только вторые производные, причем одного знака.

Некоторые более сложные уравнения нельзя однозначно подогнать под приведенную классификацию, тогда говорят о гибридных типах уравнений.

11.1.2. Пример: уравнение диффузии тепла

На протяжении всей главы мы будем использовать в качестве примера очень наглядное и имеющее различные, от очевидных до самых неожиданных, решения уравнение теплопроводности.

Двумерное динамическое уравнение

Рассмотрим следующее параболическое уравнение в частных производных, зависящее от трех переменных — двух пространственных `x` и `y`, а также от времени `t`:

$$\frac{\partial u(x, y, t)}{\partial t} = D\left(\frac{\partial^2 u(x, y, t)}{\partial x^2} + \frac{\partial^2 u(x, y, t)}{\partial y^2}\right) + \phi(x, y, t, u). \qquad (11.1)$$

Примечание

Выражение в скобках в правой части уравнения (сумму вторых пространственных производных функции `u` часто, ради краткости, обозначают при помощи *оператора Лапласа*: Δu).

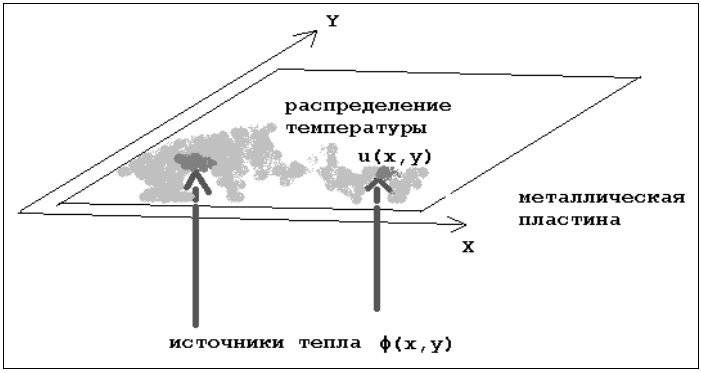


Рис. 11.1. Физическая модель, описываемая двумерным уравнением теплопроводности

Это уравнение называется *двумерным уравнением теплопроводности* или, по-другому, *уравнением диффузии тепла*. Оно описывает динамику распределения температуры `u(x,y,t)` на плоской поверхности (например, на металлической пластине) в зависимости от времени (рис. 11.1). Физический смысл коэффициента `D`, который, вообще говоря, может быть функцией как координат, так и самой температуры, заключается в задании скорости перетекания тепла от более нагретых областей в менее нагретые. Функция `ϕ(x,y,t,u)` описывает приток тепла извне, т. е. источники тепла, которые также могут зависеть как от пространственных координат (что задает локализацию источников), так и от времени и от температуры `u`.

Для того чтобы правильно поставить краевую задачу для двумерного уравнения теплопроводности, необходимо определить следующие дополнительные условия:

- граничные условия, т. е. динамику функции `u(x,y,t)` и (или) ее производных на границах расчетной области;
- начальное условие, т. е. функцию `u(x,y,t)`.

Примечание

Если рассматривается не одно уравнение в частных производных, а система уравнений, то соответствующие начальные и граничные условия должны быть поставлены для каждой из неизвестных функций.

Стационарное двумерное уравнение

Частный случай уравнения теплопроводности определяет стационарную, т. е. не зависящую от времени задачу. Стационарное уравнение описывает физическую картину распределения температуры по пластине, не изменяющуюся с течением времени. Такая картина может возникнуть при условии, что стационарный источник тепла действует довольно продолжительное время, и переходные процессы,

вызванные его включением, прекратились. Пример численного решения такого уравнения показан на рис. 11.2 в виде поверхности `u(x,y)`.

Как несложно сообразить, если искомая функция не зависит от времени, то частная производная по времени в левой части уравнения равна нулю, и само уравнение можно переписать (переобозначив ради упрощения $\phi \leftarrow \phi/D$) следующим образом:

$$\frac{\partial^2 u(x, y)}{\partial x^2} + \frac{\partial^2 u(x, y)}{\partial y^2} = -\phi(x, y, u). \qquad (11.2)$$

Полученное уравнение, согласно классификации предыдущего раздела, является эллиптическим. Его называют *уравнением Пуассона*, а для его решения в Matcad предусмотрены две встроенные функции. Если к тому же источники равны нулю, то уравнение (11.2), принимающее вид $\Delta u=0$, называют *уравнением Лапласа*.

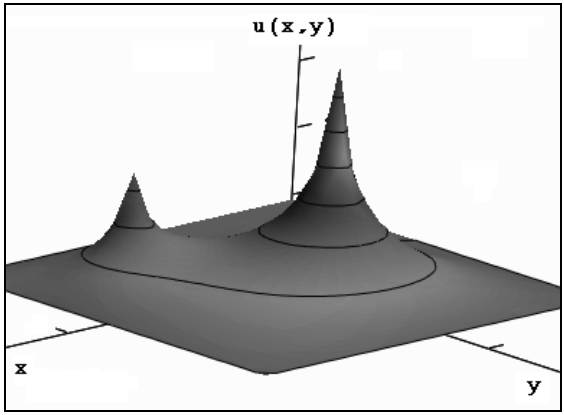


Рис. 11.2. Решение стационарного двумерного уравнения теплопроводности (см. листинг 11.7 ниже)

Одномерное динамическое уравнение

Предположим, что мы рассматриваем задачу распределения тепла не по плоской поверхности, а по удлиненному телу типа металлического стержня (рис. 11.3). В этом случае зависимость от координаты `y` в общем уравнении теплопроводности пропадает и получается одномерное уравнение:

$$\frac{\partial u(x, t)}{\partial t} = D\,\frac{\partial^2 u(x, t)}{\partial x^2} + \phi(x, t, u). \qquad (11.3)$$

Одномерное уравнение намного проще двумерного, поскольку объем вычислений для реализации алгоритма его численного решения не так велик. Типичное решение одномерного уравнения диффузии тепла с коэффициентом диффузии `D=2`, нулевым источником $\phi=0$ и начальным распределением температуры в форме нагретой центральной области стержня показано (в виде графика поверхности) на рис. 11.4.

Начиная с версии Mathcad 11, для решения одномерных параболических и гиперболических уравнений можно применять встроенную функцию `pdesolve`.

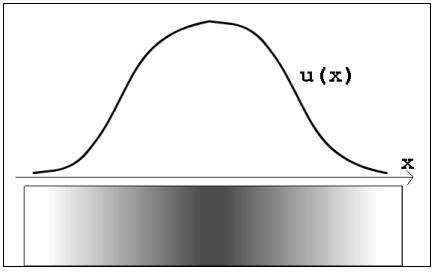


Рис. 11.3. Физическая модель одномерного уравнения теплопроводности

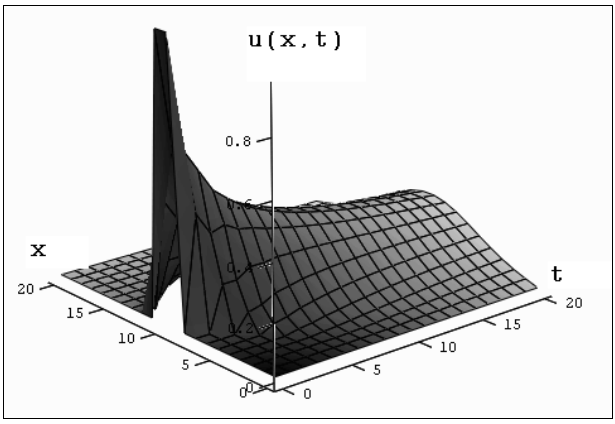


Рис. 11.4. Решение одномерного уравнения теплопроводности (см. листинг 11.1 ниже)

Линейное и нелинейное уравнения

Если присмотреться к уравнению диффузии тепла внимательнее, то можно условно разделить практические случаи его использования на два типа.

- Линейная задача — если коэффициент диффузии `D` не зависит от температуры `u` и, кроме того, если источник тепла ϕ либо также не зависит от `u`, либо зависит от `u` линейно. В этом случае неизвестная функция `u(x,t)` и все ее производные входят в уравнение только в первой степени (линейно).
- Нелинейная задача — если уравнение имеет нелинейную зависимость от `u(x,t)`, т. е. или коэффициент диффузии зависит от `u`, и/или источник тепла нелинейно зависит от `u`.

Решения линейных уравнений в частных производных, как правило, получаются вполне предсказуемыми, и их часто можно получить аналитически (этим проблемам посвящены соответствующие разделы науки, называемой математической физикой). В случае уравнения теплопроводности линейная задача описывает физически ожидаемое решение, выражающее остывание пластины или стержня в форме перетекания тепла от нагретого центра к холодной периферии.

Нелинейные уравнения, напротив, могут демонстрировать самые неожиданные решения, причем в подавляющем большинстве практических задач их можно получить только численно, а никак не аналитически.

Примечание

Различные линейные и нелинейные варианты рассматриваемого уравнения теплопроводности описывают различные модели физических сред, которые характеризуются определенными зависимостями `D(u)` и ϕ`(u)`. В частности, для металлов в большинстве случаев можно считать, что `D=const`, в то время как для плазмы имеется специфическая зависимость коэффициента диффузии от температуры.

Обратное уравнение теплопроводности

Замечательными свойствами обладает так называемое обратное уравнение диффузии тепла, которое получается путем замены в исходном (прямом) уравнении переменной `t` на `-t`. Согласно постановке задачи, обратное уравнение теплопроводности описывает реконструкцию динамики профиля температуры остывающего стержня, если известно начальное условие в виде профиля температуры в некоторый момент времени после начала остывания. Таким образом, требуется определить, как происходило остывание стержня. Мы ограничимся самым простым линейным уравнением с `D=const` без источников тепла:

$$\frac{\partial u(x,t)}{\partial t} = -D\frac{\partial^2 u(x,t)}{\partial x^2}. \qquad (11.4)$$

Это уравнение гиперболического типа и оно, несмотря на кажущуюся близость к рассмотренным вариантам уравнения теплопроводности, обладает замечательными свойствами.

Если попробовать осуществить расчет обратного уравнения диффузии тепла по тем же самым алгоритмам, что и для обычных уравнений (для этого достаточно в листинге 11.1 или 11.2 заменить значение коэффициента диффузии на отрицательное число, например, `D=-1`), то мы получим заведомо нефизичное решение. Оно показано на рис. 11.5 в виде профилей распределения температуры для нескольких последовательных моментов времени. Как видно, решение выражается в появлении все более быстрых пространственных осцилляций профиля температуры для каждого нового момента времени. Очень существенно, что такое решение является не проявлением неустойчивости численного алгоритма (как для ситуации, рассмотренной в *разд. "Устойчивость" этой главы*), а определяется спецификой самой задачи.

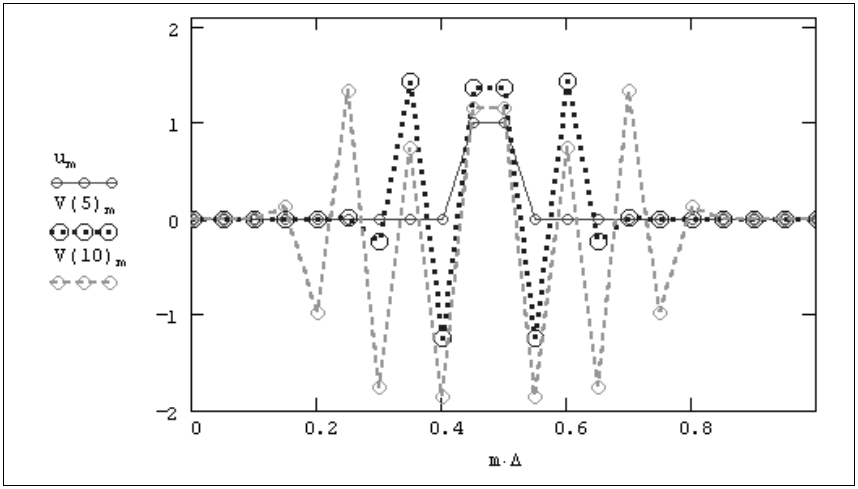


Рис. 11.5. Численное решение обратного уравнения теплопроводности дает совершенно нефизичную картину динамики температуры (см. листинг 11.2 ниже с параметром `D=-1`)

Оказывается, что обратное уравнение теплопроводности принадлежит к довольно широкому классу задач, называемых некорректными. Некорректные задачи нельзя решать стандартными методами, а для того, чтобы с ними справиться (т. е., чтобы получить осмысленное физическое решение), приходится несколько менять саму их постановку, вводя в нее дополнительную априорную информацию о строении решения.

ⓘ 🛠 11.2. Разностные схемы

Рассмотрим одномерное уравнение теплопроводности (11.3) и на его примере разберем наиболее часто использующийся для численного решения уравнений в частных производных *метод сеток*.

Выпишем еще раз само уравнение

$$\frac{\partial u(x,t)}{\partial t} = D\frac{\partial^2 u(x,t)}{\partial x^2} + \phi(x,t,u), \tag{11.5}$$

а также и начальное

$$u(x,0) = u_0(x), \tag{11.6}$$

и граничные условия

$$u(0,t) = f_0(t), \qquad u(L,t) = f_1(t), \tag{11.7}$$

которые необходимы для правильной, с математической точки зрения, постановки задачи.

Основная идея численного решения уравнений в частных производных очень похожа на метод решения краевых задач для ОДУ, рассмотренный нами в предыдущей главе. Основным отличием от ОДУ является необходимость дискретизации уравнения не по одной, а по нескольким переменным (в зависимости от размерности задачи).

Таким образом, сначала следует покрыть расчетную область (x,t) *сеткой* и использовать затем узлы этой сетки для разностной аппроксимации уравнения. В результате, вместо поиска непрерывных зависимостей $u(x,t)$ достаточно будет отыскать значения функции в узлах сетки (а ее поведение в промежутках между узлами может быть получено при помощи построения какой-либо интерполяции). По этой причине дискретное представление функции u часто называют *сеточной функцией*.

Поскольку уравнения в частных производных по определению зависят от производных неизвестных функций по нескольким переменным, то способов дискретизации этих уравнений может быть, как правило, несколько. Конфигурацию узлов, используемую для разностной записи уравнений в частных производных на сетке, называют *шаблоном*, а полученную систему разностных уравнений — *разностной схемой*. О принципах построения разностных схем и, в частности, о классах явных и неявных схем мы уже подробно говорили на примере краевых задач для ОДУ (*см. разд. 10.4*), поэтому, излишне не повторяясь, перейдем к рассмотрению типичных особенностей уравнений в частных производных, которые возникают при разработке и реализации разностных схем.

11.2.1. Явная схема Эйлера

Рассмотрим сначала математические аспекты построения разностной схемы для уравнения диффузии тепла, а затем приведем примеры работы разработанного алгоритма применительно к линейному и нелинейному уравнениям.

Построение разностной схемы

Используем для решения уравнения теплопроводности шаблон, изображенный на рис. 11.6. Для дискретизации второй производной по пространственной координате необходимо использовать три последовательных узла, в то время как для разностной записи первой производной по времени достаточно двух узлов. Записывая на основании данного шаблона дискретное представление для (i,k)-го узла, получим разностное уравнение:

$$\frac{u_{i,k+1} - u_{i,k}}{\Delta_t} = D\,\frac{u_{i-1,k} - 2\,u_{i,k} + u_{i+1,k}}{\Delta_x^2} + \phi_{i,k}. \qquad (11.8)$$

Приведем в разностной схеме (11.8) подобные слагаемые, перенеся в правую часть значения сеточной функции с индексом k (как часто говорят, с предыдущего *слоя* по времени), а в левую — с индексом $k+1$ (т. е. со следующего временного слоя). Кроме этого, введем коэффициент C, который будет характеризовать отношение шагов разностной схемы по времени и пространству $C_{i,k} = \dfrac{2 \cdot \Delta_t \cdot D_{i,k}}{\Delta_x^2}$. Несколько забегая вперед, заметим, что значение параметра C, называемого *коэффициентом Куранта*, имеет большое значение для анализа устойчивости разностной схемы. С учетом этих замечаний, разностная схема (11.8) запишется в виде:

$$u_{i,k+1} = \frac{C_{i,k}}{2}\,u_{i-1,k} + (1 - C_{i,k})\,u_{i,k} + \frac{C_{i,k}}{2}\,u_{i+1,k} + \varphi_{i,k}. \qquad (11.9)$$

Множители для каждого из значений сеточной функции в узлах шаблона, соответствующие разностному уравнению (11.9), приведены рядом с каждой точкой шаблона на рис. 11.6. Фактически геометрия шаблона и эти множители задают построенную нами разностную схему.

Несложно убедиться в том, что для получения замкнутой системы разностных алгебраических уравнений систему (11.9) необходимо дополнить дискретным представлением начального и граничных условий (11.6) и (11.7). Тогда число неиз-

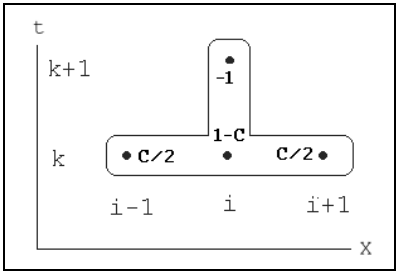


Рис. 11.6. Шаблон аппроксимации явной схемы для уравнения теплопроводности

вестных будет в точности равно числу уравнений, и процесс формирования разностной схемы будет окончательно завершен.

> ***Примечание***
>
> Важно подчеркнуть, что возможная нелинейность полученной системы алгебраических уравнений определяется зависимостями от температуры функций `D(u)` и ϕ`(u)`, т. е. как коэффициент диффузии, так и источник тепла могут быть функциями сеточной функции u_{ik}.

Если присмотреться к разностным уравнениям (11.9) повнимательнее, то можно сразу предложить несложный алгоритм реализации этой разностной схемы. Действительно, каждое неизвестное значение сеточной функции со следующего временного слоя, т. е. левая часть соотношения (11.9) явно выражается через три ее значения с предыдущего слоя (правая часть), которые уже известны. Таким образом, в случае уравнения теплопроводности нам очень повезло. Для расчета первого слоя по времени следует попросту подставить в (11.9) начальное условие (известные значения `u` с нулевого слоя в узлах сетки), для расчета второго слоя достаточно использовать вычисленный таким образом набор `u` с первого слоя и т. д. Из-за того что разностная схема сводится к такой явной подстановке, ее и называют *явной*, а благодаря пересчету значений с текущего слоя через ранее вычисленные слои — *схемой бегущего счета*.

Линейное уравнение

Сделанные замечания относительно реализации явной схемы для уравнения диффузии тепла сразу определяют алгоритм ее программирования в Mathcad. Для решения задачи нужно аккуратно ввести в листинг соответствующие формулы при помощи элементов программирования.

Решение системы разностных уравнений (11.9) для модели без источников тепла, т. е. ϕ`(x,T,t)=0`, и постоянного коэффициента диффузии `D=const` приведено в листинге 11.1. В его первых трех строках заданы шаги по временной и пространственной переменным τ и Δ, а также коэффициент диффузии `D`, равный единице. В следующих двух строках заданы начальные (нагретый центр области) и граничные (постоянная температура на краях) условия соответственно. Затем приводится возможное программное решение разностной схемы, причем функция пользователя `V(t)` задает вектор распределения искомой температуры в каждый момент времени (иными словами, на каждом слое), задаваемый целым числом `t`.

Листинг 11.1. Явная схема для линейного уравнения теплопроводности

$\tau := 0.0005 \qquad M := 20$

$\Delta := \frac{1}{M}$

$D(u) := 1$

$\phi(x, u) := 0$

$$\mathrm{Init}(x) := \Phi(x-0.45) - \Phi(x-0.55)$$

$$\mathrm{Border}(t) := 0$$

$$m := 0 .. M$$

$$u_m := \mathrm{Init}(m\cdot\Delta)$$

$$F(v) := \left| \begin{array}{l} v1_0 \leftarrow \mathrm{Border}(\tau\cdot T) + 0.\cdot(v_0 + v_1) \\ v1_M \leftarrow \mathrm{Border}(\tau\cdot T) + 0.\cdot(v_M + v_{M-1}) \\ \text{for } m \in 1 .. M-1 \\ \quad v1_m \leftarrow \phi(m\cdot\Delta, v_m)\cdot\tau + v_{m-1}\cdot\dfrac{D(v_{m-1})\cdot\tau}{\Delta^2} + v_m\cdot\left(1 - \dfrac{2\cdot D(v_m)\cdot\tau}{\Delta^2}\right) + v_{m+1}\cdot\dfrac{D(v_{m+1})\cdot\tau}{\Delta^2} \\ v1 \end{array} \right.$$

$$V(t) := \left| \begin{array}{ll} u & \text{if } t = 0 \\ F(V(t-1)) & \text{otherwise} \end{array} \right.$$

Начальное распределение температуры вдоль расчетной области и решение для двух моментов времени показано на рис. 11.7 сплошной, пунктирной и штриховой линиями соответственно. Физически такое поведение вполне естественно — с течением времени тепло из более нагретой области перетекает в менее нагретую, а зона изначально высокой температуры остывает и размывается.

Примечание

Несколько забегая вперед, заметим, что показанное на рис. 11.7 решение и соответствует коэффициенту Куранта `C=0.4`. Попробуйте осуществить расчет с увеличенным временным шагом, чтобы коэффициент `C` был больше 1, и посмотрите, что из этого получится (такой расчет и его объяснение приведены далее в *разд. "Устойчивость"*).

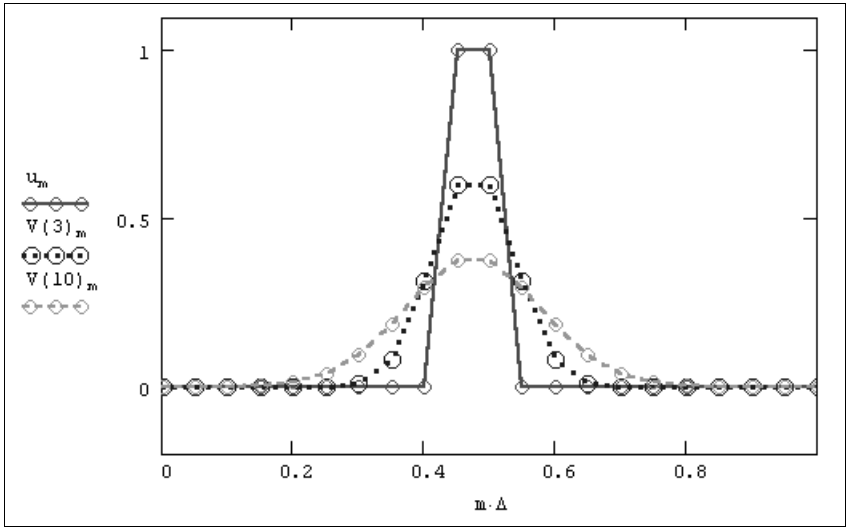


Рис. 11.7. Решение линейного уравнения теплопроводности (продолжение листинга 11.1)

Нелинейное уравнение

Намного более интересные решения можно получить для нелинейного уравнения теплопроводности, например, с нелинейным источником тепла $\phi(u)=10^3\cdot(u-u^3)$. Заметим, что в листинге 11.1 мы предусмотрительно определили коэффициент диффузии и источник тепла в виде пользовательских функций, зависящих от аргумента `u`, т. е. от температуры. Если бы мы собирались моделировать явную зависимость их от координат, то следовало бы ввести в пользовательскую функцию в качестве аргумента переменную `x`, как это сделано для источника тепла ϕ. Поэтому нет ничего проще замены определения этих функций с констант `D(u)=1` и ϕ`(x,u)=0` на новые функции, которые станут описывать другие модели диффузии тепла. Начнем с того, что поменяем четвертую строку листинга 11.1 на $\phi(x,u)=10^3\cdot(u-u^3)$, не изменяя пока постоянного значения коэффициента диффузии.

> ***Примечание***
>
> С физической точки зрения, зависимость коэффициента диффузии и функции источника тепла от температуры означает, что эти параметры будут меняться от точки к точке среды, определяясь локальными значениями текущей температуры в этих точках. Ввод ненулевого источника тепла означает, что среда получает определенное количество тепла, тем большее, чем больше локальная температура. Можно догадаться, что введение такой зависимости может моделировать, в частности, горение среды.

Если осуществить расчеты с упомянутым источником (имеющим кубическую нелинейность), то получится очень интересное решение уравнения теплопроводности, имеющее профиль тепловых фронтов. С течением времени граница раздела высокой и низкой температуры распространяется в обе стороны от зоны первичного нагрева, оставаясь весьма четко выделенной (рис. 11.8).

Еще более неожиданные решения возможны при нелинейности также и коэффициента диффузии. Например, если взять квадратичный коэффициент диффузии

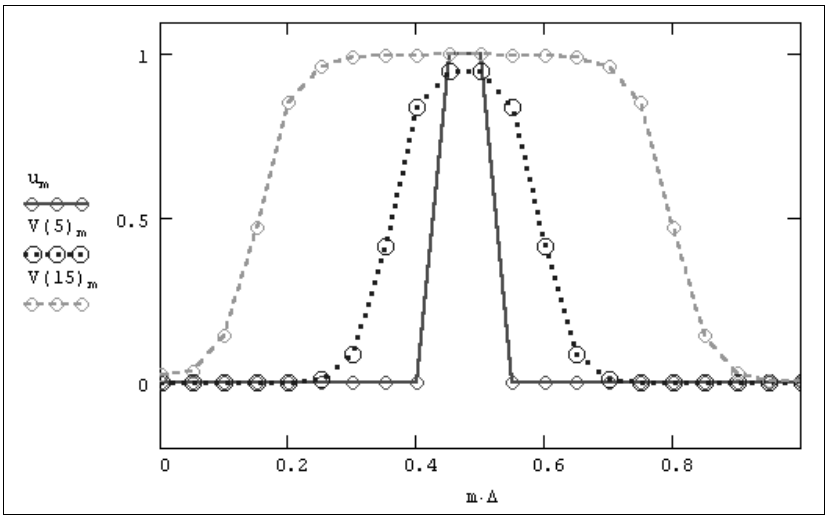


Рис. 11.8. Решение уравнения теплопроводности с нелинейным источником (тепловой фронт)

$D(x,u)=u^2$ (что с учетом его умножения на неизвестные функции создаст кубическую нелинейность уравнения), а также $\phi(x,u)=10^3 \cdot u^{3.5}$, то вы сможете наблюдать совсем иной режим горения среды. В отличие от рассмотренного эффекта распространения тепловых фронтов, горение оказывается локализованным в области первичного нагрева среды, причем температура в центре нагрева со временем возрастает до бесконечной величины (рис. 11.9). Такое решение описывает так называемый режим горения "с обострением".

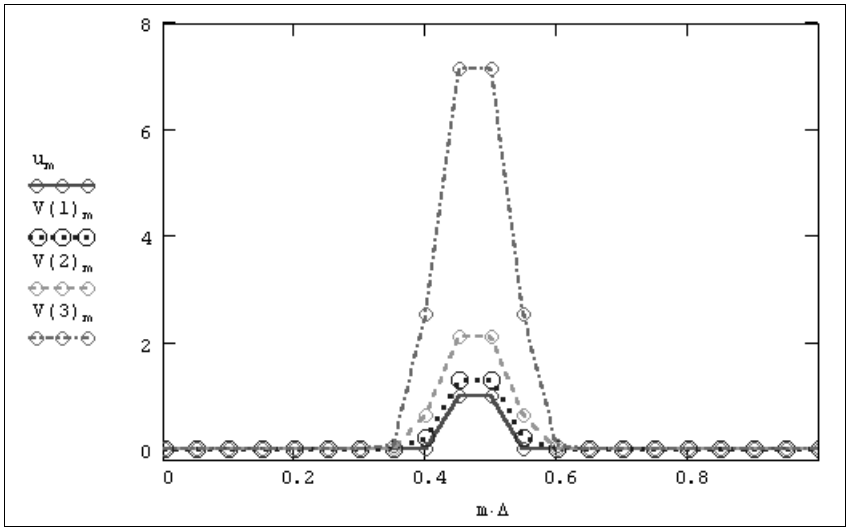


Рис. 11.9. Решение уравнения теплопроводности с нелинейным источником и коэффициентом диффузии (режим локализации горения)

Читателю предлагается поэкспериментировать с этим и другими нелинейными вариантами уравнения теплопроводности. Существенно, что такие интересные результаты удается получить лишь численно, а в Mathcad только с применением элементов программирования.

ⓘ 🛠 Устойчивость

Как мы убедились, явная разностная схема Эйлера дает вполне разумные результаты и может использоваться для практического моделирования задач, связанных с решением уравнений в частных производных. Однако теперь пришло время сказать об очень важной характеристике разностных схем, которая называется их *устойчивостью*. Не вдаваясь в детали, заметим, что производить расчеты можно только при помощи устойчивых разностных схем, а чтобы пояснить это понятие, обратимся вновь к листингу 11.1, реализующему явную схему для линейного уравнения диффузии.

Слегка изменим соотношение шагов по времени и пространственной координате, произведя расчеты сначала с $\tau=0.0005$ (эти результаты показаны на рис. 11.7 выше), а также с $\tau=0.0010$ и $\tau=0.0015$. Результат применения разностной схемы

Эйлера для τ=0.0010 примерно тот же, что и для меньшего значения τ, приведенного на рис. 11.7. А вот следующее (казалось бы, незначительное) увеличение шага по времени приводит к катастрофе (рис. 11.10).

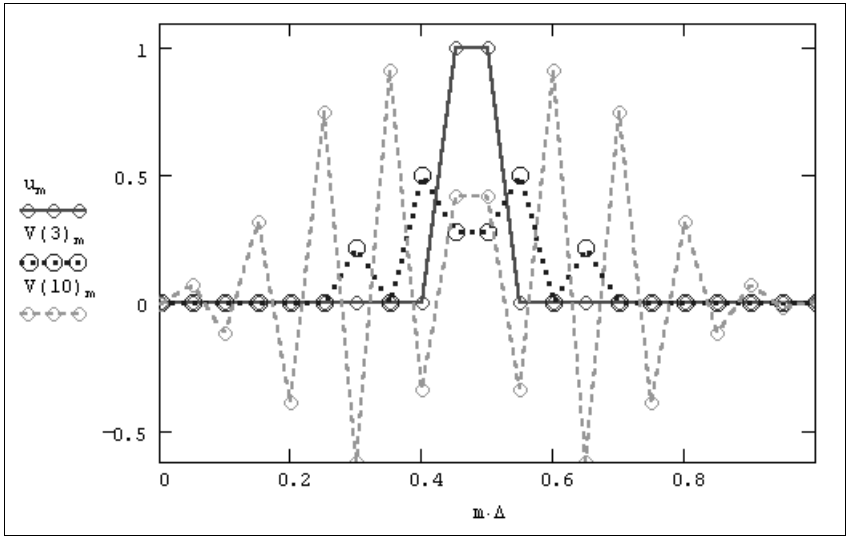


Рис. 11.10. Численное решение уравнения теплопроводности при помощи явной схемы Эйлера (см. листинг 11.1 ниже с временным шагом τ=0.0015)

Вместо ожидаемого решения получаются совершенно неожиданные профили температуры, которые быстро осциллируют вдоль пространственной координаты, причем амплитуда и число пиков этих осцилляций быстро увеличиваются от шага к шагу. Совершенно ясно, что полученное решение не имеет ничего общего с физикой моделируемого явления, а является следствием внутренних свойств самой разностной схемы, которые до этого были для нас скрыты.

Характерная "разболтка" решения как раз и является проявлением неустойчивости явной схемы Эйлера для выбранного соотношения шагов по времени и пространству. В теории численных методов показывается, что явная схема Эйлера для уравнения теплопроводности устойчива при значениях коэффициента Куранта, меньших 1, и неустойчива в противоположном случае. Иными словами, существует ограничение для выбора соотношения шагов, заключающееся в том, что для расчета на более частых пространственных сетках необходимо использовать также и малые шаги по времени.

Примечание

Как несложно убедиться, для τ=0.0005 коэффициент Куранта C=0.4, для τ=0.0010 он все еще меньше единицы: C=0.8, а для τ=0.0015 решение уже больше единицы: C=1.2, в связи с чем схема становится неустойчивой (рис. 11.10).

🛠 11.2.2. Неявная схема Эйлера

В отличие от явной схемы Эйлера, неявная является безусловно-устойчивой (т. е. не выдающей "разболтки" ни при каких значениях коэффициента Куранта). Однако ценой устойчивости является необходимость решения на каждом шаге по времени системы алгебраических уравнений.

ⓘ 🛠 Построение неявной разностной схемы

Чтобы построить неявную разностную схему для уравнения диффузии, используем шаблон, изображенный на рис. 11.11, т. е. для дискретизации пространственной производной будем брать значения сеточной функции с верхнего (неизвестного) слоя по времени. Таким образом, разностное уравнение для `(i,k)`-го узла будет отличаться от уравнения для явной схемы (11.8) только индексами по временной координате в правой части:

$$\frac{u_{i,k+1} - u_{i,k}}{\Delta_t} = D\,\frac{u_{i-1,k+1} - 2\,u_{i,k+1} + u_{i+1,k+1}}{\Delta_x^2} + \varphi_{i,k}\,. \qquad (11.10)$$

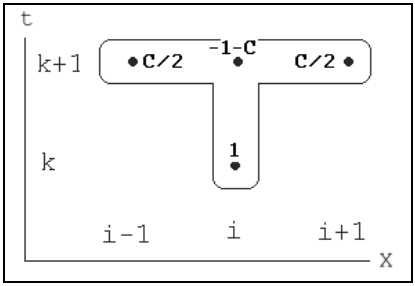


Рис. 11.11. Шаблон неявной схемы для уравнения теплопроводности

Если привести подобные слагаемые, то получится система уравнений, связывающая для каждого `i`-го узла три неизвестных значения сеточной функции (в самом этом узле и в соседних с ним слева и справа узлах). Множители при неизвестных значениях сеточной функции в узлах шаблона показаны на рис. 11.11 в виде подписей, подобно тому, как это было сделано для явной схемы (см. рис. 11.6).

Очень важно, что если само уравнение теплопроводности линейно, то `C` в левой части разностного уравнения является константой, а ϕ в его правой части может зависеть только от первой степени `u`. Поэтому система уравнений (11.10) для всех пространственных узлов `i=1...M-1` является линейной системой, что существенно упрощает ее решение (поскольку известно, что для линейных систем с ненулевым определителем решение существует и является единственным). Напомним, что для получения замкнутой системы линейных уравнений необходимо дополнить данный набор разностных уравнений граничными условиями, т. е. известными значениями сеточной функции для `i=0` и `i=M`.

Примечание

Если рассматривать нелинейный случай, то на каждом шаге по времени пришлось бы решать систему нелинейных уравнений, число решений которых могло бы быть большим, и среди них требовалось бы отыскать нужное, а не паразитное решение.

Для реализации неявной схемы, таким образом, можно использовать комбинацию средств программирования Mathcad и встроенной функции решения системы линейных уравнений `lsolve`. Один из возможных способов решения предложен в листинге 11.2. Бóльшая часть этого листинга является вводом параметров задачи (шагов, начальных и граничных условий), и только в последней его строке определяется функция пользователя, вычисляющая сеточную функцию на каждом временном слое (при помощи встроенной функции решения системы линейных уравнений `lsolve`). В нескольких предыдущих строках листинга (после расчета коэффициента Куранта `Cou`) формируется матрица системы уравнений `A`, которая записывается в подходящем для Mathcad виде, как это сделано в листинге 11.2. Как несложно убедиться, столбец правых частей разностных уравнений выражается вычисленными значениями сеточной функции с предыдущего слоя.

Результаты расчетов по неявной схеме показаны на рис. 11.12 и, как видно, они дают примерно те же результаты, что и в случае применения явной схемы (см. рис. 11.7). Обратите внимание, что решение устойчиво при любых значениях коэффициента Куранта (в том числе и бóльших `1`), поскольку, как следует из соответствующих положений теории численных методов, неявная схема является безусловно-устойчивой.

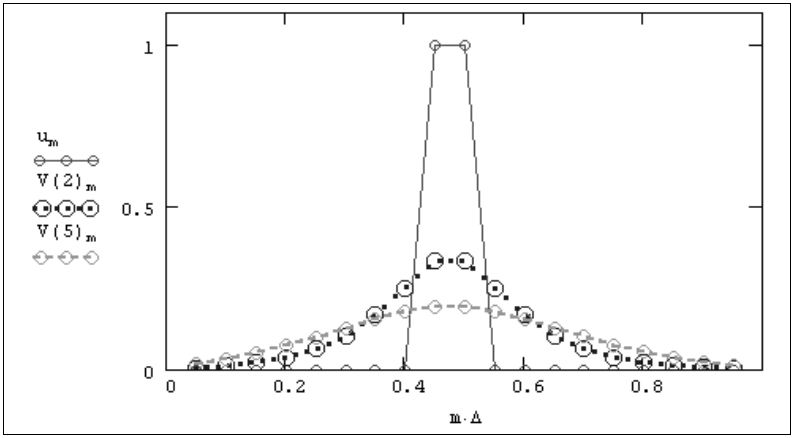


Рис. 11.12. Решение линейного уравнения теплопроводности при помощи неявной схемы на первом слое по времени (листинг 11.2)

Листинг 11.2. Неявная схема для линейного уравнения теплопроводности

```
D := 1                                  φ(x, u) := 0
Border(t) := 0
Init(x) := Φ(x − 0.45) − Φ(x − 0.55)
τ := 0.005
M := 20
Δ := 1/M                                Δ = 0.05
```

$m := 0 .. M$

$u_m := \mathrm{Init}(m \cdot \Delta)$

$Cou := \dfrac{2 \cdot D \cdot \tau}{\Delta^2}$ $\qquad Cou = 4$

$m := 1 .. M - 1$

$A_{m,m} := -Cou - 1$

$A_{m,m-1} := \dfrac{Cou}{2}$ $\qquad A_{m,m+1} := \dfrac{Cou}{2}$

$A_{0,0} := 1$ $\qquad A_{M,M} := 1$

$$V(n) := \begin{cases} u & \text{if } n = 0 \\ \mathrm{lsolve}(A, -V(n-1)) & \text{otherwise} \end{cases}$$

Алгоритм прогонки

Приведем в данном разделе описание чрезвычайно популярного алгоритма реализации неявных разностных схем, который называется методом *прогонки*. Этот алгоритм имел историческое значение для становления технологий расчетов уравнений в частных производных, и мы просто не можем не упомянуть о нем в этой книге.

> ***Примечание***
>
> Сразу оговоримся, что его применение для решения уравнений в частных производных в среде Mathcad может быть оправдано, только если вы работаете с очень частыми сетками, которые приводят к системам разностных уравнений большой размерности и, соответственно, очень большому времени вычислений.

Основным вычислительным ядром программы, реализующей на Mathcad неявную разностную схему, было решение (на каждом временном слое) системы линейных алгебраических уравнений, задаваемых матрицей `A`. Заметим, что эта матрица, как говорят, имеет *диагональное преобладание*, а точнее, является *трехдиагональной* (рис. 11.13). Все ее элементы, кроме элементов на главной диагонали и двух соседних диагоналях, равны нулю. С точки зрения оптимизации быстродействия алгоритма применение встроенной функции `lsolve` является весьма расточительным, поскольку основной объем арифметических операций, выполняемых компьютером (а он составляет, как нетрудно убедиться, величину порядка M^2), сводится к непроизводительному перемножению нулей.

Для отыскания решения линейных систем алгебраических уравнений имеется чрезвычайно эффективный алгоритм, называемый прогонкой, который позволяет уменьшить число арифметических операций на целый порядок, т. е. до значения порядка `M`. Это означает, что при использовании пространственных сеток с `1000` узлами выигрыш во времени вычислений составит величину порядка 10^3! Реализация данного алгоритма приведена в листинге 11.3, который является продолжением листинга 11.2, используя определенные в нем коэффициенты матрицы `A`, а также начальное условие.

Программа листинга 11.3 осуществляет пересчет одного шага по времени, т. е. заменяет содержимое столбца `u` с предыдущего временного слоя вычисленными значениями неизвестной функции со следующего слоя. Первые пять строк листинга 11.3 представляют так называемый *обратный* ход прогонки, а последние две строки — ее *прямой* ход. Заинтересовавшемуся читателю предлагается самому оформить представленный алгоритм прогонки в виде программы решения разностных уравнений для вычисления произвольного временного слоя по примеру листингов 11.1 и 11.2. Заметим, что описание этого знаменитого алгоритма можно отыскать практически в любом современном учебнике по численным методам.

A =

	0	1	2	3	4	5	6	7	8	9	10
0	1	0	0	0	0	0	0	0	0	0	0
1	0.5	-2	0.5	0	0	0	0	0	0	0	0
2	0	0.5	-2	0.5	0	0	0	0	0	0	0
3	0	0	0.5	-2	0.5	0	0	0	0	0	0
4	0	0	0	0.5	-2	0.5	0	0	0	0	0
5	0	0	0	0	0.5	-2	0.5	0	0	0	0
6	0	0	0	0	0	0.5	-2	0.5	0	0	0
7	0	0	0	0	0	0	0.5	-2	0.5	0	0
8	0	0	0	0	0	0	0	0.5	-2	0.5	0
9	0	0	0	0	0	0	0	0	0.5	-2	0.5
10	0	0	0	0	0	0	0	0	0	0	1

Рис. 11.13. Матрица системы линейных разностных уравнений для неявной схемы (листинг 11.2 для `M=10`)

Листинг 11.3. Алгоритм прогонки (продолжение листинга 11.2)

$\alpha_{M-1} := 0$ $\qquad$ $\beta_{M-1} := 0$

$m := M - 1 .. 1$

$$\gamma_m := -\frac{1}{A_{m,m} + A_{m,m+1} \cdot \alpha_m}$$

$\alpha_{m-1} := \gamma_m \cdot A_{m,m-1}$

$\beta_{m-1} := \gamma_m \cdot \left(A_{m,m+1} \cdot \beta_m - b_m \right)$

$m := 0 .. M - 1$

$u_{m+1} := \alpha_m \cdot u_m + \beta_m$

🛠 11.2.3. О возможности решения многомерных уравнений

Все, что было сказано до сих пор, касалось исключительно способов решения одномерных (в смысле пространственных координат) уравнений. И алгоритмы разно-

стных схем, и встроенные функции относились к уравнениям, зависящим от одной пространственной координаты.

Можно ли при помощи Mathcad решать двумерные или трехмерные (пространственные) уравнения? С точки зрения программирования пользователем численных алгоритмов типа метода сеток принципиальных ограничений нет. Разумеется, если сначала аккуратно выписать разностную схему соответствующего многомерного дифференциального уравнения, то вполне возможно запрограммировать ее при помощи описанных нами средств. Самым главным противодействием будет существенное увеличение времени расчетов. Простая оценка необходимого количества операций показывает, что ввод зависимости уравнения от второй пространственной координаты многократно увеличивает число разностных уравнений, которые должны решаться при реализации каждого шага по времени. К примеру, если используется пространственная сетка из 100 узлов по каждой координате, то вместо 10^2 разностных уравнений на каждом шаге придется решать уже 10^4 уравнений, т. е. объем вычислений сразу же возрастает в 100 раз. Вообще говоря, пакет Mathcad не является экономичной средой вычислений, и бороться с их сильно возрастающим объемом следует пользователю еще на этапе разработки алгоритма. Хорошим примером такой борьбы может служить применение специфических алгоритмов типа метода прогонки (*см. разд. "Алгоритм прогонки" этой главы*).

Приведем некоторые дополнительные замечания, связанные с возможностью осуществить редукцию громоздких (в смысле организации вычислений) двумерных задач к более простым. Рассмотрим ради примера двумерное уравнение теплопроводности (11.1) без источника с нулевыми граничными условиями и некоторым начальным двумерным распределением температуры по расчетной поверхности:

$$\frac{\partial u(x, y, t)}{\partial t} = D\left(\frac{\partial^2 u(x, y, t)}{\partial x^2} + \frac{\partial^2 u(x, y, t)}{\partial y^2}\right).$$

Произведем дискретизацию данного уравнения по временной координате, заменяя первую производную ее разностным аналогом и несколько перегруппировывая слагаемые и множители:

$$\frac{\partial^2 u_{i+1}}{\partial x^2} + \frac{\partial^2 u_{i+1}}{\partial y^2} = \frac{u_{i+1} - u_i}{D\tau}.$$

Как вы видите, мы используем неявную разностную схему, заранее заботясь о том, чтобы разностная задача была более устойчивой. Здесь $u_i(x,y)$ — известная с предыдущего шага по времени функция двух пространственных переменных, а $u_{i+1}(x,y)$ — функция, подлежащая определению при реализации каждого шага по времени.

Посмотрим на полученную задачу с несколько другой стороны — а именно как на дифференциальное уравнение относительно неизвестной функции двух переменных $u_{i+1}(x,y)$. Подчеркнем, что такое уравнение получается для каждого шага по времени, т. е. для реализации всей разностной схемы требуется решить большое число таких уравнений.

С предложенной точки зрения, на каждом временном шаге необходимо решить некоторое двумерное эллиптическое линейное уравнение, причем его граничные условия определяются граничными условиями исходной задачи. Это уравнение очень похоже на уравнение Пуассона с той лишь разницей, что в его правую часть, описывающую источник, входит неизвестная функция (к счастью, линейно). Таким образом, зависимость от найденного на предыдущем шаге по времени решения определяется зависимостью от него источника, т. е. правой части.

Суммируя сказанное, можно констатировать, что если вы имеете запрограммированный алгоритм решения выписанного эллиптического уравнения, чуть более сложного, чем уравнение Пуассона, то его с легкостью можно использовать в качестве подпрограммы реализации разностной схемы двумерного уравнения теплопроводности. Забегая вперед, приходится отметить, что, к сожалению, встроенные функции Mathcad для решения уравнения Пуассона в данном случае не годятся в качестве такой подпрограммы, поскольку предполагают независимость источника от самой неизвестной функции и могут справляться лишь с правой частью, зависящей только от пространственных координат (*см. разд. 11.3.2*).

Не будем далее останавливаться на способах решения многомерных уравнений, ограничившись этими замечаниями относительно путей оптимизации алгоритмов их решения.

11.3. Встроенные функции для решения уравнений в частных производных

Как видно из предыдущего раздела, с уравнениями в частных производных вполне можно справиться и не прибегая к специфическим средствам Mathcad. Между тем в версиях Mathcad 11 (и выше) имеется несколько встроенных функций, при помощи которых можно автоматизировать процесс решения дифференциальных уравнений в частных производных. Рассмотрим в данном разделе основные аспекты их применения, отмечая не только инструкции по их использованию, описанные разработчиками Mathcad, но также и некоторые "скрытые" возможности этих функций.

⑪ 11.3.1. Параболические и гиперболические уравнения

Разработчики впервые применили дополнительные встроенные функции для решения параболических и гиперболических уравнений в частных производных в версии Mathcad 11, отлично осознавая значимость этих задач для современного исследователя и инженера. Предусмотрены два варианта решения: при помощи вычислительного блока `Given/pdesolve`, а также при помощи встроенной функции `numol`. Первый путь проще в применении и нагляднее, зато второй позволяет автоматизировать процесс решения уравнений в частных производных, например, если нужно включить его в качестве составного шага в более сложную Mathcad-программу.

Вычислительный блок *Given / pdesolve*

Встроенная функция `pdesolve` применяется в рамках вычислительного блока, начинающегося ключевым словом `Given`, и пригодна для решения различных гиперболических и параболических уравнений. Она предназначена для решения одномерного уравнения (или системы уравнений) в частных производных (того, которое определит пользователь в рамках вычислительного блока `Given`), зависящего от времени `t` и пространственной координаты `x`, имеет целый набор различных аргументов и работает следующим образом:

- `pdesolve(u,x,xrange,t,trange,[xpts],[tpts])` — возвращает скалярную (для единственного исходного уравнения) или векторную (для системы уравнений) функцию двух аргументов `(x,t)`, являющуюся решением дифференциального уравнения (или системы уравнений) в частных производных. Результирующая функция получается интерполяцией сеточной функции, вычисляемой согласно разностной схеме:
  - `u` — явно заданный вектор имен функций (без указания имен аргументов), подлежащих вычислению. Эти функции, а также граничные условия (в форме Дирихле или Неймана) должны быть определены пользователем перед применением функции `pdesolve` в вычислительном блоке после ключевого слова `Given`. Если решается не система уравнений в частных производных, а единственное уравнение, то, соответственно, вектор `u` должен содержать только одно имя функции и вырождается в скаляр;
  - `x` — пространственная координата (имя аргумента неизвестной функции);
  - `xrange` — пространственный интервал, т. е. вектор значений аргумента `x` для граничных условий. Этот вектор должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала);
  - `t` — время (имя аргумента неизвестной функции);
  - `trange` — расчетная временная область: вектор значений аргумента `t`, который должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала по времени);
  - `xpts` — количество пространственных точек дискретизации (может не указываться явно, в таком случае будет подобрано программой автоматически);
  - `tpts` — количество временных слоев, т. е. интервалов дискретизации по времени (также может не указываться пользователем явно).

В качестве примера использования функции `pdesolve` (листинг 11.4) используем то же самое одномерное уравнение теплопроводности (11.5) с граничными и начальными условиями (11.6) и (11.7).

Листинг 11.4. Решение одномерного уравнения теплопроводности

```
D := 0.1
L := 1                    T := 10
```

Given

$u_t(x,t) = D \cdot u_{xx}(x,t)$

$u(x,0) = \Phi(x-0.45) - \Phi(x-0.55)$

$u(0,t) = 0 \qquad u(L,t) = 0$

$u := \mathrm{Pdesolve}\left[u, x, \begin{pmatrix} 0 \\ L \end{pmatrix}, t, \begin{pmatrix} 0 \\ T \end{pmatrix}, 100, 10\right]$

Для корректного использования функции `pdesolve` предварительно, после ключевого слова `Given`, следует записать само уравнение и граничные условия при помощи логических операторов (для их ввода в Mathcad существует специальная панель). Обратите внимание, что уравнение должно содержать имя неизвестной функции `u(x,t)` вместе с именами аргументов (а не так, как она записывается в пределах встроенной функции `pdesolve`). Для идентификации частных производных в пределах вычислительного блока следует использовать нижние индексы, например u_{xx}`(,t)`, для обозначения второй производной функции `u` по пространственной координате `x`.

Как видно из рис. 11.14, на котором изображены результаты расчетов по листингу 11.4, встроенная функция с успехом справляется с уравнением диффузии, отыскивая уже хорошо знакомое нам решение. Заметим, что использование встроенной функции `pdesolve` связано с довольно громоздкими вычислениями, которые могут отнимать существенное время.

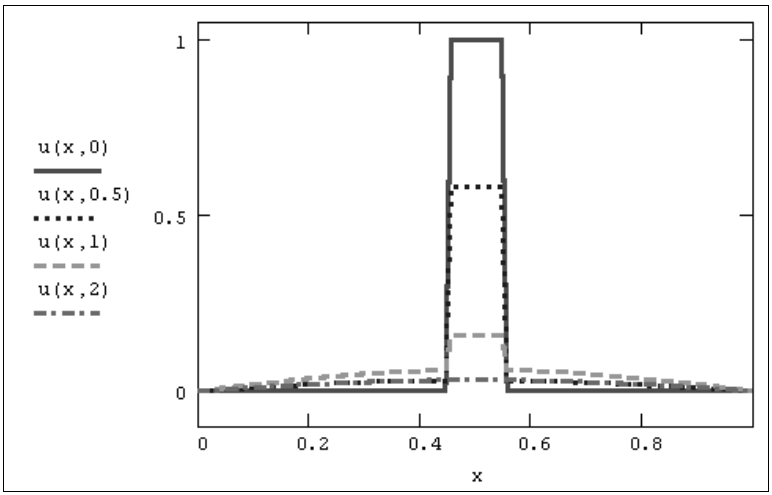


Рис. 11.14. Решение уравнения диффузии тепла при помощи встроенной функции `pdesolve` (листинг 11.4)

Примечание

Как вы можете заметить, выбирать величину шага по пространственной и временной переменным может как сам алгоритм, так и пользователь (неявным образом, через число узлов сетки). Читателю предлагается повторить вычисления листинга 11.4 для различных комби-

наций параметров (главным образом, числа узлов сетки), чтобы проверить, в каких случаях алгоритм встроенной функции справляется с задачей, выдавая верное решение, а в каких дает сбой.

Пример: волновое уравнение

Приведем еще один пример использования функции `pdesolve` для решения уравнений в частных производных.

Рассмотрим одномерное волновое уравнение, которое описывает, например, свободные колебания струны музыкального инструмента:

$$\frac{\partial^2 u(x, t)}{\partial t^2} = c^2 \frac{\partial^2 u(x, t)}{\partial x^2}. \qquad (11.11)$$

Здесь неизвестная функция `u(x,t)` описывает динамику смещения профиля струны относительно невозмущенного (прямолинейного) положения, а параметр `c` характеризует материал, из которого изготовлена струна.

Как вы видите, уравнение (11.11) содержит производные второго порядка, как по пространственной координате, так и по времени. Для того чтобы можно было использовать встроенную функцию `pdesolve`, необходимо переписать волновое уравнение в виде системы двух уравнений в частных производных, введя вторую неизвестную функцию $v=u_t$. Программа для решения волнового уравнения приведена в листинге 11.5, а результат — на рис. 11.15.

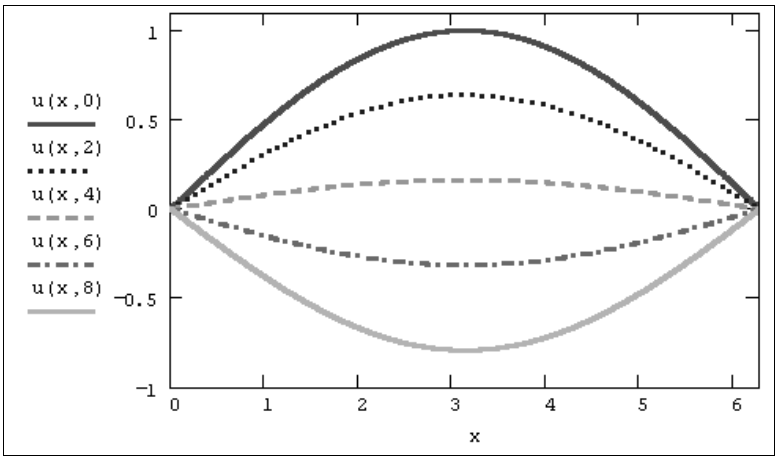


Рис. 11.15. Решение волнового уравнения (продолжение листинга 11.5)

Листинг 11.5. Решение волнового уравнения

$L := 2 \cdot \pi$ $\qquad$ $T := 1$

Given

$v_t(x, t) = c^2 \cdot u_{xx}(x, t)$

$u_t(x,t) = v(x,t)$

$u(x,0) = \sin\left(\frac{\pi \cdot x}{L}\right)$ $\qquad$ $v(x,0) = 0$

$u(0,t) = 0$ $\qquad$ $u(L,t) = 0$

$\begin{pmatrix} u \\ v \end{pmatrix} := \mathrm{Pdesolve}\left[\begin{pmatrix} u \\ v \end{pmatrix}, x, \begin{pmatrix} 0 \\ L \end{pmatrix}, t, \begin{pmatrix} 0 \\ T \end{pmatrix}\right]$

Встроенная функция *numol*

Альтернативный вариант решения дифференциальных уравнений в частных производных заключается в применении еще одной встроенной функции `numol`, которая реализует тот же самый алгоритм сеток, позволяя вручную задать большинство его параметров:

- `numol(xrange,xpts,trange,tpts,Npde,Nae,rhs,init,bc)` — возвращает матрицу решения дифференциального уравнения в частных производных, представляющую искомую сеточную функцию в каждой точке по пространственной (по строкам) и временной координате (по столбцам). Если решается не одно уравнение, а система уравнений, то результатом является составная матрица, образованная путем слияния (слева направо) матриц со значениями каждой искомой сеточной функции:
  - `Npde` — общее количество дифференциальных уравнений в частных производных в системе;
  - `Nae` — общее количество дополнительных алгебраических уравнений, которые также могут входить в систему;
  - `rhs` — векторная функция, определяющая систему дифференциальных и алгебраических уравнений (формат этого и двух следующих матричных параметров объяснен в листинге 11.9);
  - `init` — векторная функция, определяющая начальные условия для каждой неизвестной функции;
  - `bc` — функциональная матрица граничных условий;
  - `xrange` — пространственный интервал, т. е. вектор значений аргумента `x` для граничных условий. Этот вектор должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала);
  - `xpts` — количество пространственных точек дискретизации (может не указываться явно, в таком случае будет подобрано программой автоматически);
  - `trange` — расчетная временная область: вектор значений аргумента `t`, который должен состоять из двух действительных чисел (представляющих левую и правую границу расчетного интервала по времени);
  - `tpts` — количество временных слоев, т. е. интервалов дискретизации по времени (также может не указываться пользователем явно);

Пример решения волнового уравнения при помощи функции `numol` приведен в листинге 11.6, особое внимание в котором мы призываем уделить формату представления векторов `rhs`, `init` и `bc`, а также принципу извлечения отдельных сеточных решений из матрицы-результата. График решения, показанный на рис. 11.16, полезно сравнить с результатом применения вычислительного блока из предыдущего раздела (см. листинг 11.5 и рис. 11.15).

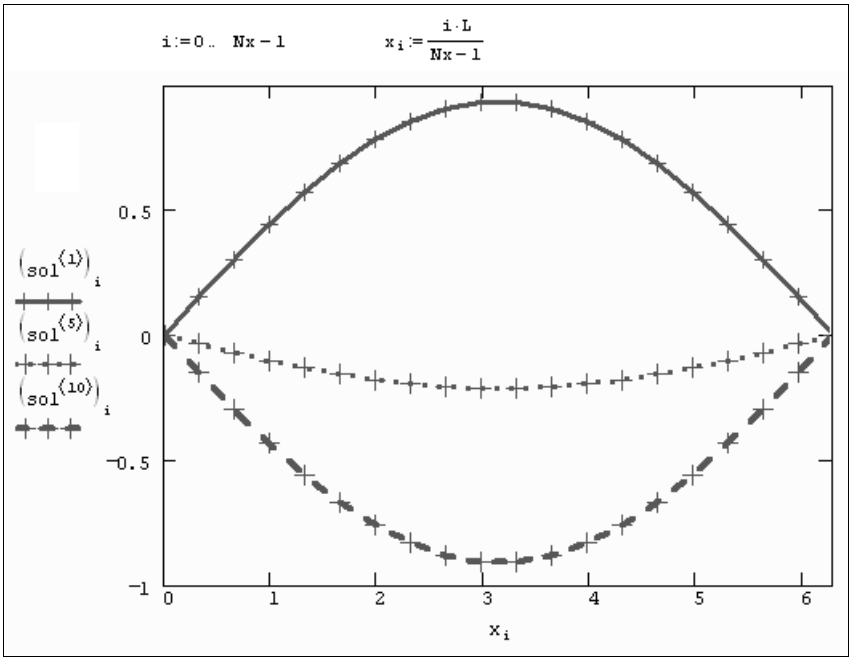


Рис. 11.16. Решение волнового уравнения (продолжение листинга 11.6)

Листинг 11.6. Решение волнового уравнения при помощи функции `numol`

$c := 1$

$L := 2\cdot\pi \qquad Nx := 20$

$T := 10 \qquad Nt := 15$

$num_pde := 2 \qquad num_pae := 0$

$$rhs(x, t, u, u_x, u_{xx}) := \begin{pmatrix} u_1 \\ c^2\cdot u_{xx_0} \end{pmatrix}$$

$$init(x) := \begin{pmatrix} \sin\left(\frac{\pi\cdot x}{L}\right) \\ 0 \end{pmatrix}$$

$$bc_func(t) := \begin{pmatrix} init(0)_0 & init(L)_0 & \text{"D"} \\ \text{"NA"} & \text{"NA"} & \text{"D"} \end{pmatrix}$$

$$sol := numol\left[\begin{pmatrix}0\\L\end{pmatrix}, Nx, \begin{pmatrix}0\\T\end{pmatrix}, Nt, num_pde, num_pae, rhs, init, bc_func\right]$$

$rows(sol) = 20$ $\qquad$ $Nx = 20$

$cols(sol) = 30$ $\qquad$ $Nt \cdot 2 = 30$

$SOL := submatrix(sol, 0, Nt - 1, 0, Nx - 1)$

Как вы видите, функция `numol` имеет еще большее число аргументов, нежели `pdesolve`, и позволяет автоматизировать применение метода сеток. Однако пользоваться ею намного сложнее, чем вычислительным блоком, поскольку и уравнения, и начальные и граничные условия должны быть записаны в специальном формате. Применение функции `numol` оправданно, когда необходимо включить решение уравнений в частных производных в более сложные вычисления в качестве подпрограммы, организовать серию расчетов с меняющимся параметром, подготовить анимацию графиков решения и т. п.

Именно в целях визуализации решения параболических и гиперболических уравнений в частных производных использование функции `numol` наиболее полезно. График решения динамических уравнений (зависящих от времени `t`) выглядит намного эффектнее и воспринимается несравненно лучше, если он оформлен в виде анимации. Для создания анимационных роликов расчетное время следует выразить через константу `FRAME` и затем применить команду **Вид / Анимация** (View / Animate).

11.3.2. Эллиптические уравнения

Решение эллиптических уравнений в частных производных реализовано только для единственного типа задач — двумерного уравнения Пуассона.

Это уравнение содержит вторые производные функции `u(x,y)` по двум пространственным переменным:

$$\frac{\partial^2 u(x, y)}{\partial x^2} + \frac{\partial^2 u(x, y)}{\partial y^2} = -f(x, y). \qquad (11.12)$$

Уравнение Пуассона описывает, например, распределение электростатического поля `u(x,y)` в двумерной области с плотностью заряда `f(x,y)`, или (*см. разд. 11.1.2*) стационарное распределение температуры `u(x,y)` на плоскости, в которой имеются источники (или поглотители) тепла с интенсивностью `f(x,y)`.

Примечание

Несмотря на то, что применение встроенных функций, описанных в данном разделе, анонсировано разработчиками Mathcad только для уравнения Пуассона, их можно применять и для решения других уравнений, даже необязательно эллиптического типа. О том, как осуществить такие расчеты, написано в конце данного раздела.

Уравнение Пуассона с нулевыми граничными условиями

Корректная постановка краевой задачи для уравнения Пуассона требует задания граничных условий. В Mathcad решение ищется на плоской квадратной области,

состоящей из (M+1)×(M+1) точек. Поэтому граничные условия должны быть определены пользователем для всех четырех сторон упомянутого квадрата. Самый простой вариант — это нулевые граничные условия, т. е. постоянная температура по всему периметру расчетной области. В таком случае можно использовать встроенную функцию multigrid:

- multigrid(F,ncycle) — матрица решения уравнения Пуассона размера (M+1)×(M+1) на квадратной области с нулевыми граничными условиями:
  - F — матрица размера (M+1)×(M+1), задающая правую часть уравнения Пуассона;
  - ncycle — параметр численного алгоритма (количество циклов в пределах каждой итерации).

Внимание!

Сторона квадрата расчетной области должна включать точно $M=2^n$ шагов, т. е. 2^n+1 узлов, где n — целое число.

Параметр численного метода ncycle в большинстве случаев достаточно взять равным 2. Листинг 11.7 содержит пример использования функции multigrid для расчета краевой задачи на области 33×33 точки и точечным источником тепла в месте, задаваемом координатами (15,20) внутри этой области.

Листинг 11.7. Решение уравнения Пуассона с нулевыми граничными условиями

$M := 32$

$F_{M,M} := 0$

$F_{15,20} := 10^4$

$G := \text{multigrid}(-F, 2)$

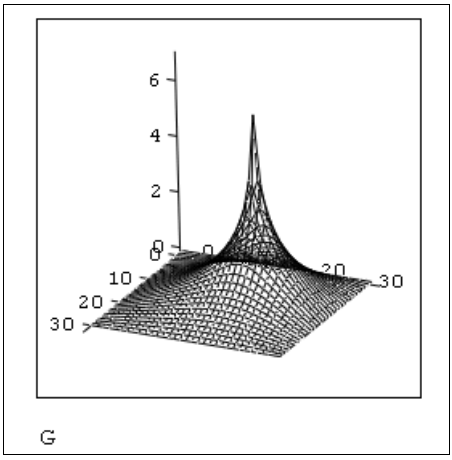


Рис. 11.17. График поверхности решения уравнения Пуассона (продолжение листинга 11.7)

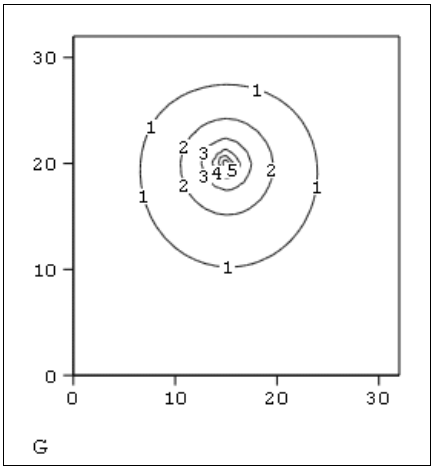


Рис. 11.18. График линий уровня решения уравнения Пуассона (продолжение листинга 11.7)

В первой строке листинга задается значение `M=32`, в двух следующих строках создается матрица правой части уравнения Пуассона, состоящая из всех нулевых элементов, за исключением одного, задающего расположение источника. В последней строке матрице `G` присваивается результат действия функции `multigrid`. Обратите внимание, первый ее аргумент сопровождается знаком "минус", что соответствует записи правой части уравнения Пуассона (11.11). Графики решения показаны на рис. 11.17 и 11.18 в виде трехмерной поверхности и линий уровня соответственно.

Уравнение Пуассона с произвольными граничными условиями

В более сложных случаях, например, для решения краевой задачи с ненулевыми условиями на границах, следует использовать другую встроенную функцию `relax`, имеющуюся в Mathcad:

- `relax(a,b,c,d,e,F,v,rjac)` — матрица решения дифференциального уравнения в частных производных на квадратной области, полученного с помощью алгоритма релаксации для метода сеток:
  - `a,b,c,d,e` — квадратные матрицы коэффициентов разностной схемы, аппроксимирующей дифференциальное уравнение;
  - `F` — квадратная матрица, задающая правую часть дифференциального уравнения;
  - `v` — квадратная матрица граничных условий и начального приближения к решению;
  - `rjac` — параметр численного алгоритма (спектральный радиус итераций Якоби).

Параметр численного алгоритма характеризует скорость сходимости итераций. Он должен быть числом от `0` до `1`. В матрице граничных условий `v` необходимо задать только граничные элементы, исходя из значения краевых условий по периметру расчетной области. Прочие (внутренние) элементы этой матрицы служат для задания начального приближения к решению. Суть алгоритма релаксации сводится к тому, что в ходе итераций происходит проверка уравнений и соответствующая коррекция значений искомой функции в каждой точке. Если начальное приближение выбрано удачно, то можно надеяться, что алгоритм сойдется ("релаксирует") к правильному решению.

> ***Внимание!***
>
> Все матрицы, задающие как коэффициенты разностной схемы `a`, `b`, `c`, `d`, `e`, граничные условия `v`, так и само решение `F`, должны иметь одинаковый размер `(M+1)×(M+1)`, соответствующий размеру расчетной области. При этом целое число `M` обязательно должно быть степенью двойки: $M=2^n$.

Решение уравнения Пуассона с тремя источниками разной интенсивности при помощи функции `relax` приведено в листинге 11.8.

Листинг 11.8. Решение уравнения Пуассона с помощью функции `relax`

```
M := 32
F_{M,M} := 0
F_{15,20} := 10     F_{25,10} := 5     F_{10,10} := −5
i := 0..M     k := 0..M
a_{i,k} := 1
b := a
c := a
d := a
e := −4·a
v_{i,k} := 0
G := relax(a,b,c,d,e,−F,v,.95)
```

Первые три строки имеют тот же смысл, что и в предыдущем листинге. Только вместо одного источника тепла взято их другое распределение — один сильный источник, один более слабый и один сток тепла. В следующих шести строках задаются коэффициенты разностной схемы. Отложим их обсуждение до последнего раздела этой главы, ограничившись утверждением, что для решения уравнения Пуассона коэффициенты должны быть взяты именно такими, как показано в листинге 11.8. В предпоследней строке задана матрица нулевых граничных условий и нулевых начальных приближений, а в последней матрице `G` присваивается результат действия функции `relax`. График полученного решения в виде линий уровня показан на рис. 11.19.

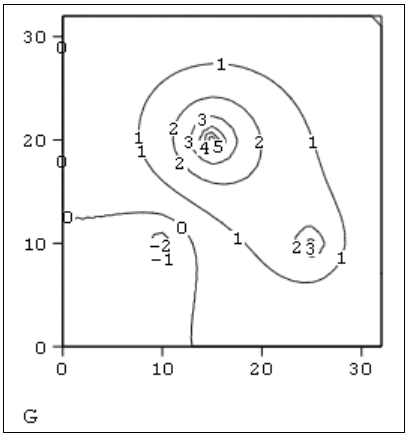


Рис. 11.19. Решение уравнения Пуассона с помощью функции `relax` (продолжение листинга 11.8)

Разностная схема для решения уравнения Пуассона

Несмотря на отсутствие сведений в справочной системе Mathcad о решении других линейных дифференциальных уравнений в частных производных, кроме уравнения

Пуассона, сделать это возможно с помощью той же функции `relax` (*см. предыдущий раздел*). Для этого нужно правильным образом задать коэффициенты разностной схемы.

Начнем с пояснения выбора этих коэффициентов (см. листинг 11.8) для уравнения Пуассона. Согласно основным идеям метода сеток (*см. разд. 11.2*), для дискретизации обеих пространственных производных в уравнении (11.12) следует использовать по три соседних узла вдоль каждой из координат. Поэтому уравнение Пуассона (11.12) может быть записано в разностной форме при помощи шаблона типа "крест" (рис. 11.20). В этом случае, после приведения подобных слагаемых в разностных уравнениях коэффициенты разностной схемы будут такими, как показано возле узлов шаблона на этом рисунке (аналогичные коэффициенты для явной и неявных схем решения уравнения теплопроводности см. на рис. 11.6 и 11.11 соответственно).

Теперь, если вы сравните полученные числа с константами, которые присвоены элементам матриц-аргументов функции `relax` (см. листинг 11.8), то увидите, что они как раз и описывают вычисленные нами только что коэффициенты разностной схемы "крест". Таким образом, нетрудно сообразить, что с помощью встроенной функции `relax` можно решать и другие линейные дифференциальные уравнения в частных производных, которые можно аппроксимировать схемой типа "крест" или схемой, являющейся ее составной частью. Конечно, для того чтобы использовать эту встроенную функцию для другого уравнения, необходимо будет составить соответствующую разностную схему.

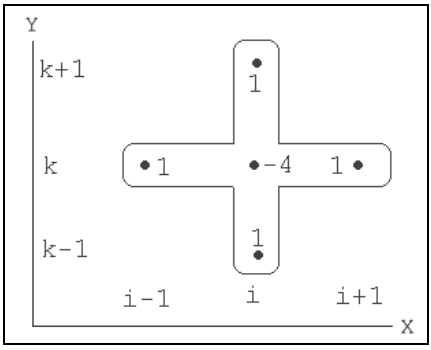


Рис. 11.20. Шаблон аппроксимации уравнения Пуассона "крест"

Решение уравнения диффузии тепла при помощи функции *relax*

Приведем пример использования встроенной функции `relax` для решения другого уравнения в частных производных (т. е. не уравнения Пуассона, для которого она изначально предназначена). Вычислим при помощи этой функции решение уже хорошо нам знакомого однородного линейного уравнения теплопроводности (*см. разд. 11.1.2*). Будем использовать явную разностную схему, шаблон которой

изображен на рис. 11.6. Для того чтобы "приспособить" для явной схемы функцию `relax`, требуется только задать ее аргументы в соответствии с коэффициентами, показанными на шаблоне (см. рис. 11.6). Программа, реализующая таким способом явную схему, представлена в листинге 11.9. Число Куранта в этом листинге обозначено переменной `C`, как и положено явной разностной схеме, она выдает устойчивое решение только для `C<1`.

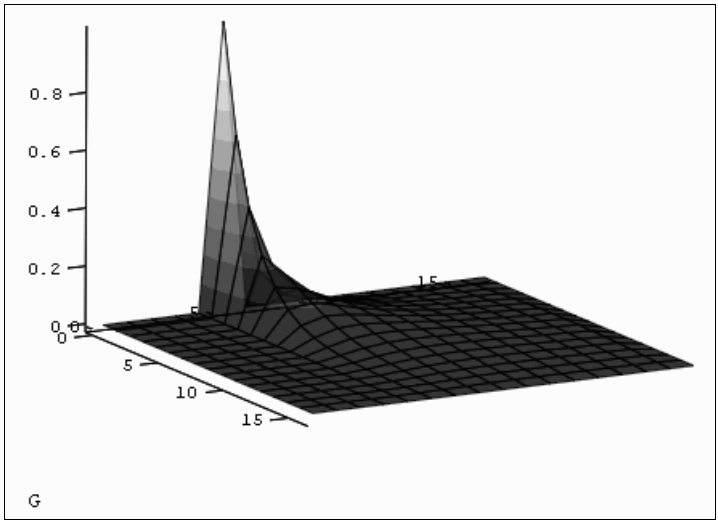


Рис. 11.21. Решение уравнения теплопроводности с помощью функции `relax` (продолжение листинга 11.9)

Листинг 11.9. Решение уравнения теплопроводности при помощи функции `relax`

```
M := 16
C := 0.9
i := 0 .. M                    k := 0 .. M
a_{i,k} := C/2
b_{i,k} := C/2
c_{i,k} := 0
d_{i,k} := 1 - C
e_{i,k} := -1
F_{i,k} := 0
v_{i,k} := 0                   v_{0,5} := 1
G := relax(a, b, c, d, e, F, v, .5)
```

Результат действия программы листинга 11.9 показан на рис. 11.21 в виде трехмерной поверхности. Если сравнить рис. 11.21 с рис. 11.4, полученным при расчетах по запрограммированной разностной схеме, то в графиках рис. 11.4 нетрудно узнать сечения этой поверхности плоскостями `t=const`. Еще раз подчеркнем, что использовать встроенную функцию можно только для тех уравнений, которые допускают построение разностной схемы типа "крест" (см. рис. 11.17) или составного фрагмента этой схемы.

ГЛАВА 12

Статистика

Mathcad имеет развитый аппарат работы с задачами математической статистики и обработки эксперимента.

Во-первых, имеется большое количество встроенных специальных функций, позволяющих рассчитывать плотности вероятности и другие основные характеристики основных законов распределения случайных величин (*см. разд. 12.1*). Наряду с этим, в Mathcad запрограммировано соответствующее количество генераторов псевдослучайных чисел для каждого закона распределения (*см. разд. 12.1*), что позволяет эффективно проводить моделирование методами Монте-Карло.

Во-вторых, предусмотрена возможность построения гистограмм и расчета статистических характеристик выборок случайных чисел и случайных процессов, таких как средние, дисперсии, корреляции и т. п. (*см. разд. 12.2*). При этом случайные последовательности могут как создаваться генераторами случайных чисел (методы Монте-Карло, *см. разд. 12.3*), так и вводиться пользователем из файлов.

В-третьих, имеется целый арсенал средств, направленных на интерполяцию-экстраполяцию данных, построение регрессии по методу наименьших квадратов, фильтрацию сигналов. Наконец, реализован ряд численных алгоритмов, осуществляющих расчет различных интегральных преобразований, что позволяет организовать спектральный анализ различного типа.

12.1. Статистические распределения

Фундаментальным понятием математической статистики является понятие *случайного числа* (*случайной величины*). Согласно определению, случайная величина принимает то или иное значение, но какое конкретно — зависит от принципиально непредсказуемых обстоятельств опыта и заранее точно предсказано быть не может. Можно лишь говорить о *вероятности* $P(x_K)$ принятия *случайной дискретной* величиной того или иного значения x_K или о вероятности попадания *непрерывной* случайной величины в тот или иной числовой интервал $(x, x+\Delta x)$. Вероятность $P(x_K)$ или $P(x)\cdot(\Delta x)$ соответственно может принимать значения от 0 (такое значение слу-

чайной величины совершенно невероятно) до 1 (случайная величина заведомо примет значение от x до x+Δx). Соотношение $P(x_K)$ называют *законом распределения* случайной величины, а зависимость P(x) между возможными значениями непрерывной случайной величины и вероятностями попадания в их окрестность называется ее *плотностью вероятности* (probability density).

12.1.1. Статистические функции

В Mathcad имеется ряд встроенных функций, задающих используемые в математической статистике законы распределения. Они вычисляют как значение плотности вероятности различных распределений по значению случайной величины x, так и некоторые сопутствующие функции. Все они, по сути, являются либо встроенными аналитическими зависимостями, либо специальными функциями. Большой интерес представляет наличие генераторов случайных чисел, создающих *выборку* псевдослучайных данных с соответствующим законом распределения, что является основой методов Монте-Карло (*см. разд. 12.3*).

В Mathcad заложена информация о большом количестве разнообразных статистических распределений, включающая, с одной стороны, табулированные функции вероятности, а с другой — возможность генерации последовательности случайных чисел с соответствующим законом распределения. Для реализации этих возможностей имеются четыре основные категории встроенных функций. Их названия являются составными и устроены одинаковым образом: первая литера идентифицирует определенный закон распределения, а оставшаяся часть (ниже в списке функций она условно обозначена звездочкой) задает смысловую часть встроенной функции:

- d*(x,par) — плотность вероятности;
- p*(x,par) — функция распределения;
- q*(P,par) — квантиль распределения;
- r*(M,par) — вектор M независимых случайных чисел, каждое из которых имеет соответствующее распределение:
  - x — значение случайной величины (аргумент функции);
  - P — значение вероятности;
  - par — список параметров распределения.

Чтобы получить функции, относящиеся, например, к равномерному распределению, вместо * надо поставить unif и ввести соответствующий список параметров par. Он будет состоять в данном случае из двух чисел: a,b — границ интервала распределения случайной величины.

Перечислим все типы распределения, реализованные в Mathcad, вместе с их параметрами, на этот раз обозначив звездочкой * недостающую первую букву встроенных функций. Некоторые из плотностей вероятности показаны на рис. 12.1.

- *beta(x,s1,s2) — бета-распределение (s1,s2>0 — параметры, 0<x<1).
- *binom(k,n,p) — биномиальное распределение (n — целый параметр, 0≤k≤n и 0≤p≤1 — параметр, равный вероятности успеха единичного испытания).

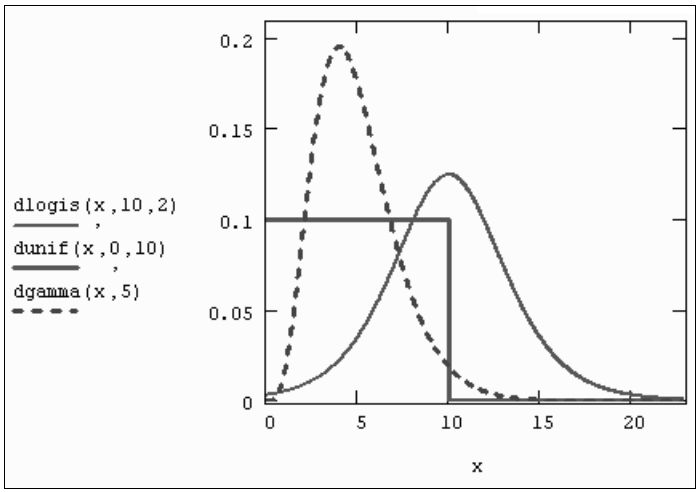


Рис. 12.1. Плотность вероятности некоторых распределений

- `*cauchy(x,l,s)` — распределение Коши (`l` — параметр разложения, `s>0` — параметр масштаба).
- `*chisq(x,d)` — χ^2 ("хи-квадрат") распределение (`d>0` — число степеней свободы).
- `*exp(x,r)` — экспоненциальное распределение (`r>0` — показатель экспоненты).
- `*F(x,d1,d2)` — распределение Фишера (`d1,d2>0` — числа степеней свободы).
- `*gamma(x,s)` — гамма-распределение (`s>0` — параметр формы).
- `*geom(k,p)` — геометрическое распределение ($0 \le p \le 1$ — параметр, равный вероятности успеха единичного испытания).
- `*hypergeom(k,a,b,n)` — гипергеометрическое распределение (`a,b,n` — целые параметры).
- `*lnorm(x,μ,σ)` — логарифмически нормальное распределение (μ — натуральный логарифм математического ожидания, $\sigma>0$ — натуральный логарифм среднеквадратичного отклонения).
- `*logis(x,l,s)` — логистическое распределение (`l` — математическое ожидание, `s>0` — параметр масштаба).
- `*nbinom(k,n,p)` — отрицательное биномиальное распределение (`n>0` — целый параметр, $0<p\le 1$).
- `*norm(x,μ,σ)` — нормальное распределение (μ — среднее значение, $\sigma>0$ — среднеквадратичное отклонение).
- `*pois(k,λ)` — распределение Пуассона ($\lambda>0$ — параметр).
- `*t(x,d)` — распределение Стьюдента (`d>0` — число степеней свободы).
- `*unif(x,a,b)` — равномерное распределение (`a<b` — границы интервала).
- `*weibull(x,s)` — распределение Вейбулла (`s>0` — параметр).

Примечание

Математический смысл каждой из четырех типов функций будет объяснен в следующем разделе на примере распределения Гаусса.

Вставку рассмотренных статистических функций в программы удобно осуществлять с помощью диалогового окна **Вставка функции** (Insert Function). Для этого необходимо выполнить следующие действия:

1. Установите курсор на место вставки функции в документе.
2. Вызовите диалоговое окно **Вставка функции** (Insert Function) нажатием кнопки **f(x)** на стандартной панели инструментов, или командой меню **Вставка / Функция** (Insert / Function), или нажатием клавиш <Ctrl>+<E>.
3. В списке **Категория функции** (Function Category) (рис. 12.2) выберите одну из категорий статистических функций. Категория **Плотность вероятности** (Probability Density) содержит встроенные функции для плотности вероятности, **Распределение вероятностей** (Probability Distribution) — для вставки функций или квантилей распределения, **Случайные числа** (Random Numbers) — для вставки функции генерации случайных чисел.
4. В списке **Имя функции** (Function Name) выберите функцию в зависимости от требующегося закона распределения. При выборе того или иного элемента списка в текстовых полях в нижней части окна будет появляться информация о назначении выбранной функции.
5. Нажмите кнопку **OK** для вставки функции в документ.

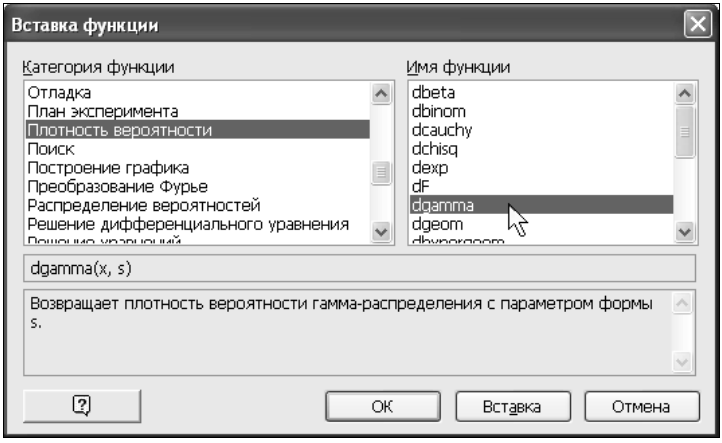


Рис. 12.2. Вставка функции распределения

12.1.2. Пример: нормальное (Гауссово) распределение

В теории вероятности доказано, что сумма различных независимых случайных слагаемых (независимо от их закона распределения) оказывается случайной величиной, распределенной согласно нормальному закону (так называемая *центральная пре-*

дельная теорема). Поэтому нормальное распределение хорошо моделирует самый широкий круг явлений, для которых известно, что на них влияют несколько независимых случайных факторов.

Перечислим еще раз встроенные функции, имеющиеся в Mathcad для описания нормального распределения вероятностей:

- `dnorm(x,μ,σ)` — плотность вероятности нормального распределения;
- `pnorm(x,μ,σ)` — функция нормального распределения;
- `cnorm(x)` — функция нормального распределения для `μ=0,σ=1`;
- `qnorm(P,μ,σ)` — обратная функция нормального распределения;
- `rnorm(M,μ,σ)` — вектор `M` независимых случайных чисел, каждое из которых имеет нормальное распределение:
  - `x` — значение случайной величины;
  - `P` — значение вероятности;
  - μ — математическое ожидание;
  - σ — среднеквадратичное отклонение.

Математическое ожидание и дисперсия являются, по сути, параметрами распределения. Плотность распределения для трех пар значений параметров показана на рис. 12.3. Напомним, что плотность распределения `dnorm` задает вероятность попадания случайной величины `x` в малый интервал от `x` до `x+Δx`. Таким образом, например, для первого графика (сплошная линия) вероятность того, что случайная величина `x` примет значение в окрестности нуля, приблизительно в три раза больше, чем вероятность того, что она примет значение в окрестности `x=2`. А значения случайной величины большие `5` и меньшие `−5` и вовсе очень маловероятны.

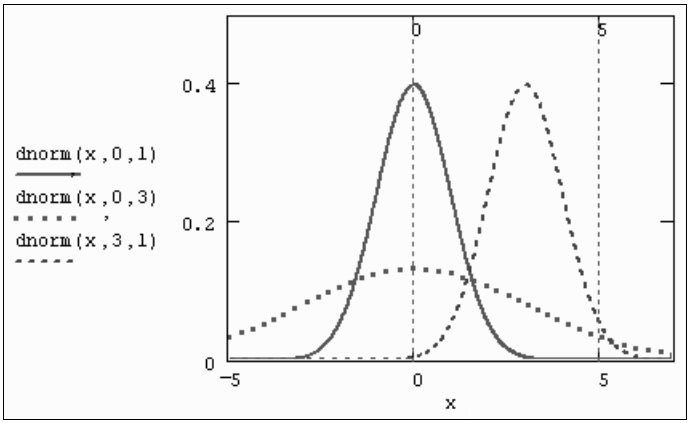


Рис. 12.3. Плотность вероятности нормальных распределений

Функция распределения `F(x)` (cumulative probability) — это вероятность того, что случайная величина примет значение, меньшее или равное `x`. Как следует из математического смысла, она является интегралом от плотности вероятности в пределах

от $-\infty$ до x. Функции распределения для упомянутых нормальных законов изображены на рис. 12.4. Функция, обратная F(x) (inverse cumulative probability), называемая еще *квантилем распределения*, позволяет по заданному аргументу p определить значение x, причем случайная величина будет меньше или равна x с вероятностью p.

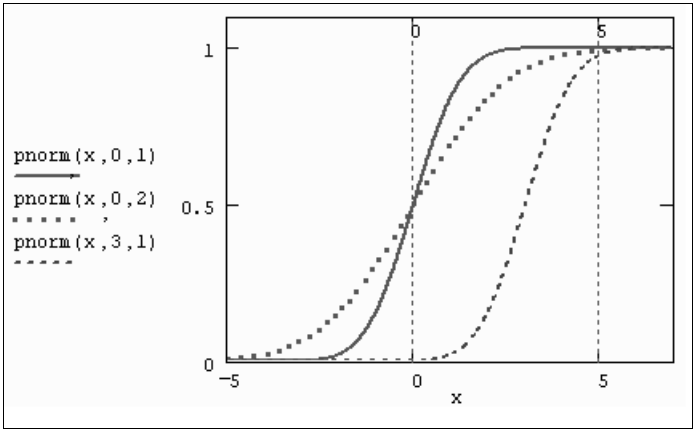


Рис. 12.4. Нормальные функции распределения

Примечание

Здесь и далее графики различных статистических функций, показанные на рисунках, получены с помощью Mathcad без каких-либо дополнительных выражений в рабочей области.

Приведем несколько примеров, позволяющих почувствовать математический смысл рассмотренных функций на примере случайной величины x, распределенной по нормальному закону с $\mu=0$ и $\sigma=1$ (листинги 12.1—12.5).

Листинг 12.1. Вероятность того, что x будет меньше 1.881

```
pnorm(1.881, 0, 1) = 0.97
```

Листинг 12.2. 97%-й квантиль нормального распределения

```
qnorm(0.97, 0, 1) = 1.881
```

Листинг 12.3. Вероятность того, что x будет больше 2

```
1 − pnorm(2, 0, 1) = 0.02275
```

Листинг 12.4. Вероятность того, что x будет находиться в интервале (2,3)

```
pnorm(3, 0, 1) − pnorm(2, 0, 1) = 0.021
```

$$\frac{1}{2}\cdot\left(\mathrm{erf}\left(\frac{3}{\sqrt{2}}\right)-\mathrm{erf}\left(\frac{2}{\sqrt{2}}\right)\right)=0.021$$

Листинг 12.5. Вероятность того, что $|x|<2$

$$\mathrm{pnorm}(2,0,1)-\mathrm{pnorm}(-2,0,1)=0.954$$

$$\mathrm{erf}\left(\frac{2}{\sqrt{2}}\right)=0.954$$

Обратите внимание, что задачи двух последних листингов решаются двумя разными способами. Второй из них связан с еще одной встроенной функцией `erf`, называемой *функцией ошибок* (или *интегралом вероятности*).

- `erf(x)` — функция ошибок.
- `erfc(x)≡1-erf(x)`.

Математический смысл функции ошибок ясен из листинга 12.5. Интеграл вероятности имеет всего один аргумент, в отличие от функции нормального распределения. Исторически последняя пересчитывалась через табулированный интеграл вероятности по формулам, приведенным в листинге 12.6 для произвольных значений параметров μ и σ (листинг 12.6).

Листинг 12.6. Вероятность того, что x будет в интервале (2,3)

$$\mu:=5 \qquad \sigma:=2$$

$$\mathrm{pnorm}(3,\mu,\sigma)-\mathrm{pnorm}(2,\mu,\sigma)=0.092$$

$$\frac{1}{2}\cdot\left(\mathrm{erf}\left(\frac{3-\mu}{\sigma\cdot\sqrt{2}}\right)-\mathrm{erf}\left(\frac{2-\mu}{\sigma\cdot\sqrt{2}}\right)\right)=0.092$$

Если вы имеете дело с моделированием методами Монте-Карло, то в качестве генератора случайных чисел с нормальным законом распределения применяйте встроенную функцию `rnorm`. В листинге 12.7 ее действие показано на примере создания двух векторов по `M=500` элементов в каждом с независимыми псевдослучайными числами $x1_i$ и $x2_i$, распределенными согласно нормальному закону. О характере распределения случайных элементов векторов можно судить по рис. 12.5. В дальнейшем мы будем часто сталкиваться с генерацией случайных чисел и расчетом их различных средних характеристик.

Листинг 12.7. Генерация двух векторов с нормальным законом распределения

$$\sigma:=1 \qquad \mu:=0$$

$$M:=500$$

$$x1:=\mathrm{rnorm}(M,\mu,\sigma)$$

$$x2:=\mathrm{rnorm}(M,\mu,\sigma)$$

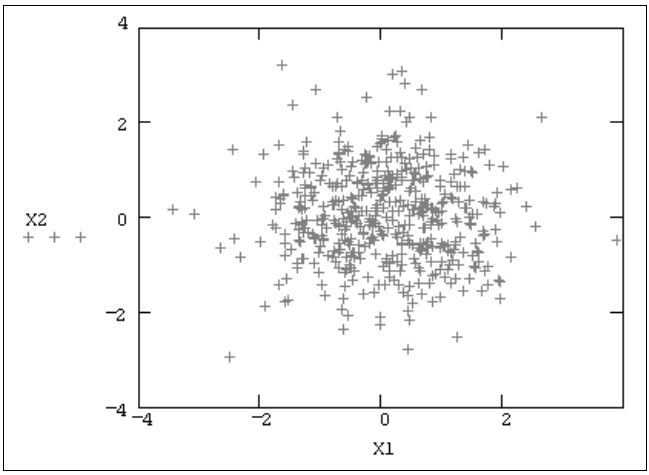


Рис. 12.5. Псевдослучайные числа с нормальным законом распределения (продолжение листинга 12.7)

12.2. Выборочные статистические характеристики

В большинстве статистических расчетов вы имеете дело с *выборками*: либо со случайными данными, полученными в ходе какого-либо эксперимента (которые выводятся из файла или печатаются непосредственно в документе), либо с результатами генерации случайных чисел, рассмотренными в предыдущих разделах встроенными функциями, моделирующими то или иное явление методом Монте-Карло (*см. разд. 12.3*). Рассмотрим возможности Mathcad по оценке функций распределения и расчету числовых характеристик случайных данных.

12.2.1. Гистограммы

Гистограммой называется график, аппроксимирующий по случайным данным плотность их распределения. При построении гистограммы область значений случайной величины `(a,b)` разбивается на некоторое количество `bin` сегментов, а затем подсчитывается процент попадания данных в каждый сегмент. Для построения гистограмм в Mathcad имеется несколько встроенных функций. Рассмотрим их, начиная с самой сложной по применению, чтобы лучше разобраться в возможностях каждой из функций.

Гистограммы с произвольными интервалами

❒ `hist(intvls,x)` — вектор частоты попадания данных в интервалы гистограммы:

- `intvls` — вектор, элементы которого задают сегменты построения гистограммы в порядке возрастания $a \le intvls_i < b$;
- `x` — вектор случайных данных.

Если вектор `intvls` имеет `bin` элементов, то и результат `hist` имеет столько же элементов. Построение гистограммы иллюстрируется листингом 12.8 и рис. 12.6.

Листинг 12.8. Построение гистограммы

$N := 1000$

$bin := 30$

$x := rnorm(N, 0, 1)$

$lower := floor(min(x))$

$upper := ceil(max(x))$

$$h := \frac{upper - lower}{bin}$$

$j := 0 .. bin$

$int_j := lower + h \cdot j$

$$f := \frac{1}{N \cdot h} \cdot hist(int, x)$$

Для анализа взято `N=1000` данных с нормальным законом распределения, созданных генератором случайных чисел (третья строка листинга). Далее определяются границы интервала `(upper,lower)`, содержащего внутри себя все случайные значения, и осуществляется его разбиение на количество `(bin)` одинаковых сегментов, начальные точки которых записываются в вектор `int` (предпоследняя строка листинга).

Примечание

В векторе `int` можно задать произвольные границы сегментов разбиения так, чтобы они имели разную ширину.

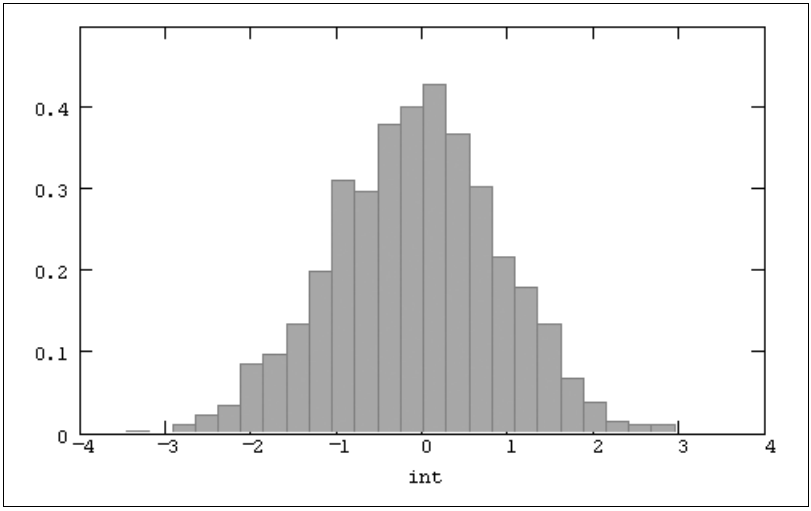


Рис. 12.6. Построение гистограммы (продолжение листинга 12.8)

Обратите внимание, что в последней строке листинга осуществлена нормировка значений гистограммы, с тем чтобы она правильно аппроксимировала плотность вероятности, также показанную на графике.

Гистограммы с равными интервалами

Если нет необходимости задавать сегменты гистограммы разной ширины, то удобнее воспользоваться упрощенным вариантом функции `hist`:

- `hist(bin,x)` — вектор частоты попадания данных в интервалы гистограммы:
  - `bin` — количество сегментов построения гистограммы;
  - `x` — вектор случайных данных.

Для того чтобы использовать этот вариант функции `hist` вместо предыдущего, достаточно заменить первый из ее аргументов в листинге 12.8 следующим образом:

$$f := \frac{1}{N \cdot h} \cdot \mathrm{hist}(\mathrm{int}, x)$$

Недостаток упрощенной формы функции `hist` в том, что по-прежнему необходимо дополнительно определять вектор сегментов построения гистограммы.

От этого недостатка свободна функция `histogram`:

- `histogram(bin,x)` — матрица гистограммы размера `bin×2`, состоящая из столбца сегментов разбиения и столбца частоты попадания в них данных:
  - `bin` — количество сегментов построения гистограммы;
  - `x` — вектор случайных данных.

Примеры использования функции `histogram` приведены в листинге 12.9 и на рис. 12.7. Сравнение с предыдущим листингом подчеркивает простоту построения гистограммы этим способом (стоит отметить, что в листинге 12.9, в отличие от предыдущего, мы не нормировали гистограмму).

Листинг 12.9. Упрощенный вариант построения гистограммы

```
N := 1000
bin := 10
x := rnorm(N, 0, 1)
f := histogram(bin, x)
```

Примечание

Для того чтобы назначить двумерному графику тип гистограммы, в диалоговом окне **Форматирование выбранного графика** (Formatting Currently Selected Graph) установите на вкладке **Трассировка** (Traces) тип списка **столбики** (bar) или **сплошные столбики** (solidbar). На рис. 12.6 и 12.7 применены установки второго типа: закрашенными столбиками (solidbar).

f =

	0	1
0	-2.916	0
1	-2.287	21
2	-1.658	63
3	-1.029	144
4	-0.4	212
5	0.229	252
6	0.858	181
7	1.487	94
8	2.116	25
9	2.745	7

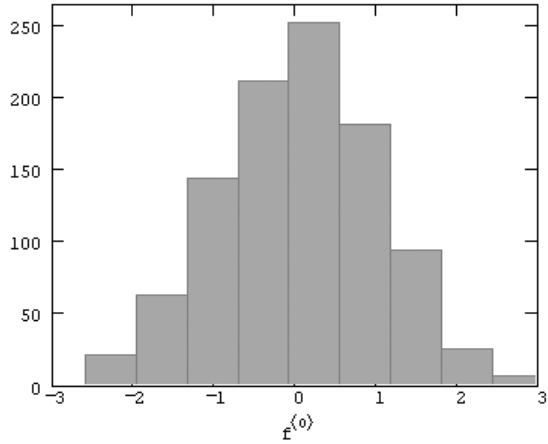


Рис. 12.7. График и матрица гистограммы (продолжение листинга 12.9)

12.2.2. Среднее и дисперсия

В Mathcad имеется ряд встроенных функций для расчетов числовых статистических характеристик рядов случайных данных:

- `mean(x)` — выборочное среднее значение;
- `median(x)` — выборочная *медиана* (median) — значение аргумента, которое делит гистограмму плотности вероятностей на две равные части;
- `var(x)` — выборочная *дисперсия* (variance);
- `stdev(x)` — *среднеквадратичное* (или *стандартное*) *отклонение* (standard deviation);
- `max(x),min(x)` — максимальное и минимальное значения выборки;
- `mode(x)` — наиболее часто встречающееся значение выборки;
- `Var(x),Stdev(x)` — выборочная дисперсия и среднеквадратичное отклонение в другой нормировке:
  - `x` — вектор (или матрица) с выборкой случайных данных.

Пример использования первых четырех функций приведен в листинге 12.10.

Листинг 12.10. Расчет числовых характеристик случайного вектора

```
N := 1000
x := runif(N, 0, 1)
mean(x) = 0.497
median(x) = 0.501
var(x) = 0.082
√Var(x) = 0.287   stdev(x) = 0.287
```

Определение статистических характеристик случайных величин приведено в листинге 12.11 на еще одном примере обработки выборки малого объема (по пяти данным). В том же листинге иллюстрируется применение еще двух функций, которые имеют смысл дисперсии и стандартного отклонения в несколько другой нормировке. Сравнивая различные выражения, вы без труда освоите связь между встроенными функциями.

> ***Внимание!***
>
> Осторожно относитесь к написанию первой литеры в этих функциях, особенно при обработке малых выборок (листинг 12.11).

Листинг 12.11. К определению статистических характеристик

$x := (5\ \ 2\ \ 14\ \ 3\ \ 2)^T$

$N := \mathrm{length}(x) \qquad N = 5$

$\frac{1}{N} \cdot \sum_{i=0}^{N-1} x_i = 5.2$

$m := \mathrm{mean}(x) \qquad m = 5.2$

$\mathrm{median}(x) = 3 \qquad \mathrm{mode}(x) = 2 \qquad \max(x) = 14 \qquad \min(x) = 2$

$\frac{1}{N} \cdot \sum_{i=0}^{N-1} (x_i - m)^2 = 20.56$

$\mathrm{var}(x) = 20.56 \qquad \mathrm{Var}(x) \cdot \frac{N-1}{N} = 20.56$

$\sqrt{\mathrm{var}(x)} = 4.534 \qquad \mathrm{stdev}(x) = 4.534$

$\mathrm{Stdev}(x) \cdot \sqrt{\frac{N-1}{N}} = 4.534$

$\frac{1}{N-1} \cdot \sum_{i=0}^{N-1} (x_i - m)^2 = 25.7$

$\mathrm{Var}(x) = 25.7 \qquad \mathrm{var}(x) \cdot \frac{N}{N-1} = 25.7$

$\sqrt{\mathrm{Var}(x)} = 5.07 \qquad \mathrm{Stdev}(x) = 5.07$

$\mathrm{stdev}(x) \cdot \sqrt{\frac{N}{N-1}} = 5.07$

Иногда в статистике встречаются и иные функции, например, помимо арифметического среднего, применяются другие средние значения:

- `gmean(x)` — геометрическое среднее выборки случайных чисел;
- `hmean(x)` — гармоническое среднее выборки случайных чисел.

Математическое определение этих функций и пример их использования в Mathcad приведены в листинге 12.12.

Листинг 12.12. Вычисление различных средних значений

$N := 10$

$x := \mathrm{runif}(N, 0, 1)$

$\frac{1}{N} \cdot \sum_{i=0}^{N-1} x_i = 0.338$ $\qquad$ $\mathrm{mean}(x) = 0.338$

$\left(\frac{1}{N} \cdot \sum_{i=0}^{N-1} \frac{1}{x_i}\right)^{-1} = 0.012$ $\qquad$ $\mathrm{hmean}(x) = 0.012$

$\sqrt[N]{\prod_{i=0}^{N-1} x_i} = 0.171$ $\qquad$ $\mathrm{gmean}(x) = 0.171$

12.2.3. Примеры: выборочная оценка дисперсии и среднего нормальной случайной величины

Типовые задачи математической статистики связаны с получением тех или иных интервальных и точечных оценок различных параметров случайной выборки. Приведем пример двух задач, иллюстрирующих назначение и принципы применения введенных в предыдущих разделах статистических функций.

Интервальная оценка дисперсии

Требуется определить числовой интервал (L,U), внутри которого будет лежать с вероятностью $1-\alpha=75\%$ дисперсия нормальной случайной величины, исходя из объема выборки в N чисел. Эта задача решается в статистике с помощью χ^2-распределения (листинг 12.13).

Листинг 12.13. Интервальное оценивание дисперсии

$N := 50$

$x := \mathrm{rnorm}(N, 0, 1)$

$\alpha := 0.25$ $\qquad$ $1 - \alpha = 0.75$

$\chi^2{}_0 := \mathrm{qchisq}\left(\frac{\alpha}{2}, N-1\right)$ $\qquad$ $\chi^2{}_0 = 37.901$

$\chi^2{}_1 := \mathrm{qchisq}\left(1 - \frac{\alpha}{2}, N-1\right)$ $\qquad$ $\chi^2{}_1 = 60.53$

$L := \frac{(N-1) \cdot \mathrm{Stdev}(x)^2}{\chi^2{}_1}$ $\qquad$ $U := \frac{(N-1) \cdot \mathrm{Stdev}(x)^2}{\chi^2{}_0}$

$L = 0.662$ $\qquad$ $U = 1.057$

Указанный интервал называется (1-α) *доверительным интервалом*. Обратите внимание на использование при решении данной задачи функции Stdev (с прописной буквы) для расчета выборочного стандартного отклонения. В статистике часто встречаются выражения, которые более удобно записывать через функции в такой нормировке, именно для этого они и появились в Mathcad.

Проверка статистических гипотез

В статистике рассматривается огромное число задач, связанных с проверкой тех или иных гипотез H. Разберем пример простой гипотезы. Пусть имеется выборка N чисел с нормальным законом распределения и неизвестными дисперсией и математическим ожиданием. Требуется принять или отвергнуть гипотезу H о том, что математическое ожидание закона распределения равно некоторому числу μ0=0.2.

Задачи проверки гипотез требуют задания *уровня критерия* проверки гипотезы α, который описывает вероятность ошибочного отклонения истинной H. Если взять α очень малым, то гипотеза, даже если она ложная, будет почти всегда приниматься; если, напротив, взять α близким к 1, то критерий будет очень строгим, и гипотеза, даже верная, скорее всего, будет отклонена.

В нашем случае гипотеза состоит в том, что μ0=0.2, а альтернатива — что μ0≠0.2. Оценка математического ожидания, как следует из курса классической статистики, решается с помощью распределения Стьюдента с параметром N-1 (этот параметр называется *степенью свободы распределения*).

Для проверки гипотезы (листинг 12.14) рассчитывается (α/2)-квантиль распределения Стьюдента T, который служит критическим значением для принятия или отклонения гипотезы. Если соответствующее выборочное значение t по модулю меньше T, то гипотеза принимается (считается верной). В противном случае гипотезу следует отвергнуть.

Листинг 12.14. Проверка гипотезы о математическом ожидании при неизвестной дисперсии

$N := 50$

$x := \mathrm{rnorm}(N, 0, 1)$

$\alpha := 0.1$ $\qquad 1 - \alpha = 0.9$

$\mu 0 := 0.2$

$$t := \frac{\mathrm{mean}(x) - \mu 0}{\left(\frac{\mathrm{Stdev}(x)}{\sqrt{N}}\right)} \qquad t = -1.883$$

$$T := \mathrm{qt}\left(1 - \frac{\alpha}{2}, N - 1\right) \qquad T = 1.677$$

$|t| < T = 0$

В последней строке листинга вычисляется истинность или ложность условия, выражающего решение задачи. Поскольку условие оказалось ложным (равным не 1, а 0), то гипотезу необходимо отвергнуть.

На рис. 12.8 показано распределение Стьюдента с `N-1` степенью свободы, вместе с критическими значениями, определяющими соответствующий интервал. Если `t` (оно тоже показано на графике) попадает в него, то гипотеза принимается; если не попадает (как произошло в данном случае) — отвергается. Если увеличить α, ужесточив критерий, то границы интервала будут сужаться по сравнению с показанными на рисунке.

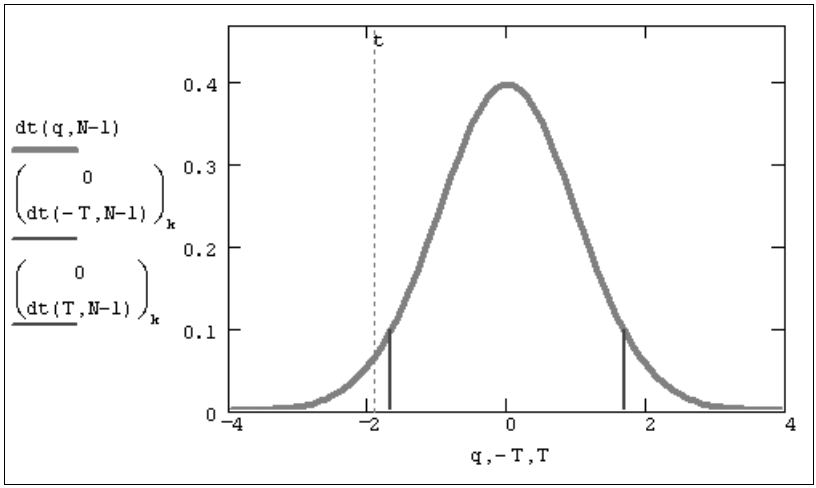


Рис. 12.8. К задаче проверки статистических гипотез
(продолжение листинга 12.14)

В листинге 12.15 приводится альтернативный способ проверки той же самой гипотезы, связанный с вычислением значения не квантиля, а самого распределения Стьюдента.

Листинг 12.15. Другой вариант проверки гипотезы (продолжение листинга 12.14)

$$pt(t, N-1) = 0.033$$

$$\frac{\alpha}{2} < pt(t, N-1) < 1 - \frac{\alpha}{2} = 0$$

Мы разобрали только два характерных примера статистических вычислений. Однако с помощью Mathcad легко решаются самые разнообразные задачи математической статистики.

12.2.4. Корреляция

Функции, устанавливающие связь между парами двух случайных векторов, называются *ковариацией* и корреляцией (или, по-другому, *коэффициентом корреляции*). Они различаются нормировкой, как следует из их определения (листинг 12.16):

- `corr(x)` — коэффициент корреляции двух выборок;
- `cvar(x)` — ковариация двух выборок:
  - `x1,x2` — векторы (или матрицы) одинакового размера с выборками случайных данных.

Листинг 12.16. Расчет ковариации и корреляции

$N := 100 \qquad \sigma := 1$

$x1 := \mathrm{rnorm}(N, 0, \sigma) \qquad x2 := \mathrm{rnorm}(N, 0, \sigma)$

$m1 := \mathrm{mean}(x1) \qquad m2 := \mathrm{mean}(x2)$

$\sigma 1 := \mathrm{stdev}(x1) \qquad \sigma 2 := \mathrm{stdev}(x2)$

$$\frac{1}{N} \cdot \sum_{i=0}^{N-1} \left[(x1_i - m1) \cdot (x2_i - m2) \right] = 5.255 \times 10^{-3}$$

$$\mathrm{cvar}(x1, x2) = 5.255 \times 10^{-3}$$

$$\frac{\mathrm{cvar}(x1, x2)}{\sigma 1 \cdot \sigma 2} = 5.337 \times 10^{-3}$$

$$\mathrm{corr}(x1, x2) = 5.337 \times 10^{-3}$$

Примечание

Как видно, полученное значение корреляции близко к нулю, т. е. псевдослучайные числа генерируются практически независимо. О том, как создать массив коррелированных чисел, написано ниже (*см. разд. 12.3.2*).

12 12.2.5. Функции корреляционного анализа сигналов

В Mathcad 12 появились две функции, связанные с корреляционной обработкой сигналов и изображений:

- `correl(x,y)` — вектор, представляющий значения коэффициента ковариации двух векторов;
- `correl2d(A,K)` — матрица, равная ковариации матрицы-аргумента и матрицы-окна:
  - `x,y` — векторы;
  - `A` — матрица-прототип;
  - `K` — матрица-окно (размера, меньшего чем `A`).

Смысл действия одномерной функции `correl` заключается в последовательном сдвиге одного вектора относительно другого, перемножении и суммировании их элементов, стоящих в таком положении друг напротив друга. Это не совсем ковариация в терминах математической статистики, поскольку не осуществляется нор-

мировки (деления полученной суммы на число перемноженных элементов). Работа двумерной функции `correl2d` состоит в последовательном позиционировании "маленькой" матрицы (окна) на фоне "крупной" (прототипа), перемножении их элементов, находящихся друг над другом, и суммировании их. В результате получается элемент матрицы корреляции, соответствующий центрам наложения матрицы-прототипа и матрицы-окна. Обе эти функции играют определенную роль в задачах обработки сигналов.

12.2.6. Коэффициенты асимметрии и эксцесса

Коэффициент асимметрии задает степень асимметричности плотности вероятности относительно оси, проходящей через ее центр тяжести. Коэффициент асимметрии определяется третьим центральным моментом распределения. В любом симметричном распределении с нулевым математическим ожиданием, например нормальным, все нечетные моменты, в том числе и третий, равны нулю, поэтому коэффициент асимметрии тоже равен нулю.

Степень сглаженности плотности вероятности в окрестности главного максимума задается еще одной величиной — *коэффициентом эксцесса*. Он показывает, насколько острую вершину имеет плотность вероятности по сравнению с нормальным распределением. Если коэффициент эксцесса больше нуля, то распределение имеет более острую вершину, чем распределение Гаусса, если меньше нуля, то более плоскую.

Для расчета коэффициентов асимметрии и эксцесса в Mathcad имеются две встроенные функции:

- `kurt(x)` — коэффициент эксцесса (*kurtosis*) выборки случайных данных `x`;
- `skew(x)` — коэффициент асимметрии (*skewness*) выборки случайных данных `x`.

Примеры расчета коэффициентов асимметрии и эксцесса для распределения Вейбулла приведены в листинге 12.17.

Листинг 12.17. Расчет выборочных коэффициентов асимметрии и эксцесса

```
x := rweibull(1000, 1.5)
skew(x) = 1.216
kurt(x) = 1.89
```

12.2.7. Статистические функции матричного аргумента

Все рассмотренные примеры работы статистических функций относились к векторам, элементы которых были случайными числами. Но точно так же все эти функции применяются и по отношению к выборкам случайных данных, сгруппированных в матрицы. При этом статистические характеристики рассчитываются для совокупности всех элементов матрицы, без разделения ее на строки и столбцы. Например, если матрица имеет размерность `M×N`, то и объем выборки будет равен `M·N`.

Соответствующий пример вычисления среднего значения приведен в листинге 12.18. В его первой строке определяется матрица данных `x` размера `4×2`. Действие встроенной функции `mean` матричного аргумента (последняя строка листинга) иллюстрируется явным суммированием элементов матрицы `x` (предпоследняя строка). Действие прочих встроенных функций на матрицы совершенно аналогично действию их на векторы (листинг 12.19).

Листинг 12.18. Вычисление среднего значения элементов матрицы

$$x := \begin{pmatrix} 1.0 & 4 \\ 1.5 & 7 \\ 0.9 & 1.2 \\ 1.2 & 12 \end{pmatrix}$$

$M := \mathrm{rows}(x) \qquad M = 4$

$N := \mathrm{cols}(x) \qquad N = 2$

$$\frac{1}{M \cdot N} \cdot \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x_{i,j} = 3.6$$

$\mathrm{mean}(x) = 3.6$

Листинг 12.19. Действие различных статистических функций на матрицу

$\mathrm{median}(x) = 1.35$

$\mathrm{mode}(x) = 1.2$

$\mathrm{var}(x) = 14.033$

$\mathrm{Var}(x) = 16.037$

$\mathrm{stdev}(x) = 3.746$

$\mathrm{Stdev}(x) = 4.005$

Примечание

Некоторые статистические функции (например, вычисления ковариации) имеют два аргумента. Они также могут быть матрицами, но, в соответствии со смыслом функции, должны иметь одинаковую размерность.

Большинству статистических функций позволяется иметь в качестве аргументов даже не одну матрицу, а любое количество матриц, векторов и скаляров. Числовые характеристики будут рассчитаны для всей совокупности значений аргументов функции. Соответствующий пример приведен в листинге 12.20.

Листинг 12.20. Статистические функции нескольких аргументов

$x := (1 \quad -1 \quad 4)$

$$y := \begin{pmatrix} 3 \\ 8 \end{pmatrix}$$

$$z := \begin{pmatrix} 3 & 7 \\ 11 & 4 \end{pmatrix}$$

mean(x, y, z) = 4.444

stdev(x, 5, 77) = 29.976

median(2) = 2

mode(y, z) = 3

12.3. Методы Монте-Карло

Для моделирования различных физических, экономических и прочих эффектов широко распространены методы, называемые *методами Монте-Карло*, обязанные своим названием европейскому центру азартных игр, основанных на случайных событиях. Основная идея этих методов состоит в создании определенной последовательности *случайных* чисел, моделирующей тот или иной эффект, например шум в физическом эксперименте, случайную динамику биржевых индексов и т. п. Фактически все содержание предыдущего раздела (*см. разд. 12.2*) являлось примером реализации методов Монте-Карло, поскольку содержало расчет тех или иных статистических характеристик имитационных данных, полученных при помощи генератора случайных чисел.

Рассмотрим еще несколько типовых задач, относящихся к методам Монте-Карло, попутно изучая соответствующие возможности, заложенные в Mathcad.

12.3.1. Генерация псевдослучайных чисел

Как уже отмечалось (*см. разд. 12.1.1*), для генерации `M`-компонентного вектора независимых псевдослучайных чисел имеется ряд встроенных функций, реализующих различные типы статистических распределений и имеющих вид `r*(M,par)`, где `*` — идентификатор, а `par` — список параметров конкретного распределения. В частности, генератор нормальных псевдослучайных чисел был рассмотрен ранее (*см. разд. 12.1.2*), а для равномерного распределения предусмотрено две встроенных функции:

- `runif(M,a,b)` — вектор `M` независимых случайных чисел, каждое из которых имеет равномерное распределение;
- `rnd(x)` — случайное число, имеющее равномерную плотность распределения на интервале `(0,x)`:
  - `x` — значение случайной величины;
  - `P` — значение вероятности;
  - `(a,b)` — интервал, на котором случайная величина распределена равномерно.

Чаще всего в несложных программах применяется последняя функция, которая приводит к генерации одного псевдослучайного числа. Наличие такой встроенной функции в Mathcad — дань традиции, применяемой в большинстве сред програм-

мирования. Возможно, именно скалярный тип этой функции определяет простоту и привычность ее использования в расчетах моделей типа Монте-Карло.

Рисунок 12.9 демонстрирует широко распространенный прием моделирования, основанный на смеси некоторого полезного сигнала `f(x)` и шумовой компоненты σ, что характерно для типичного физического эксперимента. В качестве шума используется серия равномерно распределенных псевдослучайных чисел. Различные соотношения интенсивностей сигнала и шума определяют различные условия модельной задачи и позволяют эффективно протестировать алгоритмы, которые разработаны для ее решения

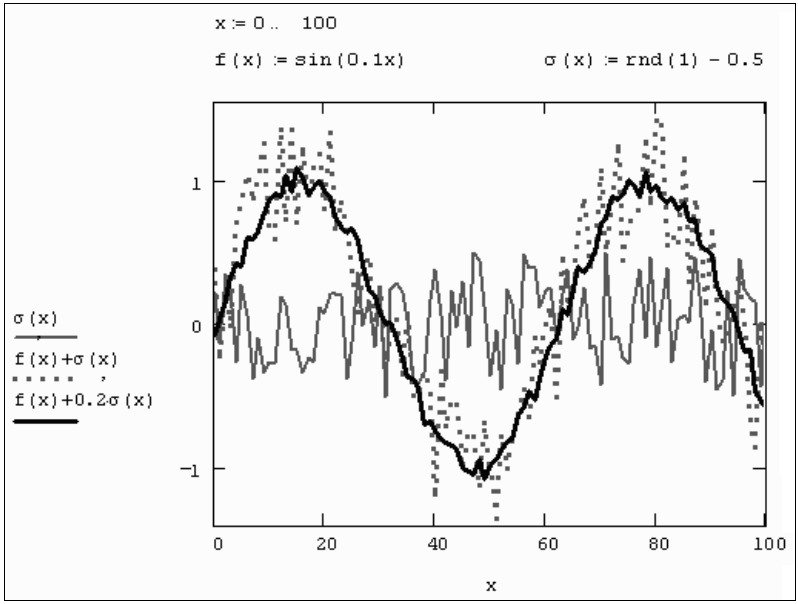


Рис. 12.9. Модель сигнал/шум с равномерным законом распределения

ⓘ Параметры генераторов псевдослучайных чисел

В Mathcad применяются типичные алгоритмы генерации последовательностей псевдослучайных чисел, которые используют в качестве "отправной точки" некоторое *инициирующее* (или *отправное*) *значение* (seed value). Это начальное значение используется для того, чтобы совершить над ним определенные математические действия (к примеру, взять остаток от деления на некоторое другое число) и получить в итоге первое псевдослучайное число последовательности. Затем те же математические операции совершаются с первым числом для получения второго и т. д.

Изменение инициирующего значения возможно в диалоговом окне **Параметры документа** (Worksheet Options), вызываемом командой **Инструменты / Параметры документа** (Tools / Worksheet Options), либо непосредственно в документе при помощи встроенной функции:

- `seed(x)` — функция установки нового начального значения для генератора псевдослучайных чисел:
  - `x` — новое начальное значение для генератора псевдослучайных чисел (целое число от `1` до `2147483647`).

Несложно догадаться, что если использовать все время одно и то же начальное значение генератора псевдослучайных чисел, то, открывая всякий раз новый документ со встроенной функцией получения тех или иных псевдослучайных чисел, будет выдаваться в точности одна и та же их последовательность. Сами числа внутри последовательности будут "почти" случайными (значимость этого "почти" будет зависеть только от качества алгоритма генерации), но вот сама последовательность при каждом открытии документа будет одной и той же.

12.3.2. Генерация коррелированных выборок

До сих пор мы рассматривали наиболее простой случай применения генераторов независимых случайных чисел. В методах Монте-Карло часто требуется создавать случайные числа с определенной *корреляцией*. Приведем пример программы, создающей два вектора `x1` и `x2` одинакового размера и одним и тем же распределением, случайные элементы которых попарно коррелированны с коэффициентом корреляции `R` (листинг 12.21).

Листинг 12.21. Генерация попарно коррелированных случайных чисел

$\sigma := 3$ $\qquad$ $N := 1000$

$R := 0.4$

$x1 := \mathrm{rnorm}(N, 0, \sigma)$

$x2 := R \cdot x1 + \sqrt{1 - R^2} \cdot \mathrm{rnorm}(N, 0, \sigma)$

$\mathrm{corr}(x1, x2) = 0.415$

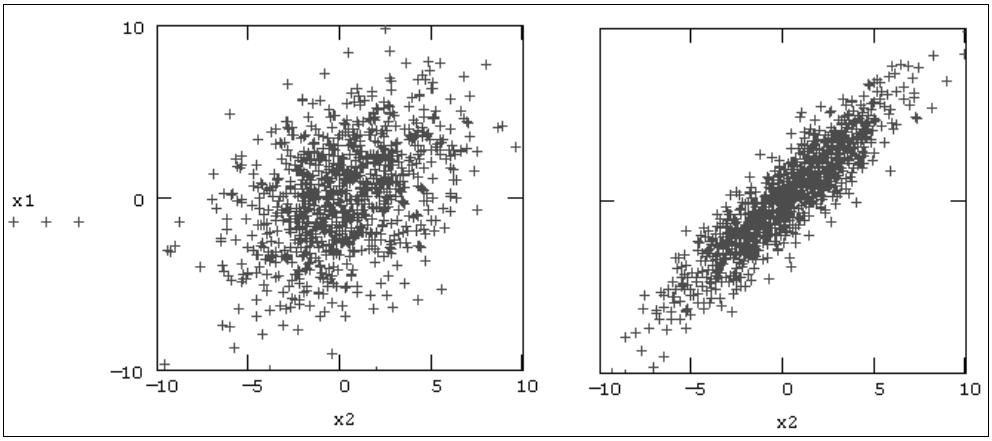


Рис. 12.10. Псевдослучайные числа с корреляцией `R=0.4` (продолжение листинга 12.21) и `R=0.9`

Результат действия программы для R=0.4 показан на рис. 12.10 (слева). Сравните полученную выборку с правым графиком, построенным для высокой корреляции (R=0.9), и с рис. 12.5 (*см. разд. 12.1.2*) для независимых данных, т. е. R=0.

🛠 12.3.3. Моделирование случайного процесса

Встроенные функции для генерации случайных чисел создают выборку из случайных данных A_i. Часто требуется создать непрерывную или дискретную случайную функцию A(t) одной или нескольких переменных (*случайный процесс* или *случайное поле*), значения которой будут упорядочены относительно своих переменных. Создать псевдослучайный процесс можно способом, представленным в листинге 12.22.

Листинг 12.22. Генерация псевдослучайного процесса

```
N := 20
τ := 0.5
Tmax := (N − 1) · τ
j := 0 .. N − 1
T_j := j · τ
x := rnorm(N, 0, 1)
KS1 := cspline(T, x)
A(t) := interp(KS1, T, x, t)
```

В первых строках листинга 12.22 определено количество N независимых случайных чисел, которые будут впоследствии сгенерированы, и радиус временной корреляции τ. В следующих трех строках определяются моменты времени T_j, которым будут отвечать случайные значения $A(t_j)$. Создание нормального случайного процесса сводится к генерации обычным способом вектора независимых случайных чисел x и построению интерполяционной зависимости в промежутках между ними. В листинге 12.22 используется сплайн-интерполяция (*см. разд. 13.1*).

В результате получается случайный процесс A(t), радиус корреляции которого определяется расстоянием τ между точками, для которых строится интерполяция. График случайного процесса A(t) вместе с исходными случайными числами показан на рис. 12.11. Случайное поле можно создать несколько более сложным способом с помощью многомерной интерполяции.

> ***Примечание***
>
> Простой пример генерации двумерного случайного поля вы найдете на компакт-диске, прилагаемом к книге.

К случайным процессам, сгенерированным таким способом, как и к данным эксперимента, применяются любые статистические методы обработки, например, корреляционный или спектральный анализ. Приведем в качестве примера листинг 12.23,

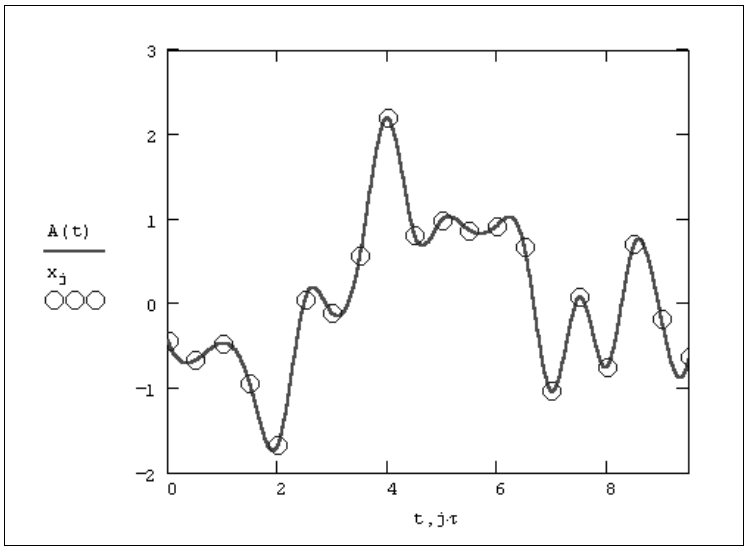


Рис. 12.11. Псевдослучайный процесс (продолжение листинга 12.22)

показывающий, как организовать расчет корреляционной функции случайного процесса.

Листинг 12.23. Дискретизация случайного процесса и вычисление корреляционной функции (продолжение листинга 12.19)

$\Delta := 0.02$

$M := 20$

$n := \mathrm{floor}\left(\frac{Tmax}{\Delta}\right)$ $\qquad n = 475$

$j := 0 .. n$

$Y_j := A(\Delta \cdot j)$

$m := \mathrm{mean}(Y)$

$D := \mathrm{var}(Y)$ $\qquad D = 0.785$

$$R(j) := \frac{1}{(n - 2 \cdot M)} \cdot \sum_{i = M}^{n-M} \frac{(Y_{i+j} - m) \cdot (Y_i - m)}{D}$$

$R(0) = 1.025$

Дискретизация интервала (0,Tmax) для случайного процесса A(t) произведена с различным элементарным интервалом Δ (первая строка листинга). В зависимости от значения Δ получается различный объем n выборки случайных чисел Y_i, являющихся значениями случайной функции A(t) в точках дискретизации. В последних четырех строках определяются различные характеристики случайной величины Y, являющиеся, по сути, характеристиками случайного процесса A(t). График рассчитанной в 2·M+1 точках корреляционной функции R(j) показан на рис. 12.12.

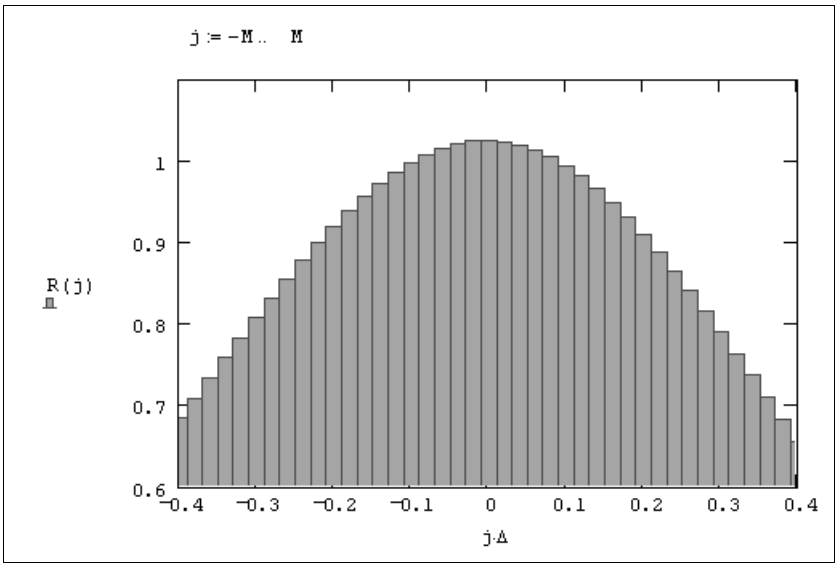


Рис. 12.12. Корреляционная функция (продолжение листингов 12.22 и 12.23)

Примечание

Внимательному читателю предлагается самостоятельно ответить на вопрос: почему при таком расчете корреляционной функции ее значение `R(0)` не равно 1, как должно быть по определению?

⑮ 12.3.4. Новые функции генерации случайных чисел

В Mathcad 15 появилось несколько новых встроенных функций, предназначенных для моделирования методом Монте-Карло и входящих в группу **План эксперимента** (Design of Experiment).

Во-первых, это функции, возвращающие вектор из `n` псевдослучайных чисел имеющих следующее стандартное распределение *(сравните с разд. 12.1.1)*.

- ❒ `Normal(n)≡rnorm(n,0,1)` — нормальное распределение;
- ❒ `LogNormal(n)≡rlnorm(n,0,1)` — логарифмически нормальное распределение;
- ❒ `Uniform(n)≡runif(n,-6,6)` — равномерное распределение;
- ❒ `Weibull(n)≡rweibull(n,1)` — распределение Вейбулла.

Во-вторых, еще одна встроенная функция предназначена для быстрого вычисления функции от нескольких случайных величин, имеющих заданный закон распределения.

- ❒ `montecarlo(F,n,Rvals,Limits,dist)` — матрица, содержащая массив (`n` строк) псевдослучайных чисел, вычисляемых по заданной формуле:
  - `F(v)` — некоторая функция векторного аргумента `v`, где элементы `v` — случайные числа;

- `n` — объем выборки;
- `Rvals` — матрица со строками, составленными из имен переменных (1-й столбец), их математическими ожиданиями (2-й столбец) и стандартными отклонениями (3-й);
- `Limits` — матрица, определяющая допустимые пределы случайных чисел;
- `dist` — вектор, определяющий закон распределения для каждой переменной.

В листинге 12.24 приведен пример вычисления случайной выборки из 5 чисел, рассчитываемой путем перемножения гауссового `X` и равномерно распределенного `Y` случайных чисел (причем предполагается их неотрицательность, что задает шестая строка листинга). Результат, представленный в последней строке листинга, выводит созданные псевдослучайные числа по столбцам матрицы.

Листинг 12.24. Моделирование методом Монте-Карло

$f(x) := x_0 \cdot x_1$

$$dist := \begin{pmatrix} \text{Normal} \\ \text{Uniform} \end{pmatrix}$$

$\mu X := 2 \qquad \sigma X := 1$

$\mu Y := 10 \qquad \sigma Y := 2$

$$Rvals := \begin{pmatrix} \text{"X"} & \mu X & \sigma X \\ \text{"Y"} & \mu Y & \sigma Y \end{pmatrix}$$

$$lim := \begin{pmatrix} 0 & \text{NaN} \\ 0 & \text{NaN} \end{pmatrix}$$

$M := montecarlo(f, 5, Rvals, lim, dist)$

$$M = \begin{pmatrix} \text{"X"} & \text{"Y"} & \text{"Output"} \\ 1.561 & 17.748 & 27.705 \\ 4.074 & 5.296 & 21.574 \\ 1.049 & 8.819 & 9.247 \\ 3.241 & 20.942 & 67.864 \\ 1.879 & 18.693 & 35.131 \end{pmatrix}$$

ГЛАВА 13

Интерполяция, регрессия, фильтрация

Посвятим данную главу самым простым методам обработки данных — интерполяции-экстраполяции и регрессии. Будем считать, что основным объектом исследования будет выборка экспериментальных данных, которые, чаще всего, представляются в виде массива, состоящего из пар чисел (x_i, y_i). В связи с этим возникает задача аппроксимации дискретной зависимости $y(x_i)$ непрерывной функцией $f(x)$. Функция $f(x)$, в зависимости от специфики задачи, может отвечать различным требованиям:

- $f(x)$ должна проходить через точки (x_i, y_i), т. е. $f(x_i)=y_i, i=1...n$. В этом случае (*см. разд. 13.1*) говорят об *интерполяции* данных функцией $f(x)$ во внутренних точках между x_i или *экстраполяции* за пределами интервала, содержащего все x_i;
- $f(x)$ должна некоторым образом (например, в виде определенной аналитической зависимости) приближать $y(xi)$, не обязательно проходя через точки (xi, yi). Такова постановка задачи *регрессии* (*см. разд. 13.2*), которую во многих случаях также можно назвать *сглаживанием* данных. Вообще говоря, сглаживание является частным случаем *фильтрации* данных (*см. разд. 13.3*), основанной на уменьшении шумовой компоненты измерений.

Различные виды построения аппроксимирующей зависимости $f(x)$ иллюстрирует рис. 13.1. На нем исходные данные обозначены кружками, интерполяция отрезками прямых линий — пунктиром, линейная регрессия — наклонной прямой линией, а фильтрация — жирной гладкой кривой. Эти зависимости приведены в качестве примера и отражают лишь малую часть возможностей Mathcad по обработке данных. Вообще говоря, в Mathcad имеется целый арсенал встроенных функций, позволяющий осуществлять самую различную регрессию, интерполяцию и экстраполяцию.

13.1. Интерполяция

Для построения интерполяции-экстраполяции в Mathcad имеется несколько встроенных функций, позволяющих "соединить" точки выборки данных (x_i, y_i) кривой

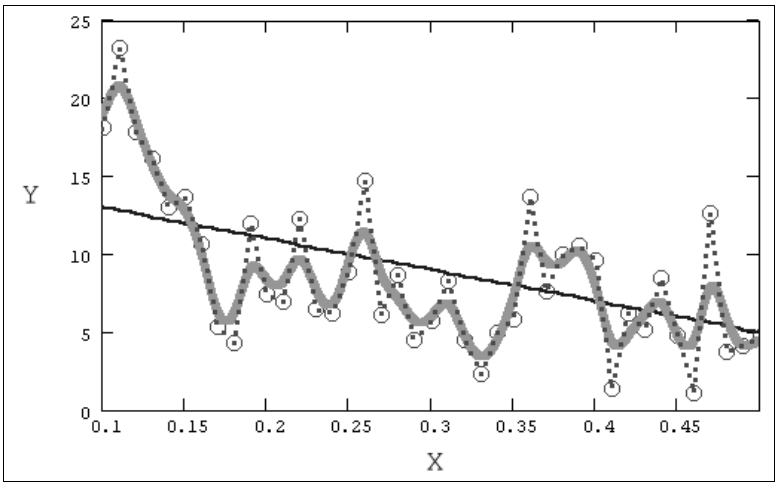


Рис. 13.1. Разные задачи аппроксимации данных

разной степени гладкости. По определению интерполяция означает построение функции `A(x)`, аппроксимирующей зависимость `y(x)` в промежуточных точках (между x_i). Поэтому интерполяцию еще по-другому называют *аппроксимацией*. В точках x_i значения интерполяционной функции должны совпадать с исходными данными, т. е. $A(x_i)=y(x_i)$.

Примечание

Везде в этом разделе, при рассказе о различных типах интерполяции, будем использовать вместо обозначения `A(x)` другое имя ее аргумента `A(t)`, чтобы не путать вектор данных `x` и скалярную переменную `t`.

13.1.1. Линейная интерполяция

Самый простой вид интерполяции — линейная, которая представляет искомую зависимость `A(x)` в виде ломаной линии. Интерполирующая функция `A(x)` состоит из отрезков прямых, соединяющих точки (рис. 13.2).

Для построения линейной интерполяции служит встроенная функция `linterp` (листинг 13.1):

- `linterp(x,y,t)` — функция, аппроксимирующая данные векторов `x` и `y` кусочно-линейной зависимостью:
  - `x` — вектор действительных данных аргумента;
  - `y` — вектор действительных данных значений того же размера;
  - `t` — значение аргумента, при котором вычисляется интерполирующая функция.

Внимание!

Элементы вектора `x` должны быть определены в порядке возрастания, т. е. $x_1<x_2<x_3<...<x_N$.

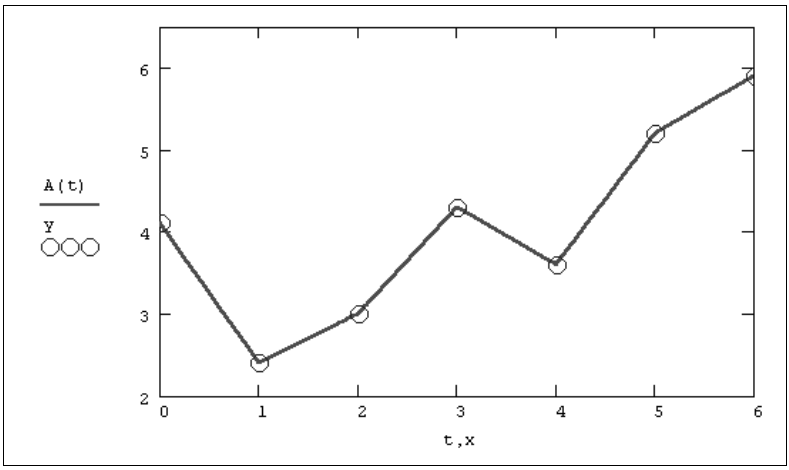


Рис. 13.2. Линейная интерполяция (продолжение листинга 13.1)

Листинг 13.1. Линейная интерполяция

```
x := (0  1  2  3  4  5  6)^T
y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T
A(t) := linterp(x, y, t)
```

Как видно из листинга, чтобы осуществить линейную интерполяцию, надо выполнить следующие действия:

1. Ввести векторы данных `x` и `y` (первые две строки листинга).
2. Определить функцию `linterp(x,y,t)`.

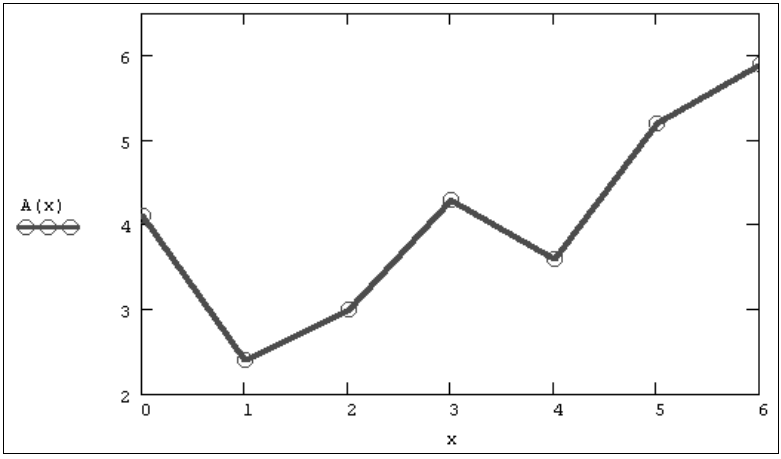


Рис. 13.3. Обычное построение графика функции от векторной переменной `x` (продолжение листинга 13.1)

3. Вычислить значения этой функции в требуемых точках, например `linterp(x,y,2.4)=3.52` или `linterp(x,y,6)=5.9`, или построить ее график, как показано на рис. 13.2.

> ***Примечание***
>
> Обратите внимание, что функция `A(t)` на графике имеет аргумент `t`, а не `x`. Это означает, что функция `A(t)` вычисляется не только при значениях аргумента (т. е. в семи точках), а при гораздо большем числе аргументов в интервале `(0,6)`, что автоматически обеспечивает Mathcad. Просто в данном случае эти различия незаметны, т. к. при обычном построении графика функции `A(x)` от векторного аргумента `x` (рис. 13.3) Mathcad по умолчанию соединяет точки графика прямыми линиями (т. е. скрытым образом осуществляет их линейную интерполяцию).

13.1.2. Кубическая сплайн-интерполяция

В большинстве практических приложений желательно соединить экспериментальные точки не ломаной линией, а гладкой кривой.

Лучше всего для этих целей подходит интерполяция кубическими сплайнами, т. е. отрезками кубических парабол (рис. 13.4):

- `interp(s,x,y,t)` — функция, аппроксимирующая данные векторов `x` и `y` кубическими сплайнами:
  - `s` — вектор вторых производных, созданный одной из сопутствующих функций `cspline`, `pspline` или `lspline`;
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - `y` — вектор действительных данных значений того же размера;
  - `t` — значение аргумента, при котором вычисляется интерполирующая функция.

Сплайн-интерполяция в Mathcad реализована чуть сложнее линейной. Перед применением функции `interp` необходимо предварительно определить первый из ее аргументов — векторную переменную `s`. Делается это при помощи одной из трех встроенных функций тех же аргументов `(x,y)`:

- `lspline(x,y)` — вектор значений коэффициентов линейного сплайна;
- `pspline(x,y)` — вектор значений коэффициентов квадратичного сплайна;
- `cspline(x,y)` — вектор значений коэффициентов кубического сплайна:
  - `x,y` — векторы данных.

Выбор конкретной функции сплайновых коэффициентов влияет на интерполяцию вблизи конечных точек интервала. Пример сплайн-интерполяции приведен в листинге 13.2.

Листинг 13.2. Кубическая сплайн-интерполяция

```
x := (0  1  2  3  4  5  6)^T
y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T
s := cspline(x, y)
A(t) := interp(s, x, y, t)
```

Смысл сплайн-интерполяции заключается в том, что в промежутках между точками осуществляется аппроксимация в виде зависимости $A(t)=a\cdot t^3+b\cdot t^2+c\cdot t+d$. Коэффициенты a, b, c, d рассчитываются независимо для каждого промежутка, исходя из значений y_i в соседних точках. Этот процесс скрыт от пользователя, поскольку смысл задачи интерполяции состоит в выдаче значения A(t) в любой точке t (рис. 13.4).

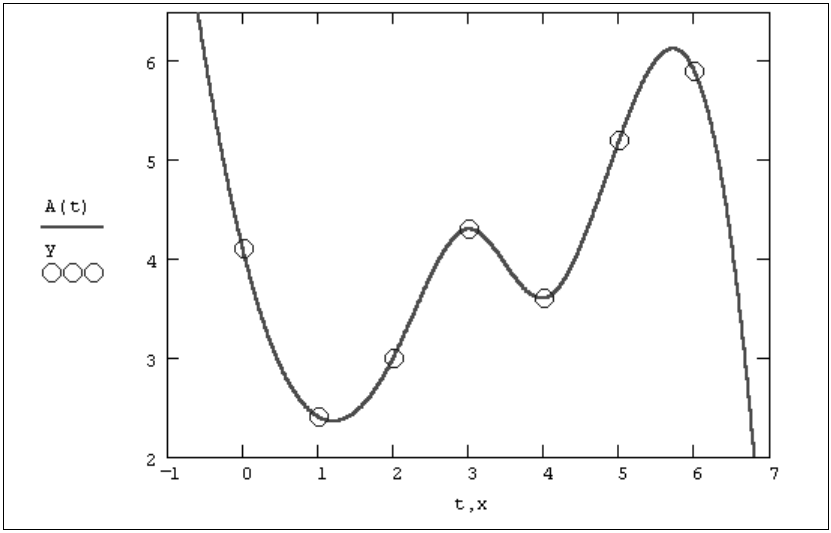


Рис. 13.4. Сплайн-интерполяция (продолжение листинга 13.2)

Чтобы подчеркнуть различия, соответствующие разным вспомогательным функциям cspline, pspline, lspline, покажем результат действия листинга 13.2 при замене функции cspline в предпоследней строке на линейную lspline (рис. 13.5). Как видно, выбор вспомогательных функций существенно влияет на поведение A(t) вблизи граничных точек рассматриваемого интервала (0,6) и особенно разительно меняет результат экстраполяции данных за его пределами.

В заключение остановимся на уже упоминавшейся в предыдущем разделе распространенной ошибке при построении графиков интерполирующей функции (см. рис. 13.3). Если на графике, например, являющемся продолжением листинга 13.2, задать построение функции A(x) вместо A(t), то будет получено просто соединение исходных точек ломаной (рис. 13.6). Так происходит потому, что в промежутках между точками вычисления интерполирующей функции не производятся.

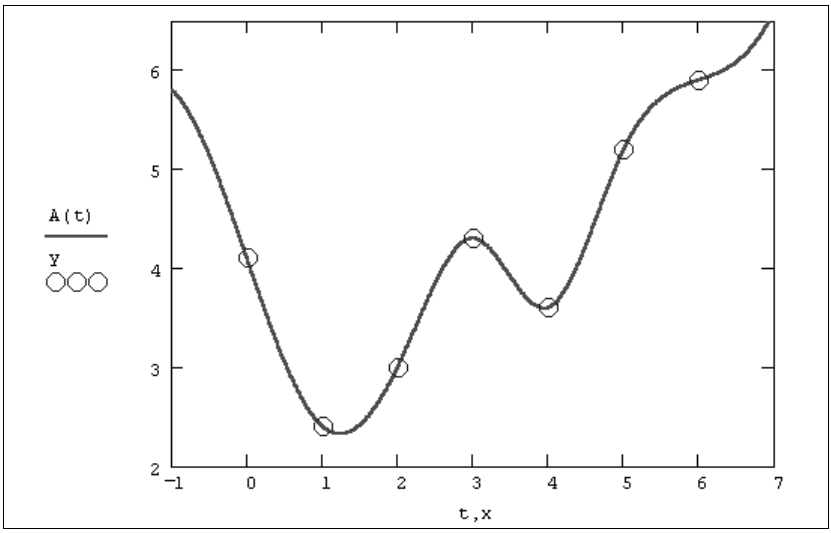


Рис. 13.5. Сплайн-интерполяция
с выбором коэффициентов линейного сплайна `lspline`

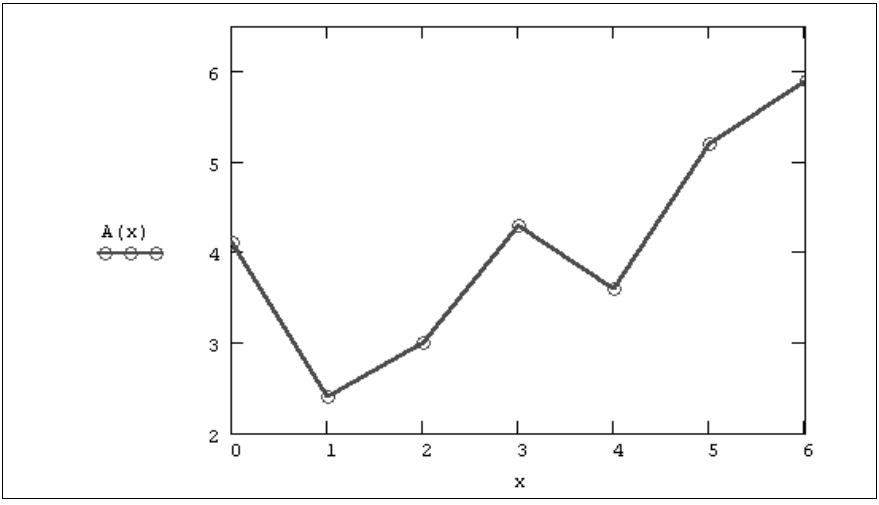


Рис. 13.6. Ошибочное построение графика сплайн-интерполяции
(продолжение листинга 13.2)

13.1.3. Полиномиальная сплайн-интерполяция

Более сложный тип интерполяции — так называемая *интерполяция B-сплайнами*. В отличие от обычной сплайн-интерполяции (*см. разд. 13.1.2*), сшивка элементарных B-сплайнов производится не в точках x_i, а в других точках u_i, координаты которых предлагается ввести пользователю. Сплайны могут быть полиномами `1`, `2` или `3` степени (линейные, квадратичные или кубические). Применяется интерполя-

ция B-сплайнами точно так же, как и обычная сплайн-интерполяция, различие состоит только в определении вспомогательной функции коэффициентов сплайна.

- `interp(s,x,y,t)` — функция, аппроксимирующая данные векторов `x` и `y` с помощью B-сплайнов.
- `bspline(x,y,u,n)` — вектор значений коэффициентов B-сплайна:
  - `s` — вектор вторых производных, созданный функцией `bspline`;
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - `y` — вектор действительных данных значений того же размера;
  - `t` — значение аргумента, при котором вычисляется интерполирующая функция;
  - `u` — вектор значений аргумента, в которых производится сшивка B-сплайнов;
  - `n` — порядок полиномов сплайновой интерполяции (`1`, `2` или `3`).

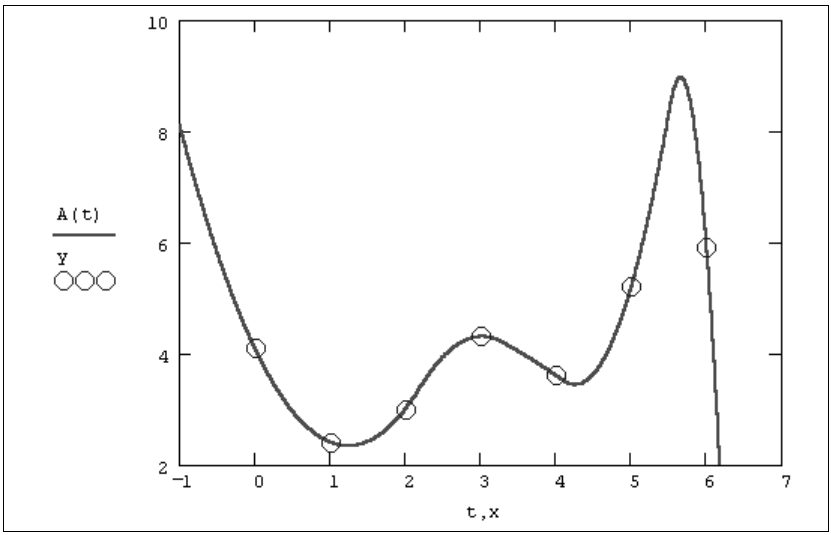


Рис. 13.7. B-сплайн-интерполяция (продолжение листинга 13.3)

Примечание

Размерность вектора `u` должна быть на `1`, `2` или `3` меньше размерности векторов `x` и `y`. Первый элемент вектора `u` должен быть меньше или равен первому элементу вектора `x`, а последний элемент `u` — больше или равен последнему элементу `x`.

Интерполяция B-сплайнами иллюстрируется листингом 13.3 и рис. 13.7.

Листинг 13.3. Интерполяция B-сплайнами

```
x:=(0  1  2  3  4  5  6)^T
y:=(4.1  2.4  3  4.3  3.6  5.2  5.9)^T
```

```
u := (−0.5  2.2  3.3  4.1  5.5  7)^T
s := bspline(x, y, u, 2)
A(t) := interp(s, x, y, t)
```

13.1.4. Сплайн-экстраполяция

Все описанные в предыдущих разделах типы интерполяции работают так же, как и функции экстраполяции данных. Для вычисления экстраполяции достаточно просто указать соответствующее значение аргумента, которое лежит за границами рассматриваемого интервала. С этой точки зрения разницы в применении в Mathcad между интерполяцией и экстраполяцией нет.

На практике при построении экстраполяции следует соблюдать известную осторожность, не забывая о том, что ее успех определяется значимостью ближайших к границе интервала точек. Чем дальше от них вы будете пытаться экстраполировать зависимость, заданную экспериментальными точками, тем сомнительнее будет результат. Сказанное иллюстрируется рис. 13.8 и 13.9, на которых изображена линейная (пунктир на обоих графиках) и сплайн- (сплошные кривые) экстраполяция. На рис. 13.8 используется линейная сплайн-экстраполяция при помощи функции `lspline` (см. рис. 13.5 в качестве примера интерполяции), а на рис. 13.9 — функции кубического сплайна `cspline` (что соответствует листингу 13.2 и рис. 13.4). Видно, что вдали от рассматриваемого интервала результаты экстраполяции совершенно различны, что, конечно, объясняется тем, что она является ничем иным, как параболической зависимостью.

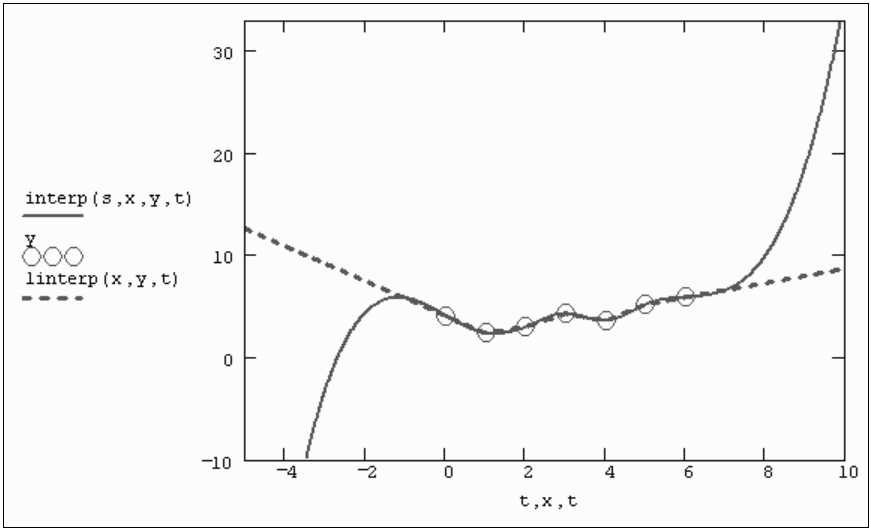


Рис. 13.8. Линейная сплайн-экстраполяция

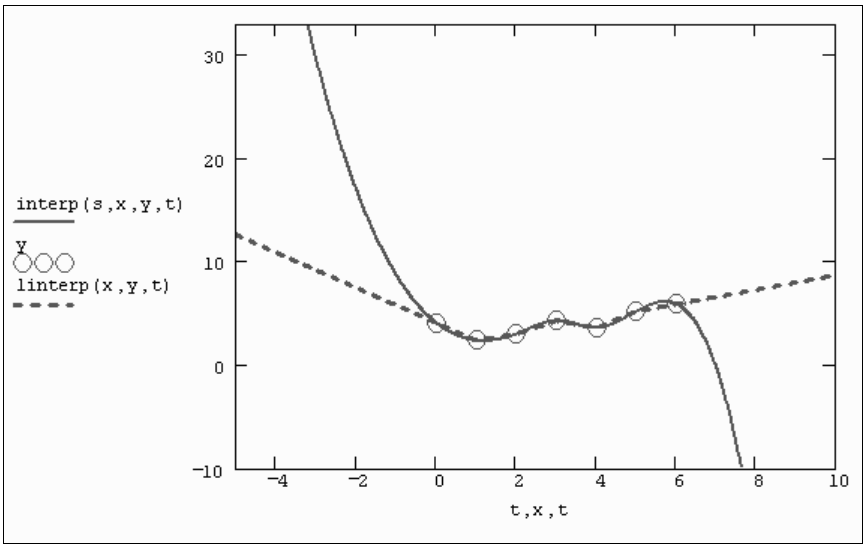


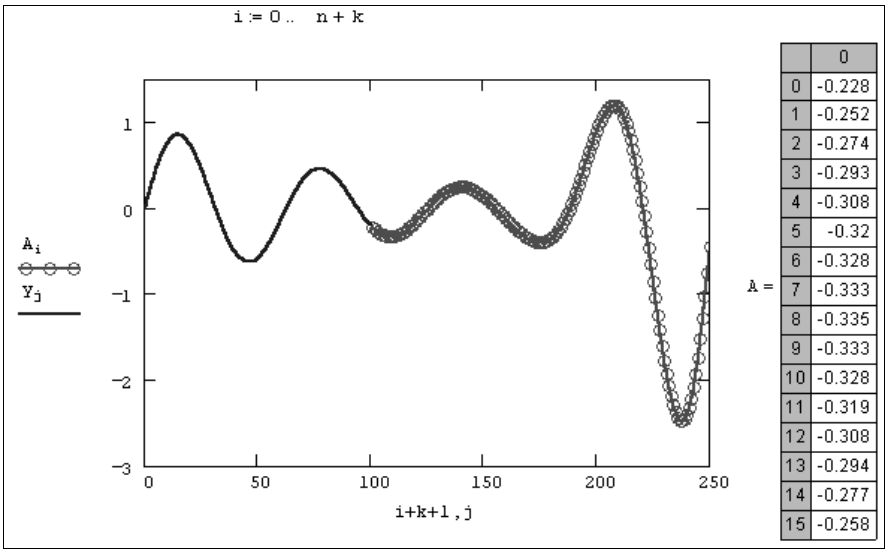
Рис. 13.9. Квадратичная сплайн-экстраполяция (продолжение листинга 13.2)

13.1.5. Экстраполяция функцией предсказания

Как мы увидели (*см. разд. 13.1.4*), стандартные функции интерполяции-экстраполяции стоит применять только в непосредственной близости границ интервала данных. В Mathcad имеется более развитый инструмент экстраполяции, который учитывает распределение данных вдоль всего интервала. В функцию `predict` встроен линейный алгоритм предсказания поведения функции, основанный на анализе, в том числе осцилляций:

- `predict(y,m,n)` — функция предсказания вектора, экстраполирующего выборку данных:
  - `y` — вектор действительных значений, взятых через равные промежутки значений аргумента;
  - `m` — количество последовательных элементов вектора `y`, согласно которым строится экстраполяция;
  - `n` — количество элементов вектора предсказаний.

Пример использования функции предсказания на примере экстраполяции осциллирующих данных y_j с меняющейся амплитудой приведен в листинге 13.4. Полученный график экстраполяции, наряду с самой функцией, показан на рис. 13.10. Аргументы и принцип действия функции `predict` отличаются от рассмотренных выше встроенных функций интерполяции-экстраполяции. Значений аргумента для данных не требуется, поскольку по определению функция действует на данные, идущие друг за другом с равномерным шагом. Обратите внимание, что результат функции `predict` вставляется "в хвост" исходных данных.



	0
0	-0.228
1	-0.252
2	-0.274
3	-0.293
4	-0.308
5	-0.32
6	-0.328
7	-0.333
8	-0.335
9	-0.333
10	-0.328
11	-0.319
12	-0.308
13	-0.294
14	-0.277
15	-0.258

Рис. 13.10. Экстраполяция при помощи функции предсказания (продолжение листинга 13.4)

Листинг 13.4. Экстраполяция при помощи функции предсказания

```
k := 100                          j := 0 ..  k
                  −j
                 ———
                 100
y_j := e          · sin( j/10 )
m := 50                           n := 150
A := predict (y, m, n)
```

Как видно из рис. 13.11, функция предсказания может быть полезна при экстраполяции данных на небольшие расстояния. Вдали от исходных данных результат часто бывает неудовлетворительным. Кроме того, функция `predict` хорошо работает в задачах анализа подробных данных с четко прослеживающейся закономерностью (типа рис. 13.10), в основном осциллирующего характера.

Если данных мало, то предсказание может оказаться бесполезным. В листинге 13.5 приведена экстраполяция небольшой выборки данных (из примеров, рассмотренных в предыдущих разделах). Соответствующий результат показан на рис. 13.11 для различных крайних точек массива исходных данных, для которых строится экстраполяция.

Листинг 13.5. Экстраполяция при помощи функции предсказания

```
y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T
k := rows (y) − 1
m := 5                            n := 10
A := predict (y, m, n)
```

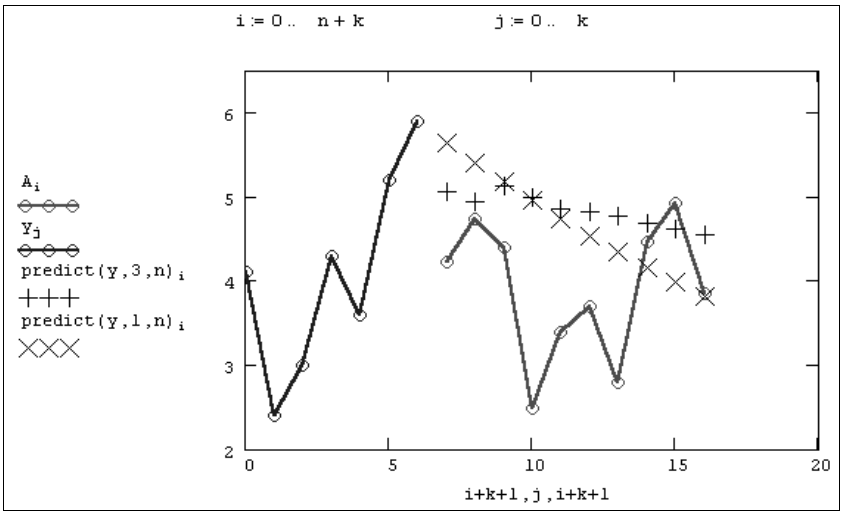


Рис. 13.11. Работа функции предсказания в случае малого количества данных (продолжение листинга 13.5)

13.1.6. Многомерная интерполяция

Двумерная сплайн-интерполяция приводит к построению поверхности `z(x,y)`, проходящей через массив точек, описывающий сетку на координатной плоскости `(x,y)`. Поверхность создается участками двумерных кубических сплайнов, являющихся функциями `(x,y)` и имеющих непрерывные первые и вторые производные по обеим координатам.

Многомерная интерполяция строится с помощью тех же встроенных функций, что и одномерная (*см. разд. 13.1.2*), но имеет в качестве аргументов не векторы, а соответствующие матрицы. Существует одно важное ограничение, связанное с возможностью интерполяции только квадратных `N×N` массивов данных:

- `interp(S,X,Z,V)` — скалярная функция, аппроксимирующая данные выборки двумерного поля по координатам `x` и `y` кубическими сплайнами:
  - `S` — вектор вторых производных, созданный одной из сопутствующих функций `cspline`, `pspline` или `lspline`;
  - `X` — матрица размерности `N×2`, определяющая диагональ сетки значений аргумента (элементы обоих столбцов соответствуют меткам `x` и `y` и расположены в порядке возрастания);
  - `Z` — матрица действительных данных размерности `N×N`;
  - `V` — вектор из двух элементов, содержащий значения аргументов `x` и `y`, для которых вычисляется интерполяция.

Примечание

Вспомогательные функции построения вторых производных имеют те же матричные аргументы, что и `interp`: `lspline(X,Y)`, `pspline(X,Y)`, `cspline(X,Y)`.

Пример исходных данных приведен на рис. 13.12 в виде графика линий уровня, программная реализация двумерной интерполяции показана в листинге 13.6, а ее результат — на рис. 13.13.

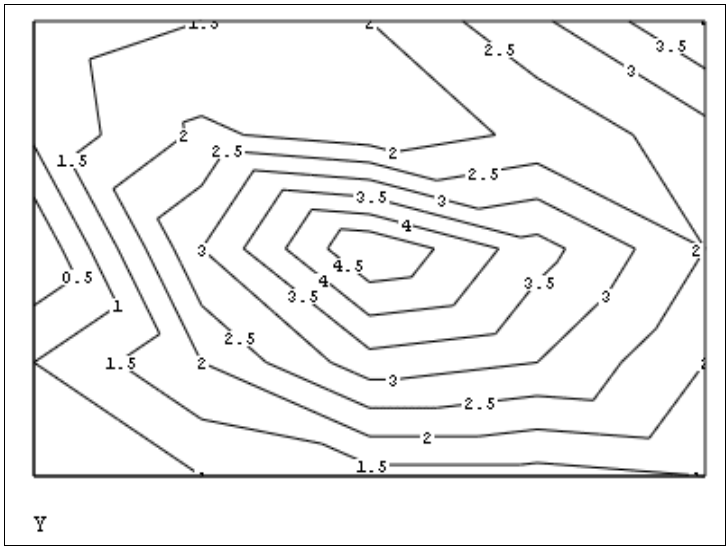


Рис. 13.12. Исходное двумерное поле данных
(продолжение листинга 13.6)

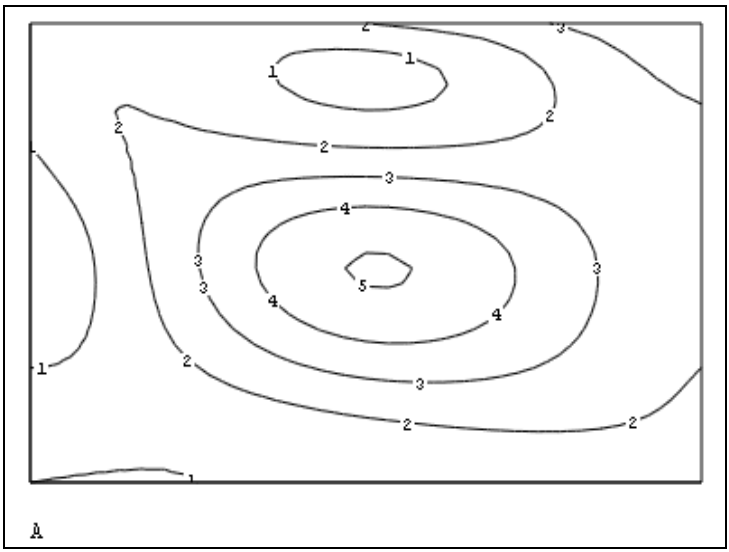


Рис. 13.13. Результат двумерной интерполяции
(продолжение листинга 13.6)

Листинг 13.6. Двумерная интерполяция

$$X := \begin{pmatrix} 0 & 0 \\ 1 & 10 \\ 2 & 20 \\ 3 & 30 \\ 4 & 40 \end{pmatrix} \qquad Y := \begin{pmatrix} 1 & 1 & 0 & 1.1 & 1.2 \\ 1 & 2 & 3 & 2.1 & 1.5 \\ 1.3 & 3.3 & 5 & 1.7 & 2 \\ 1.3 & 3 & 3.7 & 2.1 & 2.9 \\ 1.5 & 2 & 2.5 & 2.8 & 4 \end{pmatrix}$$

$$S := \mathrm{cspline}(X, Y)$$

$$V := \begin{pmatrix} 3.7 \\ 2.2 \end{pmatrix} \qquad \mathrm{interp}(S, X, Y, V) = 1.636$$

$$k := 30$$

$$i := 0 .. k \qquad j := 0 .. k$$

$$A_{i,j} := \mathrm{interp}\left[S, X, Y, \begin{pmatrix} \frac{i}{k} \cdot 4 \\ \frac{j}{k} \cdot 40 \end{pmatrix}\right]$$

13.2. Регрессия

Задачи математической регрессии имеют смысл приближения выборки данных (x_i, y_i) некоторой функцией $f(x)$, определенным образом минимизирующей совокупность ошибок $|f(x_i)-y_i|$. Регрессия сводится к подбору неизвестных коэффициентов, определяющих аналитическую зависимость $f(x)$. В силу производимого действия большинство задач регрессии являются частным случаем более общей проблемы сглаживания данных.

Как правило, регрессия очень эффективна, когда заранее известен (или, по крайней мере, хорошо угадывается) закон распределения данных (x_i, y_i).

13.2.1. Линейная регрессия

Самый простой и наиболее часто используемый вид регрессии — линейная. Приближение данных (x_i, y_i) осуществляется линейной функцией $y(x)=b+a\cdot x$. На координатной плоскости (x, y) линейная функция, как известно, представляется прямой линией (рис. 13.14). Еще линейную регрессию часто называют *методом наименьших квадратов*, поскольку коэффициенты a и b вычисляются из условия минимизации суммы квадратов ошибок $|b+a\cdot x_i-y_i|$.

Примечание 1

Чаще всего такое же условие ставится и в других задачах регрессии, т. е. приближения массива данных (x_i, y_i) другими зависимостями $y(x)$. Исключение рассмотрено в листинге 13.9.

Примечание 2

Различным расчетным аспектам реализации метода наименьших квадратов, в большинстве случаев сводящимся к решению систем алгебраических линейных уравнений, была посвящена значительная часть *главы 8*.

Для расчета линейной регрессии в Mathcad имеются два дублирующих друг друга способа. Правила их применения представлены в листингах 13.7 и 13.8. Результат обоих листингов получается одинаковым (рис. 13.14):

- `line(x,y)` — вектор из двух элементов `(b,a)` коэффициентов линейной регрессии `b+a·x`;
- `intercept(x,y)` — коэффициент `b` линейной регрессии;
- `slope(x,y)` — коэффициент `a` линейной регрессии:
  - `x` — вектор действительных данных аргумента;
  - `y` — вектор действительных данных значений того же размера.

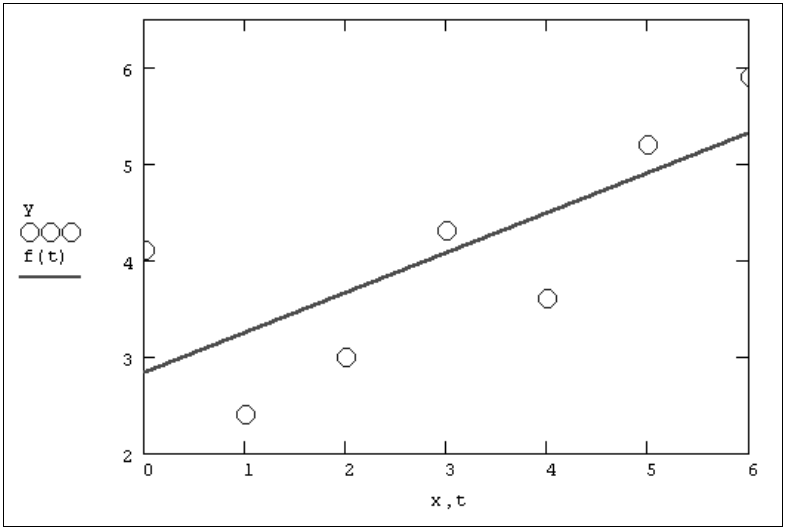


Рис. 13.14. Линейная регрессия (продолжение листинга 13.7 или 13.8)

Листинг 13.7. Линейная регрессия

$x := (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6)^T$

$y := (4.1\ \ 2.4\ \ 3\ \ 4.3\ \ 3.6\ \ 5.2\ \ 5.9)^T$

$\mathrm{line}(x,y) = \begin{pmatrix} 2.829 \\ 0.414 \end{pmatrix}$

$f(t) := \mathrm{line}(x,y)_0 + \mathrm{line}(x,y)_1 \cdot t$

Листинг 13.8. Другая форма записи линейной регрессии

```
x := (0  1  2  3  4  5  6)^T
y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T
intercept(x, y) = 2.829
slope(x, y) = 0.414
f(t) := intercept(x, y) + slope(x, y) · t
```

В Mathcad имеется альтернативный алгоритм, реализующий не минимизацию суммы квадратов ошибок, а медиан-медианную линейную регрессию для расчета коэффициентов `a` и `b` (листинг 13.9):

- ❒ `medfit(x,y)` — вектор из двух элементов `(b,a)` коэффициентов линейной медиан-медианной регрессии `b+a·x`:
  - `x,y` — векторы действительных данных одинакового размера.

Листинг 13.9. Построение линейной регрессии двумя разными методами (продолжение листинга 13.7)

$$\mathrm{medfit}(x, y) = \begin{pmatrix} 2.517 \\ 0.55 \end{pmatrix}$$

$$g(t) := \mathrm{medfit}(x, y)_0 + \mathrm{medfit}(x, y)_1 \cdot t$$

Различие результатов среднеквадратичной и медиан-медианной регрессии иллюстрируется на рис. 13.15.

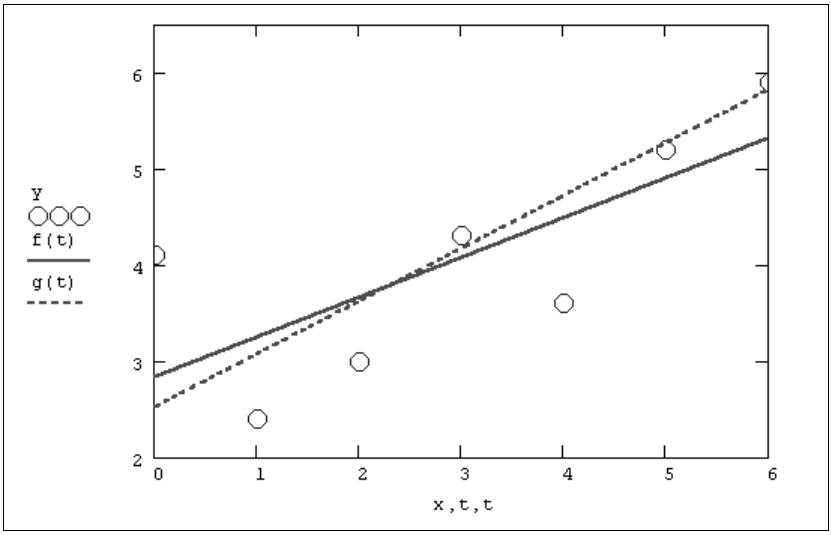


Рис. 13.15. Линейная регрессия по методу наименьших квадратов и методу медиан (продолжение листингов 13.7 и 13.9)

13.2.2. Полиномиальная регрессия

В Mathcad реализована регрессия одним полиномом, отрезками нескольких полиномов, а также двумерная регрессия массива данных.

Полиномиальная регрессия

Полиномиальная регрессия означает приближение данных (x_i, y_i) полиномом k-й степени $A(x)=a+b\cdot x+c\cdot x^2+d\cdot x^3+...+h\cdot x^k$ (рис. 13.16). При k=1 полином является прямой линией, при k=2 — параболой, при k=3 — кубической параболой и т. д. Как правило, на практике применяются k<5.

> ***Примечание***
>
> Для построения регрессии полиномом k-й степени необходимо наличие, по крайней мере, (k+1) точек данных.

В Mathcad полиномиальная регрессия осуществляется комбинацией встроенной функции regress и полиномиальной интерполяции (*см. разд. 13.1.2*):

- regress(x,y,k) — вектор коэффициентов для построения полиномиальной регрессии данных;
- interp(s,x,y,t) — результат полиномиальной регрессии:
  - s=regress(x,y,k);
  - x — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - y — вектор действительных данных значений того же размера;
  - k — степень полинома регрессии (целое положительное число);
  - t — значение аргумента полинома регрессии.

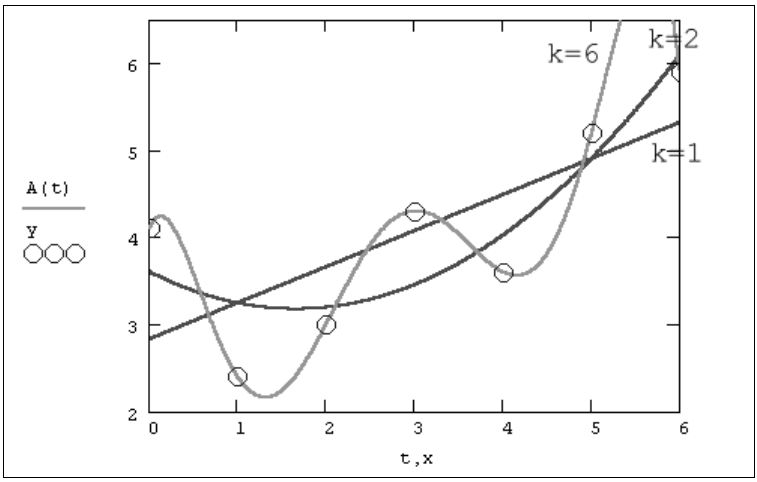


Рис. 13.16. Регрессия полиномами разной степени (коллаж результатов листинга 13.10 для разных k)

Внимание!

Для построения полиномиальной регрессии после функции `regress` вы обязаны использовать функцию `interp`.

Пример полиномиальной регрессии квадратичной параболой приведен в листинге 13.10.

Листинг 13.10. Полиномиальная регрессия

```
x := (0  1  2  3  4  5  6)^T
y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T
k := 2
s := regress(x, y, k)
A(t) := interp(s, x, y, t)
```

Регрессия отрезками полиномов

Помимо приближения массива данных одним полиномом имеется возможность осуществить регрессию сшивкой отрезков (точнее говоря, участков, т. к. они имеют криволинейную форму) нескольких полиномов. Для этого имеется встроенная функция `loess`, применение которой аналогично функции `regress` (листинг 13.11 и рис. 13.17):

- `loess(x,y,span)` — вектор коэффициентов для построения регрессии данных отрезками полиномов;
- `interp(s,x,y,t)` — результат полиномиальной регрессии:
  - `s=loess(x,y,span)`;
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания;
  - `y` — вектор действительных данных значений того же размера;
  - `span` — параметр, определяющий размер отрезков полиномов (положительное число, хорошие результаты дает значение порядка `span=0.75`).

Параметр `span` задает степень сглаженности данных. При больших значениях `span` регрессия практически не отличается от регрессии одним полиномом (например, `span=2` дает почти тот же результат, что и приближение точек параболой).

Листинг 13.11. Регрессия отрезками полиномов

```
x := (0  1  2  3  4  5  6)^T
y := (4.1  2.4  3  4.3  3.6  5.2  5.9)^T
s := loess(x, y, 0.9)
A(t) := interp(s, x, y, t)
```

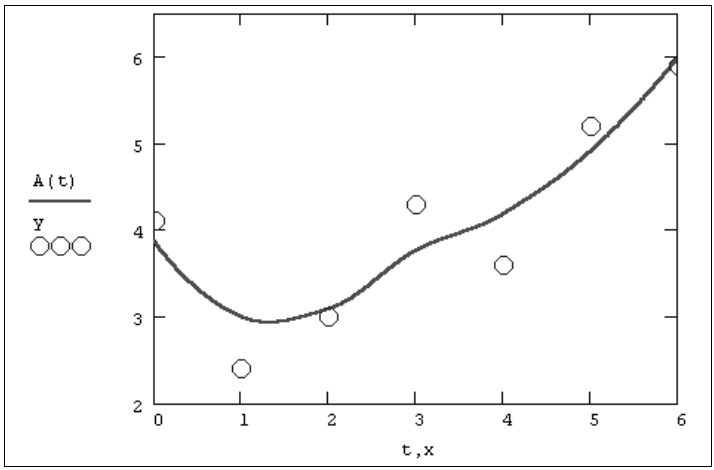


Рис. 13.17. Регрессия отрезками полиномов (продолжение листинга 13.11)

Совет

Регрессия одним полиномом эффективна, когда множество точек выглядит как полином, а регрессия отрезками полиномов оказывается полезной в противоположном случае.

Двумерная полиномиальная регрессия

По аналогии с одномерной полиномиальной регрессией и двумерной интерполяцией (*см. разд. 13.1.6*), Mathcad позволяет приблизить множество точек $z_{i,j}(x_i, y_j)$ поверхностью, которая определяется многомерной полиномиальной зависимостью. В качестве аргументов встроенных функций для построения полиномиальной регрессии должны стоять в этом случае не векторы, а соответствующие матрицы.

- `regress(X,Z,k)` — вектор коэффициентов для построения полиномиальной регрессии данных.
- `loess(X,Z,span)` — вектор коэффициентов для построения регрессии данных отрезками полиномов.
- `interp(S,X,Z,V)` — скалярная функция, аппроксимирующая данные выборки двумерного поля по координатам `x` и `y` кубическими сплайнами.
  - `S` — вектор вторых производных, созданный одной из сопутствующих функций `loess` или `regress`.
  - `X` — матрица размерности `N×2`, определяющая пары значений аргумента (столбцы соответствуют меткам `x` и `y`).
  - `Z` — вектор действительных данных размерности `N`.
  - `Span` — параметр, определяющий размер отрезков полиномов.
  - `k` — степень полинома регрессии (целое положительное число).
  - `V` — вектор из двух элементов, содержащий значения аргументов `x` и `y`, для которых вычисляется интерполяция.

Внимание!

Для построения регрессии не предполагается никакого предварительного упорядочивания данных (как, например, для двумерной интерполяции, которая требует их представления в виде матрицы N×N). В связи с этим данные представляются как вектор.

Двумерная полиномиальная регрессия иллюстрируется листингом 13.12 и рис. 13.18. Сравните стиль представления данных для двумерной регрессии с представлением тех же данных для двумерной сплайн-интерполяции (см. листинг 13.6) и ее результаты с исходными данными (см. рис. 13.12) и их сплайн-интерполяцией (см. рис. 13.13).

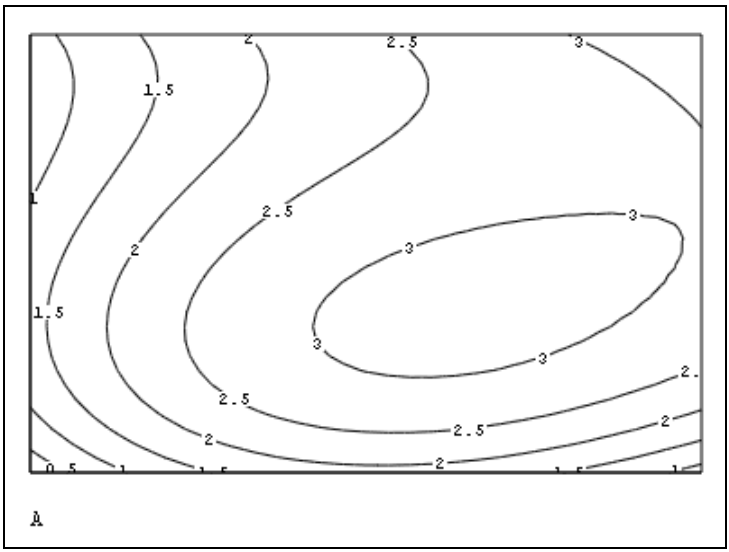


Рис. 13.18. Двумерная полиномиальная регрессия
(продолжение листинга 13.12)

Листинг 13.12. Двумерная полиномиальная регрессия

$$X := \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 4 & 4 \\ 0 & 10 & 20 & 30 & 40 & 0 & 10 & 20 & 30 & 40 & 0 & 10 & 20 & 30 & 40 & 0 & 10 & 20 & 30 & 40 & 0 & 10 & 20 & 30 & 40 \end{pmatrix}^T$$

$$Z := (1\ \ 1\ \ 0\ \ 1.1\ \ 1.2\ \ 1\ \ 2\ \ 3\ \ 2.1\ \ 1.5\ \ 1.3\ \ 3.3\ \ 5\ \ 1.7\ \ 2\ \ 1.3\ \ 3\ \ 3.7\ \ 2.1\ \ 2.9\ \ 1.5\ \ 2\ \ 2.5\ \ 2.8\ \ 4)^T$$

$$S := \mathrm{regress}(X, Z, 3)$$

$$k := 30$$

$$i := 0 .. k \qquad j := 0 .. k$$

$$A_{i,j} := \mathrm{interp}\left[S, X, Z, \begin{pmatrix} \frac{i}{k} \cdot 4 \\ \frac{j}{k} \cdot 40 \end{pmatrix}\right]$$

Примечание

Обратите внимание на знаки транспонирования в листинге. Они применены для корректного представления аргументов (например, Z, в качестве вектора, а не строки).

13.2.3. Другие типы регрессии

Кроме рассмотренных, в Mathcad встроено еще несколько видов трехпараметрической регрессии. Их реализация несколько отличается от приведенных выше вариантов регрессии тем, что для них, помимо массива данных, требуется задать некоторые начальные значения коэффициентов a, b, c. Используйте соответствующий вид регрессии, если хорошо представляете себе, какой зависимостью описывается ваш массив данных. Когда тип регрессии плохо отражает последовательность данных, то ее результат часто бывает неудовлетворительным и даже сильно различающимся в зависимости от выбора начальных значений. Каждая из функций выдает вектор уточненных параметров a, b, c.

- expfit(x,y,g) — регрессия экспонентой $f(x)=a\cdot e^{bx}+c$.
- lgsfit(x,y,g) — регрессия логистической функцией $f(x)=a/(1+b\cdot e^{-cx})$.
- sinfit(x,y,g) — регрессия синусоидой $f(x)=a\cdot\sin(x+b)+c$.
- pwfit(x,y,g) — регрессия степенной функцией $f(x)=a\cdot x^{b}+c$.
- logfit(x,y,g) — регрессия логарифмической функцией $f(x)=a\cdot\ln(x+b)+c$.
- lnfit(x,y) — регрессия двухпараметрической логарифмической функцией $f(x)=a\cdot\ln(x)+b$.
  - x — вектор действительных данных аргумента.
  - y — вектор действительных значений того же размера.
  - g — вектор из трех элементов, задающий начальные значения a, b, c.

Примечание

Правильность выбора начальных значений можно оценить по результату регрессии — если функция, выданная Mathcad, хорошо приближает зависимость y(x), значит, они были подобраны удачно.

Пример расчета одного из видов трехпараметрической регрессии (экспоненциальной) приведен в листинге 13.13. В предпоследней строке листинга выведены в виде вектора вычисленные коэффициенты a, b, c, а в последней строке через эти коэффициенты определена искомая функция f(x).

Листинг 13.13. Экспоненциальная регрессия

$$x := (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6)^T$$

$$y := (4.1\ \ 2.4\ \ 3\ \ 4.3\ \ 3.6\ \ 5.2\ \ 5.9)^T$$

$$g := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$C := \mathrm{expfit}(x, y, g)$

$$C = \begin{pmatrix} 0.111 \\ 0.544 \\ 3.099 \end{pmatrix}$$

$f(t) := C_0 \cdot \exp\left(C_1 \cdot t\right) + C_2$

Примечание

Многие задачи регрессии данных различными двухпараметрическими зависимостями `y(x)` можно свести к более надежной, с вычислительной точки зрения, линейной регрессии. Делается это с помощью соответствующей замены переменных.

13.2.4. Регрессия общего вида

В Mathcad можно осуществить регрессию в виде линейной комбинации $C_1f_1(x)+C_2f_2(x)+...$, где $f_i(x)$ — любые функции пользователя, а C_i — подлежащие определению коэффициенты. Кроме того, имеется путь проведения регрессии более общего вида, когда комбинацию функций и искомых коэффициентов задает сам пользователь.

Приведем встроенные функции для регрессии общего вида и примеры их использования (листинги 13.14 и 13.15), надеясь, что читатель при необходимости найдет более подробную информацию об этих специальных возможностях в справочной системе Mathcad.

- `linfit(x,y,F)` — вектор параметров линейной комбинации функций пользователя, осуществляющей регрессию данных.
- `genfit(x,y,G,g)` — вектор параметров, реализующих регрессию данных с помощью функций пользователя общего вида.
  - `x` — вектор действительных данных аргумента, элементы которого расположены в порядке возрастания.
  - `y` — вектор действительных значений того же размера.
  - `F(x)` — пользовательская векторная функция скалярного аргумента.
  - `G` — вектор начальных значений параметров регрессии размерности `N`.
  - `g(x,C)` — функция пользователя, на основе которой строится регрессия. Если вы пользуетесь Mathcad ранних версий (вплоть до 12-й включительно), то вид `g(x,C)` немного более сложный.

Примечание

Число данных (количество элементов в векторах `x` и `y`) должно быть не меньше, чем `N`. Это менее жесткое требование появилось в Mathcad 12, а до этого для функции регрессии общего вида `genfit` было необходимо задание не менее `N+1` данных.

Листинг 13.14. Регрессия линейной комбинацией функций пользователя

$x := (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6)^T$

$y := (4.1\ \ 2.4\ \ 3\ \ 4.3\ \ 3.6\ \ 5.2\ \ 5.9)^T$

$$F(x) := \begin{pmatrix} \frac{1}{x+1} \\ x \\ e^x \end{pmatrix}$$

$C := \mathrm{linfit}(x, y, F)$

$$C = \begin{pmatrix} 3.957 \\ 0.854 \\ 5.605 \times 10^{-4} \end{pmatrix}$$

$$f(x) := \frac{C_0}{x+1} + C_1 \cdot x + C_2 \cdot e^x$$

⑬ Листинг 13.15. Регрессия общего вида

$x := (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6)^T$

$y := (4.1\ \ 2.4\ \ 3\ \ 4.3\ \ 3.6\ \ 5.2\ \ 5.9)^T$

$f(x, C) := C_0 \cdot e^{C_1 \cdot x}$

$$C0 := \begin{pmatrix} .1 \\ .2 \end{pmatrix}$$

$C := \mathrm{genfit}(x, y, C0, f)$

$$C = \begin{pmatrix} 2.817 \\ 0.113 \end{pmatrix}$$

⑮ 13.2.5. Многопараметрическая полиномиальная регрессия

В Mathcad 15 появилась серия новых функций, реализующих технологию *планирования эксперимента* (DOE — *Design of Experiment*), причем несколько из них связано с проведением многопараметрической регрессии. Например, можно осуществить двухпараметрическую полиномиальную регрессию вида $C_{10}x+C_{11}xy+ C_{21}x^2y\ldots$, где x и y — аргументы, а C_{ij} — коэффициенты регрессии.

- `polyfit(X,Z,n,"terms",M)` — определяет функцию векторного аргумента, описывающую поверхность многомерной полиномиальной регрессии, которая подгоняет результаты, записанные в матрице `Z`, к данным матрицы `X`.
  - `X` — матрица аргументов, в которой каждый столбец представляет независимую переменную;

- `Z` — вектор или матрица с результатами измерений;
- `n` — целое число, задающее степень полинома (оно должно быть меньше, чем общее число точек данных);
- `"terms"` — строка, содержащая члены или факторы и взаимодействия, которые требуется включить в полиномиальную регрессию (например, строка `"A B AB AA BB"` определяет полином вида $c_0+c_1\cdot A+c_2\cdot B+c_3\cdot A\cdot B+c_4\cdot A^2+c_5\cdot B^2$);
- `M` — матрица, задающая полином, с начальными значениями коэффициентов в первом столбце и показателями степени независимых переменных для каждого члена в остальных столбцах.

Приведем простой пример выполнения линейной регрессии двухпараметрических данных (листинг 13.16). Аргумент (т. е. экспериментальные точки) определен в первой строке листинга. Вторая строка переписывает аргумент в вид, необходимый для применения функции `polyfit`. В трех следующих строках при помощи метода Монте-Карло генерируется и выводится вектор `Z` с результатами "измерений". В двух последних строках листинга применяется встроенная функция `polyfit` и рассчитывается ее значение для точки `(x=1,y=1)`.

В графической форме результат двухпараметрической линейной регрессии демонстрирует рис. 13.19, на котором построены графики поверхности по рассчитанной регрессии (слева) и по исходной функции `f(x,y)`, использованной в методе Монте-Карло (справа).

Листинг 13.16. Многопараметрическая полиномиальная регрессия

$$X := \begin{pmatrix} 1 \\ 2 \\ 4 \\ 7 \\ 12 \\ 15 \\ 16 \\ 20 \end{pmatrix} \qquad Y := \begin{pmatrix} 1 \\ 2 \\ 5 \\ 3 \\ 7 \\ 9 \\ 11 \\ 4 \end{pmatrix}$$

$$XY := \mathrm{augment}(X, Y)$$

$$f(x, y) := 1.7\cdot x + 2.2\cdot y$$

$$Z := f(X, Y) + \mathrm{rnd}(1)$$

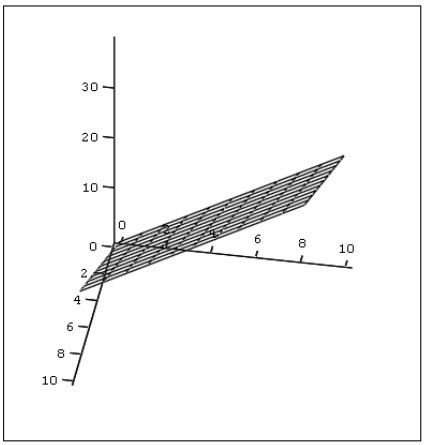
$$XY = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 4 & 5 \\ 7 & 3 \\ 12 & 7 \\ 15 & 9 \\ 16 & 11 \\ 20 & 4 \end{pmatrix} \qquad Z = \begin{pmatrix} 4.889 \\ 8.789 \\ 18.789 \\ 19.489 \\ 36.789 \\ 46.289 \\ 52.389 \\ 43.789 \end{pmatrix}$$

```
p := polyfit(XY, Z, 1)
```

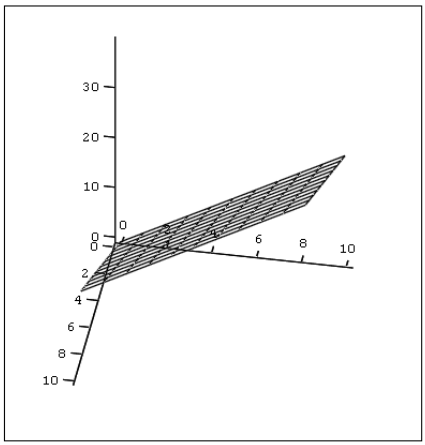
$$P\left(\begin{pmatrix}1\\1\end{pmatrix}\right) = 4.889$$

$$i := 0 .. 10 \qquad j := 0 .. 10$$

$$R_{i,j} := p\left(\begin{pmatrix}i\\j\end{pmatrix}\right) \qquad S_{i,j} := f(i,j)$$



R



S

Рис. 13.19. Результат двухпараметрической линейной регрессии (продолжение листинга 13.16)

13.3. Фильтрация

При анализе данных часто возникает задача их фильтрации, заключающаяся в устранении одной из составляющих зависимости $y(x_i)$. Наиболее часто целью фильтрации является подавление быстрых вариаций $y(x_i)$, которые чаще всего обусловлены шумом. В результате из быстроосциллирующей зависимости $y(x_i)$ получается другая, сглаженная зависимость, в которой доминирует более низкочастотная составляющая. В связи с этим считают, что сглаживание является частным случаем более общей задачи фильтрации сигнала, которая связана с подавлением определенных диапазонов частот спектра.

Наиболее простым и эффективным рецептом сглаживания (smoothing) можно считать регрессию различного вида (*см. разд. 13.2*). Однако регрессия часто уничтожает информативную составляющую данных, оставляя лишь наперед заданную пользователем зависимость.

Часто рассматривают противоположную задачу фильтрации — устранение медленно меняющихся вариаций в целях исследования высокочастотной составляющей. В этом случае говорят о задаче устранения тренда. Иногда интерес представляют смешанные задачи выделения среднемасштабных вариаций путем подавления как более быстрых, так и более медленных вариаций. Одна из возможностей решения связана с применением полосовой фильтрации.

Несколько примеров программной реализации различных вариантов фильтрации приведены в данном разделе.

13.3.1. Встроенные функции для сглаживания: ВЧ-фильтр

В Mathcad имеется несколько встроенных функций, реализующих различные алгоритмы сглаживания данных:

- `medsmooth(y,b)` — сглаживание алгоритмом "бегущих медиан";
- `ksmooth(x,y,b)` — сглаживание на основе функции Гаусса;
- `supsmooth(x,y)` — локальное сглаживание адаптивным алгоритмом, основанное на анализе ближайших соседей каждой пары данных:
  - `x` — вектор действительных данных аргумента (для `supsmooth` его элементы должны быть расположены в порядке возрастания);
  - `y` — вектор действительных значений того же размера, что и `x`;
  - `b` — ширина окна сглаживания.

Все функции имеют в качестве аргумента векторы, составленные из массива данных, и выдают в качестве результата вектор сглаженных данных того же размера. Функция `medsmooth` предполагает, что данные расположены равномерно. Сглаживание данных при помощи "бегущих медиан" с разным значением ширины окна пропускания показано на рис. 13.20.

Часто бывает полезным совместить сглаживание с последующей интерполяцией или регрессией. Соответствующий пример приведен в листинге 13.17 для функции `supsmooth`.

Листинг 13.17. Сглаживание с последующей сплайн-интерполяцией

```
x := (0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16)^T
y := (4.1  2.4  3  4.3  3.6  5.2  5.9  5  4.7  4  3.5  3.9  3  2.7  3.7  4.8  5.4)^T
z := supsmooth(x, y)
s := cspline(x, z)
A(t) := interp(s, x, z, t)
```

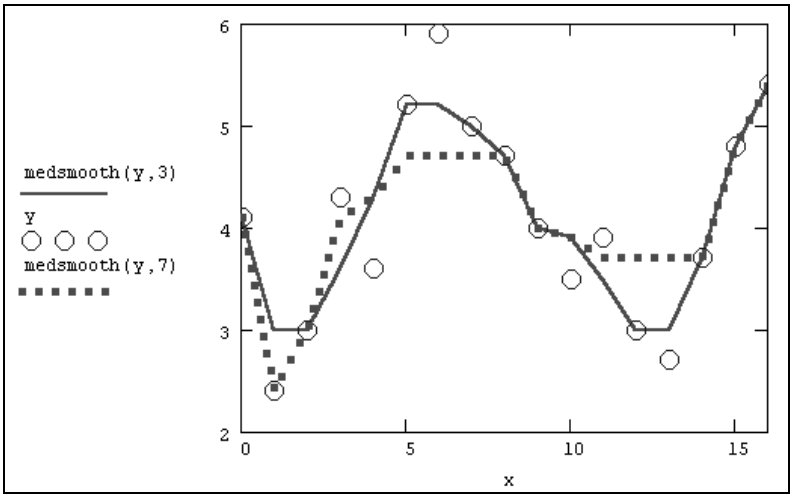


Рис. 13.20. Сглаживание "бегущими медианами"

13.3.2. Скользящее усреднение: ВЧ-фильтр

Помимо встроенных в Mathcad, существует несколько популярных алгоритмов сглаживания, на одном из которых хочется остановиться особо.

Самый простой и очень эффективный метод — это скользящее усреднение. Его суть состоит в расчете для каждого значения аргумента среднего значения по соседним `w` данным. Число `w` называют *окном* скользящего усреднения; чем оно больше, тем больше данных участвуют в расчете среднего, тем более сглаженная кривая получается. На рис. 13.21 показан результат скользящего усреднения одних и тех же данных (кружки) с разным окном `w=3` (пунктир), `w=5` (штрихованная кривая) и `w=15` (сплошная кривая). Видно, что при малых `w` сглаженные кривые практически повторяют ход изменения данных, а при больших `w` — отражают лишь закономерность их медленных вариаций.

Чтобы реализовать в Mathcad скользящее усреднение, достаточно очень простой программы, приведенной в листинге 13.8. Она использует только значения `y`, оформленные в виде вектора, неявно предполагая, что они соответствуют значениям аргумента `x`, расположенным через одинаковые промежутки. Вектор `x` применялся лишь для построения графика результата (рис. 13.21).

Листинг 13.18. Сглаживание скользящим усреднением

```
x := (0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16)^T

y := (4.1  2.4  3  4.3  3.6  5.2  5.9  5  4.7  4  3.5  3.9  3  2.7  3.7  4.8  5.4)^T
w := 15
N := rows(y)                          N = 17
i := 0 .. N − 1
```

$$m_i := \text{if}\left(i < w, \frac{\sum_{j=0}^{i} y_j}{i+1}, \frac{\sum_{j=i-w+1}^{i} y_j}{w}\right)$$

Примечание

Приведенная программная реализация скользящего усреднения самая простая, но не самая лучшая. Возможно, вы обратили внимание, что все кривые скользящего среднего на рис. 13.21 слегка "обгоняют" исходные данные. Почему так происходит, понятно: согласно алгоритму, заложенному в последнюю строку листинга 13.18, скользящее среднее для каждой точки вычисляется путем усреднения значений предыдущих `w` точек. Чтобы результат скользящего усреднения был более адекватным, лучше применить центрированный алгоритм расчета по `w/2` предыдущим и `w/2` последующим значениям. Он будет немного сложнее, поскольку придется учитывать недостаток точек не только в начале (как это сделано в программе с помощью функции условия `if`), но и в конце массива исходных данных.

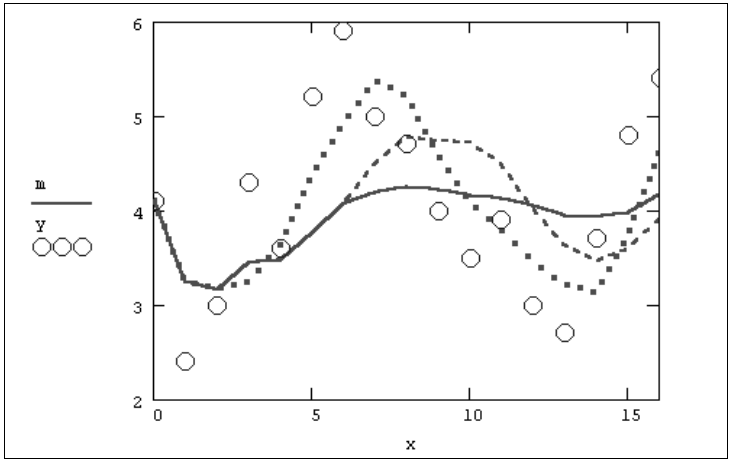


Рис. 13.21. Скользящее усреднение с разными `w=3,5,15` (листинг 13.18, коллаж трех графиков)

13.3.3. Устранение тренда: НЧ-фильтр

Еще одна типичная задача возникает, когда интерес исследований заключается не в анализе медленных (или *низкочастотных*) вариаций сигнала `y(x)` (для чего применяется сглаживание данных), а в анализе быстрых его изменений. Часто бывает, что быстрые (или *высокочастотные*) вариации накладываются определенным образом на медленные, которые обычно называют *трендом*. Часто тренд имеет заранее предсказуемый вид, например линейный. Чтобы устранить тренд, можно предложить последовательность действий, реализованную в листинге 13.19.

1. Вычислить регрессию `f(x)`, например линейную, исходя из априорной информации о тренде (предпоследняя строка листинга).
2. Вычесть из данных `y(x)` тренд `f(x)` (последняя строка листинга).

Листинг 13.19. Устранение тренда

$x := (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16)^T$

$y := (5.1\ \ 4.4\ \ 5.7\ \ 5.3\ \ 5.6\ \ 5.2\ \ 5.9\ \ 7.7\ \ 6.7\ \ 7\ \ 6.5\ \ 5.9\ \ 8.1\ \ 8.7\ \ 9.7\ \ 9.8\ \ 10.4)^T$

$N := \mathrm{rows}(y) \qquad N = 17$

$i := 0 .. N - 1$

$f(t) := \mathrm{line}(x, y)_0 + \mathrm{line}(x, y)_1 \cdot t$

$m_i := y_i - f(x_i)$

На рис. 13.22 показаны исходные данные (кружками), выделенный с помощью регрессии линейный тренд (сплошной прямой линией) и результат устранения тренда (пунктир, соединяющий крестики).

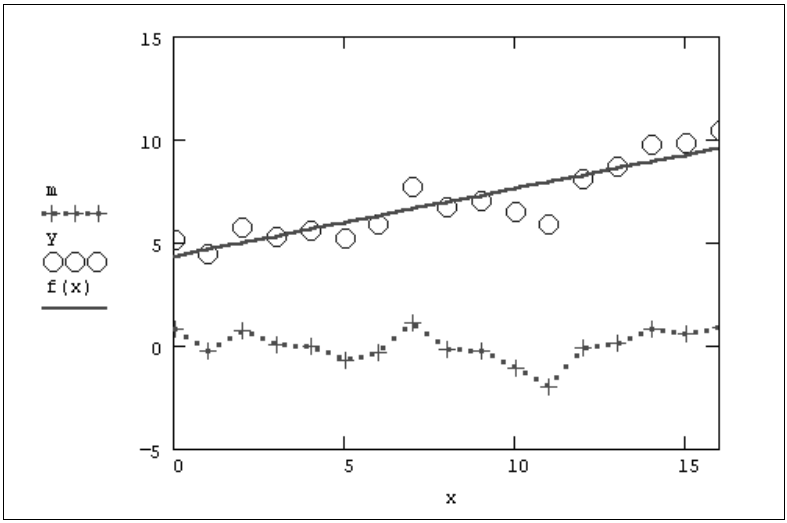


Рис. 13.22. Устранение тренда (продолжение листинга 13.19)

13.3.4. Полосовая фильтрация

В предыдущих разделах была рассмотрена фильтрация быстрых вариаций сигнала (сглаживание) и его медленных вариаций (снятие тренда). Иногда требуется выделить среднемасштабную составляющую сигнала, уменьшив как более быстрые, так и более медленные его компоненты. Одна из возможностей решения этой задачи связана с применением полосовой фильтрации на основе последовательного скользящего усреднения.

Алгоритм полосовой фильтрации приведен в листинге 13.20, а результат его применения показан на рис. 13.23 сплошной кривой.

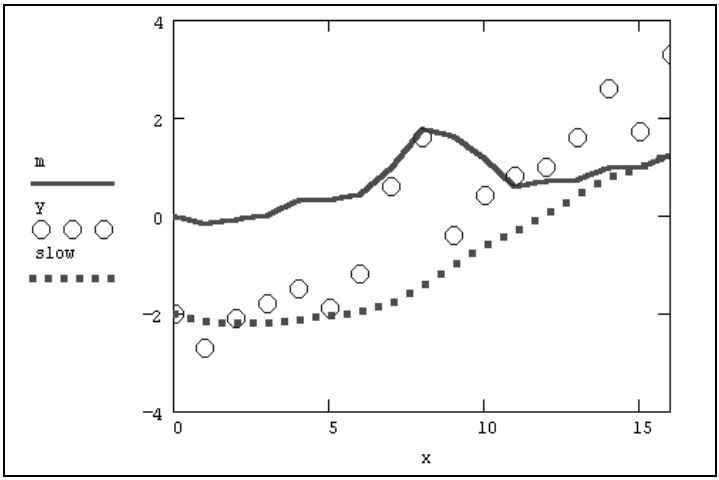


Рис. 13.23. Результат полосовой фильтрации (продолжение листинга 13.20)

Алгоритм реализует такую последовательность операций:

1. Выставление ноль-линии, т. е. приведение массива данных `y` к нулевому среднему значению путем его вычитания из каждого элемента `y` (третья и четвертая строки листинга).
2. Устранение из сигнала `y` высокочастотной составляющей, имеющее целью получить сглаженный сигнал `middle`, например, с помощью скользящего усреднения с малым окном `w` (в листинге 13.20 `w=3`).
3. Выделение из сигнала `middle` низкочастотной составляющей `slow`, например, путем скользящего усреднения с большим окном `w` (в листинге 13.19 `w=7`) либо с помощью снятия тренда (*см. разд. 13.3.3*).
4. Вычесть из сигнала `middle` тренд `slow` (последняя строка листинга), тем самым выделяя среднемасштабную составляющую исходного сигнала `y`.

Листинг 13.20. Полосовая фильтрация

$$x := (0\ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15\ \ 16)^T$$

$$y := (5.1\ \ 4.4\ \ 5\ \ 5.3\ \ 5.6\ \ 5.2\ \ 5.9\ \ 7.7\ \ 8.7\ \ 6.7\ \ 7.5\ \ 7.9\ \ 8.1\ \ 8.7\ \ 9.7\ \ 8.8\ \ 10.4)^T$$

$$meanY := mean(y)$$

$$y := y - meanY$$

$$N := rows(y) \qquad N = 17$$

$$i := 0 .. N - 1$$

$$w := 3$$

$$middle_i := if\left(i < w, \frac{\sum_{j=0}^{i} y_j}{i+1}, \frac{\sum_{j=i-w+1}^{i} y_j}{w}\right)$$

$w := 7$

$$slow_i := if\left(i < w, \frac{\sum_{j=0}^{i} middle_j}{i+1}, \frac{\sum_{j=i-w+1}^{i} middle_j}{w}\right)$$

$m := middle - slow$

13.3.5. Спектральная фильтрация

Альтернативой всем представленным до сих пор алгоритмам, в частности методу полосовой фильтрации (*см. предыдущий раздел*), является фильтрация на основе интеграла Фурье. Пока мы использовали для подавления в сигнале тех или иных частотных диапазонов определенные процедуры, основанные на арифметических преобразованиях. Между тем, для той же цели (правда, с большими вычислительными затратами) можно применять методы Фурье-анализа. Действительно, если вычислить спектр сигнала, удалить из него (или существенно уменьшить) определенные частоты, а затем выполнить обратное преобразование Фурье, то результатом будет фильтрованный сигнал.

Пример фильтрации на основе преобразования Фурье приведен в листинге 13.21. В качестве модельного сигнала использовалась смесь двух гармонических сигналов и равномерно распределенного шума (рис. 13.24).

Фурье-спектр данных z, вычисленный при помощи встроенной функции fft, показан на рис. 13.25. Листинг 13.21 отличается от обычного вычисления Фурье-спектра (*см. листинг 4.14 в разд. 4.4.3*), главным образом, последними двумя строками, в которых, собственно, и определяется явный вид спектрального фильтра $W(\Omega)$, или, по-другому, *спектральное окно*. Обратное Фурье-преобразование спектра произведения спектрального окна $W(\Omega)$ и Фурье-спектра сигнала z, представляющее собой результат фильтрации, показано на рис. 13.24 сплошной кривой.

Листинг 13.21. Фильтрация на основе преобразования Фурье

$N := 128$ $\qquad$ $xMAX := 100$

$$\Delta := \frac{xMAX}{N}$$

$i := 0 .. N - 1$

$x_i := i \cdot \Delta$

$y_i := \sin(2 \cdot \pi \cdot 0.05 \cdot x_i) + 0.5 \cdot \sin(2 \cdot \pi \cdot 0.5 \cdot x_i) + (rnd(1) - .5)$

$z := fft(y)$

$$\Omega 0 := \frac{1}{xMAX} \qquad \Omega N := \frac{N}{2 \cdot xMAX}$$

$$i := 0 .. \frac{N}{2} \qquad \Omega D_i := (i+1) \cdot \Omega 0$$

$$W(\Omega) := \exp\left(-\frac{\Omega^2}{0.1}\right)$$

$$b_i := z_i \cdot W\left|\Omega D_i\right|$$

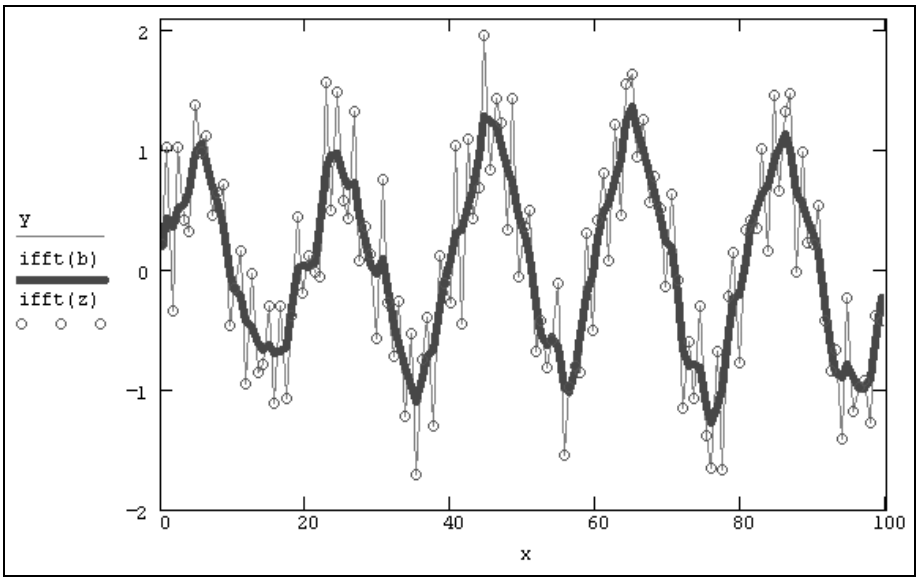


Рис. 13.24. Исходный модельный сигнал (кружки) и результат его фильтрации на основе Фурье-преобразования (продолжение листинга 13.11)

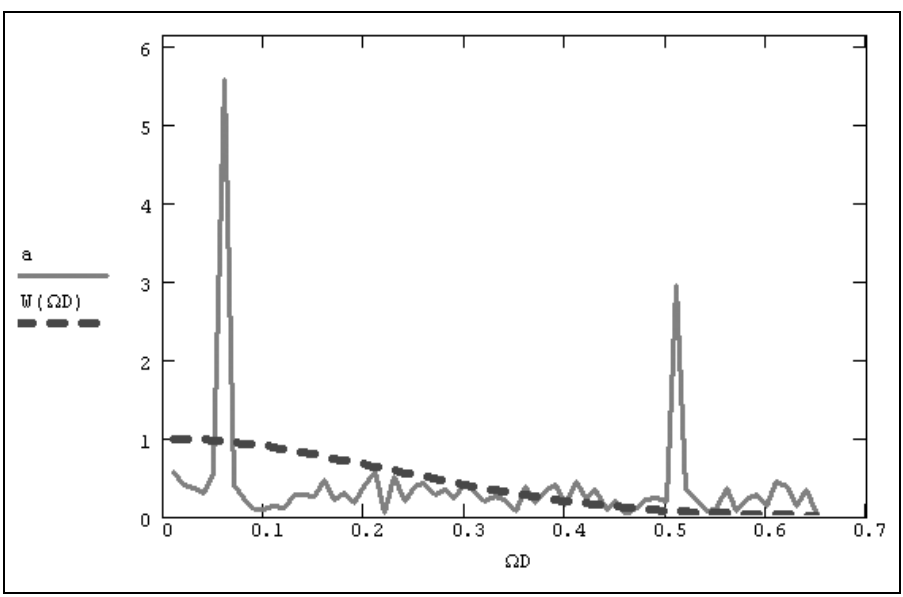


Рис. 13.25. Фурье-преобразование модельного сигнала и спектральное окно-фильтр (продолжение листинга 13.11)

ГЛАВА 14

Знакомство с Mathcad Prime

Последняя глава посвящена новому изданию Mathcad — приложению Mathcad Prime. В ней представлен интерфейс (*см. разд. 14.1*), вопросы конвертации документов прежних версий в документы Mathcad Prime (*см. разд. 14.2)* и несколько примеров решения типовых задач (*см. разд. 14.3*). Мы приведем лишь некоторые характерные примеры использования Mathcad Prime, не слишком углубляясь в его описание, поскольку данная версия русифицирована и снабжена очень качественной и подробной русскоязычной справочной системой. Эта глава будет полезной тем пользователям традиционных версий Mathcad, которые пока только раздумывают о миграции на Mathcad Prime и нуждаются в его оценке, не спеша установить эту версию на свой компьютер.

14.1. Интерфейс Mathcad Prime

Приложение Mathcad Prime сочетает традиционный функционал предыдущих версий Mathcad и современную компоновку интерфейса, основанную на использовании *ленты* (ribbon) как в Microsoft Office 2007 и 2010 (рис. 14.1). Однако в настоящее время не все возможности "обычной" версии Mathcad 15 реализованы в Mathcad Prime 1.0, поэтому компания PTC включает в комплект поставки одновременно и Mathcad 15, и Mathcad Prime 1.0.

Лента, как и в других современных приложениях, состоит из вкладок (интерфейсных групп), на которых сгруппированы соответствующие команды. На вкладках Ленты отображаются кнопки, предназначенные для выполнения тех или иных команд, причем многие из кнопок собраны в контекстно-зависимые меню.

В Mathcad Prime имеются следующие вкладки:

- **Математические** (Math);
- **Ввод/вывод** (Input/Output);
- **Функции** (Functions);
- **Матрицы/таблицы** (Matrices/Tables);

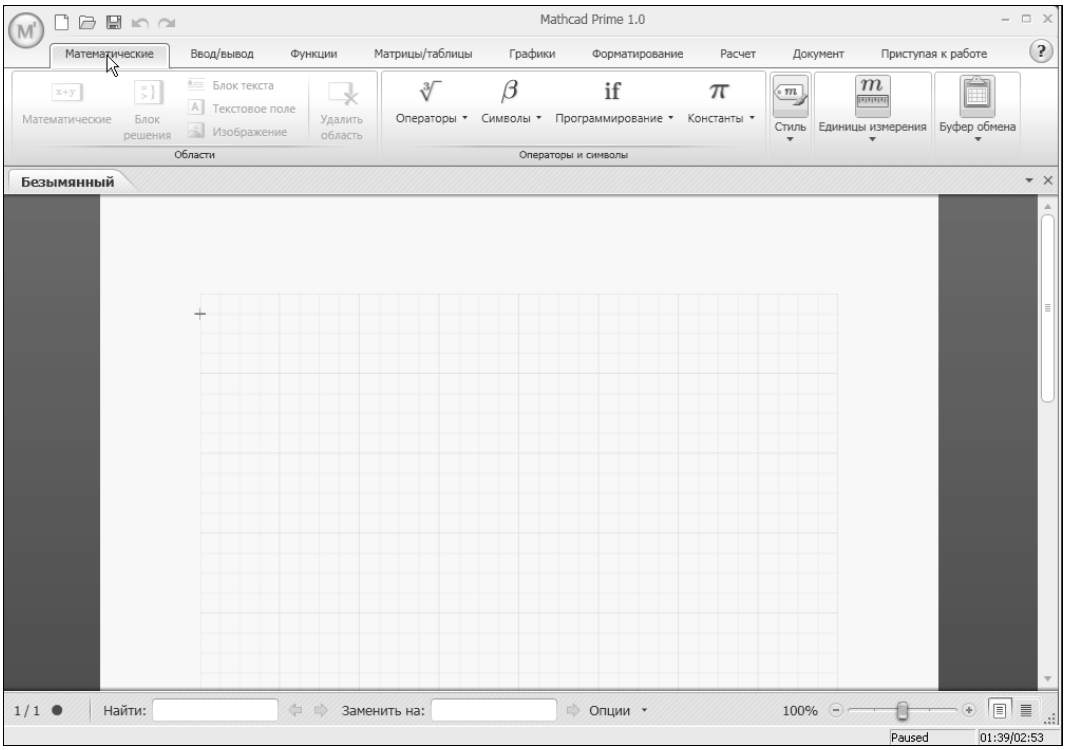


Рис. 14.1. Окно Mathcad Prime после первого запуска

- **Графики** (Plots);
- **Форматирование** (Formatting);
- **Расчет** (Calculus);
- **Документ** (Document);
- **Приступая к работе** (Getting Started).

Ленту можно скрыть, оставив на экране одни вкладки (играющие роль заголовков меню, как это показано на рис. 14.2). Для этого следует щелкнуть правой кнопкой мыши где-либо в области Ленты и выбрать в появившемся контекстном меню команду **Свернуть ленту** (рис. 14.3).

Аналогично, для повторного разворачивания Ленты достаточно в том же меню выбрать команду **Развернуть ленту** (рис. 14.2).

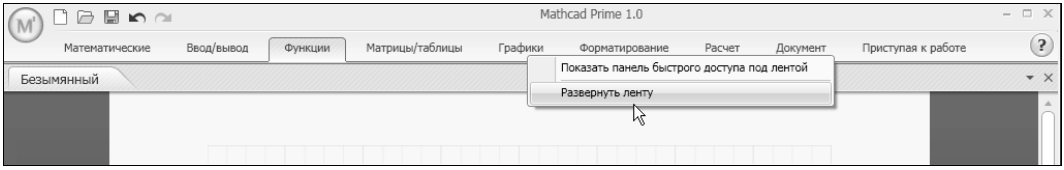


Рис. 14.2. Команды на Ленте скрыты

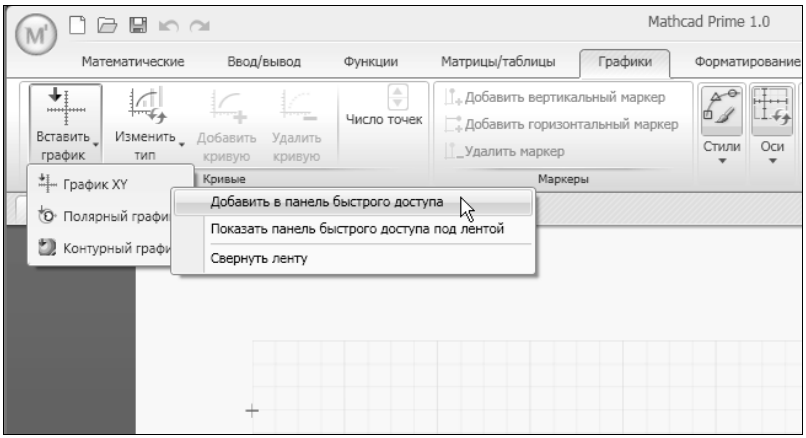


Рис. 14.3. Добавление команды на панель быстрого доступа Mathcad Prime

Еще один характерный элемент нового интерфейса — это *панель быстрого доступа*, предназначенная для быстрого осуществления наиболее частых операций. По умолчанию она содержит всего несколько кнопок, но с легкостью может быть дополнена другими командами.

Для добавления новой кнопки на панель быстрого доступа:

1. Откройте на Ленте интерфейсную группу, содержащую нужную кнопку.
2. Вызовите из области кнопки контекстное меню.
3. Выберите команду **Добавить в панель быстрого доступа**.

В результате новая кнопка появится на панели быстрого доступа (рис. 14.4).

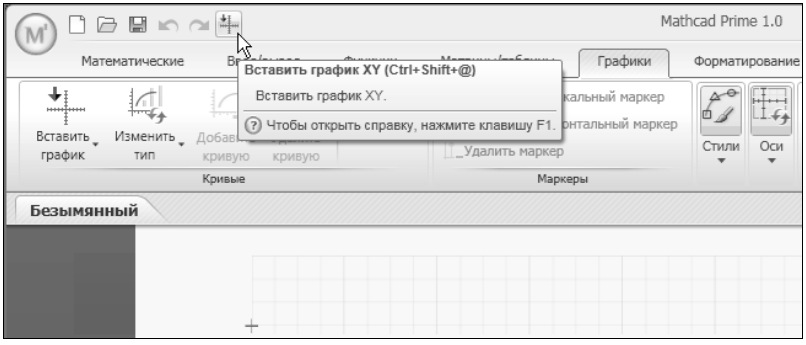


Рис. 14.4. Команда **Вставить график** добавлена на панель быстрого доступа

Для того чтобы скрыть кнопку с панели быстрого доступа, достаточно вызвать из области этой кнопки контекстное меню и выбрать команду **Удалить с панели быстрого доступа**.

Подведем итог. Основным отличием Mathcad Prime от предыдущих версий, с точки зрения пользователя, является интерфейс, основанный на применении Ленты. Те операции, которые в прежних версиях осуществлялись при помощи наборных

панелей и диалоговых окон, теперь выполняются посредством элементов управления на Ленте.

14.2. Конвертация документов Mathcad в формат Mathcad Prime

Для того чтобы использовать расчеты, подготовленные в предыдущих версиях Mathcad (они могли иметь то или иное расширение, в зависимости от типа файла и версии программы: .mcd, .xmcd, .mct, .xmct и .xmcdz), в Mathcad Prime 1.0 (.mcdx), их необходимо предварительно конвертировать в новый формат документов.

Для преобразования документов в Mathcad необходимо выполнить следующее:

1. Установите на своем компьютере "обычный" Mathcad 15 (если его не будет, программа конвертации может работать неправильно).
2. Откройте вкладку **Ввод/вывод**.
3. В группе **Mathcad-документы** нажмите кнопку **XMCD-, MCD-преобразователь** (рис. 14.5).

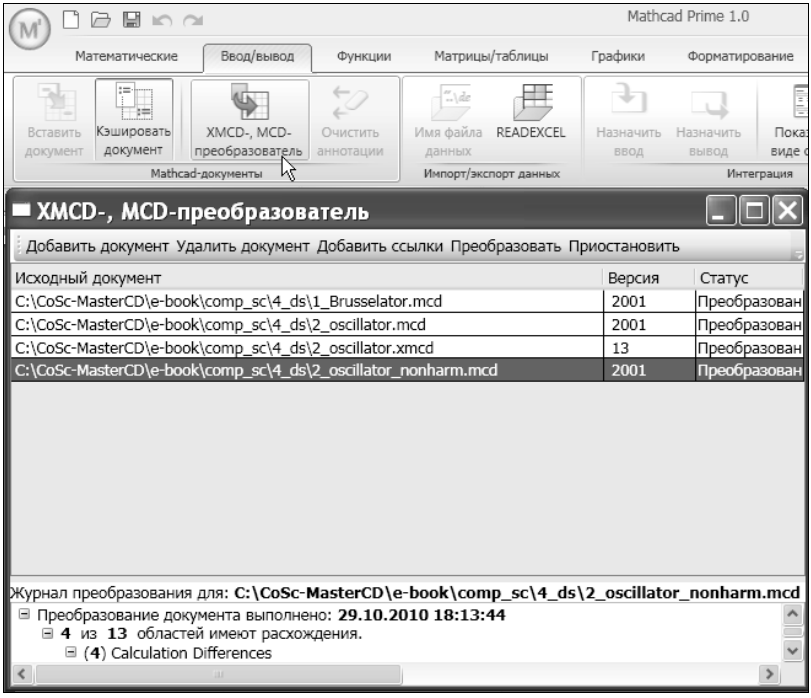


Рис. 14.5. Конвертация документов в формат Mathcad Prime

4. Нажмите кнопку **Добавить документ**, расположенную в верхней части открывшегося окна **XMCD-, MCD-преобразователь**.

Внимание!

Конвертация документов, в которых присутствуют свернутые заблокированные области, невозможна. Перед преобразованием их необходимо разблокировать. Кроме того, рекомендуется большие таблицы входных данных перед преобразованием вывести во внешний файл с тем, чтобы после конвертации вывести их из этого файла.

5. В диалоговом окне **Открыть** выберите документы, которые необходимо преобразовать, и нажмите кнопку **Открыть** (рис. 14.6).
6. Нажмите кнопку **Преобразовать**.

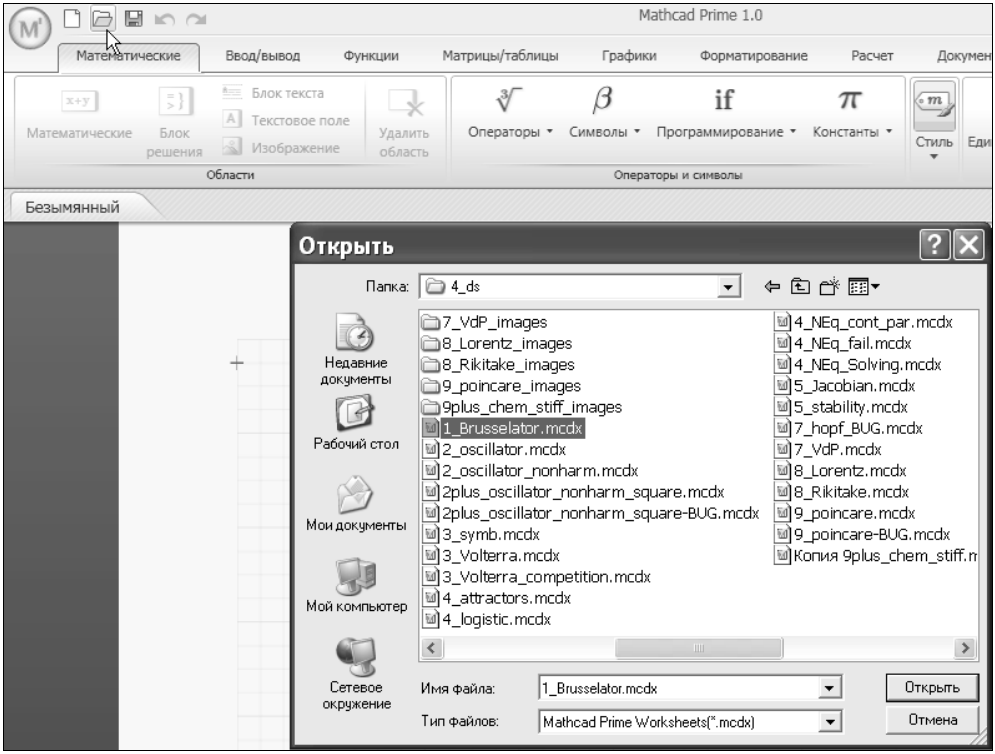


Рис. 14.6. Выберите документы для преобразования в формат Mathcad Prime

После этого будет запущен процесс конвертации, который может занять некоторое время (зависящее от объема расчетов в документах и от их общего числа). При каждом успешном преобразовании файла в папке, где находится исходный файл, будет создан новый файл формата Mathcad Prime, а исходный файл изменениям подвергаться не будет.

При этом надо учитывать, что не все возможности Mathcad 14 и 15 поддерживаются в Mathcad Prime. Фрагменты, не конвертированные в формат Mathcad Prime 1.0, будут заменены рисунками, а для всех отличий от исходного файла будет автоматически добавлена соответствующая аннотация. Пример открытия преобразованного файла в Mathcad Prime показан на рис. 14.7.

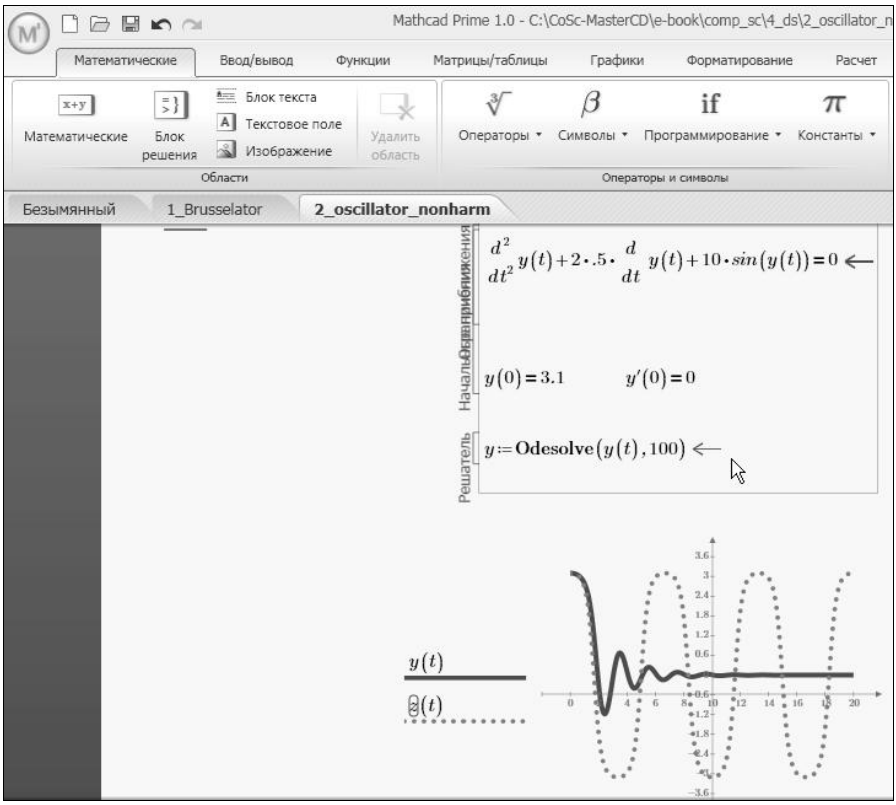


Рис. 14.7. После конвертации файл можно открыть в Mathcad Prime

Примечание

Более подробную информацию о конвертации документов вы найдете в Руководстве по миграции, которое входит в комплект поставки Mathcad Prime 1.0 (в том числе на русском языке).

14.3. Примеры расчетов

Методика расчетов в Mathcad Prime почти та же самая, что и в предыдущих версиях Mathcad, а различия связаны, в основном, с изменениями в интерфейсе (что, чаще всего, сводится к использованию команд на Ленте вместо наборных панелей). Достаточно будет всего нескольких примеров, чтобы освоить особенности Mathcad Prime.

14.3.1. Типы областей

В Mathcad Prime предусмотрены следующие виды областей:

- формула (или математические);
- таблица;

- график;
- блок решения;
- рисунок (изображение);
- текстовое поле и блок текста.

Для каждой области (рис. 14.8) предусмотрены соответствующие инструменты редактирования.

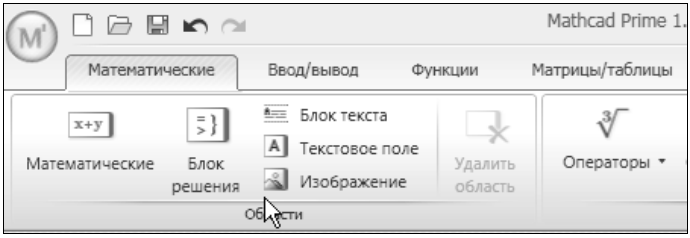


Рис. 14.8. Типы областей документов Mathcad Prime

14.3.2. Формулы

Для того чтобы добавить в документ новую формулу, достаточно щелчком в нужном месте документа поместить в него курсор ввода и сразу начать ввод математического выражения при помощи клавиатуры или команд Ленты. Создать область формул в месте документа, отмеченном курсором ввода, также можно нажатием соответствующей кнопки на Ленте (рис. 14.9).

Для ввода простых формул в документ в Mathcad Prime следует использовать группу команд **Операторы и символы** вкладки Ленты **Математические** (рис. 14.9).

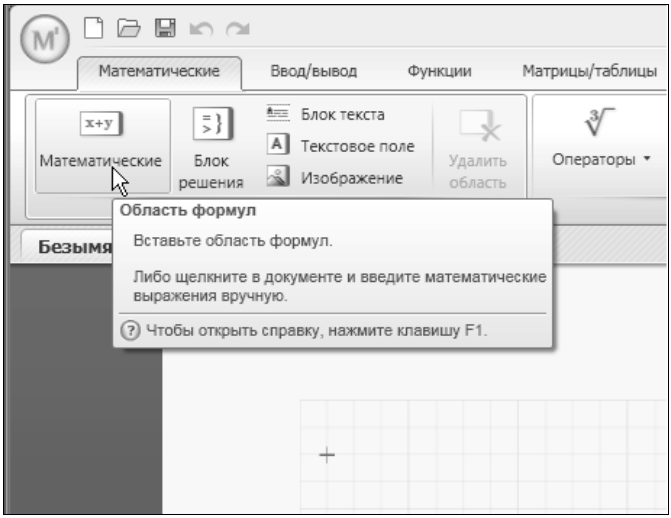


Рис. 14.9. Ввод формулы в документ

Как и в прошлых версиях Mathcad, достаточно завершить формулу знаком равенства (оператором численного вывода), чтобы рассчитанный результат появился в правой части формулы (рис. 14.10). Конечно, результат появится автоматически, если на вкладке **Расчет** включен режим фоновых вычислений.

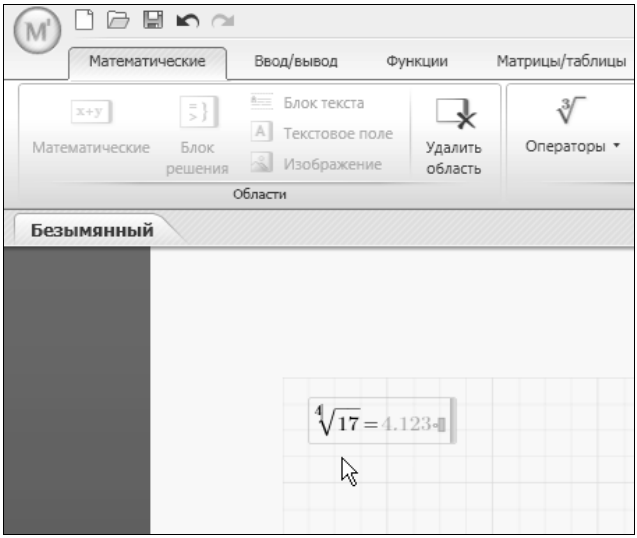


Рис. 14.10. Ввести формулу в документ можно, начав ввод символов в рабочей области

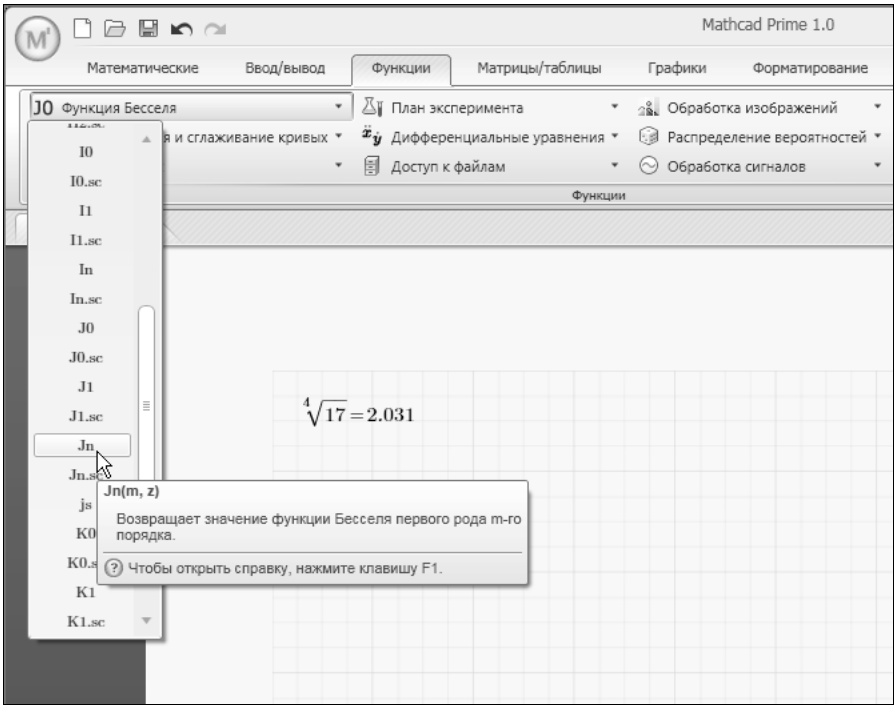


Рис. 14.11. Вставка функции

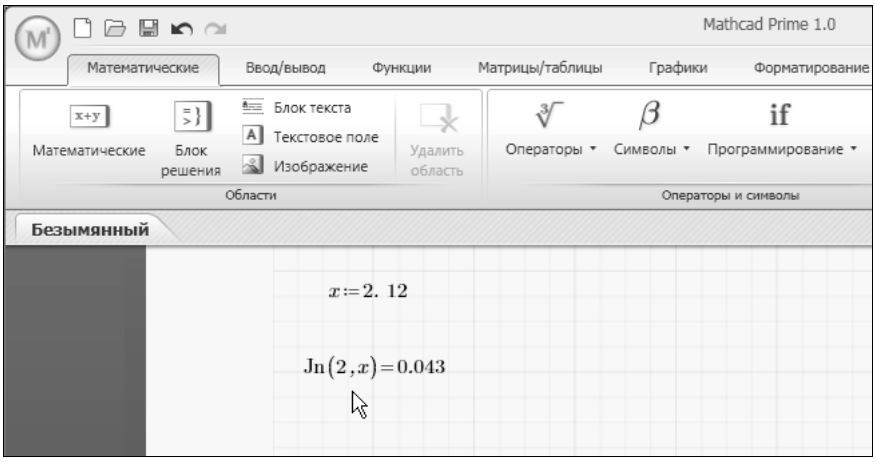


Рис. 14.12. Функция в документе

Для ввода функций предназначена вкладка Ленты **Функции** (рис. 14.11). Пример использования функции в формуле приведен на рис. 14.12.

Таким образом, вы видите, что работа с формулами в Mathcad Prime осуществляется точно так же, как и в традиционных версиях Mathcad, а единственным отличием является использование команд Ленты вместо наборных панелей, составлявших основу интерфейса в предыдущих версиях. Небольшие отличия, которые вы, возможно, даже не заметите, связаны с улучшениями редактора формул, облегчающими ввод математических выражений, применением "комбинированных операторов" (например, совмещением квадратного и N-го корней) и т. п.

14.3.3. Блоки решения

Более существенные изменения — опять-таки, формы, но не содержания — коснулись *вычислительных блоков*, или, как они называются в русской версии, *блоков решения* (*solve blocks*).

Если в прежних версиях компоненты блоков решения (начальное приближение, формулировка задачи и результат) могли быть "разбросаны" по странице, то теперь они жестко структурированы и объединены в единый контейнер, представляясь пользователю законченными элементами интерфейса, которые можно целиком перемещать по странице.

Для вставки блока решения:

1. Нажмите на Ленте кнопку **Блок решения** (рис. 14.13).
2. Внутрь появившегося в документе блока введите его составляющие: задание начальных приближений, необходимых для работы встроенной функции-решателя, дополнительные уравнения и ограничения (если они содержатся в формулировке задачи) и, собственно, решатель, задающий алгоритм использования вычислительного блока.

Пример работы введенного вычислительного блока приведен на рис. 14.14.

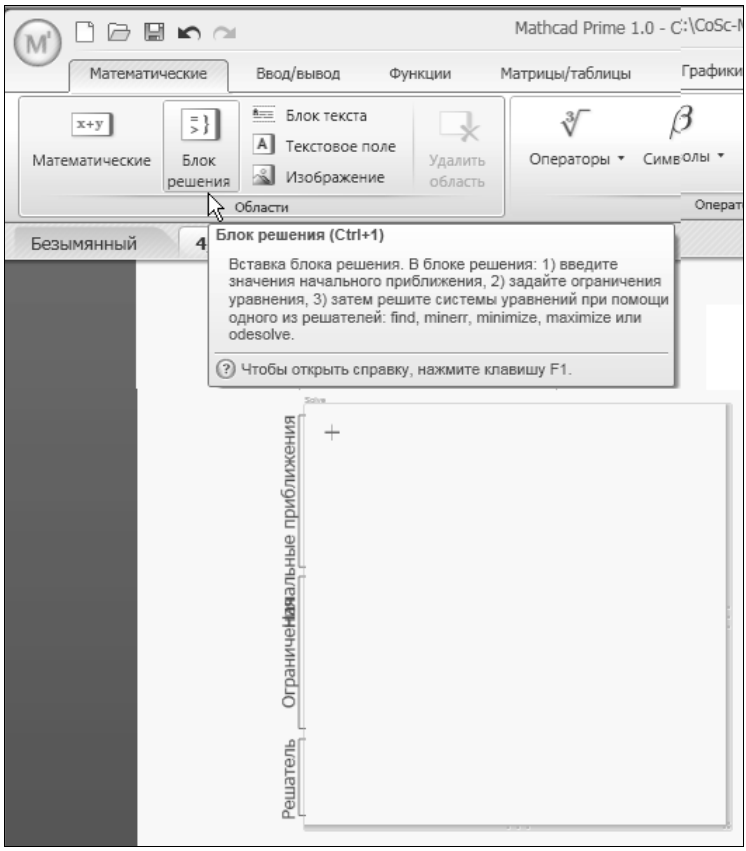


Рис. 14.13. Вставка блока решения

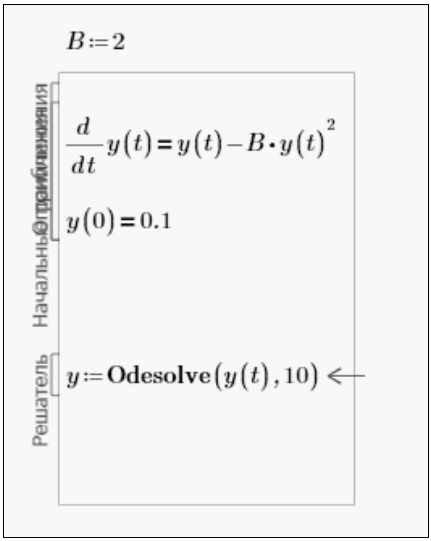


Рис. 14.14. Блок решения в документе

14.3.4. Текст

Для вставки текстового объекта нажмите соответствующую кнопку на вкладке **Математические** или **Документ** (рис. 14.15).

Учтите, что в Mathcad Prime документ может содержать текстовые примечания к расчетам двух типов:

- ❒ текстовые блоки — фрагменты текста, занимающие всю ширину страницы (рис. 14.15);
- ❒ текстовые поля — области произвольного размера, привязанные к определенному месту документа.

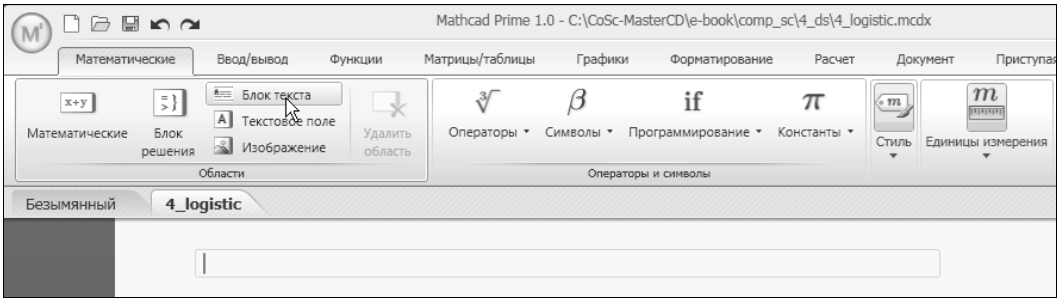


Рис. 14.15. Вставка текстового блока

Различия между текстовым блоком и текстовым полем иллюстрирует рис. 14.16. При вводе или удалении строк в текстовом блоке области, находящиеся ниже блока, соответственно сдвигаются вниз или вверх.

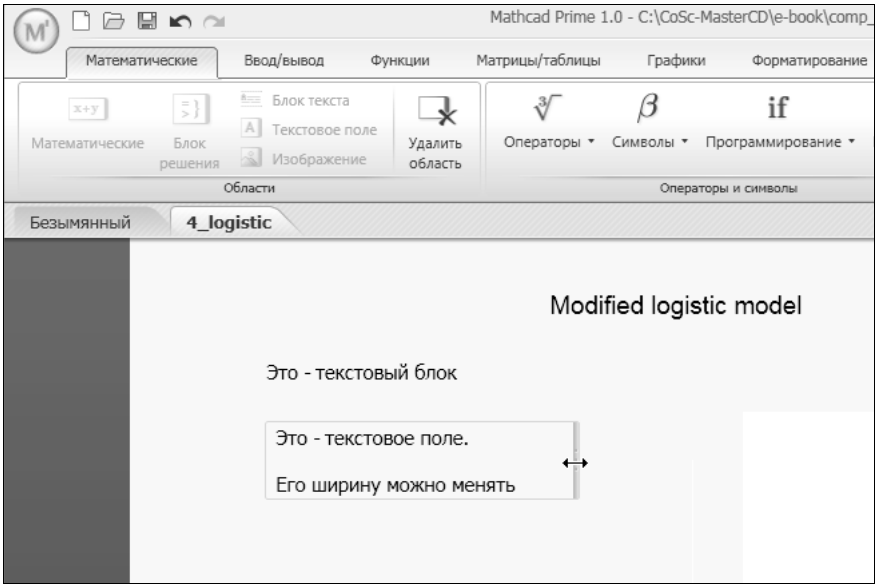


Рис. 14.16. Текстовый блок и текстовое поле

Техника форматирования текста в Mathcad Prime очень похожа на технику использования современных текстовых редакторов (в первую очередь, Microsoft Word последних версий, основанных на тех же самых принципах компоновки интерфейса, включая Ленту). В частности, для выбора параметров шрифта и абзацев применяется вкладка **Форматирование**. Поэтому не будем подробно останавливаться на этом вопросе, упомянув только расширенные возможности по созданию колонтитулов, в том числе добавления на них номеров страниц и рисунков (рис. 14.17).

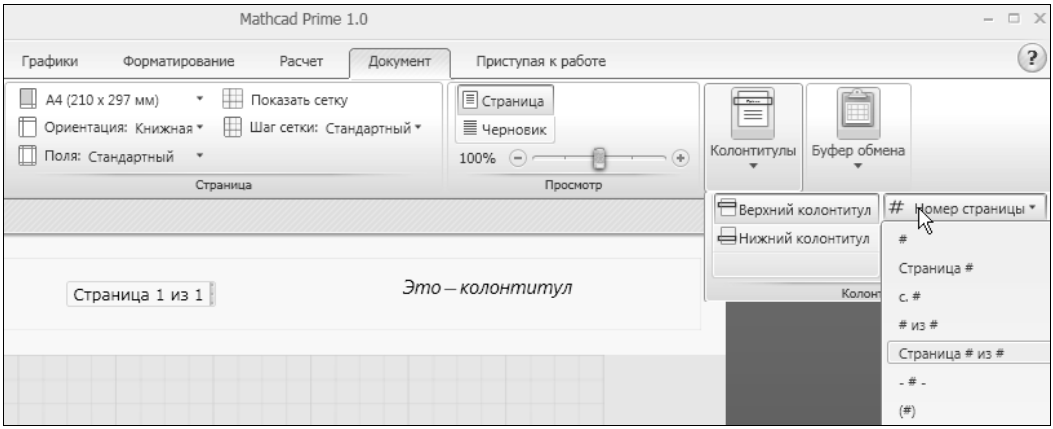


Рис. 14.17. Правка колонтитулов

Текст колонтитулов разрешается править наравне с текстом, собственно, документа. Кроме того, можно выбирать расположение текстовых областей в колонтитулах (правда, стоит отметить, что расчет формул в колонтитулах не производится). Для редактирования колонтитулов предназначена интерфейсная группа **Колонтитулы**, расположенная на Ленте на вкладке **Документ**.

Попутно стоит отметить, что в Mathcad Prime для редактирования документа предусмотрены два режима — черновика (с одной стороны, требующий меньше вычислительных ресурсов на отображение интерфейса, а с другой — максимизирующий размер рабочей области документа) и предварительного просмотра страницы. В режиме предварительного просмотра (рис. 14.17) документ отобража-

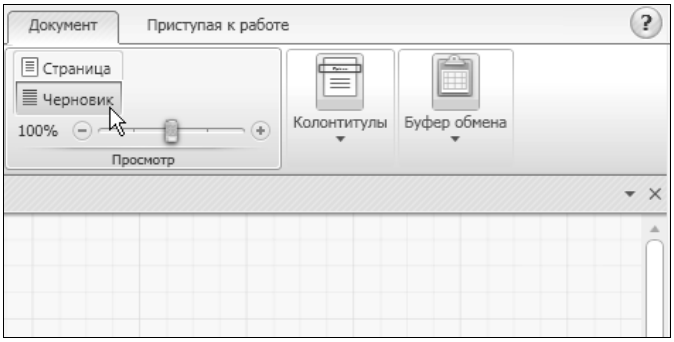


Рис. 14.18. Переключение в режим черновика

ется в том же виде, в котором он должен быть распечатан на принтере, а в режиме черновика второстепенные элементы оформления, не влияющие на расчеты (в том числе колонтитулы), скрыты (рис. 14.18).

14.3.5. Размерные переменные

В Mathcad Prime большинство объектов, в том числе графики и матрицы, способны работать с единицами измерения. При этом каждый элемент вектора, матрицы или таблицы, а также каждая кривая на графике может иметь собственную единицу измерения. Для вставки единиц измерения достаточно использовать соответствующую кнопку Ленты (рис. 14.19).

Важно отметить, что обработка единиц измерения является динамической, в частности, проверка корректности использования единиц измерения осуществляется при вычислении функции, а не при ее определении.

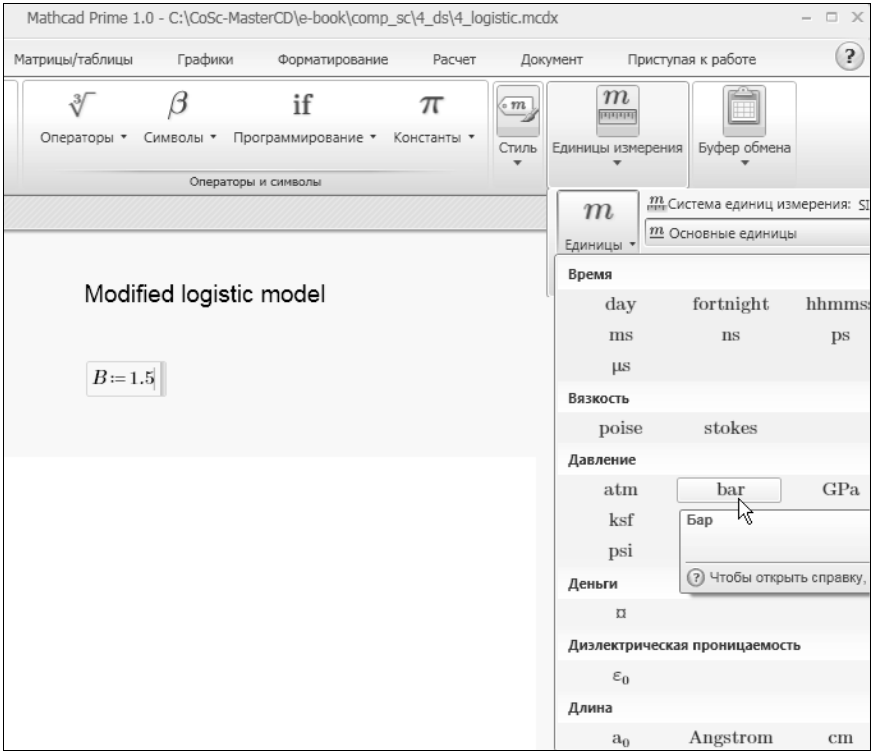


Рис. 14.19. Определение размерной переменной

14.3.6. Отладка

Mathcad Prime обладает всеми средствами отладки, что и предыдущие версии Mathcad, включая возможность трассировки ошибок. При возникновении ошибки в какой-либо формуле ее проблемный фрагмент выделяется красным обрамлением, а рядом возникает всплывающий текст, конкретизирующий ошибку (рис. 14.20).

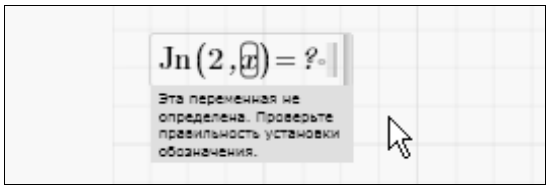


Рис. 14.20. Сообщения об ошибках

14.4. Графики

Существенным ограничением первой версии Mathcad Prime 1.0, по сравнению с предшествующими версиями Mathcad, является отсутствие трехмерных графиков (по заверениям разработчиков, трехмерные графики должны появиться в следующих релизах программы). В то же время, появились новые типы двумерных графиков.

Работа с двумерными графиками в Mathcad Prime осуществляется примерно в соответствии с теми же принципами, что и в традиционных версиях Mathcad, отличаясь тем, что проведение большинства операций теперь перенесено на вкладку Ленты **Графики**. Отметим, что оси графика теперь можно перетаскивать мышью.

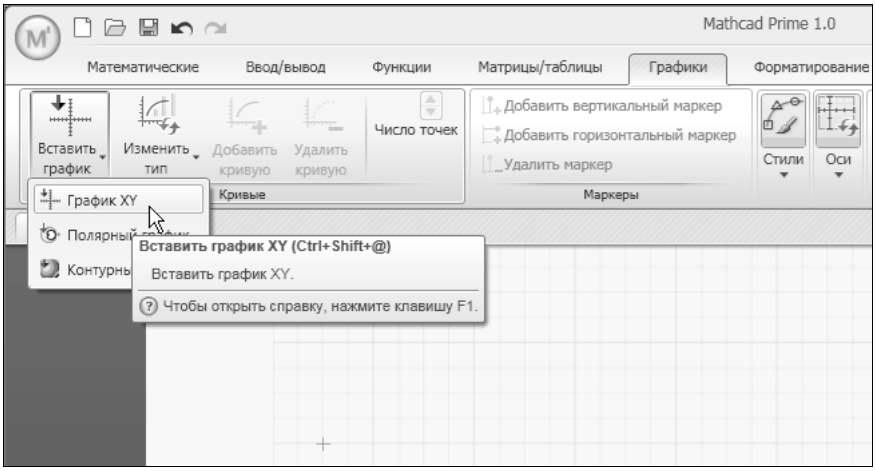


Рис. 14.21. Вставка графика

Конспективно рассмотрим основные шаги по вставке в документы графиков.

1. Откройте на Ленте вкладку **Графики** (рис. 14.21).
2. Нажмите кнопку **Вставить график** и выберите в меню желаемый тип графика.
3. В местозаполнители, находящиеся рядом с осями координат, введите имена переменных или выражения, которые вы хотите отложить по осям (рис. 14.22). В результате, если они введены корректно, на графике появится соответствующая кривая (рис. 14.23).

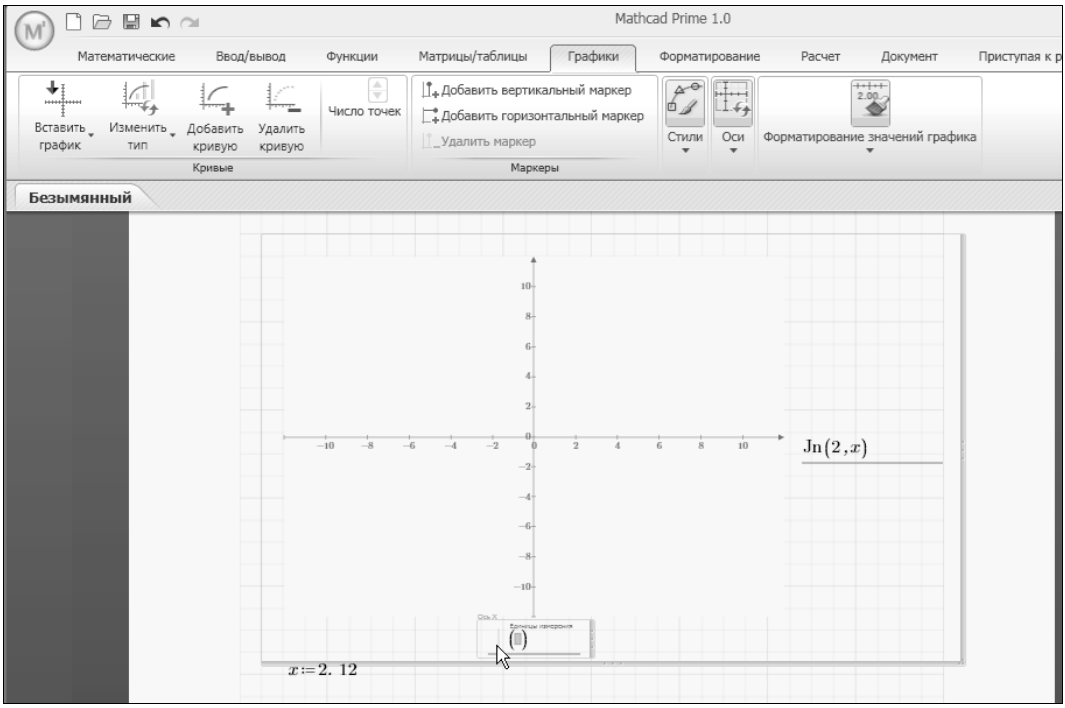


Рис. 14.22. График в документе

4. При желании справа от имени переменных укажите их размерности.
5. Если вы хотите отложить на одном и том же графике несколько зависимостей, нажмите на Ленте кнопку **Добавить кривую** (рис. 14.23). Затем укажите выражения, которые задают новую кривую, и она появится на графике (рис. 14.24).
6. Для того чтобы удалить кривую, выделите ее на графике и нажмите на Ленте кнопку **Удалить кривую** (рис. 14.24).
7. Для того чтобы добавить маркер значения на какой-либо оси, нажмите на Ленте соответствующую кнопку **Добавить вертикальный маркер** или **Добавить горизонтальный маркер** (рис. 14.25).
8. Для удаления маркера выделите его на графике и нажмите кнопку **Удалить маркер** (рис. 14.25).
9. Чтобы отредактировать цвет и стиль отображения самих кривых, а также символов, отображаемых на кривой, используйте остальные команды Ленты, расположенные на вкладке **Графики** (рис. 14.26).

В заключение отметим, что в Mathcad Prime появилось несколько новых типов двумерных графиков, например "ящик с усами и фронт Парето", а контурные графики снабжены цветовой полосой, выполняющей роль оси z.

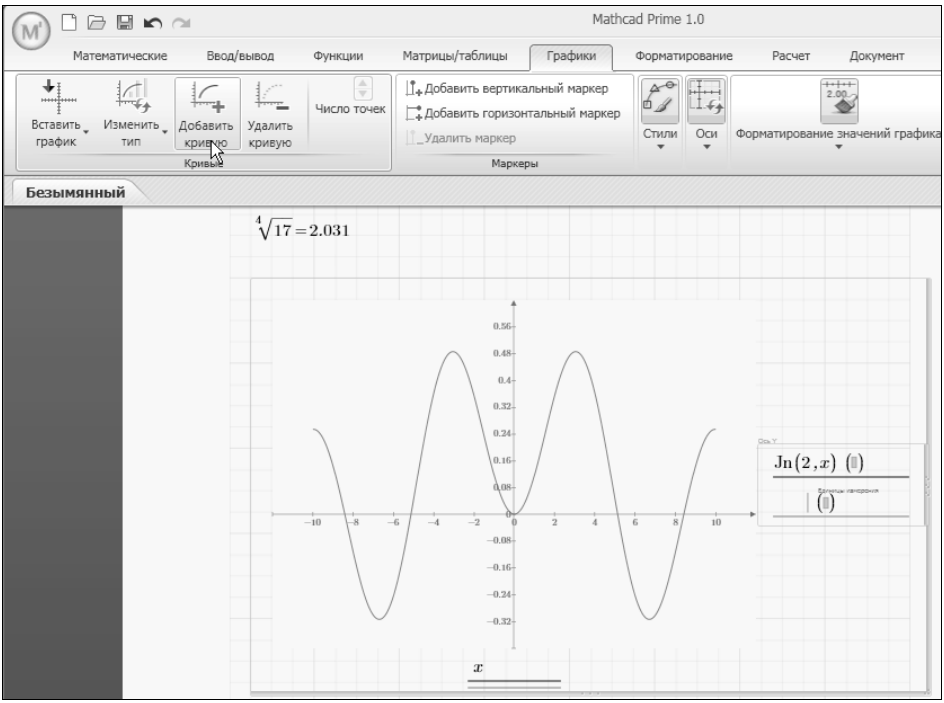


Рис. 14.23. Добавление рядов данных

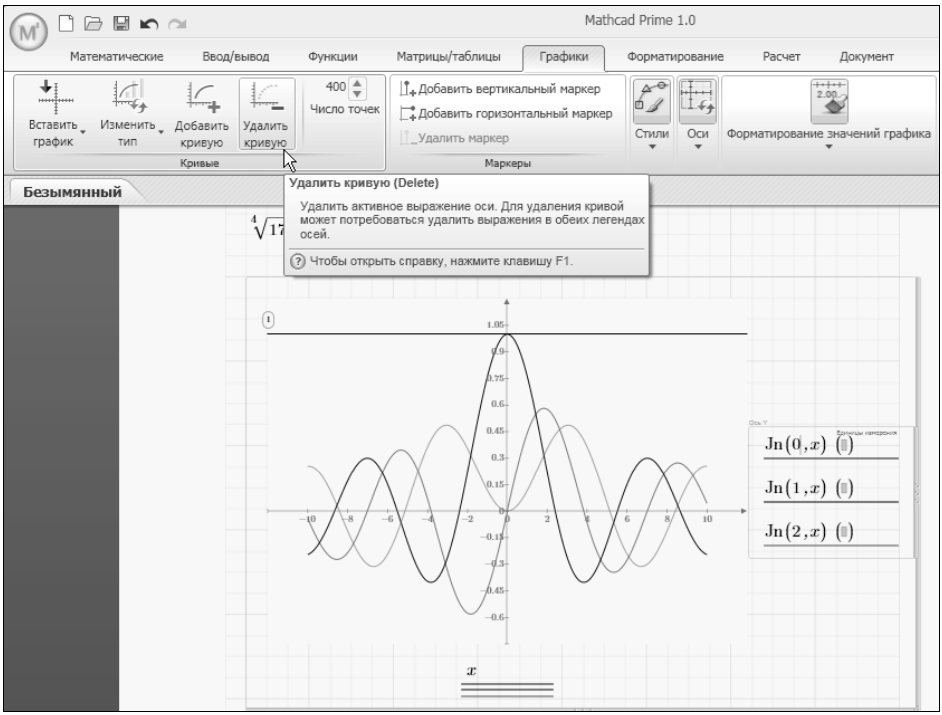


Рис. 14.24. Удаление кривой графика

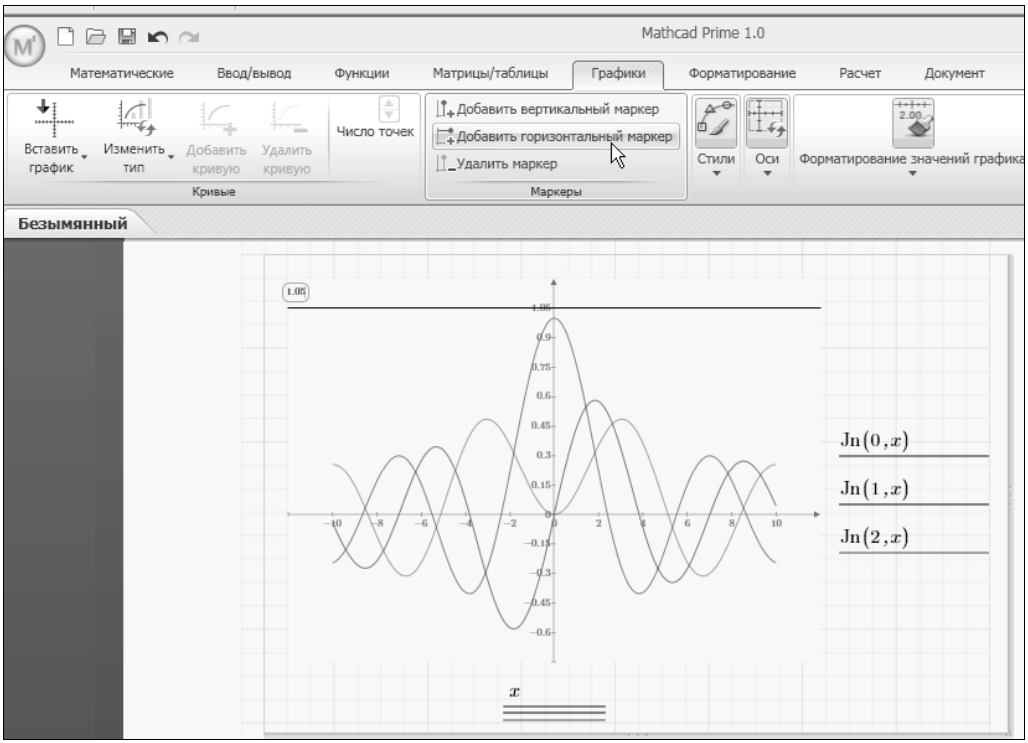


Рис. 14.25. Вставка маркера

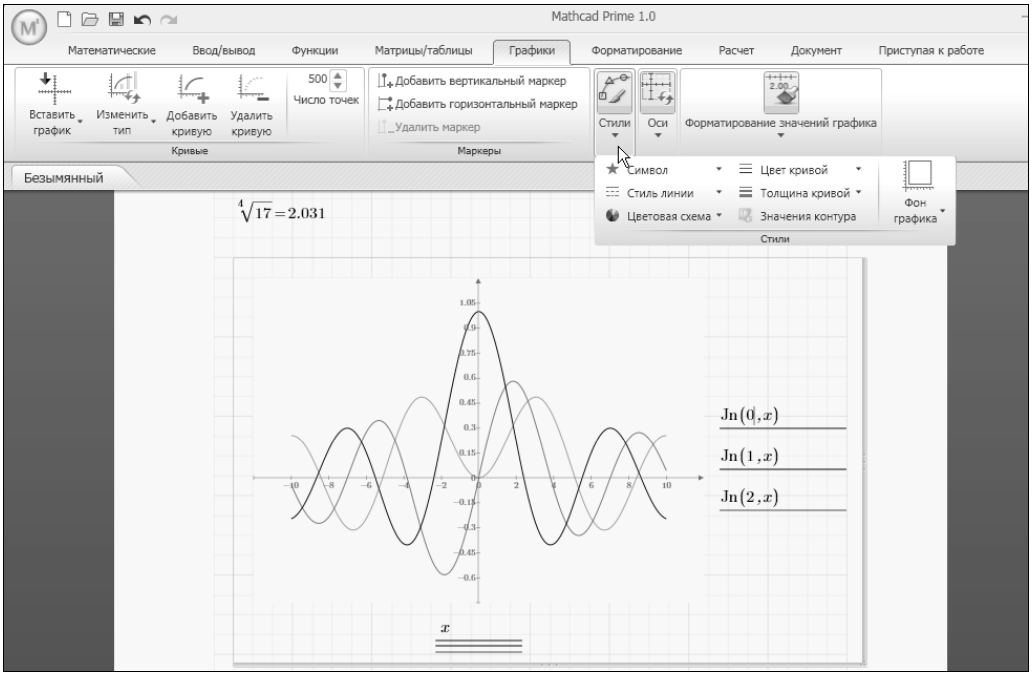


Рис. 14.26. Форматирование кривых графика

ПРИЛОЖЕНИЕ

Описание компакт-диска

Архив прилагаемого к книге компакт-диска можно скачать по ссылке **ftp://85.249.45.166/9785977507462.zip**. Ссылка также доступна на странице книги на сайте **www.bhv.ru**.

П.1. Содержимое диска

На компакт-диске вы найдете (табл. П.1):

- полное содержание почти всех примеров и листингов в формате Mathcad (по возможности 2001), объединенных в электронную книгу Mathcad;
- HTML-учебник автора по вычислительной математике (зеркало сайта **www.keldysh.ru/comma**) с примерами расчетов в Mathcad;
- полный текст книги "Вычислительная физика" (Полибук Мультимедиа, 2006) с мультимедийным видеокурсом и примерами в формате Mathcad;
- видеокурс по работе с редактором Mathcad и демо-версию мультимедийного курса по вычислительной математике (виртуальные видеолекции).

Таблица П.1. Содержание компакт-диска

index.html	Web-страница с приглашением и справочной информацией об использовании диска
kirianov.hbk	Стартовый файл электронной книги с примерами листингов и рисунков
course	Папка с видеокурсом
course / pdf	Папка с книгой автора по вычислительной физике в формате PDF
kirianov	Папка с документами Mathcad (листинги из книги)
book/comma	Папка с электронным учебником по вычислительной математике
html	Папка со служебными файлами

П.2. Как пользоваться расчетами Mathcad

Диск содержит большинство листингов и рисунков к книге в формате документов Mathcad. Вы можете просматривать их и использовать фрагменты для копирования в ваши собственные расчеты. Наиболее удобный вариант работы с документами —

это просмотр в виде электронной книги Mathcad. Такой способ, во-первых, позволяет быстро переходить от одного документа к другому (не разыскивая и не открывая каждый раз нужный файл) и, во-вторых, допускает возможность одновременно держать открытым в основном окне Mathcad ваш собственный документ.

Для того чтобы открыть примеры в формате электронной книги Mathcad, выполните следующее:

1. Запустите Mathcad.
2. Выберите в меню команду **Help / Open Book** (Справка / Открыть книгу).
3. В диалоговом окне **Open Book** (Открыть книгу) перейдите к корневой папке компакт-диска и откройте файл kirianov.hbk.
4. На появившейся странице-приглашении (рис. П.1) нажмите кнопку **Go!**. Имейте в виду, что электронная книга открывается в новом окне (независимом от основного окна Mathcad), снабженном специальной панелью инструментов навигации.

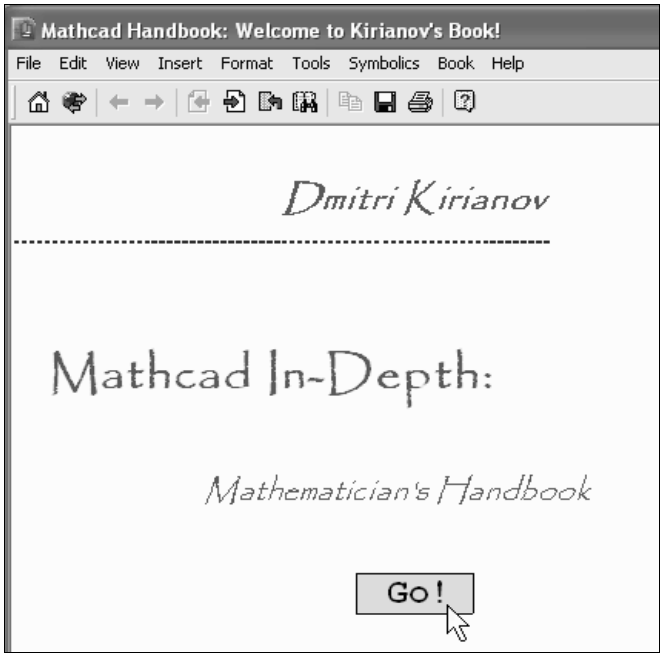


Рис. П.1. Стартовая страница-заставка

Примечание

Во избежание недоразумений, связанных с воспроизведением кириллицы в прежних версиях Mathcad (2001—13), электронная книга имеет англоязычный интерфейс.

Просмотр документов Mathcad, входящих в электронную книгу, организован следующим образом.

1. Со страницы-оглавления (рис. П.2) можно перейти к содержанию отдельных глав.

2. Для того чтобы возвратиться к оглавлению из любого места электронной книги, достаточно нажать кнопку **Home** (Домой) на панели инструментов навигации (на рис. П.2 на нее наведен указатель мыши).
3. Просматривайте все страницы электронной книги (по порядку) при помощи кнопок **Next topic** и **Previous topic** на панели инструментов (рис. П.3).

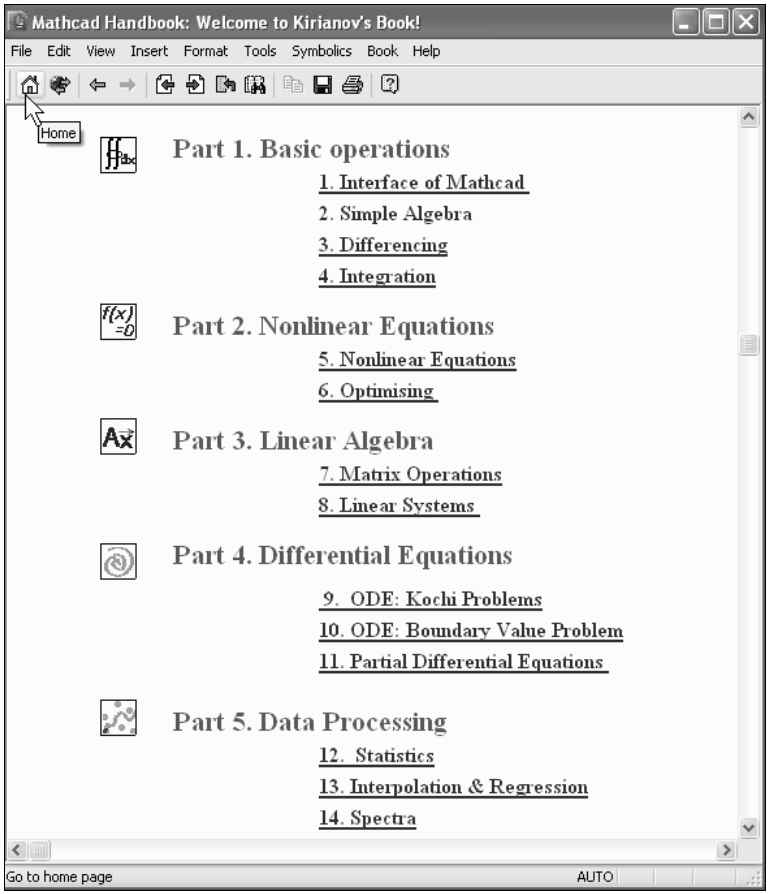


Рис. П.2. Домашняя страница с общим оглавлением электронной книги

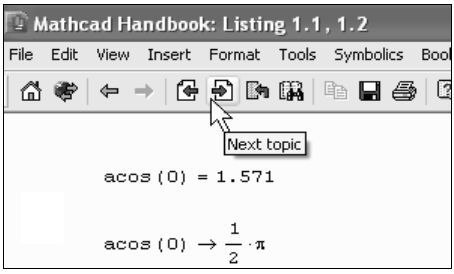


Рис. П.3. Просматривать документы книги по порядку можно при помощи кнопок **Next topic** и **Previous topic**

4. Чтобы просмотреть избранные расчеты, пользуйтесь гиперссылками в оглавлениях.
5. Если вы хотите использовать фрагменты страниц книги в ваших собственных расчетах, выделите их и нажмите клавиши <Ctrl>+<C>. Затем вернитесь в свой документ и вставьте фрагмент нажатием клавиш <Ctrl>+<V>.

Если вы по каким-либо причинам не хотите использовать документы Mathcad в форме электронной книги, то можно просматривать их как обычные документы. Для этого:

1. Выберите в Mathcad команду **File / Open** (Файл / Открыть) и перейдите к папке kirianov, находящейся на компакт-диске. В ней откройте вложенную папку с номером главы книги.
2. Откройте нужный файл с расширением mcd или xmcd (Mathcad 12—15). Например, для просмотра листинга 8.1 откройте файл 8_1.mcd, рисунка 3.4 — файл pic_3_4.mcd и т. д.

П.3. Как пользоваться видеокурсом

Книга содержит дополнительные мультимедийные материалы, объединенные в видеокурс, выполненный в формате Adobe Flash.

1. Запустите видеокурс, щелкнув соответствующую ссылку на домашней странице, которая открывается при вставке компакт-диска в устройство для чтения компакт-диска.

   Если функция автозапуска у вас отключена, откройте папку course и запустите файл index.swf (если на вашем компьютере установлен проигрыватель Adobe Flash версии не ниже 8) либо файл index.exe (не требующий дополнительных компонентов).
2. Выберите желаемую видеолекцию из оглавления курса в верхней части экрана или используйте находящиеся рядом кнопки навигации для листания лекций вперед и назад (рис. П.4).
3. Для управления воспроизведением видеолекций, используйте стандартные кнопки медиапроигрывателя в нижней части экрана.

П.4. Как пользоваться учебником по вычислительной физике

На компакт-диске, помимо примеров Mathcad, вы найдете электронный учебник по вычислительной математике (для студентов вузов), созданный автором этой книги, а также учебник по вычислительной физике в формате PDF. Для просмотра последнего воспользуйтесь соответствующей ссылкой на домашней странице или откройте соответствующий файл в папке pdf, вложенной в папку course. Просмотр PDF осуществляется при помощи программы Adobe Reader (рис. П.5).

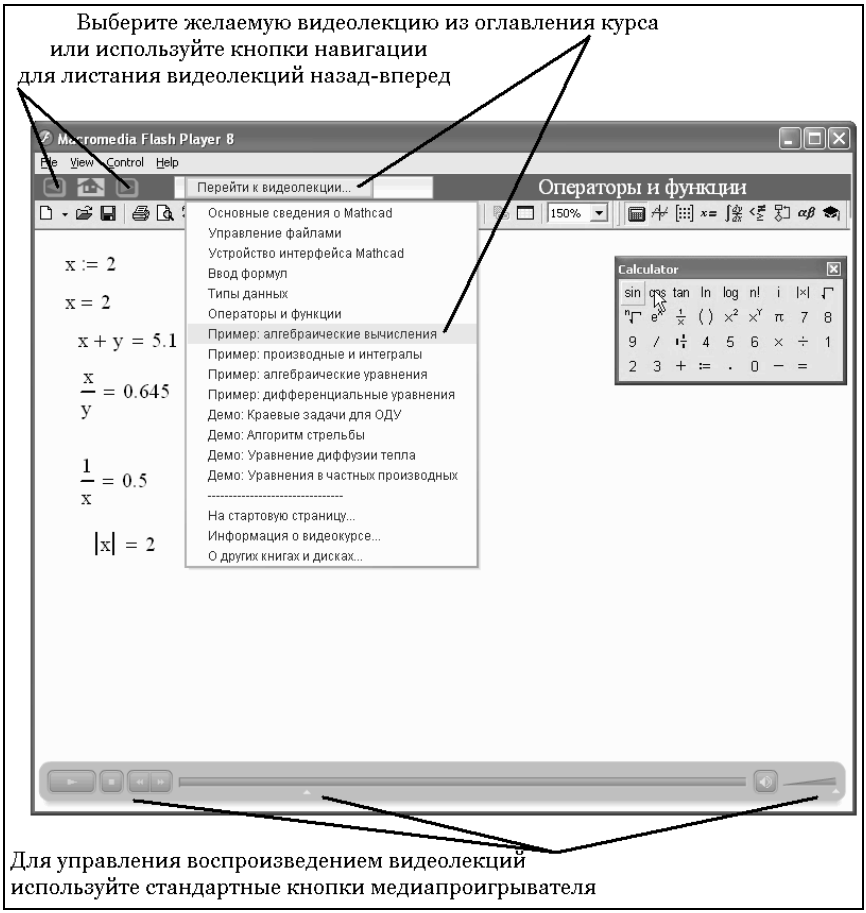


Рис. П.4. Видеокурс представляет собой мультимедийные лекции по работе с Mathcad в формате Adobe Flash

Для того чтобы открыть стартовую страницу учебника по вычислительной математике, перейдите к ней по гиперссылке с главной Web-страницы (index.html) в браузере. Домашняя страница учебника (рис. П.6) находится на компакт-диске, путь к ней выглядит следующим образом: cd:/book/comma/index.html. Учебник не требует дополнительной установки и работает исключительно в браузере.

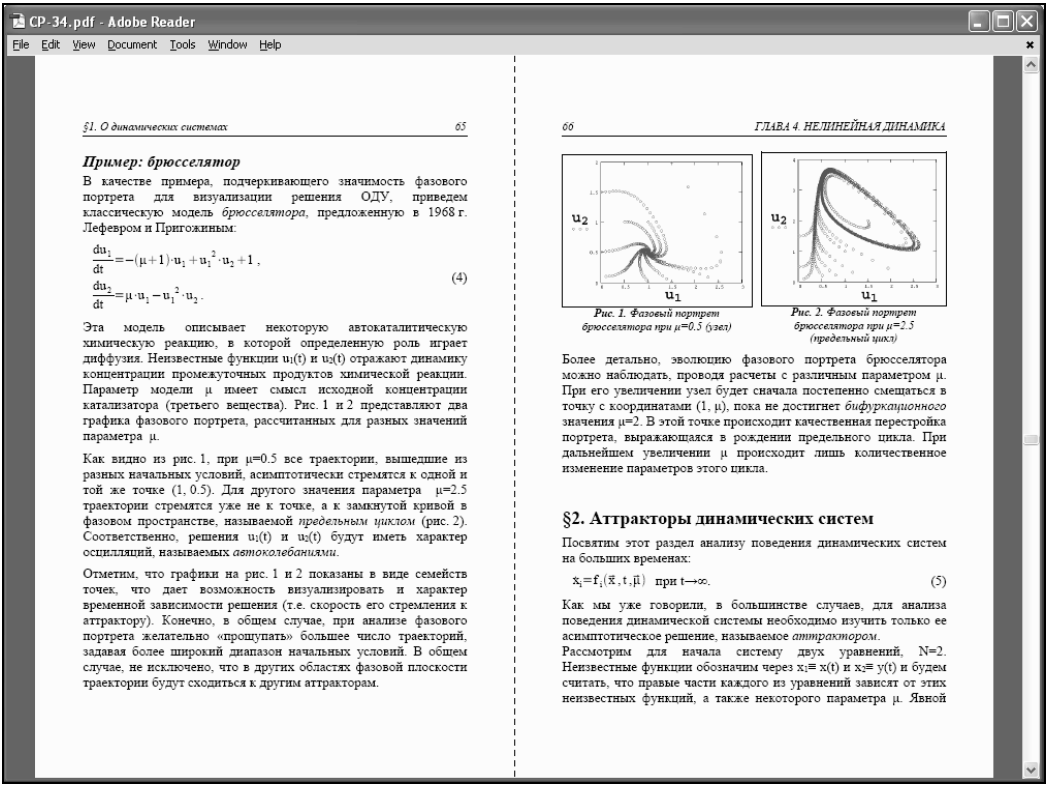
Электронный учебник выполнен в виде комбинации аудио-, видео- и информационных HTML-файлов и примерами в формате Mathcad. Перечень составляющих учебника сведен в табл. П.2. Обратите внимание, что данный диск содержит полную HTML- и Mathcad-версии учебника, а также демо-версию его мультимедийной части.

Таблица П.2. Курс по вычислительной математике

HTML-учебник	Обычный учебник с множеством гиперссылок, представляющий собой сжатое изложение материала лекционного курса. Для его просмотра не требуется никаких дополнительных средств, помимо обычного браузера

Таблица П.2 *(окончание)*

Видеолекции	Лекционный курс, содержащий видео и звук. Видеолекции могут запускаться как обычные видеофайлы (компакт-диск содержит только несколько демонстрационных лекций, полная редакция выходит на компакт-диске в издательстве "Новый Диск")
Примеры	Большое количество практических примеров, дополняющих курс. Для их полноценного просмотра требуется только браузер, а если вы хотите воспользоваться расчетами для своих задач, откройте соответствующие файлы в Mathcad
Практикум	Несколько примеров, оформленных как Java-апплеты, требуют поддержки Java



CP-34.pdf - Adobe Reader

File Edit View Document Tools Window Help

§1. О динамических системах 65

Пример: брюсселятор

В качестве примера, подчеркивающего значимость фазового портрета для визуализации решения ОДУ, приведем классическую модель *брюсселятора*, предложенную в 1968 г. Лефевром и Пригожиным:

$$\frac{du_1}{dt}=-(\mu+1)\cdot u_1+u_1^2\cdot u_2+1\,,\qquad \frac{du_2}{dt}=\mu\cdot u_2-u_1^2\cdot u_2\,. \tag{4}$$

Эта модель описывает некоторую автокаталитическую химическую реакцию, в которой определенную роль играет диффузия. Неизвестные функции $u_1(t)$ и $u_2(t)$ отражают динамику концентрации промежуточных продуктов химической реакции. Параметр модели μ имеет смысл исходной концентрации катализатора (третьего вещества). Рис. 1 и 2 представляют два графика фазового портрета, рассчитанных для разных значений параметра μ.

Как видно из рис. 1, при μ=0.5 все траектории, вышедшие из разных начальных условий, асимптотически стремятся к одной и той же точке (1, 0.5). Для другого значения параметра μ=2.5 траектории стремятся уже не к точке, а к замкнутой кривой в фазовом пространстве, называемой *предельным циклом* (рис. 2). Соответственно, решения $u_1(t)$ и $u_2(t)$ будут иметь характер осцилляций, называемых *автоколебаниями*.

Отметим, что графики на рис. 1 и 2 показаны в виде семейств точек, что дает возможность визуализировать и характер временной зависимости решения (т.е. скорость его стремления к аттрактору). Конечно, в общем случае, при анализе фазового портрета желательно «прощупать» большее число траекторий, задавая более широкий диапазон начальных условий. В общем случае, не исключено, что в других областях фазовой плоскости траектории будут сходиться к другим аттракторам.

66 ГЛАВА 4. НЕЛИНЕЙНАЯ ДИНАМИКА

Рис. 1. Фазовый портрет брюсселятора при μ=0.5 (узел)

Рис. 2. Фазовый портрет брюсселятора при μ=2.5 (предельный цикл)

Более детально, эволюцию фазового портрета брюсселятора можно наблюдать, проводя расчеты с различным параметром μ. При его увеличении узел будет сначала постепенно смещаться в точку с координатами (1, μ), пока не достигнет *бифуркационного* значения μ=2. В этой точке происходит качественная перестройка портрета, выражающаяся в рождении предельного цикла. При дальнейшем увеличении μ происходит лишь количественное изменение параметров этого цикла.

§2. Аттракторы динамических систем

Посвятим этот раздел анализу поведения динамических систем на больших временах:

$$\dot{x}_i=f_i(\vec{x},t,\vec{\mu})\quad \text{при } t\to\infty. \tag{5}$$

Как мы уже говорили, в большинстве случаев, для анализа поведения динамической системы необходимо изучить только ее асимптотическое решение, называемое *аттрактором*.

Рассмотрим для начала систему двух уравнений, N=2. Неизвестные функции обозначим через $x_1\equiv x(t)$ и $x_2\equiv y(t)$ и будем считать, что правые части каждого из уравнений зависят от этих неизвестных функций, а также некоторого параметра μ. Явной

Рис. П.5. Учебник по вычислительной физике

Наряду с видеолекциями в медиаформате имеется соответствующее количество гипертекстовых документов, а также иллюстраций в виде Mathcad-примеров. Эти иллюстрации дают простое, но понятное представление об изучаемом предмете (т. е. о работе вычислительных методов). Главными отличительными особенностями Mathcad-примеров являются, во-первых, интерактивность (т. е. возможность пользователя изменять параметры каждого метода и наблюдать соответствующий результат) и, во-вторых, возможность применения данных Mathcad-фрагментов

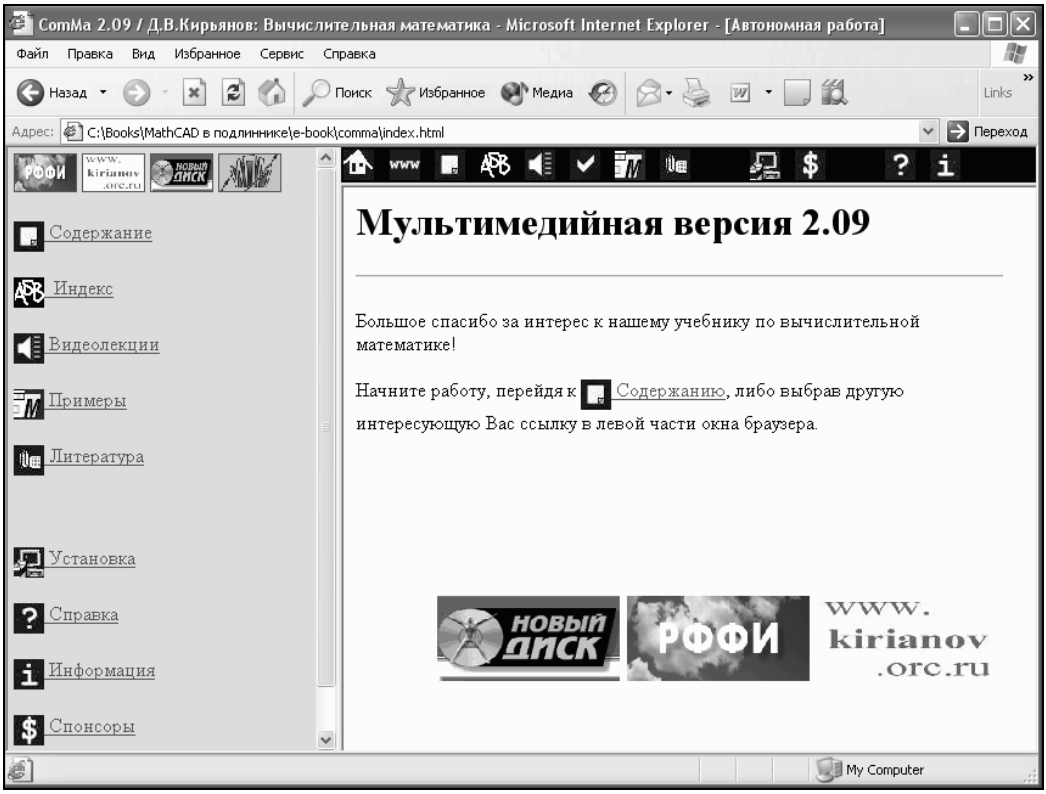


Рис. П.6. Электронный учебник по вычислительной математике открывается в браузере

в собственных математических расчетах пользователя. На рис. П.7 изображена открытая в браузере страница учебника с содержанием Mathcad-примеров. Для просмотра нужного документа с расчетами (также в браузере) достаточно щелкнуть на соответствующей гиперссылке.

Переход от одного стиля освоения материала к другому максимально упрощен благодаря единой гипертекстовой компоновке материала и возможности просмотра средствами браузеров. Таким образом, для работы с электронным учебником пользователю нужен лишь компьютер, оснащенный средствами мультимедиа, и браузер. Для дополнительной информации о работе с электронным учебником обратитесь к его разделу "Справка".

При работе с виртуальными видеолекциями учебника имейте в виду, что для прослушивания доступны только первые из мультимедийных лекций основных разделов.

Примечание

Электронный учебник по вычислительной математике (© авторы — Д. В. Кирьянов и Е. Н. Кирьянова) создан при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-07-90135). Интернет-версия учебника доступна на сервере ИПМ им. Келдыша РАН: **www.keldysh.ru/comma**.

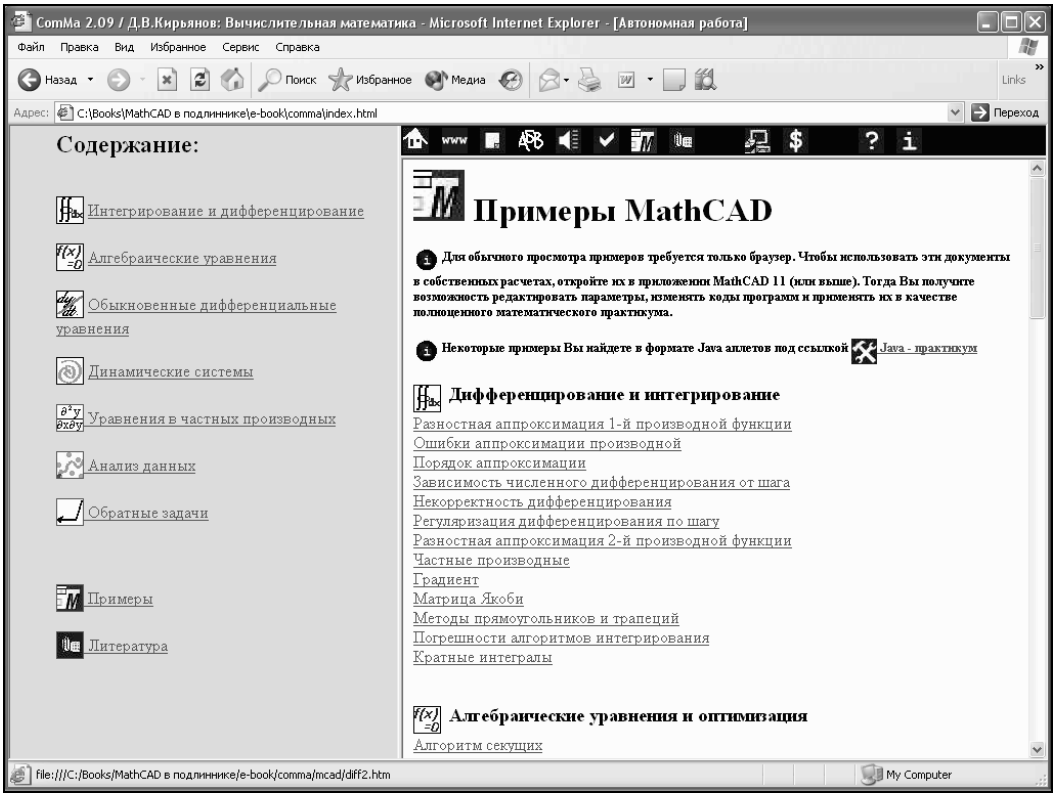


Рис. П.7. Страница учебника с оглавлением расчетов Mathcad (в формате HTML)

Предметный указатель

3

3D Bar Plot 47
3D Scatter Plot 47

A

Area 68
Array 34

B

Boundary value problem 284
Built-in constants 32
B-сплайн 370

C

Component 66
Condition number 206
Context menu 18
Contour Plot 47
Crosshair 38
CTOL 33, 160
Cumulative probability 343
CWD 33

D

Determinant 202

E

Editing lines 38
Eigenvalue 292
Error 57
Explicit 94

G

Graph area 66, 397

I

If 46
Imaginary unit 31
Insertion line 38
Inverse problems 217

L

Live symbolic evaluation 82
Local Definition 47
LU-разложение 220, 241

M

Math area 65
Math region 39
Menu bar 18
Microsoft Word 17

N

Norm 205

O

Object 66, 397
On error 57
ORIGIN 33, 35, 214
Otherwise 46

P

Placeholder 39
Polar Plot 47
Pop-up menu 18
PRNPRECISION 33

R

Range variable 34
Rank 203
Resources 19, 20, 23
Ribbon 397
RNCOLWIDTH 33

S

Seed value 358
Singular value decomposition 246
Smoothing 388
Status line 18
Surface Plot 47

T

Template 61
Text area 65
Text insertion point 38
TOL 167
TOL, встроенная константа 123
Toolbars 18
Trace 49, 197
Transpose 195

U

Underline 38

V

Vector Field Plot 47

W

Worksheet 18
WYSIWYG 16

X

XML 17
X-Y Plot 47

А

Автосохранение 63
Алгоритм
◊ Ньютона-Котеса 126
◊ последовательных исключений Гаусса 220
◊ прогонки 322
◊ прямоугольников 125
◊ Симпсона 126
Альтернатива гипотезе 352
Аппаратная функция 190
Аппроксимация 366
Аттрактор 255

Б

Бесконечность 32
Бифуркация 256
Блок решения 405
Быстрое построение графика 50

В

Вейвлет-преобразование 145
Вейвлет-спектр 147
Вектор 34
◊ собственный 248
Векторная функция 112
Векторное поле 112
Векторное произведение 200
Векторный анализ 110
Вероятность 339
Версии Mathcad 24
Верхний индекс 211
Вихрь 113
Всплывающая подсказка 20
Вставка функции 29
Встроенные константы 32
Выборка 340
Выделение области 67
Вычислительный блок 150, 157, 218, 257, 405

Г

Гиперссылка 58
Гипотеза статистическая 352
Гистограмма 346
Градиент 110
График 21, 29, 48, 52, 53
◊ поверхности 208
◊ трехмерной кривой 208

Д

Дивергенция 112
Динамическая система 254
Дисперсия 349
Дифференциальные уравнения в частных производных 307
Дифференцирование 102
◊ функций многих переменных 108
Доверительный интервал 352
Документ 60
◊ открытие 61
◊ создание 60
◊ сохранение 61
Документ Mathcad 18
Допуск
◊ ограничения 33
◊ сходимости 33

Е

Единица измерения 35
◊ вставка 35

З

Завихренность 113
Задача
◊ Коши 253, 254
◊ краевая 253
◊ некорректная 190, 193, 234, 313
◊ обратная 190
◊ транспортная 182

И

Индекс 34
Интеграл вероятности 345
Интегральное уравнение 191
Интегрирование 129, 130
◊ алгоритмы 123
◊ кратные интегралы 131
◊ оператор 121
◊ расходящиеся интегралы 130
◊ с бесконечными пределами 122, 130, 132
◊ с переменным пределом 131
Интервальная оценка 351
Интерполяция 365
◊ линейная 366
◊ сплайнами 368

К

Калькулятор 21, 26, 27, 39
Квазирешение 193
Ковариация 353
Корень уравнения 158, 162
Корреляция 353, 359
Коэффициент
◊ корреляции 353
◊ Куранта 314
Краевая задача 283
◊ для ОДУ 286
◊ жесткая 297
◊ Штурма-Лиувилля 293
Курсор 22
◊ ввода 38

Л

Лента 397
Линейная модель измерений 190
Линейное программирование 182
Линии
◊ ввода 38, 40, 202
◊ редактирования 22
Линия
◊ ввода текста 38
◊ программы 44, 45
Локальное присваивание 47

М

Максимум функции 177
Маскировка частот 142
Массив 34
◊ доступ к столбцу 35
◊ доступ к элементу 35
Математическая область 39
Матрица 34
◊ возведение в степень 204
◊ доступ к столбцу 35
◊ доступ к элементу 35
◊ единичная 210
◊ матричные операторы 195
◊ обратная 204, 212
◊ сингулярная 231
◊ создание 80
◊ треугольная 240
◊ трехдиагональная 322
◊ Якоби 115

Медиана случайной величины 349
Мексиканская шляпа 145
Местозаполнитель 25, 39
Метод
◊ градиентный 169
◊ Монте-Карло 357
◊ наименьших квадратов 223, 377
◊ Ньютона 168
◊ продолжения по параметру 174
◊ разностный 295
◊ релаксации 333
◊ Ромберга 128
◊ Рунге—Кутты 257, 259
◊ секущих 167
◊ сеток 313
◊ скользящее усреднение 390
◊ стрельбы 285
◊ Эйлера 295
Минимизация 223
Минимум функции 177
Мнимая единица 31, 32
Модель
◊ встречных световых пучков 284
◊ оптимизация транспортных издержек 184
◊ осциллятор 260
◊ собственных колебаний струны 293
◊ химической кинетики (Робертсона) 274
Модуль вектора 199

Н

Начальное значение 158, 162
◊ генератора псевдослучайных чисел 358
Невязка 223
Некорректная задача 193
Непрерывные дроби 90
Неравенство 161
Ноль-линия 141
Норма матрицы 205
Нормальное псевдорешение 228

О

Обобщенное собственное значение 249
Обратные задачи 190
Обратный ход 239
Объединение выражений 86
ОДУ 251
◊ высшего порядка 252
◊ жесткие 271, 273
◊ задача на собственные значения 292
◊ краевая задача 286
◊ начальные условия 254, 257
◊ первого порядка 252
◊ решение в одной точке 263
◊ системы 253
◊ стандартная форма (Коши) 252, 253
Окно 389, 390
Операнд 41, 77
Оператор 41, 77
◊ векторизации 201
◊ глобального присваивания 81
◊ Лапласа 309
◊ логический, или булев 79
◊ отрицания 41
Определение 27, 81
Определитель 202
Оптимизация 177
Оси графика 48
Ошибка 57

П

Палитра 21
Панели инструментов математические 21
Панель быстрого доступа 399
Переменная
◊ ранжированная 34
◊ строковая 33
Переменная — диапазон 34
Перехват ошибок 57
Подобные слагаемые 91
Подстановка 239
◊ переменной 96
Полином 92
◊ корень 164
Присваивание 81
Прогонка 239, 322
Производная 103
Простые
◊ дроби 90
◊ числа 89
Процент 32
Прямой ход 239
Псевдорешение 223

Р

Рабочая область 18
Разболтка 298

Разложение
◊ в ряд 117
◊ на элементарные функции 89
◊ Холецкого 240
Размерность 35
Разностная схема 296
◊ неявная 298
◊ устойчивость 318
◊ явная 296
Ранг матрицы 203
Ранжированная переменная 123
Регрессия 365
◊ двумерная 382
◊ линейная 377
◊ общего вида 385
Регуляризация 236
Результат 32, 37
Ресурсы 20, 23
Ротор 113
Ряд 95
◊ данных 49
◊ Тейлора 117

С

Свертка 140
Сглаживание 365, 388
Сетка 313
Сеточная функция 313
СИ 37
Сингулярное разложение 246
Сингулярность 217
Система
◊ алгебраических уравнений 150, 157
◊ дифференциальных уравнений 253
◊ линейных алгебраических уравнений 217
◊ счисления 31
Сканирование 165, 178, 255
Скрытие зоны 68
СЛАУ 217
◊ вырожденные 231
◊ несовместные 221
◊ переопределенные 221
◊ плохо обусловленные 231
◊ с прямоугольной матрицей 221
◊ с треугольной матрицей 239
◊ хорошо обусловленные 218
След матрицы 197
Случайная величина 357
◊ закон распределения 340
◊ квантиль распределения 344
◊ плотность вероятности 340
◊ с нормальным распределением 342
◊ функция распределения 343
Случайное поле 360
Случайный процесс 360
Собственное значение 248
Собственные значения 292
◊ краевой задачи 293
Собственные функции краевой задачи 293
Сортировка массива 213
Спектр
◊ двумерный 139
◊ Добеши 146
◊ Фурье 134
Сплайн-коэффициенты 368
Сплайн-интерполяция 368
Среднее значение случайной величины 349
Среднеквадратичное отклонение 349
Стационарная точка 255
Стиль текста и формул 43
Строка 33
Схема
◊ бегущего счета 315
◊ неявная 320
◊ явная 315

Т

Текстовое поле 407
Текстовый блок 407
Тензор 34
Тепловой фронт 317
Траектория 254
Транспонирование 195
Тренд 391
Треугольное разложение матрицы 241

У

Указатель мыши 38
Упрощение выражений 87
Уравнение
◊ волновое 328
◊ в частных производных 251
◊ гиперболическое 308
◊ дифференциальное в частных производных 307, 325
◊ диффузии тепла 309
◊ интегральное 191
◊ Лапласа 310
◊ обыкновенное дифференциальное (ОДУ) 251

Уравнение *(прод.)*
◊ приближенное решение 187
◊ параболическое 308
◊ Пуассона 310, 331
◊ теплопроводности 309
◊ теплопроводности, нелинейное 317
◊ теплопроводности, обратное 312
◊ теплопроводности, одномерное 310
◊ теплопроводности, стационарное 309
◊ эллиптическое 308
Уравнения в частных производных 307
Условия
◊ граничные 308
◊ начальные 308
Устойчивость 192, 318

Ф

Фазовый портрет 255
Фильтрация 365, 388, 392
Формула, правка 42
Функционал 190
Функция
◊ аппаратная 190
◊ встроенная 24
◊ Добеши 145
◊ максимум 177
◊ минимум 177
◊ ошибок 345
◊ разложение в ряд 117

Ц

Центральная предельная теорема 343

Ч

Частные производные 308
Частота
◊ граничная 139
◊ Найквиста 139
Черновик 408
Число
◊ восьмеричное 31
◊ двоичное 31
◊ действительное 30
◊ комплексное 31
◊ Куранта 314
◊ мнимое 31
◊ обусловленности 206
◊ пи 32
◊ шестнадцатеричное 31

Ш

Шаблон 60, 313

Э

Экстраполяция 365
Экстремум 177
◊ локальный 178
◊ условный 180
Электронные книги Mathcad 23

Я

Явные вычисления 94
Якобиан 115, 275