С. В. Поршнев

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

Курс лекций

- *Уравнения и системы уравнений*
- *Задачи интерполяции и аппроксимации*
- *Фурье-анализ*
- *Численное дифференцирование и интегрирование*
- *Обыкновенные дифференциальные уравнения*
- *Дифференциальные уравнения в частных производных*
- *Интегральные уравнения*

УЧЕБНОЕ ПОСОБИЕ

bhv

Сергей Поршнев

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

Курс лекций

Допущено учебно-методическим объединением вузов по университетскому политехническому образованию в качестве учебного пособия для студентов высших учебных заведений

Санкт-Петербург
«БХВ-Петербург»
2004

УДК 681.3.06+519.6
ББК 32.973Я73
П59

Поршнев С. В.

П59 Вычислительная математика. Курс лекций. — СПб.: БХВ-Петербург, 2004. — 320 с.: ил.

ISBN 5-94157-400-2

Книга представляет собой расширенный вариант лекций по курсу «Вычислительная математика», прочитанных автором в Нижнетагильском технологическом институте Уральского государственного технического университета для специальности «Информатика и вычислительная техника». Материал укладывается в перечень требований обязательного минимума содержания Государственного образовательного стандарта высшего профессионального образования по специальности 654600 «Информатика и вычислительная техника» дисциплины «Вычислительная математика». Основная особенность курса — прикладная направленность. Для каждого описанного в книге вычислительного метода приведены его программные реализации, а также соответствующие функции математического пакета MATLAB. Выбранный подход позволяет сформировать понимание математического содержания конкретного метода (границы его применимости, погрешности метода и т. д.) и умение использовать современные программные средства.

Для студентов и преподавателей профильных специальностей

УДК 681.3.06+519.6
ББК 32.973Я73

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Екатерина Капалыгина*
Компьютерная верстка	*Екатерины Трубниковой*
Корректор	*Виктория Голуб*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 23.10.03.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 25,8.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

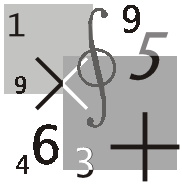
Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-400-2

Содержание

Лекция 1



Теория погрешностей

Цель лекции: изложить общие сведения об источниках и видах погрешностей, правилах вычисления абсолютной и относительной погрешностей, формах записи чисел, познакомить со способами хранения действительных чисел в памяти компьютера и ввести понятие о конечной точности машинной арифметики.

1.1. Общие сведения об источниках погрешностей, их классификация

Источниками возникновения погрешности численного решения задачи являются:

- неточность математического описания, в частности, неточность задания начальных данных;
- неточность численного метода решения задачи, возникающая, например, когда решение математической задачи требует неограниченного или неприемлемо большого числа арифметических операций, что приводит к необходимости ограничения их числа, т. е. использования приближенного решения;
- конечная точность машинной арифметики.

1.1.1. Виды погрешностей

Неустранимая погрешность состоит из двух частей:

- погрешности, обусловленной неточностью задания числовых данных, входящих в математическое описание задачи;
- погрешности, являющейся следствием несоответствия математического описания задачи реальной действительности (погрешность математической модели).

Для вычислителя погрешность задачи следует считать неустранимой, хотя постановщик задачи иногда может ее изменить.

Результирующая погрешность определяется как сумма величин всех перечисленных выше погрешностей.

Погрешность метода связана со способом решения поставленной математической задачи. Она появляется в результате замены исходной математической модели другой и/или конечной последовательностью других более простых (например, линейных) моделей. При создании численных методов закладывается возможность отслеживания таких погрешностей и доведения их до сколь угодно малого уровня. Отсюда естественно отношение к погрешности метода как устранимой (или условной).

Вычислительная погрешность (погрешность округлений) обусловлена необходимостью выполнять арифметические операции над числами, усеченными до количества разрядов, зависящего от применяемой вычислительной техники.

Рассмотрим простой пример, иллюстрирующий описанные ранее виды погрешностей, на основе задачи описания движения маятника (рис. 1.1), в которой требуется предсказать угол отклонения маятника от вертикали Θ, начинающего движение в момент времени $t = t_0$.

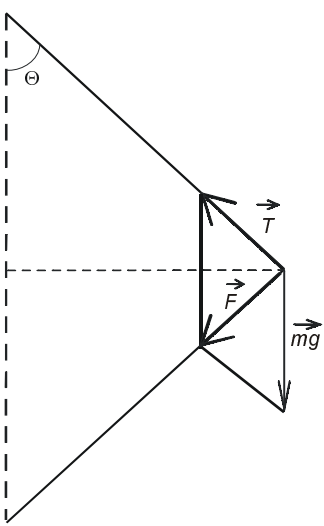


Рис. 1.1. Модель математического маятника

Движение маятника может быть описано дифференциальным уравнением второго порядка:

$$l\frac{d^2\theta}{dt^2} + g\sin\theta + \mu\frac{d\theta}{dt} = 0, \qquad (1.1)$$

где l — длина маятника, g — ускорение свободного падения, μ — коэффициент трения.

Причины возникновения погрешностей в данной задаче могут быть различными.

- Реальная сила трения зависит от скорости движения маятника по нелинейному закону.
- Значения величин l, g, μ, t_0, $y \pm e_y$, $\theta'(t_0)$ известны с некоторыми погрешностями.
- Для решения уравнения (1.1), не имеющего аналитического решения, приходится использовать численный метод, вследствие чего возникает погрешность метода.
- Вычислительная погрешность, возникающая вследствие конечной точности представления чисел в компьютере.

1.2. Абсолютная и относительная погрешности. Формы записи данных

Определение 1.1. Если a — точное значение некоторой величины и a^* — известное приближение к нему, то абсолютной погрешностью приближенного значения a^* называют некоторую величину $\Delta(a^*)$, про которую известно, что

$$|a^* - a| \le \Delta(a^*). \qquad (1.2)$$

Определение 1.2. Относительной погрешностью приближенного значения называют некоторую величину $\delta(a^*)$, про которую известно, что

$$\left|\frac{a^* - a}{a^*}\right| \le \delta(a^*). \qquad (1.3)$$

Относительную погрешность часто выражают в процентах.

Определение 1.3. Значащими цифрами числа называют все цифры в его записи, начиная с первой ненулевой слева.

Пример 1.1.

$$a^* = 0.0\mathbf{3045} \qquad a^* = 0.0\mathbf{3045000}\ ^{1}$$

[1] При записи чисел в качестве знака, разделяющего целую и дробную часть числа, мы используем, как это принято в программировании, точку.

Примечание

Здесь цифры, записанные курсивом, значащие.

Определение 1.4. Значащую цифру называют верной, если модуль погрешности числа не превосходит единицы разряда, соответствующего этой цифре.

Пример 1.2.

$$a^* = 0.0\mathbf{3045} \qquad \Delta(a^*) = 0.000003$$

$$a^* = 0.0\mathbf{30450000} \qquad \Delta(a^*) = 0.00000007$$

Примечание

Здесь цифры, записанные курсивом, верные.

Определение 1.5. Число записано со всеми верными цифрами, если в его записи представлены только верные значащие цифры.

Иногда употребляется термин *число верных цифр после запятой*: подсчитывается число верных цифр после запятой от первой цифры до последней верной цифры.

Довольно часто информация о некоторой величине задается пределами измерений

$$a_1 \le a \le a_2 .$$

Принято записывать эти пределы с одинаковым числом знаков после запятой, так как обычно достаточно грубого представления о погрешности. В записи чисел a_1, a_2 обычно берут столько значащих цифр, сколько нужно для того, чтобы разность $a_2 - a_1$ содержала одну, две значащие цифры.

Информацию о том, что a^* является приближенным значением числа a с абсолютной погрешностью $\Delta(a^*)$, принято также записывать в виде:

$$a = a^* \pm \Delta(a^*). \tag{1.4}$$

Числа a^*, $\Delta(a^*)$ принято записывать с одинаковым числом знаков после запятой.

Пример 1.3.

$$\begin{cases} a = 1.123 \pm 0.004 \\ a = 1.123 \pm 4 \cdot 10^{-3} \\ 1.123 - 0.004 \le a \le 1.123 + 0.004 \end{cases}$$

Информацию о том, что a^* является приближенным значением числа a с относительной погрешностью $\delta(a^*)$, записывают в виде:

$$a = a^*(1 \pm \delta(a^*)).$$

Пример 1.4.

$$1.123 \cdot (1 - 0.003) \le a \le 1.123 \cdot (1 + 0.003)$$

Примечание

Данная запись числа эквивалентна записи чисел из примера 1.3.

1.3. Вычислительная погрешность

Далее для краткости будем обозначать абсолютную погрешность числа x как e_x, относительную погрешность — δ_x.

❒ Погрешность суммирования чисел $x \pm e_x$, $y \pm e_y$:

- абсолютная погрешность

$$z = (x \pm e_x) + (y \pm e_y) = (x + y) \pm (e_x + e_y);$$

- относительная погрешность

$$\delta_z = \frac{e_x + e_y}{|x + y|} = \frac{e_x}{|x + y|}\frac{|x|}{|x|} + \frac{e_y}{|x + y|}\frac{|y|}{|y|} = \frac{|x|}{|x + y|}\delta_x + \frac{|y|}{|x + y|}\delta_y.$$

❒ Погрешность вычитания чисел $x \pm e_x$, $y \pm e_y$:

- абсолютная погрешность

$$z = (x \pm e_x) - (y \pm e_y) = (x - y) \pm (e_x + e_y);$$

- относительная погрешность

$$\delta_z = \frac{e_x + e_y}{|x - y|} = \frac{e_x}{|x - y|}\frac{|x|}{|x|} + \frac{e_y}{|x - y|}\frac{|y|}{|y|} = \frac{|x|}{|x - y|}\delta x + \frac{|y|}{|x - y|}\delta y.$$

❒ Погрешность умножения чисел $x \pm e_x$, $y \pm e_y$:

- абсолютная погрешность

$$z = (x \pm e_x) \cdot (y \pm e_y) = x \cdot y \pm y \cdot e_x \pm x \cdot e_y + e_x \cdot e_y \approx x \cdot y \pm y \cdot e_x \pm x \cdot e_y;$$

- относительная погрешность

$$\delta_z = \frac{|y| \cdot e_x + |x| \cdot e_y}{|x \cdot y|} = \frac{e_x}{|x|} + \frac{e_y}{|y|} = \delta_x + \delta_y.$$

❒ Погрешность деления чисел $x \pm e_x$, $y \pm e_y$:

- абсолютная погрешность

$$z = \frac{x \pm e_x}{y \pm e_y} = \frac{(x \pm e_x) \cdot (y \pm e_y)}{(y \pm e_y) \cdot (y \pm e_y)} \approx \frac{x}{y} \pm \frac{y \cdot e_x + x \cdot e_y}{y^2};$$

- относительная погрешность

$$\delta_z = \frac{|y| \cdot e_x + |x| \cdot e_y}{\left|\frac{x}{y}\right|} = \frac{|y| \cdot e_x + |x| \cdot e_y}{y^2 \cdot \left|\frac{x}{y}\right|} = \frac{e_x}{|x|} + \frac{e_y}{|y|} = \delta_x + \delta_y.$$

❒ Погрешность функции, зависящей от одной переменной:

- абсолютная погрешность

$$f(x \pm e_x) \approx f(x) \pm f'(x) \cdot e_x,$$

$$\Delta f = f(x \pm e_x) - f(x) = |f'(x)| e_x;$$

- относительная погрешность

$$\left|\frac{\Delta f}{f}\right| = \frac{|f'(x)|}{|f(x)|} e_x.$$

Аналогично получают формулы для оценки абсолютной и относительной погрешностей для функций, зависящих от *n* переменных.

1.4. Понятия о погрешности машинных вычислений

Для представления чисел в памяти компьютера применяют два способа.

❒ С фиксированной запятой[2].

❒ С плавающей запятой.

Пусть в основу запоминающего устройства положены однотипные физические устройства, имеющие *r* устойчивых состояний, называемых регистрами. Причем каждому устройству ставится в соответствие одинаковое количество *k* регистров и, кроме того, с помощью регистров может фиксироваться знак. Упорядоченные элементы образуют разрядную сетку машинного слова: в каждом разряде может быть записано одно из базисных чисел 0,1,..., *r* – 1 (одна из *r* "цифр" *r*-ой системы счисления) и в специаль-

[2] Здесь в обозначении устройств хранения информации мы используем устоявшуюся в отечественной литературе терминологию, в которой в качестве разделительного знака между целой и дробными частями числа используется запятая.

ном разряде отображен знак "+" или "–". При записи чисел с фиксированной запятой кроме упомянутых r параметров (основания системы счисления) и k (количество разрядов, отводимых под запись числа) указывается еще общее количество l разрядов, выделяемых под дробную часть числа. Таким образом, положительное вещественное число a, представляющее собой в r-ой системе бесконечную, непериодическую дробь, отображается конечной последовательностью

$$\alpha_1\alpha_2 \ldots \alpha_{k-l}\alpha_{k-l+1} \ldots \alpha_{k-1}\alpha_k ,$$

где $\alpha_i \in \{0;1;\ldots;r-1\}$, т. е.

$$a^* \approx \alpha_1 r^{k-l-1} + \alpha_2 r^{k-l-2} + \ldots + \alpha_{k-l} r^0 + \alpha_{k-l+1} r^{-1} + \ldots + \alpha_{k-1} r^{-(l-1)} + \alpha_k r^{-l}$$

Диапазон представляемых таким способом чисел определяется числами с наибольшими цифрами во всех разрядах, т. е. наименьшим числом $-(r-1)(r-1)\ldots(r-1)$ и наибольшим $(r-1)(r-1)\ldots(r-1)$, а абсолютная точность представления — есть оценка величины $\left|a - a^*\right|$, зависящая от способа округления. Абсолютная точность представления вещественных чисел с фиксированной запятой одинакова в любой части диапазона. В то же время относительная точность представления числа может значительно различаться в зависимости от того, берется ли a близким к нулю или к границе диапазона.

Пример 1.5. Рассмотрим запоминающее устройство с фиксированной запятой, состоящее из $k = 7$ элементов и имеющее $r = 10$ ($r = 0, 1,..., 9$). Будем считать, что общее количество разрядов, выделяемых под дробную часть, $l = 3$, под знак — один разряд, под целую часть — три разряда (рис. 1.2).

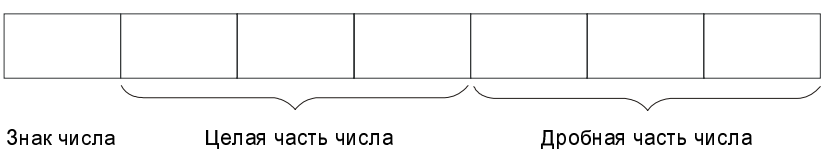


Рис. 1.2. Схема машинного слова запоминающего устройства с фиксированной запятой

Тогда наибольшее число, которое можно сохранить в данном запоминающем устройстве, равно 999.999, а наименьшее равно –999.999. У любого числа из указанного диапазона, являющегося бесконечной периодической дробью, вне зависимости от его величины после запятой сохраняется только 3 цифры. Поэтому абсолютная точность представления чисел $a_1 = 1.123456$ и $a_2 = 999.123456$ оказывается одинаковой.

$$\Delta_1 = 1.123456 - 1.123 = 0.000456,$$

$$\Delta_2 = 999.123456 - 999.123 = 0.000456.$$

Относительные погрешности представления этих чисел в запоминающем устройстве будут различны:

$$\delta_1 = \frac{\Delta_1}{a_1} \approx 0.04\ \%,$$

$$\delta_2 = \frac{\Delta_2}{a_2} = 0.5 \cdot 10^{-4}\ \%.$$

В основе часто употребляемого представления **с плавающей запятой** лежит экспоненциальная форма записи числа:

$$a = M \cdot r^p,$$

где r — основание, p — порядок, M — мантисса ($r^{-1} \le |M| \le 1$). Если под мантиссу выделяется l r-ичных элементов, а под порядок — m, то в системе записи с плавающей запятой вещественное число a представляется конечным числом

$$a^* \approx \pm\left(\beta_1 r^{-1} + \beta_2 r^{-2} + \ldots + \beta_l r^{-l}\right) \cdot r^{\gamma},$$

где γ — целое число из промежутка $\left[-r^m, r^m - 1\right]$; $\beta_1 \in \{1;\ldots;r-1\}$, $i = 2,\ldots,l$.

Структура машинного слова запоминающего устройства с плавающей запятой представлена на рис. 1.3.

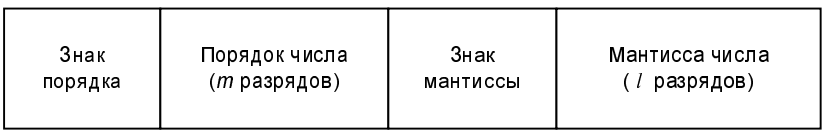


Рис. 1.3. Схема машинного слова запоминающего устройства с плавающей запятой

Числа $\pm r^{r^m}$ определяют границы допустимого числового диапазона.

Относительная точность представления вещественных чисел равна

$$\left|\frac{a - a^*}{a}\right| = \frac{\beta_{l+1} r^{-(l+1)} + \beta_{l+1} r^{-(l+2)} + \ldots}{\beta_1 r^{-1} + \beta_2 r^{-2} + \ldots} \le \frac{r^{-l}}{\beta_1 \cdot r^{-1}} \le r^{1-l},$$

т. е. относительная точность одинакова в любой части числового диапазона и зависит лишь от числа разрядов, отводимых под мантиссу числа.

Например, для записи числа в 48-разрядном машинном слове БЭСМ-6 40 двоичных разрядов выделялись под мантиссу, 6 — под порядок числа и 2 — под знаки мантиссы, т. е. $r = 2$, $l = 40$, $m = 6$. Следовательно, точность представления чисел с плавающей запятой не хуже 2^{-39} ($\approx 10^{-12}$), граница машинного нуля 2^{-64} ($\approx 10^{-19}$), машинной бесконечности 2^{63}.

Пример 1.6. Рассмотрим запоминающее устройство, состоящее из $k = 8$ элементов и имеющее $r = 10$ ($r = 0, 1,..., 9$). Будем считать, что общее количество разрядов, выделяемых под дробную часть $l = 5$, под знак порядка числа — 1 ячейка, под знак мантиссы числа — 1 ячейка, под порядок — 2 ячейки.

Оценим точность представления чисел $a_1 = 1.123123$ и $a_2 = 999.123123$.

Запишем числа в форме представления с плавающей запятой:

$$a_1 = 0.1123123 \cdot 10^1,$$

$$a_2 = 0.999123123 \cdot 10^3.$$

В запоминающем устройстве числа a_1, a_2 будут записаны в виде, показанном на рис. 1.4.

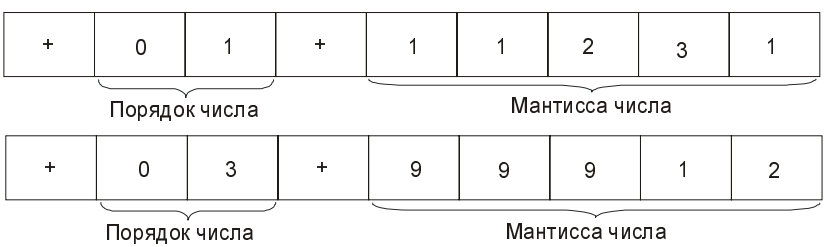


Рис. 1.4. Представление чисел a_1, a_2 с плавающей запятой в запоминающем устройстве

Абсолютные погрешности чисел равны:

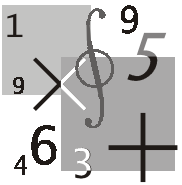
$$\Delta_1 = (0.1123123 - 0.11231) \cdot 10^1 \approx \left(0.23 \cdot 10^{-5}\right) \cdot 10^1$$

$$\Delta_2 = (0.999123123 - 0.99912) \cdot 10^3 \approx \left(0.31 \cdot 10^{-5}\right) \cdot 10^3.$$

Относительные погрешности чисел равны:

$$\delta_1 = \frac{\Delta_1}{a_1} = \frac{\left(0.23 \cdot 10^{-5}\right) \cdot 10^1}{0.1123 \cdot 10^1} \approx 2 \cdot 10^{-3}\,\%,$$

$$\delta_2 = \frac{\Delta_2}{a_2} = \frac{\left(0.31 \cdot 10^{-5}\right) \cdot 10^3}{0.99912 \cdot 10^3} \approx 3 \cdot 10^{-3}\,\%.$$



Лекция 2

Решение уравнений с одной переменной

Цель лекции: на примере методов половинного деления и простой итерации продемонстрировать основные подходы к решению задачи нахождения корней нелинейных уравнений, описать соответствующие алгоритмы и их программные реализации, обсудить использование соответствующих функций пакета MATLAB.

2.1. Общие сведения и основные определения

Наиболее общий вид нелинейного уравнения:

$$F(x) = 0\,, \tag{2.1}$$

где функция $F(x)$ определена и непрерывна на конечном или бесконечном интервале $[a, b]$.

Определение 2.1. Всякое число $\xi \in [a,b]$, обращающее функцию $F(x)$ в нуль, называется корнем уравнения (2.1).

Определение 2.2. Число ξ называется корнем k-ой кратности, если при $x = \xi$ вместе с функцией $F(x)$ равны нулю ее производные до $(k–1)$-го порядка включительно:

$$F(\xi) = F'(\xi) = \ldots = F^{(k-1)}(\xi) = 0. \tag{2.2}$$

Определение 2.3. Однократный корень называется простым.

Определение 2.4. Уравнения $F(x)=0$ и $G(x)=0$ называются равносильными (эквивалентными), если множества решений данных уравнений совпадают.

Нелинейные уравнения с одной переменной подразделяются на *алгебраические* и *трансцендентные*.

Определение 2.5. Уравнение (2.1) называется алгебраическим, если функция $F(x)$ является алгебраической.

Путем алгебраических преобразований из всякого алгебраического уравнения можно получить уравнение в канонической форме:

$$P_n(x) = a_0 x^n + a_1 x^{n-1} + \dots a_n, \qquad (2.3)$$

где a_0, a_1,..., a_n — действительные коэффициенты уравнения, x — неизвестное.

Из алгебры известно, что всякое алгебраическое уравнение имеет, по крайней мере, один вещественный или два комплексно сопряженных корня.

Определение 2.6. Уравнение (2.1) называется трансцендентным, если функция $F(x)$ не является алгебраической.

Определение 2.7. Решить уравнение (2.1) означает:

1. Установить, имеет ли уравнение корни.
2. Определить число корней уравнения.
3. Найти значения корней уравнения с заданной точностью.

2.2. Отделение корней

Определение 2.8. Отделение корней — процедура нахождения отрезков, на которых уравнение (2.1) имеет только одно решение.

В большинстве случаев отделение корней можно провести графически. Для этого достаточно построить график функции $F(x)$ и определить отрезки, на которых эта функция имеет только одну точку пересечения с осью абсцисс.

В сомнительных случаях графическое отделение корней необходимо подкреплять вычислениями. При этом можно использовать следующие очевидные положения:

- если непрерывная функция принимает на концах отрезка $[a, b]$ значения разных знаков (т. е. $F(a) \cdot F(b) < 0$), то уравнение (2.1) имеет на этом отрезке по меньшей мере один корень;
- если функция $F(x)$ к тому же и строго монотонна, то корень на отрезке единственный.

2.3. Метод половинного деления

Пусть уравнение (2.1) имеет на отрезке $[a, b]$ единственный корень, причем функция $F(x)$ на данном отрезке непрерывна (рис. 2.1).

Разделим отрезок [*a*, *b*] пополам точкой $c=(a+b)/2$. Если $F(c)\neq 0$, то возможны два случая:

1. Функция $F(x)$ меняет знак на отрезке [*a*, *c*].
2. Функция $F(x)$ меняет знак на отрезке [*c*, *b*].

Выбирая в каждом случае тот отрезок, на котором функция меняет знак, и продолжая процесс половинного деления дальше, можно дойти до сколь угодно малого отрезка, содержащего корень уравнения.

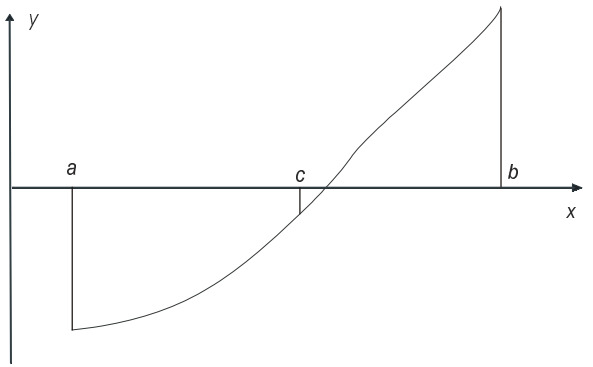


Рис. 2.1. К объяснению метода половинного деления

Пример 2.1. Найти, используя пакет MATLAB, методом половинного деления корень уравнения $x^4-11x^3+x^2+x+0.1=0$.

1. Создайте файл Func.m (листинг 2.1), содержащий описание функции $f(x)=x^4-11x^3+x^2+x+0.1$.

Листинг 2.1. Файл Func.m

```
function z=Func(x)
z=x.^4-11*x.^3+x.^2+x+0.1;
```

2. Создайте файл Div2.m (листинг 2.2), содержащий описание функции, возвращающей значение корня уравнения $f(x)=0$ методом половинного деления.

Листинг 2.2. Файл Div2.m

```
function z=Div2(f,x1,x2,eps);
% f - имя m-файла, содержащего описание функции f(x)
% x1 – левая граница отрезка, на котором производится поиск решения
% уравнения
% x2 – правая граница отрезка, на котором производится поиск решения
```

```
% уравнения
% eps – точность решения
L=x2-x1;
while L>eps
   c=(x2+x1)/2;
   if feval(f,c)*feval(f,x1)<0
   % feval(f,c) – оператор вычисления в точке x=c значения функции,
   % описание которой находится в соответствующем файле.
   % Имя файла хранится в строковой переменной f
      x2=c;
   else
      x1=c;
   end;
   L=x2-x1;
end;
z=c;
```

Примечание

Обращаем внимание читателя, что в пакете MATLAB существует "проблема буквы "я", суть которой состоит в том, что код строчной буквы "я" русского алфавита рассматривается пакетом MATLAB, как код команды. Поэтому обнаружение прописной буквы "я" в комментарии приводит к останову выполняемого файла и появлению сообщения об ошибке этапа исполнения. Для предотвращения описанной ситуации следует использовать в комментариях только строчную букву "Я".

3. Постройте график функции на интервале [–1, 1] (рис. 2.2), выполнив в командном окне пакета MATLAB следующую последовательность операторов:

```
>> x1=-1;
>> x2=1;
>> dx=10^-3;
>> x=x1:dx:x2;
>> plot(x,Func(x)); grid on
```

4. Вычислите значения корня уравнения:

```
>> Div2('Func',x1,x2,10^-5)
ans =
    0.3942
```

5. Проверьте полученное значение корня:

```
>> Func(ans)
ans =
  7.4926e-006
```

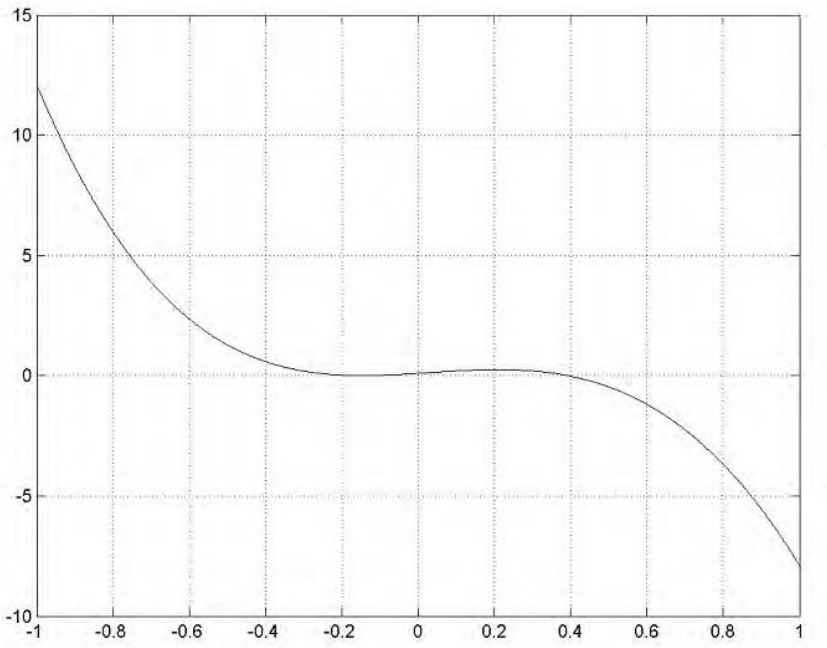


Рис. 2.2. График функции $f(x) = x^4 - 11x^3 + x^2 + x + 0.1$

Для рассмотрения процесса нахождения корня уравнения в динамике, необходимо сохранить значение корня на каждом шаге вычислительной процедуры и построить зависимость значения корня от номера шага. Далее приведен листинг файла Div2I.m, содержащего описание функции, возвращающей значение корня и длины отрезка (на котором данный корень находится) на каждом шаге метода половинного деления.

Листинг 2.3. Файл Div2I.m

```
function [z1,z2]=Div2(f,x1,x2,eps);
k=1;
L(1)=x2-x1; % начальная длина отрезка
c(1)=(x2+x1)/2; % начальное значение корня
while L(k)>eps
    if feval(f,c(k))*feval(f,x1)<0
      x2=c(k);
    else
      x1=c(k);
    end;
    k=k+1;
    c(k)=(x2+x1)/2;
```

```
    L(k)=x2-x1;
end;
z1=c;
z2=L;
```

6. На каждом шаге итерационного процесса вычислите значения корня и длины отрезка, на котором производится поиск решения.

```
>> [c L]=Div2I('Func',x1,x2,10^-5);
```

7. Визуализируйте зависимость значения корня от номера итерационного процесса (рис. 2.3).

```
>> plot(c, '-o')
```

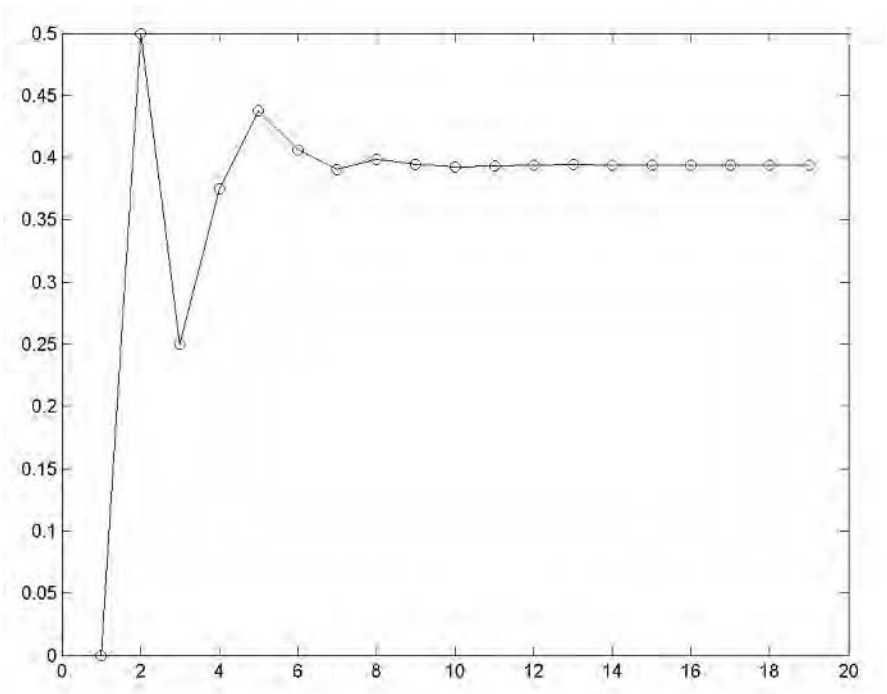


Рис. 2.3. Зависимость значения корня от номера шага вычислительной процедуры

8. Визуализируйте зависимость длины отрезка, на котором ищется значение корня, от номера итерации (рис. 2.4).

```
>> plot(L, '-o')
```

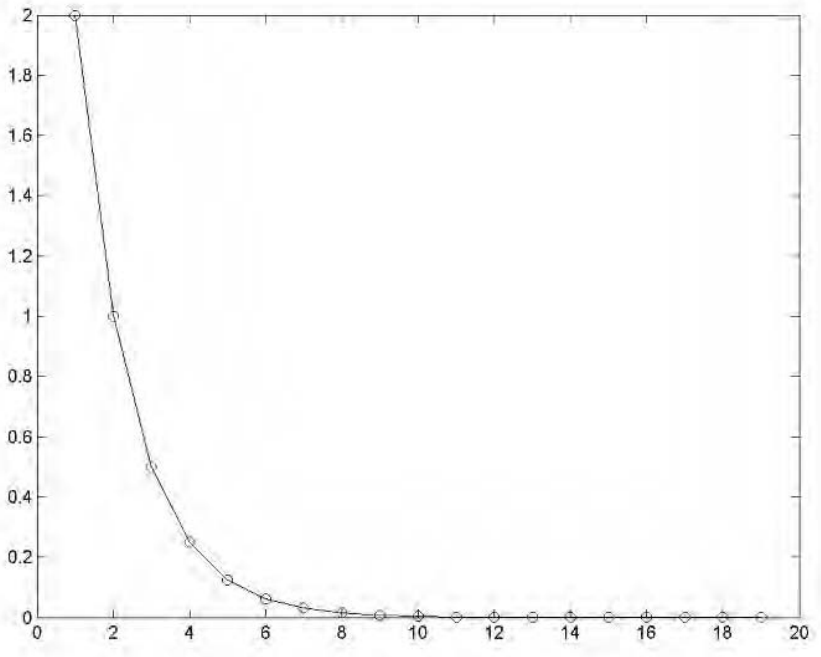


Рис. 2.4. Зависимость длины отрезка, на котором ищется значение корня, от номера шага вычислительной процедуры

2.4. Метод простой итерации

Заменим уравнение (2.1) равносильным уравнением:

$$x = f(x) \tag{2.4}$$

Пусть ξ — корень уравнения (2.4), а x_0 — полученное каким-либо способом нулевое приближение к корню ξ. Подставляя x_0 в правую часть уравнения (2.4), получим некоторое число $x_1 = f(x_0)$. Повторим данную процедуру с x_1 и получим $x_2 = f(x_1)$. Повторяя описанную процедуру, получим последовательность

$$x_0, x_1, \ldots, x_n, \ldots, \tag{2.5}$$

называемую итерационной последовательностью.

Геометрическая интерпретация данного алгоритма представлена на рис. 2.5.

Итерационная последовательность, вообще говоря, может быть как сходящейся, так и расходящейся, что определяется видом функции $f(x)$.

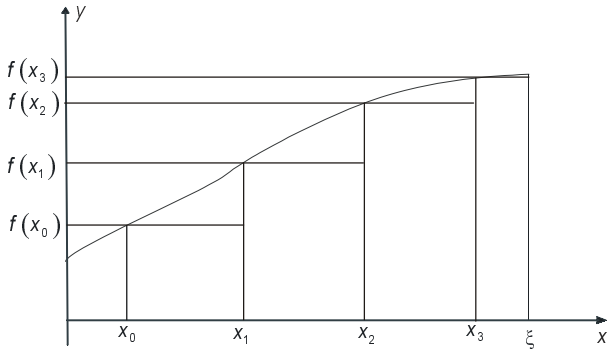


Рис. 2.5. К объяснению метода простой итерации

Теорема 2.1. Если функция $f(x)$ непрерывна, а последовательность (2.5) сходится, то предел последовательности (2.5) является корнем уравнения (2.4).

Действительно, пусть $\xi = \lim\limits_{n\to\infty} x_n$. Перейдем к пределу в равенстве $x_n = f(x_{n-1})$:

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} f(x_{n-1}) = f\left(\lim_{n\to\infty} x_{n-1}\right) = f(\xi). \qquad (2.6)$$

Условие сходимости итерационного процесса определяется следующей теоремой.

Теорема 2.2. Достаточное условие сходимости итерационного процесса.

Пусть уравнение $x = f(x)$ имеет единственный корень на отрезке $[a,b]$ и выполнены условия:

1. $f(x)$ определена и дифференцируема на $[a,b]$.
2. $f(x) \in [a,b]$ для всех $x \in [a,b]$.
3. Существует такое вещественное q, что $|f'(x)| \le q < 1$ для всех $x \in [a,b]$.

Тогда итерационная последовательность $x_n = f(x_{n-1})$ (n = 1, 2,) сходится при любом начальном приближении $x_0 \in [a,b]$.

Доказательство. Построим итерационную последовательность вида (2.5) с любым начальным значением $x_0 \in [a,b]$. В силу условия 2 теоремы 2.2 все члены последовательности находятся в отрезке $[a,b]$.

Рассмотрим два последовательных приближения $x_n = f(x_{n-1})$ и $x_{n+1} = f(x_n)$. По теореме Лагранжа о конечных приращениях имеем:

$$x_{n+1} - x_n = f(x_n) - f(x_{n-1}) = f'(c)(x_n - x_{n-1}), c \in [x_{n-1}, x_n].$$

Переходя к модулям и принимая во внимание условие 3 теоремы 2.2, получим:

$$|x_{n+1} - x_n| = |f'(c)| \cdot |x_n - x_{n-1}| \le q|x_n - x_{n-1}|,$$

$$|x_{n+1} - x_n| \le q|x_n - x_{n-1}| .$$

При $n = 1, 2,$ имеем:

$$|x_2 - x_1| \le q|x_1 - x_0|,$$

$$|x_3 - x_2| \le q \cdot |x_2 - x_1| \le q^2|x_1 - x_0|, \qquad (2.7)$$

$$|x_{n+1} - x_n| \le q^n|x_1 - x_0| .$$

Рассмотрим ряд

$$x_0 + (x_1 - x_0) + (x_2 - x_1) + \ldots (x_n - x_{n-1}) + \ldots \qquad (2.8)$$

Составим частичные суммы этого ряда

$$S_1 = x_0, S_2 = x_1, \ldots, S_{n+1} = x_n .$$

Заметим, что (n+1)-я частичная сумма ряда (2.8) совпадает с n-ым членом итерационной последовательности (2.5), т. е.

$$S_{n+1} = x_n . \qquad (2.9)$$

Сравним ряд (2.8) с рядом

$$|x_1 - x_0| + q|x_1 - x_0| + q^2|x_1 - x_0| + \ldots \qquad (2.10)$$

Заметим, что в силу соотношения (2.7) абсолютные величины членов ряда (2.8) не превосходят соответствующих членов ряда (2.10). Но ряд (2.10) сходится как бесконечно убывающая геометрическая прогрессия (q<1, по условию). Следовательно, и ряд (2.8) сходится, т. е. его частичная сумма (2.9) имеет предел. Пусть $\xi = \lim\limits_{n\to\infty} x_n$. В силу непрерывности функции f получаем (см. (2.6)):

$$\xi = f(\xi)$$

т.е. ξ — корень уравнения $x = f(x)$.

Отметим, что условия теоремы не являются необходимыми. Это означает, что итерационная последовательность может оказаться сходящейся и при невыполнении этих условий.

Отыщем погрешность корня уравнения, найденного методом простой итерации. Пусть x_n — приближение к истинному значению корня уравнения $x = f(x)$. Абсолютная ошибка приближения x_n оценивается модулем

$$\Delta x_n = |\xi - x_n| .$$

Принимая во внимание (2.8) и (2.9), имеем

$$\xi - x_n = \xi - S_{n+1} = (x_{n+1} - x_n) + (x_{n+2} - x_{n+1}) + \ldots \tag{2.11}$$

Сравним (2.11) с остатком ряда (2.9):

$$q^n|x_1 - x_0| + q^{n+1}|x_1 - x_0| + \ldots \tag{2.12}$$

Учитывая оценку (2.7), получаем

$$|\xi - x_n| \le q^n |x_1 - x_0| + q^{n+1}|x_1 - x_0| + \ldots = \frac{q^n}{1-q}|x_1 - x_0|.$$

Таким образом, для оценки погрешности n-го приближения получается формула

$$\Delta x_n \le \frac{q^n}{1-q}|x_1 - x_0|. \tag{2.13}$$

На практике удобнее использовать модификацию формулы (2.13).

Примем за нулевое приближение x_{n-1} (вместо x_0). Следующим приближением будет x_n (вместо x_1). Так как $|x_n - x_{n-1}| \le q^{n-1}|x_1 - x_0|$, то

$$\Delta x_n \le \frac{q}{1-q}|x_n - x_{n-1}|. \tag{2.14}$$

При заданной точности ответа ε итерационный процесс прекращается, если $\Delta x_n \le \varepsilon$.

2.5. Преобразование уравнения к итерационному виду

Уравнение $F(x) = 0$ преобразуется к виду, пригодному для итерационного процесса, следующим образом:

$$x = x - mF(x),$$

где m — отличная от нуля константа.

В этом случае

$$f(x) = x - mF(x). \tag{2.15}$$

Функция $f(x)$ должна удовлетворять условиям теоремы 2.2. Дифференцируя (2.15), получим

$$f'(x) = 1 - mF(x). \tag{2.16}$$

Для выполнения условия 3 теоремы 2.2 достаточно подобрать m, так чтобы для всех $x \in [a,b]$

$$|1 - mF'(x)| \le 1. \qquad (2.17)$$

Пример 2.2. Найти решение уравнения $x^4 - 11x^3 + x^2 + x + 0.1 = 0$ методом простой итерации, используя пакет MATLAB.

3. Создайте файл Func.m (листинг 2.4), содержащий описание функции $f(x) = x^4 - 11x^3 + x^2 + x + 0.1$.

Листинг 2.4. Файл Func.m

```
function z=Func(x)
z=x.^4-11*x.^3+x.^2+x+0.1;
```

4. Создайте файл Func1.m (листинг 2.5), содержащий описание функции $f1(x, m, f) := x - m \cdot f(x)$.

Листинг 2.5. Файл Func1.m

```
function z=Func1(x,m,f)
z=x-m*feval(f,x);
```

5. Создайте файл Func2.m (листинг 2.6), содержащий описание функции $f2$ (2.16).

Листинг 2.6. Файл Func2.m

```
function z=Func2(x,m,f)
dx=10^-7;
x1=x+dx;
tmp1=x-m*feval(f,x);
tmp2=x1-m*feval(f,x1);
z=abs((tmp2-tmp1)/dx);
```

6. Постройте графики функций $f1, f2$ (рис. 2.6).

```
>> dx=10^-3;
>> x1=-0.1;x2=0.8;
>> x=x1:dx:x2;
>> m=-0.05;
>> plot(x,Func1(x,m,'Func'));
>> hold on
>> plot(x,Func2(x,m,'Func'),'--');
>> grid on
```

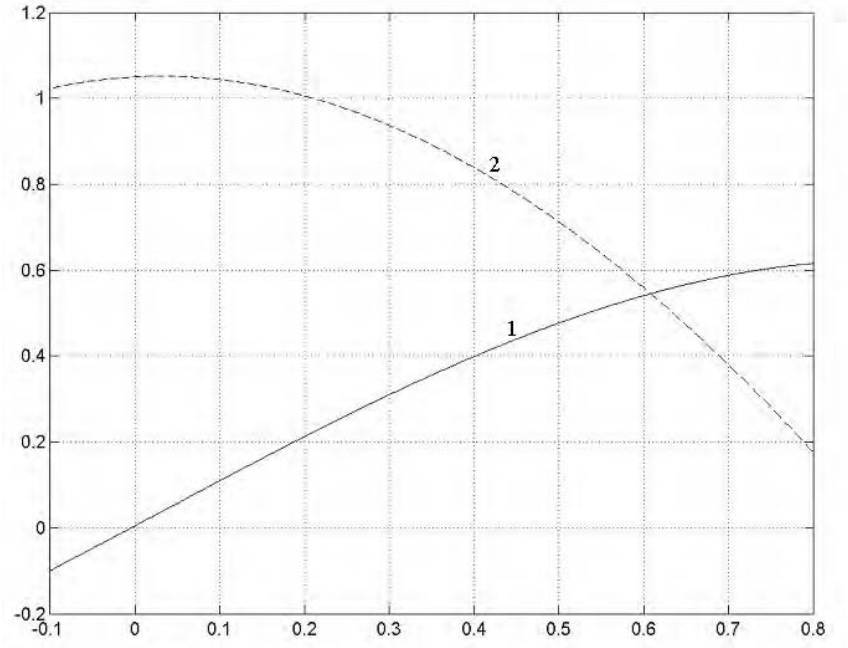


Рис. 2.6. Графики функций $f(x) = x - mF(x)$ — **1** и $f'(x) = 1 - mF(x)$ — **2**

Из рис. 2.6 видно, что на интервале [0.21; 0.8] функция удовлетворяет условиям теоремы 2.2.

7. Создайте файл My_Iter.m (листинг 2.7), который описывает функцию, возвращающую значение производной на каждом шаге итерационного процесса.

Листинг 2.7. Функция My_Iter.m

```
function z=My_Iter(f,x0,eps,q,m)
x(1)=x0;
i=1;
while abs(x(i)-Func1(x(i),m,f))>q/(1-q)*eps
   x(i+1)=Func1(x(i),m,f);
   i=i+1;
end;
z=x;
```

8. Задайте параметры итерационного процесса:

```
>> q=0.01;
>> eps=10^-5;
```

9. Вычислите значения корня уравнения на каждом шаге итерационного процесса.

```
>> z=My_Iter('Func',x0,eps,q,m)
```

10. Визуализируйте итерационный процесс (рис. 2.7).

```
>> plot(z,'-o');
```

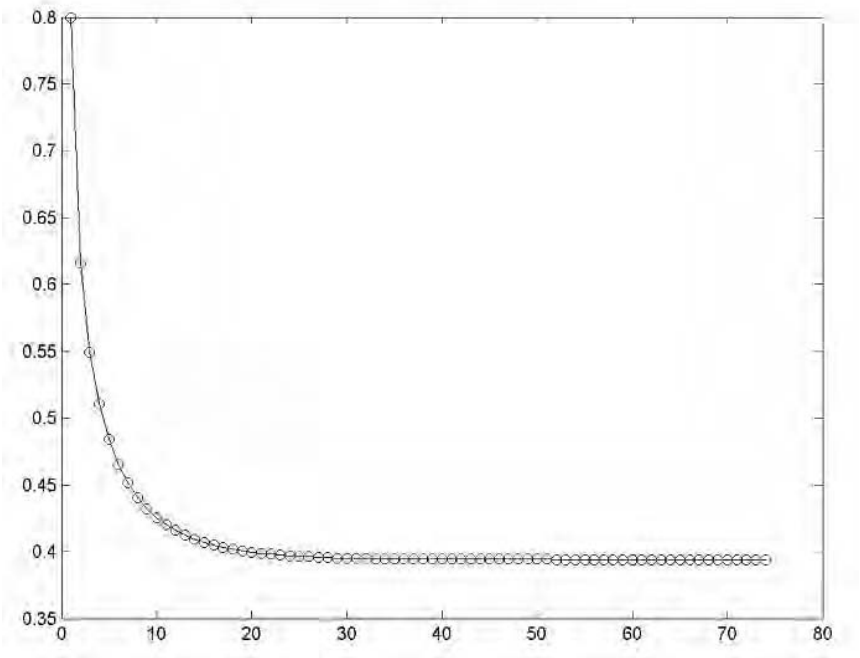


Рис. 2.7. Зависимость значения корня уравнения от номера шага итерационного процесса

11. Выведите точное значение корня.

```
>> Ni=length(z);
>> z(Ni)
ans =
    0.3942
```

12. Выведите значения функции.

```
>> Func(z(Ni))
ans =
 -1.8185e-006
```

Для вычисления нулей функций, зависящих от одной перемененной, в пакете MATLAB предусмотрена специальная функция **`fzero()`**, реализующая

в зависимости от вида функции методы половинного деления, секущих или обратной квадратичной интерполяции. Обращение к данной функции имеет следующий вид:

```
x = fzero(fun,x0,options,P1,P2,...)
[x,fval,exitflag,output] = fzero(...)
```

Здесь:

`fun` — строковая переменная, содержащая имя файла.

`x0` — начальное приближение или интервал поиска решения.

`options` — параметры, задающие точность и способ представления результатов вычислений.

`P1, P2, …` — дополнительные аргументы, передаваемые в функцию fun (например, `F=feval(FUN,X,P1,P2,...)`).

`fval` — переменная, в которую функция `fzero()` возвращает значение корня уравнения $f(x) = 0$.

`exitflag` — переменная, знак которой свидетельствует о наличии корня на данном интервале (например, `exitflag=1` — корень существует);

`output` — переменная, в которую функция `fzero()` возвращает название метода, использованного для нахождения корня уравнения.

Пример 2.3. Решение уравнения $x^4 - 11x^3 + x^2 + x + 0.1 = 0$ с использованием функции `fzero()`.

```
>> x=fzero('Func',0.8)
x =
    0.3942
```

Поиск корня на отрезке [−2, 2]:

```
>> x=fzero('Func',[-2,2])
x =
    0.3942
```

Поиск корня с точностью до 10^{-2}, вывод на экран значение корня и соответствующего значения функции на каждом шаге итерационного процесса:

```
>> x=fzero('Func',0.8,optimset('TolX',10^-2,'disp','iter'))
 Func-count      x          f(x)         Procedure
    1            0.8       -3.6824        initial
    2         0.777373     -3.32063       search
    3         0.822627     -4.06625       search
    4           0.768      -3.17712       search
    5           0.832      -4.23184       search
    6         0.754745     -2.98039       search
    7         0.845255     -4.47271       search
```

```
    8         0.736         -2.71444         search
    9         0.864         -4.82695         search
   10         0.70949       -2.36229         search
   11         0.89051       -5.35561         search
   12         0.672         -1.9106          search
   13         0.928         -6.16014         search
   14         0.618981      -1.35979         search
   15         0.981019      -7.41582         search
   16         0.544         -0.743367        search
   17         1.056         -9.43876         search
   18         0.437961      -0.157497        search
   19         1.16204       -12.8248         search
   20         0.288          0.215057        search
    Looking for a zero in the interval [0.288, 1.162]
   21            0.308       0.190464         interpolation
   22         0.460543      -0.256865         interpolation
   23          0.37295      0.0607713         interpolation
   24         0.397562      -0.010605         interpolation
Zero found in the interval: [0.288, 1.162].
x =
    0.3976
```

Поиск корня с точностью до 10^{-5}, вывод на экран значение корня и соответствующего значения функции на каждом шаге итерационного процесса:

```
>> [x fval]=fzero('Func',0.8,optimset('TolX',10^-5,'disp','iter'))

 Func-count      x           f(x)          Procedure
    1             0.8       -3.6824          initial
    2         0.777373      -3.32063         search
    3         0.822627      -4.06625         search
    4            0.768      -3.17712         search
    5            0.832      -4.23184         search
    6         0.754745      -2.98039         search
    7         0.845255      -4.47271         search
    8            0.736      -2.71444         search
    9            0.864      -4.82695         search
   10          0.70949      -2.36229         search
   11          0.89051      -5.35561         search
   12            0.672      -1.9106          search
   13            0.928      -6.16014         search
   14         0.618981      -1.35979         search
   15         0.981019      -7.41582         search
   16            0.544      -0.743367        search
   17            1.056      -9.43876         search
   18         0.437961      -0.157497        search
```

```
   19           1.16204    -12.8248        search
   20           0.288        0.215057      search

   Looking for a zero in the interval [0.288, 1.162]

   21        0.302415      0.198003         interpolation
   22        0.467237     -0.288821         interpolation
   23        0.369452     0.0698667         interpolation
   24        0.398884    -0.0148167         interpolation
   25        0.393735    0.00136079         interpolation
   26        0.394168  2.27096e-005         interpolation
   27        0.394188 -3.91718e-005         interpolation
Zero found in the interval: [0.288, 1.162].
x =
    0.3942
fval =
  2.2710e-005
```

Поиск корня с точностью до 10^{-3}, вывод на экран значение корня на последнем шаге итерационного процесса:

```
>> X = fzero('Func',3, optimset('disp','final'))
Zero found in the interval: [0.28471, 4.92].
X =
    0.3942
```

Для нахождения корней полинома в пакете MATLAB предусмотрена соответствующая функция **`roots()`**, возвращающая вектор-столбец, компоненты которого являются корнями полинома (действительными или комплексными).

Обращение к данной функции имеет следующий вид:

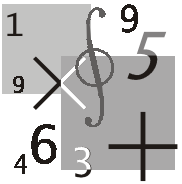
```
r=roots(c)
```

Здесь `c` — вектор-строка, содержащая значения коэффициентов полинома $c_1x^n + c_2x^{n-1} + \ldots + c_{n+1}x^0$.

Пример 2.4. Найти корень уравнения $x^4 - 11x^3 + x^2 + x + 0.1 = 0$, используя функцию `roots()`.

```
>> c=[1 -11 1 1 0.1];
>> roots(c)
ans =
  10.8998
   0.3942
  -0.1470 + 0.0409i
  -0.1470 - 0.0409i
```

Лекция 3



Методы решения систем линейных алгебраических уравнений

Цель лекции: рассмотреть прямые и итерационные методы решения систем линейных уравнений, описать соответствующие алгоритмы и их программные реализации, обсудить использование соответствующих функций пакета MATLAB.

3.1. Общие сведения и основные определения

Рассмотрим систему, состоящую из m линейных алгебраических уравнений с n неизвестными:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &= b_2, \\ \ldots & \\ a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n &= b_m, \end{aligned} \tag{3.1}$$

которая может быть записана в матричном виде

$$A \cdot x = b, \tag{3.2}$$

где A — прямоугольная матрица размерности $m \times n$:

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & . & . \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}, \tag{3.3}$$

x — вектор n-го порядка:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

b — вектор m-порядка:

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Определение 3.1. Решением системы (3.1) называется такая упорядоченная совокупность чисел $x_1 = c_1$, $x_2 = c_2$, ... , $x_n = c_n$, которая обращает все уравнения системы (3.1) в верные равенства.

Определение 3.2. Прямыми методами решения систем линейных уравнений называются методы, дающие решение системы за конечное число арифметических операций. Если отсутствуют ошибки округления, то получаемые решения *всегда* являются точными.

Определение 3.3. Итерационными методами решения систем линейных уравнений называются методы, дающие решение системы уравнений как предел последовательности приближений, вычисляемых по единообразной схеме.

3.2. Метод Гаусса и его реализация в пакете MATLAB

Рассмотрим систему линейных алгебраических уравнений

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \dots a_{2n}x_n &= b_2, \\ \dots & \\ a_{n1}x_1 + a_{n2}x_2 + \dots a_{nn}x_n &= b_n, \end{aligned} \tag{3.4}$$

при условии, что матрица $A = (a_{ij})$ невырождена.

Метод Гаусса состоит в преобразовании системы (3.4) последовательным исключением переменных к равносильной системе с треугольной матрицей

$$\begin{aligned} x_1 + \alpha_{12} x_2 + \ldots \alpha_{1n} x_n &= \beta_1, \\ x_2 + \ldots \alpha_{2n} x_n &= \beta_2, \\ \ldots \quad \ldots \quad \ldots & \\ x_n &= \beta_n. \end{aligned} \tag{3.5}$$

Затем из системы (3.5) последовательно находят значения всех неизвестных $x_n, x_{n-1}, \ldots x_1$.

Таким образом, процесс решения системы (3.4) распадается на два этапа:

1. Прямой ход — приведение системы (3.4) к треугольному виду.
2. Обратный ход — нахождение значений неизвестных переменных, в соответствие с (3.5).

Для реализации метода Гаусса в пакете MATLAB необходимо:

1. Создать файл Exchange.m (листинг 3.1), содержащий описание функции, осуществляющей перестановку строк при обнаружении в текущей строке нулевого элемента на главной диагонали.

Листинг 3.1. Файл Exchange.m

```
function z=Exchange(C,i)
k=i+1;
while C(k,i)==0
  k=k+1;
end;
for j=1:size(C,1)
  s=C(i,j);
  C(i,j)=C(k,j);
  C(k,j)=s;
end;
z=C;
```

2. Создать файл Simplex.m (листинг 3.2), содержащий описание функции, возвращающей расширенную матрицу системы к диагональному виду.

Листинг 3.2. Файл Simplex.m

```
function z=Simplex(A,b)
N=size(A,1); % определение числа уравнений системы
C=cat(2,A,b); % создание расширенной матрицы системы
for i=1:N-1
  if  C(i,i)==0
     C=Exchange(C,i);
  end;
```

```
  for j=0:N
     C(i,N+1-j)=C(i,N+1-j)/C(i,i);
  end;
  for m=i+1:N
     alpha=C(m,i);
     for j=i:N+1
        C(m,j)=C(m,j)-alpha*C(i,j);
     end;
  end;
end;
C(N,N+1)=C(N,N+1)/C(N,N);
C(N,N)=1;
z=C;
```

3. Создать файл Gauss.m (листинг 3.3), содержащий описание функции, возвращающей решение системы линейных уравнений методом Гаусса.

Листинг 3.3. Файл Gauss.m

```
function z=Gauss(A,b)
C=Simplex(A,b);
N=size(A,1);
v(N)=C(N,N+1);
for j=1:N-1
  s=0;
  for k=0:j-1
    s=s+C(N-j,N-k)*v(N-k);
  end;
  v(N-j)=(C(N-j,N+1)-s)/C(N-j,N-j);
end;
z=v';
```

4. Задать матрицу системы линейных уравнений:

```
>> A=(1,2,3,4,5;10,9,8,7,6;5,9,11,12,13;20,1,3,17,14;12,10,4,16,15)
A =
      1     2     3     4     5
     10     9     8     7     6
      5     9    11    12    13
     20     1     3    17    14
     12    10     4    16    15
```

5. Задать вектор-столбец свободных членов:

```
>> b=[10;20;30;40;50]
b =
    10.0000
```

```
    20.0000
    30.0000
    40.0000
    50.0000
```

6. Решить систему уравнений, используя функцию **Gauss()**:

```
>> x=Gauss(A,b)
x =
    0.0964
    1.4324
   -1.3530
    1.6593
    0.8921
```

7. Проверить правильность решения системы линейных уравнений:

```
>> A*x
ans =
   10.0000
   20.0000
   30.0000
   40.0000
   50.0000
```

3.3. Вычисление определителей

Решение системы (3.4) существует только в том случаем, если определитель матрицы A отличен от нуля, поэтому решение любой системы линейных уравнений следует предварять вычислением ее определителя.

Для вычисления определителя используют известное свойство треугольных матриц: определитель треугольной матрицы равен произведению ее диагональных элементов.

Пусть задана квадратная матрица X n-го порядка:

$$X = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}. \qquad (3.6)$$

Представим матрицу X в виде:

$$X = Y \cdot Z\,, \qquad (3.7)$$

где

$$Y = \begin{pmatrix} y_{11} & 0 & \cdots & 0 \\ y_{21} & y_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & z_{12} & \cdots & z_{1n} \\ 0 & 1 & \cdots & z_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}. \tag{3.8}$$

Известно, что определитель матрицы равен произведению определителей:

$$|X| = |Y| \cdot |Z|, \tag{3.9}$$

но $|Z| = 1$, поэтому

$$|X| = y_{11} \cdot y_{22} \cdot \ldots \cdot y_{nn}. \tag{3.10}$$

Формулы для вычисления элементов матриц Y и Z получаются перемножением этих матриц и приравниванием элементов матрицы Y к соответствующим элементам матрицы X:

$$y_{i1} = x_{i1};\ y_{ij} = x_{ij} - \sum_{k=1}^{i-1} y_{ik} \cdot z_{kj},\ \text{при}\ i \geq j \geq 2 \tag{3.11}$$

$$z_{1j} = \frac{x_{1j}}{y_{11}};\ y_{ij} = \left(x_{ij} - \sum_{k=1}^{j-1} y_{ik} \cdot z_{kj} \right) \Big/ y_{ii},\ \text{при}\ 1 < i \leq j. \tag{3.12}$$

Далее приводится листинг файла Determinant.m, содержащий описание функции, которая возвращает значение определителя матрицы, вычисляемого в соответствие с (3.11), (3.12).

Листинг 3.4. Файл Determinant.m

```
function Z=Determinant(A)
P=1;
C=0;
N=size(A,1);
y=zeros(N);
z=zeros(N);
for i=1:N
   y(i,1)=A(i,1);
   z(i,i)=1;
end;
for j=2:N
   z(1,j)=A(1,j)/y(1,1);
end;
for i=2:N
   for j=2:N
      if (j>=2)&(j<=i)
```

```
        s=0;
        for k=1:j-1
           s=s+y(i,k)*z(k,j);
        end;
        y(i,j)=A(i,j)-s;
      end;
      if (i>1)&(i<j)
        s=0;
        for k=1:i-1
           s=s+y(i,k)*z(k,j);
        end;
        z(i,j)=(A(i,j)-s)/y(i,i);
      end;
   end;
end;
s=1;
for i=1:N
   s=s*y(i,i);
end;
Z=s;
```

Для вычисления определителя квадратной матрицы A, созданной в предыдущем примере, следует ввести команду:

```
>> Determinant(A)
ans =
  7.0510e+003
```

Для проверки правильности работы данной функции полезно сравнить результат, возвращаемый функцией **`Determinant()`**, и результат, полученный функцией **`Det()`**, встроенной в пакет MATLAB:

```
>> det(A)
ans =
        7051
```

3.4. Решение систем линейных уравнений методом простой итерации

Запишем исходную систему (3.4) в следующем виде:

$$\begin{aligned} x_1 &= \alpha_{11}x_1 + \alpha_{12}x_2 + \ldots \alpha_{1n}x_n + \beta_1, \\ x_2 &= \alpha_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n + \beta_2, \\ &\ldots \\ x_n &= \alpha_{n1}x_1 + \alpha_{n2}x_2 + \ldots \alpha_{nn}x_n + \beta_n \end{aligned} \qquad (3.13)$$

или сокращенно

$$x_i = \sum_{j=1}^{n} \alpha_{ij} x_j + \beta_i \,, \tag{3.14}$$

где $i = 1,2,\ldots n$.

Правая часть (3.14) определяет отображение F

$$F : y_i = \sum_{j=1}^{n} \alpha_{ij} x_j + \beta_i, i = 1,2,\ldots n \,, \tag{3.15}$$

преобразующее точку $x = (x_1, x_2, \ldots, x_n)$ n-мерного векторного пространства в точку $y = (y_1, y_2, \ldots, y_n)$ того же пространства. Используя систему (3.4) и выбрав начальную точку $x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}\right)$, можно построить итерационную последовательность точек n-мерного пространства (аналогично методу простой итерации для скалярного уравнения $x = f(x)$):

$$x^{(0)}, x^{(1)}, \ldots, x^{(n)}, \ldots \tag{3.16}$$

Оказывается, что при определенных условиях последовательность (3.16) сходится и ее предел является решением системы (3.13). Предваряя обсуждение данных условий, напомним необходимые сведения из математического анализа.

Определение 3.4. Функцию $\rho(x, y)$, определяющую расстояние между точками x и y множества X, называют метрикой, если выполнены следующие условия:

1. $\rho(x, y) \geq 0$.
2. $\rho(x, y) = 0$ тогда и только тогда, когда $x = y$.
3. $\rho(x, y) = \rho(y, x)$.
4. $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$.

Определение 3.5. Множество с введенной в нем метрикой ρ становится *метрическим пространством*.

Определение 3.6. Последовательность точек метрического пространства называется *фундаментальной*, если для любого $\varepsilon > 0$ существует такое число $N(\varepsilon)$, что для всех m, $n > N$ выполняется неравенство $\rho(x_m, x_n) < \varepsilon$.

Определение 3.7. Пространство называется полным, если в нем любая фундаментальная последовательность сходится.

Пусть F — отображение, действующее в метрическом пространстве E с метрикой ρ, x и y — точки пространства E, а Fx, Fy — образы этих точек.

Определение 3.8. Отображение F пространства E в себя называется сжимающим отображением, если существует такое число $\alpha \in (0, 1)$, что для любых двух точек x, $y \in E$ выполняется неравенство

$$\rho(Fx, Fy) \le \alpha \cdot \rho(x, y). \tag{3.17}$$

Определение 3.9. Точка x называется неподвижной точкой отображения F, если $Fx = x$.

Аналогично одномерному случаю можно доказать теорему о достаточном условии сходимости итерационного процесса в n-мерном пространстве. (Доказательство теоремы приводится практически во всех учебниках, посвященных численным методам, например, [1].)

Теорема. 3.1. Если F — сжимающее отображение, определенное в полном метрическом пространстве, то существует единственная неподвижная точка x, такая что $x = Fx$. При этом итерационная последовательность, построенная для отображения F с любым начальным членом $x^{(0)}$, сходится к x.

В ходе доказательства данной теоремы показывается, что

$$\rho\left(x, x^{(k)}\right) \le \frac{\alpha^k}{1-\alpha} \rho\left(x^{(0)}, x^{(1)}\right). \tag{3.18}$$

Приняв k-ое приближение за нулевое, из (3.18) получаем полезное для приложений неравенство:

$$\rho\left(x, x^{(k)}\right) \le \frac{\alpha}{1-\alpha} \rho\left(x^{(k-1)}, x^{(k)}\right). \tag{3.19}$$

Рассмотрим условия, при которых отображение (3.15) будет сжимающим. Как следует из определения (3.17), решение данного вопроса зависит от способа метризации данного пространства. Обычно при решении систем линейных уравнений используют одну из следующих метрик:

$$\rho_1(x, y) = \max_{1 \le i \le n} |x_i - y_i|, \tag{3.20}$$

$$\rho_2(x, y) = \sum_{i=1}^{n} |x_i - y_i|, \tag{3.21}$$

$$\rho_3(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{3.22}$$

Для того чтобы отображение F, заданное в метрическом пространстве уравнениями (3.15), было сжимающим, достаточно выполнения одного из следующих условий:

1. В пространстве с метрикой ρ_1:

$$\alpha = \max_{1 \le i \le n} \sum_{j=1}^{n} |\alpha_{ij}| < 1 . \qquad (3.23)$$

2. В пространстве с метрикой ρ_2:

$$\alpha = \max_{1 \le j \le n} \sum_{i=1}^{n} |\alpha_{ij}| < 1 . \qquad (3.24)$$

3. В пространстве с метрикой ρ_3:

$$\alpha = \sqrt{\sum_{i}^{n} \sum_{j}^{n} \alpha_{ij}^2} < 1 . \qquad (3.25)$$

Для установления момента прекращения итераций при достижении заданной точности может быть использована, например, формула, следующая из (3.19):

$$\rho\left(x^{(k-1)}, x^{(k)}\right) \le \varepsilon(1-\alpha)/\alpha . \qquad (3.26)$$

Алгоритм решения системы методом итераций реализуется следующим образом:

1. Приведите систему (3.4) к виду с преобладающими диагональными коэффициентами.
2. Разделите каждое уравнение на соответствующий диагональный коэффициент.
3. Проверьте выполнение условий (3.23)—(3.25).
4. Выберите метрику, для которой выполняется условие сходимости итерационного процесса.
5. Реализуйте итерационный процесс (обычно за начальное приближение берется столбец свободных членов).

Пример 3.1. Преобразование системы уравнений к виду пригодному для итерационного процесса.

$$\begin{aligned} 2.34x_1 - 4.21x_2 - 11.61x_3 &= 14.41 \\ 8.04x_1 + 5.22x_2 + 0.27x_3 &= -6.44 \\ 3.92x_1 - 7.99x_2 + 8.37x_3 &= 55.56 \end{aligned}$$

1. Возьмите первым уравнением второе, третьим первое, а вторым — сумму первого и третьего уравнений:

$$8.04x_1 + 5.22x_2 + 0.27x_3 = -6.44$$
$$6.26x_1 - 12.20x_2 - 3.24x_3 = 69.97$$
$$2.34x_1 - 4.21x_2 - 11.61x_3 = 14.41$$

2. Разделите каждое уравнение на диагональный коэффициент и выразите из каждого уравнения диагональное неизвестное:

$$x_1 = -0.6492537x_2 - 0.033582x_3 - 0.800995$$
$$x_2 = 0.5131147x_1 - 0.2655737x_3 - 5.7352459$$
$$x_3 = 0.2015503x_1 - 0.3626184x_2 - 1.2411714$$

На этом приведение уравнения к виду, пригодному для использования итерационного процесса, завершается. Для дальнейших вычислений целесообразно использовать пакет MATLAB.

1. Создайте файлы Ro1.m, Ro2.m, Ro3.m (листинги 3.5–3.7), содержащие описания функций, возвращающих значение расстояния между точками в соответствующем метрическом пространстве.

Листинг 3.5. Файл Ro1.m

```
function z=Ro1(x,y)
z=max(abs(x-y));
```

Листинг 3.6. Файл Ro2.m

```
function z=Ro2(x,y)
z=sum(abs(x-y));
```

Листинг 3.7. Файл Ro3.m

```
function z=Ro3(x,y)
z=norm(x-y);
```

2. Создайте файл Check.m (листинг 3.8), содержащий описание функции, возвращающей значение коэффициентов α для каждого типа метрик.

Листинг 3.8. Файл Check.m

```
function z=Check(A)
alpha1=sum(abs(A),2);
z(1)=max(alpha1);
alpha2=sum(abs(A),1);
z(2)=max(alpha2);
```

```
alpha3=A.^2;
z(3)=sum(sum(alpha3))^0.5;
```

3. Создайте файл LS_Iter.m (листинг 3.9), содержащий описание функции, возвращающей решение системы линейных уравнений методом простой итерации.

Листинг 3.9. Файл LS_Iter.m

```
function z=LS_Iter(A,b,Ro,alpha,eps)
% аргументы функции:
% A – матрица приведенной системы уравнений
% b – вектор столбец свободных членов приведенной системы уравнений
% Ro – имя файла, содержащего функцию, возвращающую значение метрики
% alpha – коэффициент сжимающего отображения в пространстве с
% выбранной метрикой
% eps – переменная, определяющая точность численного решения
delta=eps*(1-alpha)/alpha;
N=size(A,1);
x0=zeros(N,1);
x1=A*x0+b;
Delta=feval(Ro,x0,x1);
while Delta>delta
  x0=x1;
  x1=A*x0+b;
  Delta=feval(Ro,x0,x1);
end;
z=x1;
```

4. Задайте матрицу системы, приведенной к виду, пригодному для метода простой итерации:

```
>>A=[0,-0.6492537,-0.033582;0.5131147,0,-0.2655737;0.2015503,-
0.3626184,0]
A =
         0   -0.6493   -0.0336
    0.5131         0   -0.2656
    0.2016   -0.3626         0
```

5. Задайте вектор-столбец свободных членов:

```
>> b=[-0.800995;-5.7352459;-1.2411714]
b =
   -0.8010
   -5.7352
   -1.2412
```

6. Вычислите значения коэффициентов α для каждого метрического пространства:

```
>> alpha=Check(A)
alpha =
    0.7787    1.0119    0.9636
```

7. Найдите решения системы линейных уравнений:

```
>> LS_Iter(A,b,'Ro1',ans(1),10^-6)
ans =
    2.2930
   -4.8155
    0.9672
```

3.5. Метод Зейделя

Напомним, что при решении системы линейных уравнений вычислительные формулы имеют вид:

$$y_i = \sum_{j=1}^{n} \alpha_{ij} x_j + \beta_i \,, \tag{3.27}$$

где $i = 1,2,\ldots n$.

Из (3.27) видно, что в методе простой итерации для получения нового значения вектора решений на $(i+1)$-ом шаге используются значения переменных, полученные на предыдущем шаге.

Основная идея метода Зейделя состоит в том, что на каждом шаге итерационного процесса при вычислении значения переменной y_i учитываются уже найденные значения $y_1, y_2, \ldots, y_{i-1}$:

$$\begin{aligned}
y_1 &= \sum_{j=1}^{n} \alpha_{1j} x_j + \beta_1 \,, \\
y_2 &= \alpha_{21} y_1 + \sum_{j=2}^{n} \alpha_{2j} x_j + \beta_2 \,, \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
y_i &= \sum_{j=1}^{i-1} \alpha_{ij} y_j + \sum_{j=i}^{n} \alpha_{ij} x_j + \beta_i \,, \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
y_n &= \sum_{j=1}^{n-1} \alpha_{nj} y_j + \alpha_{nn} x_n + \beta_n \,.
\end{aligned} \tag{3.28}$$

Достаточные условия сходимости итерационного процесса (3.23)—(3.25) также являются достаточным условиями сходимости метода Зейделя.

Существует возможность автоматического преобразования исходной системы к виду, обеспечивающему сходимость итерационного процесса метода Зейделя. Для этого умножим левую и правую части системы (3.2) на транспонированную матрицу системы A^T, получим равносильную систему

$$C \cdot X = D, \tag{3.29}$$

где $C = A^T A$, $D = A^T b$.

Система (3.29) называется нормальной системой уравнений. Нормальные системы уравнений обладают рядом свойств, среди которых можно выделить следующие:

- матрица C коэффициентов при неизвестных нормальной системы является симметрической (т. е. $a_{ij} = a_{ji}$, $i, j = 1, 2, \ldots, n$);
- все элементы, стоящие на главной диагонали матрицы C, положительны (т. е. $a_{ii} > 0$, $i = 1, 2, \ldots, n$).

Последнее свойство дает возможность "автоматически" приводить нормальную систему (3.29) к виду, пригодному для итерационного процесса Зейделя:

$$x_i = \sum_{i \neq j} \alpha_{ij} x_j + \beta_j, \left(i = 1, 2, \ldots, n\right), \tag{3.30}$$

где

$$\alpha_{ij} = -\frac{c_{ij}}{c_{ii}} (i \neq j), \tag{3.31}$$

и

$$\beta_i = \frac{d_i}{c_{ii}}. \tag{3.32}$$

Целесообразность приведения системы к нормальному виду и использования метода Зейделя вытекает из следующей теоремы:

Теорема 3.2. Итерационный процесс метода Зейделя для приведенной системы (3.30), эквивалентной нормальной системе (3.29), всегда сходится к единственному решению этой системы при любом выборе начального приближения [2].

Таким образом, решение произвольной системы линейных уравнений вида (3.1) методом Зейделя реализуется в соответствие со следующим алгоритмом:

1. Ввод матрицы A коэффициентов исходной системы и вектор-столбца свободных членов.
2. Приведение системы к нормальной умножением обеих частей системы на транспонированную матрицу A^{T}.

3. Приведение нормальной системы к виду, пригодному для итерационного процесса Зейделя (3.30), (3.31).
4. Задание требуемой точности решения.
5. Циклическое выполнение итерационного процесса до достижения требуемой точности.

Для реализации метода Зейделя в пакете MATLAB необходимо:

1. Создать файл Zeidel.m (листинг 3.10), содержащий описание функции, выполняющей последовательно: *а*) приведение системы к нормальному виду; *б*) приведение нормальной системы к виду, пригодному для итерационного процесса Зейделя; *в*) реализацию итерационного процесса Зейделя.

Листинг 3.10. Файл Zeidel.m

```
function [z1,z2]=Zeidel(A,b,eps)
N=size(A,1);
% приведение системы к нормальному виду
C=A'*A;
D=A'*b;

%приведение системы к виду, пригодному для итерационного процесса Зейделя
for i=1:N
   D1(i)=D(i)/C(i,i);
end;
D1=D1'; транспонирование матрицы
d1=D1;
for i=1:N
  for j=1:N
    if i==j
       C1(i,j)=0;
    else
       C1(i,j)=-C(i,j)/C(i,i);
    end;
  end;
end;
% решение системы линейных уравнений методом Зейделя
R1=d1;
while Flag==0
  for i=1:N
     v=C1(i,1:N); % выделение i-ой строки матрицы
     a=dot(v',d1); % вычисление скалярного произведения
     d1(i)=a+D1(i);
  end;
```

```
  R2=d1;
  s=max(abs(R2-R1));
  if s<eps
    z1=d1;
    z2=s;
    return
  end;
  R1=R2;
end;
```

2. Задать значения коэффициентов при неизвестных исходной системы линейных уравнений и столбец свободных членов:

```
>> A=[1,2,3,4,5;10,9,8,7,6;5,9,11,12,13;20,1,3,17,14;12,10,4,16,15]
>> b=[10,20,30,40,50];
```

3. Вычислить решение системы линейных уравнений, используя функцию **Zeidel()**:

```
>> Zeidel1(A,b,10^-6)
ans =
    0.0972
    1.4325
   -1.3533
    1.6564
    0.8947
```

4. Проверить полученное решение:

```
>> A*ans
ans =
   10.0013
   20.0003
   29.9995
   39.9999
   50.0000
```

3.6. Решение систем линейных уравнений средствами пакета MATLAB

Для решения систем линейных уравнений и связанных с ними матричных операций применяются операторы: сложения (+), вычитания (–), умножения (*), деления справа (/), деления слева (\), возведения в степень (^), транспонирования (′), действие которых определяется правилами линейной алгебры.

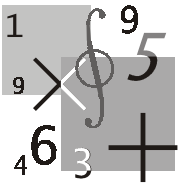
Для решения системы линейных уравнений вида

$$Ax = b\ ,$$

где A — матрица коэффициентов при неизвестных, x — вектор-столбец неизвестных, b — вектор-столбец свободных членов, в пакете MATLAB достаточно выполнить следующую команду:

```
>> A^-1*b
```

Отметим, что сравнение скорости решения системы линейных уравнений с помощью средств матричной алгебры пакета MATLAB и функции `Zeidel()`, листинг которой приведен в предыдущем разделе, свидетельствует о неоспоримом преимуществе первых. Данное обстоятельство обусловлено тем, что в пакете MATLAB (который изначально разрабатывался для проведения матричных вычислений) используются специальные быстрые алгоритмы для выполнения арифметических операций с матрицами. Поэтому при решении прикладных задач, в ходе которых возникает необходимость решения систем линейных уравнений, целесообразнее использовать встроенные возможности пакета MATLAB.



Лекция 4

Методы решения систем нелинейных уравнений

Цель лекции: на примере метода Ньютона и метода спуска продемонстрировать основные подходы к решению задачи нахождения корней системы нелинейных уравнений, описать соответствующие алгоритмы и их программные реализации, обсудить использование соответствующих функций пакета MATLAB.

4.1. Векторная запись нелинейных систем. Метод простых итераций

Пусть требуется решить систему уравнений

$$\begin{cases} f_1(x_1,\ x_2,\ \ldots,\ x_n) = 0, \\ f_2(x_1,\ x_2,\ \ldots,\ x_n) = 0, \\ \ldots\ \ \ldots\ \ \ldots\ \ \ldots\ \ \ldots \\ f_n(x_1,\ x_2,\ \ldots,\ x_n) = 0, \end{cases} \tag{4.1}$$

где $f_1, f_2 \ldots, f_n$ — заданные, вообще говоря, нелинейные вещественнозначные функции n вещественных переменных $x_1, x_2, \ldots, x_n$.

Введем обозначения:

$$\bar{x} \equiv \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

$$F(\bar{x}) \equiv \begin{pmatrix} f_1(\bar{x}) \\ f_2(\bar{x}) \\ \vdots \\ f_n(\bar{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, \dots x_n) \\ f_2(x_1, x_2, \dots x_n) \\ \vdots \\ f_n(x_1, x_2, \dots x_n) \end{pmatrix},$$

$$\bar{0} \equiv \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Тогда систему (4.1) можно заменить одним уравнением

$$F(\bar{x}) = \bar{0} \tag{4.2}$$

относительно векторной функции F векторного аргумента $\bar{x}$.

Следовательно, исходную задачу можно рассматривать как задачу о нулях нелинейного отображения $F : R_n \to R_n$. В этой постановке данная задача является прямым обобщением задачи о нахождении решения нелинейного уравнения для случая пространств большей размерности. Это означает, что можно строить методы ее решения как на основе обсужденных в предыдущей лекции подходов, так и осуществлять формальный перенос выведенных для скалярного случая расчетных формул. Однако не все результаты и не все методы оказывается возможным перенести формально (например, метод половинного деления). В любом случае следует позаботиться о правомерности тех или иных операций над векторными переменными и векторными функциями, а также о сходимости получаемых таким способом итерационных процессах. Отметим, что переход от $n = 1$ к $n \geq 2$ вносит в задачу нахождения нулей нелинейного отображения свою специфику, учет которой привел к появлению новых методов и различных модификаций уже имеющихся методов. В частности, большая вариативность методов решения нелинейных систем связана с разнообразием способов, которыми можно решать линейные алгебраические задачи, возникающие при пошаговой линеаризации данной нелинейной вектор-функции $F(\bar{x})$.

Начнем изучение методов решения нелинейных систем с метода простых итераций.

Пусть система (4.1) преобразована к следующей эквивалентной нелинейной системе:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n), \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n), \\ \dots \\ x_n = \varphi_n(x_1, x_2, \dots, x_n) \end{cases} \tag{4.3}$$

или в компактной записи:

$$\bar{x} = \Phi(\bar{x}) = \begin{pmatrix} \varphi_1(\bar{x}) \\ \varphi_2(\bar{x}) \\ \dots \\ \varphi_n(\bar{x}) \end{pmatrix} = \begin{pmatrix} \varphi_1(x_1, x_2, \dots, x_n) \\ \varphi_2(x_1, x_2, \dots, x_n) \\ \dots \\ \varphi_n(x_1, x_2, \dots, x_n) \end{pmatrix}. \qquad (4.4)$$

Для задачи о неподвижной точке нелинейного отображения $\Phi : R_n \to R_n$ запишем формальное рекуррентное равенство:

$$\bar{x}^{(k+1)} = \Phi\left(\bar{x}^{(k)}\right), \qquad (4.5)$$

где k определяет метод простых итераций.

для задачи (4.3) и $k = 0, 1, 2, \dots, n$.

Если начать процесс построения последовательности $\left(\bar{x}^{(k)}\right)$ с некоторого вектора $\bar{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}\right)^T$ и продолжить вычислительный процесс по формуле (4.5), то при определенных условиях данная последовательность со скоростью геометрической прогрессии будет приближаться к вектору $\bar{x}^*$ — неподвижной точке отображения $\Phi(x)$.

Справедлива следующая теорема, которую мы приводим без доказательства.

Теорема 4.1. Пусть функция $\Phi(x)$ и замкнутое множество $M \subseteq D(\Phi) \subseteq R_n$ таковы, что:

1. $\Phi(x) \in M$, $\quad \forall \bar{x} \in M$.
2. $\exists q < 1 : \left\| \Phi(\bar{x}) - \Phi\left(\tilde{\bar{x}}\right) \right\| \le q \left\| \bar{x} - \tilde{\bar{x}} \right\|$, $\quad \forall \bar{x}, \tilde{\bar{x}} \in M$.

Тогда $\Phi(x)$ имеет в M единственную неподвижную точку $\bar{x}^*$; последовательность $\left(\bar{x}^{(k)}\right)$, определяемая (4.5), сходится при любом $\bar{x}^{(0)} \in M$ к $\bar{x}^*$; справедливы оценки:

$$\left\| \bar{x}^* - \bar{x}^{(k)} \right\| \le \frac{q}{1-q} \left\| \bar{x}^{(k)} - \bar{x}^{(k-1)} \right\| \le \frac{q^k}{1-q} \left\| \bar{x}^{(1)} - \bar{x}^{(0)} \right\|, \ \forall k \in N .$$

Отметим низкую практическую ценность данной теоремы из-за не конструктивности ее условий. В случаях, когда выбрано хорошее начальное приближение $\bar{x}^{(0)}$ решению $\bar{x}^*$, больший практический интерес представляет следующая теорема.

Теорема 4.2. Пусть $\Phi(x)$ дифференцируема в замкнутом шаре $S\left(x^{(0)}, r\right) \subseteq D(\Phi)$, причем $\exists q \in (0;1) : \sup_{x \in S} \left\| \Phi'(\bar{x}) \right\| \le q$. Тогда если центр $\bar{x}^{(0)}$ и ра-

диус r шара S таковы, что $\left\|\bar{x}^{(0)} - \Phi\left(\bar{x}^{(0)}\right)\right\| \le r(1-q)$, то справедливо заключение теоремы 4.1 с $M = S$.

Запишем метод последовательных приближений (4.5) в развернутом виде:

$$\begin{cases} x_1^{(k+1)} = \varphi_1\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right), \\ x_2^{(k+1)} = \varphi_2\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right), \\ \qquad \ldots \\ x_n^{(k+1)} = \varphi_n\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right). \end{cases} \tag{4.6}$$

Сравнение (4.6) с вычислительной формулой метода простой итерации решения систем линейных уравнений (4.13) обнаруживает их сходство. Учитывая, что в линейном случае, как правило, более эффективен метод Зейделя, в данном случае также может оказаться более эффективным его многомерный аналог, называемый методом покоординатных итераций:

$$\begin{cases} x_1^{(k+1)} = \varphi_1\left(x_1^{(k)}, x_2^{(k)}, \ldots, x_{n-1}^{(k)}, x_n^{(k)}\right), \\ x_2^{(k+1)} = \varphi_2\left(x_1^{(k+1)}, x_2^{(k)}, \ldots, x_{n-1}^{(k)}, x_n^{(k)}\right), \\ \qquad \ldots \\ x_n^{(k+1)} = \varphi_n\left(x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{n-1}^{(k+1)}, x_n^{(k)}\right). \end{cases} \tag{4.7}$$

Заметим, что, как и для линейных систем, отдельные уравнения в (4.7) неравноправны, т. е. перемена местами уравнений системы (4.3) может изменить в некоторых пределах число итераций и вообще ситуацию со сходимостью последовательности итераций. Для того чтобы применить метод простых итераций (4.6) или его "зейделеву" модификацию (4.7) к исходной системе (4.1), необходимо сначала тем или иным способом привести эту систему к виду (4.3). Это можно сделать, например, умножив (4.2) на неособенную $n \times n$ матрицу A и прибавив к обеим частям уравнения $-A \cdot F(\bar{x})$ вектор неизвестных. Полученная система

$$\bar{x} = \bar{x} - A \cdot F(\bar{x}) \tag{4.8}$$

эквивалентна исходной и имеет вид, аналогичный уравнению в методе итераций в одномерном случае. Проблема состоит лишь в правильном подборе матричного параметра.

4.2. Метод Ньютона решения систем нелинейных уравнений

Для решения системы (4.3) будем пользоваться методом последовательных приближений.

Предположим, известно k-е приближение $\bar{x}^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}\right)$ одного из изолированных корней $\bar{x}^* = \left(x_1^*, x_2^*, \ldots, x_n^*\right)$ векторного уравнения (4.2). Тогда точный корень уравнения (4.2) можно представить в виде:

$$\bar{x}^* = \bar{x}^{(k)} + \Delta\bar{x}^{(k)}, \tag{4.9}$$

где $\Delta\bar{x}^{(k)} = \left(\Delta x_1^{(k)}, \Delta x_2^{(k)}, \ldots, \Delta x_n^k\right)$ — поправка (погрешность корня).

Подставляя выражение (4.9) в (4.2), имеем

$$F\left(\bar{x}^{(k)} + \Delta\bar{x}^{(k)}\right) = 0\,. \tag{4.10}$$

Предполагая, что функция $F(\bar{x})$ непрерывно дифференцируема в некоторой выпуклой области, содержащей $\bar{x}$ и $\bar{x}^{(k)}$, разложим левую часть уравнения (4.10) по степеням малого вектора $\Delta\bar{x}^{(k)}$, ограничиваясь линейными членами:

$$F\left(\bar{x}^{(k)} + \Delta\bar{x}^{(k)}\right) = F\left(\bar{x}^{(k)}\right) + F'\left(\bar{x}^k\right)\Delta\bar{x}^{(k)} \tag{4.11}$$

или в развернутом виде,

$$\begin{cases} f_1\left(x_1^{(k)} + \Delta x_1^{(k)},\ \ldots,\ x_n^{(k)} + \Delta x_n^{(k)}\right) = f_1\left(x_1^{(k)},\ \ldots,\ x_n^{(k)}\right) + \Delta x_1^{(k)}\dfrac{\partial f_1}{\partial x_1} + \ldots \\ \quad \ldots + \Delta x_n^{(k)}\dfrac{\partial f_1}{\partial x_n} = 0, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ f_n\left(x_1^{(k)} + \Delta x_1^{(k)},\ \ldots,\ x_n^{(k)} + \Delta x_n^{(k)}\right) = f_n\left(x_1^{(k)},\ \ldots,\ x_n^{(k)}\right) + \Delta x_1^{(k)}\dfrac{\partial f_n}{\partial x_1} + \ldots \\ \quad \ldots + \Delta x_n^{(k)}\dfrac{\partial f_n}{\partial x_n} = 0. \end{cases} \tag{4.12}$$

Из формул (4.11) и (4.12) видно, что под производной $F'(\bar{x})$ следует понимать матрицу Якоби системы функций f_1, f_2, ..., f_n относительно переменных x_1, x_2, ..., x_n, т. е.

$$F'(\bar{x}) = W(x) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \ldots & \dfrac{\partial f_1}{\partial x_n} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \ldots & \dfrac{\partial f_2}{\partial x_n} \\ \ldots & \ldots & \ldots & \ldots \\ \dfrac{\partial fn}{\partial x_1} & \dfrac{\partial fn}{\partial x_2} & \ldots & \dfrac{\partial fn}{\partial x_n} \end{bmatrix}$$

или в краткой записи

$$F'(\bar{x}) = W(x) = \left[\frac{\partial f_i}{\partial x_j}\right] \quad (i, j = 1, 2, , n),$$

поэтому формула (4.12) может быть записана в следующем виде:

$$F\left(\bar{x}^{(k)}\right) + W\left(\bar{x}^{(k)}\right)\Delta\bar{x}^{(k)} = 0. \tag{4.13}$$

Если $\det W(\bar{x}) = \det\left[\frac{\partial f}{\partial x}\right] \neq 0$, то

$$\Delta\bar{x}^{(k)} = -W^{-1}\left(\bar{x}^{(k)}\right)F\left(\bar{x}^{(k)}\right). \tag{4.14}$$

Отсюда видно, что метод Ньютона для решения системы (4.1) состоит в построении итерационной последовательности:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - W^{-1}\left(\bar{x}^{(k)}\right)F\left(\bar{x}^{(k)}\right), \tag{4.15}$$

где $k = 0, 1, 2,$

Если все поправки становятся достаточно малыми, счет прекращается. Иначе новые значения x_i используются как приближенные значения корней, и процесс повторяется до тех пор, пока не будет найдено решение или не станет ясно, что получить его не удастся.

Пример 4.1. Найти методом Ньютона приближенное положительное решение системы уравнений

$$\begin{cases} f_1(x, y, z) = x^2 + y^2 + z^2 - 1, \\ f_2(x, y, z) = 2x^2 + y^2 - 4z, \\ f_3(x, y, z) = 3x^2 - 4y + z^2, \end{cases}$$

исходя из начального приближения $x_0 = y_0 = z_0 = 0.5$.

Полагая

$$x^{(0)} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}, \quad f(x) = \begin{bmatrix} f_1(x, y, z) \\ f_2(x, y, z) \\ f_3(x, y, z) \end{bmatrix},$$

имеем

$$f(x) = \begin{bmatrix} x^2 + y^2 + z^2 - 1 \\ 2x^2 + y^2 - 4z \\ 3x^2 - 4y + z^2 \end{bmatrix}.$$

Отсюда

$$f\left(x^{(0)}\right)=\begin{bmatrix}0.25+0.25+0.25-1\\0.50+0.25-2.00\\0.75-2.00+0.25\end{bmatrix}=\begin{bmatrix}-0.25\\-1.25\\-1.00\end{bmatrix}.$$

Составим матрицу Якоби:

$$W(x)=\begin{bmatrix}\frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z}\\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z}\\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z}\end{bmatrix}=\begin{bmatrix}2x & 2y & 2z\\ 4x & 2y & -4\\ 6x & -4 & 2z\end{bmatrix}.$$

Имеем

$$W\left(\bar{x}^0\right)=\begin{bmatrix}1 & 1 & 1\\ 2 & 1 & -4\\ 3 & -4 & 1\end{bmatrix},$$

причем

$$\Delta=W\left(\bar{x}^0\right)=\begin{vmatrix}1 & 1 & 1\\ 2 & 1 & -4\\ 3 & -4 & 1\end{vmatrix}=-40.$$

Следовательно, матрица $W\left(\bar{x}^0\right)$ — неособенная. Вычисляем обратную ей матрицу:

$$W^{-1}\left(\bar{x}^0\right)=-\frac{1}{40}\begin{bmatrix}-15 & -5 & -5\\ -14 & -2 & 6\\ -11 & 7 & -1\end{bmatrix}=\begin{bmatrix}\frac{3}{8} & \frac{1}{8} & \frac{1}{8}\\ \frac{7}{20} & \frac{1}{20} & -\frac{3}{20}\\ \frac{11}{40} & -\frac{7}{40} & \frac{1}{40}\end{bmatrix}.$$

По формуле (4.15) получаем первое приближение:

$$\bar{x}^{(1)}=\bar{x}^{(0)}-W^{-1}\left(\bar{x}^{(0)}\right)F\left(\bar{x}^{(0)}\right)=$$

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}-\begin{bmatrix}\frac{3}{8} & \frac{1}{8} & \frac{1}{8}\\ \frac{7}{20} & \frac{1}{20} & -\frac{3}{20}\\ \frac{11}{40} & -\frac{7}{40} & \frac{1}{40}\end{bmatrix}\cdot\begin{bmatrix}-0.25\\-1.25\\-1.00\end{bmatrix}=\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}0.375\\0\\-0.125\end{bmatrix}=\begin{bmatrix}0.875\\0.500\\0.375\end{bmatrix}$$

Аналогично находятся дальнейшие приближения. Результаты вычислений приведены в табл. 4.1.

Таблица 4.1. *Последовательные приближения корней*

i	x	y	z
0	0.5	0.5	0.5
1	0.875	0.5	0.375
2	0.78981	0.49662	0.36993
3	0.78521	0.49662	0.36992

Останавливаясь на приближении $x^{(3)}$, будем иметь:

$$x = 0.7852;\ y = 0.4966;\ z = 0.3699.$$

4.3. Решение нелинейных систем методами спуска

Общий недостаток всех рассмотренных ранее методов решения систем нелинейных уравнений состоит в локальном характере сходимости, затрудняющем их применение в случаях (достаточно типичных), когда существуют проблемы с выбором начального приближения, обеспечивающего сходимость итерационной вычислительной процедуры. В этих случаях можно использовать численные методы оптимизации — данный раздел вычислительной математики выделяется в самостоятельную дисциплину. Для использования наглядной геометрической интерпретации приводимых ниже рассуждений и их результатов ограничимся, как и в предыдущем пункте, рассмотрением системы, состоящей из двух уравнений с двумя неизвестными

$$\begin{cases} f(x,y) = 0, \\ g(x,y) = 0. \end{cases} \tag{4.16}$$

Из функций $f(x,y)$, $g(x,y)$ системы (4.16) образуем новую функцию

$$\Phi(x,y) = f(x,y)^2 + g(x,y)^2 . \tag{4.17}$$

Так как эта функция не отрицательная, то найдется точка (вообще говоря, не единственная) (x^*, y^*) такая, что

$$\Phi(x,y) \geq \Phi(x^*, y^*) \geq 0 \qquad \forall (x,y) \in R_2 .$$

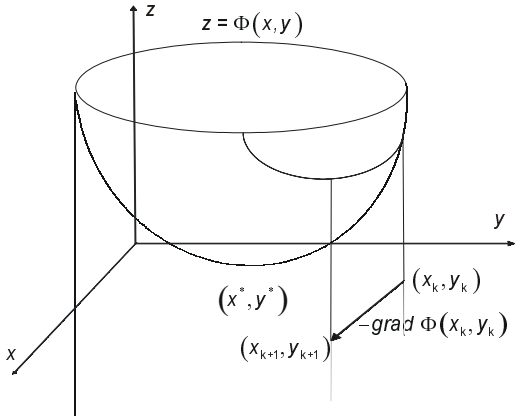


Рис. 4.1. Пространственная интерпретация метода наискорейшего спуска для функции (4.17)

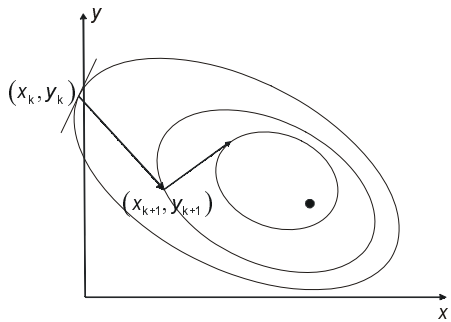


Рис. 4.2. Траектория наискорейшего спуска для функции (4.17) в плоскости *XOY*

Следовательно, если тем или иным способом удается получить точку (x^*, y^*), минимизирующую функцию $\Phi(x, y)$, и если при этом окажется, что $\min\limits_{(x,y)\in R_2} \Phi(x, y) = \Phi(x^*, y^*) = 0$, то точка (x^*, y^*) — истинное решение системы (4.16), поскольку

$$\Phi(x^*, y^*) = 0 \Leftrightarrow \begin{cases} f(x^*, y^*) = 0, \\ g(x^*, y^*) = 0. \end{cases}$$

Последовательность точек (x_k, y_k) — приближений к точке (x^*, y^*) минимума функции $\Phi(x, y)$ — обычно получают по рекуррентной формуле

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \alpha_k \begin{pmatrix} p_k \\ q_k \end{pmatrix}, \tag{4.18}$$

где $k = 0,1,2,\ldots$, $(p_k, q_k)^T$ — вектор, определяющий направление минимизации, а α_k — скалярная величина, характеризующая величину шага минимизации (шаговый множитель). Учитывая геометрический смысл задачи минимизации функций двух переменных $\Phi(x, y)$ — "спуск на дно" поверхности $z = \Phi(x, y)$ (рис. 4.1), итерационный метод (4.18) можно назвать методом спуска, если вектор $(p_k, q_k)^T$ при каждом k является направлением спуска (т. е. существует такое $\alpha > 0$, что $\Phi(x_k + \alpha p_k, y_k + \alpha q_k) < \Phi(x_k, y_k)$), и если множитель α_k подбирается так, чтобы выполнялось условие релаксации $\Phi(x_{k+1}, y_{k+1}) < \Phi(x_k, y_k)$, означающее переход на каждой итерации в точку с меньшим значением минимизируемой функции.

Таким образом, при построении численного метода вида (4.18) минимизации функции $\Phi(x, y)$ следует ответить на два главных вопроса: как выбирать направление спуска $(p_k, q_k)^T$ и как регулировать длину шага в выбранном направлении с помощью скалярного параметра — шагового множителя α_k. Приведем простые соображения по этому поводу.

При выборе направления спуска естественным является выбор такого направления, в котором минимизируемая функция убывает наиболее быстро.

Как известно из математического анализа функций нескольких переменных, направление наибольшего возрастания функции в данной точке показывает ее градиент в этой точке. Поэтому примем за направление спуска вектор

$$\begin{pmatrix} p_k \\ q_k \end{pmatrix} = -\vec{\nabla}\Phi(x_k, y_k) = -\begin{pmatrix} \Phi'_x(x_k, y_k) \\ \Phi'_y(x_k, y_k) \end{pmatrix}$$

— антиградиент функции $\Phi(x, y)$. Таким образом, из семейства методов (4.18) выделяем *градиентный метод*

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \alpha_k \begin{pmatrix} \Phi'_x(x_k, y_k) \\ \Phi'_y(x_k, y_k) \end{pmatrix}. \tag{4.19}$$

Оптимальный шаг в направлении антиградиента — это такой шаг, при котором значение $\Phi(x_{k+1}, y_{k+1})$ наименьшее среди всех других значений $\Phi(x, y)$

в этом фиксированном направлении, т. е. когда точка (x_{k+1}, y_{k+1}) является точкой условного минимума. Следовательно, можно рассчитывать на наиболее быструю сходимость метода (4.19), если полагать в нем

$$\alpha_k = \arg\min_{\alpha>0} \Phi\left(x_k - \alpha\Phi'_x(x_k, y_k), y_k - \alpha\Phi'_y(x_k, y_k)\right). \qquad (4.20)$$

Такой выбор шагового множителя, называемый *исчерпывающим спуском,* вместе с формулой (4.19) определяет *метод наискорейшего спуска.*

Геометрическая интерпретация этого метода хорошо видна на рис. 4.1, 4.2. Характерны девяностоградусные изломы траектории наискорейшего спуска, что объясняется исчерпываемостью спуска и свойством градиента (а значит, и антиградиента) быть перпендикулярным к линии уровня в соответствующей точке.

Наиболее типичной является ситуация, когда найти точно (аналитическими методами) оптимальное значение α_k не удается. Следовательно, приходится делать ставку на применение каких-либо численных методов одномерной минимизации и находить α_k в (4.18) лишь приближенно.

Несмотря на то, что задача нахождения минимума функции одной переменной $\varphi_k(\alpha) = \Phi\left(x_k - \alpha\Phi'_x(x_k, y_k), y_k - \alpha\Phi'_y(x_k, y_k)\right)$ намного проще, чем решаемая задача, применение тех или иных численных методов нахождения значений $\alpha_k = \arg\min \varphi_k(\alpha)$ с той или иной точностью требует вычисления нескольких значений минимизируемой функции. Так как это нужно делать на каждом итерационном шаге, то при большом числе шагов реализация метода наискорейшего спуска в чистом виде является достаточно высокозатратной. Существуют эффективные схемы приближенного вычисления квазиоптимальных α_k, в которых учитывается специфика минимизируемых функций (типа сумм квадратов функций) [11].

Зачастую успешной является такая стратегия градиентного метода, при которой шаговый множитель α_k в (4.18) берется либо сразу достаточно малым постоянным, либо предусматривается его уменьшение, например, делением пополам для удовлетворения условию релаксации на очередном шаге. Хотя каждый отдельный шаг градиентного метода при этом, вообще говоря, далек от оптимального, такой процесс по числу вычислений функции может оказаться более эффективным, чем метод наискорейшего спуска.

Главное достоинство градиентных методов решения нелинейных систем — глобальная сходимость. Нетрудно доказать, что процесс градиентного спуска приведет к какой-либо точке минимума функции из любой начальной точки. При определенных условиях найденная точка минимума будет искомым решением исходной нелинейной системы.

Главный недостаток — медленная сходимость. Доказано, что сходимость этих методов только линейная, причем, если для многих методов, таких как метод Ньютона, характерно ускорение сходимости при приближении к решению, то здесь имеет место скорее обратное. Поэтому есть необходимость построения гибридных алгоритмов, которые начинали бы поиск искомой точки — решения данной нелинейной системы — глобально сходящимся градиентным методом, а затем производили уточнение каким-то быстросходящимся методом, например методом Ньютона (разумеется, если данные функции обладают нужными свойствами).

Примечание

Порядком сходимости последовательности (x_k) к x^* называют такое число p, что $\left|x^* - x_{k+1}\right| \le C\left|x^* - x_k\right|^p$, где $C > 0$, при всех $k > k_0$.

Разработан ряд методов решения экстремальных задач, которые соединяют в себе низкую требовательность к выбору начальной точки и высокую скорость сходимости. К таким методам, называемым *квази-ньютоновскими*, можно отнести, например, *метод переменной метрики* (Дэвидона-Флетчера-Пауэлла), *симметричный* и *положительно определенный методы секущих* (на основе формулы пересчета Бройдена).

При наличии негладких функций в решаемой задаче следует отказаться от использования производных или их аппроксимаций и прибегнуть к так называемым *методам прямого поиска* (циклического покоординатного спуска, Хука и Дживса, Роленброка и т. п.). Описание упомянутых и многих других методов такого типа можно найти в учебной и в специальной литературе, посвященной решению экстремальных задач (см., например, [4,11,15,29]).

Замечание 1. Для разных семейств численных методов минимизации могут быть рекомендованы свои критерии останова итерационного процесса. Например, учитывая, что в точке минимума дифференцируемой функции должно выполняться необходимое условие экстремума, на конец счета градиентным методом можно выходить, когда достаточно малой становится норма градиента. Если принять во внимание, что минимизация применяется к решению нелинейной системы, то целесообразнее отслеживать близость к нулю минимизируемой неотрицательной функции, т. е. судить о точности получаемого приближения по квадрату его евклидовой метрике.

Замечание 2. Для решения n-мерной системы (4.1) следует свести задачу к решению экстремальной задачи:

$$\Phi(\bar{x}) = \sum_{i=1}^{n} f_i^2(\bar{x}) \to \min.$$

Рассмотрим далее примеры реализации некоторых алгоритмов поиска экстремумов функций, зависящих от нескольких переменных, в пакете MATLAB.

Пример 4.1. Алгоритм поиска экстремума функции с шагом, не зависящим от свойств минимизируемой функции.

Простейший вариант метода наискорейшего спуска рассмотрим на примере поиска минимума квадратической функции $f(x,y)=x^2+\mu y^2$ двух переменных с оврагом, пологость которого определяется параметром μ. Решение данной задачи в пакете MATLAB находится выполнением следующей последовательности команд:

1. Создайте файл F_L4.m (листинг 4.1), содержащий описание функции, возвращающей значения функции $f(x,y)$ в узлах координатной сетки.

Листинг 4.1. Файл F_L4.m

```
function z=F_L4(x,y,mu)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
     z(i,j)=x(i).^2+mu*y(j).^2;
  end;
end;
```

2. Постройте графики исследуемой функции при различных значениях параметра μ.

```
>> N=23;
>> Xmin=-5;Xmax=5;
>> Ymin=-5;Ymax=5;
>> i=1:N;j=1:N;
>> x(i)=Xmin+i*(Xmax-Xmin)/N;
>> y(j)=Ymin+j*(Ymax-Ymin)/N;
>> M1=F_L4(x,y,0.5);M2=F_L4(x,y,1);M3=F_L4(x,y,1.5);
>> [X Y]=meshgrid(x,y);
>> surfc(X,Y,M1); colormap gray
>> surfc(X,Y,M2); colorap gray
>> surfc(X,Y,M3); colormap gray
```

Из рис. 4.3–4.5 видно, что при $\mu=1$ функция $f(x,y,\mu)$ представляет собой параболоид вращения, при $\mu>1$ параболоид становится эллиптическим, "вытягиваясь" вдоль оси oX (при $\mu<1$ — вдоль оси oY).

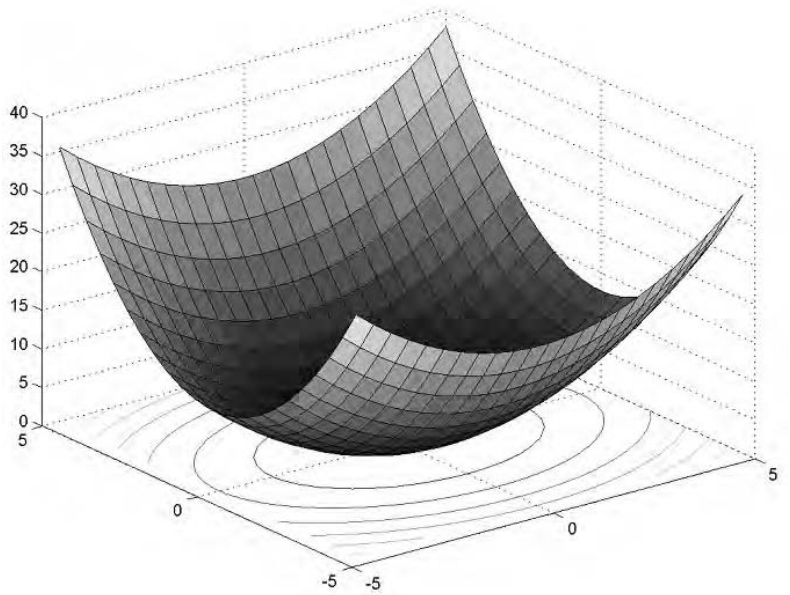


Рис. 4.3. Поверхность и карта линий уровня функции $f(x,y) = x^2 + 0.5 \cdot y^2$

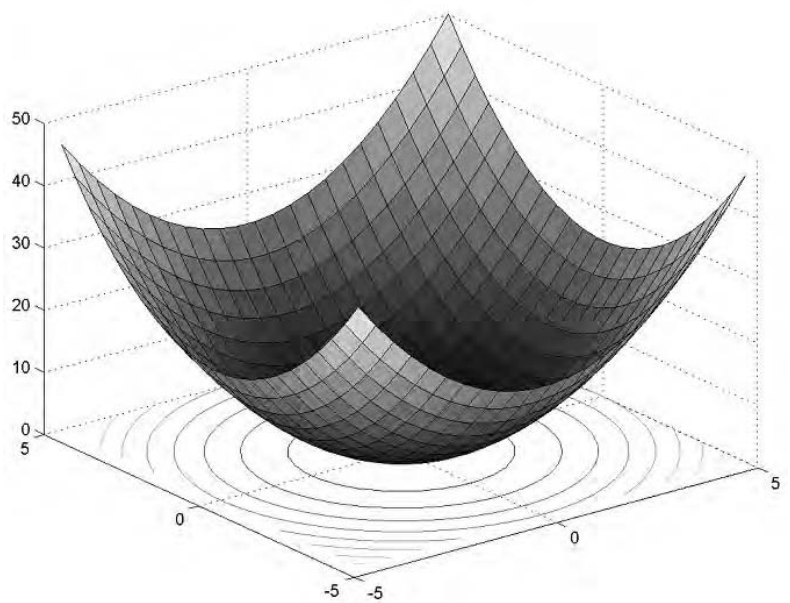


Рис. 4.4. Поверхность и карта линий уровня функции $f(x,y) = x^2 + y^2$

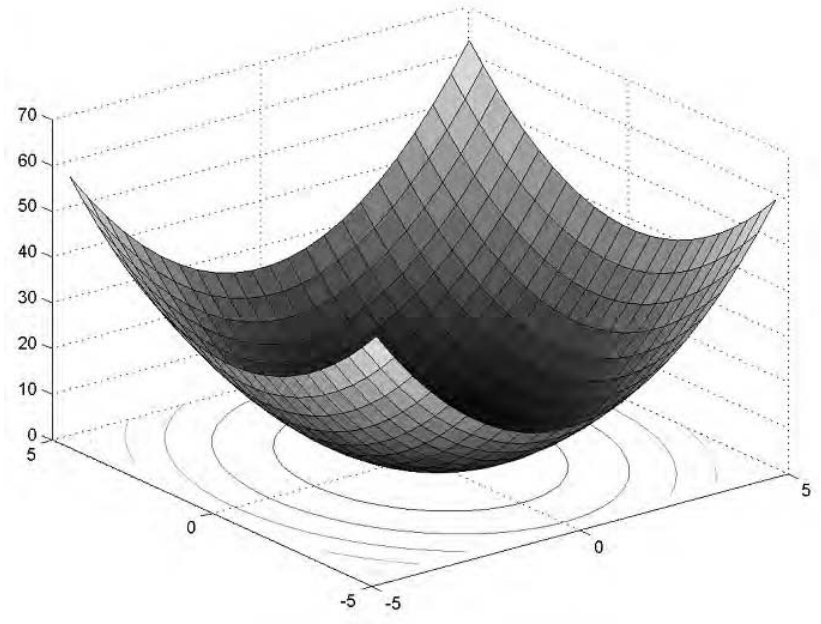


Рис. 4.5. Поверхность и карта линий уровня функции $f(x,y)=x^2+1.5\cdot y^2$

3. Создайте файл Dx_F_L4.m (листинг 4.2), содержащий описание функции, возвращающей значения частной производной, по переменной *x*.

Листинг 4.2. Файл Dx_F_L4.m

```
function z=Dx_F_L4(x,y,mu)
N=length(x);
z=zeros(N);
i=1:N;
j=1:N;
for i=1:N
  for j=1:N
    z(i,j)=2*x(i);
  end;
end;
```

4. Создайте файл Dy_F_L4.m (листинг 4.3), содержащий описание функции, возвращающей значения частной производной функции $f(x,y,\mu)$ по переменной *y*.

Листинг 4.3. Файл Dy_F_L4.m

```
function z=Dy_F_L4(x,y,mu)
N=length(x);
z=zeros(N);
i=1:N;
j=1:N;
for i=1:N
  for j=1:N
    z(i,j)=2*mu*x(i);
  end;
end;
```

5. Создайте файл L_Grad.m (листинг 4.4), содержащий описание функции, возвращающей длину градиента функции $f(x, y, \mu)$.

Листинг 4.4. Файл L_Grad.m

```
function z=L_Grad(x,y,mu)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=Dx_F_L4(x(i),y(j),mu).^2+Dy_F_L4(x(i),y(j),mu).^2;
    z(i,j)=z(i,j).^0.5;
  end;
end;
```

6. Создайте файл S_x.m (листинг 4.5), содержащий описание функции, возвращающей значение проекции на ось oX нормированного единичного вектора, противоположно направленного вектору градиента.

Листинг 4.5. Файл S_x.m

```
function z=S_x(x,y,mu);
N=length(x);
z=zeros(N);
i=1:N;
j=1:N;
for i=1:N
  for j=1:N
    z(i,j)=-Dx_F_L4(x(i),y(j),mu)./L_Grad(x(i),y(j),mu);
  end;
end;
```

7. Создайте файл S_y.m (листинг 4.6), содержащий описание функции, возвращающей значение проекции на ось oY нормированного единичного вектора, противоположно направленного вектору градиента.

Листинг 4.6. Файл S_y.m

```
function z=S_y(x,y,mu);
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=-Dy_F_L4(x(i),y(j),mu)./L_Grad(x(i),y(j),mu);
  end;
end;
```

8. Создайте файл Lambda.m (листинг 4.7), содержащий описание функции, возвращающей значение шага.

Листинг 4.7. Файл Lambda.m

```
function z=Lambda(x,y,nu,alpha,beta,gamma,lambda0);
z=alpha*lambda0/(beta+gamma*nu);
```

9. Создайте файл MG.m (листинг 4.8), содержащий описание функции, возвращающей значения переменных x, y и соответствующее значение функции $f(x, y, \mu)$ на каждом шаге итерационного процесса.

Листинг 4.8. Файл MG.m

```
function [X,Y,ff]=MG(x0,y0,mu,nu,alpha,beta,gamma,lambda0)
X(1)=x0;
Y(1)=y0;
ff(1)=F_L4(x0,y0,mu);
for i=2:nu
   X(i)=X(i-1)+Lambda(X(i-1),Y(i-1),nu,alpha,beta,gamma,…
lambda0)*s_x(X(i-1),Y(i-1),mu);
   Y(i)=Y(i-1)+Lambda(X(i-1),Y(i-1),nu,alpha,beta,gamma,…
ambda0)*s_y(X(i-1),Y(i-1),mu);
    ff(i)=F_L4(X(i),Y(i),mu);
end;
```

10. Найдите минимум функции $f(x, y, \mu)$ и визуализируйте итерационный процесс.

```
>> Vmax=20; % максимальное число шагов итерационного процесса
>> x0=2; y0=-1; % начальное приближение
>> lambda0=0.3; % начальное значение шага к экстремуму
>> beta=1; alpha=1; gamma=1; % параметры, используемые для
                             % определения шага
>> mu=1;nu=20;
>> [X,Y,ff]=MG(x0,y0,mu,nu,alpha,beta,gamma,lambda0);
>> plot(X);
>> hold on
>> plot(Y); % номера итерационного процесса (рис. 4.6)
>> hold off;
>> plot(ff); % зависимость значения функции f(x,y) от номера
            % итерации (рис. 4.7)
>> stairs(X,Y); % построение траектории  итерационного процесса
                % на плоскости XoY (рис. 4.8)
>> plot3(X,Y,ff) % построение траектории итерационного
                 % процесса в трехмерном пространстве (рис. 4.9)
```

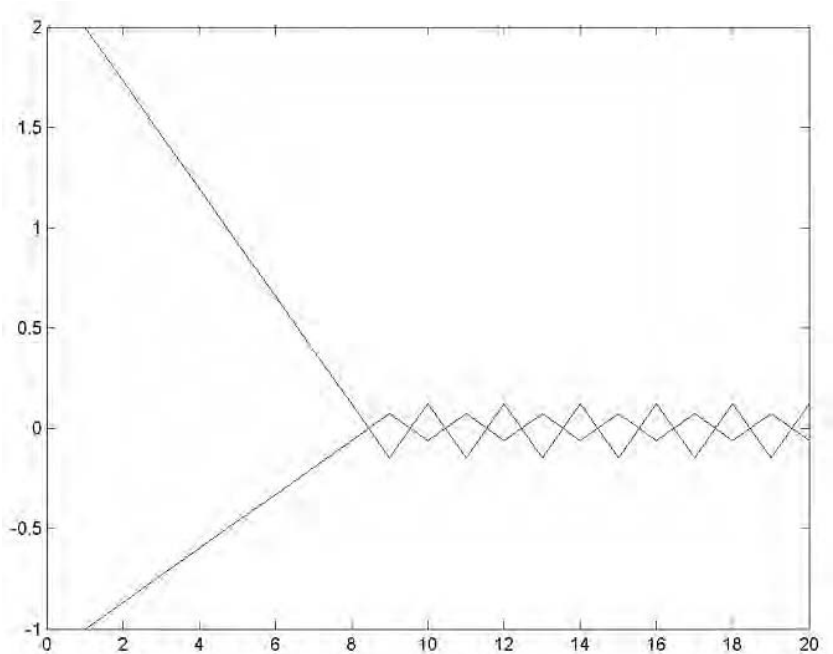


Рис. 4.6. Траектория решения системы нелинейных уравнений на плоскости *XoY*

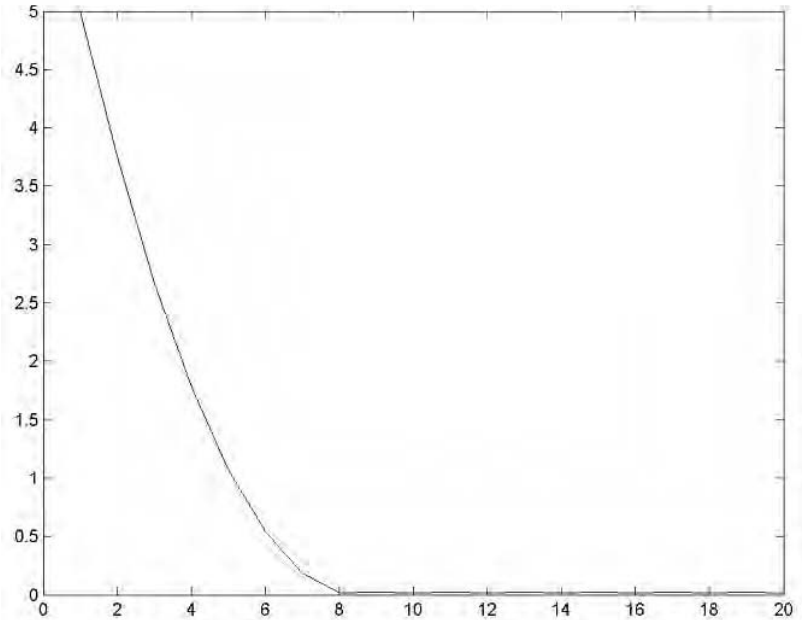


Рис. 4.7. Зависимость значения минимума функции от номера итерации

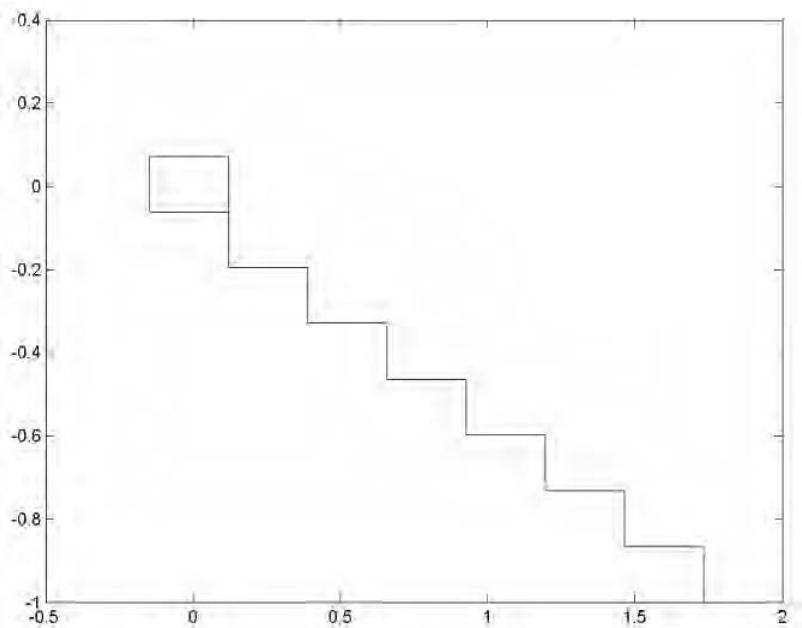


Рис. 4.8. Траектория итерационного процесса на плоскости *XoY*

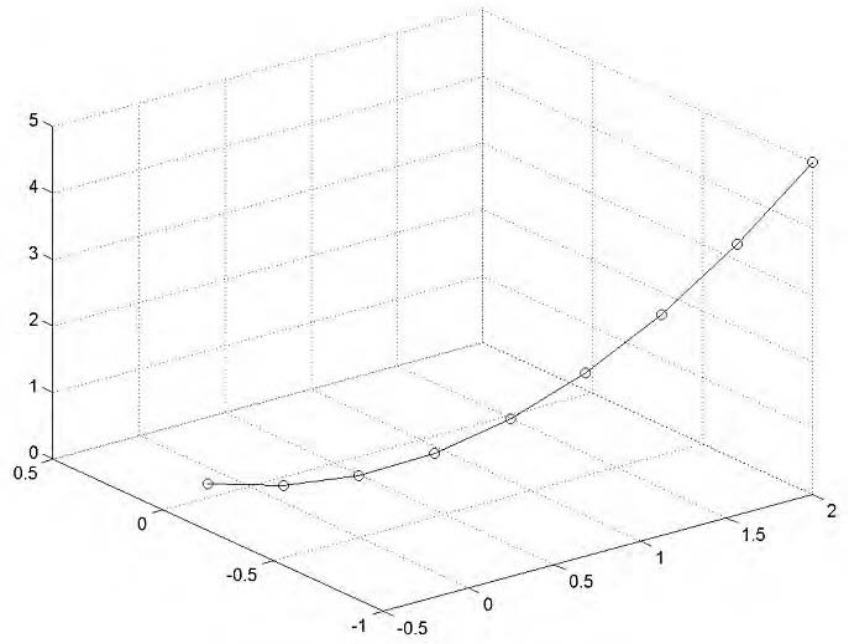


Рис. 4.9. Визуализация итерационного процесса в трехмерном пространстве

Пример 4.2. Алгоритм поиска экстремума с шагом, зависящим от свойств минимизируемой функции (использование производной по направлению).

Исследуем данный алгоритм применительно к минимизации функции двух переменных, заданной полиномом 4-го порядка:

$$f(x,y) = k_x x^4 + k_y y^4 + l_x x^3 + l_y y^3 + o_x x^2 + o_y y^2 + p_x x + p_y y + q + k_{xy} x^4 y^4 + \ldots + \\ + \ldots + l_{xy} x^3 y^3 + o_{xy} x^2 y^2 + p_{xy} xy$$

Форма функции определяется коэффициентами k_x, k_y, l_x, l_y и т. д.

1. Создайте файл F2_L4.m (листинг 4.9), содержащий описание функции, возвращающей значения функции $f(x, y)$.

Листинг 4.9. Файл F2_L4.m

```
function z=F2_L4(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
```

```
  for j=1:N
    z(i,j)=Kx*x(i).^4+Ky*y(j).^4+Lx*x(i).^3+Ly*y(j).^3+…
           Ox*x(i).^2+Oy*y(j).^2+Px*x(i)+Py*y(j)+Q+…
           Kxy*x(i).^4*y(j).^4+Lxy*x(i).^3*y(j).^3+…
           Oxy*x(i).^2*y(j).^2+Pxy*x(i).*y(j);
  end;
end;
```

2. Постройте график исследуемой функции при выбранных значениях параметров.

```
>> Lx=0.3;Ly=0.4;Ox=1;Px=1.2;Lxy=0.4;Oxy=0.3;Q=3;
>> Kx=0.5;Ky=1;Oy=2;Py=0.6;Kxy=0.2;Pxy=0.1;
>> Xmin=-1;Xmax=1;
>> Ymin=-1;Ymax=1;
>> N=23;
>> x(i)=Xmin+(Xmax-Xmin)/(N-1)*(i-1);
>> y(j)=Ymin+(Ymax-Ymin)/(N-1)*(j-1);
>> M=F2_L4(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
>> [X Y]=meshgrid(x,y);
>> surfc(X,Y,M)
```

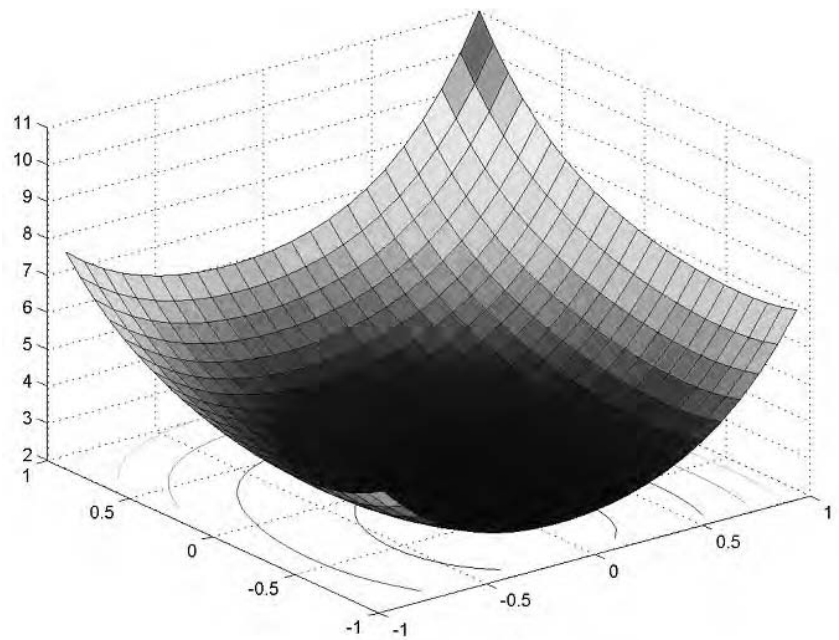


Рис. 4.10. Визуализация функции $f(x, y)$

График функции, изображенной на рис. 4.10, имеет "плато" — чрезвычайно пологое "дно". Понятно, что градиент в области "дна" будет иметь малую величину, поэтому можно ожидать, что алгоритмы с шагом, не зависящим от формы функции, будут иметь плохую сходимость.

3. Создайте файлы g2_x.m (листинг 4.10) и g2_y.m (листинг 4.11), содержащие описание функций, возвращающих значения частных производных.

Листинг 4.10. Файл g2_x.m

```
function z=g2_x(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=4*Kx*x(i).^3+3*Lx*x(i).^2+2*Ox*x(i)+Px+...
           4*Kxy*x(i).^3*y(j).^4+3*Lxy*x(i).^2*y(j).^3+…
           2*Oxy*x(i).*y(j).^2+Pxy*y(j);
  end;
end;
```

Листинг 4.11. Файл g2_y.m

```
function z=g2_y(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=4*Ky*y(j).^3+3*Ly*y(j).^2+2*Oy*y(j)+...
           Py+4*Kxy*x(i).^4*y(j).^3+3*Lxy*x(i).^3*y(j).^2+…
           2*Oxy*x(i).^2*y(j)+Pxy*x(i);
  end;
end;
```

4. Создайте файл L2_Grad.m (листинг 4.12), содержащий описание скалярной функции, возвращающей значение длины градиента функции $f(x, y)$.

Листинг 4.12. Файл L2_Grad.m

```
function z=L2_Grad(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
```

```
    z(i,j)=g2_x(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,…
               Kxy,Pxy).^2+...
           g2_y(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,…
                Kxy,Pxy).^2;
    z(i,j)=z(i,j).^0.5;
  end;
end;
```

5. Создайте файлы S2_x.m (листинг 4.13) и S2_y.m (листинг 4.14), содержащие описание функций, возвращающих координаты нормированного единичного вектора, сонаправленного с вектором, противоположным направлению вектора градиента.

Листинг 4.13. Файл S2_x.m

```
function z=S2_x(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
N=length(x);
z=zeros(N);
i=1:N;
j=1:N;
for i=1:N
  for j=1:N
    z(i,j)=-
      g2_x(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)./...
      Grad(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
  end;
end;
```

Листинг 4.14. Файл S2_y.m

```
function
z=S2_y(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
N=length(x);
z=zeros(N);
i=1:N;
j=1:N;
for i=1:N
  for j=1:N
    z(i,j)=
      -g2_y(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)./...
   L2_Grad(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
  end;
end;
```

6. Создайте файлы g2_xx.m (листинг 4.15), g2_xy.m (листинг 4.16), g2_yy.m (листинг 5.17), содержащие описание функций, возвращающих значения производных $f''_{xx}(x,y)$, $f''_{xy}(x,y)$, $f''_{yy}(x,y)$, соответственно.

Листинг 4.15. Файл g2_xx.m

```
function
z=g2_xx(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=12*Kx*x(i).^2+6*Lx*x(i)+2*Ox...
          +12*Kxy*x(i).^2*y(j).^4+6*Lxy*x(i).*y(j).^3+2*Oxy*y(j).^2;
  end;
end;
```

Листинг 4.16. Файл g2_xy.m

```
function
z=g2_xy(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
  z(i,j)=16*Kxy*x(i).^3*y(j).^3+9*Lxy*x(i).^2*y(j).^2+…
             4*Oxy*x(i).*y(j)+Pxy;
  end;
end;
```

Листинг 4.17. Файл g2_yy.m

```
function
z=g2_yy(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=12*Ky*y(j).^2+6*Ly*y(j)+2*Oy+...
           12*Kxy*x(i).^4*y(j).^2+6*Lxy*x(i).^3*y(j)+2*Oxy*x(i).^2;
  end;
end;
```

7. Создайте файл Lambda2.m (листинг 4.18), содержащий описание функции, возвращающей значения коэффициента λ.

Листинг 4.18. Файла Lambda2.m

```
function
z=Lambda2(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
a1=g2_x(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy).*...
       S2_x(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
a2=g2_y(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy).*...
       S2_y(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
      b1=g2_xx(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy).*...
  S2_x(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy).^2;
  b2=2*g2_xy(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)*…
  S2_x(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy).*...
  S2_y(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
     g3=g2_yy(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy).*...
  S2_y(x(i),y(j),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy).^2;
  z(i,j)=-(a1+a2)./(b1+b2+b3);
  end;
end;
```

8. Создайте файл MG2.m (листинг 4.19), содержащий описание функции, возвращающей значения переменных x, y и соответствующее значение функции $f(x, y, \mu)$ на каждом шаге итерационного процесса.

Листинг 4.19. Файл MG2.m

```
function [X,Y,ff]=MG(x0,y0,nu,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy)
X(1)=x0;
Y(1)=y0;
ff(1)=F2_L4(x0,y0,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
for i=2:nu
  X(i)=X(i-1)+Lambda2(X(i-1),Y(i-1),Lx,Ly,Ox,Px,Lxy,Oxy,Q,…
       Kx,Ky,Oy,Py,Kxy,Pxy)*s2_x(X(i-1),Y(i-1),Lx,Ly,Ox,Px,Lxy,Oxy,…
                                  Q,Kx,Ky,Oy,Py,Kxy,Pxy);
  Y(i)=Y(i-1)+Lambda2(X(i-1),Y(i-1),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,…
     Ky,Oy,Py,Kxy,Pxy)*s2_y(X(i-1),Y(i-1),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,…
      Oy,Py,Kxy,Pxy);
  ff(i)=F2_L4(X(i),Y(i),Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,Kxy,Pxy);
end;
```

9. Найдите минимум функции $f(x, y, \mu)$ и визуализируйте итерационный процесс.

```
>> x0=1;y0=0.5; % начальное приближение
>> nu=10; % число шагов итерационного процесса
>>[X Y ff]=MG2(x0,y0,nu,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,… Kxy,Pxy);
>> plot(X); hold on; plot(Y); hold off % рис. 4.11
>> plot(ff); % рис. 4.12
>> stairs(X,Y); % рис. 4.13
>> plot3(X,Y,ff); % рис. 4.14
```

Обратите внимание на чрезвычайно быструю сходимость и на отсутствие колебаний решения. Это достигается за счет того, что формула для шага строится с учетом формы минимизируемой функции.

Из графика, представленного на рис. 4.10, видно, что функция имеет очень пологое "дно", поэтому для алгоритмов с шагом, не зависящим от формы функции, это было бы причиной очень плохой сходимости.

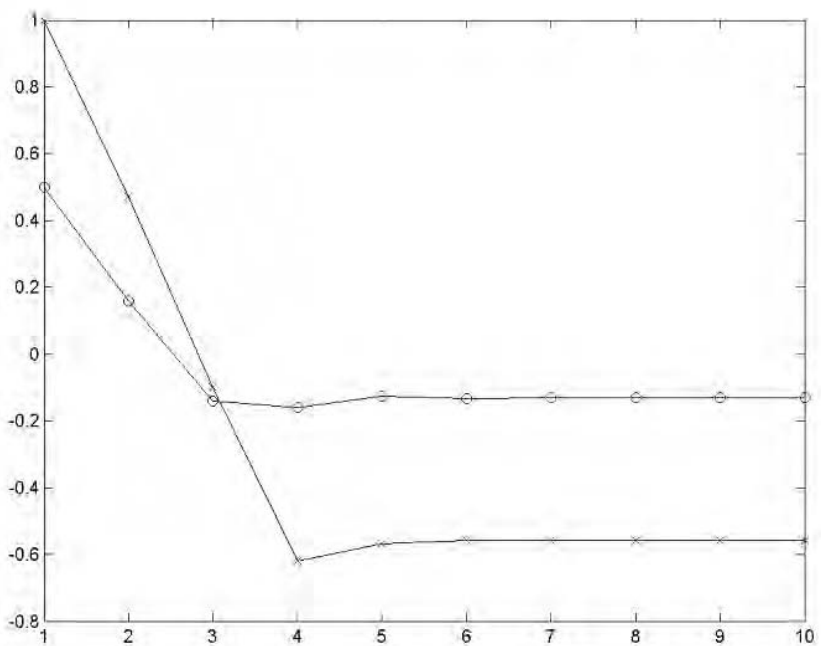


Рис. 4.11. Зависимость координат точки решения от номера итерации

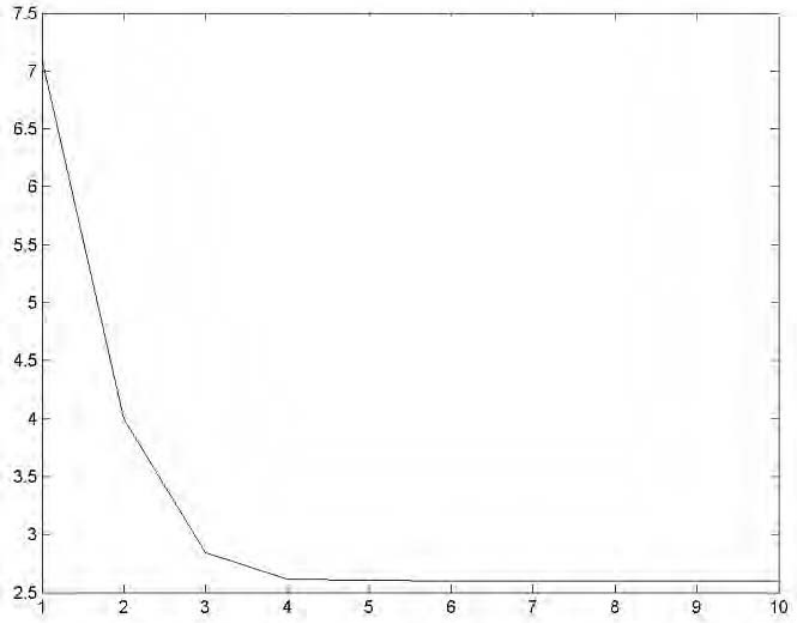


Рис. 4.12. Зависимость значения исследуемой функции от номера итерации

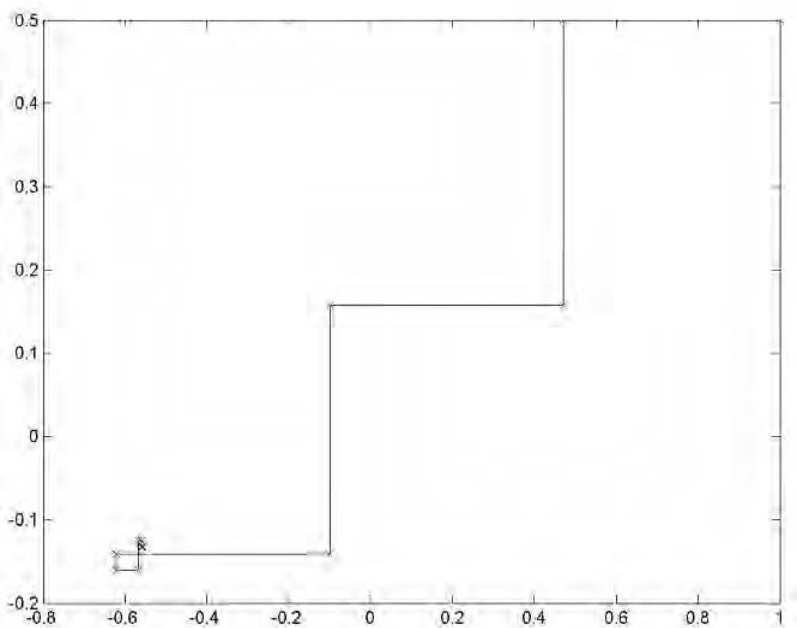


Рис. 4.13. Траектория итерационного процесса в плоскости *XoY*

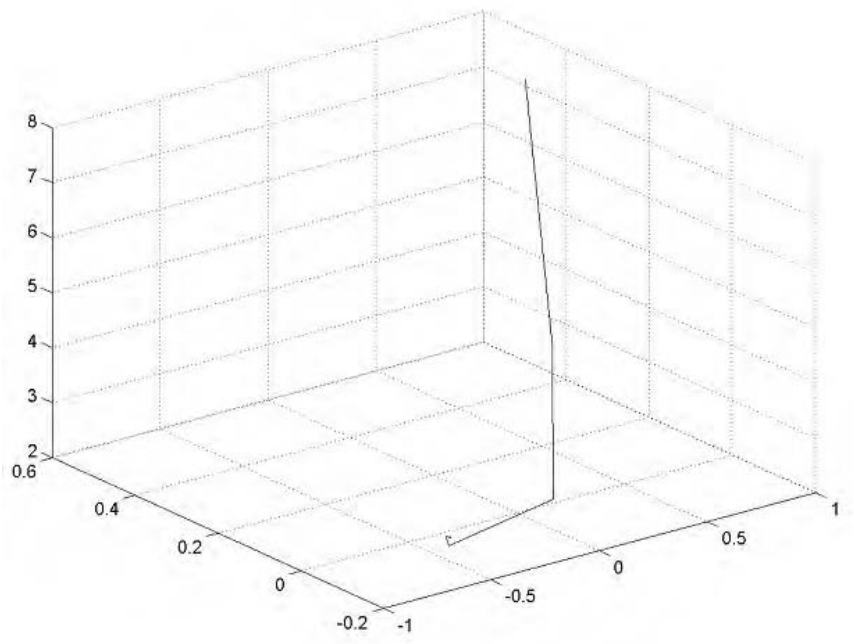


Рис. 4.14. Траектория итерационного процесса в пространстве

4.3.1. Метод Ньютона

Метод Ньютона основан на квадратической аппроксимации минимизируемой функции в окрестности точки $x^{(k)}$. Минимум квадратической функции легко найти, приравнивая ее градиент к нулю. Можно сразу же вычислить положение экстремума и выбрать его в качестве следующего приближения к точке минимума.

Вычисляя точку нового приближения по формуле: $\bar{x}^{(k+1)} = \bar{x}^{(k)} + \Delta\bar{x}^{(k)}$ и разлагая $F\left(\bar{x}^{(k+1)}\right)$ в ряд Тейлора, получим формулу квадратической аппроксимации $F_{sq}\left(\bar{x}^{(k+1)}\right)$:

$$F\left(\bar{x}^{(k+1)}\right) = F\left(\bar{x}^{(k)} + \Delta\bar{x}^{(k)}\right) \cong F_{sq}\left(x^{(k+1)}\right),$$

где

$$F_{sq}\left(\bar{x}^{(k+1)}\right) = F\left(\bar{x}^{(k)}\right) + \left[\nabla f(\bar{x}^{(k)})\right]^T \cdot \Delta\bar{x}^{(k)} + \ldots + \\ + \ldots + \left(1/2!\right) \cdot \left[\Delta\bar{x}^{(k)}\right]^T \cdot \vec{\nabla}^2 F\left(\bar{x}^{(k)}\right) \cdot \Delta\bar{x}^{(k)}, \quad (4.21)$$

$\vec{\nabla}^2 F\left(x^{(k)}\right)$ — матрица вторых производных:

$$\nabla^2 F(\bar{x}^{(k)}) = \begin{vmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2^2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \cdots \quad \cdots & \cdots \quad \cdots \quad \cdots & \cdots & \cdots \quad \cdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{vmatrix}.$$

Условие минимума $F_{sq}\left(\bar{x}^{(k+1)}\right)$ по $\Delta\bar{x}^{(k)}$: $\vec{\nabla} F_{sq}\left(\bar{x}^{(k+1)}\right)=0$. Вычислим градиент $\vec{\nabla} F_{sq}\left(\bar{x}^{(k+1)}\right)=0$ из (4.22) и найдем значение $\Delta\bar{x}^{(k)}$:

$$\nabla F_{sq}\left(\bar{x}^{(k+1)}\right)=\nabla F_{sq}\left(\bar{x}^{(k)}\right)+\vec{\nabla}^2 F_{sq}\left(\bar{x}^{(k)}\right)\Delta\bar{x}^{(k)}=0. \qquad (4.22)$$

Для учета фактических особенностей минимизируемой функции будем использовать в (4.22) значения градиента и матрицы вторых производных, вычисленных не по аппроксимирующей $F_{sq}(.)$, а непосредственно по минимизируемой функции $F(x)$. Заменяя $F_{sq}(.)$ в (4.22), найдем длину шага $\Delta\bar{x}^{(k)}$:

$$\Delta\bar{x}^{(k)} = -\left[\vec{\nabla}^2 F\left(\bar{x}^{(k)}\right)\right]^{-1} \cdot \nabla F\left(\bar{x}^{(k)}\right). \qquad (4.23)$$

Метод Ньютона реализуется следующей последовательностью действий:

1. Задается (*произвольно*) точка начального приближения $\bar{x}^0$.
2. Вычисляется в цикле по номеру итерации k=0,1,:

- значение вектора градиента $\vec{\nabla} F(\bar{x})^{(k)}$ в точке $x=\bar{x}^{(k)}$;
- значение матрицы вторых производных $\vec{\nabla}^2 F(\bar{x})^{(k)}$;
- значение матрицы обратной матрице вторых производных $\vec{\nabla}^2 F(\bar{x})^{(k)^{-1}}$;
- значение шага $\Delta\bar{x}^{(k)}$ по формуле (4.23);
- новое значение приближения $x^{(k+1)}$ по формуле (4.19).

3. Закончить итерационный процесс, используя одно из условий, описанных в *разд. 2.5.*

Оценка погрешности метода Ньютона дается формулой

$$\left\|\tilde{x}^{(k)} - x^{(k)}\right\| \le \delta\left(q^{k-1} + q^{k-2} + \ldots + 1\right) = \frac{\delta\left(1-q^k\right)}{1-q} \le \frac{\delta}{1-q},$$

аналогичной формуле, полученной при доказательстве теоремы о достаточном условии сходимости итерационного процесса.

Достоинства метода Ньютона.

- Для квадратической функции метод позволяет найти минимум за один шаг.
- Для функций, относящихся к классу поверхностей вращения (т. е. обладающих симметрией), метод также обеспечивает сходимость за один шаг (поскольку в точке минимума аргументы минимизируемой функции и ее квадратической аппроксимации совпадают).
- Для несимметричных функций метод обеспечивает более высокую скорость сходимости, чем при использовании других модификаций метода наискорейшего спуска.

К недостаткам метода Ньютона следует отнести необходимость вычислений и (главное!) обращения матриц вторых производных. При этом не только расходуется машинное время, но, что более важно, могут появиться значительные вычислительные погрешности, если матрица $\vec{\nabla}^2 F(\bar{x})^{(k)}$ окажется плохо обусловленной.

Примечание

В качестве количественной характеристики обусловленности матрицы *A*, обозначаемой cond*A*, принимают произведение $\|A\| \cdot \|A^{-1}\|$, где $\| \ \|$ — норма матрицы, определяемая по аналогии с нормой вектора:

$$\|A\|_1 = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2} \ ,$$

$$\|A\|_2 = \max_{1 \le i \le m} \left(\sum_{k=1}^{n} |a_{ik}| \right).$$

Можно показать, что при использовании нормы $\|A\|_1$ число обусловленности матрицы связано с максимальным собственным значением матрицы AA^* (здесь A^* — эрмитово сопряженная матрица) ρ соотношением $\|A\| = \sqrt{\rho}$.

Пример 4.3. Поиск минимума функции Розенброка $f(x,y) = 100 \cdot \left(y - x^2\right)^2 + (1-x)^2$ методом, в котором шаг зависит от свойств минимизируемой функции (метод Ньютона).

1. Создайте файл Rozenbrok.m (листинг 4.20), содержащий описание функции, возвращающей значения функции Розенброка.

Листинг 4.20. Файл Rozenbrok.m

```
function z=Rozenbrok(x,y)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=100*(y(j)-x(i).^2).^2+(1-x(i)).^2;
  end;
end;
```

2. Задайте координатную сетку.

```
>> N=23; %задание числа узлов координатной сетки
>> i=1:N; j=1:N;
>> Xmin=-0.2; Xmax=1.2;
>> Ymin=-0.2; Ymax=1.2;
% задание координат узлов сетки
>> x(i)=Xmin+(Xmax-Xmin)/(N-1)*(i-1);
>> y(j)=Ymin+(Ymax-Ymin)/(N-1)*(j-1);
```

3. Вычислите значения функции в узлах координатной сетки.

```
>> z=Rozenbrok(x,y);
```

4. Визуализируйте функцию Розенброка и карту линий уровня.

```
>> [X Y]=meshgrid(x,y);
>> surf(X,Y,z); % визуализация поверхности, задаваемой функцией
                % Розенброка (рис. 4.15)
>> contour(X,Y,z,53); % визуализация карты эквипотенциалей
                      % функции Розенброка (рис. 4.16)
```

5. Создайте файл R_x.m (листинг 4.21), содержащий описание функции, возвращающей значение первой производной по переменной *x*.

Листинг 4.21. Файл R_x.m

```
function z=R_x(x,y)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=-400*(y(j)-x(i).^2).*x(i)-2*(1-x(i));
  end;
end;
```

6. Создайте файл R_y.m (листинг 4.22), содержащий описание функции, возвращающей значение первой производной по переменной *y*.

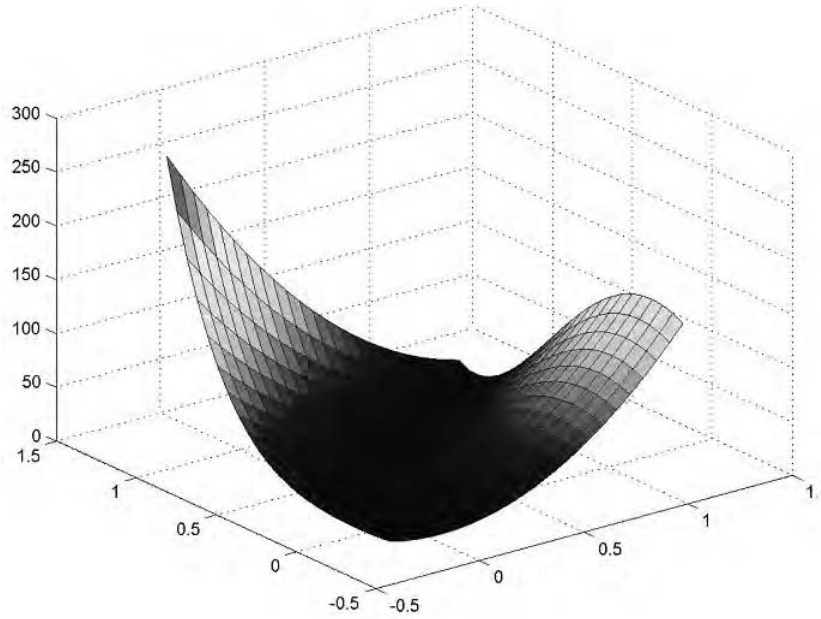


Рис. 4.15. Поверхность, задаваемая функцией Розенброка

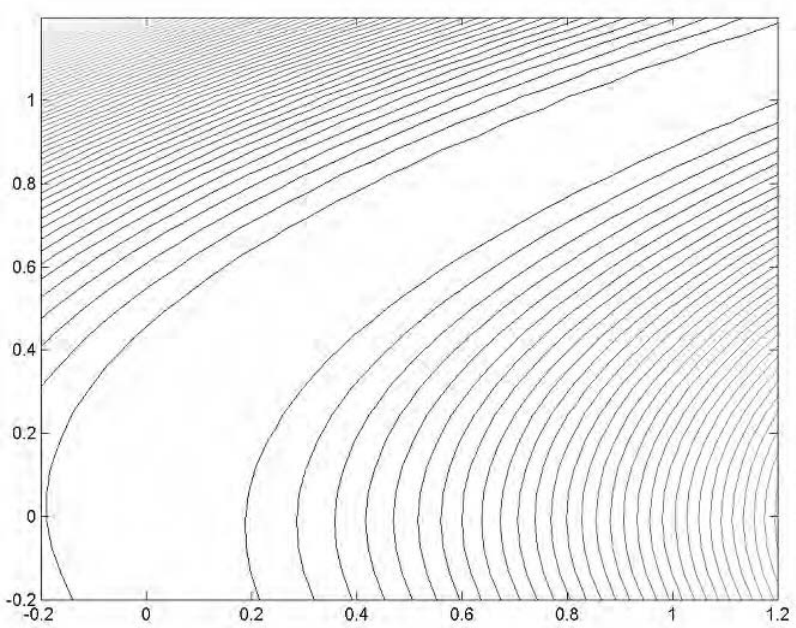


Рис. 4.16. Карта линий уровня функции Розенброка

Листинг 4.22. Файл R_y.m

```
function z=R_y(x,y)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=200*(y(j)-x(i).^2);
  end;
end;
```

7. Создайте файл R_xx.m (листинг 4.23), содержащий описание функции, возвращающей значение второй производной по переменной *x*.

Листинг 4.23. Файл R_xx.m

```
function z=R_xx(x,y)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=1200*x(i).^2-400*y(j)+2;
  end;
end;
```

8. Создайте файл R_yy.m (листинг 4.24), содержащий описание функции, возвращающей значение второй производной по переменной *y*.

Листинг 4.24. Файл R_yy.m

```
function z=R_yy(x,y)
N=length(x);
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=200;
  end;
end;
```

9. Создайте файл R_xy.m (листинг 4.25), содержащий описание функции, возвращающей значение смешанной производной по переменным *x*, *y*.

Листинг 4.25. Файл R_xy.m

```
function z=R_xy(x,y)
N=length(x);
```

```
z=zeros(N);
for i=1:N
  for j=1:N
    z(i,j)=-400*x(i);
  end;
end;
```

10. Создайте файл Rg.m (листинг 4.25), содержащий описание функции, возвращающей значения обратной матрицы вторых производных на вектор-градиент.

Листинг 4.25. Файл Rg.m

```
function z=Rg(x,y)
A(1,1)=R_xx(x,y);
A(1,2)=R_xy(x,y);
A(2,1)=R_xy(x,y);
A(2,2)=R_yy(x,y);
B(1,1)=R_x(x,y);
B(2,1)=R_y(x,y);
z=A^-1*B;
```

11. Создайте файл Rn0.m (листинг 4.26), содержащий описание функции, возвращающей значения переменных и соответствующих значений функции Розенброка.

Листинг 4.26. Файл Rn0.m

```
function z=Rg(x,y)
A(1,1)=R_xx(x,y);
A(1,2)=R_xy(x,y);
A(2,1)=R_xy(x,y);
A(2,2)=R_yy(x,y);
B(1,1)=R_x(x,y);
B(2,1)=R_y(x,y);
z=A^-1*B;
```

12. Визуализируйте итерационный процесс (рис. 4.17–4.19).

```
>> plot(x,'-x');
>> hold on
>> plot(y,'-o');
>> hold off
>> plot(z,'-o')
>> plot3(x,y,z,'-o'); grid on
```

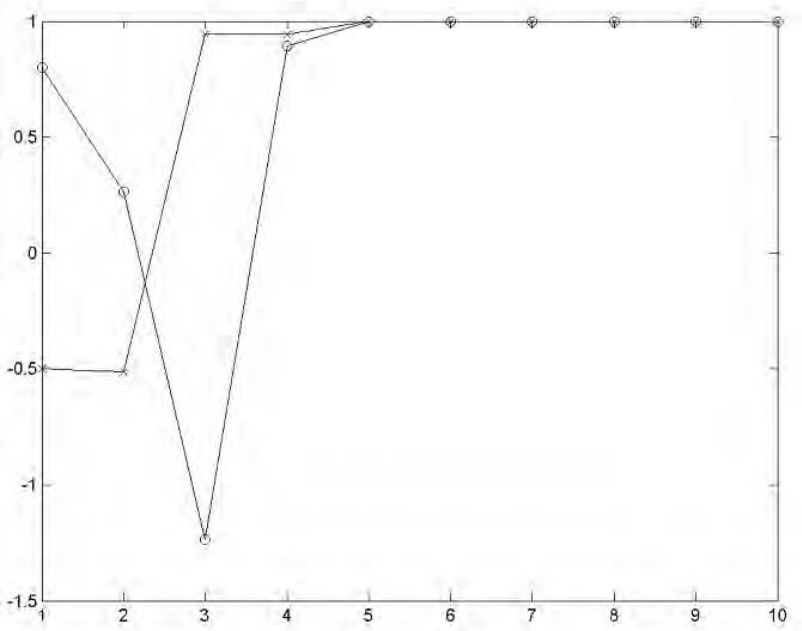


Рис. 4.17. Зависимость значений координат решения уравнения от номера итерации

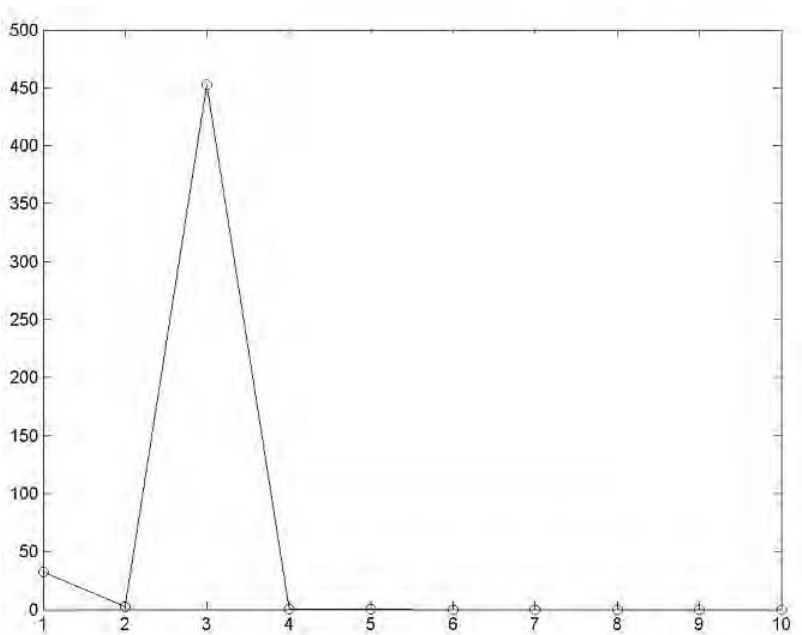


Рис. 4.18. Зависимость значения исследуемой функции от номера итерации

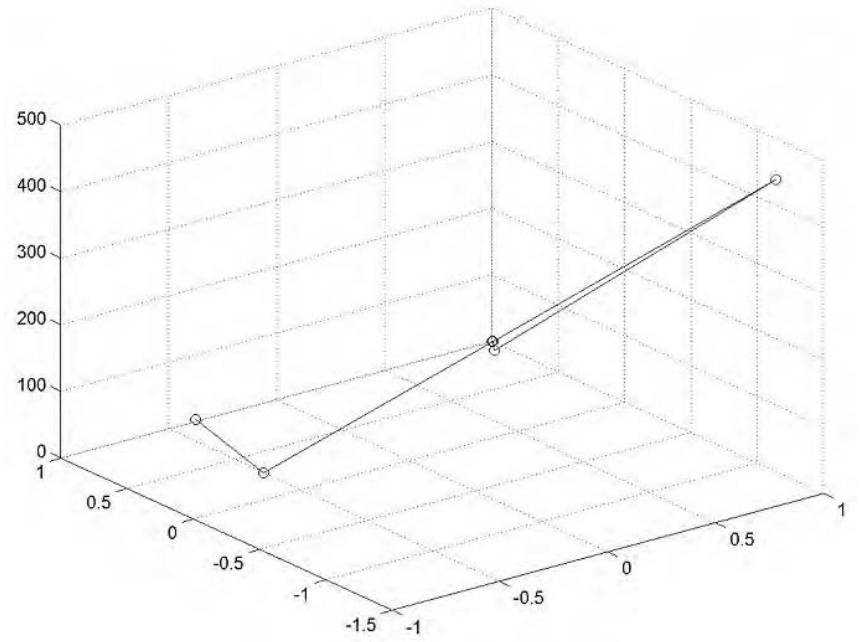


Рис. 4.19. Траектория итерационного процесса в пространстве

4.4. Решение систем нелинейных уравнений средствами пакета MATLAB

Средствами пакета MATLAB найдем решение системы нелинейных уравнений

$$\begin{cases} x^2 + y^2 - 4 = 0 \\ y - x^2 - 1 = 0 \end{cases},$$

которая может быть записана в векторном виде

$$\bar{F}(\bar{x}) = 0,$$

где

$$\bar{F}(\bar{x}) = \begin{pmatrix} x_1^2 + x_2^2 - 4 \\ x_2 - x_1^2 - 1 \end{pmatrix},$$

$$\bar{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

Для этого необходимо выполнить следующую последовательность действий:

1. Создать файл fm.m (листинг 4.27), содержащий описание функции, возвращающей значения функции $\bar{F}(\bar{x})$.

Листинг 4.27. Файл fm.m

```
function z=fm(x)
z(1,1)=x(1).^2+x(2)^2-4;
z(2,1)=x(2)-x(1)^2-1;
```

2. Задать вектор начального приближения.

```
>> z(1,1)=1;
>> z(2,1)=1;
```

3. Обратиться к встроенной функции **fsolve()**, возвращающей решение системы линейных уравнений.

```
>> x = fsolve('fm',z,optimset('fsolve'))
Optimization terminated successfully:
Relative function value changing by less than OPTIONS.TolFun
% Процесс оптимизации завершен
% Относительная величина изменения функции меньше чем
% переменная OPTIONS.TolFun
x =
    0.8895
    1.7913
```

4. Проверить полученное решение.

```
>> fm(x)
ans =
  1.0e-008 *
    0.5162
   -0.4888
```

Для одновременного вывода координат вектора решения уравнения и соответствующего значения вектор-функции следует выполнить команду:

```
>> [x fval] = fsolve('fm',z,optimset('fsolve'))
Optimization terminated successfully:
 Relative function value changing by less than OPTIONS.TolFun
x =
    0.8895
    1.7913
```

```
fval =
   1.0e-008 *
    0.5162
   -0.4888
```

Для вывода на экран монитора значений вектора-решения и соответствующего значения функции $\bar{F}(\bar{x})$ следует выполнить команду:

```
>> [x fval exitflag] = fsolve('fm',z,optimset('Display','Iter'));

                         Norm of      First-order
 Iteration  Func-count     f(x)    step      optimality   CG-iterations
     1          4          5      1            5              0
     2          7          1      1            4              1
     3         10     0.0026      0.223607     0.17           1
     4         13   4.53078e-008  0.013546     0.00055        1
     5         16   5.05378e-017 7.18274e-005 1.79e-008       1
Optimization terminated successfully:
Relative function value changing by less than OPTIONS.TolFun
```

В ряде случаев более удобным оказывается метод Ньютона, при использовании которого (как описано в разделе 4.2) необходимо знать в данной точке и значения функции, и значения якобиана, что позволяет реализовать итерационный процесс. Метод Ньютона в пакете MATLAB реализуется следующей последовательностью действий:

1. Создайте файл Fm1.m (листинг 4.28), содержащий описание функции, возвращающей одновременно значения функции $\bar{F}(\bar{x})$ и значения якобиана.

Листинг 4.28. Файл Fm1.m

```
function [z,J]=fm(x)
z(1,1)=x(1).^2+x(2)^2-4;
z(2,1)=x(2)-x(1)^2-1;
J(1,1)=2*x(1);
J(1,2)=2*x(2);
J(2,1)=-2*x(1);
J(2,2)=1;
```

2. Включите режим использования метода Ньютона и отображения итерационного процесса на экране.

```
>> options=optimset('Jacobian','on','Display','Iter');
```

3. Обратитесь к встроенной функции `fsolve()`, возвращающей решение системы линейных уравнений.

```
>> [x fval exitflag] = fsolve('fm1',z,options);
       Norm of  First-order
Iteration  Func-count     f(x)      step     optimality  CG-iterations
     1         2           5         1          5           0
     2         3           1         1          4           1
     3         4          0.0026     0.223607   0.17        1
     4         5     4.53076e-008    0.013546   0.00055     1
     5         6     5.04984e-017  7.18272e-005 1.79e-008    1
Optimization terminated successfully:
Relative function value changing by less than OPTIONS.TolFun
```

Рассмотрим реализацию метода спуска средствами пакета MATLAB на примере системы нелинейных уравнений, который был разобран ранее в настоящем разделе. Напомним, что для этого (следуя подходу, описанному в разделе 4.3) нужно из уравнений исходной системы создать новую положительно определенную функцию, минимум которой и будет искомым решением системы нелинейных уравнений. Следовательно, для нахождения решения рассматриваемой системы нелинейных уравнений необходимо выполнить следующую последовательность действий:

1. Создать файл F_sq.m (листинг 4.29), содержащий описание функции, возвращающей значения суммы квадратов функций $f(x,y) = x^2 + y^2 - 4$ и $g(x,y) = y - x^2 - 1$.

Листинг 4.29. Файл F_sq.m

```
function z=F_sq(x)
s=fm(x);
z=s(1,1).^2+s(2,1).^2;
```

Примечание

Здесь мы предполагаем, что ранее файл fm.m, содержащий описание функции, возвращающей значения функций *f* (*x*, *y*) и *g* (*x*, *y*), уже создан.

2. Задать начальное приближение, в окрестности которого будет находиться минимум функции $f(x,y)^2 + g(x,y)^2$.

```
>> x=[1;1];
```

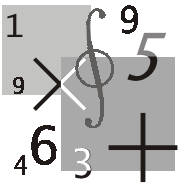
3. Обратиться к встроенной функции **`fminsearch()`**

```
>> fminsearch('f_sq',x)
ans =
    0.8896
    1.7913
```

Примечание

Способы обращения к функции `fminsearch()` и список ее формальных параметров аналогичны способам обращения к функции `fsolve()` и списку ее формальных параметров.

Лекция 5



Интерполирование функций

Цель лекции: рассмотреть основные методы решения задачи об интерполяции функции, заданной таблично (интерполяционный полином, полином Лагранжа, полином Ньютона, сплайн-интерполяция), описать соответствующие алгоритмы и их программные реализации, обсудить использование соответствующих функций пакета MATLAB.

5.1. Постановка задачи

Пусть известные значения некоторой функции $f(x)$ образуют следующую таблицу:

Таблица 5.1. *Исходные данные в задаче интерполяции*

x	x_0	x_1	...	x_n
f(x)	y_0	y_1	...	y_n

Требуется получить значение функции $f(x)$ для значения аргумента $x \in [x_0, x_n]$, несовпадающего ни с одним из значений x_i ($i = 0, 1, ..., n$).

Решение задачи сводится к поиску некоторой приближающей функции $F(x)$, близкой в определенном смысле к функции $f(x)$, для которой известно аналитическое выражение.

Классический подход к решению задачи построения приближающей функции основан на требовании строгого совпадения значений функций $f(x)$ и $F(x)$ в точках $x_i \left(i = 0, 1, \ldots, n\right)$

$$F(x_0) = y_0, F(x_1) = y_1, \ldots, F(x_n) = y_n. \qquad (5.1)$$

В данном случае нахождение приближенной функции называется интерполированием, а точки $x_0, x_1, \ldots, x_n$ называются узлами интерполяции.

Будем искать интерполирующую функцию $F(x)$ в виде многочлена степени n:

$$P_n(x) = a_0 x^n + a_1 x^{n-1} + \dots a_{n-1} x + a_n = \sum_{k=0}^{n} a_k x^{n-k} . \qquad (5.2)$$

Условия (5.1), наложенные на многочлен, позволяют однозначно определить его коэффициенты. Действительно, требуя для $P_n(x)$ выполнения условий (5.1), получаем линейную систему, состоящую из $(n+1)$ уравнения:

$$\sum_{k=0}^{n} a_k x_i^{n-k} = y_i, \quad i = 0,1,\dots,n . \qquad (5.3)$$

Решив систему (5.3) относительно неизвестных $a_0, a_1, \dots, a_n$, находим значения этих неизвестных и, подставив в (5.2), находим аналитическое выражение аппроксимирующей функции.

Система (5.4) всегда имеет единственное решение, т. к. ее определитель

$$\begin{vmatrix} x_0^n & x_0^{n-1} & \dots & 1 \\ x_1^n & x_1^{n-1} & \dots & 1 \\ \vdots & \vdots & \dots & \vdots \\ x_n^n & x_n^{n-1} & \dots & 1 \end{vmatrix}, \qquad (5.4)$$

известный в алгебре как определитель Вандермонда, отличен от нуля.

Следовательно, интерполяционный многочлен $P_n(x)$ существует и единственен.

Пример 5.1. Решить, используя пакет MATLAB, задачу интерполяции с помощью полинома n-ой степени для функции $f(x) = \sin(x)$, заданной табличо в восьми точках на интервале $[0, 2\pi]$.

1. Задайте табличные значения интерполируемой функции.

```
>> N=8;
>> i=1:N;
>> x(i)=2*pi/(N-1)*(i-1);
>> y=sin(x);
```

2. Визуализируйте табличную зависимость и истинные значения функции (рис. 5.1).

```
>> M=1000;
>> j=1:M;
>> X(j)=2*pi/(M-1)*(j-1);
>> Y=sin(X);
>> plot(x,y,'o');↳
```

```
>> hold on
>> plot(X,Y)
```

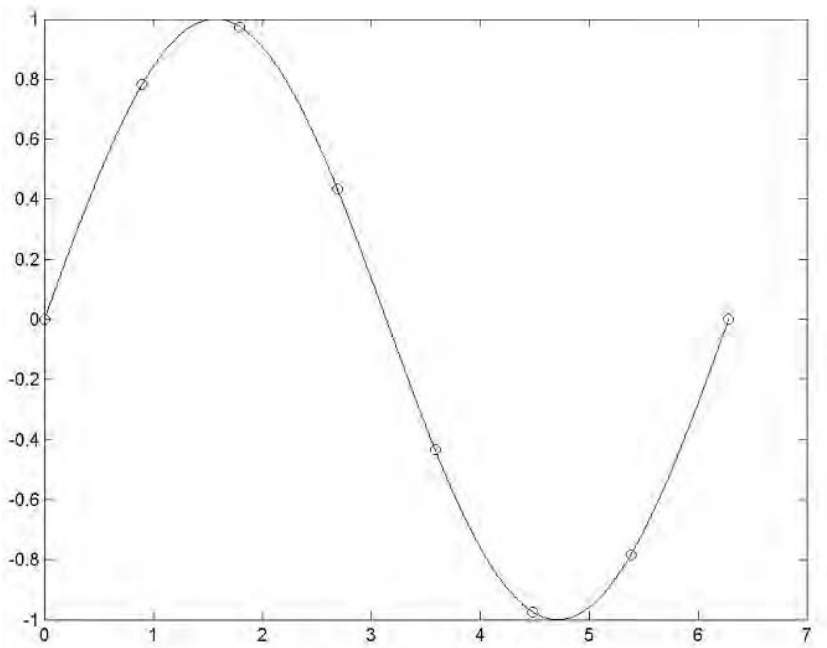


Рис. 5.1. График функции $f(x) = \sin(x)$ и табличных значений, используемых для решения задачи интерполяции

3. Создайте файл Pol.m (листинг 5.1), содержащий описание функции, возвращающей значения полинома (5.2).

Листинг 5.1. Файл Pol.m

```
function z=Pol(x,a)
N=length(a);
M=length(x);
for j=1:M
  s=0;
  for i=1:N
     s=s+a(i)*x(j).^(N-i);
  end;
  z(j)=s;
end;
```

4. Создайте файл Vandermond.m (листинг 5.2), содержащий описание функции, возвращающей значения элементов матрицы Вандермонда.

Листинг 5.2. Файл Vandermond.m

```
function z=Vandermond(x)
N=length(x);
z=ones(N,N);
for i=1:N
  for j=1:N
    z(i,j)=x(i).^(N-j);
  end;
end;
```

5. Вычислите значения элементов матрицы Вандермонда.

```
>> M=Vandermond(x);
```

6. Вычислите значения коэффициентов полинома.

```
>> a=M^-1*y';
```

7. Вычислите значения полинома в заданных промежуточных точках.

```
>> Y1=Pol(X,a);
```

8. Постройте разность между точными и интерполированными значениями функции (рис. 5.2).

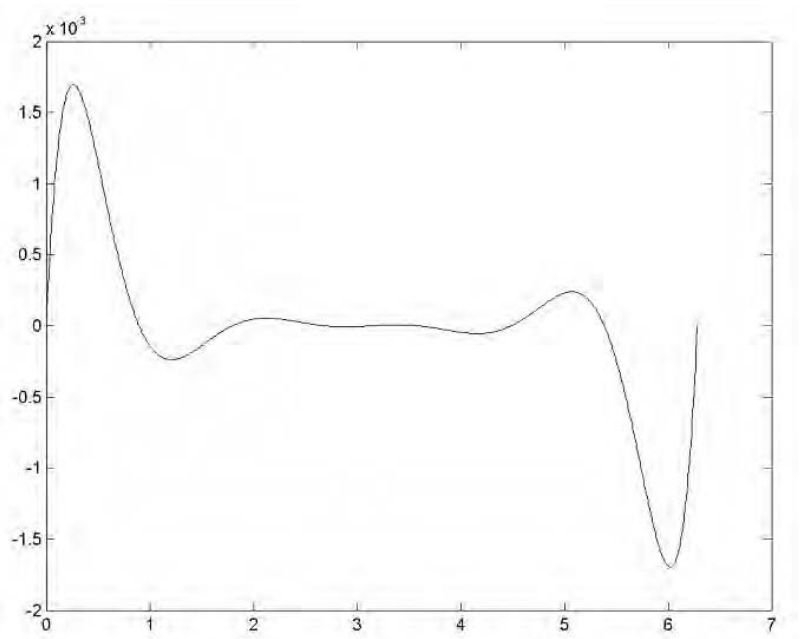


Рис. 5.2. Погрешность аппроксимации функции $f(x) = \sin(x)$ полиномом 8-й степени

5.2. Интерполяционный полином Лагранжа

Для функции, заданной табл. 5.1, построим интерполяционный многочлен $L_n(x)$, степень которого не выше n и выполнены условия (5.1).

Будем искать $L_n(x)$ в виде

$$L_n(x) = l_0(x) + l_1(x) + \ldots + l_n(x), \tag{5.5}$$

где $l_i(x)$ — многочлен степени n, причем

$$l_i(x_k) = \begin{cases} y_i, \text{если } i = k \\ 0, \text{если } i \neq k \end{cases}. \tag{5.6}$$

Очевидно, что требование (5.6) с учетом (5.5) обеспечивают выполнение условий (5.1).

Многочлены $l_i(x)$ составим следующим образом:

$$l_i(x) = c_i(x - x_0)(x - x_1)\cdot\ldots\cdot(x - x_{i-1})(x - x_{i+1})\cdot\ldots\cdot(x - x_n), \tag{5.7}$$

где c_i — постоянный коэффициент, значение которого находится из первой части условия (5.6):

$$c_i = \frac{y_i}{(x_i - x_0)\cdot\ldots\cdot(x_i - x_{i-1})(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}. \tag{5.8}$$

Подставив c_i в (5.7) и далее в (5.5), окончательно получим:

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)\cdot\ldots\cdot(x - x_{i-1})\cdot(x - x_{i+1})\cdot\ldots\cdot(x - x_n)}{(x_i - x_0)\cdot\ldots\cdot(x_i - x_{i-1})\cdot(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}. \tag{5.9}$$

Формула (5.9) решает поставленную задачу.

Пример 5.2. Решить, используя пакет MATLAB, задачу интерполяции с помощью многочлена Лагранжа для функции $f(x) = \sin(x)$, заданной таблично в восьми точках на интервале $[0, 2\pi]$.

1. Задайте табличные значения интерполируемой функции.

```
>> N=8;
>> i=1:N;
>> x(i)=2*pi/(N-1)*(i-1);
>> y=sin(x);
```

2. Создайте файл Lagrange.m (листинг 5.3), содержащий описание функции возвращающей значение многочлена $l_i(x)$.

Листинг 5.3. Файл Lagrange.m

```
function z=Lagrange(x,i,X,Y)
% x – абсцисса точки интерполяции
% i – номер полинома Лагранжа
% X – вектор, содержащий абсциссы узлов интерполяции
% Y – вектор, содержащий ординаты точек интерполяции
N=length(X);
L=1;
for j=1:N
  if not(j==i)
    L=L*(x-X(j))/(X(i)-X(j));
  end;
end;
z=L*Y(i);
```

3. Создайте файл Pol_Lagr.m (листинг 5.4), содержащий описание функции, возвращающей значения полинома Лагранжа.

Листинг 5.4. Файл Lagr.m

```
function z=Pol_Lagr(x,X,Y)
% x – абсцисса точки интерполяции
% i – номер полинома Лагранжа
% X – вектор, содержащий абсциссы узлов интерполяции
% Y – вектор, содержащий ординаты точек интерполяции
N=length(X);
s=0;
for i=1:N
  s=s+Lagrange(x,i,X,Y);
end;
z=s;
```

4. Задайте число промежуточных точек, вычислите их координаты и точные значения интерполируемой функции.

```
>> M=1000;
>> j=1:M;
>> X(j)=2*pi/(M-1)*(j-1);
>> Y=sin(X);
```

5. Вычислите значения полинома Лагранжа в промежуточных точках.

```
>> for j=1:M
     Y2(j)=Pol_Lagr(X(j),x,y);
   end;
```

6. Постройте разность между точными и интерполированными значениями функции (рис. 5.3).

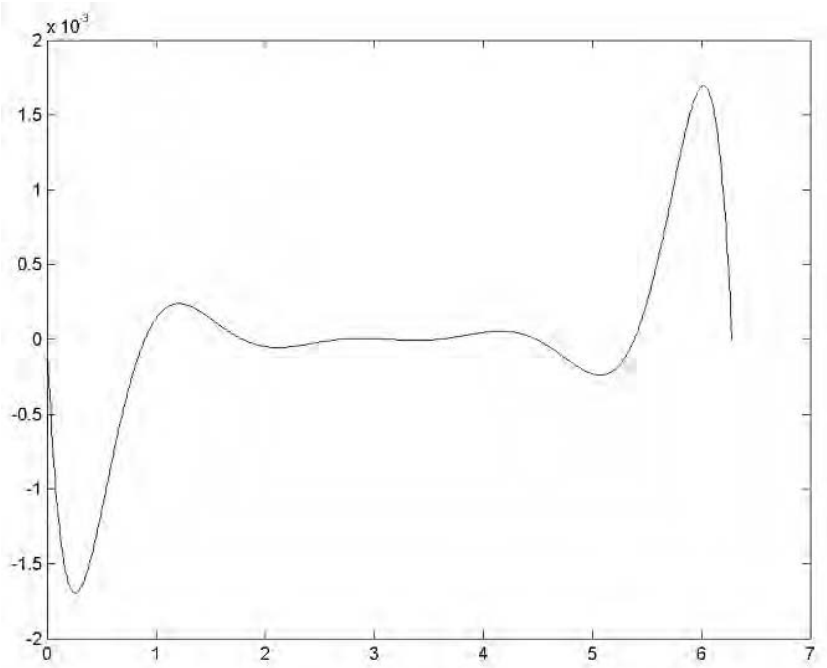


Рис. 5.3. Погрешность аппроксимации функции $f(x) = \sin(x)$ полиномом Лагранжа

5.3. Интерполяционный полином Ньютона для равноотстоящих узлов

Интерполяционные формулы Ньютона строятся для функций, заданных таблицами с равноотстоящими значениями аргумента h:

$$h = x_{i+1} - x_i \ (i = 1, 2, \ldots, n). \tag{5.10}$$

5.3.1. Конечные разности

Для функции, заданной табл. 5.1 с постоянным шагом (5.10), определим разности между значениями функции в соседних узлах интерполяции:

$$\Delta y_i = y_{i+1} - y_i. \tag{5.11}$$

Такое выражение называют конечными разностями первого порядка.

Из конечных разностей первого порядка можно образовать конечные разности второго порядка:

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i = \left(y_{i+2} - y_{i+1}\right) - \left(y_{i+1} - y_i\right) = y_{i+2} - 2y_{i+1} + y_i . \qquad (5.12)$$

Аналогично получают выражение для конечных разностей третьего порядка:

$$\Delta^3 y_i = \Delta^2 y_{i+1} - \Delta^2 y_i = \left(y_{i+3} - 2y_{i+2} + y_{i+1}\right) - \left(y_{i+2} - 2y_{i+1} + y_i\right) = \\ = y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i . \qquad (5.13)$$

Методом математической индукции можно доказать, что

$$\Delta^k y_i = y_{i+k} - k y_{i+k-1} + \frac{k(k-1)}{2!} y_{i+k-2} - \ldots + (-1)^k y_i . \qquad (5.14)$$

5.3.2. Первая интерполяционная формула Ньютона

Будем искать интерполяционный полином в виде:

$$P_n(x) = a_0 + a_1\left(x - x_0\right) + a_2\left(x - x_0\right)\left(x - x_1\right) + \ldots + \\ + \ldots + a_n\left(x - x_0\right)\cdot\ldots\cdot\left(x - x_{n-1}\right) \qquad (5.15)$$

Значения коэффициентов $a_0, a_1, \ldots, a_n$ найдем из условия совпадения значений исходной функции и многочлена в узлах. Полагая $x = x_0$, из (5.15) найдем $y_0 = P_0(x_0) = a_0$, откуда $a_0 = y_0$. Далее, последовательно придавая x значения x_1 и x_2, получаем:

$$y_1 = P_n(x_1) = a_0 + a_1(x_1 - x_0) = a_0 + a_1 h,$$

откуда

$$a_1 = \frac{\Delta y_0}{h},$$

$$y_2 = P_n\left(x_2\right) = a_0 + a_1\left(x_2 - x_0\right) + a_2\left(x_2 - x_0\right)\left(x_2 - x_1\right) = \\ = a_0 + a_1 2h + a_2 2h^2,$$

т. е.

$$a_2 2h^2 = y_2 - a_1 2h - a_0 ,$$

или

$$2ha_2 = y_2 - 2\Delta y_0 - y_0 = y_2 - 2(y_1 - y_0) - y_0 = y_2 - 2y_1 + y_0 = \Delta^2 y_0 ,$$

откуда

$$a_2 = \frac{\Delta^2 y_0}{2!h^2} .$$

Затем, проведя аналогичные действия, можно получить

$$a_3 = \frac{\Delta^3 y_0}{3!h^2} .$$

В общем случае выражение для a_k будет иметь вид

$$a_n = \frac{\Delta^n y_0}{n!\,h^2}. \tag{5.16}$$

Подставляя (5.16) в выражение для многочлена (5.15), получаем

$$P_n(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!h^2}(x - x_0)(x - x_1) + \ldots + \\ + \ldots + \frac{\Delta^n y_0}{n!h}(x - x_0)\ldots(x - x_{n-1}). \tag{5.17}$$

Формула (5.17) называется первым интерполяционным полиномом Ньютона.

Пример 5.3. Решить, используя пакет MATLAB, задачу интерполяции с помощью первого интерполяционного полинома Ньютона для функции $f(x) = \sin(x)$, заданной таблично в восьми точках на интервале $[0, 2\pi]$.

1. Создайте файл Newton1.m (листинг 5.5), содержащий описание функции, возвращающей локальное значение интерполяционного полинома Ньютона.

Листинг 5.5. Файл Newton1.m

```
function z=Newton1(t,x,y)
% t – абсцисса точки, в которой вычисляется значение
% интерполяционного полинома
% x, y – координаты точек, заданных таблично
N=length(x);
for i=1:N
  f(i,1)=y(i);
end;
for k=2:N
  for i=1:N-k+1
    f(i,k)=(f(i+1,k-1)-f(i,k-1))/(x(i+k-1)-x(i));
  end;
end;
s=y(1);
for k=2:N
  r=1;
  for i=1:k-1
    r=r.*(t-x(i));
  end;
  s=f(1,k)*r+s;
end;
z=s;
```

2. Задайте табличные значения интерполируемой функции.

```
>> N=8;
>> i=1:N;
>> x(i)=2*pi/(N-1)*(i-1);
>> y=sin(x);
```

3. Задайте значения абсцисс точек, в которых вычисляется значение интерполяционного полинома.

```
>> M=1000;
>> j=1:M;
>> X(j)=2*pi/(M-1)*(j-1);
>> Y=sin(X); % вычисление точных значений интерполируемой функции
```

4. Вычислите значения полинома Ньютона в узлах заданной координатной сетки.

```
>> for k=1:M
     Z(j)=Newton1(X(j),x,y);
   end;
```

5. Постройте разности между точными и интерполированными значениями функции (рис. 5.4).

```
>> plot(X,Y-Z)
```

Отдавая дань традициям преподавания численных методов в прошлом веке, приведем описание модификации формулы (5.17), применявшейся при ручных вычислениях. Положим $\frac{x-x_0}{h}=t$, т. е. $x=x_0+ht$, тогда:

$$\frac{x-x_1}{h}=\frac{x-x_0-h}{h}=t-1,$$

$$\frac{x-x_2}{h}=\frac{x-x_0-2h}{h}=t-2,$$

$$\ldots$$

$$\frac{x-x_{n-1}}{h}=\frac{x-x_0-(n-1)h}{h}=t-n+1.$$

Подставляя данные выражения в (5.17), окончательно получаем

$$P_n(x)=P_n(x_0+th)=y_0+t\Delta y_0+\frac{t(t-1)}{2!}\Delta^2 y_0+\ldots+ \\ +\ldots+\frac{t(t-1)\cdot\ldots\cdot(t-n+1)}{n!}\Delta^n y_0. \qquad (5.18)$$

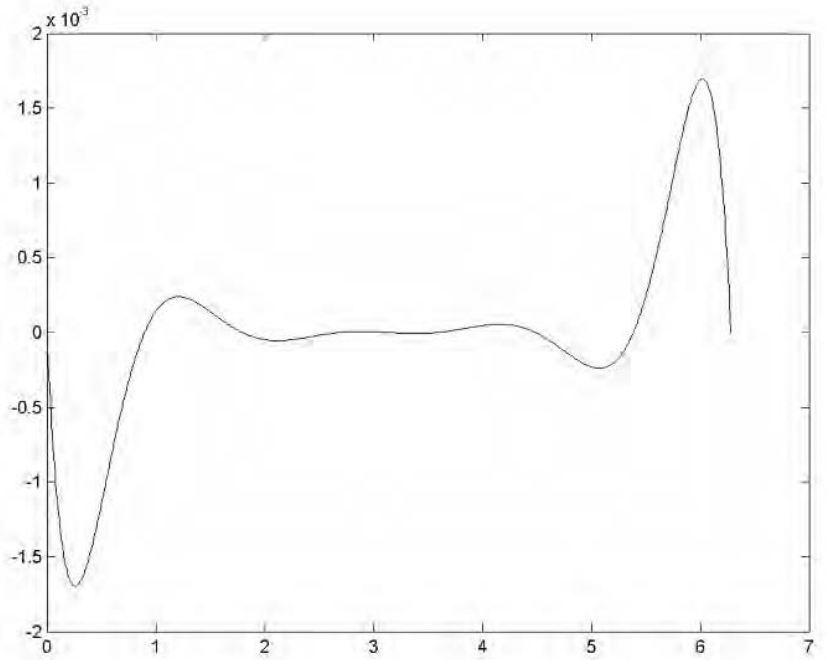


Рис. 5.4. Разность между точными и интерполированными значениями функции с помощью первого полинома Ньютона

Формула (5.18) называется первой интерполяционной формулой Ньютона. Данная формула применяется для интегрирования в начале отрезка, когда t мало по абсолютной величине.

5.3.3. Вторая интерполяционная формула Ньютона

Когда значение аргумента находится ближе к концу отрезка интерполяции, используется вторая интерполяционная формула Ньютона, которая получается, если искать интерполяционный полином в виде:

$$P_n(x) = a_0 + a_1(x - x_n) + a_2(x - x_n)(x - x_{n-1}) + \ldots + \\ + \ldots + a_n(x - x_n) \cdot \ldots \cdot (x - x_1). \qquad (5.19)$$

Коэффициенты полинома (5.19), находятся из условия совпадения значений функции и интерполяционного многочлена в узлах:

$$a_k = \frac{\Delta^k y_{n-k}}{k!\, h^k}. \qquad (5.20)$$

Подставив (5.20) в (5.19) и перейдя к переменной $t = \frac{x - x_n}{h}$, получим окончательный вид интерполяционной формулы Ньютона, используемой при ручных вычислениях:

$$P_n(x) = P_n(x_n + th) = y_n + t\Delta y_{n-1} + \frac{t(t+1)}{2!}\Delta^2 y_{n-2} + \ldots + \\ + \ldots + \frac{t(t+1)\cdot\ldots\cdot(t+n-1)}{n!}\Delta^n y_0 \qquad (5.21)$$

Пример 5.4. Решить, используя пакет MATLAB, задачу интерполяции с помощью второго интерполяционного полинома Ньютона для функции $f(x) = \sin(x)$, заданной таблично в восьми точках на интервале $[0, 2\pi]$.

1. Создайте файл Newton2.m (листинг 5.6), содержащий описание функции, возвращающей локальное значение интерполяционного полинома Ньютона.

Листинг 5.6. Файл Newton2.m

```
function z=Newton2(t,x,y)
N=length(x);
for i=1:N
  f(i,1)=y(i);
end;
for k=2:N
  for i=1:N-k+1
    f(i,k)=(f(i+1,k-1)-f(i,k-1))/(x(i+k-1)-x(i));
  end;
end;
s=y(1);
for k=1:N
  r=1;
  for i=1:k-1
    r=r*(t-x(N-i));
  end;
  s=f(N-k,k)*r+s
end;
z=s;
```

2. Задайте табличные значения интерполируемой функции.

```
>> N=8;
>> i=1:N;
>> x(i)=2*pi/(N-1)*(i-1);
>> y=sin(x);
```

3. Задайте значения абсцисс точек, в которых вычисляется значение интерполяционного полинома.

```
>> M=1000;
>> j=1:M;
>> X(j)=2*pi/(M-1)*(j-1);
>> Y=sin(X); % вычисление точных значений
```

4. Вычислите значения полинома Ньютона в узлах заданной координатной сетки.

```
>> for k=1:M
     Z(j)=Newton1(X(j),x,y);
   end;
```

5. Постройте разности между точными и интерполированными значениями функции (рис. 5.5).

```
>> plot(X,Y-Z)
```

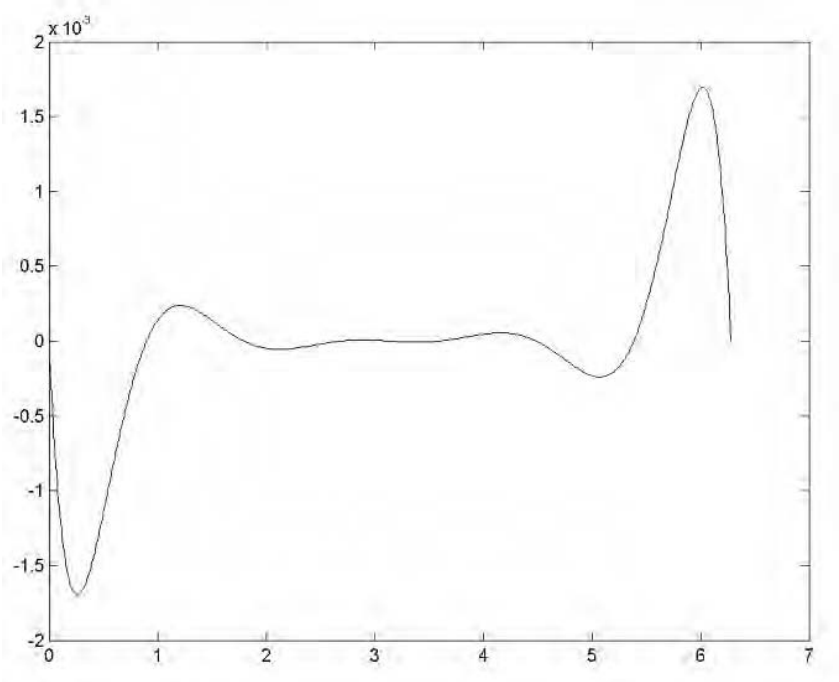


Рис. 5.5. Разность между точными и интерполированными значениями функции с помощью второго полинома Ньютона

5.4. Погрешность интерполяции

Погрешность интерполяции полиномом Лагранжа оценивается по формуле:

$$|R_n(x)| \le \frac{M_{n+1}}{(n+1)!}|\Pi_{n+1}(x)|, \tag{5.22}$$

где

$$M_{n+1} = \max_{x_0 \le x \le x_n} \left|f^{(n+1)}(x)\right|, \tag{5.23}$$

$$\Pi_{n+1}(x) = (x - x_0)(x - x_1)\cdot\ldots\cdot(x - x_n). \tag{5.24}$$

Погрешность интерполяции полиномом Ньютона оценивается по формулам:

$$|R_n(x)| \le \frac{h^{n+1}}{(n+1)!}|t(t-1)(t-2)\cdot\ldots\cdot(t-n)|, \tag{5.25}$$

$$|R_n(x)| \le \frac{h^{n+1}}{(n+1)!}|t(t+1)(t+2)\cdot\ldots\cdot(t+n)|. \tag{5.26}$$

5.5. Сплайн-интерполяция

При большом количестве узлов интерполяции приходится использовать интерполяционные полиномы высокой степени, что создает определенные неудобства при вычислениях. Можно избежать высокой степени интерполяционного многочлена, разбив отрезок интерполяции на несколько частей и построив на каждой части самостоятельный интерполяционный многочлен. Однако такое интерполирование обладает существенным недостатком: в точках сшивки разных интерполяционных полиномов будет разрывной их первая производная, поэтому для решения задачи кусочно-линейной интерполяции используют особый вид кусочно-полиномиальной интерполяции — сплайн-интерполяцию.

Сплайн — это функция, которая на каждом частичном отрезке интерполяции является алгебраическим многочленом, а на всем заданном отрезке непрерывна вместе с несколькими своими производными.

Пусть интерполируемая функция $f(x)$ задана своими значениями y_i в узлах x_i, $(i = 0, 1, \ldots, n)$. Длину частичного отрезка $[x_{i-1}, x_i]$ обозначим h_i:

$$h_i = x_i - x_{i-1}, (i = 1, 2, \ldots, n).$$

Будем искать кубический сплайн на каждом из частичных отрезков $[x_{i-1}, x_i]$ в виде:

$$S(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, \quad (5.27)$$

где a_i, b_i, c_i, d_i — четверка неизвестных коэффициентов. Можно доказать, что задача нахождения кубического сплайна имеет единственное решение.

Потребуем совпадения значений $S(x)$ в узлах с табличными значениями функции $f(x)$:

$$S(x_{i-1}) = y_{i-1} = a_i, \quad (5.28)$$

$$S(x_i) = y_i = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3. \quad (5.29)$$

Число этих уравнений $2n$ в два раза меньше числа неизвестных коэффициентов. Для того чтобы получить дополнительные условия, потребуем также непрерывности первой и второй производных сплайна во всех точках, включая узлы. Для этого следует приравнять левые и правые производные $S'(x-0)$, $S'(x+0)$, $S''(x-0)$, $S''(x+0)$ во внутреннем узле x_i.

Вычислив выражения для производных $S'(x)$, $S''(x)$ последовательным дифференцированием (5.27):

$$S'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2, \quad (5.30)$$

$$S''(x) = 2c_i + 6d_i(x - x_{i-1}), \quad (5.31)$$

найдем правые и левые производные в узле:

$$S'(x_i - 0) = b_i + 2c_i h_i + 2d_i h_i^2,$$

$$S'(x_i + 0) = b_{i+1},$$

где $i = 1, 2, \ldots, n-1$.

Аналогично поступаем для второй производной:

$$S''(x - 0) = 2c_i + 6d_i h_i,$$

$$S''(x + 0) = 2c_{i+1}.$$

Приравняв левые и правые производные, получаем:

$$b_{i+1} = b_i + 2c_i h_i + 2d_i h_i^2, \quad (5.32)$$

$$c_{i+1} = c_i + 3d_i h_i, \quad (5.33)$$

где $i = 0, 1, \ldots, n-1$.

Уравнения (5.32), (5.33) дают еще $2(n - 1)$ условий. Для получения недостающих уравнений накладывают требования к поведению сплайна на кон-

цах отрезка интерполяции. Если потребовать нулевой кривизны сплайна на концах отрезка интерполяции (т. е. равенство нулю второй производной), то получим:

$$c_1 = 0, \quad c_n + 3d_n h_n = 0. \tag{5.34}$$

Исключив из уравнений (5.28)–(5.33) n неизвестных a_i, получаем систему уравнений:

$$\begin{aligned} & b_i h_i - c_i h_i^2 - d_i h_i^3 = y_i - y_{i-1}, \\ & b_{i+1} - b_i - 2c_i h_i - 3d_i h_i^2 = 0, \\ & c_{i+1} - c_i - 3d_i h_i = 0, \\ & c_1 = 0, \\ & c_n + 3d_n h_n = 0, \end{aligned} \tag{5.35}$$

где $i = 0, 1, \ldots, n-1$.

Система (5.35) состоит из $3n$ уравнений. Решив систему (5.35), получаем значения неизвестных b_i, c_i, d_i, определяющих совокупность всех формул для искомого интерполяционного сплайна

$$S_i(x) = y_{i-1} + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, \tag{5.36}$$

где $i = 0, 1, \ldots, n-1$.

Программа, реализующая метод сплайн-интерполяции, оказывается достаточно громоздкой, поэтому мы ограничимся обсуждением решения задачи об интерполяции синуса с помощью кубических сплайнов, используя функцию пакета MATLAB **`spline()`**.

Пример 5.5. Решить, используя пакет MATLAB, задачу интерполяции с помощью кубических сплайнов для функции $f(x) = \sin(x)$, заданной таблично в восьми точках на интервале $[0, 2\pi]$.

1. Задайте табличные значения интерполируемой функции.

```
>> N=8;
>> i=1:N;
>> x(i)=2*pi/(N-1)*(i-1);
>> y=sin(x);
```

2. Задайте значения абсцисс точек, в которых вычисляется значение интерполяционного полинома.

```
>> M=1000;
>> j=1:M;
>> X(j)=2*pi/(M-1)*(j-1);
>> Y=sin(X); % вычисление точных значений интерполируемой функции
```

3. Вычислите интерполируемые значения функции в узлах координатной сетки.

```
>> yy=spline(x,y,X);
```

4. Визуализируйте результаты сплайн-интерполяции и разности между точными и интерполированными значениями (рис. 5.6, 5.7).

```
>> plot(x,y, 'o',X,yy);
>> plot(X,Y-yy);
```

5. Вычислите и визуализируйте значения первых производных сплайна (рис. 5.8).

```
>> m=1:M-1
>> yy1(m)=(yy(m+1)-yy(m))/(2*pi/(M-1));
>> plot(X(m),yy1(m))
```

6. Вычислите и визуализируйте значения вторых производных сплайна (рис. 5.9).

```
>> m=1:M-2;
>> yy2(m)=(yy1(m+1)-yy1(m))/(2*pi/(M-1));
>> plot(X(m),yy2(m))
```

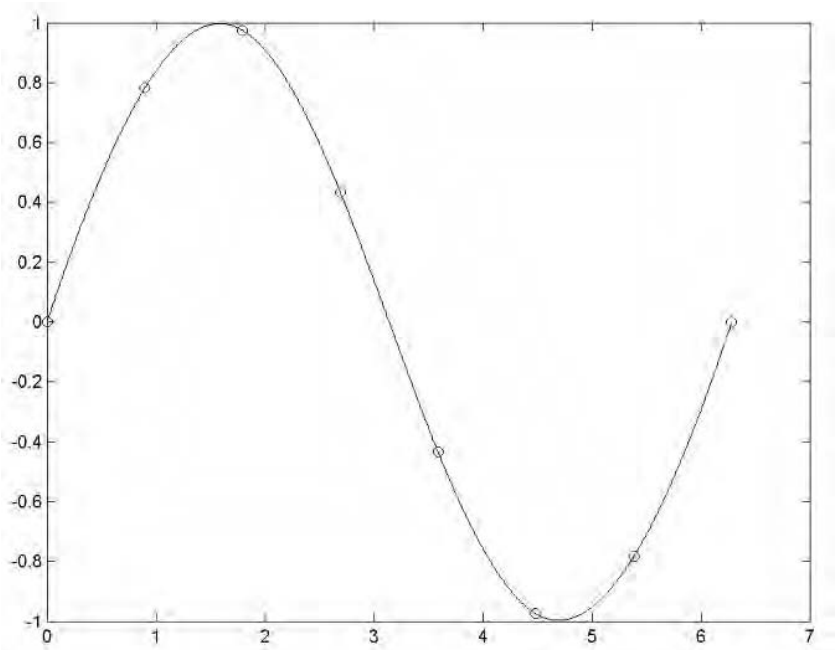


Рис. 5.6. Визуализация исходных данных и результатов сплайн-интерполяции

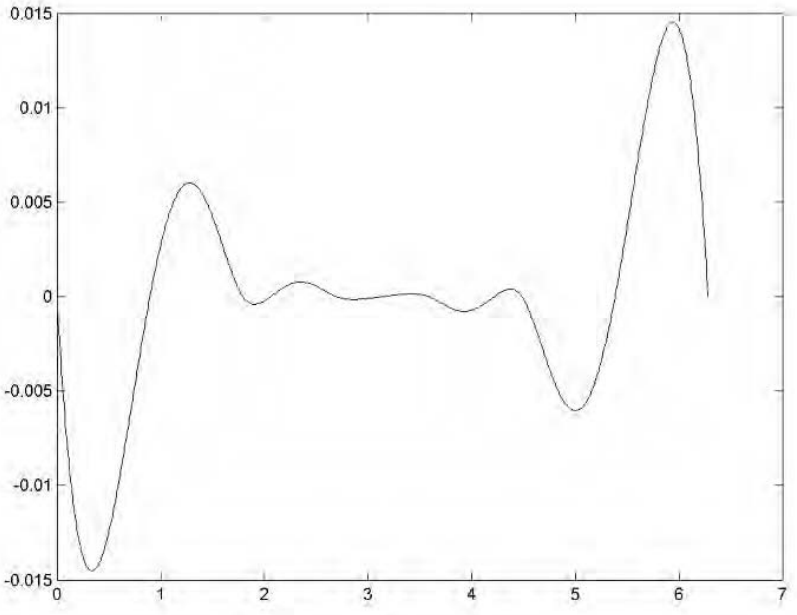


Рис. 5.7. Разность между точными и интерполированными значениями функции $f(x) = \sin(x)$

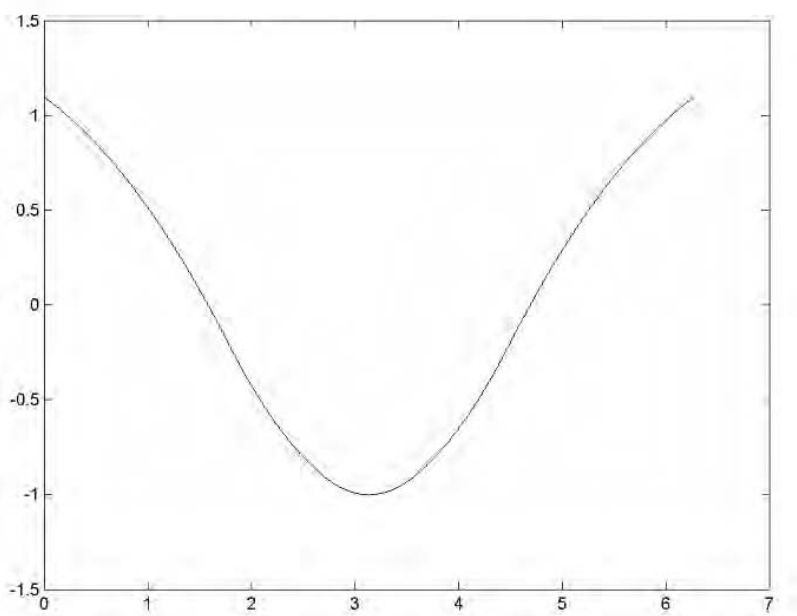


Рис. 5.8. Первая производная, вычисленная по численным значениям, полученным сплайн-интерполяцией

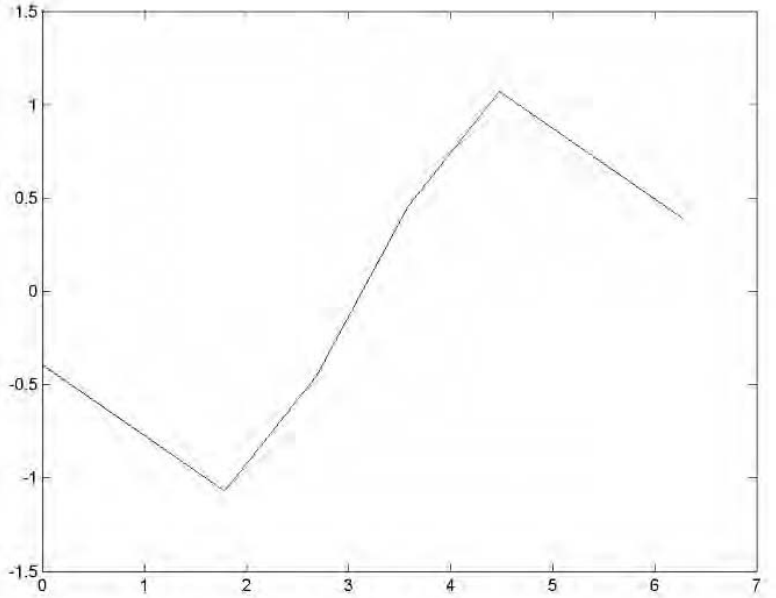


Рис. 5.9. Вторая производная, вычисленная по численным значениям, полученным сплайн-интерполяцией

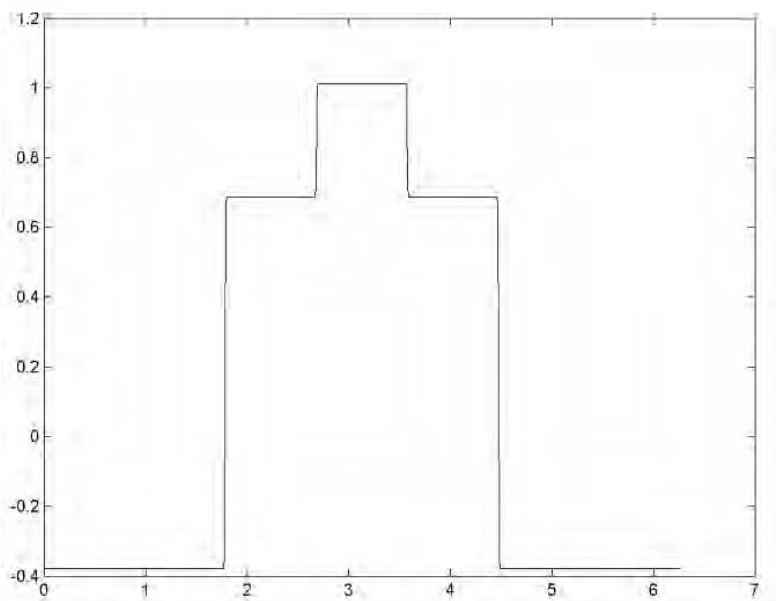


Рис. 5.10. Третья производная, вычисленная по численным значениям, полученным сплайн-интерполяцией

7. Вычислите и визуализируйте значения третьих производных сплайнов (рис. 5.10).

```
>> yy3(m)=(yy2(m+1)-yy2(m))/(2*pi/(M-1));
>> plot(X(m),yy3);
>> m=1:M-3;
```

Как видно из рис. 5.7—5.10, первая и вторая производные сплайнов являются непрерывными функциями, третья и более высокого порядка производные — разрывными функциями.

5.6. Решение задачи одномерной интерполяции средствами пакета MATLAB

Для решения задачи одномерной сплайн-интерполяции в пакете MATLAB используется функция **`interp1()`**, имеющая следующий синтаксис:

`yi = interp1(x,y,xi)` — линейная интерполяция табличных значений `x`, `y` в точках, абсциссы которых находятся в векторе `xi`.

`yi = interp1(y,xi)` — линейная интерполяция табличных значений `y` в точках, абсциссы которых находятся в векторе `xi` в предположении, что `x=1:length(y)`.

`yi = interp1(x,y,xi,method)` — интерполяция линейных значений с использованием выбранного метода интерполяции.

Возможные значения переменной `method`:

- `'nearest'` — интерполяция с использованием ближайших узлов;
- `'linear'` — линейная интерполяция (по умолчанию);
- `'spline'` — интерполяция кубическим сплайном;
- `'pchip'` — интерполяция полиномами Эрмита третьей степени;
- `'cubic'` — аналогично `pchip`.

Пример 5.6. Решить задачу интерполяции для функции $f(x)=\sin(x)$, заданной таблично в восьми точках на интервале $[0, 2\pi]$, используя встроенную функцию пакета MATLAB `interp1()`.

1. Задайте табличные значения интерполируемой функции.

```
>> N=8;
>> i=1:N;
>> x(i)=2*pi/(N-1)*(i-1);
>> y=sin(x);
```

2. Задайте значения абсцисс точек, в которых вычисляется значение интерполяционного полинома.

```
>> M=1000;
>> j=1:M;
>> X(j)=2*pi/(M-1)*(j-1);
>> Y=sin(X); % вычисление точных значений интерполируемой функции
```

3. Вычислите интерполируемые значения функции в узлах координатной сетки и визуализируйте точные исходные данные и точные интерполированные значения (рис. 5.11)

```
>> yi=interp1(x,y,X);
>> plot(x,y,'o',X,Y,X,yi)
```

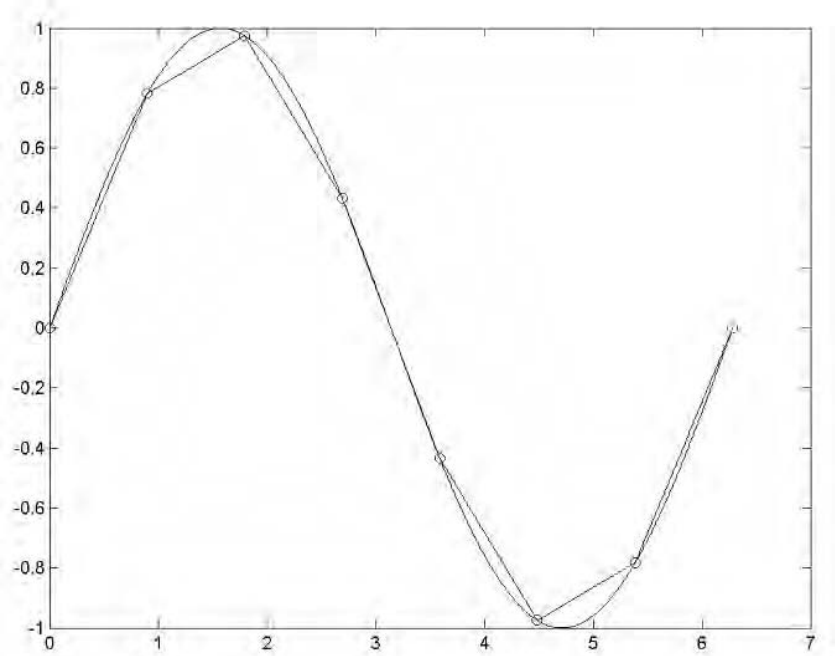
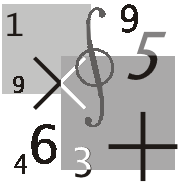


Рис. 5.11. Исходные данные и результат линейной интерполяции



Лекция 6

Численное дифференцирование и интегрирование

Цель лекции: рассмотреть формулы численного дифференцирования и интегрирования (формулу прямоугольников, формулу трапеций, формулу Симпсона), погрешности данных формул, изложить основные идеи вычисления определенных интегралов методом Монте-Карло, обсудить использование соответствующих функций пакета MATLAB.

6.1. Численное дифференцирование функций, заданных аналитически

По определению производная функции $f(x)$ равна:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}. \qquad (6.1)$$

Переходя в (6.1) от бесконечно малых разностей к конечным, получаем приближенную формулу численного дифференцирования:

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}. \qquad (6.2)$$

Формула (6.2) позволяет построить простой вычислительный алгоритм:

1. Задать значение точки, в которой вычисляется производная.
2. Задать значение приращения Δx.
3. Вычислить производную в соответствие с формулой (6.2).

Замена бесконечно малых приращений конечными является причиной возникновения ошибки. Для оценки ее величины разложим функцию $f(x)$ в точке $x+\Delta x$ в ряд Тэйлора:

$$f(x+\Delta x)=f(x)+\frac{f'(x)}{1!}\Delta x+\frac{f''(x)}{2!}(\Delta x)^2+\ldots+\frac{f^{(n)}}{n!}(\Delta x)^n\ldots \qquad (6.3)$$

Подставив (6.3) в (6.2) и приведя подобные члены, получим:

$$f'(x)\approx f'(x)+\frac{f''(x)}{2!}\Delta x+\ldots \qquad (6.4)$$

Из (6.4) видно, что все члены начиная со второго, определяют отличие численного значения производной от ее точного значения. Основной член погрешности равен $f''(x)\Delta x/2!$, т. к. данный член $\sim\Delta x$, говорят, что формула (6.2) имеет первый порядок точности по Δx.

Можно вычислять производную, используя симметричную разностную схему:

$$f'(x)\approx\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}. \qquad (6.5)$$

Для оценки точности данной формулы необходимо удержать первые четыре члена в разложении в ряд Тэйлора:

$$f'(x)\approx\frac{1}{2\Delta x}\left(f(x)+f'(x)\frac{\Delta x}{1!}+f''(x)\frac{(\Delta x)^2}{2!}+f'''(x)\frac{(\Delta x)^3}{3!}+\ldots\right)-$$

$$-\frac{1}{2\Delta x}\left(f(x)-f'(x)\frac{\Delta x}{1!}+f''(x)\frac{(\Delta x)^2}{2!}-f'''(x)\frac{(\Delta x)^3}{3!}+\ldots\right). \qquad (6.6)$$

Раскрыв в (6.6) скобки и приведя подобные члены, получаем:

$$f'(x)\approx f'(x)+f'''(x)\frac{(\Delta x)^2}{3!}+\ldots \qquad (6.7)$$

Из (6.7) видно, что основной член погрешности равен $f'''(x)(\Delta x)^2/3!$, т. к. данный член $\sim(\Delta x)^2$, говорят, что формула (6.5) имеет второй порядок точности по Δx.

Пример 6.1. Вычислить значения производной функции $f(x)=\sin\left(0.01x^2\right)$ на интервале [0, 10π], используя пакет MATLAB.

Решение:

```
>> f=inline('sin(0.01*x.^2)'); % задание дифференцируемой функции
>> dx=0.01; % шаг изменения координатной сетки
```

```
>> x=0:dx:10*pi; % вычисление координат узлов
>> yf=feval(f,x);% вычисление значений функции в узлах
                 % выполнение процедуры численного дифференцирования
>> N=length(x);
>> m=1:N-1;
>> df(m)=(yf(m+1)-yf(m))/dx;
>> plot(df) % визуализация производной функции (рис. 6.1)
>> f1=inline('0.02*x.*cos(0.01*x.^2)'); % задание функции, описывающей
                                        % первую производную
>> ya=feval(f1,x); % вычисление значений первой производной по
                   % аналитической формуле
>> plot(x(m),abs(yf(m)-ya(m)); % визуализация разности между
                               % численными и аналитическими
                               % значениями производной (рис. 6.2)
```

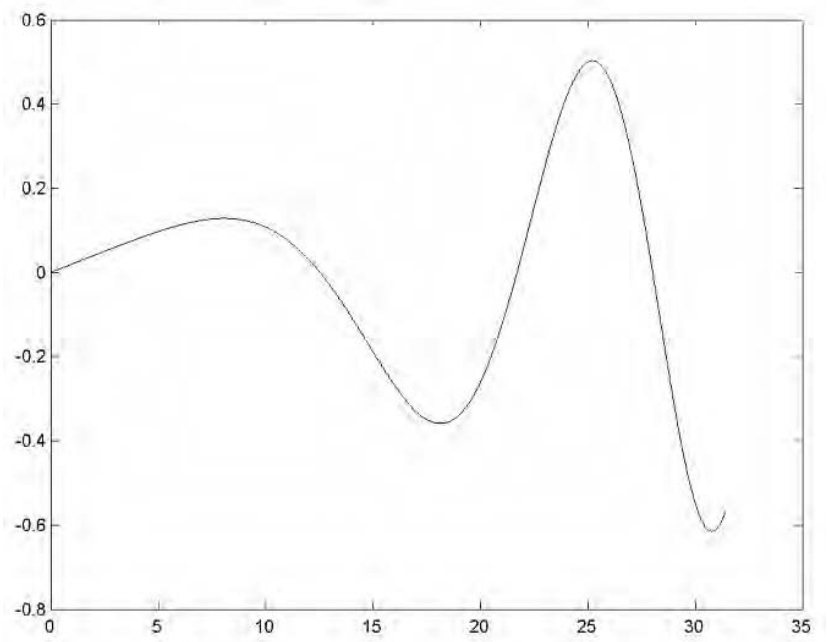


Рис. 6.1. График производной функции $f(x) = \sin(0.01x^2)$

Аналогичным образом поступают при вычислении производных высших порядков.

Отметим, что для аппроксимации производных конечными разностями в пакете MATLAB имеется функция **`diff()`**, синтаксис которой имеет следующий вид:

- `diff(X)` — возвращает конечные разности, вычисленные по смежным элементам вектора `X`. Длина вектора, возвращаемого функцией `diff(X)`, на единицу меньше длины вектора `X`. Если `X` — матрица, то возвращает матрицу, содержащую конечные разности, вычисленные по каждому столбцу;
- `diff(X,n)` — возвращает конечные разности n-го порядка;
- `diff(X,n,dim)` — возвращает конечные разности n-го порядка по столбцам (`dim=1`) или строкам матрицы.

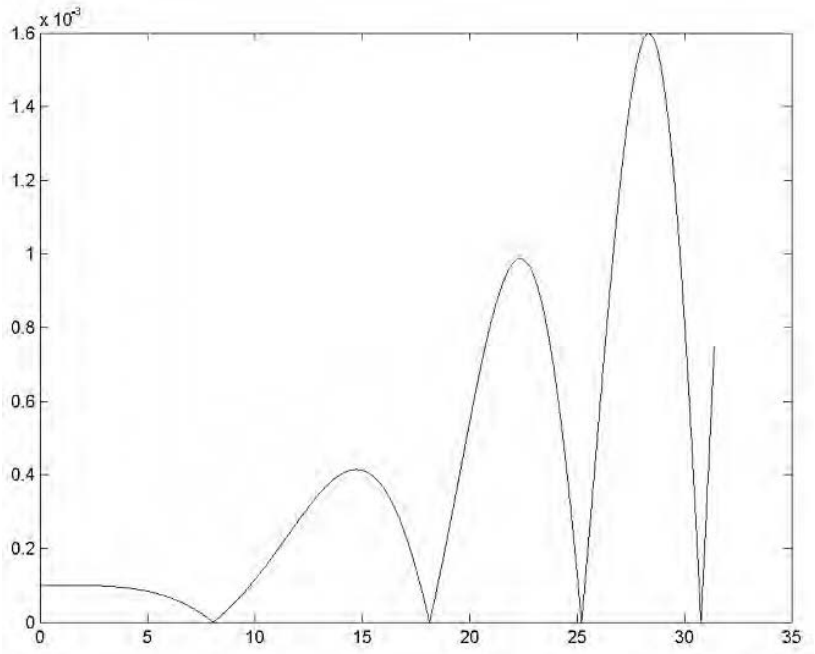


Рис. 6.2. Разность между точными и численными значениями производных функции $f(x) = \sin(0.01x^2)$

6.2. Особенности задачи численного дифференцирования функций, заданных таблицей

Пусть известны табличные значения функции $f(x)$ в конечном числе точек отрезка $[a,b]$. Требуется определить значение производной в некоторой точке отрезка $[a,b]$.

Для вычисления значений функции, заданной таблично, в лекции № 5 мы строили интерполяционный полином $F_n(x)$, продифференцировав который, можно получить значение производной

$$f'(x) \approx F_n'(x). \tag{6.8}$$

Полагая, что погрешность интерполирования определяется формулой

$$R_n(x) = f(x) - F_n(x), \tag{6.9}$$

находим оценку погрешности вычисления полинома

$$r_n(x) = f'(x) - F_n'(x). \tag{6.10}$$

Отметим, что задача численного дифференцирования является некорректной, т. к. погрешность производной полинома может существенно превышать погрешность интерполяции.

6.3. Интегрирование функций, заданных аналитически

С геометрической точки зрения определенный интеграл

$$F = \int_a^b f(x)\,dx \tag{6.11}$$

— есть площадь фигуры, ограниченная графиком функции $f(x)$ и прямыми $x = a$, $x = b$ (рис. 6.3).

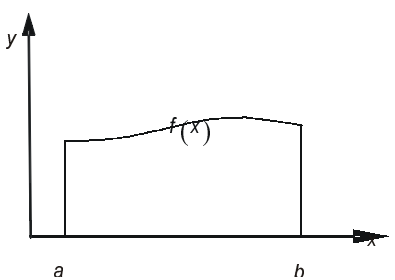


Рис. 6.3. К объяснению геометрического смысла определенного интеграла

Разделим отрезок $[a, b]$ на N равных отрезков длиной Δx, где

$$\Delta x = \frac{b-a}{N}. \tag{6.12}$$

Тогда координата правого конца i-го отрезка определяется по формуле

$$x_i = x_0 + i\Delta x\,, \tag{6.13}$$

где $x_0 = a$, $i = 0, 1, \ldots, N$.

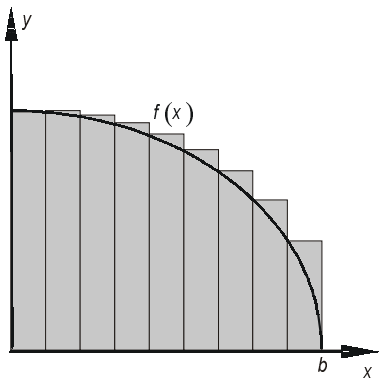


Рис. 6.4. Геометрическая интерпретация метода левых прямоугольников

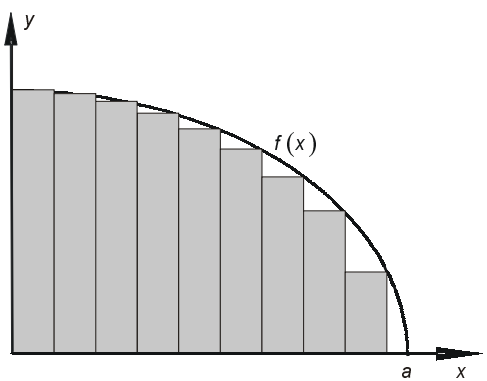


Рис. 6.5. Геометрическая интерпретация метода правых прямоугольников

Простейшая оценка площади под кривой $f(x)$ может быть получена как сумма площадей прямоугольников, одна из сторон которого совпадает с отрезком $[x_i, x_{i+1}]$, а высота равна значению функции в точке x_i (метод левых прямоугольников, рис. 6.4) или в точке x_{i+1} (метод правых прямоугольников, рис. 6.5). Погрешность вычисления значения интеграла на каждом шаге показана на рисунках закрашенными фигурами.

Значение определенного интеграла вычисляется по формулам

$$F_L = \sum_{i=0}^{N-1} f(x_i)\Delta x\,, \tag{6.14}$$

$$F_R = \sum_{i=1}^{N} f(x_i)\Delta x \tag{6.15}$$

для методов левых и правых прямоугольников, соответственно.

Можно повысить точность вычисления определенного интеграла, если заменять реальную функцию на каждом интервале $[x_i, x_{i+1}]$, $i = 0, 1, \ldots, N-1$ отрезком прямой, проходящей через точки с координатами $(x_i, f(x_i)), (x_{i+1}, f(x_{i+1}))$ — линейная интерполяция.

В этом случае фигура, ограниченная графиком функции и прямыми $x = x_i$, $x = x_{i+1}$, является трапецией. Искомый определенный интеграл определяется как сумма площадей всех трапеций:

$$F_N = \sum_{i=0}^{N-1} \frac{1}{2}(f(x_{i+1}) + f(x_i))\Delta x = \left[\frac{1}{2} f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2} f(x_N)\right]\Delta x\,. \tag{6.16}$$

Более высокая точность вычисления интегралов обеспечивается при использовании параболической интерполяции (полиномом второй степени) по трем соседним точкам:

$$y = ax^2 + bx + c. \tag{6.17}$$

Для нахождения коэффициентов a, b, c полинома, проходящего через точки (x_0, y_0), (x_1, y_1), (x_2, y_2), нужно найти решение следующей системы линейных уравнений:

$$\begin{cases} y_0 = ax_0^2 + bx_0 + c, \\ y_1 = ax_1^2 + bx_1 + c, \\ y_2 = ax_2^2 + bx_2 + c, \end{cases} \tag{6.18}$$

относительно неизвестных a, b, c.

Решив систему (6.18) относительно неизвестных a, b, c любым известным методом (например, Крамера), подставив найденные выражения в (6.17) и выполнив элементарные преобразования, получаем

$$y = y_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + y_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_2)(x_1-x_2)} + y_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}\,. \tag{6.19}$$

Отметим, что формула (6.19) совпадает с интерполяционным полиномом Лагранжа, подробно рассмотренным в *лекции 5*, при интерполяции по трем точкам.

Площадь под параболой $y = y(x)$ на интервале $[x_0, x_2]$ находится посредством элементарного интегрирования (6.19):

$$F_0 = \frac{1}{3}(y_0 + 4y_1 + y_2)\Delta x, \tag{6.20}$$

где $\Delta x = x_1 - x_0 = x_2 - x_1$.

Искомый определенный интеграл находится как площадь всех параболических сегментов (формула Симпсона):

$$F_N = \frac{1}{3}\Big[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + \\ + \ldots + 2f(x_{N-2}) + 4f(x_{N-1}) + f(x_N)\Big]\Delta x. \tag{6.21}$$

Обратите внимание на то обстоятельство, что в формуле Симпсона N должно быть четным числом.

Пример 6.2. Вычислить интеграл $\int_0^{\pi/2} \sin(x)\,dx$ методом прямоугольников в пакете MATLAB.

Решение:

```
>> f=inline('sin(x)'); % задание подынтегральной функции
>> Xmin=0;
>> Xmax=pi/2;
>> N=2001;
>> i=1:N;
>> dx=(Xmax-Xmin)/(N-1); % шаг интегрирования
>> x=Xmin:dx:Xmax; % вычисление координат узлов сетки
>> y=feval(f,x); % вычисление значений функции в узлах сетки
% вычисление интеграла по формуле правых прямоугольников
>> m=2:N;
>> y1(m-1)=y(m);
>> Fr=sum(y1)*dx
Fr =
    1.0004
>> Fr-1
ans =
  3.9284e-004
% вычисление интеграла по формуле левых прямоугольников
>> m=1:N-1;
```

```
>> y1(m)=y(m);
>> Fl=sum(y1)*dx
Fl =
    0.9996
>> Fl-1
ans =
 -3.9295e-004
% вычисление интеграла методом трапеций
>>  s=0;
7       for i=2:N-1
         s=s+y(i);
       end;
       Ft=(0.5*y(1)+s+0.5*y(N))*dx
       Ft =
        1.0000
>> Ft-1
ans =
 -5.1456e-008
% вычисление интеграла методом Симпсона
>> s=0;
       for i=2:N-1
          if i-2*ceil(i/2)==0
            k=4;
          else
            k=2;
          end;
          s=s+k*y(i);
       end;
       Fs=(y(1)+s+y(N))*dx/3
       Fs =
         1.0000
>> Fs-1
ans =
  1.5543e-015
```

Для вычисления значений определенных интегралов в пакете MATLAB есть функции **quad()**, **quadl()**, **trapz()**, **cumtrapz()**, которые имеют следующий синтаксис:

- `q = quad(fun,a,b)` — возвращает значение интеграла от функции `fun` на интервале `[a, b]`, при вычислении используется адаптивный метод Симпсона;
- `q = quad(fun,a,b,tol)` — возвращает значение интеграла от функции `fun` с заданной относительной погрешностью `tol` (по умолчанию `tol=10^-3`);

- `q = quad(fun,a,b,tol,trace)` — возвращает значение интеграла от функции `fun` на интервале `[a, b]` на каждом шаге итерационного процесса;
- `q = quad(fun,a,b,tol,trace,p1,p2,...)` — возвращает значение интеграла от функции `fun` на на интервале `[a, b]` на каждом шаге итерационного процесса, `p1`, `p2` — параметры, передаваемые в функцию `fun`;
- `[q,fcnt] = quadl(fun,a,b,...)` — возвращает в переменную `fcnt` дополнительно к значению интеграла число выполненных итераций.

Функция **`quadl()`** возвращает значения интеграла, используя для вычислений метод Лоббато (Lobbato). Синтаксис этой функции аналогичен синтаксису функции `quad()`.

Пример 6.3. Вычислить интеграл $\int_0^{\pi/2} \sin(x)dx$ с использованием функций `quard()`.

Решение:

```
>> q=quad('sin',0,pi/2,10^-4)
q =
    1.0000
>> q-1
ans =
 -3.7216e-008

>> q=quad('sin',0,pi/2,10^-6,'trace');
       9      0.0000000000    4.26596866e-001     0.0896208493
      11     0.4265968664    7.17602594e-001     0.4966040522
      13     0.4265968664    3.58801297e-001     0.2032723690
      15     0.7853981634    3.58801297e-001     0.2933317183
      17     1.1441994604    4.26596866e-001     0.4137750613

>> q-1
ans =
 -2.1269e-009

>> [q,fnct]=quad('sin',0,pi/2,10^-6,'trace');
       9      0.0000000000    4.26596866e-001     0.0896208493
      11     0.4265968664    7.17602594e-001     0.4966040522
      13     0.4265968664    3.58801297e-001     0.2032723690
      15     0.7853981634    3.58801297e-001     0.2933317183
      17     1.1441994604    4.26596866e-001     0.4137750613
>> 17
```

Функция **trapz()** вычисляет интеграл, используя метод трапеций. Синтаксис этой функции следующий:

- `Z = trapz(Y)` — возвращает значение определенного интеграла в предположении, что `X=1:length(Y)`;
- `Z = trapz(X,Y)` — возвращает значение интеграла на интервале `[X(1),X(N)]`;
- `Z = trapz(X,Y,dim)` — интегрирует вектор `Y`, формируемый из чисел, расположенных в размерности `dim` многомерного массива.

Функция **cumtrapz()**, вычисляет интеграл, как функцию с переменным верхним пределом. Синтаксис функции **cumtrapz()** аналогичен синтаксису функции **trapz()**.

Пример 6.4. Вычислить определенный интеграл $\int_0^{\pi/2} \sin(x)dx$ с использованием встроенной функции пакета MATLAB.

Решение:

```
>> x=0:0.01:pi/2;  % задание координат узловых точек
>> y=sin(x); % вычисление значений подынтегральной функции в
             % узловых точках
>> trapz(y) % вычисление значения интеграла в предположении о
            % том, что шаг интегрирования равен единице
ans =
   99.9195

>> trapz(x,y) % вычисление значения интеграла на отрезке [0, π/2]
              % с шагом интегрирования 0.01
ans =
    0.9992
```

Пример 6.5. Вычислить интеграл с переменным верхним пределом $\int_0^x \sin(\xi)d\xi$ на интервале [0, 3π/2].

Решение:

```
>> x=0:0.01:3*pi/2; % задание координат узловых точек
>> y=sin(x); % вычисление значений подынтегральной функции в
             % узловых точках
>> z=cumtrapz(x,y); % вычисление значений интеграла с
                    % переменным  верхним пределом в  узловых точках

>> plot(x,y,x,z) % построение графиков подынтегральной функции и
                 % интеграла с переменным верхним пределом (рис. 6.6)
```

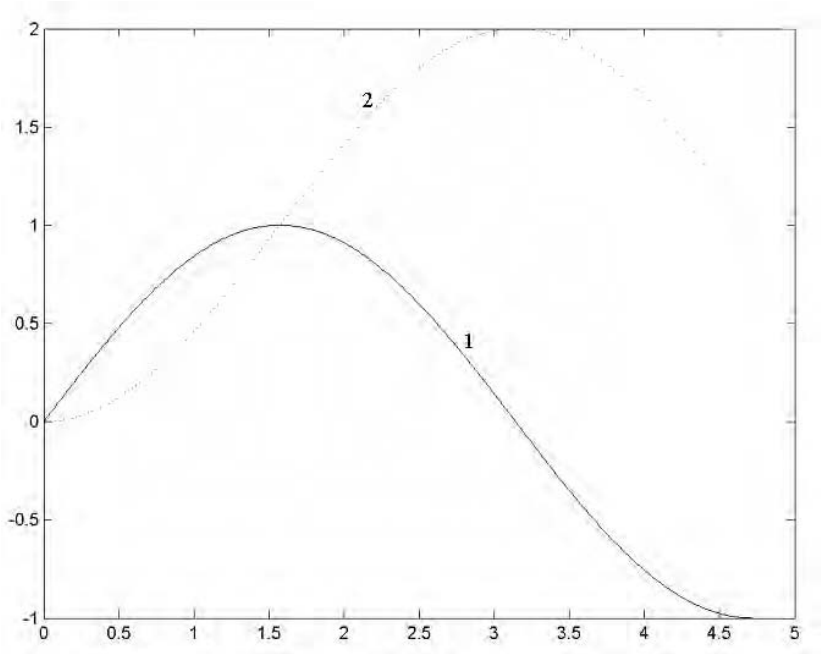


Рис. 6.6. Графики функций $f(x)=\sin(x)$ (**1**), $\varphi(x)=\int_0^x \sin(\xi)d\zeta$ (**2**)

6.4. Погрешность численного интегрирования

Для нахождения зависимостей погрешности вычисления определенного интеграла на отрезке $[a,b]$ от числа отрезков разбиения интервала интегрирования разложим подынтегральную функцию в ряд Тейлора:

$$f(x)=f(x_i)+f'(x_i)(x-x_i)+\frac{1}{2}f''(x_i)(x-x_i)^2+\ldots \qquad (6.22)$$

Тогда интеграл от данной функции на отрезке $[x_i,x_{i+1}]$ будет равен:

$$\int_{x_i}^{x_{i+1}} f(x)dx=f(x_i)\Delta x+\frac{1}{2}f'(x_i)(\Delta x)^2+\frac{1}{6}f''(x_i)(\Delta x)^3+\ldots \qquad (6.23)$$

Оценим погрешность метода левых прямоугольников. Погрешность интегрирования Δ_i на отрезке $[x_i,x_{i+1}]$ равняется разности между точным значением интеграла и его оценкой $f(x_i)\Delta x$:

$$\Delta_i = \left[\int_{x_i}^{x_{i+1}} f(x)dx\right] - f(x_i)\Delta x \approx \frac{1}{2} f'(x_i)(\Delta x)^2 . \qquad (6.24)$$

Из (6.24) видно, что основной член погрешности на каждом отрезке имеет порядок $(\Delta x)^2$ или в символической записи $O\left((\Delta x)^2\right)$. Поскольку полное число отрезков равно N, а $\Delta x = (b-a)/N$, то полная погрешность метода левых прямоугольников по порядку величины равна $N\Delta_i \approx N(\Delta x)^2 \approx O\left(N^{-1}\right)$. Аналогично можно показать, что погрешность метода правых прямоугольников также пропорциональна N^{-1}.

Погрешность формулы трапеций оценивается аналогичным образом. Так как значение интеграла на отрезке $[x_i, x_{i+1}]$ вычисляется по формуле $[f(x_i) + f(x_{i+1})]\Delta x/2$, то погрешность равна:

$$\Delta_i = \left[\int_{x_i}^{x_{i+1}} f(x)\,dx\right] - \frac{1}{2}\left[f(x_i) + f(x_{i+1})\right]\Delta x . \qquad (6.25)$$

Заменив в (6.25) первый член выражением (6.23), значение функции в точке x_{i+1} будет разложением в ряд Тэйлора:

$$f(x_{i+1}) = f(x_i) + f'(x_i)\Delta x + \frac{1}{2} f''(x_i)(\Delta x)^2 + \ldots$$

Раскрыв скобки и приведя подобные члены, обнаруживаем, что член, пропорциональный первой производной функции, сокращается, и погрешность на одном отрезке равна $\frac{1}{4} f''(x)(\Delta x)^3 \approx O\left((\Delta x)^3\right) \approx O\left(N^{-3}\right)$. Следовательно, полная погрешность формулы трапеций на отрезке $[a,b]$ по порядку величины равна $O\left(N^{-2}\right)$.

Так как формула Симпсона основывается на приближении функции $f(x)$ параболой, можно ожидать, что в данном случае погрешность по порядку величины будет определяться членами, пропорциональными третьей производной функции. Однако последовательное повторение действий, выполненных при оценке погрешности метода трапеций, показывает, что эти члены сокращаются в силу их симметричности, поэтому в разложении в ряд Тейлора следует удержать член, пропорциональный $f'''(x)(\Delta x)^4$. Следовательно, погрешность формулы Симпсона на отрезке $[x_i, x_{i+1}]$ пропорциональна $f''''(x_i)(\Delta x)^5$, а полная погрешность на отрезке $[a,b]$ по порядку величины составляет $O\left(N^{-4}\right)$.

Полезно получить оценку погрешности вычисления интеграла от функции, зависящей от двух переменных, который с геометрической точки зрения представляет собой объем фигуры под поверхностью, заданной функцией $f(x,y)$. В прямоугольном приближении данный интеграл равен сумме объемов параллелепипедов с площадью основания $\Delta x\Delta y$ и высотой, равной значению функции $f(x,y)$ в одном из углов. Для определения погрешности разложим функцию $f(x,y)$ в ряд Тейлора:

$$f(x,y)=f(x_i,y_i)+f'_x(x_i,y_i)(x-x_i)+f'_y(x_i,y_i)(y-y_i)+\dots, \tag{6.26}$$

где f'_x, f'_y — частные производные по соответствующим переменным.

Погрешность вычисления интеграла Δ_i равна

$$\Delta_i=\iint f\left(x,y\right)dxdy-f\left(x_i,y_i\right)\Delta x\Delta y\,. \tag{6.27}$$

Подставив (6.26) в (6.27), выполнив интегрирование и приведя подобные члены, получаем, что член, пропорциональный $f(x_i,y_i)$, сокращается, а интеграл от $(x-x_i)dx$ дает $(\Delta x)^2/2$. Интеграл от данного выражения по dy дает еще один множитель Δy. Аналогичный вклад дает интеграл от члена, пропорционального $(y-y_i)$. Так как порядок погрешности Δy также составляет $O(\Delta x)$, то погрешность интегрирования по прямоугольнику $x_i\le x\le x_{i+1}$, $y_i\le y\le y_{i+1}$ равна

$$\Delta_i\approx\frac{1}{2}\left[f'_x(x_i,y_i)+f'_y(x_i,y_i)\right](\Delta x)^3. \tag{6.28}$$

Из (6.28) видно, что погрешность интегрирования по одному параллелепипеду составляет $O\left((\Delta x)^3\right)$. Так как имеется N параллелепипедов, полная погрешность по порядку величины равна $N(\Delta x)^3$. Однако в двумерном случае $N\sim\dfrac{1}{(\Delta x)^2}$, поэтому полная погрешность $\Delta_i\sim(\Delta x)\sim O\left(N^{-1/2}\right)$. Напомним, что в одномерном случае полная погрешность метода прямоугольников $\Delta_i\sim O\left(N^{-1}\right)$.

Аналогичные оценки для двумерных обобщений формул трапеций и Симпсона показывают, что они соответственно равны $O\left(N^{-1}\right)$ и $O\left(N^{-2}\right)$. Вообще можно показать, что если для одномерного случая погрешность составляет $O\left(N^{-\alpha}\right)$, то в d-мерном случае она равна $O\left(N^{-\alpha/d}\right)$.

6.5. Вычисление интегралов методом Монте-Карло

Проиллюстрируем идеи метода Монте-Карло на примере вычисления определенного интеграла от функции, зависящей от одной переменной. Пусть нам необходимо вычислить интеграл (6.11) от некоторой заданной функции $f(x)$ на интервале $[a,b]$. В предыдущем разделе мы рассмотрели несколько различных формул интегрирования, в которых использовались значения функции $f(x)$, вычисляемые в равноотстоящих точках. Однако можно использовать и другой подход, суть которого легко понять из следующего примера.

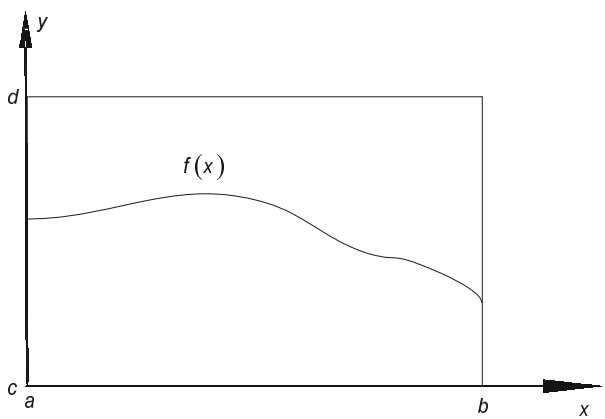


Рис. 6.7. К объяснению метода Монте-Карло

Представим себе прямоугольник высотой H и длиной $(b - a)$, такой, что функция $f(x)$ целиком лежит внутри данного прямоугольника (рис. 6.7). Сгенерируем N пар случайных чисел, равномерно распределенных в данном прямоугольнике:

$$a \le x_i \le b, 0 \le y_i \le H . \tag{6.29}$$

Тогда доля точек (x_i, y_i), удовлетворяющих условию $y_i \le f(x_i)$, является оценкой отношения интеграла от функции $f(x)$ к площади рассматриваемого прямоугольника. Следовательно, оценка интеграла в данном методе может быть получена по формуле:

$$F_N = A\frac{n_s}{N}, \tag{6.30}$$

где n_s — число точек, удовлетворяющих условию $y_i \le f(x_i)$, N — полное количество точек, A — площадь прямоугольника.

Можно предложить и другой путь вычисления определенного интеграла, рассматривая его как среднее значение функции $f(x)$ на отрезке $[a,b]$:

$$F_N = (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i), \tag{6.31}$$

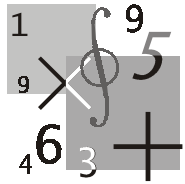
где x_i — последовательность случайных чисел с равномерным законом распределения на отрезке $[a,b]$.

Отметим, что в отличие от ранее упомянутых методов погрешность метода Монте-Карло не зависит от размерности и меняется как $O\left(N^{-1/2}\right)$. Следовательно, для достаточно больших d интегрирование по методу Монте-Карло будет приводить к меньшим погрешностям при тех же значениях N.

Пример 6.6. Вычисление интеграла $\int_0^{\pi/2} \sin(x)\,dx$ методом Монте-Карло в пакете MATLAB.

Решение:

```
% задание координат вершит прямоугольника
>> Xmin=0;
>> Xmax=pi/2;
>> Ymin=0;
>> Ymax=1.5;
% генерация случайных координат
>> N=2000;
>> x=Xmin+(Xmax-Xmin)*rand(N,1);
>> y=Ymin+(Ymax-Ymin)*rand(N,1);
% подсчет  числа точек, попавших под график функции
>> s=0;
>> for i=1:N
        if y(i)<=feval(f,x(i))
          s=s+1;
        end;
     end;
>> s*(Xmax-Xmin)*(Ymax-Ymin)/N % вычисление значения интеграла
ans =
    1.0261
% вычисление интеграла в соответствие с (6.31)
>> Fr=feval(f,x);
>> (Xmax-Xmin)/N*sum(Fr)
ans =
    1.0091
```



Лекция 7

Методы обработки экспериментальных данных

Цель лекции: изложить подход к решению задачи о среднеквадратичном приближении функции, заданной таблично, рассмотреть аппроксимацию элементарными функциями, линейными комбинациями элементарных функций и функциями произвольного вида, рассмотреть реализацию данных методов в пакете MATLAB.

7.1. Метод наименьших квадратов

Пусть в результате измерений в процессе опыта получена таблица некоторой зависимости $f(x)$ (табл. 7.1).

Таблица 7.1. *Исходные данные для метода наименьших квадратов*

x	x_1	x_2	...	x_n
$F(x)$	y_1	y_2	...	y_n

Требуется найти формулу, выражающую данную зависимость аналитически.

Один из подходов к решению данной задачи состоит в построении интерполяционного многочлена, значения которого будут в точках x_1, x_2,..., x_n совпадать с соответствующими значениями $f(x)$ из табл. 7.1. Однако совпадение значений в узлах может вовсе не означать совпадения характеров исходной и интерполирующей функций. Требование неукоснительного совпадения значений тем более не оправдано, если значения функций $f(x)$ известны с некоторой погрешностью (рис. 7.1).

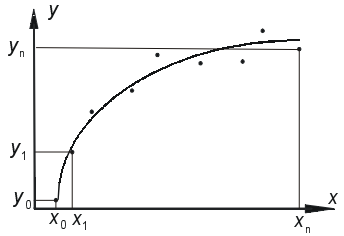


Рис. 7.1. К объяснению метода наименьших квадратов

Поставим задачу так, чтобы с самого начала обязательно учитывался характер исходной функции: найти функцию заданного вида

$$y = F(x), \tag{7.1}$$

которая в точках x_1, x_2,..., x_n принимает значения как можно более близкие к табличным значениям y_1, y_2,..., y_n.

Следует отметить, что строгая функциональная зависимость для табл. 7.1. наблюдается редко, т. к. каждая из входящих в нее величин может зависеть от многих случайных факторов, поэтому обычно используют простые по виду аналитические функции.

Рассмотрим один из наиболее распространенных способов нахождения функции $F(x)$. Предположим, что приближающая функция $F(x)$ в точках x_1, x_2,..., x_n имеет значения

$$\bar{y}_1,\ \bar{y}_2,\ldots,\ \bar{y}_n. \tag{7.2}$$

Требование близости табличных значений y_1, y_2,..., y_n и значений (7.2) можно истолковать таким образом. Будем рассматривать совокупность значений функции $f(x)$ из табл. 7.1 и совокупность значений (7.2) как координаты двух точек n-мерного пространства. С учетом этого задача приближения функции может быть переформулирована следующим образом: найти такую функцию $F(x)$ заданного вида, чтобы расстояние между точками $M(y_1, y_2, \ldots, y_n)$ и $\overline{M}(\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_n)$ было наименьшим. Воспользовавшись метрикой Евклидова пространства, приходим к требованию, чтобы величина

$$\sqrt{(y_1-\bar{y}_1)^2+(y_2-\bar{y}_2)^2+\ldots+(y_n-\bar{y}_n)^2}, \tag{7.3}$$

была наименьшей. Это равносильно следующему: сумма квадратов

$$(y_1-\bar{y}_1)^2+(y_2-\bar{y}_2)^2+\ldots+(y_n-\bar{y}_n)^2 \tag{7.4}$$

должна быть наименьшей.

Приведем окончательную формулировку задачи приближения функции $f(x)$: для функции $f(x)$, заданной табл. 7.1, найти функцию $F(x)$ определенного вида так, чтобы сумма квадратов (7.4) была наименьшей. Эта задача называется приближением функции методом наименьших квадратов. В качестве приближающих функций в зависимости от характера точечного графика $f(x)$ часто используют функции, представленные далее. (Здесь a, b, m — неизвестные параметры.)

$y = ax + b$	$y = \dfrac{1}{ax + b}$
$y = ax^2 + bx + c$	$y = a \ln x + b$
$y = ax^m$	$y = a\dfrac{1}{x} + b$
$y = ae^{mx}$	$y = \dfrac{x}{ax + b}$

Когда вид приближающей функции установлен, задача сводится к отысканию значений параметров.

Рассмотрим метод нахождения параметров приближающей функции в общем виде на примере приближающей функции, зависящей от трех параметров:

$$y = F(x, a, b, c). \tag{7.5}$$

Имеем:

$$F(x_i, a, b, c) = \bar{y}_i. \tag{7.6}$$

Сумма квадратов разностей соответствующих значений функций $f(x)$ и $F(x)$ имеет вид:

$$\sum_{i=1}^{n} (y_i - F(x_i, a, b, c))^2 = \Phi(a, b, c). \tag{7.7}$$

Сумма является функцией $\Phi(a, b, c)$ трех переменных. Используя необходимое условие экстремума:

$$\frac{\partial \Phi}{\partial a} = 0, \frac{\partial \Phi}{\partial b} = 0, \frac{\partial \Phi}{\partial c} = 0,$$

получаем систему уравнений:

$$\sum_{i=1}^{n}[y_i - F(x_i, a, b, c,)]\cdot F'_a(x_i, a, b, c) = 0\ ,$$

$$\sum_{i=1}^{n}[y_i - F(x_i, a, b, c,)]\cdot F'_b(x_i, a, b, c) = 0\ , \qquad (7.8)$$

$$\sum_{i=1}^{n}[y_i - F(x_i, a, b, c,)]\cdot F'_c(x_i, a, b, c) = 0\ .$$

Решив систему (7.8) относительно параметров a, b, c, получаем конкретный вид функции $F(x, a, b, c)$. Изменение количества параметров не приведет к изменению сути самого подхода, а выразится в изменении количества уравнений в системе (7.8).

Значения разностей

$$y_i - F(x_i, a, b, c) = \varepsilon_i \qquad (7.9)$$

называют отклонениями измеренных значений от вычисленных по формуле (7.5).

Сумма квадратов отклонений

$$\sigma = \sum_{i}^{n}\varepsilon_i^2 \qquad (7.10)$$

в соответствии с принципом наименьших квадратов для заданного вида приближающей функции должна быть наименьшей.

Из двух разных приближений одной и той же табличной функции лучшим считается то, для которого (7.10) имеет наименьшее значение.

7.2. Нахождение приближающей функции в виде линейной функции и квадратичного трехчлена

Ищем приближающую функцию в виде:

$$F(x, a, b) = ax + b\ . \qquad (7.11)$$

Находим частные производные

$$\frac{\partial F}{\partial a} = x, \frac{\partial F}{\partial b} = 1\ . \qquad (7.12)$$

Составляем систему вида (7.8)

$$\sum (y_i - ax_i - b)x_i = 0,$$

$$\sum (y_i - ax_i - b) = 0.$$

Примечание

Здесь и далее сумма берется по переменной $i = 1, 2, \ldots, n$.

Имеем:

$$\sum x_i y_i - a\sum x_i^2 - b\sum x_i = 0\,,$$

$$\sum y_i - a\sum x_i^2 - bn = 0. \qquad (7.13)$$

Разделив каждое уравнение (7.13) на n, получаем:

$$\left(\frac{1}{n}\sum x_i^2\right)\cdot a + \left(\frac{1}{n}\sum x_i\right)\cdot b = \frac{1}{n}\sum x_i y_i\,,$$

$$\left(\frac{1}{n}\sum x_i^2\right)\cdot a + b = \frac{1}{n}\sum y_i\,.$$

Введем обозначения:

$$\frac{1}{n}\sum x_i = M_x, \frac{1}{n}\sum y_i = M_y\,,$$

$$\frac{1}{n}\sum x_i y_i = M_{xy}, \frac{1}{n}\sum x_i^2 = M_{x^2}\,.$$

Тогда последняя система будет иметь вид:

$$M_{x^2}\cdot a + M_x \cdot b = M_{xy},$$

$$M_x \cdot a + b = M_y$$

или в матричной форме:

$$\begin{pmatrix} M_{x^2} & M_x \\ M_x & 1 \end{pmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} M_{xy} \\ M_y \end{pmatrix}.$$

Откуда:

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} M_{x^2} & M_x \\ M_x & 1 \end{pmatrix}^{-1}\begin{pmatrix} M_{xy} \\ M_y \end{pmatrix}. \qquad (7.14)$$

Вычислив значения параметров a, b в соответствие с (7.14), получаем конкретные значения и, следовательно, конкретный вид линейной функции (7.11).

В случае нахождения приближающей функции в форме квадратного трехчлена имеем:

$$F(x,a,b,c)=ax^2+bx+c\,. \tag{7.15}$$

Находим частные производные:

$$\frac{\partial F}{\partial a}=x^2,\frac{\partial F}{\partial b}=b,\frac{\partial F}{\partial c}=1.$$

Составляем систему вида (7.8):

$$\sum\left(y_i-ax_i^2-bx_i-c\right)x_i^2=0\,,$$

$$\sum\left(y_i-ax_i^2-bx_i-c\right)x_i=0\,,$$

$$\sum\left(y_i-ax_i^2-bx_i-c\right)=0\,.$$

Далее имеем:

$$\sum y_ix_i^2-a\sum x_i^4-b\sum x_i^3-c\sum x_i^2=0\,,$$

$$\sum y_ix_i-a\sum x_i^3-b\sum x_i^2-c\sum x_i=0\,,$$

$$\sum y_i-a\sum x_i^2-b\sum x_i-cn=0\,.$$

Разделив каждое уравнение на n и перенеся члены, не содержащие неизвестные параметры в правую часть, получаем:

$$\left(\frac{1}{n}\sum x_i^4\right)a+\left(\frac{1}{n}\sum x_i^3\right)b+\left(\frac{1}{n}\sum x_i^2\right)c=\frac{1}{n}\sum y_ix_i^2\,,$$

$$\left(\frac{1}{n}\sum x_i^3\right)a+\left(\frac{1}{n}\sum x_i^2\right)b+\left(\frac{1}{n}\sum x_i\right)c=\frac{1}{n}\sum y_ix_i\,, \tag{7.16}$$

$$\left(\frac{1}{n}\sum x_i^2\right)a+\left(\frac{1}{n}\sum x_i\right)b+c=\frac{1}{n}\sum y_i\,.$$

Решив систему (7.16) относительно неизвестных *a*, *b*, c, находим значения параметров приближающей функции.

Пример 7.1. Даны табличные значения линейной зависимости вида $y=0.2x$, каждому из которых добавлены случайные числа, имеющие равномерный закон распределения на интервале [–0.1, 0.2]. Найти коэффициенты линейной и квадратичной аппроксимирующих функций.

Решение:

```
% задание исходных данных
>> N=10;
>> i=1:N;
>> Xmin=0;
```

```
>> Xmax=10;
>> x(i)=Xmin+(Xmax-Xmin)/(N-1)*(i-1);
>> y(i)=0.2*x(i); % точные значения функции
% задание шума с равномерным законом распределения на
% отрезке [b,a]
>> a=0.2
>> b=-0.1;
>> Yrnd=b+(a-b)*rand(N,1);
>> y1=y+Yrnd'; %  создание зашумленных данных
>> plot(x,y,x,y1,'o'); % визуализация точной и зашумленной
                       % последовательностей (рис. 7.2)
```

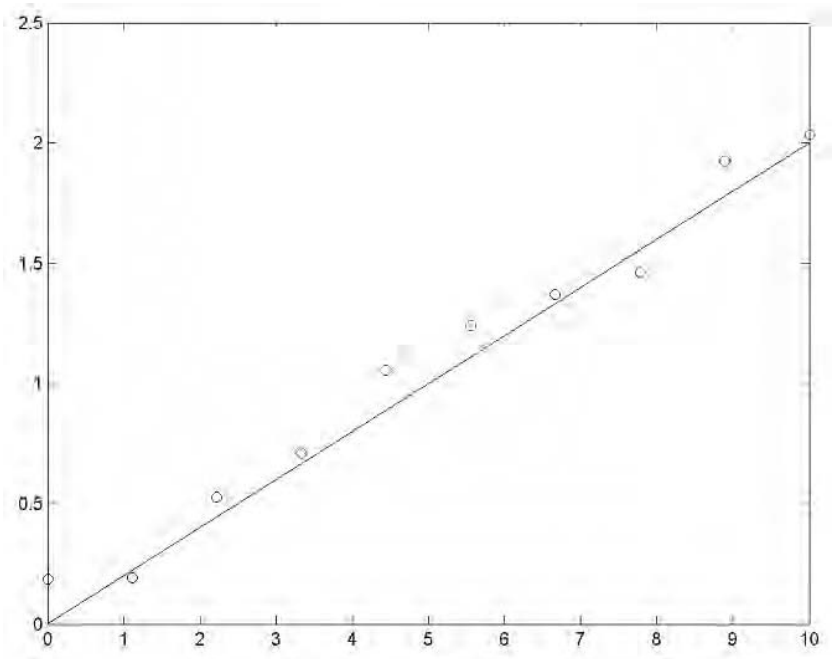


Рис. 7.2. Исходные данные и точная зависимость $y = 0.2x$

```
% вычисление элементов матрицы M в (7.14)
>>  tmp=x(i).^2;
>> M(1,1)=1/N*sum(tmp);
>> M(1,2)=1/N*sum(x);
>> M(2,1)=M(1,2);
>> M(2,2)=1;
% вычисление элементов вектора d
>> d(1,1)=1/N*dot(x,y1);
>> d(2,1)=1/N*sum(y1);
```

```
% решение системы линейных уравнений (7.14)
>> Coeff=A^-1*d;
Coeff =
    0.1941
    0.0993
>> F=inline('a*x+b','a','b','x'); % задание аппроксимирующей функции
>> tmp1(i)=feval(F,Coeff(1,1),Coeff(2,1),x(i)); % вычисление значений
                                                % аппроксимирующей
                                                % функции
% вычисление суммы квадратов отклонений
% при линейной аппроксимации
>> tmp=tmp1-y1;
>> dot(tmp,tmp)
ans =
    0.0685
% аппроксимация исходных данных полиномом второй степени
% задание матрицы системы линейных уравнений в (7.16)
>> A(1,1)=1/N*sum(x.^4);
>> A(1,2)=1/N*sum(x.^3);
>> A(1,3)=1/N*sum(x.^2);
>> A(2,1)=A(1,2);
>> A(2,2)=A(1,3);
>> A(2,3)=1/N*sum(x);
>> A(3,1)=A(2,2);
>> A(3,2)=A(2,3);
>> A(3,3)=1;
% задание вектора столбца свободных членов
>> d(1,1)=1/N*dot(x.^2,y1);
>> d(2,1)=1/N*dot(x,y1);
>> d(3,1)=1/N*sum(y1);
% решение системы линейных уравнений (7.16)
>> Coeff=A^-1*d
Coeff =
    0.0004
    0.1904
    0.1049
>> F=inline('a*x.^2+b*x+c','a','b','c','x'); % задание
                                          % аппроксимирующей функции
% вычисление суммы квадратов отклонений при квадратичной
% интерполяции
>> tmp2(i)=feval(F,Coeff(1,1),Coeff(2,1),Coeff(3,1),x(i));
>> tmp=tmp2-y1;
dot(tmp,tmp)
ans =
    0.0687
```

Найти решение рассмотренной ранее задачи в пакете MATLAB можно другим способом. Для этого следует использовать тот факт, что коэффициенты искомой функции, минимизирующей сумму квадратов отклонений, являются решением переопределенной системы уравнений. Для случая интерполяции полиномом второй степени данных, приведенных ранее, система уравнений имеет вид:

$$\begin{pmatrix} y1_1 \\ y1_2 \\ \vdots \\ y1_{10} \end{pmatrix} = \begin{pmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_{10} & t_{10}^2 \end{pmatrix} \times \begin{pmatrix} \mathrm{Coeff}_0 \\ \mathrm{Coeff}_1 \\ \mathrm{Coeff}_2 \end{pmatrix}.$$

Решение данной системы уравнений, удовлетворяющее методу наименьших квадратов, находится с помощью оператора \:

```
Coeff = A\y1,
```

где

$$y1 = \begin{pmatrix} y1_1 \\ y1_2 \\ \vdots \\ y1_{10} \end{pmatrix},$$

$$A = \begin{pmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_{10} & t_{10}^2 \end{pmatrix}.$$

Таким образом, альтернативный подход к нахождению коэффициентов аппроксимирующего полинома реализуется выполнением следующей последовательности команд:

```
>> B=[ones(size(x')) x' x'.^2]
B =
    1.0000         0         0
    1.0000    1.1111    1.2346
    1.0000    2.2222    4.9383
    1.0000    3.3333   11.1111
    1.0000    4.4444   19.7531
    1.0000    5.5556   30.8642
    1.0000    6.6667   44.4444
    1.0000    7.7778   60.4938
```

```
    1.0000    8.8889   79.0123
    1.0000   10.0000  100.0000
>> Coeff=B\y1
ans =
    0.1049
    0.1904
    0.0004
```

Линейную комбинацию известных функций пакета MATLAB можно применять с целью решения системы линейных уравнений методом наименьших квадратов с использованием аппроксимации. Для этого можно использовать описанный в предыдущем разделе метод.

Пример 7.2. Используя метод наименьших квадратов, найти коэффициенты функции $F(x,a,b,c) = a \cdot x^2 + b \cdot x + c \cdot 1/(x+1)$, аппроксимирующей данные, представленные в табл. 7.2.

Таблица 7.2. *Исходные данные примера 7.2*

x	0	0.2	0.4	0.6	0.8	1.0
y	0.43	0.22	0.8	0.12	1.0	2.0

Решение:

```
% Задание исходных данных
>> vx=[0;0.2;0.4;0.6;0.8;1]
vx =
         0
    0.2000
    0.4000
    0.6000
    0.8000
    1.0000
>> vy=[0.43;0.22;0.8;0.12;1;2]
vy =
    0.4300
    0.2200
    0.8000
    0.1200
    1.0000
    2.0000
% задание матрицы переопределенной системы уравнений
>> D=[vx.^2 vx 1./(vx+1)]
>> D
```

```
D =
         0              0        1.0000
    0.0400    0.2000    0.8333
    0.1600    0.4000    0.7143
    0.3600    0.6000    0.6250
    0.6400    0.8000    0.5556
    1.0000    1.0000    0.5000
>> Coeff=D\vy
Coeff =
    3.0521
   -1.4391
    0.5126
% визуализация исходных данных и аппроксимирующей функции
>> i=1:length(vx);
>> j=1:length(X);
>> X=vx(1):0.01:vx(6);
>> Y=[ones(size(X')) X' X'.^2]*Coeff; % вычисление значений
                                      % аппроксимирующей
                                      % функции
>> plot(vx(i),vy(i),'o',X(j),Y(j)) % визуализация исходных данных и
                                   % аппроксимирующей функции рис. 7.3
```

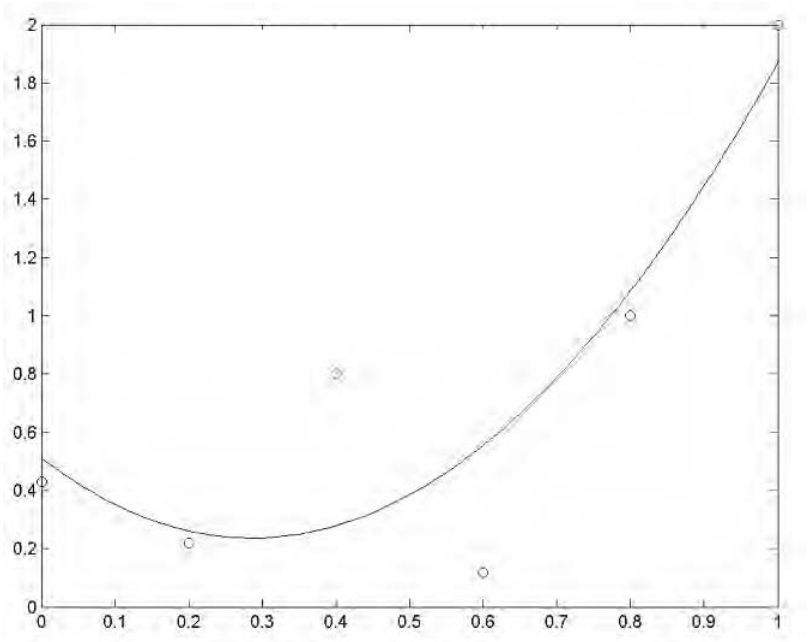


Рис. 7.3. Исходные данные и аппроксимирующая функция вида $F(x,a,b,c) = a \cdot x^2 + b \cdot x + c \cdot 1/(x+1)$

7.3. Аппроксимация функцией произвольного вида

Для решения задачи обобщенной нелинейной регрессии в пакете MATLAB имеется функция **lsqnonlin()**, возвращающая решение задачи нахождения точки минимума функции $f(x)$:

$$\min_x\left(f\left(x\right)\right)=f_1\left(x\right)^2+f_2\left(x\right)^2+\ldots f_m\left(x\right)^2+L\ ,$$

где в общем случае $f(x)$ — вектор-функция, x — вектор-столбец искомых переменных, L — некоторая константа.

Синтаксис функции `lsqnonlin()` имеет следующий вид:

```
x = lsqnonlin(fun,x0,eb,ub,options,P1,P2, ... )
   [x,resnorm,residual,exitflag,output,lambda,jacobian]=lsqnonlin(..
   )
```

Он аналогичен синтаксису функции `fsolve()`, подробно описанной в *лекции 4*. Поэтому далее мы ограничимся только примером, демонстрирующим использование данной функции для нахождения параметров функции $F\left(x,a,b,c\right)=\exp\left(a+b\cdot x+c\cdot x^2\right)$.

Пример 7.3. Найти, используя метод наименьших квадратов, для данных, представленных в табл. 7.3, коэффициенты аппроксимирующей функции $F\left(x,a,b,c\right)=\exp\left(a+b\cdot x+c\cdot x^2\right)$.

Таблица 7.3. *Исходные данные примера 7.3*

x	0.3	0.4	1.0	1.4	2.0	4.0
y	9.4	11.2	5.0	3.0	6.0	0.2

1. Создайте файл F77.m (листинг 7.1), содержащий описание функции, возвращающей значения вектор-функции $f(x)$.

Листинг 7.1. Файл F77.m

```
function z=F77(Coeff,vx,vy)
k=1:length(vx);
z=vy-exp(Coeff(1)+Coeff(2)*vx+Coeff(3)*vx.^2);
```

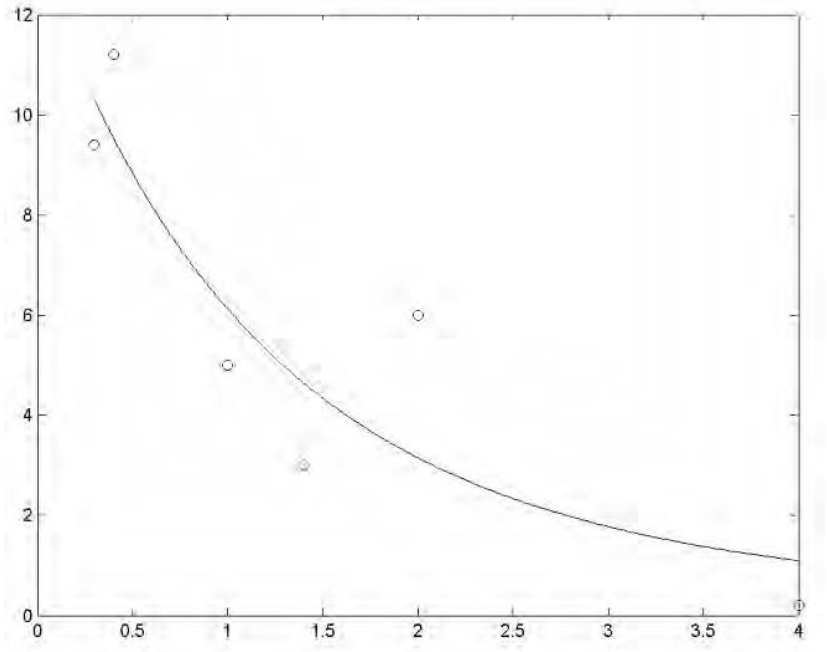
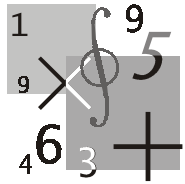


Рис. 7.4. Исходные данные и аппроксимирующая функция вида $F(x,a,b,c)=\exp\left(a+b\cdot x+c\cdot x^2\right)$

2. Выполните следующую последовательность команд:

```
% задание исходных данных
>> vx=[0.3;0.4;1;1.4;2;4]
vx =
    0.3000
    0.4000
    1.0000
    1.4000
    2.0000
    4.0000
>> vy=[9.4;11.2;5;3;6;0.2]
vy =
    9.4000
   11.2000
    5.0000
    3.0000
    6.0000
    0.2000
```

```
>> z=[1 0 -1] % начальное приближение
z =
     1     0    -1
% вычисление коэффициентов аппроксимирующей функции
>> Coeff = lsqnonlin('F77',z',[],[],[],vx,vy)
Optimization terminated successfully:
Relative function value changing by less than OPTIONS.TolFun
Coeff =
    2.5696
   -0.8037
    0.0462
>> F=inline('exp(a+b*x+c*x.^2)','x','a','b','c'); % задание
                                                   % аппроксимирующей функции
>> X=vx(1):0.01:vx(length(vx)); % координаты абсцисс, в которых
                                % будут вычисляться значения
                                % аппроксимирующей функции
>>  Y=feval(F,X,Coeff(1),Coeff(2),Coeff(3)); % вычисление значений
                                             % аппроксимирующей функции
>> i=1:length(vx);
>> j=1:length(X);
>> plot(vx(i),vy(i),'o',X(j),Y(j)) % визуализация исходных данных и
                                   % аппроксимирующей функции (рис. 7.4)
```



Лекция 8

Преобразование Фурье

Цель лекции: изложить основы Фурье-анализа периодических, непериодических и дискретных функций, обсудить эффект Гиббса, алгоритм быстрого преобразования Фурье, рассмотреть средства реализации спектрального анализа в пакете MATLAB.

8.1. Разложение периодических функций в ряд Фурье

По определению периодической функцией называют функцию, отвечающую условию:

$$s(t) = s(t + nT), \tag{8.1}$$

где T — период функции, $n = 1, 2, \ldots$

Для нахождения спектрального разложения функции $s(t)$ введем в рассмотрение следующие наборы функций:

$$\begin{aligned} &1/\sqrt{T}, \sqrt{2/T}\sin(2\pi\, t/T), \sqrt{2/T}\sin(4\pi\, t/T), .., \sqrt{2/T}\sin(2\pi\, nt/T)\ldots \\ &\sqrt{2/T}\cos(2\pi\, t/T), \sqrt{2/T}\cos(2\pi 2t/T), .., \sqrt{2/T}\cos(2\pi\, nt/T)\ldots \end{aligned} \tag{8.2}$$

Любая из функций (8.2), которую для краткости обозначим $u_m(t)$, удовлетворяет условию периодичности (8.1).

Рассмотрим три следующие интеграла:

$$\frac{1}{T}\int_0^T \cos(2\pi\, mt/T)\cos(2\pi\, nt/T)\, dt = \begin{cases} 0, m \neq n \\ 1, m = n \neq 0 \\ 2, m = n = 0 \end{cases},$$

$$\frac{1}{T}\int_0^T \cos\left(2\pi\, mt/T\right)\sin\left(2\pi\, nt/T\right)dt = 0\,, \tag{8.3}$$

$$\frac{1}{T}\int_0^T \sin\left(2\pi\, mt/T\right)\sin\left(2\pi\, nt/T\right)dt = \begin{cases} 0, m \neq n \\ 1, m = n \neq 0 \\ 0, m = n = 0 \end{cases}.$$

Функции, удовлетворяющие условию (8.3), называют *ортогональными*, а систему функций (8.2) — *ортонормированным базисом*, образованным гармоническими функциями с кратными частотами. Условие ортогональности можно записать в компактной форме, используя символ Кронекера:

$$\int_0^T u_i u_k dt = \delta_{ik}\,, \tag{8.4}$$

где $$\delta_{ik} = \begin{cases} 0, i \neq k \\ 1, i = k \end{cases}.$$

Разложим произвольную периодическую функцию $s(t)$ в ряд

$$s(t) = \sum_{i=0}^{\infty} c_i u_i(t)\,. \tag{8.5}$$

Представление (8.5) называется обобщенным рядом Фурье функции $s(t)$ в выбранном базисе.

Коэффициенты данного ряда находятся умножением (8.5) на базисную функцию $u_k(t)$ и интегрированием по периоду функции $s(t)$:

$$\int_0^T s(t)u_k(t)dt = \sum_{i=0}^{\infty} c_i \int_0^T u_i u_k dt\,. \tag{8.6}$$

Откуда, используя свойство ортонормированности (8.4), найдем

$$c_k = \int_0^T s(t)u_k dt\,. \tag{8.7}$$

Подставляя в (8.7) набор функций (8.2), найдем значения коэффициентов ряда:

$$a_0 = \frac{2}{T}\int_0^T s(t)dt\,, \tag{8.8а}$$

$$a = \frac{2}{T}\int s(t)\cos(2\pi\, n/T)\,dt\,, \tag{8.8b}$$

$$b = \frac{2}{T}\int s(t)\sin(2\pi\, n/T)\,dt\,. \tag{8.8c}$$

Введя основную частоту $\omega = 2\pi/T$ последовательности, образующей периодическую функцию $s(t)$, запишем ряд Фурье для периодического сигнала:

$$s(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos(n\omega\, t) + b_n\sin(n\omega\, t)\right). \tag{8.9}$$

Анализ (8.9) показывает, что функция $s(t)$ содержит независящую от времени постоянную составляющую и бесконечный набор гармонических колебаний, так называемых, гармоник с частотами $\omega_n = n\omega$ $(n = 1, 2, \ldots)$, кратными основной частоте последовательности. Можно показать, что имеет место равенство

$$\frac{1}{T}\int_0^T s(t)^2\,dt = \frac{a_0^2}{2} + \sum_{k=1}^{\infty}\left[a_k^2 + b_k^2\right] \tag{8.10}$$

Если записать коэффициенты ряда Фурье в виде

$$a_n = A_n\cos\varphi_n, b_n = A_n\sin\varphi_n,$$

где

$$A_n = \sqrt{a_n^2 + b_n^2}, \operatorname{tg}\varphi_n = b_n/a_n\,,$$

то получим эквивалентную форму ряда Фурье:

$$s(t) = \frac{a}{2} + A\cos(n\omega\, t - \varphi)\,. \tag{8.11}$$

Спектральное разложение периодической функции $s(t)$ можно выполнить, используя систему базисных функций в виде экспонент с мнимыми показателями:

$$\{u_k\} = \left\{\frac{\exp(ik\omega\, t)}{\sqrt{T}}\right\},\ k = 0, \pm1, \pm2, \ldots, \tag{8.12}$$

которые являются ортогональными, как легко убедиться, вычислив интеграл:

$$\int_0^T u_m u_k^* dt = \frac{1}{T}\int_0^T e^{im\,t}e^{-ik\,t}dt = \delta_{mk}\,.$$

Ряд Фурье в данном случае принимает вид:

$$s(t)=\frac{1}{\sqrt{T}}\sum_{n=-\infty}^{\infty}c_n e^{in\omega t} \tag{8.13}$$

с коэффициентами

$$c_n=\frac{1}{\sqrt{T}}\int_0^T s(t)e^{-in\omega t}dt\,. \tag{8.14}$$

На практике принято использовать и другую форму записи ряда Фурье:

$$s(t)=\sum_{n=-\infty}^{\infty}C_n e^{in\omega t}\,, \tag{8.15}$$

где

$$C_n=\frac{1}{T}\int_0^T s(t)e^{-in\omega t}dt\,. \tag{8.16}$$

Выражения (8.13)–(8.16) представляют собой ряд Фурье в комплексной форме. Спектр функции $s(t)$ в соответствии с формулами (8.15), (8.16) содержит компоненты на отрицательной полуоси частот, причем $C_{-n}=C_n^*$, поэтому слагаемые с положительными и отрицательными частотами объединяются в пары, например:

$$C_n e^{in\omega t}+C_{-n}e^{-in\omega t}=|C_n|e^{i(n\omega t+\varphi_n)}+|C_n|e^{-i(n\omega t+\varphi_n)}=2|C_n|\cos(n\omega t-\varphi_n)\,.$$

Таким образом, отрицательная частота является не физическим, а математическим понятием, вытекающим из способа представления комплексных чисел.

Примечание

В технической литературе, посвященной анализу сигналов, задачу вычисления коэффициентов разложения функции в ряд Фурье называют *задачей анализа*, а задачу восстановления функции по известным коэффициентам ряда Фурье — *задачей синтеза*.

8.2. Эффект Гиббса

Эффект Гиббса возникает при решении задачи синтеза для разрывных функций. Изучим основные его проявления на примере прямоугольной импульсной функции

$$s(t)=\begin{cases}-1/2, 0\le t\le T/2\\ 1/2, T/2<t\le T\end{cases},$$

которая имеет скачек, равный 1 в точке разрыва $T/2$. Формальное разложение в ряд Фурье находится подстановкой функции $s(t)$ в формулы (8.8), вычислением интегралов, оказывающихся равными

$$a_k = 0\,,\ b_k = \frac{1}{\pi}\frac{\left[1+(-1)^{k+1}\right]}{k} = \begin{cases} \dfrac{2}{\pi k}, k - \textit{четное} \\ 0, k - \textit{нечетное} \end{cases},$$

и подстановкой выражений для коэффициентов в (8.9):

$$s(t) = \frac{2}{\pi}\left[\sin\left(\frac{2\pi}{T}t\right) + \frac{1}{3}\sin\left(\frac{2\pi}{T}3t\right) + \frac{1}{5}\sin\left(\frac{2\pi}{T}5t\right) + \ldots\right] =$$

$$= \frac{2}{\pi}\sum_{k=0}^{\infty}\frac{1}{2k+1}\sin\left(\frac{2\pi}{T}(2k+1)\ t\right)$$

Использование конечного числа членов ряда приводит к тому, что частичные суммы ряда, являются функциями, в которых присутствуют периодические составляющие. Их период равен периоду последнего удержанного члена или первого оставленного.

К. Ланцош предложил способ уменьшения эффекта пульсаций за счет сглаживания усеченного ряда

$$s_N(t) = \frac{a_0}{2} + \sum_{k=1}^{N}\left(a_k\cos\left(\frac{2\pi\ k}{T}t\right) + b_k\sin\left(\frac{2\pi\ k}{T}t\right)\right) \qquad (8.17)$$

интегрированием (усреднением) по периоду последнего оставленного члена.

При выборе в качестве интервала сглаживания периода последнего члена усеченного ряда Фурье сглаженное значение $h_N(t)$ получим как среднее от $s_N(t)$:

$$h_N(t) = \frac{1}{T/N}\int_{t-(T/2N)}^{t+(T/2N)} s_N(\xi)d\xi\,. \qquad (8.18)$$

Примечание

На практике N берется как номер первого отбрасываемого члена ряда.

Подставляя (8.17) в (8.18), получим:

$$h_N(t) = \frac{N}{T}\int_{t-(T/2N)}^{t+(T/2N)}\frac{a_0}{2}d\xi +$$

$$+\frac{N}{T}\sum_{k=1}^{N}\left[a_k\int_{t-(T/2N)}^{t+(T/2N)}\cos\left(\frac{2\pi k}{T}\xi\right)d\xi + b_k\int_{t-(T/2N)}^{t+(T/2N)}\sin\left(\frac{2\pi k}{T}\xi\right)d\xi\right] =$$

$$= \frac{a_0}{2} + \frac{N}{T}\sum_{k=1}^{N}\left[a_k \left\{ \frac{\sin[2\pi k/T\,(t+T/2N)] - \sin[2\pi k/T\,(t-T/2N)]}{2\pi k/T} \right\} - \right.$$

$$\left. - b_k \left\{ \frac{\cos[2\pi k/T\,(t+T/2N)] - \cos[2\pi k/T\,(t-T/2N)]}{2\pi k/T} \right\} \right].$$

Применяя известные тригонометрические формулы, окончательно находим:

$$h_N(t) = \frac{a_0}{2} + \sum_{k=1}^{N} \sigma(N,k)\left[a_k \cos\left(\frac{2\pi\, k}{T} t\right) + b_k \sin\left(\frac{2\pi\, k}{T} t\right)\right], \qquad (8.19)$$

где $\sigma(N,k))$ — так называемые *сигма-факторы*:

$$\sigma(N,k) = \frac{\sin(\pi\, k/N)}{\pi\, k/N}. \qquad (8.20)$$

Таким образом, сглаженный ряд Фурье есть исходный ряд Фурье с коэффициентами, умноженными на соответствующие сигма-факторы. Отметим, что эффект сглаживания достигается за счет того, что при $k = N$ сигма-фактор N-го члена оказывается равным нулю.

С физической точки зрения формулу (8.19) можно трактовать как наблюдение исходной функции $s_N(t)$ через узкое просвечивающее прямоугольное окно шириной T/N. Наблюдаемая функция $h_N(t)$ есть средняя интенсивность света от исходной функции $s_N(t)$. Она оказывается более яркой, когда функция $s_N(t)$ больше, и менее яркой, когда функция меньше.

Несмотря на то, что полученное разложение позволяет познакомиться с основными особенностями эффекта Гиббса, мы предварим дальнейшее рассмотрение этого вопроса описанием последовательности команд, позволяющей получить в пакете MATLAB разложение в ряд Фурье произвольных функций.

Пример 8.1. Исследование эффекта Гиббса в пакете MATLAB

1. Создайте файл FF.m (листинг 8.1), содержащий описание функции, раскладываемой в ряд Фурье.

Листинг 8.1. Файл FF.m

```
function z=FF(t,T)
% описание функции, раскладываемой в ряд Фурье
N=length(t);
for i=1:N
  if t(i)<0
    z(i)=0;
  end;
  if (t(i)>=0)&(t(i)<=T/2)
```

```
    z(i)=1/2;
  end;
  if (t(i)>T/2)&(t(i)<=T)
    z(i)=-1/2;
  end;
  if t(i)>T
    z(i)=0;
  end;
end;
```

2. Создайте файл AF.m (листинг 8.2), содержащий описание функции, возвращающей значение заданного коэффициента разложения в ряд Фурье по косинусам в соответствие с (8.8a), (8.8b).

Листинг 8.2. Файл AF.m

```
function z=AF(k,T)
% описание функции, возвращающей значение k-го коэффициента
dt=T/1000;
t=0:dt:T;
% вычисление значения подынтегральной функции
F=FF(t,T).*cos(2*pi*k/T*t);
z=2/T*trapz(t,F); % вычисление интеграла
```

3. Создайте файл BF.m (листинг 8.3), содержащий описание функции, возвращающей значение заданного коэффициента разложения в ряд Фурье по синусам в соответствие с (8.8c).

Листинг 8.3. Файл BF.m

```
function z=BF(k,T)
% описание функции, возвращающей значение k-го коэффициента
dt=T/1000;
t=0:dt:T;
% вычисление значений подынтегральной функции
F=FF(t,T).*sin(2*pi*k/T*t);
z=2/T*trapz(t,F); % вычисление интеграла методом трапеций
```

4. Далее необходимо выполнить следующую последовательность команд:

```
>> clear all % очистка рабочей области
>> Nf=9; % число гармоник
>> k=1:Nf;
>> T=1; % длительность импульса
>> A0=AF(0,1); % вычисление A(0) в соответствие с (8.8a)
>> for k=1:Nf  % вычисление коэффициентов A(k), B(k)
>> A(k)=AF(k,T);
```

```
>> B(k)=BF(k,T);
>> end;
% вычисление значений усеченного ряда Фурье на временном
% интервале [0,1]
>> Np=1000;
>> t=0:T/Np:1;
>> for i=1:Np+1
>> S=A0/2;
>> for k=1:Nf
>> S=S+A(k)*cos(2*pi*k/T*t(i))+B(k)*sin(2*pi*k/T*t(i));
>> end;
>> s(i)=S;
>> end;
% визуализация усеченного ряда Фурье и исходной функции
>> plot(t,s,t,FF(t,T),'--')
```

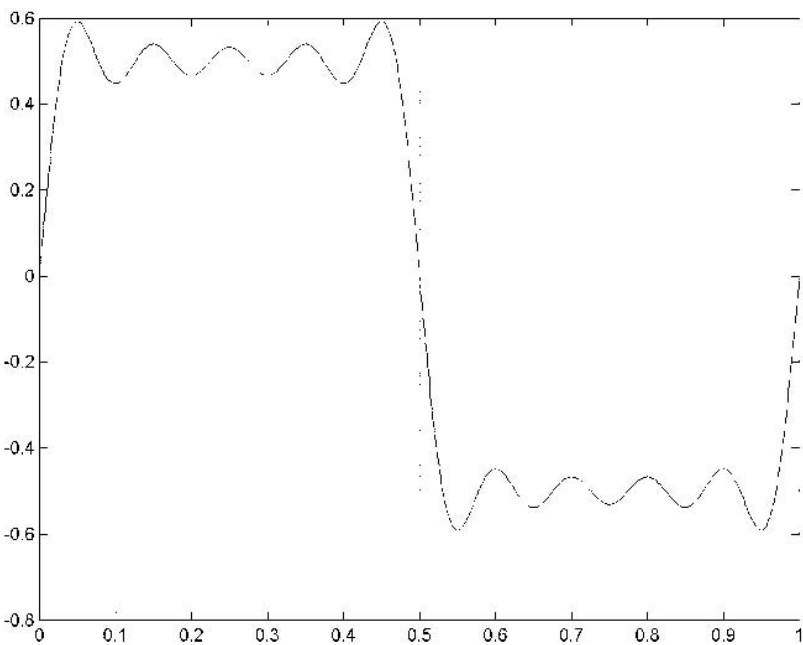


Рис. 8.1. Графики исходной функции и ее усеченного ряда Фурье (использовано 9 первых членов ряда Фурье)

Результаты выполнения описанной ранее последовательности команд представлены на рис. 8.1. Для выявления причины обнаруженного эффекта следует изменить в этой последовательности команд число гармоник ряда Фурье и пересчитать все заново. Графики исходной функции и ее ряда Фурье

с использованием 18 гармоник (Nf = 18) представлены на рис. 8.2. Анализ полученных результатов позволяет сделать качественный вывод о том, что увеличение числа членов усеченного ряда Фурье приводит к уменьшению отличий между функциями $f(t)$ и $s(t)$.

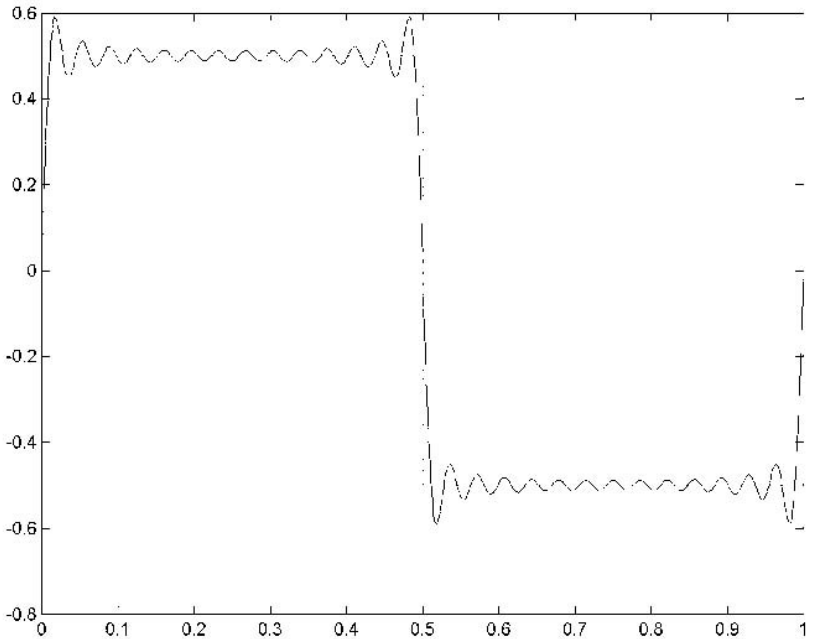


Рис. 8.2. Графики исходной функции и ее усеченного ряда Фурье (использовано 18 первых членов ряда Фурье)

8.3. Спектральный анализ дискретных функций конечной длительности

Рассмотрим особенности спектрального представления дискретной функции, заданной на временном интервале конечной длительности [0, T] N отсчетами $s_0, s_1, s_2, \ldots, s_{N-1}$, взятыми соответственно в моменты времени $0, \Delta, 2\Delta, \ldots, (N-1)\Delta$. Полное число отсчетов $N = T/\Delta$.

Можно показать, что для дискретной последовательности коэффициенты ряда Фурье определяются таким образом:

$$C_n = \frac{1}{N}\sum_{k=0}^{N-1} s_k e^{-i2\pi nk/N} . \qquad (8.21)$$

Формула (8.21) устанавливает последовательность коэффициентов, образующих дискретное преобразование Фурье (ДПФ), которое имеет следующие свойства:

- ДПФ есть линейное преобразование;
- число коэффициентов $C_0, C_1, C_2, \ldots, C_{N-1}$, вычисляемых в соответствии с (8.21), равно числу отсчетов дискретной последовательности;
- коэффициент C_0 (постоянная составляющая) есть среднее значение дискретной последовательности;
- если N — четное число, то

$$C_{N/2} = \frac{1}{N}\sum_{k=0}^{N-1}(-1)^k s_k ;$$

- для вещественной дискретной последовательности коэффициенты ДПФ, номера которых расположены симметрично относительно $N/2$, образуют сопряженные пары:

$$C_{N-n} = \frac{1}{N}\sum_{k=0}^{N-1} s_k e^{-i2\pi(N-n)k/N} = \frac{1}{N}\sum_{k=0}^{N-1} s_k e^{i2\pi nk/N} = C_n^*,$$

  поэтому можно считать, что коэффициенты $C_{N/2+1}, \ldots, C_{N-1}$ отвечают отрицательным частотам;
- если для дискретной последовательности $s_0, s_1, s_2, \ldots, s_{N-1}$ найдены коэффициенты ДПФ $C_0, C_1, C_2, \ldots, C_{N-1}$, то восстановление исходной дискретной последовательности может быть осуществлено по формуле

$$x_k = \sum_{n=0}^{N-1} C_n e^{i2\pi nk/N} . \qquad (8.22)$$

8.4. Быстрое преобразование Фурье

Из формул (8.21) и (8.22) видно, что для вычисления ДПФ или обратного ДПФ последовательности из N элементов требуется выполнить N^2 операций с комплексными числами. Если число элементов обрабатываемых массивов составляет порядка тысячи и более, то время, необходимое на выполнение этих преобразований, резко возрастает и теряется возможность обработки сигналов в реальном масштабе времени.

В 60-е годы Кули и Тьюки был предложен метод вычисления коэффициентов Фурье, позволивший снизить объем вычислений до $N \log_2 N$ операций. Он получил название алгоритма *быстрого преобразования Фурье* (БПФ).

В настоящее время процедуры, реализующие алгоритм БПФ входят во все математические библиотеки, используемые при написании программ на языках программирования высокого уровня и специализированные пакеты для математических вычислений. В связи с тем, что подробное рассмотрение алгоритма выходит за рамки нашей книги, мы рассмотрим только основную идею БПФ для случая, когда число отсчетов $N = 2^p$, где p — целое число.

Разобьем входную последовательность $\{s_k\}$ на две части с четными и нечетными номерами:

$$\{s_k\}_E = \{s_{2k}\}, \ \{s_k\}_O = \{s_{2k+1}\}, \tag{8.23}$$

где $k = 0, 1, 2, \ldots N/2 - 1$.

Это позволяет представить n-й коэффициент ДПФ в виде

$$C_n = \frac{1}{N}\sum_{k=0}^{N/2-1}\left(s_{2k}e^{-i\frac{2\pi n}{N}2k} + s_{2k+1}e^{-i\frac{2\pi n}{N}(2k+1)}\right) = \frac{1}{N}\left(\sum_{k=0}^{N/2-1}s_{2kE}e^{-i\frac{2\pi n}{N/2}k} + e^{-i\frac{2\pi n}{N}}\sum_{k=0}^{N/2-1}s_{2k+1O}e^{-i\frac{2\pi n}{N/2}k}\right). \tag{8.24}$$

Из (8.24) видно, что первая половина коэффициентов ДПФ исходного сигнала с номерами от 0 до $N/2-1$ выражается через коэффициенты ДПФ двух частных последовательностей:

$$C_n = C_{nE} + e^{-i\frac{2\pi n}{N}}C_{nO}, \tag{8.25}$$

где $k = 0, 1, 2, \ldots N/2 - 1$.

Так как последовательности коэффициентов массивов $\{s_k\}_E$ и $\{s_k\}_O$ являются периодическими с периодом $N/2$, то

$$C_{nE} = C_{n+N/2\,E}, C_{nO} = C_{n+N/2\,O}. \tag{8.26}$$

Подставляя (8.26) в (8.25) и учитывая, что

$$e^{-i\frac{2\pi(n+N/2)}{N}} = e^{-i\pi}e^{-i\frac{2\pi n}{N}} = -e^{-i\frac{2\pi n}{N}}, \tag{8.27}$$

получаем выражение для второй половины множества коэффициентов ДПФ:

$$C_{n+N/2} = C_{nE} - e^{-i\frac{2\pi n}{N}}C_{nO}, \tag{8.28}$$

где $n = 0, 1, 2, \ldots N/2 - 1$.

Формулы (8.26), (8.28) лежат в основе алгоритма БПФ: последовательности отсчетов с четными и нечетными номерами вновь разбиваются на две части. Процесс продолжается до тех пор, пока не получится последовательность, состоящая из одного элемента. ДПФ данной последовательности совпадет с сами элементом. Затем последовательно находятся коэффициенты ДПФ предыдущих последовательностей.

В пакете MATLAB быстрое одномерное преобразование Фурье реализовано парой функций, выполняющих прямое и обратное БПФ: **`fft()`** и **`ifft()`**. Данные функции используются как для действительных, так и для комплексных последовательностей, при этом длина последовательностей может быть произвольной.

Обращение к функциям:

- `fft(v)` возвращает дискретное преобразование Фурье 2^m-мерного вектора, аргумент которого есть результат дискретизации через равные промежутки времени некоторой функции. Результат работы программы комплексный вектор размерности 2^m+1. Элементы вектора, возвращаемого функцией `fft()`, вычисляются по формуле

$$c_n = \sum_{k=0}^{N-1} v_k e^{i2\pi\, nk/N},$$

  где N — число элементов вектора `v`.

- `ifft(v)` — обратное дискретное преобразование Фурье комплексного вектора, содержащего значения ДПФ. Вектор `v` должен иметь 2^m+1 элементов. Результат работы программы — действительный вектор размерности 2^{m+1}. Элементы вектора, возвращаемого функцией `ifft()`, вычисляются по формуле

$$c_n = \frac{1}{N} \sum_{k=0}^{N-1} v_k e^{-i2\pi\, nk/N},$$

  где N — число элементов вектора `v`.

Примечание

Для всех векторов справедливо соотношение `ifft(fft(v))=v`.

- `fft(v,n)` возвращает дискретное преобразование Фурье 2^n-мерного вектора, аргумент которого есть результат дискретизации через равные промежутки времени некоторой функции. Результат работы программы есть комплексный вектор размерности 2^n+1. Если `n>length(v)`, то последовательность, хранящаяся в векторе `v`, дополняется нулями.

- `ifft(v,n)` — обратное дискретное преобразование Фурье комплексного вектора, содержащего значения ДПФ. Результат работы программы

комплексный вектор размерности 2^n+1. Если `n>length(v)`, то последовательность, хранящаяся в векторе `v`, дополняется нулями.

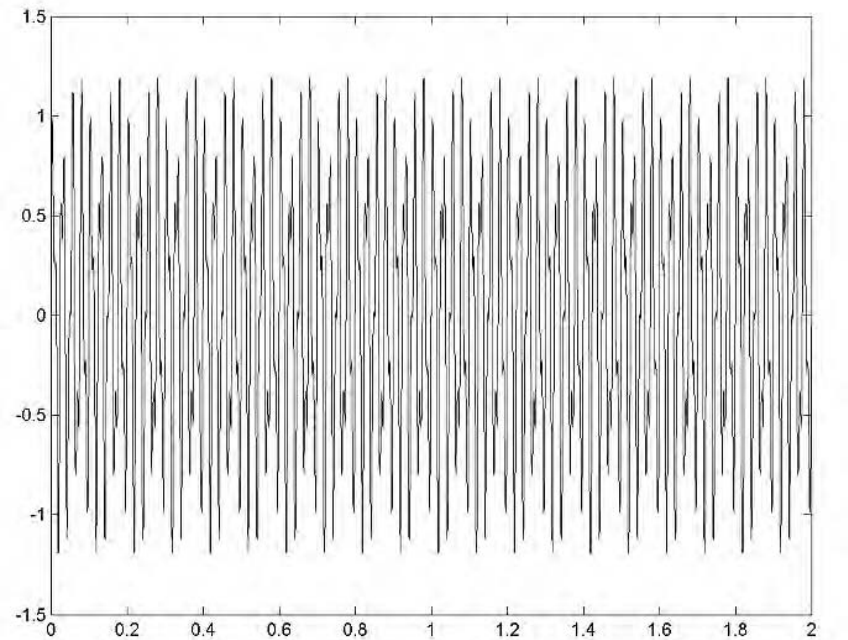


Рис. 8.3. Зависимость мгновенных значений функции $f = f(t)$ от времени

Пример 8.2. Используя функции пакета MATLAB для вычисления БПФ, найти спектр функции, заданной набором дискретных значений в $N = 8192$ точках на интервале [0, 2] с. Данная функция является суммой двух периодических функций

$$s(t) = A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t),$$

где $A_1 = 0.8$, $f_1 = 40$ Гц, $A_2 = 0.4$, $f_2 = 90$ Гц.

Для решения задачи необходимо выполнить следующую последовательность команд:

```
>> N=8192; % число точек для вычисления функции
>> i=1:N;
>> Tmax=2; % длительность временного интервала в секундах
>> t(i)=Tmax/(N-1)*(i-1); % задание временной сетки
>> f1=40 % частота первой составляющей в Гц
>> f2=90 % частота второй составляющей в Гц
% вычисление мгновенных значений функции
>> f(i)=0.8*sin(2*pi*f1*t(i))+0.4*sin(2*pi*f2*t(i));
```

```
>> figure(1);plot(t,f);
% вычисление спектра
>> c=fft(f);
% вычисление нормированного амплитудно-частотного спектра
>> j=2:N/2;
>> Cm(j-1)=abs(c(j-1))/(N/2);
>> Freq(j-1)=(j-1)/Tmax; % вычисление вектора частот в Гц
>> plot(Freq,Cm);
>> axis([0 100 0 1]);
```

Результат выполнения описанной ранее последовательности команд представлен на рис. 8.3, 8.4.

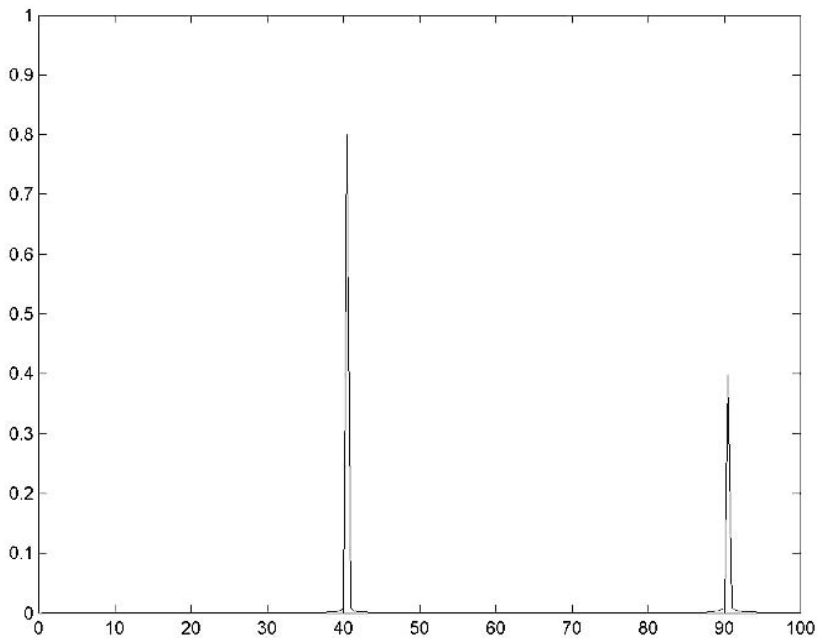
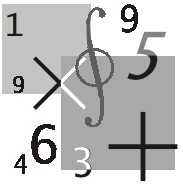


Рис. 8.4. Спектр функции $f = f(t)$



Лекция 9

Численные методы решения обыкновенных дифференциальных уравнений

Цель лекции: рассмотреть подходы и методы решения обыкновенных дифференциальных уравнений (методы Пикара, Эйлера, Эйлера-Коши, Рунге-Кутта), обсудить использование соответствующих функций пакета MATLAB.

9.1. Общие сведения и определения

Определение 9.1. Дифференциальным уравнением 1-го порядка называют соотношение вида

$$F\left(x, y, \frac{dy}{dx}\right) = 0\,. \tag{9.1}$$

Определение 9.2. Дифференциальное уравнение вида

$$\frac{dy}{dx} = f(x, y)\,, \tag{9.2}$$

где $f(x, y)$ — заданная функция двух переменных, называется дифференциальным уравнением первого порядка, разрешенным относительно производной.

Определение 9.3. Решением дифференциального уравнения на интервале I называется непрерывно дифференцируемая функция $y = \varphi(x)$, превращающая уравнение в тождество на I.

Для дифференциального уравнения первого порядка (9.1), (9.2), по определению получаем:

$$F\left(x, \varphi(x), \frac{d\varphi(x)}{dx}\right) = 0, \tag{9.3}$$

$$\frac{d\varphi(x)}{dx} = f(x, \varphi(x)). \tag{9.4}$$

Определение 9.4. Соотношение (9.3) называется решением уравнения (9.4) в неявной форме (или интегралом уравнения (9.2)), если оно определяет y как функцию от x: $y = \varphi(x)$, которая есть решение уравнения (9.2).

Определение 9.5. График решения $y = \varphi(x)$ уравнения (9.2) называется интегральной кривой данного уравнения.

Определение 9.6. Проекция графика решения на ось ординат называется фазовой кривой или траекторией дифференциального уравнения.

Определение 9.7. Задача о нахождении решения $y = \varphi(x)$ уравнения (9.2), удовлетворяющего начальному условию $\varphi(x_0) = y_0$, называется задачей Коши.

Определение 9.8. Через каждую точку (x, y) из области определения уравнения (9.2) проведем прямую, тангенс угла которой к оси абсцисс равен $f(x, y)$. Данное семейство прямых называется *полем направлений, соответствующим уравнению* (9.2) (или полем направлений функции $f(x, y)$).

Интегральная кривая в каждой своей точке касается поля направлений функции $f(x, y)$.

Существование и единственность задачи Коши дифференциальных уравнений (9.1), (9.2) обеспечивается теоремой Пикара.

Теорема Пикара. Если функция f определена и непрерывна в некоторой области G, определяемой неравенствами

$$|x - x_0| \le a, \quad |y - y_0| \le b, \tag{9.5}$$

и удовлетворяет в этой области условию Липшица по y:

$$|f(x, y_1) - f(x, y_2)| \le M|y_1 - y_2|, \tag{9.6}$$

то на некотором отрезке $|x - x_0| \le h$, где h — положительное число, существует и притом только одно решение $y = y(x)$ уравнения (9.2), удовлетворяющее начальному условию $y_0 = y(x_0)$.

Здесь M — константа Липшица, зависящая в общем случае от a и b. Если $f(x,y)$ имеет в G ограниченную производную $f'_y(x,y)$, то при $(x,y)\notin G$ можно принять

$$M = \max\left|f'_y(x)\right|. \tag{9.7}$$

Определение 9.9. Дифференциальным уравнением n-го порядка называют соотношение вида

$$F\left(x, y, y', y'', \ldots, y^{(k)}, \ldots, y^{(n)}\right)=0, \tag{9.8}$$

где x — независимая переменная, $y = y(x)$ — неизвестная функция аргумента x, $F\left(x, y, y', y'', \ldots, y^{(k)}, \ldots, y^{(n)}\right)$ — заданная функция переменных $x, y, y', y'', \ldots, y^{(k)}, \ldots, y^{(n)}$.

Определение 9.10. Задача о нахождении решения уравнения (9.8), удовлетворяющего начальным условиям

$$y(x_0) = y_0,\ y'(x_0) = y'_0,\ ,\ y^{(n-1)}\left(x_0\right) = y_0^{(n-1)}, \tag{9.9}$$

где $y_0,\ y'_0,\ ,\ y_0^{(n-1)}$ — заданные числа, называется задачей Коши для системы дифференциальных уравнений.

Разрешив дифференциальное уравнение (9.8) относительно производной $y^{(n)}$ и выполнив следующую замену переменных:

$$\begin{aligned} y' &= z_1, \\ y'' &= z_2, \\ &\ldots \\ y^{(n-1)} &= z_{n-1}, \end{aligned} \tag{9.10}$$

дифференциальное уравнение n-го порядка сводится к системе дифференциальных уравнений первого порядка:

$$\begin{aligned} y' &= z_1, \\ y'' &= z_2, \\ &\ldots \\ y^{(n-1)} &= z_{n-1} \\ z_n{}' &= f\left(x, y, z_1, z_2, \ldots z_{n-1}\right). \end{aligned} \tag{9.11}$$

Например, уравнение второго порядка

$$y'' = -\omega^2 y \tag{9.12}$$

можно записать в виде двух уравнений

$$y' = z \ ,$$

$$z' = -\omega^2 y \ . \qquad (9.13)$$

Методы решений дифференциальных уравнений подразделяются на три основные группы:

- Аналитические методы решения.
- Графические методы.
- Численные методы.

9.2. Метод Пикара

Метод Пикара позволяет получить приближенное решение дифференциального уравнения (9.2) в виде функции, заданной аналитически.

Пусть в условиях теоремы существования требуется найти решения (9.2) с начальным условием $y_0 = y(x_0)$. Запишем уравнение (9.1) в следующем эквивалентном виде:

$$dy = f(x, y)dx \ . \qquad (9.14)$$

Проинтегрируем обе части (9.14) от x_0 до x:

$$\int_{y_0}^{y} dy = \int_{x_0}^{x} f(x, y)\, dx \ . \qquad (9.15)$$

Вычислив интеграл в правой части, получим:

$$y(x) = y_0 + \int_{x_0}^{x} f(x, y)\, dx \ . \qquad (9.16)$$

Очевидно, что решение интегрального уравнения (9.16) будет удовлетворять дифференциальному уравнению (9.2) и начальному условию $y_0 = y(x_0)$. Действительно, при $x = x_0$ получим:

$$y(x_0) = y_0 + \int_{x_0}^{x_0} f(x, y)dx = y_0 \ .$$

Интегральное уравнение (9.16) позволяет использовать метод последовательных приближений. Положим $y = y_0$ и получим из (9.16) первое приближение:

$$y_1(x) = y_0 + \int_{x_0}^{x} f(x, y_0)\,dx\,. \tag{9.17}$$

Интеграл, стоящий в правой части (9.17), содержит только переменную x, после нахождения этого интеграла будет получено аналитическое выражение приближения y_1 как функции переменной x. Заменим теперь в уравнении (9.16) y найденным значением $y_1(x)$ и получим второе приближение:

$$y_2(x) = y_0 + \int_{x_0}^{x} f(x, y_1)\,dx \tag{9.18}$$

и т. д.

В общем случае итерационная формула имеет вид:

$$y_n(x) = y_0 + \int_{x_0}^{x} f(x, y_{n-1})\,dx \quad (n = 1,2,\ldots) \tag{9.19}$$

Последовательное применение формулы (9.19) дает последовательность функций:

$$y_1(x), y_2(x), \ldots, y_n(x)\ldots \tag{9.20}$$

Так как функция f непрерывна в области G, то она ограничена в некоторой области $G' \subset G$, содержащей точку (x_0, y_0), т. е.

$$|f(x, y)| \le N\,. \tag{9.21}$$

Применяя к уравнению (9.19) принцип сжимающих отображений, можно показать, что последовательность (9.20) сходится по метрике $\rho(\varphi_1, \varphi_2) = \max|\varphi_1(x) - \varphi_2(x)|$ в пространстве непрерывных функций φ, определенных на сегменте $|x - x_0| \le d$, таких, что $|\varphi(x) - y_0| \le Nd$. Предел последовательности является решением интегрального уравнения (9.16), а, следовательно, и дифференциального уравнения (9.2) с начальными условиями $y_0 = y(x_0)$. Это означает, что k-й член последовательности (9.20) является приближением к точному решению уравнения (9.2) с определенной степенью точности.

Оценка погрешности k-го приближения дается формулой:

$$|y(x) - y_k(x)| \le M^k N \frac{d^{k+1}}{(k+1)!}\,, \tag{9.22}$$

где M — константа Липшица (9.7), N — верхняя грань модуля функции f из неравенства (9.21), а величина d для определения окрестности $|x - x_0| \le d$ вычисляется по формуле:

$$d = \min\left(a, \frac{b}{N}\right). \tag{9.23}$$

9.3. Метод Эйлера

В основе метода Эйлера лежит идея графического построения решения дифференциальных уравнений (рис. 9.1).

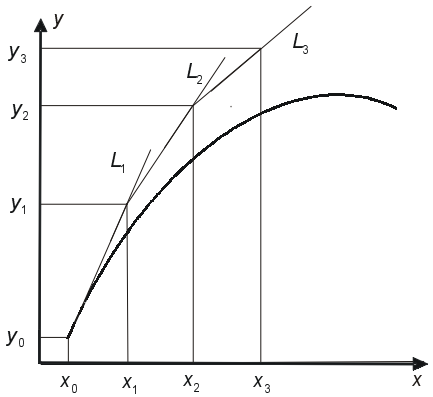


Рис. 9.1. Графическая интерпретация метода Эйлера

Пусть дано уравнение (9.2) с начальным условием $y_0 = y(x_0)$. Выбрав достаточно малый шаг h, построим, начиная с точки x_0, систему равноотстоящих точек $x_i = x_0 + ih$ (i = 0, 1, 2,...). Вместо искомой интегральной кривой на отрезке $[x_0, x_1]$ рассмотрим отрезок касательной к ней в точке $M_0(x_0, y_0)$, уравнение которой $y = y_0 + f(x_0, y_0)\cdot(x - x_0)$.

При $x = x_1$ из уравнения касательной получаем $y_1 = y_0 + hf(x_0, y_0)$. Следовательно, приращение функции на первом шаге равно $\Delta y_0 = hf(x_0, y_0)$.

Проведя аналогично касательную к интегральной кривой в точке (x_1, y_1), получим:

$$y = y_1 + f(x_1, y_1)\cdot(x - x_1),$$

что при $x = x_2$ дает $y_2 = y_1 + hf(x_1, y_1)$, т. е. y_2 получается из y_1 добавлением приращения $\Delta y_1 = hf(x_1, y_1)$.

Таким образом, вычисление таблицы значений функции, являющейся решением дифференциального уравнения (9.2), состоит в последовательном применении пары формул:

$$\Delta y_k = hf(x_k, y_k), \tag{9.24}$$

$$y_{k+1} = y_k + \Delta y_k \ . \tag{9.25}$$

Метод Эйлера, как видно из рис. 9.1, имеет погрешность. Найдем локальную погрешность, присутствующую на каждом шаге, которая определяется разностью между точным значением функции и соответствующим значением касательной. Для первого шага:

$$\Delta = y(x_1) - (y_0 + hf(x_0, y_0)) = y(x_0 + h) - (y_0 + hf(x_0, y_0)) =$$
$$y_0 + y'(x_0)h + y''(x_0)\frac{h^2}{2!} + \ldots - (y_0 + hf(x_0, y_0)) \approx y''(x_0)\frac{h^2}{2!} \tag{9.26}$$

Из (9.26) видно, что локальная погрешность пропорциональна h^2. Суммарная погрешность Δ_S после N шагов пропорциональна $N \cdot O(h^2)$, поскольку $N = 1/h$, то $\Delta_S = O(h)$, т. е. метод Эйлера — метод первого порядка точности по h.

Известны различные уточнения метода Эйлера. Модификации данных методов направлены на уточнение направления перехода из точки (x_i, y_i) в точку (x_{i+1}, y_{i+1}). Например, в методе Эйлера-Коши используют следующий порядок вычислений:

$$y^*_{i+1} = y_i + hf(x_i, y_i)\,,$$
$$y_{i+1} = y_i + h\frac{f(x_i, y_i) + f\left(x_{i+1}, y^*_{i+1}\right)}{2}. \tag{9.27}$$

Геометрически это означает, что определяется направление интегральной кривой в исходной точке (x_i, y_i) и во вспомогательной точке $\left(x_{i+1}, y^*_{i+1}\right)$, а в качестве окончательного берется среднее значение этих направлений.

Пример 9.1. Найти решение задачи Коши дифференциального уравнения

$$\frac{dy}{dx} = x^2 \ , \quad y(0) = 1.3 \ ,$$

методами Эйлера и Эйлера-Коши.

1. Метод Эйлера.

- Создайте файл Euler.m (листинг 9.1), содержащий описание функции, возвращающей решение дифференциального уравнения методом Эйлера.

Листинг 9.1. Файл Euler_g9.m

```
function [X,Y]=Euler_g9(y0,x0,x1,N)
dx=(x1-x0)/N;
x(1)=x0;
y(1)=y0;
```

```
for i=1:N
   x(i+1)=x(1)+dx*i;
   y(i+1)=y(i)+dx*F9(x(i));
end;
X=x;
Y=y;

function z=F9(x)
z=x.^2;
```

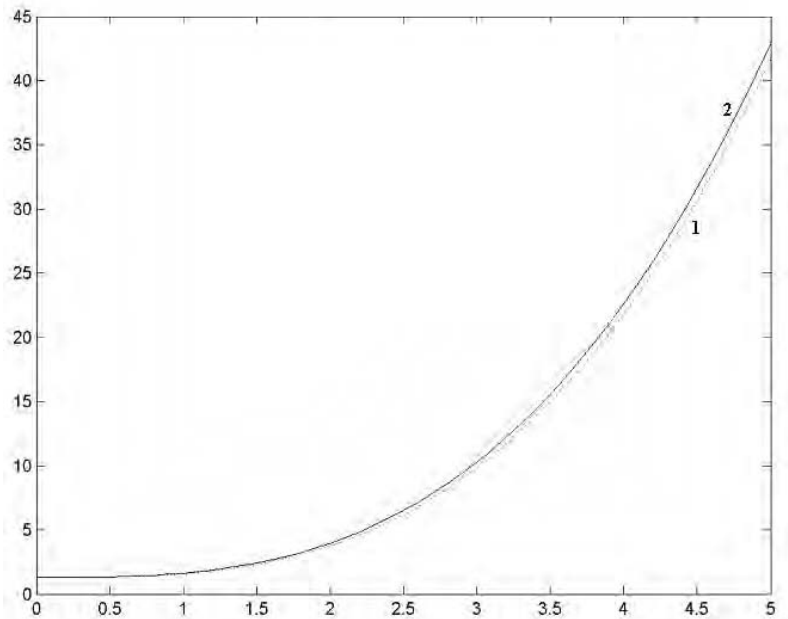


Рис. 9.2. Визуализация точного (**1**) и численного (**2**) решений задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2$, $y(0) = 1.3$ методом Эйлера

- Выполните следующую последовательность команд:

```
>> x0=0; % левая граница отрезка интегрирования
>> x1=5; % правая граница отрезка интегрирования
>> y0=1.3; % начальное условие
>> N=50; % число узлов разбиения отрезка интегрирования
>> [X Y]=Euler_g9(y0,x0,x1,N); % нахождение численного решения
                               % задачи Коши
>> i=1:length(X);
>> Z(i)=y0+1/3*X(i).^3; % вычисление значений точного решения
>> plot(X,Z,X,Y,':') % визуализация численного и точного решений
```

```
                    % (рис. 9.2)
>> plot(X,abs(Z-Y)) % визуализация разности между численным и
                    % точным решениями ДУ (рис. 9.3)
```

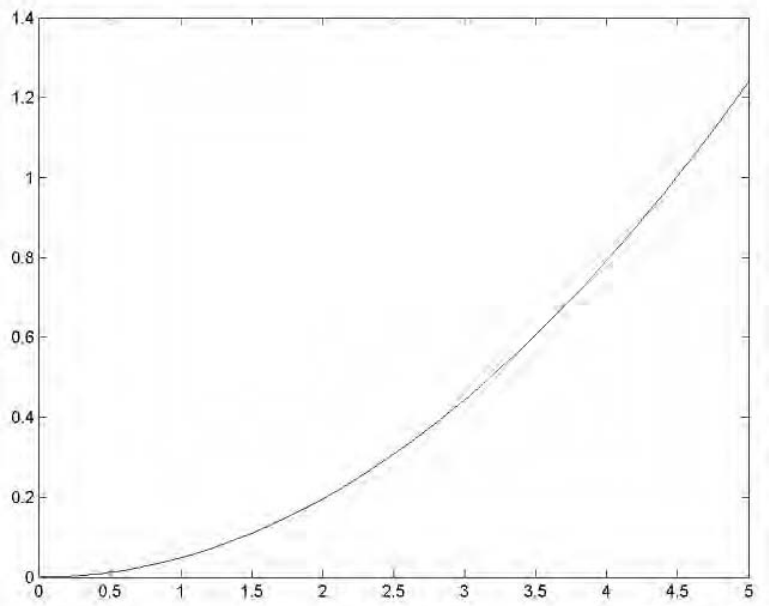


Рис. 9.3. Разность между численным и точным решениями задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2$, $y(0) = 1.3$

2. Метод Эйлера-Коши.

- Создайте файл EulerKoshi.m (листинг 9.2), содержащий описание функции, возвращающей решение дифференциального уравнения методом Эйлера-Коши.

Листинг 9.2. Файл EulerKoshi.m

```
function [X,Y]=EulerKoshi(y0,x0,x1,N)
dx=(x1-x0)/N;
x(1)=x0;
y(1)=y0;
for i=2:N
  x(i)=x(1)+dx*(i-1);
  Z=y(i-1)+dx*F9(x(i-1),y(i-1));
  y(i)=y(i-1)+(F9(x(i-1),y(i-1))+F9(x(i),Z))*dx/2;
end;
X=x;
```

```
Y=y;

function z=F9(x,y)
z=x.^2;
```

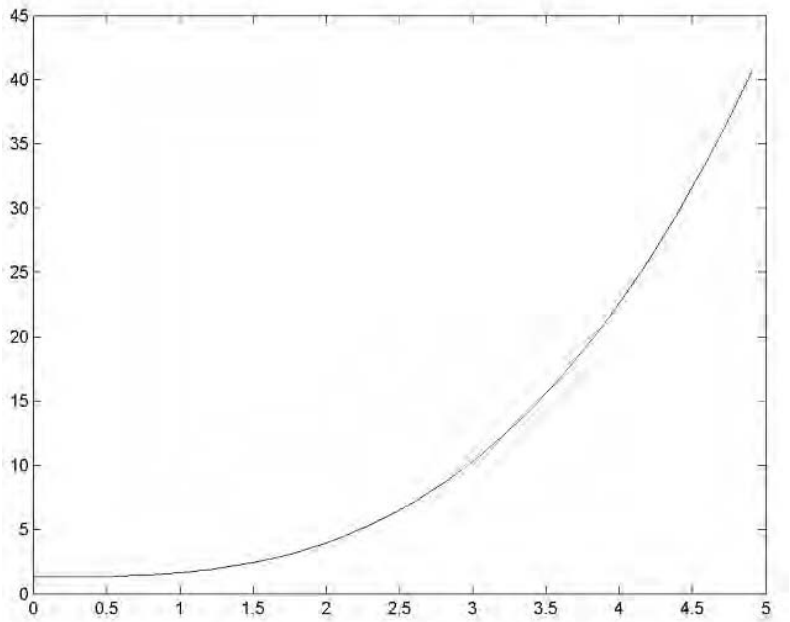


Рис. 9.4. Визуализация точного и численного решений задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2$, $y(0) = 1.3$ методом Эйлера-Коши

- Выполните следующую последовательность команд:

```
>> x0=0; % левая граница отрезка интегрирования
>> x1=5; % правая граница отрезка интегрирования
>> y0=1.3; % начальное условие
>> N=50; % число узлов разбиения отрезка интегрирования
>> [X Y]=EulerKoshi(y0,x0,x1,N); % нахождение численного
                                 % решения задачи Коши
>> i=1:length(X);
>> Z(i)=y0+1/3*X(i).^3; % вычисление значений точного решения
>> plot(X,Z,X,Y,':') % визуализация точного и численного решений
                     % (рис. 9.4)
>> plot(X,abs(Z-Y))  % визуализация разности между численным и
                     % точным решениями ДУ (рис. 9.5)
```

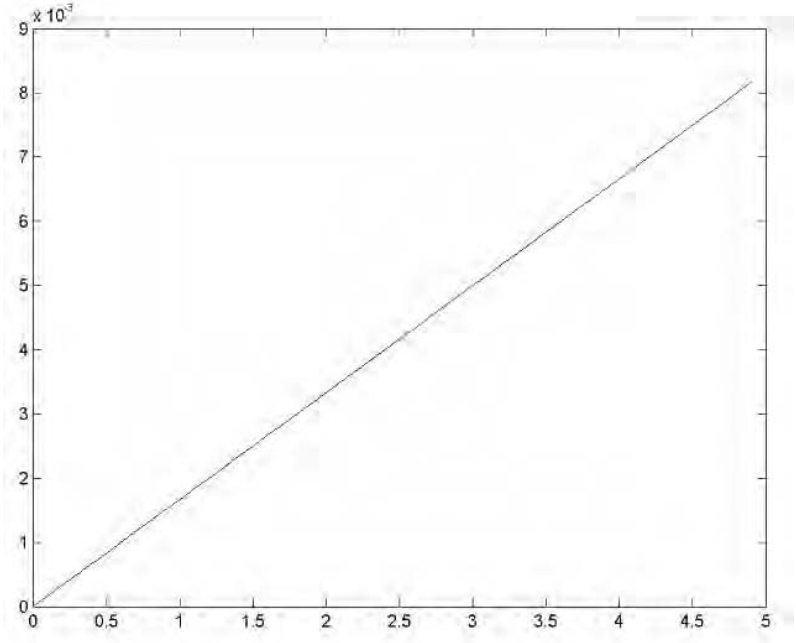


Рис. 9.5. Разность между точным решением задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2$, $y(0) = 1.3$ и численным решением, полученным методом Эйлера-Коши

Из сравнения рис. 9.3, 9.5 видно, что погрешность, как и ожидалось, уменьшилась в 10^2 раз ($h = 0.1$).

9.4. Метод Рунге-Кутта

Метод Эйлера и метод Эйлера-Коши относятся к семейству методов Рунге-Кутта. Для построения данных методов можно использовать следующий общий подход. Фиксируем некоторые числа:

$$\alpha_2, \ldots \alpha_q; p_1, \ldots p_q; \beta_{ij}, 0 < j < i \le q .$$

Последовательно вычисляем:

$$k_1(h) = h \cdot f(x, y),$$

$$k_2(h) = h \cdot f(x + \alpha_2 \cdot h, y + \beta_{12} k_1(h)),$$

...

$$k_q(h) = h \cdot f\left(x + \alpha_q h, y + \beta_{q1} k_1(h) + \ldots \beta_{qq-1} k_{q-1}(h)\right)$$

и полагаем:

$$y(x+h) \approx y(x) + \sum_{i=1}^{q} p_i k_i(h) = z(h). \tag{9.28}$$

Рассмотрим вопрос о выборе параметров $\alpha_i, p_i, \beta_{ij}$. Обозначим

$$\varphi(h) = y(x+h) - z(h).$$

Будем предполагать, что

$$\varphi(0) = \varphi'(0) \ldots = \varphi^{(s)}(0) = 0,$$

а $\varphi^{(s+1)}(0) \neq 0$ для некоторой функции $f(x,y)$.

По формуле Тэйлора справедливо равенство

$$\varphi(h) = \sum_{i=0}^{s} \frac{\varphi^{(i)}(0)}{i!} h^i + \frac{\varphi^{(s+1)}(\theta h)}{(s+1)!} h^{s+1}, \tag{9.29}$$

где $0 < \theta < 1$.

При $q = 1$ будем иметь:

$$\varphi(h) = y(x+h) - y(x) - p_1 \cdot h \cdot f(x,y),$$

$$\varphi(0) = 0,$$

$$\varphi'(0) = (y'(x+h) - p_1 f(x,y))|_{h=0} = f(x,y)(1-p_1),$$

$$\varphi''(h) = y''(x+h).$$

Ясно, что равенство $\varphi'(0) = 0$ выполняется для любых функций $f(x,y)$ лишь при условии, что $p_1 = 1$. При данном значении p_1 из формулы (9.28) получаются формулы (9.24), (9.25) метода Эйлера. Погрешность данного метода на шаге согласно (9.29) равна

$$\varphi(h) = \frac{\varphi''(x + h\theta) \cdot h^2}{2}.$$

Рассмотрим случай $q = 2$, тогда

$$\varphi(h) = y(x+h) - y(x) - p_1 h f(x,y) - p_2 h f(\bar{x}, \bar{y}),$$

где $\bar{x} = x + \alpha_2 h, \bar{y} = y + \beta_{21} h f(x,y)$.

Согласно исходному дифференциальному уравнению

$$y' = f,$$

$$y'' = f_x + \frac{\partial f}{\partial y}\frac{\partial y}{\partial x} = f_x + f_y f(x,y),$$

$$y''' = f_{xx} + f_{xy}f + f\frac{\partial f_y}{\partial x} + f_y\frac{\partial f}{\partial x} =$$
$$= f_{xx} + f_{xy}f + f\left(f_{xy} + f_{yy}f\right) + f_y\left(f_x + f_y f\right) = \qquad (9.30)$$
$$= f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y y''.$$

Вычисляя производные функции $\varphi(h)$ и, подставляя в выражения для $\varphi(h), \varphi'(h), \varphi''(h)$ значение $h = 0$, получаем:

$$\varphi(0) = 0,$$
$$\varphi'(0) = (1 - p_1 - p_2)f,$$
$$\varphi''(0) = (1 - 2p_2\alpha_2)f_x + (1 - 2p_2\beta_{21})f_y f. \qquad (9.31)$$

Требование

$$\varphi(0) = \varphi'(0) = \varphi''(0) = 0$$

будет выполняться для всех $f(x, y)$ только в том случае, если одновременно справедливы следующие три равенства относительно четырех параметров:

$$1 - p_1 - p_2 = 0,$$
$$1 - 2p_2\alpha_2 = 0, \qquad (9.32)$$
$$1 - 2p_2\beta_{21} = 0.$$

Задавая произвольно значения одного из параметров и определяя значения остальных параметров из системы (9.32), можно получать различные методы Рунге-Кутта с порядком погрешности $s = 2$. Например, при $p_1 = \frac{1}{2}$ из (9.32) получаем: $p_2 = \frac{1}{2}$, $\alpha_2 = 1$, $\beta_{21} = 1$.

Для выбранных значений параметров формула (9.28) приобретает следующий вид:

$$y_{i+1} = y_i + h\frac{f(x_i, y_i) + f\left(x_{i+1}, y^*_{i+1}\right)}{2}.$$

Здесь y_{i+1} записано вместо $y(x + h)$, y_i — вместо $y(x)$, а с помощью y^*_{i+1} обозначено выражение $y_i + h \cdot f(x_i, y_i)$.

Таким образом, для рассматриваемого случая приходим к расчетным формулам (9.27) метода Эйлера-Коши. Из (9.29) следует, что главная часть погрешности на шаге есть

$$\varphi'''(0)h^3/6,$$

т. е. погрешность пропорциональна третьей степени шага.

На практике наиболее часто используют метод Рунге-Кутта с $q=4, s=4$. Данный метод реализуется в соответствии со следующими расчетными формулами:

$$k_1 = h \cdot f(x, y),$$
$$k_2 = h \cdot f\left(x + \frac{h}{2}, y + \frac{k_1}{2}\right),$$
$$k_3 = h \cdot f\left(x + \frac{h}{2}, y + \frac{k_2}{2}\right), \qquad (9.33)$$
$$k_4 = h \cdot f(x + h, y + k_3),$$
$$\Delta y = z(h) - y(x) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Погрешность рассматриваемого метода Рунге-Кутта на шаге пропорциональна пятой степени шага.

Геометрический смысл использования метода Рунге-Кутта с расчетными формулами состоит в следующем. Из точки (x_i, y_i) сдвигаются в направлении, определяемом 13углом α_1, для которого $tg\alpha_1 = f(x_i, y_i)$. На этом направлении выбирается точка с координатами $\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$. Затем из точки (x_i, y_i) сдвигаются в направлении, определяемым углом α_2, для которого $tg\alpha_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$, и на этом направлении выбирается точка с координатами $\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$. Наконец, из точки (x_i, y_i), сдвигаются в направлении, определяемом углом α_3, для которого $tg\alpha_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$, и на этом направлении выбирается точка с координатами $(x_i + h, y_i + k_3)$. Этим задается еще одно направление, определяемое углом α_4, для которого $tg\alpha_4 = f(x_i + h, y_i + k_3)$. Четыре полученные направления усредняются в соответствие с (9.33). На этом окончательном направлении и выбирается очередная точка $(x_{i+1}, y_{i+1}) = (x_i + h, y_i + \Delta y)$.

Пример 9.2. Найти решение задачи Коши дифференциального уравнения

$$\frac{dy}{dx} = x^2, \quad y(0) = 1.3$$

методом Рунге-Кутта четвертого порядка.

1. Создайте файл RungeKutt4.m (листинг 9.3), содержащий описание функции, возвращающей решение дифференциального уравнения методом Рунге-Кутта четвертого порядка.

Листинг 9.3. Файл RungeKutt4.m

```
function [X,Y]=RungeKutt4(y0,x0,x1,N)
dx=(x1-x0)/N;
x(1)=x0;
y(1)=y0;
for i=2:N
   x(i)=x(1)+dx*(i-1);
   k1=dx*F9(x(i-1),y(i-1));
   k2=dx*F9(x(i-1)+dx/2,y(i-1)+k1/2);
   k3=dx*F9(x(i-1)+dx/2,y(i-1)+k2/2);
   k4=dx*F9(x(i-1)+dx,y(i-1)+k3);
   y(i)=y(i-1)+1/6*(k1+2*k2+2*k3+k4);
end;
X=x;
Y=y;

function z=F9(x,y)
z=x.^2;
```

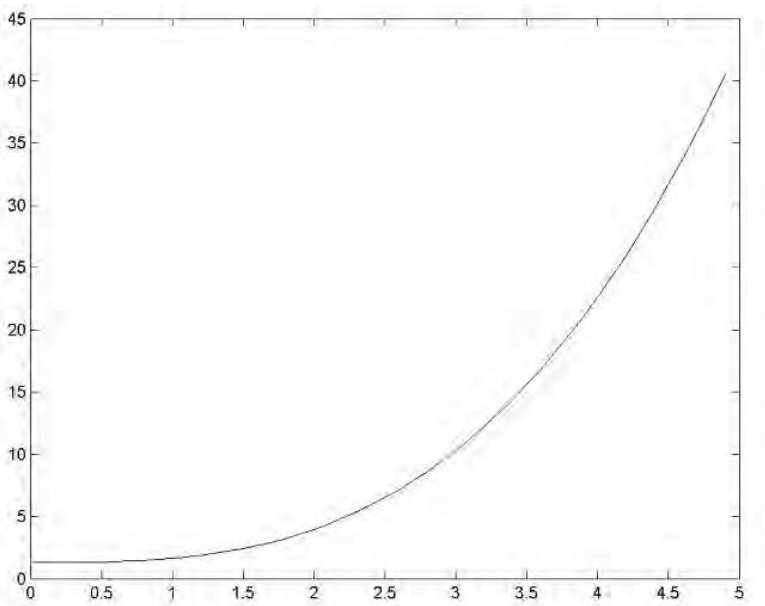


Рис. 9.6. Визуализация точного решения задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2$, $y(0) = 1.3$ и численного решения, полученного методом Рунге-Кутта

2. Выполните следующую последовательность команд:

```
>> x0=0; % левая граница отрезка интегрирования
>> x1=5; % правая граница отрезка интегрирования
>> y0=1.3; % начальное условие
>> N=50; % число узлов разбиения отрезка интегрирования
>> [X Y]=RungeKutt4(y0,x0,x1,N); % нахождение численного
                                 % решения задачи Коши
>> i=1:length(X);
>> Z(i)=y0+1/3*X(i).^3; % вычисление значений точного решения
>> plot(X,Z,X,Y,':') % визуализация точного и численного решений
                     % (рис. 9.6)
>> plot(X,abs(Z-Y))  % визуализация разности между численным и
                     % точным решениями ДУ (рис. 9.7)
```

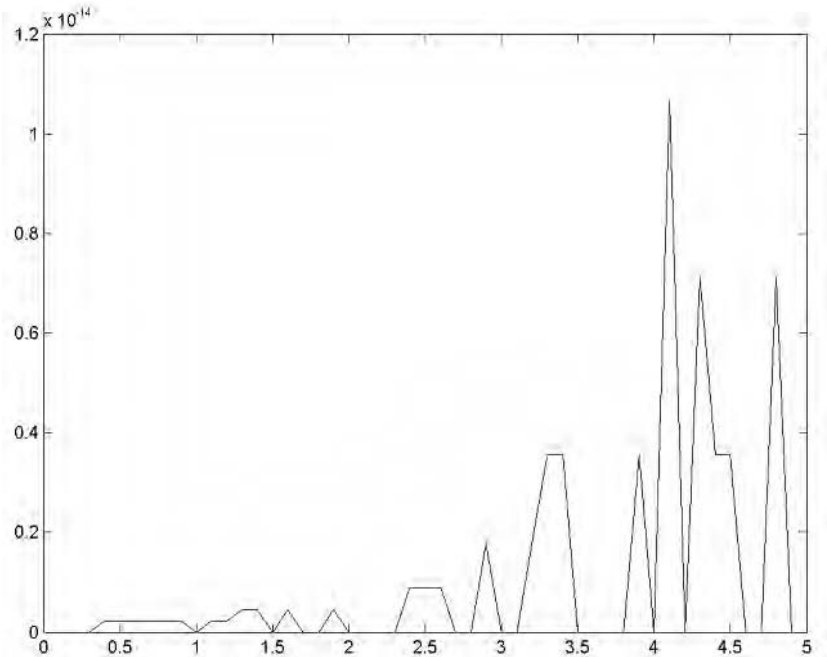


Рис. 9.7. Разность между точным решением задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2$, $y(0) = 1.3$ и численным решением, полученным методом Рунге-Кутта

9.5. Средства пакета MATLAB для решения обыкновенных дифференциальных уравнений

Решение задачи Коши для дифференциальных уравнений методом Рунге-Кутта 4-го порядка реализовано в пакете MATLAB в виде функции **ode45**. Этот метод рекомендуется использовать при первой попытке нахождения численного решения задачи.

Помимо данной функции в пакете MATLAB реализованы и другие методы решения дифференциальных уравнений и их систем.

- **ode23** — функция реализует одношаговые явные методы Рунге-Кутта второго и третьего порядков. Используется при решении нежестких систем дифференциальных уравнений и обеспечивает удовлетворительную точность при меньших (нежели функция ode45) временных затратах.
- **ode113** — функция реализует многошаговый метод Адамса-Башворта-Мултона переменного порядка. Используется при необходимости обеспечить высокую точность численного решения.
- **Ode15s** — функция реализует многошаговый метод переменного порядка (от 1 до 5 по умолчанию), основанный на формулах численного дифференцирования. Данный метод следует использовать в том случае, если не удается найти численное решение с помощью функции ode45.
- **ode23s** — функция реализует одношаговый метод, использующий модифицированную формулу Розенброка 2-го порядка. Данный метод обеспечивает более высокую скорость вычислений по сравнению с другими методами при относительно более низкой точности вычислений.
- **ode23t** — функция реализует метод трапеций с интерполяцией. Данный метод используют при решении уравнений, описывающих колебательные системы с почти гармоническим выходным сигналом.
- **ode23tb** — функция реализует неявный метод Рунге-Кутта в начале интервала интегрирования и далее метод, использующий формулы обратного дифференцирования 2-го порядка. Данный метод обладает большей скоростью нежели метод ode15s при, соответственно, меньшей точности.

Все перечисленные ранее функции, описываемые в документации пакета Solver (Решатель), могут решать системы дифференциальных уравнений явного вида $\vec{y}' = \vec{F}(t, \vec{y})$. Кроме того, решатели ode15s, ode23s, ode23t и ode23tb системы дифференциальных уравнений неявного вида

$M(t,\vec{y})\vec{y}' = \vec{F}(t,\vec{y})$, а также все решатели, кроме `ode23s`, могут находить решения уравнения вида $M(\vec{y})\vec{y}' = \vec{F}(t,\vec{y})$.

Пример 9.3. Найти решение задачи Коши для дифференциального уравнения

$$\frac{dT}{dt} = -r(T - T_s), \tag{9.34}$$

где T_s, r — заданные постоянные, имеющие физический смысл температуры окружающей среды и коэффициента остывания, соответственно, с начальным условием $T(0) = 80$.

1. Для решения дифференциального уравнения (9.34) сначала создайте файл Tempr.m (листинг 9.4), содержащий определение функции, стоящей в правой части уравнения (2.1):

Листинг 9.4. Файл Tempr.m

```
function Z=Tempr(t,T)
% определение функции, стоящей в правой части уравнения (2.1)
global Ts r
Z(1)=-r*(T-Ts);
```

2. Далее выполните в командном окне MATLAB следующую последовательность операторов:

```
>> global Ts r % объявление глобальных переменных
>> Ts = 22 % задание значения температуры окружающей среды
>> r = 0.024 % задание значения коэффициента остывания
>> T0 = 80 % задание начальной температуры тела
>> [t,T]=ode45('Tempr',[0:0.01:15],T0); % Tempr – имя файла, содержащего
                                        % определение функции,
                                        % стоящей в правой части
                                        % уравнения (2.1);
                                        % [0:0.01:15] – вектор,
                                        % определяющий интервал
                                        % интегрирования,
                                        % T0 – переменная, содержащая
                                        % начальную температуру

>> plot(t,T)
```

После выполнения приведенной ранее последовательности команд будет создано окно, содержащее график зависимости температуры тела от времени, представленный на рис. 9.8.

По умолчанию решатели систем дифференциальных уравнений пакета MATLAB используют параметры, относительная погрешность которых не превосходит переменной `RelTol=10^-3`, граница абсолютной погрешности численного решения — переменная `AbsTol=10^-6`. Для изменения значений этих переменных используется команда

```
>> options = odeset('RelTol',1e-4,'AbsTol',1e-4);
```

предваряющая команду вызова функции решателя системы дифференциальных уравнений.

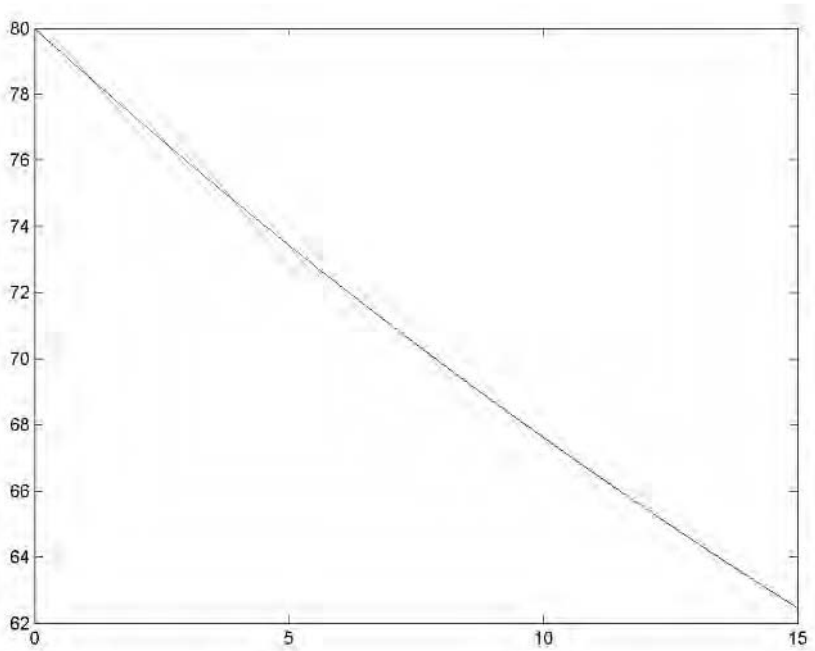
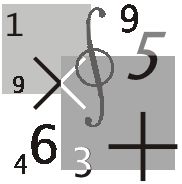


Рис. 9.8. Численное решение уравнения теплопроводности, возвращенное функцией `ode45`



Лекция 10

Численные методы решения дифференциальных уравнений в частных производных

Цель лекции: изложить общие сведения и классификацию уравнений в частных производных (УЧП), рассмотреть явные и неявные разностные схемы для эллиптических, параболических и гиперболических уравнений и их программные реализации, продемонстрировать основные идеи использования метода Монте-Карло для решения УЧП.

10.1. Общие сведения и классификация уравнений в частных производных

Определение 10.1. Дифференциальные уравнения, содержащие частные производные, называются дифференциальными уравнениями в частных производных.

В отличие от обыкновенных дифференциальных уравнений (ОДУ), в которых неизвестная функция зависит только от одной переменной, в уравнениях с частными производными неизвестная функция зависит от нескольких переменных (например, температура зависит от координаты x и времени t).

Для упрощения записи будем использовать следующие обозначения:

$$u_t \equiv \frac{\partial u}{\partial t}, \quad S(t), \quad u_{xx} \equiv \frac{\partial^2 u}{\partial x^2}, \quad .$$

Примеры уравнений с частными производными:

- $u_t = u_{xx}$ (одномерное уравнение теплопроводности);
- $u_t = u_{xx} + u_{yy}$ (двумерное уравнение теплопроводности);
- $u_{rr} + \frac{1}{r}u_r + \frac{1}{r^2}u_{\theta\theta} = 0$ (уравнение Лапласа в полярных координатах);
- $u_{tt} = u_{xx} + u_{yy} + u_{zz}$ (трехмерное волновое уравнение);
- $u_{tt} = u_{xx} + \alpha u_t + \beta u$ (телеграфное уравнение).

Отметим, что в приведенных ранее примерах неизвестная функция u зависит более чем от одной переменной. Функция u, от которой находятся производные, называется зависимой переменной, переменные t, x, по которым производится дифференцирование, называются независимыми переменными.

Методы решения уравнений в частных производных:

- метод разделения переменных
- метод интегральных преобразований
- метод преобразования координат
- метод преобразования зависимой переменной
- численные методы
- метод теории возмущений
- метод функций Грина
- метод интегральных уравнений
- вариационные методы
- метод разложения по собственным функциям
- метод обратной задачи рассеяния

Важность классификации УПЧ обусловлена тем что, для каждого класса существует своя общая теория и методы решения уравнений. Уравнения в частных производных (УЧП) можно классифицировать по многим признакам.

Методы классификации УЧП:

1. По порядку уравнения (*порядком уравнения* называют наивысший порядок частных производных, входящих в уравнение):

- $u_t = u_x$ (уравнение первого порядка);
- $u_t = u_{xx}$ (уравнение второго порядка);
- $u_t = uu_{xxx} + \sin x$ (уравнение третьего порядка).

2. По числу переменных (*числом переменных* называют число независимых переменных).

- $u_t = u_{xx}$ (уравнение с двумя переменными)
- $u_t = u_{rr} + \frac{1}{r}u_r + \frac{1}{r^2}u_{\theta\theta}$ (уравнение с тремя переменными r, θ, t)

3. По критерию линейное/нелинейное.

   Линейным уравнением второго порядка с двумя независимыми переменными называется уравнение вида

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G, \qquad (10.1)$$

   где A, B, C, D, E, F и G константы или заданные функции переменных x и y.

- $u_{xx} + yu_{yy} = 0$, $u_{tt} = e^{-t}u_{xx} + \sin t$ (линейные уравнения);
- $uu_{xx} + u_t = 0$, $xu_x + yu_y + u^2 = 0$ (нелинейные уравнения).

4. По критерию однородное/неоднородное.

   Уравнение (10.1) называется *однородным*, если правая часть $G(x, y)$ тождественно равна нулю для всех x и y. Если $G(x, y)$ не равна нулю тождественно, то уравнение называется *неоднородным*.

5. По виду коэффициентов.

   Если коэффициенты A, B, C, D, E, F и G уравнения (10.1) — константы, то уравнение (10.1) называется уравнением *с постоянными коэффициентами*, в противном случае уравнением *с переменными коэффициентами*.

Существует несколько типов линейных уравнений.

Параболический тип. Уравнения параболического типа описывают процессы теплопроводности и диффузии и определяются условием

$$B^2 - 4AC = 0.$$

Гиперболический тип. Уравнения гиперболического типа описывают колебательные системы и волновые движения и определяются условием

$$B^2 - 4AC > 0.$$

Эллиптический тип. Уравнения эллиптического типа описывают установившиеся процессы и определяются условием

$$B^2 - 4AC < 0.$$

Примеры линейных уравнений разных типов:

- $u_t = u_{xx}$, $B^2 - 4AC = 0$ (параболическое);

- $u_{tt} = u_{xx}$, $B^2 - 4AC = 4$ (гиперболическое);
- $u_{\xi\eta} = 0$, $B^2 - 4AC = 1$ (гиперболическое);
- $u_{xx} + u_{yy} = 0$, $B^2 - 4AC = -4$ (эллиптическое);
- $yu_{xx} + u_{yy} = 0$, $B^2 - 4AC = -4y$ (эллиптическое при $y > 0$, параболическое при $y = 0$, гиперболическое при $y < 0$).

10.2. Численные методы решения эллиптических уравнений

При решении эллиптических УЧП ставится задача отыскания решения в некоторой области пространства при заданных значениях функции на границе области (задача Дирихле). Иллюстрацией задачи Дирихле является задача нахождения решения уравнения Лапласа

$$u_{rr} + \frac{1}{r}u_r + \frac{1}{r^2}u_{\theta\theta} = 0, \quad 0 < r < 1,$$

с заданными граничными условиями (ГУ)

$$u(1, \theta) = \sin\theta, \quad 0 \le \theta < 2\pi.$$

Рассмотрение численных методов решения задачи Дирихле уравнения Лапласа

$$u_{xx} + u_{yy} = 0,$$

в прямоугольной области $0 \le x \le 1$, $0 \le y \le 1$ с граничными условиями

$$u(x,0) = g_1(y), \; u(1,y) = g_2(y), \; u(x,1) = g_3(y), \; u(0,y) = g_4(y),$$

начнем со знакомства с понятием *конечно-разностная частная производная.*

Напомним, что при численном интегрировании производную функции, определяемую в соответствие с выражением

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

мы заменяли выражением

$$f'(x) \cong \frac{f(x+h) - f(x)}{h}, \qquad (10.2)$$

с точностью до членов порядка h аппроксимирующей производную в точке x.

Выражение, стоящее в правой части (10.2), называется правой разностной производной.

В разложении Тейлора функции $f(x)$ можно заменить h на $-h$ и получить левую разностную производную:

$$f'(x) \cong \frac{f(x) - f(x-h)}{h}. \qquad (10.3)$$

Складывая (10.2) и (10.3), получаем центральную разностную производную:

$$f'(x) \cong \frac{f(x+h) - f(x-h)}{2h}. \qquad (10.4)$$

Поступая аналогично, можно получить центральную разностную производную — аппроксимацию второй производной:

$$f''(x) \cong \frac{1}{h^2}[f(x+h) - 2f(x) + f(x+h)]. \qquad (10.5)$$

Теперь можно распространить понятие конечно-разностной аппроксимации на частные производные. Используя разложение функции двух переменных в ряд Тэйлора

$$u(x+h, y) = u(x, y) + u_x(x, y)h + u_{xx}(x, y)\frac{h^2}{2!} + \ldots,$$

$$u(x-h, y) = u(x, y) - u_x(x, y)h + u_{xx}(x, y)\frac{h^2}{2!} - \ldots,$$

находим выражения для конечно-разностной аппроксимации частных производных:

$$u_x(x, y) \cong \frac{u(x+h, y) - u(x, y)}{h},$$

$$u_{xx}(x, y) \cong \frac{1}{h^2}[u(x+h, y) - 2u(x, y) + u(x-h, y)],$$

$$u_y(x, y) \cong \frac{u(x, y+k) - u(x, y)}{k},$$

$$u_{yy}(x, y) \cong \frac{1}{h^2}[u(x, y+k) - 2u(x, y) + u(x, y-k)].$$

В данных формулах частные производные аппроксимируются правыми, центральными и левыми разностями, но мы далее будем использовать центральные разностные аппроксимации.

Построим на плоскости (x, y) равномерную сетку (рис. 10.1).

Будем использовать следующие обозначения:

$$u(x_j, y_i) = u_{ij},$$

$$u(x_j, y_i + k) = u_{i+1, j},$$

$$u\left(x_j, y_i - k\right) = u_{i-1,\,j}\ ,$$

$$u\left(x_j - h, y_i\right) = u_{ij-1}\ ,$$

$$u\left(x_j + h, y_i\right) = u_{ij+1}\ ,$$

$$u_x\left(x_j, y_i\right) = \frac{1}{2h}\left(u_{i,\,j+1} - u_{i,\,j-1}\right),$$

$$u_y\left(x_j, y_i\right) = \frac{1}{2h}\left(u_{i+1,\,j} - u_{i-1,\,j}\right),$$

$$u_{xx}\left(x_j, y_i\right) = \frac{1}{h^2}\left(u_{i,\,j+1} - 2u_{i,\,j} + u_{i,\,j-1}\right),$$

$$u_{yy}\left(x_j, y_i\right) = \frac{1}{k^2}\left(u_{i+1,\,j} - 2u_{i,\,j} + u_{i-1,\,j}\right).$$

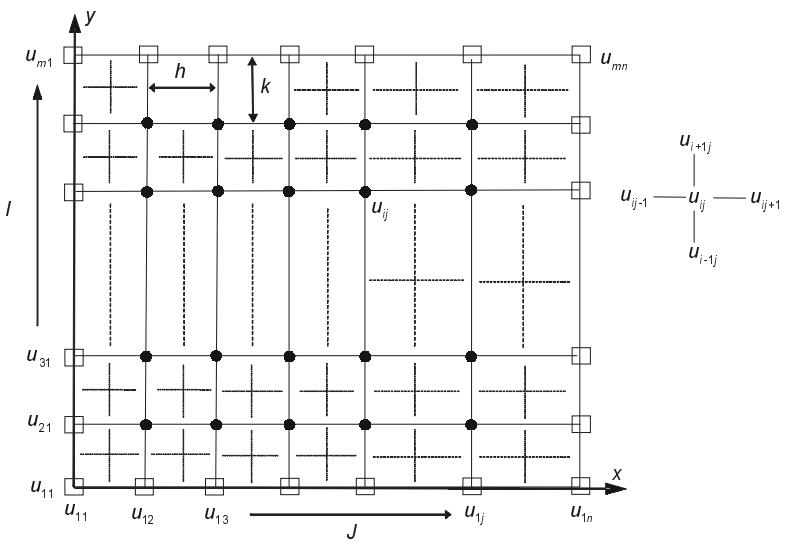


Рис. 10.1. Координатная сетка, используемая для решения УЧП эллиптического типа

Заменим частные производные в уравнении Лапласа их конечно-разностной аппроксимацией:

$$\frac{1}{h^2}\left(u_{i,\,j+1} - 2u_{i,\,j} + u_{i,\,j-1}\right) + \frac{1}{k^2}\left(u_{i+1,\,j} - 2u_{i,\,j} + u_{i-1,\,j}\right) = 0\ . \qquad (10.6)$$

Если $k = h$, то уравнение Лапласа приводится к виду

$$u_{i+1,\,j} + u_{i-1,\,j} + u_{i,\,j+1} + u_{i,\,j-1} - 4u_{ij} = 0\ . \qquad (10.7)$$

Разрешив (10.7) относительно u_{ij}, получим

$$u_{ij} = \frac{1}{4}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}\right). \qquad (10.8)$$

Отметим, что в (10.8) все соотношения берутся для внутренних узлов сетки, так как значения функции на границе заданы. Соотношение (10.8) можно трактовать, как аппроксимацию значения функции в точке (x_i, y_j) средним значением решения по четырем соседним точкам. На первый взгляд, польза от выражения (10.8) не очень очевидна, т. к. мы нашли связь между значением функции в *ij*-том узле и значениями функции в соседних узлах: (*ij* − 1)-ом, (*ij* + 1)-ом, (*I* − 1*j*)-ом, (*i* + 1*j*)-ом, которые в свою очередь также неизвестны. Однако оказывается, что если задать некоторые начальные значения решения уравнения Лапласа в узлах сетки и затем их последовательно уточнять в соответствие с (10.8), то итерационный процесс будет сходиться (релаксировать) к точному решению.

Таким образом, численное решение задачи Дирихле методом релаксации находится в соответствие со следующим алгоритмом:

1. Присвоить величинам u_{ij} во внутренних узлах сетки некоторые численные значения (например, равные среднему значению всех граничных условий).
2. Пересчитать значения во всех внутренних точках сетки в соответствии с (10.8) (заменяя старое значение средним значением, вычисленным по четырем соседним точкам).

Отметим, что при этом неважно, по строкам или по столбцам организован процесс счета. Недостатком данной схемы является низкая скорость сходимости итерационного процесса, которую однако можно улучшить, используя специальные методы, например метод верхней и нижней релаксации [18], основанные на обсуждавшемся нами ранее методе Зейделя.

Пример 10.1. Найти в пакете MATLAB решение задачи Дирихле уравнения Лапласа

$$u_{xx} + u_{yy} = 0\,,$$

с граничными условиями

$$u(x,0) = 0\,,\ u(1,y) = g_2(y)\,,\ u(x,1) = 0\,,\ u(0,y) = g_4(y)\,.$$

1. Создайте файл IterationL.m (листинг 10.1), содержащий описание функции, возвращающей численное решение уравнения Лапласа методом релаксаций (для получения вычислительной схемы, обсуждение которой проводилось ранее, следует положить `Omega=1`).

Листинг 10.1 Файл IterationL.m

```
function z=interationL(N,Omega,Number_of_Iteration,phi)
% функция, возвращающая значения потенциала на каждом шаге
% итерационного процесса
h=1/N; % шаг сетки
% вычисление координат узлов сетки
i=1:N+1;
x(i)=(i-1)*h;
j=1:N+1;
y(j)=(j-1)*h;
% итерационный цикл
for k=1:Number_of_Iteration
  % прохождение по узлам сетки
  for j=2:N
    for i=2:N
     phi(i,j)=(1-Omega)*phi(i,j)+Omega/4*(phi(i+1,j)+phi(i-1,j)+…
              phi(i,j+1)+phi(i,j-1)-h.^2*f(i+1,j+1)); % релаксация
      end;
  end;
  if k==1
     q=phi;
  else
     q=cat(2,q,phi);
  end;
end;
z=q;
```

2. Выполните следующую последовательность команд:

```
N=15;
i=1:N+1;
j=1:N+1;
mu(i,1)=10; % потенциал на левой границе
mu(i,N+1)=-10; % потенциал на правой границе
mu(1,j)=5; % потенциал на нижней границе
mu(N+1,j)=5; % потенциал на верхней границе
% задание начального приближения
kx=2:N;
ky=2:N;
mu(kx,ky)=12; % параметр релаксации
Omega=1;
Niter=200; % число итераций
z=interationL(N,Omega,Niter,mu); % решение уравнения Лапласа
% вычисление векторов и матриц для построения карты линий уровня
```

```
x(i)=(i-1)/N;
y(j)=(j-1)/N;
[x1 y1]=meshgrid(x,y);
K=100; % номер итерации для построения карты линий уровня
N1=(N+1)*K+1;
N2=(N+1)*(K+1);
A=z(1:N+1,N1:N2); % выделение K-го решения из общей матрицы решений
[C,h] = contour(x1,y1,A,17); % построение карты эквипотенциальных
                             % поверхностей K-го решения
                             % уравнения Лапласа (рис. 10.2)
```

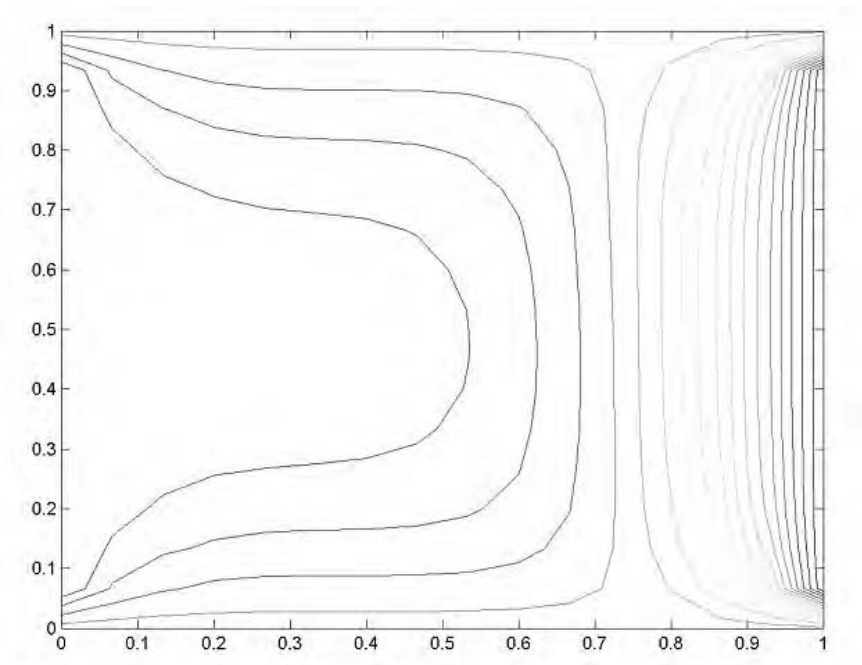


Рис. 10.2. Карта линий равного потенциала численного решения уравнения Лапласа

Для просмотра процесса релаксации решения уравнения Лапласа следует выполнить следующую последовательность команд:

```
% последовательность команд для создания анимационного клипа
set(gca,'nextplot','replacechildren'); % задание режима
                                       % перерисовки карты
                                       % в одном и том же окне
% создание кадров анимационного клипа
for K=2:Niter-1
  N1=(N+1)*(K-1)+1;
```

```
  N2=(N+1)*K;
  A=z(1:N+1,N1:N2);
  [C,h]=contour(x1,y1,A,17);
  F(K-1)=getframe; % создание одного кадра
end;
```

MATLAB, в процессе выполнения цикла по переменной `K`, выводит на экран каждый вновь создаваемый кадр. Для повторного воспроизведения анимационного клипа используется команда `movie(F,n)`, где `F` — имя переменной, в которую помещены кадры клипа, `n` — число повторений клипа, если значение `n` не указано, то клип воспроизводится бесконечное число раз (до закрытия графического окна). Для получения стоп-кадра следует щелкнуть по любому пункту меню графического окна. Для продолжения воспроизведения нужно щелкнуть мышью в поле графика. Можно сохранить созданный анимационный клип в файле, используя команду `save`). Например, для созданного ранее анимационного клипа данная команда имеет следующий синтаксис:

```
>> save  имя_файла  F
```

Для загрузки и выполнения ранее созданного анимационного клипа используется следующая последовательность команд:

```
>> load имя_файла имя_переменной
>> movie(имя_переменной)
```

Рассмотрим решения краевой задачи двумерного уравнения Лапласа для квадратной области ($0 \le x \le 1$ см, $0 \le y \le 1$ см) с известными потенциалами на границах ($u(x,0)=10$, $u(x,1)=-10$, $u(0,y)=5$, $u(1,y)=5$), полученного на сетке, состоящей из 15×15 узлов (рис. 10.2). Из рис. 10.2 видно, что линии, расположенные на карте эквипотенциальных уровней оказываются негладкими (имеют в некоторых точках изломы). Наличие изломов, в свою очередь, у линий равных потенциалов свидетельствует о наличии разрывов у производной функции $\vec{\nabla}\varphi(x,y)$, описывающей напряженность электрического поля. С другой стороны, как известно из теории функций комплексной переменной, функция $\varphi(x,y)$, удовлетворяющая уравнению Лапласа, является аналитической. Необходимым и достаточным условием аналитичности функции $\varphi(x,y)$ является непрерывность ее производных. Этому, как очевидно, не отвечает полученное нами численное решение. Обнаруженный "дефект" численного решения связан с большим шагом сетки, на которой производится поиск решения уравнения Лапласа. Для его устранения можно использовать два способа:

1. находить численное решение на сетке с меньшим шагом;
2. используя найденное числовое решение на сетке, состоящей из 15×15 узлов, находить значения в точках, не совпадающих с узлами сетки, с помощью интерполяционной процедуры.

Для использования сплайн-интерполяции следует дополнить приведенную выше последовательность команд следующими командами:

```
% задание координатной сетки для вычисления значений сплайна
M=100;
n=1:M;
x2(n)=(n-1)/M;
y2(n)=(n-1)/M;
[X2 Y2]=meshgrid(x2,y2);
zi3 = interp2(x1,y1,A,X2,Y2,'splain'); % вычисление значений сплайна
 [C,h]=contour(X2,Y2,zi3,17);
```

Результаты выполнения дополнительных команд представлены на рис. 10.3. Анализ карты эквипотенциальных поверхностей, представленных на рис. 10.2, 10.3, показывает, что, используя сплайн-интерполяцию, удалось устранить недостатки численного решения, проявляющиеся в наличии изломов линий равного потенциала.

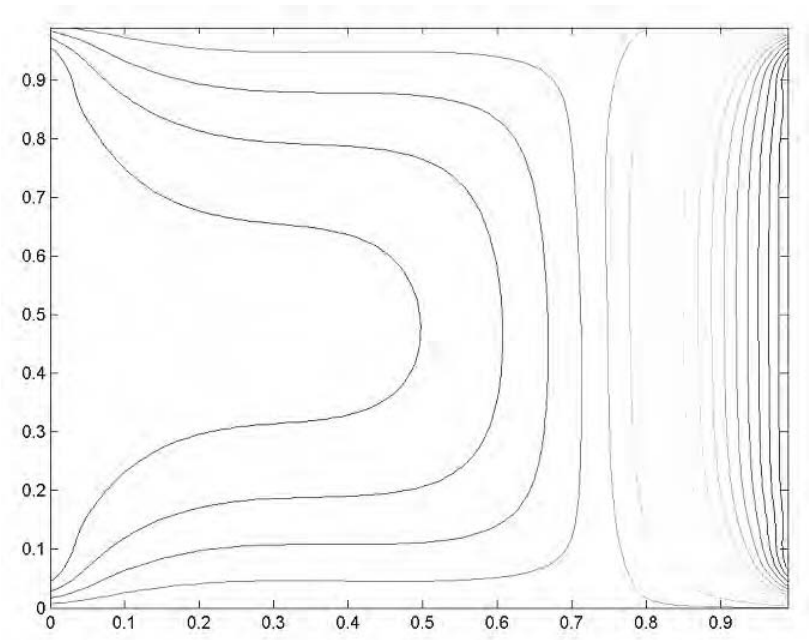


Рис. 10.3. Карта линий уровня интерполированного решения уравнения Лапласа

10.3. Явные разностные схемы для уравнений параболического и эллиптического типов

Явные разностные схемы используются для решения уравнений, в которые входят производные по времени. Для решения данных задач используются явные разностные схемы, с которыми мы познакомимся на примере явной схемы бегущего счета для уравнения теплопроводности.

Рассмотрим решение смешанной краевой задачи для следующего УЧП:

$$u_t = u_{xx}, \quad 0 < x < 1, 0 < t < \infty, \tag{10.9}$$

с граничными условиями

$$\begin{cases} u(0,t) = 1 \\ u_x(1,t) = -[u(1,t) - g(t)] \end{cases}, \quad 0 < t < \infty$$

и начальными условиями

$$u(x,0) = 0, \quad 0 \le x \le 1.$$

Рассматриваемое УПЧ описывает распределение температуры в стержне, начальная температура которого равна нулю. Левый конец стержня находится при постоянной температуре, на правом конце стержня происходит теплообмен с окружающей средой, температура которой определяется функцией $g(t)$.

Для решения задачи методом конечных разностей построим прямоугольную сетку (рис. 10.4), узлы которой определяются формулами:

$$x_j = jh, j = 0,1,2,\ldots,n,$$

$$t_i = ik, i = 0,1,2,\ldots,m.$$

Отметим, что значения u_{ij} на левой и нижней сторонах сетки известны из начальных и граничных условий, на правой границе известно значение частной производной решения уравнения по переменной x.

Заменим частные производные в уравнении теплопроводности их конечно-разностными аппроксимациями:

$$u_t = \frac{1}{k}[u(x,t+k) - u(x,t)] = \frac{1}{k}[u_{i+1\,j} - u_{ij}],$$

$$u_{xx} = \frac{1}{h^2}[u(x+h,t) - 2u(x,t) + u(x-h,t)] = \frac{1}{h^2}[u_{ij-1} - 2u_{ij} + u_{ij-1}].$$

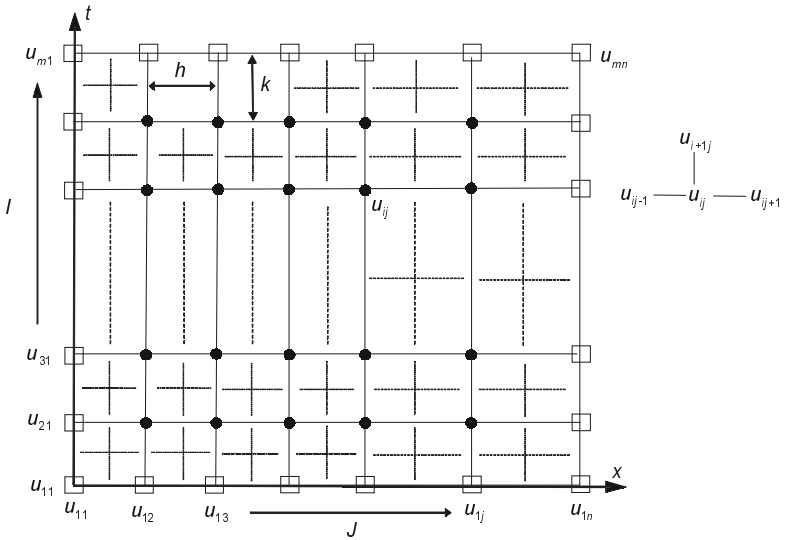


Рис. 10.4. Пространственно-временная сетка, используемая для решения одномерного уравнения теплопроводности

Подставим данные выражения в исходное уравнение $u_t = u_{xx}$ и разрешим получившееся уравнение относительно значений функции на верхнем временном слое. В результате получаем:

$$u_{i+1j} = u_{ij} + \frac{k}{h^2}\left[u_{ij+1} - 2u_{ij} + u_{ij-1}\right]. \qquad (10.10)$$

Формула (10.10) решает поставленную задачу, поскольку она выражает решение в данный момент времени через решение в предыдущий момент времени.

Для решения смешенной краевой задачи необходимо аппроксимировать производную в граничном условии на правом конце:

$$u_x(1,t) = -[u(1,t) - g(t)].$$

Используя конечно-разностную аппроксимацию, получаем:

$$\frac{1}{h}[u_{in} - u_{in-1}] = -[u_{in} - g_i]. \qquad (10.11)$$

Из (10.11) находим:

$$u_{in} = \frac{u_{in-1} - hg_i}{1+h}. \qquad (10.12)$$

Формулы (10.10), (10.12) позволяют начать вычисления по явной схеме.

Алгоритм вычислений по явной схеме реализуется следующей последовательностью действий:

1. Находим решение на сеточном слое $t = \Delta t$, используя явную формулу:

$$u_{2j} = u_{1j} + \frac{k}{h^2}\left[u_{1\,j+1} - 2u_{1j} + u_{1\,j-1}\right], \quad j = 2,3,\ldots,n-1 \tag{10.13}$$

2. Находим величину u_{2n} по формуле (10.12):

$$u_{2,n} = \frac{u_{2,n-1} + hg_2}{1+h}. \tag{10.14}$$

Завершив шаги 1, 2, получаем решение при $t = \Delta t$. Для получения решения при $t = 2\Delta t$ повторяют шаги 1, 2, поднявшись на одну строку вверх, т. е. увеличив i на единицу и используя u_{ij} с предыдущей строки. Аналогично вычисляется решение в последующие моменты времени $t = 3\Delta t, 4\Delta t, \ldots$

У явной схемы имеется один существенный недостаток: если шаг по времени оказывается достаточно большим по сравнению с шагом по x, то погрешности округления могут стать настолько большими, что полученное решение теряет смысл, т. е. решение становится неустойчивым. Можно показать, что для применимости явной схемы должно выполняться условие $k/h^2 \le 0.5$.

Пример 10.2. Найти в пакете MATLAB решение краевой задачи уравнения теплопроводности (10.8) с начальными условиями:

$$T(x,0) = \begin{cases} 0, & \text{если } x \neq 25 \\ 1, & \text{если } x = 25 \end{cases},$$

где $x \in [1, 50]$, и граничными условиями:

$$T(t,1) = 0, \quad T(t,50) = 0,$$

на временном интервале [0, 49].

Решение:

```
>> Nt=50; % число шагов по времени
>> Nx=50; % число узлов координатной сетки
>> t=1:Nt;
>> x=1:Nx;
% задание начальных и граничных условий
>> f(1,x)=0;
>> f(t,1)=0;
>> f(t,Nx)=0;
>> f(1,25)=1;
% вычисление решения в последовательные моменты времени в
% соответствии с (10.10)
```

```
>> for t=2:Nt
     for x=2:Nx-1
       f(t,x)=f(t-1,x)+0.15*(f(t-1,x-1)-2*f(t-1,x)+f(t-1,x+1));
     end;
 end;
>> surf(f) % визуализация численного решения (рис. 10.5)
```

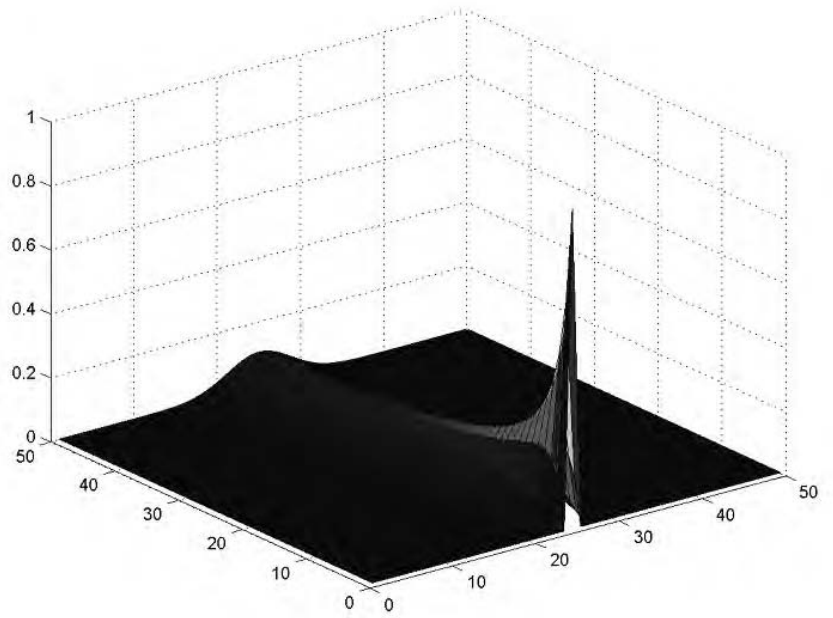


Рис. 10.5. Численное решение уравнения теплопроводности

Получим явную вычислительную схему для нахождения численного решения УЧП гиперболического типа, на примере волнового уравнения:

$$\frac{\partial^2 U}{\partial^2 x} = \frac{1}{v^2}\frac{\partial^2 U}{\partial t^2}. \qquad (10.15)$$

Запишем уравнение (10.15) в конечных разностях:

$$\frac{u_{ij+1} - 2u_{ij} + u_{ij-1}}{h^2} = \frac{1}{v^2}\frac{u_{i+1j} - 2u_{ij} + u_{i-1j}}{\tau^2}. \qquad (10.16)$$

Полученное уравнение позволяет выразить значение функции u в момент времени t_{i+1} через значения функции в предыдущие моменты времени:

$$u_{i+1j} = v^2\left(\frac{\tau}{h}\right)^2\left(u_{ij+1} - 2u_{ij} + u_{ij-1}\right) + 2u_{ij} - u_{i-1j}. \qquad (10.17)$$

Разностная схема (10.17) устойчива, если $\tau \le h/v$.

Отметим, что для начала вычислений в соответствии с (10.17) необходимо знать значения функции в моменты времени t_0 и t_1. Данное обстоятельство обусловлено тем, что мы решаем уравнение второго порядка по времени. Для нахождения решений в моменты времени t_0 и t_1 необходимо использовать, например, начальное условие для первых производных функции u (рис. 10.6).

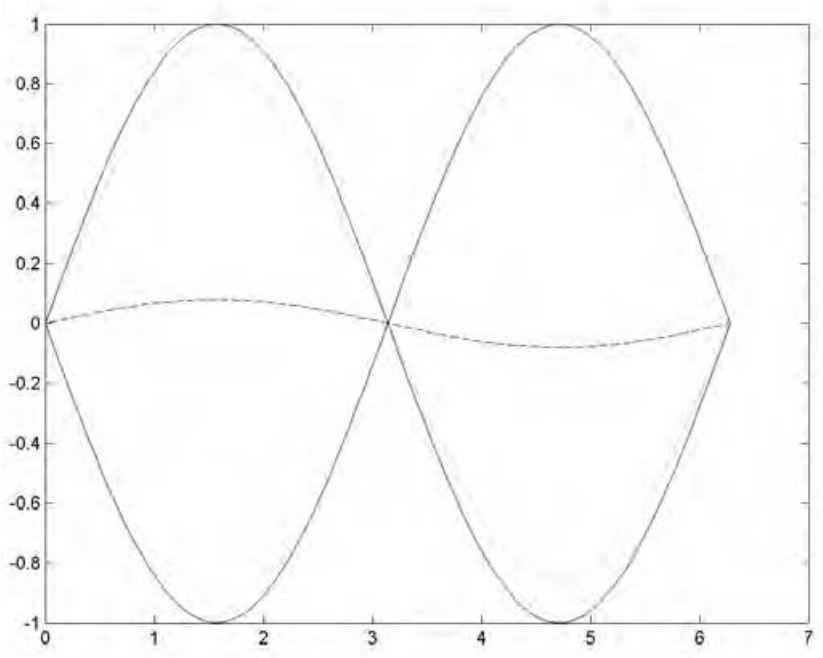


Рис. 10.6. Решения волнового уравнения в моменты времени t = 0, 9, 24, 49

Пример 10.3. Найти в пакете MATLAB решение краевой задачи волнового уравнения (10.14), в котором $v = 1$, с начальными условиями:

$$U(x,0) = \sin(x),$$

где $x \in [0, 2\pi]$, и граничными условиями:

$$U(0, t) = 0,\ U(2\pi,t) = 0$$

на временном интервале [0,59].

Решение:

```
>> Nt=60; % число шагов по времени
>> Nx=100; % число узлов по координате
```

```
% задание начальных и граничных условий
>> j=1:Nx;
>> U(1,j)=sin(2*pi*(j-1)/(Nx-1)); % начальное условие
>> U(2,j)=U(1,j); % граничное условие (концы струны неподвижны)
% вычисление численного решения уравнения в соответствие с (10.17)
>> a=1;k=1;
>> for i=2:Nt
     for j=2:Nx-1
        U(i+1,j)=a.^2*k*(U(i,j+1)-2*U(i,j)+U(i,j-1))+2*U(i,j)-U(i-1,j);
      end;
    end;
% визуализация решений уравнения в выбранные моменты времени
>> j=1:Nx;
>> plot(2*pi*(j-1)/(Nx-1),U(1,j))
>> hold on
>> plot(2*pi*(j-1)/(Nx-1),U(10,j),':')
>> plot(2*pi*(j-1)/(Nx-1),U(25,j),'--')
>> plot(2*pi*(j-1)/(Nx-1),U(50,j),'-')
```

10.4. Неявная разностная схема для уравнения параболического типа

Построим неявную разностную схему для уравнения теплопроводности. Рассмотрим задачу нахождения решения УЧП

$$u_t = u_{xx}, \quad 0 < x < 1, \quad 0 < t < \infty, \tag{10.18}$$

с граничными условиями

$$\begin{cases} u(0,t) = 0 \\ u(1,t) = 0, \end{cases} \quad 0 < t < \infty,$$

и начальными условиями

$$u(x,0) = 1, \quad 0 \le x \le 1.$$

Воспользуемся следующими конечно-разностными аппроксимациями частных производных u_t и u_{xx}:

$$u_t(x,t) = \frac{1}{k}\left[u(x,t+k) - u(x,t)\right],$$

$$u_{xx}(x,t) = \frac{\lambda}{h^2}\left[u(x+h,t+k) - 2u(x,t+k) + u(x-h,t+k)\right] +$$

$$\frac{1-\lambda}{h^2}\left[u(x+h,t) - 2u(x,t) + u(x-h,t)\right],$$

где λ — выбирается из отрезка $[0,1]$.

Отметим, что u_{xx} аппроксимируется взвешенным средним центральных разностных производных в момент времени t и $(t + k)$. При $\lambda = 0.5$ получаем обычное среднее двух центральных производных, при $\lambda = 0$ получаем обычную явную схему, о которой упоминалось ранее.

После замены частных производных u_t, u_{xx} в задаче (10.18) получаем разностную задачу:

$$\frac{1}{k}\left(u_{i+1,j} - u_{i,j}\right) = \frac{\lambda}{h^2}\left(u_{i+1\,j+1} - 2u_{i+1\,j} + u_{i+1,\,j-1}\right) + \frac{1-\lambda}{h^2}\left(u_{ij+1} - 2u_{ij} + u_{ij-1}\right),$$

$$\begin{cases} u_{i,1} = 0, \\ u_{i,n} = 0, \end{cases} \quad i = 1,2,\ldots, m, \tag{10.19}$$

$$u_{1j} = 1, \quad j = 2,\ldots, n-1.$$

Перенесем все неизвестные значения u с верхнего временного слоя (с индексом $(i+1)$) в левую часть уравнения:

$$\begin{aligned} -\lambda r u_{i+1\,j+1} + \left(1 + 2r\lambda\right)u_{i+1,j-1} = \\ = r\left(1-\lambda\right)u_{i,j+1} + \left[1 - 2r\left(1-\lambda\right)\right]u_{ij} + r\left(1-\lambda\right)u_{ij-1}, \end{aligned} \tag{10.20}$$

где $r = k/h^2$.

Таблица 10.1. *Схема уравнения системы (10.20).*

	j - 1		j		j + 1
i + 1	$-r\lambda$	+	$1 + 2r\lambda$	+	$-r\lambda$
i	$r(1 - \lambda)$	+	$1 - 2r\lambda(1 - \lambda)$	+	$r(1 - \lambda)$

Если i фиксировано, а j изменяется от 2 до $(n-1)$, соотношения (10.20) определяют систему $(n-2)$ уравнений с $(n-2)$ неизвестными $u_{i+1,2}, u_{i+1,3}, u_{i+1,4}, \ldots, u_{i+1,n-1}$, которые являются решением задачи во внутренних узлах сетки на временном слое $t = (i+1)\Delta t$.

Наглядное представление о структуре каждого уравнения (10.20) дает таблица 10.1.

Обсудим более подробно алгоритм решения смешанной краевой задачи одномерного уравнения теплопроводности для случаев $\lambda = 0.5$ (схема Кранка-Никольсона), $h = \Delta x = 0.2$, $k = \Delta t = 0.08$ (при этом $r = k/h^2 = 2$). В данном случае сетка содержит 6 узлов вдоль оси x. В соответствии с вычислитель-

ной схемой (10.20), двигаясь слева направо $j = 2, 3, 4, 5$ по первым двум слоям ($i = 1$), получаем следующие четыре уравнения:

$$-u_{21} + 3u_{22} - u_{23} = u_{11} - u_{12} + u_{13} = 1,$$

$$-u_{22} + 3u_{23} - u_{24} = u_{12} - u_{13} + u_{14} = 1,$$

$$-u_{23} + 3u_{24} - u_{25} = u_{13} - u_{14} + u_{15} = 1,$$

$$-u_{24} + 3u_{25} - u_{26} = u_{14} - u_{15} + u_{16} = 1.$$

Перепишем уравнения в матричной форме:

$$\begin{bmatrix} 3 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} u_{22} \\ u_{23} \\ u_{24} \\ u_{25} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Матрица данной системы называется *трехдиагональной*. В наиболее общем виде трехдиагональная система имеет вид:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & \dots & 0 \\ a_1 & b_2 & c_2 & 0 & \dots & 0 \\ 0 & a_2 & b_3 & c_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & c_{n-1} \\ \dots & \dots & \dots & \dots & a_{n-1} & b_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \dots \\ \dots \\ \dots \\ d_n \end{bmatrix}.$$

Для нахождения ее решения преобразуем систему к эквивалентному виду:

$$\begin{bmatrix} 1 & c_1^* & 0 & 0 & \dots & 0 \\ 0 & 1 & c_2^* & 0 & \dots & 0 \\ 0 & 0 & 1 & c_3^* & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & c_{n-1}^* \\ \dots & \dots & \dots & \dots & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ \dots \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ d_3^* \\ \dots \\ \dots \\ \dots \\ d_n^* \end{bmatrix}, \qquad (10.21)$$

где

$$c_1^* = c_1 / b_1 ,$$

$$c_{j+1}^* = \frac{c_{j+1}}{b_{j+1} - a_j c_j^*}, \quad j = 1,2,\dots, n-2,$$

$$d_1^* = d_1 / b_1 ,$$

$$d_{j+1}^* = \frac{d_{j+1} - a_j d_j^*}{b_{j+1} - a_j c_j^*} .$$

Решение системы (10.22) находится последовательно снизу вверх. Для нахождения решения на следующем шаге во времени следует вновь решить аналогичную систему. Отметим, что объем вычислений на каждом шаге больше, чем в явной схеме, но хорошую точность удается получить даже при гораздо большем шаге.

Предваряя решение смешанной краевой задачи, запишем в явном виде систему линейных уравнений для схемы с $\lambda = 1$:

$$\begin{pmatrix} 1+2r & -r & & & \\ -r & 1+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & r & 1+2r & -r \\ & & & -r & 1+2r \end{pmatrix} \begin{pmatrix} u_{i+1,2} \\ u_{i+1,3} \\ \vdots \\ u_{i+1,n-2} \\ u_{i+1,n-1} \end{pmatrix} = \begin{pmatrix} u_{i,1} + r \cdot u_{i,1} \\ u_{i,2} \\ \vdots \\ u_{i,n-2} \\ u_{i,n-1} + r \cdot u_{i,n} \end{pmatrix}, \qquad (10.22)$$

где $u_{i,1} = u(t,0)$, $u_{i,n} = u(t,1)$, которую будем использовать далее при решении смешанной краевой задачи для уравнения параболического вида.

Пример 10.4. Найти в пакете MATLAB решение смешанной краевой задачи УЧП

$$u_t = u_{xx} ,$$

с начальными условиями

$$u(0,x) = \sin(\pi x) + x , \quad 0 \le x \le 1 ,$$

и граничными условиями

$$u(t,0) = 0 , \; u(t,1) = 1$$

с использованием неявной разностной схемы ($\lambda = 1$).

Для нахождения решения смешанной краевой задачи необходимо:

1. Создать файл U1.m (листинг 10.2), содержащий описание функции, возвращающей значения решения смешанной краевой задачи.

Листинг 10.2. Файл U1.m

```
function z=U1(u,A,r,Nt,Nx)
% u – матрица, содержащая начальные и граничные условия
% A – матрица системы (10.22)
% r = k/h^2 – переменная, введенная в (10.20)
% Nt – число шагов по времени
```

```
% Nx – число узлов по координате
z=u;
u1=u(1:1,2:Nx-1);
for i=1:Nt-1
  a=r*z(i,1);
  b=r*z(i,Nx);
  u1(1,1)=u1(1,1)+a;
  u1(1,Nx-2)=u1(1,Nx-2)+b;
  u2=A^-1*u1';
  u1=u2';
  for j=2:Nx-1
    z(i+1,j)=u2(j-1,1);
  end;
end;
```

2. Выполнить следующую последовательность команд:

```
>> Nx=30; % число узлов по координате x
>> Nt=100; % число шагов по времени
% задание начальных условий
>> j=1:Nx;
>> u(1,j)=sin(pi*(j-1)/(Nx-1))+(j-1)/(Nx-1);
% задание граничных условий
>> alpha=0;
>> beta=1;
>> i=1:Nt;
>> u(i,1)=alpha;
>> u(i,Nx)=beta;
% создание матрицы системы линейных уравнений (10.22)
>> r=5;
>> for k=1:Nx-2
        A(k,k)=1+2*r;
      end;
>> for m=2:Nx-2
        A(m-1,m)=-r;
        A(m,m-1)=-r;
     end;
>> z=U1(u,A,r,Nt,Nx); % вычисление решения системы смешанной
                      % краевой задачи
>> surf(z); view(160,70); colormap gray % визуализация решения
                                        % смешанной краевой задачи
                                        % (рис. 10.7)
```

Завершая обсуждение разностных методов решения дифференциальных уравнений в частных производных, отметим, что различные методы и средства для решения данного типа уравнений пакета MATLAB объединены в

специализированный пакет Partial Differential Equations Toolboox (PDE Toolboox), подробное обсуждение которого выходит за рамки нашего курса. Заинтересованный читатель может изучить приемы работы с PDE Toolboox, ознакомившись с руководством пользователя, которое входит в состав документации, поставляемой в электронной форме вместе с пакетом MATLAB (файл PDE.pdf).

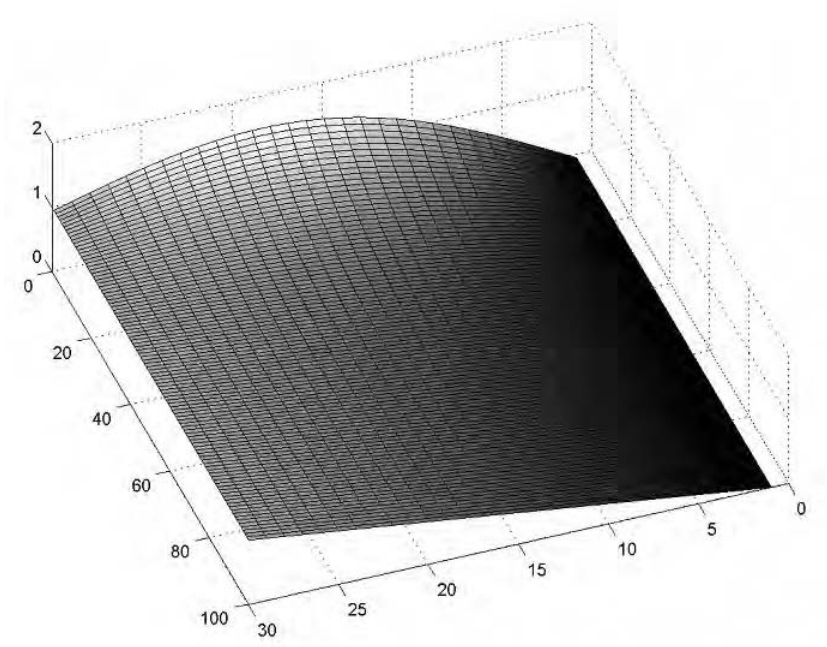


Рис. 10.7. Решение смешанной краевой задачи УЧП параболического типа

10.5. Решение уравнений с частными производными методом Монте-Карло

Рассмотрим решение задачи Дирихле уравнения Лапласа методом случайных блужданий:

$$u_{xx} + u_{yy} = 0, \quad 0 < x < 1, \; 0 < y < 1.$$

Для иллюстрации метода Монте-Карло рассмотрим игру, которая называется "Блуждающий пьяница".

Правила игры:

1. Блуждания пьяницы начинаются из произвольной точки сетки (в нашем случае это точка A (рис. 10.8)).

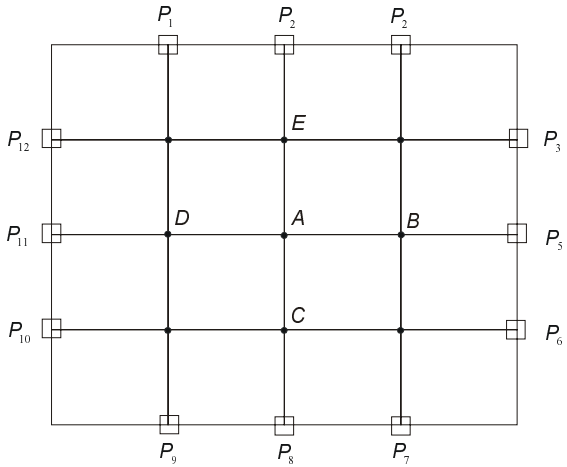


Рис. 10.8. Координатная сетка, используемая в методе Монте-Карло

2. На каждом шаге пьяница случайным образом перемещается в одну из четырех соседних точек сетки (в нашем случае — одна из точек C, B, D, E). Вероятность попадания в каждую из точек равна.
3. После перехода в соседнюю точку процесс блуждания возобновляется. Перемещения пьяного продолжаются до тех пор, пока он не достигнет одной из граничных точек P_i. Номер граничной точки фиксируется, и на этом случайная прогулка заканчивается.
4. Производим повторение шагов 1–3 достаточное количество раз и определяем количество посещений пьяным каждой граничной точки. Отношение числа посещений пьяным каждой точки границы к полному числу испытаний определяет вероятность попадания пьяного в данную точку границы.
5. Предположим, что пьяница получает вознаграждение g_i, если он достигает точки p_i, и предположим, что цель игры — вычислить среднее вознаграждение $R(A)$ для всех случайных прогулок, начинающихся из точки A. Тогда искомый средний выигрыш определяется формулой:

$$R(A) = g_1 P_A(p_1) + g_2 P_A(p_2) + \ldots + g_{12} P_A(p_{12}). \qquad (10.23)$$

Для чего надо играть в "Блуждающего пьяницу"?

Оказывается, что среднее вознаграждение (10.23) является решением задачи Дирихле в точке A. Данный вывод основан на двух фактах.

1. Предположим, что пьяница начал свое движение из точки, лежащей на границе. Каждая такая прогулка заканчивается в той же точке, и пьяница

немедленно получает вознаграждение g_i. Таким образом, среднее вознаграждение для каждой точки равно g_i.

2. Теперь предположим, что прогулка начинается из внутренней точки. Тогда ясно, что среднее вознаграждение для точки $R(A)$ будет средним арифметическим от средних вознаграждений для четырех соседних точек:

$$R(A) = \frac{1}{4}[R(B) + R(C) + R(D) + R(E)]. \qquad (10.24)$$

Итак, мы видим, что $R(A)$ удовлетворяет уравнению (10.24) в каждой внутренней точке и равно g_i в граничной точке. Если g_i — это значения функции $g(x, y)$ из граничного условия в граничных точках p_i, то два наших уравнения точно совпадают с двумя уравнениями, которые получены при решении задачи Дирихле методом конечных разностей. То есть величина $R(A)$ соответствует величине u_{ij} в разностных уравнениях

$$u_{ij} = \frac{1}{4}\left[u_{i-1\,j} + u_{i+1\,j} + u_{ij-1} + u_{i,\,j+1}\right], \qquad (10.25)$$

где (i, j) — внутренняя точка,

$$u_{ij} = g_{ij}, \qquad (10.26)$$

где g_{ij} — значение решения в граничной точке (i, j).

Таким образом, величина $R(A)$ действительно аппроксимирует решение уравнения с частными производными в точке A.

Пример 10.5. Решение методом случайных блужданий в пакете MATLAB уравнения Лапласа:

$$u_{xx} + u_{yy} = 0, \quad 0 < x < 1, \; 0 < y < 1,$$

$$u(0, y) = 0, u(1, y) = 0, \; u(x, 0) = 5, u(x, 1) = 0.$$

1. Создайте файл Monte_Karlo.m (листинг 10.3), содержащий описание функции, возвращающей численное решение уравнения Лапласа.

Листинг 10.3. Файл Monte_Carlo.m

```
function z=Monte_Carlo(Nx,Ny,NTrial,XL,XR,Yu,Yd)
for j=1:Nx
  U(1,j)=Yd(j);
  U(Ny,j)=Yu(j);
end;
for i=1:Ny
  U(i,1)=XL(i);
```

```
  U(i,Ny)=XR(i);
end;
for i=2:Nx-1
  for j=2:Ny-2
    Xs=j;
    Ys=i;
    for m=1:Nx
      VL(m)=0;
      VR(m)=0;
    end;
    for m=1:Ny
      Gd(m)=0;
      Gu(m)=0;
    end;
    for k=1:NTrial
      x=Xs;
      y=Ys;
      while not(x==1)&not(x==Nx)&not(y==1)&not(y==Ny)
        R=rand(1,1);
        if R<0.25
          x=x-1;
        end;
        if (0.25<=R)&(R<0.5)
          x=x+1;
        end;
        if (0.5<=R)&(R<0.75)
          y=y-1;
        end;
        if (0.75<=R)&(R<=1)
          y=y+1;
        end;
      end;
      if x==1
        VL(y)=VL(y)+1;
      end;
      if x==Nx
        VR(y)=VR(y)+1;
      end;
      if y==1
        Gd(x)=Gd(x)+1;
      end;
      if y==Ny
        Gu(x)=Gu(x)+1;
      end;
    end;
```

```
    s1=0;
    for m=2:Nx-1
      s1=s1+Gd(m)*U(1,m)+Gu(m)*U(Ny,m);
    end;
    s2=0;
    for m=2:Ny
      s2=s2+VL(m)*U(m,1)+VR(m)*U(m,Nx);
    end;
    U(i,j)=(s1+s2)/NTrial;
  end;
end;
z=U;
```

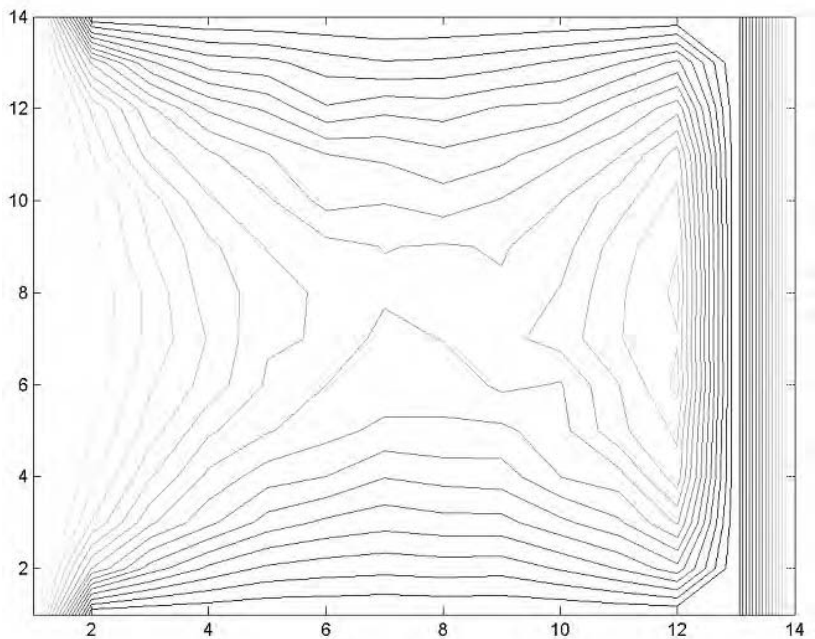


Рис. 10.9. Карта линий уровня решения уравнения Лапласа, полученного методом Монте-Карло

2. Далее необходимо выполнить следующую последовательность команд:

```
% задание числа узлов координатной сетки
>> Nx=14;
>> Ny=14;
% задание граничных условий
>> i=1:Nx;
>> j=1:Ny;
>> XL(i)=5;
```

```
>> Yu(j)=0;
>> XR(i)=5;
>> Yd(j)=0;
% нахождение численного решения
>> U=Monte_Carlo(Nx,Ny,XL,XR,Yu,Yd);
>> contour(U,17) % визуализация численного решения (рис. 10.9)
>> colormap gray
```

Для сравнения на рис 10.10 представлено решение уравнения Лапласа, полученное методом релаксаций.

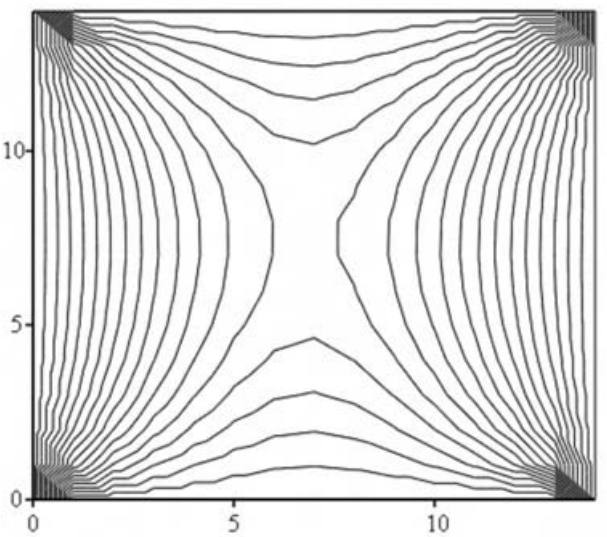


Рис. 10.10. Решение уравнения Лапласа, полученное методом релаксации

Рассмотрим эллиптическую краевую задачу в квадрате

$$u_{xx} + (\sin x)u_{xx} = 0, \quad 0 < x < \pi\ , \quad 0 < y < \pi\ ,$$

$$u(x, y) = g(x, y)$$

на его границе.

Для того чтобы решить эту задачу, заменим u_{xx}, u_{yy} и $\sin x$ следующим образом:

$$u_{xx} = u_{ij+1} - 2u_{ij} + u_{ij-1},$$

$$u_{yy} = u_{i+1j} - 2u_{ij} + u_{i-1j},$$

$$\sin x = \sin x_j\ .$$

Подставим эту замену в уравнение с частными производными. Разрешив получившееся уравнение относительно u_{ij}, получаем

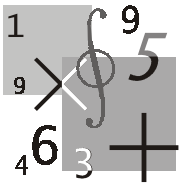
$$u_{ij} = \frac{u_{ij+1} + u_{i,j-1} - \sin x_j \left(u_{i+1\,j} + u_{i-1\,j}\right)}{2\left(1 + \sin x_j\right)}. \qquad (10.27)$$

Коэффициенты при $u_{i+1\,j}, u_{ij+1}, u_{ij-1}, u_{i-1\,j}$ в (10.27) положительны, и их сумма равна единице. Другими словами, решение u_{ij} является взвешенным средним решением в четырех соседних точках.

Следовательно, можно модифицировать метод "Блуждающего пьяницы" так, чтобы вероятности перехода в соседние точки равнялись коэффициенту в соответствующем члене. Другими словами, если пьяница находится в точке (i, j), то он переходит в точку:

- $(i, j+1)$ с вероятностью $\dfrac{1}{2\left(1 + \sin x_j\right)}$,
- $(i, j-1)$ с вероятностью $\dfrac{1}{2\left(1 + \sin x_j\right)}$,
- $(i+1, j)$ с вероятностью $\dfrac{\sin x_j}{2\left(1 + \sin x_j\right)}$,
- $(i-1, j)$ с вероятностью $\dfrac{\sin x_j}{2\left(1 + \sin x_j\right)}$.

В остальном правила игры не меняются. Модификация для других задач может оказаться более хитроумной, однако основная идея метода не меняется.



Лекция 11

Численные методы решения интегральных уравнений

Цель лекции: изложить общие сведения и классификацию интегральных уравнений, рассмотреть квадратурные методы решения интегральных уравнений Фредгольма и Вольтерра и их программные реализации.

11.1. Общие сведения об интегральных уравнениях

Определение 11.1. *Интегральным уравнением* называется уравнение относительно неизвестной функции, содержащейся под знаком интеграла.

К интегральным уравнениям приводятся многие задачи, возникающие в математике и математической физике. Исторически первой задачей, сведенной к интегральному уравнению

$$\int_0^z \frac{\varphi(\xi)}{\sqrt{z-\xi}}\,d\xi = f(z),$$

считается *задача Абеля*, имеющая следующую формулировку.

Определить вид кривой $x = \varphi(z)$, по которой в вертикальной плоскости XoZ под действием силы тяжести должна скатываться материальная точка, так чтобы, начав свое движение с нулевой начальной скоростью из точки $z = z_0$, она достигала оси oX за заданное время $T = f(z)$.

Интегральные уравнения широко используются в моделях, рассматриваемых в теории упругости, газовой динамике, электродинамике, экологии и других областях физики, в которых они являются следствием законов сохранения массы, импульса и энергии. Достоинство данных моделей состоит в том, что интегральные уравнения, в отличие от дифференциальных, не содержат

производных искомой функции и, следовательно, жесткие ограничения на гладкость решения отсутствуют.

Приведем примеры интегральных уравнений:

1. Зависимости между напряжениями и деформациями в упруго-вязких материалах описываются уравнениями

$$\varepsilon(t) = \frac{\sigma(t)}{E} + \frac{1}{E}\int_0^t K(t-\tau)\sigma(\tau)d\tau ,$$

$$\sigma(t) = E\varepsilon(t) - E\int_0^t T(t-\tau)\varepsilon(\tau)d\tau ,$$

где E — модуль упругости, $K(t-\tau)$ — функция влияния напряжения $\sigma(t)$ в момент времени на деформацию $\varepsilon(t)$ в момент времени t, $T(t-\tau)$ — аналогичная функция влияния деформации.

2. Решение задачи об определении вида потенциальной энергии по периоду колебаний [14, § 12] сводится к решению интегрального уравнения

$$T(E) = \sqrt{2m}\int_{x_1(E)}^{x_2(E)} \frac{dx}{\sqrt{E-U(x)}},$$

где E — энергия частицы, m — масса частицы, $x_1(E), x_2(E)$ — координаты точки остановки, являющиеся решением уравнения

$$U(x) = E .$$

Интегральное уравнение в достаточно общем виде можно представить в следующей форме:

$$x(t) = \int_D K(t, s, x(s))ds + f(t), \qquad (11.1)$$

где D — некоторая область n-мерного пространства, x — неизвестная функция, f — известная функция, K — функция относительно x (линейная или нелинейная).

Примечание

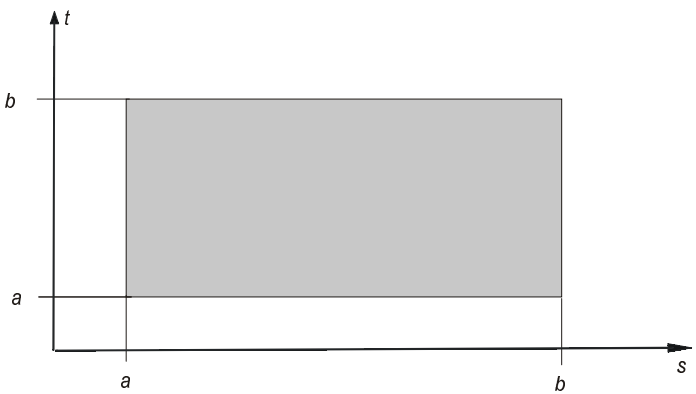
В общем случае функции x(t), f(t) — вектор-функции.

Далее мы ограничим повествование одномерными линейными интегральными уравнениями, в которых функция $x(t)$ является функцией, зависящей от одной переменной, а область D — отрезком конечной длины. В этих уравнениях подынтегральная функция $K(t, s, x(s))$ представима в виде $Q(t, s)x(s)$.

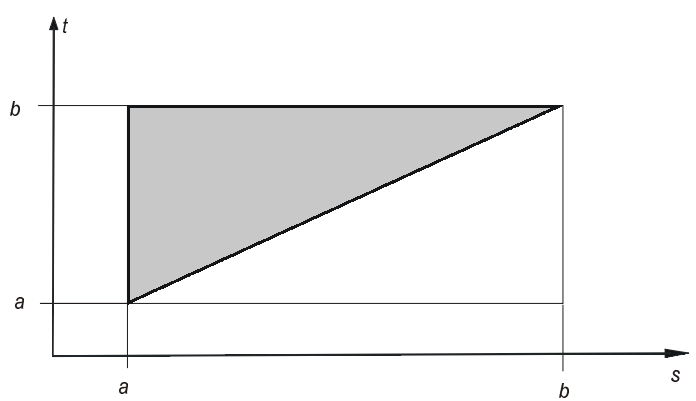
Классификация типов линейных интегральных уравнений проводится по виду верхней границы интеграла в (11.1): если верхняя граница интегрирования является постоянной, то уравнение называется *уравнением Фредгольма*, если переменной — *уравнением Вольтерра*, которые, в свою очередь, подразделяются на уравнения первого и второго рода. На практике наиболее широко применяются линейные интегральные уравнения второго рода:

- Фредгольма

$$x(t) = \lambda \int_a^b Q(t,s)x(s)ds + f(t); \qquad (11.2)$$



а



б

Рис. 11.1. Области задания ядер $Q(t,s)$ интегральных уравнений Фредгольма (а) и Вольтерра (б)

❒ Вольтерра

$$x(t) = \lambda \int_a^t Q(t,s)x(s)ds + f(t), \qquad (11.3)$$

где $f(t)$ — неизвестная функция, $x(t)$ —решение уравнения, $Q(t,s)$ — ядро интегрального уравнения.

Ядро интегрального уравнения Фредгольма определяется на множестве точек квадрата $[a,b]\times[a,b]$ (рис. 11.1а), уравнения Вольтерра — в треугольнике $a \le s \le t \le b$ (рис. 11.1б).

Уравнение Вольтерра можно считать уравнением Фредгольма, если доопределить его ядро $Q(t,s)$ нулем в треугольнике $a \le s \le t \le b$. Тогда можно применять для его решения методы уравнения Фредгольма. Однако при этом могут быть упущены некоторые специфические особенности уравнения Вольтерра, что определяет необходимость их раздельного рассмотрения.

Дополнительный множитель λ, который может быть отнесен к интегральному ядру, в (11.1), (11.2) введен для придания уравнениям более общего вида. Имеются теоремы, устанавливающие существование решений интегральных уравнений при различных значениях λ, которые доказываются подобно тому, как это делается в теории линейных дифференциальных уравнений, через рассмотрение соответствующих однородных уравнений ($f(t) = 0$).

Значительно более сложной задачей оказывается необходимость доказательства существования, единственности и непрерывной зависимости решений от функции $f(t)$ для интегральных уравнений первого рода:

❒ Фредгольма

$$\int_a^b Q(t,s)x(s)ds = f(t); \qquad (11.4)$$

❒ Вольтерра

$$\int_a^t Q(t,s)x(s)ds = f(t). \qquad (11.5)$$

Эта задача относится к классу некорректных задач.

Уравнения первого и второго рода можно записать в общем виде, используя функцию $h(t)$, тожественно равную нулю для уравнений первого рода и единице для уравнений второго рода:

$$h(t)x(t) = \lambda \int_D Q(t,s)x(s)ds + f(t). \qquad (11.6)$$

Когда функция $h(t)$ обращается в ноль в некоторых точках прямоугольника интегрирования, уравнение (11.6) относится к интегральным уравнения третьего рода. Уравнения данного типа встречаются в приложениях значительно реже, чем уравнения первых двух типов, и значительно менее изучены.

Многие используемые на практике интегральные уравнения имеют ядро, зависящее только от разности $(t - s)$. Интегральные уравнения с данным типом ядра называются *уравнениями с разностным ядром*. Примером таких уравнений является уравнение, получаемое в задаче Абеля.

Если $Q(t,s)$ и $f(t)$ — непрерывные функции, то при любых значениях параметра λ существует единственно непрерывное решение уравнения Вольтерра второго рода (11.5). Для уравнения Фредгольма второго рода (11.4) при тех же требованиях единственное непрерывное решение существует, например, при условии, что

$$|\lambda| < \frac{1}{C(b-a)}, \tag{11.7}$$

где $C = \max\limits_{t,s\in[a,b]} |Q(t,s)|$.

При снижении требований к гладкости возможных решений условие (11.7) ослабляется. Например, для функций, интегрируемых с квадратом, в роли достаточного условия фигурирует неравенство

$$|\lambda| < \frac{1}{\sqrt{\int\limits_a^b \int\limits_a^b Q^2(t,s)\,dt\,ds}}. \tag{11.8}$$

Известны формулы (или совокупности формул) позволяющие найти точное решение $x(t)$. Например, решение уравнения Вольтерра с $\lambda = 1$ и мультипликативным ядром

$$Q(t,s) = p(t)s(t) \tag{11.9}$$

вычисляется по формуле

$$x(t) = \int\limits_a^t R(t,s) f(s)\,ds + f(t), \tag{11.10}$$

где

$$R(t,s) = p(t)q(s)e^{\int\limits_s^t p(\xi)\,q(\xi)\,d\xi}. \tag{11.11}$$

Решение уравнения Фредгольма с вырожденным ядром

$$Q(t,s)=\sum_{i=1}^{m} p_i(t)q_i(s) \tag{11.12}$$

имеет вид

$$x(t)=\lambda\sum_{i=1}^{m} z_i p_i(t)+f(t)\,, \tag{11.13}$$

где числа z_i — решения системы линейных алгебраических уравнений:

$$\begin{cases} z_1-\lambda c_{11}z_1-\lambda c_{12}z_2-\ldots-\lambda c_{1m}z_m=d_1 \\ z_2-\lambda c_{21}z_1-\lambda c_{22}z_2-\ldots-\lambda c_{2m}z_m=d_2 \\ \qquad\qquad\qquad\vdots \\ z_1-\lambda c_{m1}z_1-\lambda c_{m2}z_2-\ldots-\lambda c_{mm}z_m=d_m \end{cases}, \tag{11.14}$$

где $c_{ij}=\int_a^b q_i(s)p_j(s)ds$, $d_i=\int_a^b q_i(s)f(s)ds$.

Условие существования и единственности решения уравнения Фредгольма с вырожденным ядром, как очевидно, зависит от значения определителя $D(\lambda)$ системы линейных алгебраических уравнений (11.14), называемого *определителем Фредгольма*. Если $D(\lambda)\neq 0$, то решение существует и единственно.

Наличие методов нахождения точного решения интегрального уравнения с вырожденным ядром позволяет построить приближенный метод, в основе которого лежит замена одного уравнения другим, ядро которого вырождено и в некотором смысле близко к ядру исходного уравнения. Данная замена ядра опирается на различные способы локальной аппроксимации функций, зависящих от двух переменных. Помимо упомянутого ранее метода замены ядра на вырожденное известен ряд других приближенно-аналитических методов решения интегральных уравнений, например, метод последовательных приближений, метод моментов и др. Подробное описание таких методов можно найти в книгах [16, 17, 19–21].

Далее мы рассмотрим численные методы решения интегральных уравнений, в основе которых лежит замена интеграла в интегральном уравнении конечной суммой, с помощью какой-либо квадратурной формулы. Это позволят свести решение исходной задачи к решению системы линейных алгебраических уравнений, число которых определяется числом узлов временной сетки. Методы решения интегральных уравнений, основанные на данном подходе, называются *квадратурными методами* или *методами конечных сумм*. Преимущество данных методов состоит в простоте их реализации. Отме-

тим, что без каких-либо изменений данные методы можно применять для решения нелинейных интегральных уравнений, имея в виду, что в этом случае приходится решать систему нелинейных алгебраических уравнений.

11.2. Квадратурный метод решения интегральных уравнений Фредгольма

Заменим определенный интеграл (11.4) его приближенным значением, вычисляемым с помощью конкретной квадратурной формулы

$$\int_a^b \varphi(s)ds \approx \sum_j^n A_j \varphi(s_j), \tag{11.14}$$

где $j = 1n$ — номера узлов сетки по переменной, A_j — весовые коэффициенты квадратурной формулы.

Подставив правую часть приближенного равенства (11.14) с $\varphi(s) = Q(t,s)\cdot x(s)$ вместо интеграла в интегральное уравнение Фредгольма второго рода (11.4), получим

$$x(t) \approx \lambda \sum_{j=1}^n A_j Q(t, s_j) x(s_j) + f(t). \tag{11.15}$$

Выражение (11.15) задает функцию, описывающую приближенное решение интегрального уравнения (11.4). Введем на отрезке $[a, b]$ дискретную временную сетку $t_1, t_2, , t_n$, узлы которой совпадают с узлами сетки $s_1, s_2, , s_n$. Для каждого момента времени t_j выполняется равенство

$$x(t_i) \approx \lambda \sum_{j=1}^n A_j Q(t_i, s_j) x(s_j) + f(t_i), \tag{11.16}$$

где $i = 1n$.

Введем обозначения $Q_{ij} = Q(t_i, s_j)$, $f_i = f(t_i)$, $x_i = x(t_i)$ и запишем (11.16) в виде системы n линейных алгебраических уравнений с n неизвестными:

$$\begin{cases} (1-\lambda A_1 Q_{11})x_1 - \lambda A_2 Q_{12} x_2 - \ldots - \lambda A_n Q_{1n} x_n = f_1 \\ (1-\lambda A_2 Q_{21})x_1 - \lambda A_2 Q_{22} x_2 - \ldots - \lambda A_n Q_{2n} x_n = f_2 \\ \vdots \\ (1-\lambda A_n Q_{n1})x_1 - \lambda A_2 Q_{n2} x_2 - \ldots - \lambda A_n Q_{nn} x_n = f_n. \end{cases} \tag{11.17}$$

для решения которой можно использовать любой из методов, подробно обсуждавшихся в *лекции 3*.

Таким образом, нахождение решения уравнения Фредгольма второго рода осуществляется в соответствие со следующим алгоритмом:

1. Задать временную сетку t_i.
2. Вычислить значения функции $f(t)$ в узлах временной сетки.
3. Вычислить элементы матрицы, составленной из коэффициентов системы линейных алгебраических уравнений.
4. Решить систему линейных уравнений.

Пример 11.1. Найти в пакете MATLAB решение интегрального уравнения

$$x(t) = \int_1^2 \frac{x(s)ds}{\sqrt{t+s^2}} + \sqrt{t+1} - \sqrt{t+4} + t\,.$$

(Точное решение уравнения $x(t) = t$.)

1. Создайте файл q11.m (листинг 11.1), содержащий описание функции, возвращающей значения функции $Q(t,s)$.

Листинг 11.1. Файл q11.m

```
function z=Q(t,s)
z=1./sqrt(t+s.^2);
```

2. Создайте файл F11.m (листинг 11.2), содержащий описание функции, возвращающей значения функции $f(t)$.

Листинг 11.2. Файл f11.m

```
function z=f11(t)
z=sqrt(t+1)-sqrt(t+4)+t;
```

3. Создайте файл Solve_g11.m (листинг 11.3), содержащий описание функции, возвращающей решение интегрального уравнения.

Листинг 11.3. Файл Solve.m

```
function [X,Y]=solve_g11(x1,x2,N,Lambda)
% задание временной сетки
h=(x2-x1)/(N-1);
i=1:N;
t(i)=x1+h*(i-1);
s=t;
% задание коэффициентов квадратурной формулы метода трапеций
A(1)=0.5;
m=2:N-1;
```

```
A(m)=1;
A(N)=0.5;
% вычисление значений функции Q(t,s) в узлах сетки
for i=1:N
  for j=1:N
    q(i,j)=Q11(t(i),s(j));
  end;
end;
% вычисление значений функции f(t) в узлах временной сетки
F=f11(t);
for i=1:N
  for j=1:N
    if i==j
      M(i,j)=1-Lambda*A(i)*q(i,j)*h;
    else
      M(i,j)=-Lambda*A(i)*q(i,j)*h;
    end;
  end;
end;
X=t;
Y=M^-1*F'; % нахождение решения интегрального уравнения
```

4. Выполните следующую последовательность команд:

```
>> x1=1; % левая граница отрезка поиска решения
>> x2=2; % правая граница отрезка поиска решения
>> Lambda=1;
>> N=300; % число узлов разбиения отрезка
>> [X Y]=solve_g11(x1,x2,N,Lambda); % вычисление решения
                                    % интегрального уравнения
>> plot(X,Y) % визуализация решения интегрального уравнения (рис. 11.2)
```

Точность численного решения интегрального уравнения зависит от нескольких факторов: применяемой квадратурной формулы, числа узлов временной сетки, свойств функции $Q(t,s)$. В ряде книг приводятся аналитические выражения, позволяющие оценить максимальную погрешность численного решения при использовании различных вычислительных схем [16,19–21]. Однако эти оценки оказываются малопригодными из-за их громоздкости, поэтому на практике используют менее строгий метод контроля точности численного решения — принцип Рунге. Данный принцип заключается в сравнении численных решений, полученных на временных сетках с шагом $2h$ и h, в одних и тех же узлах временной сетки. Абсолютное значение разности этих решений характеризует величину погрешности численного решения. Недостаток настоящего подхода состоит в том, что при данном способе контроля приходится ограничиваться квадратурными формулами, пригодными только для сеток с равномерным шагом.

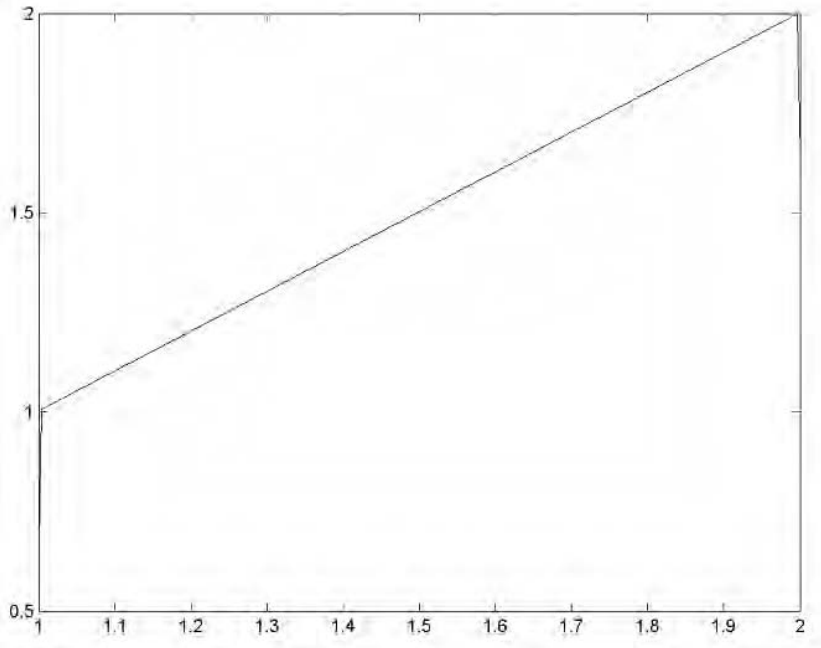


Рис. 11.2. Численное решение интегрального уравнения $x(t)=\int_1^2 \frac{x(s)\,ds}{\sqrt{t+s^2}}+\sqrt{t+1}-\sqrt{t+4}+t$

Важно понимать, что необходимо согласовывать выбор конкретной квадратурной формулы (порядок ее точности) со степенью гладкости ядра интегрального уравнения. Если ядро и свободный член оказываются недостаточно гладкими, то для вычисления интеграла не следует применять высокоточные квадратуры, а лучше ограничится такими формулами, как формулы трапеций и прямоугольников.

Это замечание иллюстрирует рис. 11.3, на котором представлено решение уравнения из примера 11.1, полученное при использовании квадратурной формулы Симпсона. Далее приведен листинг файла Solve2_g11.m (листинг 11.4), содержащего описание соответствующей функции.

Листинг 11.4. Файл Solve2_g11.m

```
function [X,Y]=solve2_g11(x1,x2,N,Lambda)
% задание временной сетки
h=(x2-x1)/(N-1);
i=1:N;
t(i)=x1+h*(i-1);
```

```
s=t;
% задание коэффициентов квадратурной формулы метода Симпсона
A(1)=1/3;
A(N)=1/3;
k=0;
for i=2:N-1
  if k==0
    A(i)=4/3;
    k=1;
  else
    A(i)=2/3;
    k=0;
  end;
end;
% вычисление значений функции Q(t,s) в узлах сетки
for i=1:N
  for j=1:N
    q(i,j)=Q11(t(i),s(j));
  end;
end;
% вычисление значений функции f(t) в узлах временной сетки
F=f11(t);
for i=1:N
  for j=1:N
    if i==j
      M(i,j)=1-Lambda*A(i)*q(i,j)*h;
    else
      M(i,j)=-Lambda*A(i)*q(i,j)*h;
    end;
  end;
end;
X=t;
Y=M^-1*F'; % нахождение решения интегрального уравнения
```

Для получения численного решения интегрального уравнения следует создать файлы q11.m, f11.m, Solve2_g11 и затем выполнить следующую последовательность команд:

```
>> x1=1; % левая граница отрезка поиска решения
>> x2=2; % правая граница отрезка поиска решения
>> Lambda=1;
>> N=300; % число узлов разбиения отрезка
>> [X Y]=solve2_g11(x1,x2,N,Lambda); % вычисление решения
                                     % интегрального уравнения
>> plot(X,Y) % визуализация решения интегрального уравнения (рис. 11.3)
```

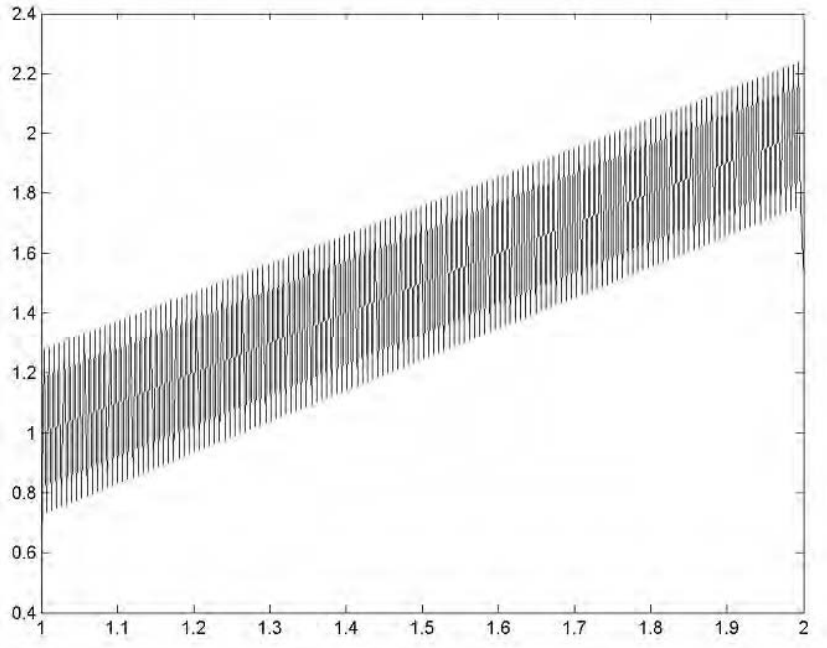


Рис. 11.3. Численное решение интегрального уравнения $x(t)=\int_1^2 \frac{x(s)\,ds}{\sqrt{t+s^2}}+\sqrt{t+1}-\sqrt{t+4}+t$ при использовании формулы Симпсона

11.3. Квадратурный метод решения интегральных уравнений Вольтерра

Так как линейные интегральные уравнения Вольтерра в отличие от уравнения Фредгольма имеют единственное непрерывное решение при любых значениях параметра λ, при нахождении численного решения уравнения

$$x(t)=\int_a^t Q(t,s)x(s)ds+f(t), \tag{11.18}$$

где $t\in[a,b]$, можно положить $\lambda=1$.

Учитывая, что уравнение Вольтерра формально можно считать уравнением Фредгольма вида

$$x(t)=\int_a^t Q(t,s)x(s)ds+f(t) \tag{11.19}$$

с ядром

$$K(t,s)=\begin{cases} Q(t,s) \text{ при } a \le s \le t \le b \\ 0 \text{ при } a \le t \le s \le b \end{cases}, \qquad (11.20)$$

для нахождения решения рассматриваемого уравнения воспользуемся результатами предыдущего раздела.

Введем в рассмотрение временную сетку $s_j \in [a, b]$, состоящую из n узлов, и выберем конкретную квадратурную формулу с весами A_j, тогда приближенное решение интегрального уравнения принимает вид (11.16). Составим систему линейных алгебраических уравнений, аналогичную системе (11.17), которая в силу свойств ядра интегрального уравнения (11.20) вырождается в треугольную:

$$\begin{cases} (1-A_1Q_{11})x_1 = f_1 \\ -A_1Q_{21}x_1 + (1-A_2Q_{22})x_2 = f_2 \\ \vdots \\ -A_1Q_{n1}x_1 - A_2Q_{n2}x_2 - \ldots + (1-A_nQ_{nn})x_n = f_n. \end{cases} \qquad (11.21)$$

Из (11.21) видно, что искомые значения x_1, x_2, , x_n находятся последовательными вычислениями по следующим формулам:

$$x_1 = \frac{f_1}{1-A_1Q_{11}}, \qquad (11.22)$$

$$x_i = \frac{f_i + \sum_{j=1}^{i-1} A_jQ_{ij}x_j}{1-A_iQ_{ii}}, \qquad (11.23)$$

где $i = 2, , n$.

Пример 11.2. Решение в пакете MATLAB интегрального уравнения

$$x(t) = \int_0^t t\cos^2\left(ts^3\right)x(s)ds + t^2 - \frac{1}{3}tg\left(t^4\right).$$

(Точное решение уравнения $x = t^2$.)

1. Создайте файл q11_2.m (листинг 11.5), содержащий описание функции, возвращающей значения подынтегральной функции.

Листинг 11.5. Файл q11_2.m

```
function z=Q11_2(t,s)
z=t*cos(t*s.^3).^2;
```

2. Создайте файл F11_2.m (листинг 11.6), содержащий описание функции, возвращающей значения функции $f(t)$.

Листинг 11.6. Файл F11_2.m

```
function z=f11_2(t)
z=t.^2-1/3*tan(t.^4);
```

3. Создайте файл Solve3_g11.m (листинг 11.7), содержащий описание функции, возвращающей решение интегрального уравнения.

Листинг 11.7. Файл Solve3_g11.m

```
function [T,Y]=Solve3_g11(t1,t2,N)
% задание временной сетки
h=(t2-t1)/(N-1);
i=1:N;
t(i)=t1+h*(i-1);
s=t;
% задание коэффициентов квадратурной формулы метода трапеций
A(1)=0.5;
m=2:N-1;
A(m)=1;
A(N)=0.5;
% вычисление значений функции Q(t,s) в узлах сетки
for i=1:N
  for j=1:N
    Q(i,j)=Q11_2(t(i),s(j));
  end;
end;
% вычисление значений функции f(t) в узлах временной сетки
F=f11_2(t);
% вычисление решения интегрального уравнения
x(1)=F(1)./(1-A(1)*Q(1,1));
for m=2:N
  S=0;
  for j=1:m-1
    S=S+h.*A(j).*Q(m,j).*x(j);
  end;
  x(m)=(F(m)+S)./(1-h.*A(m).*Q(m,m));
end;
T=t;
Y=x;
```

4. Выполните следующую последовательность команд:

```
>> t1=0; % левая граница отрезка поиска решения
>> t2=0.5; % правая граница отрезка поиска решения
>> N=300; % число узлов разбиения отрезка
>> [X Y]=solve3_g11(x1,x2,N,Lambda); % вычисление решения
                                     % интегрального уравнения
>> plot(X,Y) % визуализация решения интегрального
             % уравнения (рис. 11.4)
```

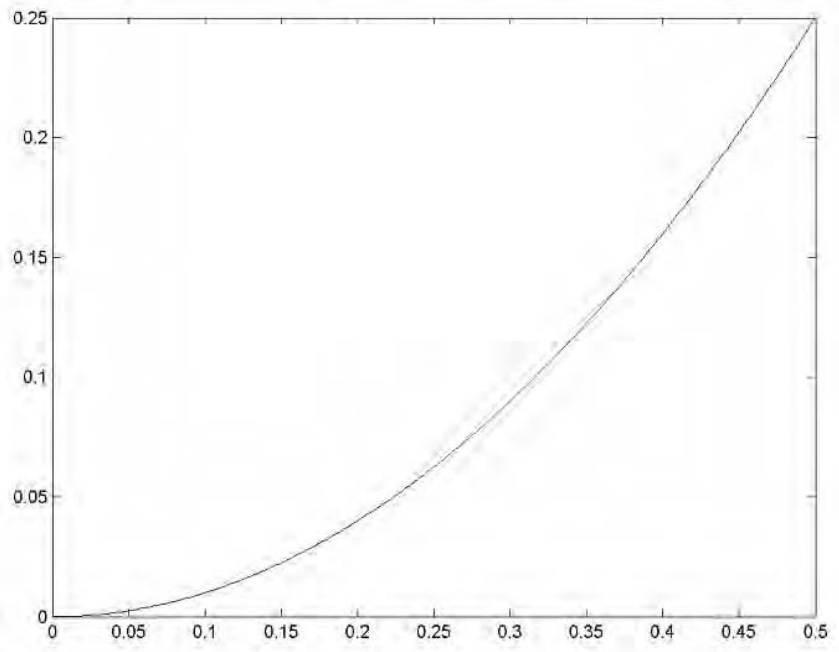


Рис. 11.4. Численное решение интегрального уравнения

$$x(t) = \int_0^t t\cos^2\left(ts^3\right)x(s)\,ds + t^2 - \frac{1}{3}tg\left(t^4\right)$$

```
% оценка параметров решения интегрального уравнения методом
% наименьших квадратов
>> Z=[ones(size(X')) X' X'.^2];
>> a=Z\Y'
a =

   -0.0000
   -0.0000
    1.0003
```

```
>> format long
>> a

a =

   1.0e+002 *

  -0.00000187595514
  -0.00000470664690
   1.00030838604155
```

Получим расчетные формулы для решения уравнения Вольтерра первого рода (11.5) при использовании метода средних прямоугольников. Решения уравнения будем находить в узлах временной сетки

$$t_1 = a + h,\ t_2 = t_1 + h,\ \ldots,\ t_i = t_{i-1} + h\,. \tag{11.24}$$

Подставляя (11.24) в (11.4), получаем равенства

$$\int_a^{t_i} Q(t_i, s)x(s)ds = f(t_i)\,. \tag{11.25}$$

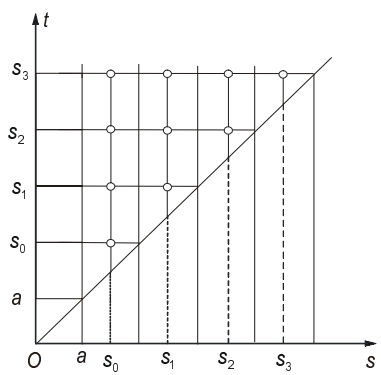


Рис. 11.5. Пространственно-временная сетка, используемая для решения уравнения Вольтерра

Из (11.25) видно, что в данном случае условие совпадения узлов квадратур s_i с узлами временной сетки t_i не является обязательным, поэтому их можно выбрать посередине элементарных промежутков интегрирования $[t_{i-1}, t_i]$ (рис. 11.5). Выбор данной сетки означает, что

$$x_1 \approx x(s_1), x_2 \approx x(s_2), \ldots, x_n \approx x(s_n)\,. \tag{11.26}$$

Учитывая выбор квадратурной формулы и условия (11.26), запишем (11.25) в следующем виде:

$$\begin{aligned} hQ(t_1,s_1)x_1 &= f(t_1),\\ hQ(t_2,s_1)x_1 + hQ(t_2,s_2)x_2 &= f(t_2),\\ hQ(t_3,s_1)x_1 + hQ(t_3,s_2)x_2 + hQ(t_3,s_3)x_3 &= f(t_3),\\ &\dots, \end{aligned} \qquad (11.27)$$

где $s_i = t_i - \frac{h}{2}$.

Из равенств (11.27) последовательно находим:

$$x_1 = \frac{f(t_1)}{hQ(t_1,s_1)}, \qquad (11.28)$$

$$x_i = \frac{f(t_i) - h\sum_{j=1}^{i-1} Q(t_i,s_j)x_j}{hQ(t_i,s_i)}, \qquad (11.29)$$

где $i = 2, 3,$

Пример 11.4. Найти в пакете MATLAB решение интегрального уравнения

$$\int_1^t \left(t^2 + s^2 + 1\right)x(s)ds = t^2 - \frac{1}{t},$$

используя систему узлов координатно-временной стеки, представленную на рис. 11.5.

(Точное решение уравнения $x(t) = \frac{1}{t^2}$.)

1. Создайте файл F11_4.m (листинг 11.8), содержащий описание функции, возвращающей значения функции $f(t)$.

Листинг 11.8. Файл F11_4.m

```
function z=F11_4(t)
z=t.^2-1./t;
```

2. Создайте файл Q11_4.m (листинг 11.9), содержащий описание функции, возвращающей значения функции $Q(t,s)$.

Листинг 11.9. Файл Q11_4.m

```
function z=Q11_4(t,s)
z=t.^2+s.^2+1;
```

3. Создайте файл Volterra11.m (листинг 11.10), содержащий описание функции, возвращающей решения интегрального уравнения Вольтерра.

Листинг 11.10. Файл Volterra11.m

```
function [T,Z]=Volterra11(t1,t2,N)
% задание временной сетки
h=(t2-t1)/(N-1);
i=1:N;
t(i)=t1+h*i;
s=t+h/2;
% вычисление значений ядра интегрального уравнения в узлах
% временной сетки
for i=1:N
  for j=1:N
    q(i+1,j+1)=Q11_4(t(i),s(j));
  end;
end;
% вычисление значений функции f(t) в узлах временной сетки
F(1)=F11_4(t(1));
F(i+1)=F11_4(t(i));
% Вычисление значений решения интегрального уравнения в
% соответствие с (11.28),(11.29)
x(1)=F(1)/(h*q(1,1));
for m=2:N
  s=0;
  for j=2:m-1
    s=s+q(m,j)*x(j);
  end;
  x(m)=(F(m)-h*s)/(h*q(m,m));
end;
T=t;
Z=x;
```

4. Выполните следующую последовательность команд:

```
>> t1=1; % левая граница интервала поиска решения
>> t2=1.3; % правая граница интервала поиска решения
>> N=300; % число интервалов разбиения отрезка [t1,t2]
>> [T Z]=Volterra11(t1,t2,N); % численного решения
                              % интегрального уравнения Вольтерра
>> plot(T,Z); % визуализация решения уравнения Вольтерра (рис. 11.6)
```

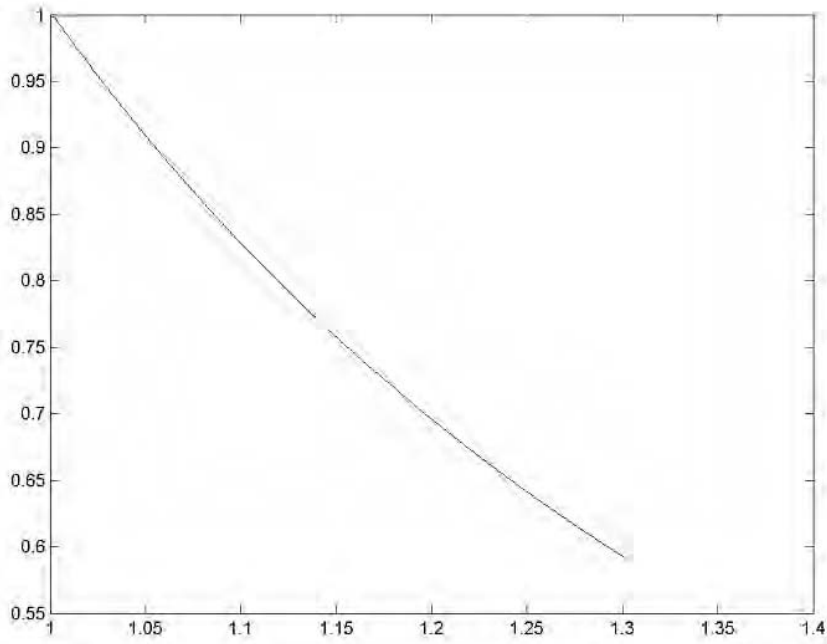


Рис. 11.6. Численное решение интегрального уравнения $\int_1^t \left(t^2 + s^2 + 1\right) x(s) ds = t^2 - \frac{1}{t}$, полученное на пространственно-временной сетке, представленной на рис. 11.5

При использовании квадратурных формул замкнутого типа и совпадающей системы узлов возникает проблема вычисления значения $x_1 \approx x(t_1) = x(s_1) = x(a)$, которая не возникала при решении уравнений Вольтерра второго рода. Действительно, при $i = 1$ равенство (11.25) теряет всякий смысл.

Для нахождения начального значения x_1 продифференцируем уравнение (11.5) по t:

$$Q(t,t)x(t) + \int_a^t Q'_t(t,s)x(s)ds = f'(t). \qquad (11.30)$$

Положив в (11.30) $t = a$, получим

$$Q(a,a)x(a) = f'(a). \qquad (11.31)$$

Откуда следует, что

$$x(a) = \frac{f'(a)}{Q(a,a)},$$

следовательно,

$$x_1 = \frac{f'(a)}{Q_{11}}. \tag{11.32}$$

При использовании квадратурной формулы трапеций далее получаем:

$$\frac{h}{2}Q_{21}x_1 + \frac{h}{2}Q_{22}x_2 = f_2 \Rightarrow x_2 = \frac{f_2 - \frac{h}{2}Q_{21}x_1}{\frac{h}{2}Q_{22}},$$

$$\frac{h}{2}Q_{31}x_1 + hQ_{32}x_2 + \frac{h}{2}Q_{33}x_3 = f_3 \Rightarrow x_3 = \frac{f_3 - \frac{h}{2}Q_{31}x_1 - hQ_{32}x_2}{\frac{h}{2}Q_{33}},$$

т. е. в общем случае при любом $j = 2, 3,$

$$x(s_j) \approx x_j = \frac{f_j - \frac{h}{2}Q_{j1}x_1 - h\sum_{k=2}^{j-1}Q_{jk}x_k}{\frac{h}{2}Q_{jj}}. \tag{11.33}$$

Пример 11.4. Найти в пакете MATLAB решение интегрального уравнения

$$\int_1^t \left(t^2 + s^2 + 1\right)x(s)ds = t^2 - \frac{1}{t},$$

используя замкнутые квадратурные формулы.

1. Создайте файл F11_4.m (листинг 11.11), содержащий описание функции, возвращающей значения функции $f(t)$.

Листинг 11.11. Файл F11_4.m

```
function z=F11_4(t)
z=t.^2-1./t;
```

2. Создайте файл dF11_4.m (листинг 11.12), содержащий описание функции, возвращающей значения производной функции $f(t)$.

Листинг 11.12. Файл dF11_4.m

```
function z=dF11_4(t)
z=2*t+1/t.^2;
```

3. Создайте файл Q11_4.m (листинг 11.13), содержащий описание функции, возвращающей значения функции $Q(t,s)$.

Листинг 11.13. Файл Q11_4.m

```
function z=Q11_4(t,s)
z=t.^2+s.^2+1;
```

4. Создайте файл Volterra11_2.m (листинг 11.14), содержащий описание функции, возвращающей решения интегрального уравнения Вольтерра на совпадающей сетке.

Листинг 11.14. Файл Volterra11_2.m

```
function [T,Z]=Volterra11_2(t1,t2,N)
% задание временной сетки
h=(t2-t1)/(N-1);
i=1:N;
t(i)=t1+h*(i-1);
s=t;
% вычисление значений ядра интегрального уравнения в узлах
% временной сетки
for i=1:N
  for j=1:N
    q(i,j)=Q11_4(t(i),s(j));
  end;
end;
% вычисление значений решения интегрального уравнения в
% соответствие с (11.32), (11.33)
F=F11_4(t);
F(1)=dF11_4(t(1));
x(1)=F(1)/q(1,1);
x(2)=(F(2)-h/2*q(2,1)*x(1))/(h/2*q(2,2));
for m=2:N
  s=0;
  for j=2:m-1
    s=s+q(m,j)*x(j);
  end;
  x(m)=(F(m)-h/2*q(m,1)*x(1)-h*s)/(h/2*q(m,m));
end;
T=t;
Z=x;
```

5. Выполните следующую последовательность команд:

```
>> t1=1; % левая граница интервала поиска решения
>> t2=1.3; % правая граница интервала поиска решения
>> N=300; % число интервалов разбиения отрезка [t1,t2]
```

```
>> [T Z]=Volterra11_2(t1,t2,N); % численного решения
                               % интегрального уравнения Вольтерра
>> plot(T,Z); % визуализация решения уравнения Вольтерра (рис. 11.7)
```

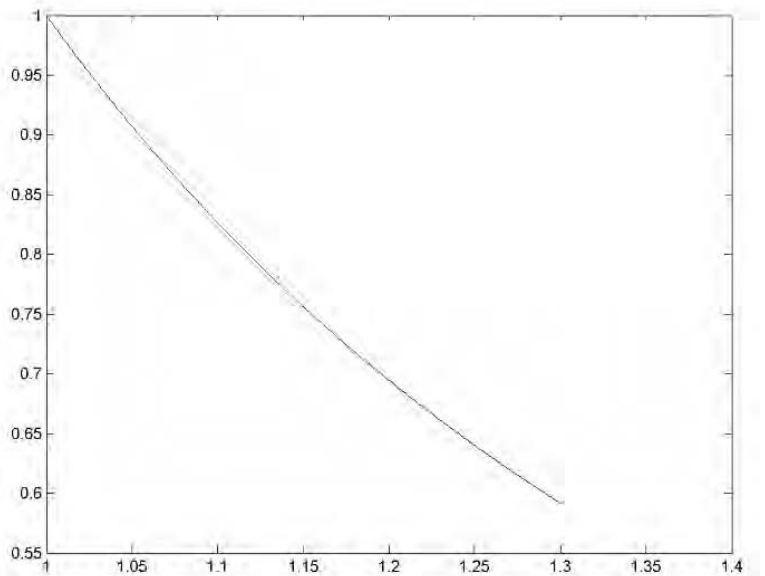
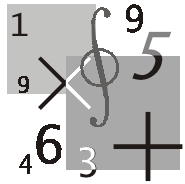


Рис. 11.7. Численное решение интегрального уравнения $\int_1^t \left(t^2+s^2+1\right)x\left(s\right)ds = t^2 - \frac{1}{t}$, полученное при использовании замкнутых квадратурных формул



Приложение 1

Основные приемы работы с пакетом MATLAB

MATLAB — интерактивный матрично-ориентированный пакет, предназначенный для выполнения научных и инженерных расчетов. Имя пакета MATLAB является аббревиатурой двух слов MATrix LABoratory (МАТричная ЛАБоратория). Цель данного раздела — познакомить читателя с необходимыми сведениями для начала работы с пакетом MATLAB. Мы рекомендуем совместить изучение данного раздела с одновременным выполнением приведенных здесь примеров.

Система MATLAB работает на таких платформах, как Sun/Apollo/VAXstation/HP workstations, VAX, MicroVAX, Gould, PC- и AT-совместимые, компьютерах с процессором 80386 и выше, Apple Macintosh и на ряде параллельных машин. В настоящем руководстве будут описаны основные свойства системы, которые одинаково применимы при использовании версий 5.0, 5.1, 5.2 и 6.0. MATLAB является собственным знаком фирмы MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760, (508)653-1415, Fax: (508)653-2997, E-mail: **info@mathworks.com**.

П1.1. Работа в командном окне

В данном разделе излагаются минимально необходимые сведения в объеме, достаточном для начала работы с пакетом MATLAB, в том числе: запуск пакета, получение справочной информации, работа с пакетом в командном режиме, редактирование и ввод кодов в командную строку, форматы выводов результатов вычислений и сохранение списка выполненных команд в ходе текущего сеанса работы.

П1.1.1. Вход в систему MATLAB

Войти в пакет MATLAB в большинстве операционных систем (например, в системе MS DOS) после входа в саму операционную среду можно, набрав в ответ на системный запрос команду **matlab**. При работе в операционных системах (ОС) Windows 9*x*, Windows 2000, Windows XP необходимо найти соответствующую иконку и кликнуть левой кнопкой мыши на ней. Выход из пакета осуществляется с помощью команды **quit**. В версиях пакета 5.0 и выше, работающих в ОС семейства Windows используется собственный встроенный редактор текста. В обоих случаях командное окно пакета MATLAB находится в одном окне, а редактор текста — в другом окне.

Примечание

Далее речь будет идти только о версии 5.*x* и выше и работе в ОС семейства Windows 95.

В командном окне MATLAB помимо собственно команд MATLAB можно использовать системные команды ОС MS DOS. Например, команда **dir** выводит на экран содержимое текущего каталога, команда **what** выводит только список m-файлов текущего каталога. Команда **cd** позволяет сменить текущий каталог, а команды **delete** <**имя_файла**> и **type** <**имя_файла**> стирают и печатают на экране содержимое файла соответственно.

Примечание

Здесь и далее m-файлом мы будем называть любой текстовой файл, содержащий набор команд пакета MATLAB, имеющий расширение m. Подробную информацию об этих файлах и их роли в системе MATLAB см. разд. П1.6.

П1.1.2. Интерактивный доступ к справочной информации и документации

Существуют следующие способы получить информацию о функциях пакета MATLAB в процессе работы:

- команда **help**;
- команда **lookfor**;
- меню **Help**;
- просмотр и вывод на печать страниц документации;
- обращение к WEB-серверу фирмы The MathWorks.

П1.1.2.1. Команда *help*

Отметим, что во время работы с системой можно пользоваться встроенной оперативной помощью, которая содержит сведения, необходимые для ус-

пешной работы с пакетом. После входа в систему команда **help** выведет список групп, в которые объединены функции:

```
>> help
```

Команда **help <имя_группы>** выведет на экран список функций, размещенных в этой группе с краткими пояснениями. Например, команда `help wavelet` выдаст список функций, реализующих методы Вэйвлет-анализа:

```
>> help wavelet
```

Основной и наиболее быстрый способ выяснить синтаксис и особенности применения m-функции — использовать команду **help <имя m-функции>**. Соответствующая информация появляется непосредственно в командном окне.

Например, команда `help zeros` выведет в командное окно следующую информацию на английском языке:

```
>> help zeros
    ZEROS  Zeros array.
    ZEROS(N) is an N-by-N matrix of zeros.
    ZEROS(M,N) or ZEROS([M,N]) is an M-by-N matrix of zeros.
    ZEROS(M,N,P,...) or ZEROS([M N P ...]) is an M-by-N-by-P-by-...
    array of zeros.
    ZEROS(SIZE(A)) is the same size as A and all zeros.
    See also ONES.
```

Примечание

Текст интерактивной справки использует верхний регистр для написания имен функций и переменных, чтобы выделить их из основной части текста. Однако при использовании функций их имена необходимо вводить с помощью символов нижнего регистра.

Команда **help** без аргументов выводит на экран список каталогов, которые имеются в системе с кратким описанием их содержимого. Повторный набор этой команды с именем каталога, например, `help elmat`, выведет список функций, предназначенных для создания и работы с матрицами специального вида. Ввод команды с именем определенной функции выдаст на экран описание этой функции.

Внимание

В качестве ответа на запрос о помощи выводятся все строки комментариев, которые написаны в начале каждой функции — как созданной разработчиками системы, так и собственными функциями пользователя.

П1.1.2.2. Команда *lookfor*

Эта команда позволяет выполнить поиск m-функции по ключевому слову; при этом анализируется первая строка комментария, и она же выводится на

экран, если в ней встретилось ключевое слово. Например, в системе MATLAB нет m-функции с именем `inverse`, и поэтому на команду

```
>> help inverse
```

ответом будет "inverse.m не найден":

```
inverse.m not found
```

Однако команда `lookfor inverse` покажет все совпадения, найденные в справочных файлах пакетов прикладных программ, подключенных к MATLAB.

Добавление к команде `lookfor` опции `-all` в виде **`lookfor <слово> -all`** расширяет область поиска — `<слово>` ищется в первом блоке комментариев, т. е. в блоке комментариев между заголовком функции и первым оператором.

П1.1.2.3. Меню *Help*

Это меню командного окна системы MATLAB позволяет активизировать следующие окна:

- **HelpWindow**
- **Help Tips**
- **Help Desk(HTML)**
- **Examples and Demos**
- **AboutMATLAB**
- **Subscribe (HTML)**

Окно справки **HelpWindow** позволяет получить в отдельном окне то же самое, что и команда `help`. Отличие состоит в том, что в этом окне можно погружаться внутрь раздела с помощью двойного щелчка мыши, а не повторно набирая команду `help` с новыми аргументами.

В пункте меню **Help Tips** приведена краткая справка по использованию помощи, т. е. описаны все пункты меню **Help**.

Вызов пункта меню **Help Desk** позволяет получить доступ к большому объему справочной информации и к документации по системе, размещаемой на жестких дисках, либо на CD в формате HTML. При использовании этого пункта меню желательно, чтобы в системе было установлено какой-либо Интернет-браузер (например, Internet Explorer или Netscape Navigator). Все операторы и функции пакета MATLAB описаны подробно и с большим числом примеров. Реализована поисковая система, позволяющая выполнять необходимые запросы.

В пункте меню **Examples and Demos** приведено большое число примеров использования пакета MATLAB для решения задач из разных областей.

Пункт меню **AboutMatlab** выводит на экран стандартную заставку MATLAB, а в пункте меню **Subscribe HTML** предлагается подписаться (при наличии лицензии на продукт) на доступ через сеть Интернет к услугам фирмы MathWorks.

Версии справочной документации доступны для просмотра и распечатки в формате PDF с помощью программы Adobe Acrobat Reader. Данный формат позволяет просматривать текст в виде печатной страницы с набором шрифтов, графики и изображений с полным ощущением чтения книги. Одновременно это и наилучший способ получения копий нужных страниц.

П1.1.3. Редактирование и повторный вызов командной строки

Командная строка MATLAB легко редактируется. Курсор можно перемещать с помощью стрелок ← → и удалять неправильно набранные символы с помощью клавиш <Backspace> или <Delete>. При работе на IBM PC совместимом компьютере доступны также и другие редактирующие клавиши: <Home>, <End> и <Delete>. При работе на других системах необходимо использовать команду **edit** для ознакомления со списком доступных команд. Удобным свойством системы является возможность использования клавиш-стрелок ↑ ↓ для доступа к буферу с ранее введенными командами. Это дает возможность осуществить повторный вызов ранее выполненной команды, ее редактирование и вторичное выполнение. Для небольших процедур это гораздо удобнее, чем писать и отлаживать m-файлы, что требует постоянного перехода из командного окна MATLAB в окно текстового редактора *(см. разд. П1.6)*. Например, если вы хотите сравнить графики функций sin(*mx*) и sin(*nx*) на интервале [0, 2] для различных значений *n* и *m*, то можете воспользоваться командной строкой

```
>> m=2;n=3;x=0:.01:2*pi;y=sin(m*x);z=sin(n*x); plot(x,y,x,z)
```

с последующим ее вызовом и редактированием. Проделайте все описанное ранее самостоятельно.

П1.1.4. Формат вывода

Поскольку все вычисления в MATLAB выполняются с двойной точностью, формат вывода может изменяться с помощью команд, представленных в табл. П.1.1.

Таблица П.1.1. *Форматы представления чисел в пакете MATLAB*

Название формата	Описание формата представления числа
format short	с фиксированной точкой и 4 знаками после точки (по умолчанию)

Таблица П.1.1. *(окончание)*

Название формата	Описание формата представления числа
format long	с фиксированной точкой и 14 знаками после точки
format short e	научная нотация с 4 десятичными знаками
format long e	научная нотация с 15 десятичными знаками
format short g	с фиксированной точкой и 3 знаками после точки
format long g	научная нотация с 13 десятичными знаками

После вызова одного из приведенных ранее форматов он сохраняется до вызова другого формата. Команда **`format compact`** подавляет большинство пустых строк, позволяя вывести на экран или страницу большее количество информации. Она не зависит от других команд формата.

П1.1.5. Копия протокола текущей сессии

Легче всего протокол текущей сессии получить с помощью команды `diary`. Вызов команды **`diary <имя_файла>`** приведет к тому, что все появившееся далее на экране (кроме графиков) будет записано в файл `<имя_файла>`. Если имя файла в команде будет опущено, то протокол сессии будет записан в файл с именем `diary`. Команда **`diary off`** приведет к выключению режима записи команд в протокол сессии, а команда **`diary on`** — к восстановлению режима записи. После завершения записи сессии вы можете отредактировать этот файл как обычный текстовый и при необходимости распечатать его или использовать для последующего написания m-файла.

П1.2. Создание матриц

MATLAB работает практически с одним видом объектов — с числовыми прямоугольными матрицами, элементами которых могут быть в общем случае комплексные числа. Все переменные представляют собой матрицы. В некоторых случаях матрицы 1×1 интерпретируются как скаляры, а матрицы с одной строкой или одним столбцом интерпретируются как вектора. В системе MATLAB матрицы могут быть созданы разными способами:

- введены явно с помощью списка элементов;
- сгенерированы встроенными операторами или функциями;
- созданы в m-файлах *(см. разд. П1.6)*;
- загружены из внешнего файла данных.

П1.2.1. Явное задание матриц

Например, в результате выполнения любого из приведенных далее операторов

```
>> A=[1 2 3; 4 5 6; 7 8 9] <Enter>
```

или

```
>> A=[1 2 3 <Shift>+<Enter>
4 5 6 <Shift>+<Enter>
7 8 9] <Enter>
```

MATLAB создает матрицу 3×3 и присваивает ее значение переменной:

```
A =
     1     2     3
     4     5     6
     7     8     9
```

Элементы внутри строки матрицы могут отделяться друг от друга не только пробелами, но и запятыми. При вводе чисел в экспоненциальной форме (например, `2.34e-9`) не следует использовать пробелы. Ввод больших матриц лучше выполнять с помощью m-файлов, в которых легко находить и исправлять ошибки *(см. разд. П1.6)*. В пакет MATLAB встроен ряд функций, позволяющих создавать функции специального вида, например, `rand`, `magic`, `zeros` и `ones` позволяют легко сгенерировать матрицы.

- Функция **`rand(n)`** создает матрицу размером n×n, каждый элемент которой — случайное число с равномерным законом распределения в диапазоне [0, 1].
- Функция **`rand(m,n)`** создает матрицу размера m×n, каждый элемент которой — случайное число с равномерным законом распределения в диапазоне [0, 1].
- Функция **`magic(n)`** создает матрицу размером n×n, которая является магическим квадратом (суммы элементов по строкам и столбцам равны).
- Функция **`zeros(m,n)`** создает нулевую матрицу размера m×n.
- Функция **`ones(m,n)`** создает матрицу размера m×n, каждый элемент которой равен единице.

Матрицы могут быть сгенерированы также с помощью цикла **`for`** *(см. разд. П1.5.1)*.

Ссылки на отдельные элементы матриц и векторов осуществляются с помощью индексов в круглых скобках. Например, `A(1,3)` означает элемент матрицы, стоящий на 1-й строке и 3-м столбце матрицы `A`, а `x(3)` означает 3-й элемент вектора `x`. В качестве индексов векторов и матриц могут использоваться только положительные числа. Ссылаться на элементы матрицы `A` можно, используя единственный индекс `A(k)`. В этом случае данная

матрица рассматривается как один длинный вектор-столбец, сформированный из столбцов исходной матрицы. Например, обратиться ко второму элементу второй строки матрицы `A` можно, указав `A(2,2)` или `A(5)`.

П1.2.2. Подматрицы и использование двоеточия

Для записи алгоритмов сложной обработки данных в компактной форме в системе MATLAB используются векторы и подматрицы. Использование нотации с двоеточием (которая используется и для генерации векторов и подматриц) и векторов вместо индексов является ключом к эффективной манипуляции этими объектами. Эффективное использование данных возможностей позволяет минимизировать число явных циклов, применение которых существенно замедляет работу MATLAB и делает написанную программу простой и легко читаемой. (Правда, овладение данной технологией требует от пользователя определенных усилий.) Например, выражение `1:5` фактически является вектор-строкой `[1 2 3 4 5]`. Отметим, элементы вектора могут быть не только целыми, но и действительными. Например, команда

```
>> x=0.2:0.2:1.2
```

создает вектор `x=[0.2, 0.4, 0.6, 0.8, 1.0, 1.2]`, а команда

```
>> 5:-1:1
```

создает в результате вектор `[5 4 3 2 1]`.

Для создания и вывода на экран таблицы синусов необходимо выполнить следующую последовательность команд:

```
>> x = [0.0:0.1:2.0]'; % транспонирование вектора
>> y = sin(x); % вычисление вектора, содержащего значения sin
>> [x y] % вывод таблицы значений
```

Отметим, что, т. к. `sin` является скалярной функцией (т. е. функцией, действующей поэлементно), результатом ее применения к вектору `x` будет вектор `y`.

Для доступа к подматрицам может быть использовано двоеточие. Например, `A(1:2,3)` является вектор-столбцом, состоящим из двух первых элементов третьего столбца матрицы `A`:

```
>> A(2:3,3)
ans =
     6
     9
```

Двоеточие само по себе означает всю строку или весь столбец, например:

```
>> A(:,3)
```

```
ans =

     3
     6
     9

>> A(3,:);
ans =
     7     8     9
```

В качестве индекса подматрицы может использоваться произвольный целый вектор:

```
>> A(1,[2 3])
ans =
     2     3
```

Описанные способы индексирования могут использоваться с обеих сторон знака присваивания. Например, после выполнения команды

```
>>  A(:,[2 4 5]) = B(:,1:3)
```

2, 4 и 5-й столбцы матрицы `A` будут заменены на первые три столбца матрицы `B`.

П1.2.3. Функции построения матриц

Пакет MATLAB, созданный для проведения матричных вычислений, работает практически с одним видом объектов — числовыми прямоугольными матрицами. (Отметим, что даже переменные, принимающие единственное значение (скаляр), во внутреннем представлении пакета являются массивом, состоящим из единственного элемента.) Существует возможность в процессе выполнения вычислений изменять размеры матрицы, при этом помимо записи нового элемента в массив MATLAB перестраивать всю структуру памяти, отведенную под данный массив. С каждым следующим присваиванием процедура перестройки памяти повторяется. Очевидно, что такой способ создания массива не является эффективным и существенно замедляет работу пакета. Существует простой способ, позволяющий приблизительно в 100 раз увеличить скорость вычислений. Для этого следует предварительно выделить память под конечный размер массива, т. е. создать матрицу заданного размера. Кроме того, можно ускорить вычисления, если изначально создавать матрицы требуемого вида (например, диагональные). Для решения данной задачи в MATLAB имеются следующие стандартные функции построения матриц:

- **`eye(m,n)`** — создание единичной матрицы размером m×n;
- **`zeros(m,n)`** — создание нулевой матрицы размером m×n;

- **ones(m,n)** — создание матрицы размером m×n, каждый элемент которой равен единице;
- **diag(x)** — создание матрицы, у которой на главной диагонали стоят элементы вектора x;
- **diag(A)** — создание матрицы, у которой на главной диагонали стоят диагональные элементы матрицы A;
- **triu(A)** — создание верхней треугольной матрицы, у которой элементы, расположенные на главной диагонали и выше нее, равны соответствующим элементам матрицы A;
- **tril(A)** — создание нижней треугольной матрицы, у которой элементы, расположенные на главной диагонали и ниже нее, равны соответствующим элементам матрицы A.

П1.3. Операции, выражения и переменные

MATLAB можно использовать, как обыкновенный калькулятор, который выполняет основные арифметические операции: + (сложение), – (вычитание), * (умножение), / (деление), ^ (возведение в степень). Для более сложных вычислений в пакете предусмотрены дополнительные операции, а также возможность создания и использования выражений, состоящих из нескольких выполняемых операторов.

П1.3.1. Правила записи операторов

Пакет MATLAB является интерпретирующим языком непосредственных вычислений. Это означает, что выражения, которые вы вводите, интерпретируются и вычисляются. Операторы пакета MATLAB обычно имеют форму:

```
>> имя_переменной = выражение
```

или просто

```
>> выражение
```

Выражение, как правило, формируется из операторов, функций и имен переменных. После выполнения выражения генерируется матрица, которая выводится на экран и присваивается соответствующей переменной для последующего использования. Если имя переменной в левой части и знак = отсутствуют, автоматически генерируется переменная ans (answer — ответ), которой присваивается результат вычислений.

Обычно оператор завершается клавишей <Enter>. Однако при необходимости оператор может быть продолжен на следующей строке. Для этого его необходимо завершить тремя или более точками, после которых следует

<Enter>. С другой стороны, в одной строке может быть несколько операторов, разделенных запятой или точкой с запятой.

Если последним символом в строке является точка с запятой, то вывод значений результата не производится, но присвоение значения выражения переменной выполняется. Это помогает подавить вывод ненужных промежуточных результатов. Важно помнить, что MATLAB различает строчные и прописные буквы в именах команд, функций и переменных.

Для получения списка всех переменных, расположенных в рабочем пространстве используется команда **who**. Переменная может быть удалена из рабочего пространства командой **clear <имя_переменной>**. Команда `clear` без аргументов очищает все непостоянные переменные, расположенные в рабочем пространстве. Примером постоянной переменной является переменная `eps` (epsilon), значение которой по умолчанию равно 10^{-6}. Данная переменная используется для оценки точности вычислений итеративных процессов.

Вывод на дисплей или вычисления могут быть прерваны на большинстве компьютеров, не покидая MATLAB, с помощью комбинации клавиш <Ctrl>+<C> (<Ctrl>+<Break> на PC).

П1.3.2. Матричные операции

В пакете MATLAB доступны следующие матричные операции: сложение (+); вычитание (-); умножение (*); возведение в степень (^); транспонирование (′); левое деление (\); правое деление (/).

Данные матричные операции применимы, конечно, и к скалярам матрицам 1×1. Если размерность матриц не соответствует используемой операции, то система генерирует сообщение об ошибке за исключением случаев, когда одним из операндов является скаляр, потому что в этом случае операция выполняется между скаляром и каждым элементом матрицы второго операнда.

Отметим, что операция возведения в степень применима только для квадратных матриц.

Если `A` является невырожденной квадратной матрицей, а `b` — вектор-столбец или вектор-строка, тогда вектор `x=A\b` является решением уравнения `A·x=b`, а `x=b/A` является решением уравнения `x·A=b` соответственно. Если `A` — квадратная матрица, то при левом делении для факторизации используется метод исключения Гаусса. Если матрица не квадратная, то для ее факторизации используется метод ортогонализации Хаусхольдера с ведущим столбцом, а приведенная матрица используется для решения переопределенной системы уравнений в смысле наименьших квадратов. Правое деление определяется в терминах левого деления по формуле `b/A=(A′\b′)′`.

П1.3.3. Операции с массивами

Матричные операции сложения и вычитания действуют поэлементно, а остальные приведенные ранее операции — нет, они являются матричными операциями. Следует отметить, что приведенные ранее операции *, . , \, / могут стать поэлементными, если перед ними поставить точку. Например, команды

```
>> [1,2,3,4].*[1,2,3,4]
```

или

```
>> [1,2,3,4].^2
```

дадут один и тот же результат:

```
ans=
[1,4,9,16]
```

П1.3.4. Сохранение данных из рабочей области

При выходе из системы MATLAB теряются значения всех переменных, находящихся в рабочей области. Для сохранения их значений на диске в нетекстовом формате в файле с именем matlab.mat перед выходом из пакета необходимо выполнить команду **save**. Для восстановления ранее сохраненных значений переменных используется команда **load**.

П1.4. Операторы *for*, *while*, *if*, *case* и операторы отношения

Операторы управления MATLAB (`for`, `while`, `if`, `case`) и операторы отношения (`and`, `or`, `xor`, `not`) при использовании в своей основной форме работают так же, как и в большинстве языков программирования.

П1.4.1. Цикл *for*

Например, для данного `n` оператор

```
>> x = [ ]; for i = 1:5,x=[x,i^2], end
x =
     1
x =
     1     4
x =
     1     4     9
x =
    1     4     9    16
x =
     1     4     9    16    25
```

создает определенный вектор размерности n, оператор

```
>> x = [ ]; for i = 5:-1:1,x=[x,i^2], end
x =
    25
x =
    25    16
x =
    25    16     9
x =
    25    16     9     4
x =
    25    16     9     4     1
```

создает вектор с теми же элементами, но размещенными в обратном порядке. Заметим, что матрица может быть пустой (например, в случае оператора `x=[]`). Последовательность операторов

```
>> m=3;
      n=3
      for i = 1:m
         for j = 1:n
            H(i, j) = 1/(i+j-1);
         end
      end
     H
H =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
```

создает и выводит на экран матрицу Гильберта размерности m×n. Точка с запятой, которая завершает внутренний оператор, предотвращает вывод на экран ненужных промежуточных результатов.

П1.4.2. Цикл *while*

В общем виде цикл **while** записывается таким образом:

```
while <условие>
   <операторы>
end
```

<операторы> будут повторяться до тех пор, пока <условие> будет оставаться истинным. Например, для заданного числа a приведенная далее последовательность операторов вычислит и выведет на дисплей наименьшее неотрицательное число n, такое что n^2<a:

```
>> n = 0;a=100;
```

```
>> while n^2 < a
         n = n + 1;
      end
      n
n=
    10
```

П1.4.3. Условный оператор *if*

В общем виде простой оператор **if** имеет следующий синтаксис:

```
if <условие>
  <операторы1>
else
  <операторы2>
end
```

<операторы1> выполняются, если <условие> истинно, <операторы2> — если <условие> ложно. В пакете MATLAB возможно также множественное ветвление, например:

```
if n < 0
   parity = 0;
elseif rem(n,2) == 0 % если условие n < 0 не выполняется
   parity = 2; % если число n четное
else
   parity = 1; % если число n нечетное
end
```

При использовании двухвариантного условного оператора часть, связанная с elseif не используется.

П1.4.4. Оператор переключения *case*

При необходимости построить конструкцию ветвления с более чем двумя логическими условиями удобнее использовать не вложенные операторы if, а оператор переключения switch...case. Этот оператор имеет следующую структуру:

```
switch <выражение>  % <выражение> – обязательно скаляр или строка
case <значение1>
   <операторы1>     % <операторы1> выполняются, если
                            % <выражение> = <значение1>
case <значение2>
   <операторы2>     % <операторы2> выполняются, если
                            % <выражение> = <значение2>
...
otherwise
   <операторы>      % <операторы> выполняются, если <выражение>
                          %  не совпало не с одним из значений
```

П1.4.5. Условия (операторы отношения)

В MATLAB используются следующие операторы отношения: меньше чем (<), больше чем (>), меньше или равно (<=), больше или равно (>=), равно (==), не равно (.=).

Отметим, что знак = используется в операторах присваивания, в то время как в операторах отношения используется знак ==. Операторы отношения (или, другими словами, логические переменные, которые они создают) могут объединяться с помощью следующих логических операторов: И (&), ИЛИ (|), НЕ (.).

Операторы отношения, примененные к скалярам, также возвращают скаляр, значение которого равняется 1 или 0 в зависимости от того, является ли результат *истиной* или *ложью*.

Операторы отношения, примененные к матрицам одного размера, возвращают матрицу того же размера, у которой в качестве элементов стоят 0 или 1, в зависимости от соотношения между соответствующими элементами исходных матриц.

Операторы while и if интерпретируют отношение между матрицами как истинное в том случае, если результирующая матрица не имеет нулевых элементов.

Например, если необходимо выполнить оператор <операторы>, когда матрицы A и B совпадают, следует использовать оператор соотношения:

```
if A == B <операторы> end
```

Если необходимо выполнить оператор в том случае, когда матрицы A и B не равны, можно использовать следующий оператор:

```
if A == B else <операторы> end
```

П1.5. Функции MATLAB

В системе MATLAB существует большое количество функций, созданных разработчиками системы, большинство из которых представлено в виде m-файлов, содержащих исходные тексты. Можно подойти к классификации данных функций различными способами, например, по областям их использования (тригонометрические, спецфункции, функции линейной алгебры и т. д.). Далее мы используем подход, основанный на виде аргумента функции: скаляр, вектор, матрица.

П1.5.1. Скалярные функции

Скалярные функции MATLAB действуют только на скаляры. Если аргументом данных функций является матрица, то они действуют поэлементно. К

таким функциям, например, относятся (в скобках указана запись функции в обычной математической нотации или возвращаемый ею результат): **sin** (sin(x)), **asin** (arcsin(x)), **exp** (ex), **abs** (абсолютное значение числа), **round** (округление числа до ближайшего целого), **cos** (cos(x)), **acos** (arcos(x)), **log** (натуральный логарифм), **sqrt** (извлечение квадратного корня из числа), **floor** (наименьшее целое число для данного действительного x, floor(2.8)=2), **tan** (tg(x)), **atan** (arctg(x)), **rem** (остаток от деления двух чисел), **sign** (знак числа), **ceil** (наибольшее целое для данного действительного числа, ceil(2.8)=3).

П1.5.2. Векторные функции

Аргументами векторных функций являются векторы (строки или столбцы). Если в качестве аргумента функции указана матрица размера m×n (`m>=2`), то данная функция действует по столбцам, т. е. результат — это вектор-столбец, каждый элемент которого является результатом действия этой функции на соответствующий столбец. Построчное действие такой функции (если необходимо) может быть достигнуто использованием операции транспонирования.

Примеры некоторых векторных функций: **max**, **sum**, **median**, **any**, **min**, **prod**, **mean**, **all**, **sort**, **std**.

В частности, максимальный элемент прямоугольной матрицы находится с помощью команды `max(max(A))`, а не с помощью `max(A)`, т. к. результат, возвращенный функцией `max(A)` — вектор, каждый компонент которого есть максимальный элемент соответствующего столбца.

П1.5.3. Матричные функции

Наибольшую мощь системе MATLAB дают матричные функции, самые употребляемые из которых приведены далее:

- **eig** — собственные значения и собственные вектора;
- **chol** — факторизация Холецкого;
- **svd** — сингулярная декомпозиция;
- **inv** — обратная матрица;
- **lu** — LU-факторизация;
- **qr** — QR-факторизация;
- **hess** — вычисление формы Хессенберга;
- **schur** — декомпозиция Шура;
- **rref** — приведение к треугольной форме методом Гаусса;
- **expm** — матричная экспонента;

- **sqrtm** — матричный квадратный корень;
- **poly** — характеристический полином;
- **det** — определитель;
- **size** — размерность;
- **norm** — норма вектора или матрицы;
- **cond** — число обусловленности;
- **rank** — ранг матрицы.

Функции MATLAB могут возвращать одновременно несколько переменных. Например, функция `y=eig(A)`, или просто `eig(A)`, генерирует вектор-столбец, содержащий собственные значения матрицы `A`, в то время как оператор `[U,D]=eig(A)` генерирует матрицу `U`, чьи столбцы являются собственными векторами `A`, а диагональная матрица `D` содержит на главной диагонали собственные значения этой матрицы.

П1.6. M-файлы

MATLAB может выполнять последовательность операторов, записанных в файл на диске. Такие файлы называются *m-файлами*, потому что имена этих файлов имеют вид **<имя>.m**. Большая часть работы в MATLAB состоит в создании, редактировании и выполнении таких m-файлов. Существует два типа m-файлов: файлы-программы (или сценарии) и файлы-функции.

П1.6.1. Файлы-программы

Файлы-программы состоят из последовательности обычных операторов пакета MATLAB. Если m-файл с таким сценарием имеет имя, например, inverse.m, то команда `inverse`, введенная в командной строке, вызовет выполнение последовательности операторов, находящейся в данном файле. Переменные в программе являются глобальными и изменят значения таких же переменных (если таковые есть) в рабочей области текущей сессии. Программы или сценарии часто используются для ввода данных в большие матрицы; в таких файлах легко исправить ошибки ввода. Если, например, файл на диске с именем data.m содержит строки

```
A=[
1 2 3 4
5 6 7 8
];
```

то выполнение команды `data` приведет к появлению в рабочей области матрицы `A`. Внутри m-файла можно ссылаться на другие m-файлы, в том числе и рекурсивно на самого себя.

П1.6.2. Файлы-функции

Файлы-функции фактически дают возможность расширять MATLAB, поскольку определенные пользователем новые функции, специфические для решения конкретных задач, имеют тот же статус, что и другие функции MATLAB. Переменные в функциях являются по умолчанию локальными, но, начиная с версии 4.0 и выше, разрешено объявлять глобальные переменные (`global`).

Рассмотрим пример функции, находящейся в файле randint.m (листинг П1.1)

Листинг П1.1. Файл randint.m

```
function z = randint(m,n,a,b)
% randint(m,n,a,b) - функция, возвращающая случайную матрицу с
% целыми элементами из диапазона [a, b]
% randint(m,n) - функция, возвращающая случайную матрицу с целыми
% элементами из диапазона [0, 9]
if nargin < 3, a = 0; b = 9; end
z = floor((b-a+1)*rand(m,n)) + a;
```

Совпадение имени функции и имени файла является в пакете MATLAB обязательным условием.

Первая строка функции содержит объявление выходных аргументов, имени функции и входных аргументов (списка формальных параметров). Без такой строки весь следующий файл является программой (или сценарием), но не функцией. Так, например, оператор `z=randint(4,5)` приведет к передаче чисел `4` и `5` переменным `m` и `n`, а выходной результат будет передан переменной `z`. Поскольку переменные в файле-функции локальные, их имена никак не влияют на имена и значения переменных в текущей рабочей области MATLAB.

Символ `%` указывает на то, что вся строка символов после него является комментарием и игнорируется при исполнении. Тем не менее, несколько первых строк-комментариев, которые являются кратким описанием данной функции, доступны при вводе оператора, т. е. являются той помощью, которая вызывается с помощью команды `help randint`.

Отметим, что здесь мы использовали функцию `nargin`, возвращающую длину списка формальных параметров функции. Использование функции `nargin` в данном примере позволяет установить значение отсутствующих входных аргументов, по умолчанию переменных `a` и `b`. В общем случае на-

личие такой функции позволяет использовать функции с переменным числом входных аргументов и в зависимости от их числа направлять вычисления по разным логическим веткам функции.

Функция может иметь множественные выходные аргументы (листинг П1.2).

Листинг П1.2. Файл stat.m

```
function [mean, stdev] = STAT(x)
% STAT – функция, возвращающая для вектора x, среднее значение и
% стандартное отклонение
% функция STAT возвращает для матрицы x две строки,
% первая из которых содержит средние значения, вторая – стандартные
% отклонения, соответствующих столбцов матрицы x
[m n] = size(x);
if m == 1
   m = n; % если входные данные - вектор
end
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);
```

Данный файл следует записать на диск под именем stat.m. Если ввести команду

```
>> [xm,xd]=stat(x)
```

то, переменным `xm`, `xd` будут присвоены среднее значение и стандартное отклонение элементов вектора `x`. Несмотря на наличие нескольких выходных аргументов, можно присвоить значение функции одной переменной. Например, `xm=stat(x)` (никаких скобок вокруг `xm` не требуется) присвоит `xm` среднее значение `x`.

Оператор `x.^2` является возведением в степень каждого элемента вектора `x`, функция `sum` является векторной функцией *(см. разд. П1.5.2.)*, функция `sqrt` является скалярной *(см. разд. П1.5.1.)*, деление в выражении `sum(x)/m` является матрично-скалярной операцией.

Другим примером является функция, позволяющая вычислить наибольший общий делитель двух чисел (листинг П1.3).

Листинг П1.3. Файл bisect.m

```
function [b, steps] = bisect(fun, x, tol)
% функция возвращающая решение уравнения fun(x)=0 методом
% половинного деления
% входные переменные:
% fun – строка, содержащая имя вещественной функции, зависящей от
% вещественной переменной
```

```
% x – начальное приближение, используемое для поиска корня
% tol – переменная, определяющая точность нахождения корня
% выходные переменные:
% b – значение корня
% step – матрица, содержащая значения корня и соответствующие значения
% функции на концах отрезка на каждом шаге вычислительного процесса
if nargin < 3, tol = eps; end % проверка длины списка формальных
                              % параметров, если значение переменной
                              % tol не указано, то точность корня
                              % определяется встроенной переменой eps
trace = (nargout == 2); % проверка длины списка выходных параметров,
                        % если список состоит из двух переменных, то trace=1
% выбор шага изменения переменной x
if x ~= 0
   dx = x/20;
else
   dx = 1/20;
end;
a = x - dx; % левый конец отрезка
fa = feval(fun,a); % значение функции в левом конце отрезка
b = x + dx; % правый конец отрезка
fb = feval(fun,b); % значение функции в правом конце отрезка
% поиск отрезка, на котором функция меняет знак
while (fa > 0) == (fb > 0)
   dx = 2.0*dx;
   a = x - dx;
   fa = feval(fun,a);
   if (fa > 0) ~= (fb > 0)
      break
   end;
   b = x + dx;
   fb = feval(fun,b);
end;
if trace
   steps = [a fa; b fb];
end;
% основной цикл
while abs(b - a) > 2.0*tol*max(abs(b),1.0)
    c = a + 0.5*(b - a);
    fc = feval(fun,c);
    if trace
       steps = [steps; [c fc]];
    end;
    if (fb > 0) == (fc > 0)
```

```
        b = c; fb = fc;
    else
        a = c; fa = fc;
    end;
end;
```

Здесь для вычисления значения функции, имя которой передается в функцию `bisect` через строковую переменную `fun`, используется функция `feval`, аргументами которой являются имя функции и координата точки, в которой вычисляется значение данной функции.

Отметим, что некоторые функции MATLAB являются встроенными, в то время как другие поставляются в виде m-файлов. Текст реально имеющихся m-файлов (пакета MATLAB или ваших собственных) можно просмотреть с помощью команды **`type <имя_функции>`**.

П1.6.3. Текстовые строки, сообщения об ошибках

Текстовые строки вводятся в MATLAB в виде текста в одинарных кавычках. Например, оператор `s='This is a test'` присваивает данный текст переменной `s`.

Вывод текстовой строки осуществляется с помощью оператора **`disp`**. Например, оператор `disp(s)` выведет на экран сообщение:

```
This is a test
```

Сообщения об ошибках лучше выводить с помощью функции **`error`**, так как после обращения к данной функции выполнение m-файла будет прекращено. Например, если в процессе выполнения m-файла будет выполнен оператор `error('Sorry, the matrix must be symmetric')`, то после вывода сообщения на экран выполнение m-файла будет прекращено.

В m-файле может быть запрос на интерактивный ввод данных, организованный с помощью оператора **`input`**. Когда вводится оператор `iter=input('Введите число итераций: ')`, на экран выводится запрос на ввод, и выполнение программы приостанавливается до того момента, пока пользователь не введет с клавиатуры требуемые входные данные. После нажатия клавиши <Enter> данные присваиваются переменной `iter`, и выполнение программы будет продолжено.

П.1.6.4. Работа с m-файлами

Во время работы в MATLAB часто возникает необходимость создавать или редактировать m-файлы, после чего возвращаться в командное окно пакета MATLAB для отладки и/или вычислений. Начиная с версии 5.0 (в ОС Windows 95), в пакете имеется специальный редактор/отладчик, в котором можно исправлять текст m-файлов и выполнять пошаговую отладку про-

граммы. Для работы с редактором/отладчиком необходимые m-файлы должны быть доступны. Для этого либо текущий каталог должен быть каталогом с вашими файлами, либо необходимо проложить туда путь (в смысле MS DOS). Это можно сделать либо с помощью команд MS DOS непосредственно из командного окна (команда `cd`), либо с помощью пункта меню **File | Set Path**, который позволяет установить путь к соответствующим папкам.

П1.6.5. Список путей доступа

Для поиска m-файлов система MATLAB использует механизм путей доступа, поскольку m-файлы записываются в каталоги или папки файловой системы. Например, при поиске файла с именем test пакет MATLAB выполняет следующие действия:

1. Просматривает, не является ли test именем переменной.
2. Просматривает, не является ли test встроенной функцией.
3. Ищет в текущем каталоге m-файл с именем test.m.
4. Ищет m-файл с именем test.m во всех каталогах *списка путей доступа*.

Реально применяемые правила поиска являются более сложными из-за ограничений, которые связаны с использованием подфункций[3], личных (private) функций и объектно-ориентированных механизмов. Однако приведенный ранее упрощенный порядок поиска точно отражает механизм поиска m-файлов, с которыми обычно работает пользователь.

П1.6.5.1. Работа со списком путей доступа

В процессе сеанса работы можно вывести на терминал или внести изменения в список путей доступа, используя следующие функции:

- **`path`** — выводит на экран список путей доступа;
- **`path (s)`** — заменяет существующий список списком **`s`**;
- **`addpath /home/lib`** и **`path(path,'/home/lib')`** — добавляют новый каталог текущего подкаталога в список путей доступа;
- **`rmpath /home/lib`** — удаляет путь **`/home/lib`** из списка.

Список путей доступа, используемый по умолчанию, определен в файле pathdef.m, который размещен в каталоге local; этот файл выполняется при каждом запуске пакета MATLAB.

---

[3] Внутри функций допускается определение других функций, при этом следует иметь в виду, что доступ к таким функциям возможен только из функций, внутри которых они определены.

Кроме работы из командной строки, существует средство просмотра путей доступа Path Browser (см. далее), которое поддерживает удобный графический интерфейс для просмотра и изменения списка путей.

П1.6.5.2. Текущий каталог

Пакет MATLAB использует понятие *текущего каталога* при работе с m- и mat-файлами во время сеанса работы.

Начальный текущий каталог определен в файле запуска, который ассоциирован с ярлыком запуска пакета MATLAB, расположенном на рабочем столе.

Для вывода текущего каталога на экран терминала предназначена команда `cd`. Для изменения текущего каталога следует использовать команду `cd <новый путь доступа>`.

П1.6.5.3. Средство просмотра и редактирования путей доступа Path Browser

Как было указано ранее, при работе в системе Windows 95 имеется специальное средство для просмотра и изменения путей доступа Path Browser (рис. П1.1). Показанное далее окно открывается либо из меню **File | Set Path** командного окна, либо с помощью кнопки на панели инструментов.

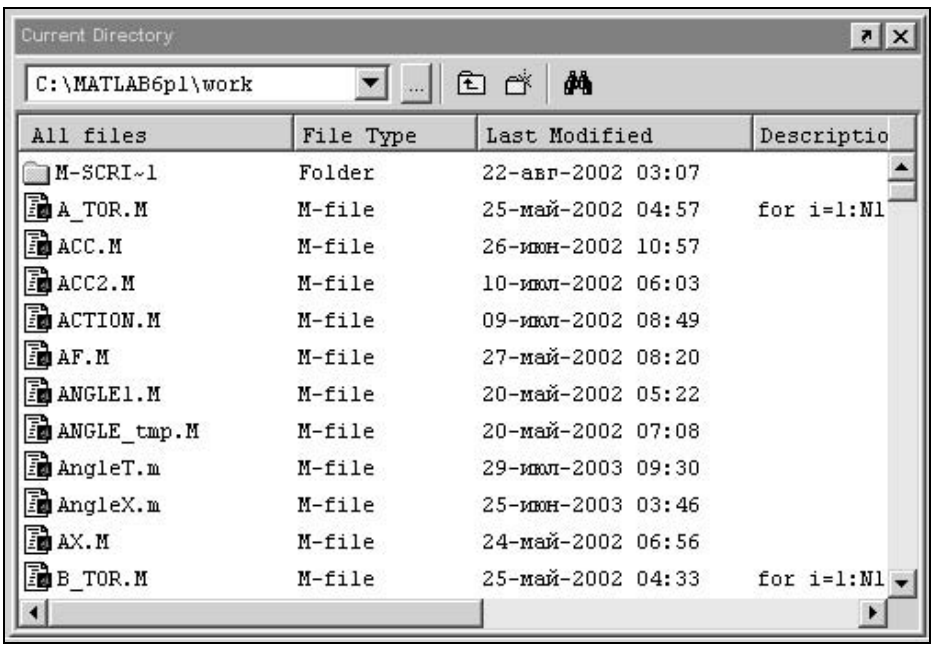


Рис. П.1.1. Окно Path Browse

После дополнения списка путей доступа необходимо сохранить новый путь с помощью пункта меню **File | Save Path**, в противном случае установленный путь будет известен системе только на время одного сеанса работы.

П1.6.5.4. Использование редактора/отладчика

Редактор/отладчик предоставляет как средства редактирования текста m-файла, так и средства пошаговой его отладки. Один из способов вызова редактора — вызов из командной строки MATLAB с помощью команды **edit**. Например, команда `edit test` откроет встроенный редактор для редактирования файла test.m, если в меню **File** в диалоговом окне **Preferences** вами не установлен другой редактор.

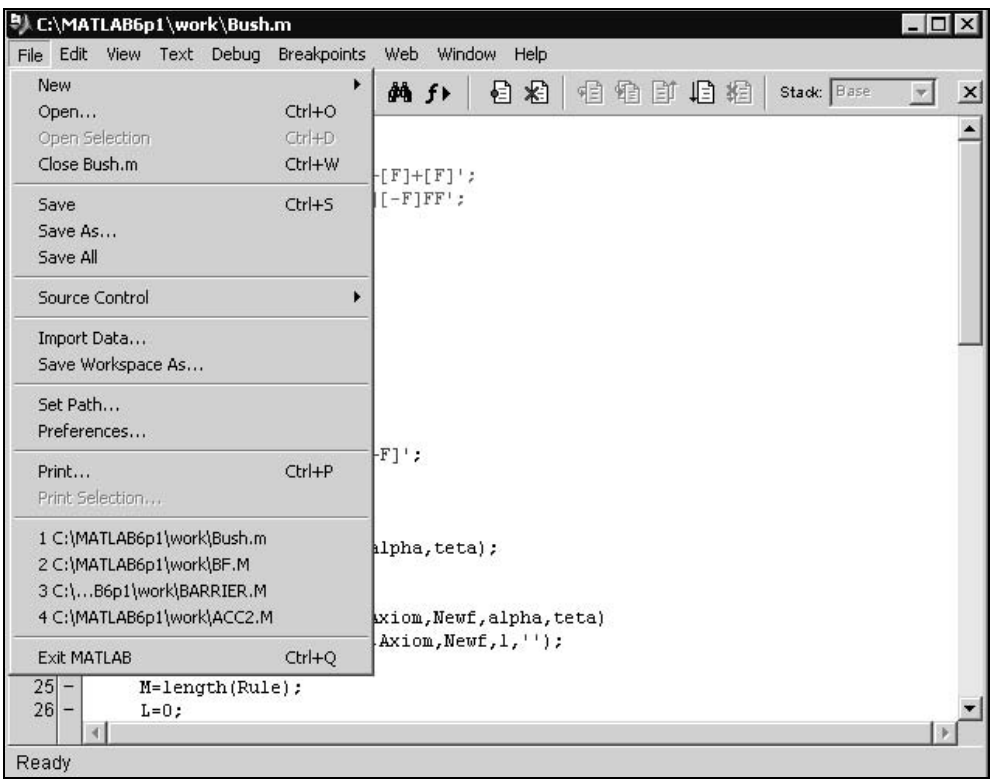


Рис. П1.2. Общий вид редактора/отладчика

Можно открыть редактор и другим способом — с помощью меню **File | New** или кнопки **New File** на панели инструментов (см. далее). Для открытия существующего m-файла выберите пункт **File | Open** или щелкните на кнопке **Open File**.

После вызова редактор/отладчик будет иметь вид, показанный на рис. П1.2.

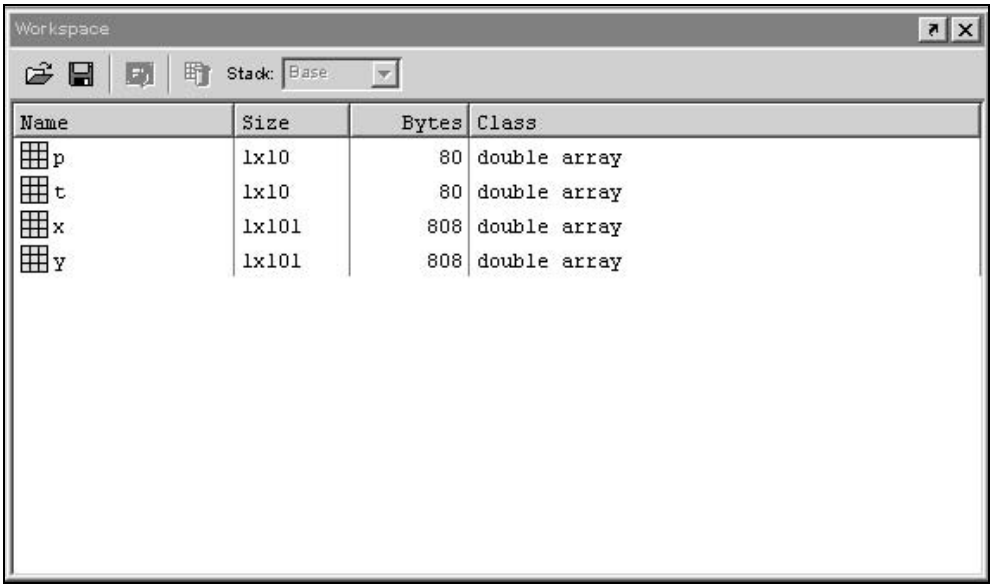


Рис. П1.3. Окно просмотра и редактирования используемых переменных

Редактор, используемый в системе, имеет синтаксическую раскраску, т. е. слово или символ по мере ввода приобретают тот цвет, который соответствует их типу.

Редактор различает такие типы вводимых слов:

- комментарии;
- ключевые слова;
- незаконченные строки;
- законченные строки;
- другой текст.

С помощью пункта меню **Tools | Fonts** можно настроить такие важные параметры, как используемый шрифт. Это особенно важно для работы с русским текстом, поскольку не все шрифты правильно его воспроизводят. В остальном данный редактор не отличается от обычного многооконного текстового редактора — в нем работают все необходимые клавиши (<Del>, <Bspace>, <Home> и т. д.). При редактировании файлов вы можете непосредственно перейти к требуемой строке при помощи пункта меню **Edit | GoTo Line** и указать ее номер в появившемся окне. После редактирования файла и повторного его запуска из командного окна желательно предварительно сохранить новый вариант файла. Но можно запускать ре-

дактируемый файл на счет, не выходя из редактора (т. е. не переходя в командное окно), с помощью пункта меню **Debug | Save and Run**. При этом предварительное сохранение текста исправлений не требуется.

Одна из важных особенностей данного редактора состоит в том, что после проведения вычислений можно в редакторе просмотреть значения переменных, которые они имеют в текущий момент в рабочей области. Для этого достаточно установить курсор мыши на этой переменной, и появится прямоугольник с желтым фоном, на котором выводится текущее значение переменной. Если переменная представляет собой большую матрицу, то таким образом увидеть целиком ее не удастся. Для просмотра (и возможного исправления при отладке) всех значений матрицы необходимо перейти в окно **Workspace** и выполнить двойной клик мышью по имени строки, содержащей имя выбранной переменной.

П.1.6.6. Отладка m-файлов

Отладка программного кода — это процесс, в ходе которого могут быть выявлены ошибки двух видов.

- Синтаксические, связанные с неточностью записи имен m-функций и/или арифметических выражений. Пакет MATLAB обнаруживает большинство синтаксических ошибок, сопровождая их сообщением об ошибке с указанием номера строки соответствующего m-файла.
- Ошибки периода выполнения, связанные в первую очередь с ошибками алгоритма и приводящие к непредвиденным результатам.

Как показывает опыт работы достаточно легко можно исправить синтаксические ошибки, которые сопровождаются сообщениями о причинах их возникновения. Ошибки времени выполнения выявить сложнее, потому что локальная рабочая область m-функции оказывается потерянной, если ошибка приводит к возврату в рабочую область системы пакета MATLAB. Для выявления причины данной ошибки, можно использовать один из следующих приемов:

- реализовать вывод результатов промежуточных вычислений на дисплей, удалив в соответствующих операторах точки с запятой, которые подавляют вывод на экран промежуточных результатов;
- добавить в m-файл команды `keyboard`, которые останавливают выполнение m-файла и разрешают проверить и изменить переменные рабочей области вызываемой m-функции. (В этом режиме в командном окне появляется специальное приглашение **K>>**. Для просмотра значений доступных переменных можно использовать окно **Workspace**, или ввести имя соответствующей переменной в командной строке и нажать клавишу <Enter>. Возврат к выполнению функции реализуется командой `return`);

- закомментировать заголовок функции и выполнить m-файл как сценарий. Это позволяет проследить результаты промежуточных вычислений в рабочей области системы;
- использовать отладчик системы MATLAB.

Отладчик полезен для исправления ошибок во время выполнения программы, так как он дает возможность отслеживать рабочие области функции и проверять или изменять значения соответствующих переменных. Отладчик позволяет устанавливать и удалять контрольные точки, то есть специальным образом помеченные строки m-файла, в которых выполнение останавливается. Это дает возможность изменять содержимое рабочей области, просматривать стек вызова m-функций и выполнять m-файл построчно. Отладчик может функционировать как в режиме командной строки, так и в режиме графического интерфейса пользователя. Далее мы рассмотрим отладку только в режиме графического интерфейса пользователя, так как он наиболее прост и нагляден.

Рассмотрим возможности отладки, которые нам предоставляет редактор/отладчик (**Editor/Debuger**).

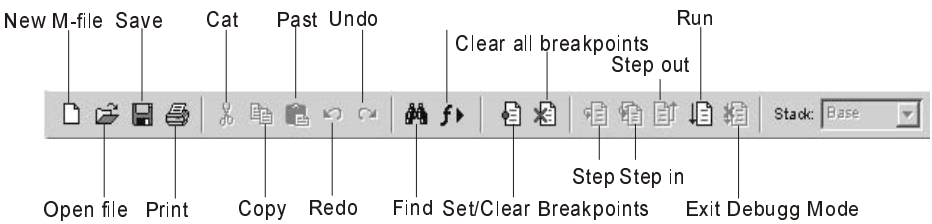
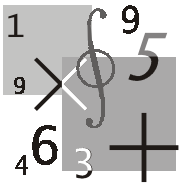


Рис. П1.4. Кнопки основной панели редактора/отладчика

Для его запуска используется команда `edit <имя_файла>` или пункт меню **File | Open.** Можно открыть окно редактора/отладчика и с помощью пункта меню **File | New | M-file**. При таком варианте имя отлаживаемого файла открывается уже из меню самого редактора/отладчика. Общий вид окна редактора показан на рис. П1.2, а кнопки панели инструментов, которые удобнее всего использовать при отладке, показаны на рис. П1.4.

Способ использования этих кнопок понятен из их названия. Часть кнопок представляют собой обычные редактирующие кнопки, используемые при сохранении, копировании, печати и поиске файлов в программных продуктах, работающих под управлением ОС Windows. Другая часть связана непосредственно с отладкой. Это кнопки установки и очистки точек остановки, кнопки пошагового перемещения по программе (**Step**) с заходом в подпрограммы (**Step in**) и без захода в подпрограммы (**Step out**), кнопка начала вычислений (**Run**) и кнопка остановки отладки (**Exit Debugg Mode**).



Приложение 2

Точные методы решения дифференциальных уравнений в частных производных

В данном разделе кратко рассмотрены общие принципы точных методов решения УЧП: метода разделения переменных, метода интегральных преобразований, метода преобразования зависимых переменных, метода преобразования переменных.

Отметим, что первый из рассмотренных методов – метод разделения переменных — имеет наглядную физическую интерпретацию: начальное условие, рассматриваемое как некоторое внешнее воздействие на систему, описываемую заданным УЧП, раскладывается на сумму простейших компонентов. Затем находятся отклики системы на каждый компонент внешнего воздействия. Результирующий отклик системы находится суммированием откликов системы на каждый из компонентов.

П2.1. Метод разделения переменных

Проиллюстрируем основную идею метода разделения переменных на примере нахождения решения краевой задачи уравнения теплопроводности:

$$u_t = \alpha^2 u_{xx},\ 0 \le x \le 1,\ 0 \le t \le \infty\ , \qquad (П2.1)$$

с граничными условиями:

$$\begin{cases} u(t,0) = 0 \\ u(1,t) = 1 \end{cases},\ 0 \le t \le \infty \qquad (П2.2)$$

и начальными условиями:

$$u(x,0)=\varphi(x),\ 0\le x\le 1. \qquad (П2.3)$$

Физическая интерпретация данной задачи: есть некоторый стержень, длина которого равна единице, концы стрежня имеют постоянную температуру, равную нулю. Задано распределение температуры по длине стержня в момент времени $t=0$, описываемое функцией $\varphi(x)$. Требуется найти распределение температуры $u(x,t)$ в последующие моменты времени.

Для рассматриваемого уравнения с частными производными метод разделения переменных состоит в поиске решения УЧП в виде

$$u(x,t)=X(x)T(t), \qquad (П2.4)$$

где $X(x)$ — функция, зависящая только от переменной *x*, а $T(t)$ — функция, зависящая только от переменной *t*.

Общая идея заключается в нахождении бесконечного числа решений УЧП $u_n(x,t)=X_n(x)T_n(t)$, которые удовлетворяют заданным граничным условиям. Функции $u_n(x,t)$ называются фундаментальными решениями задачи. Общее решение задачи $u(x,t)$ находится в виде линейной комбинации фундаментальных решений $u_n(x,t)=X_n(x)T_n(t)$:

$$u(t,x)=\sum_{n=1}^{\infty}A_nX_n(x)T_n(t), \qquad (П2.5)$$

такое что $u(x,t)$ удовлетворяет начальным и граничным условиям задачи.

Таким образом, нахождение решения УЧП методом разделения переменных осуществляется в несколько шагов, суть которых описывается далее.

Шаг 1. Нахождение элементарных решений УЧП.

Подставляя выражение (П2.4) в УЧП (П2.1), получаем:

$$X(x)T'(t)=\alpha^2X''(x)T(t). \qquad (П2.6)$$

Далее разделим обе части (П2.6) на $\alpha^2X(x)T(t)$:

$$\frac{T'(t)}{\alpha^2T(t)}=\frac{X''(x)}{X(x)}. \qquad (П2.7)$$

В (П2.7) левая часть зависит только от переменной *t*, а правая — только от переменной *x*, поэтому говорят, что выполнено разделение переменных. Так как *x* и *t* являются независимыми переменным, то обе части уравнения (П2.7) должны равняться некоторой постоянной величине *k*, называемой константой разделения переменных:

$$\frac{T'(t)}{\alpha^2 T(t)} = \frac{X''(x)}{X(x)} = k\,,$$

или

$$T' - k\alpha^2 T = 0\,, \qquad (П2.8)$$

$$X'' - kX = 0\,. \qquad (П2.9)$$

Таким образом, решение УЧП сводится к нахождению решений двух обыкновенных дифференциальных уравнений (П2.8), (П2.9).

Отметим, что константа разделения переменных должна быть отрицательной, так как в противном случае решение уравнения (П2.6) будет бесконечно возрастающей функцией $T(t) = T(0)e^{kt}$. Учитывая отмеченное обстоятельство, оказывается удобным ввести следующее обозначение:

$$k = -\lambda^2\,,\ \lambda \neq 0\,. \qquad (П2.10)$$

Используя (П2.10), запишем уравнения (П2.8), (П2.9) в виде:

$$T' + \lambda^2\alpha^2 T = 0\,, \qquad (П2.11)$$

$$X'' + \lambda^2 X = 0\,. \qquad (П2.10)$$

Общие решения уравнений (П2.11) и (П2.12) хорошо известны:

$$T(t) = ae^{-\lambda^2\alpha^2 t}\,, \qquad (П2.13)$$

$$X(x) = b\sin(\lambda x) + c\cos(\lambda x)\,, \qquad (П2.14)$$

где a, b, c — произвольные постоянные.

Подставляя (П2.13), (П2.14) в (П2.4), получаем общее выражение для функций, являющихся решением УЧП (П2.1):

$$u(x,t) = e^{-\lambda^2\alpha^2 t}\left[A\sin(\lambda x) + B\cos(\lambda x)\right], \qquad (П2.15)$$

где $A = ab$, $B = ac$.

Шаг 2. Нахождение решений удовлетворяющих граничным условиям.

После нахождения общего вида функций, являющихся решением УЧП (П2.1), необходимо выделить подмножество решений, удовлетворяющее начальным и граничным условиям (П2.2), П(2.3). Для этого подставим (П2.13) в (П2.2), (П2.3). В результате получаем:

$$u(0,t) = Be^{-\lambda^2\alpha^2 t} = 0 \Rightarrow B \equiv 0\,,$$

$$u(1,t) = Ae^{-\lambda^2\alpha^2 t}\sin\lambda = 0 \Rightarrow \sin\lambda \equiv 0\,.$$

Таким образом, второе граничное условие накладывает ограничения на возможные значения константы разделения λ, являющейся решением уравне-

ния $\sin\lambda = 0$. То есть для того, чтобы удовлетворить условию $u(1,t) = 0$, необходимо потребовать выполнения соотношений:

$$\lambda = \pm\pi, \pm 2\pi, \ldots$$

или

$$\lambda_n = \pm n\pi\,,\ n = 1,2,\ \ldots \qquad (\text{П}2.16)$$

Подставляя (П2.16) в (П2.15) находим выражение для бесконечного набора функций, удовлетворяющих УЧП и граничным условиям:

$$u_n(x,t) = A_n e^{-(n\pi\alpha)^2 t}\sin(n\pi x)\,,\quad n = 1,2,\ \ldots \qquad (2.17)$$

Общее решение УЧП (П2.1) имеет вид:

$$u(x,t) = \sum_{n=1}^{\infty} A_n e^{-(n\pi\alpha)^2 t}\sin(n\pi x)\,. \qquad (\text{П}2.18)$$

Шаг 3. Нахождение решения, удовлетворяющего граничным и начальным условиям.

Последний шаг в нахождении решения УЧП состоит в нахождении коэффициентов A_n, обеспечивающих соответствие функции $u(x,t)$ начальному условию:

$$u(x,0) = \varphi(x)\,. \qquad (\text{П}2.19)$$

Положив в (П2.18) $t = 0$ и подставив в (П2.19), получаем:

$$\varphi(x) = \sum_{n=1}^{\infty} A_n \sin(n\pi x)\,. \qquad (\text{П}2.20)$$

Для нахождения коэффициентов A_n умножим обе части (П2.20) на $\sin(m\pi x)$ (m — произвольное целое число) и проинтегрируем полученные выражения на интервале [0, 1]:

$$\int_0^1 \varphi(x)\sin(m\pi x)\,dx = A_m \int_0^1 \sin(m\pi x)\sin(n\pi x)\,dx\,. \qquad (\text{П}2.21)$$

Используя свойство ортогональности функций $\sin(m\pi x)$, $\sin(n\pi x)$:

$$\int_0^1 \sin(m\pi x)\sin(n\pi x)\,dx = \begin{cases} 0, & m \neq n \\ \dfrac{1}{2}, & m = n \end{cases},$$

получаем

$$\int_0^1 \varphi(x)\sin(n\pi x)\,dx = \frac{A_n}{2}\,. \qquad (\text{П}2.22)$$

Решив (П2.22) относительно A_n, найдем

$$A_n = 2\int_0^1 \varphi(x)\sin(\pi n x)\,dx, \qquad (П2.23)$$

т. е. A_n — коэффициенты разложения функции A_n в ряд Фурье по синусам.

Таким образом, искомое решение УЧП (П2.1) дается выражением (П2.18), в котором коэффициенты A_n вычисляются в соответствии с (П2.23).

П2.2. Метод интегральных преобразований

Основная идея метода интегральных преобразований схожа с идей метода разделения переменных. Внешнее воздействие на систему раскладывается в частотной области на элементарные составляющие. Далее находится реакция системы на каждую составляющую внешнего воздействия, а искомое решение УЧП находится суммированием найденных реакций системы (переход из частотной области во временную область).

П2.2.1. Общие сведения об интегральных преобразованиях

Интегральным преобразованием называют преобразование, ставящее в соответствие каждой функции $f(t)$ новую функцию $F(s)$ по формуле

$$F(s) = \int_A^B K(s,t)\,f(t)dt, \qquad (П2.24)$$

где $K(s,t)$ — ядро преобразования, а пределы интегрирования A, B зависят от конкретного вида функции $K(s,t)$.

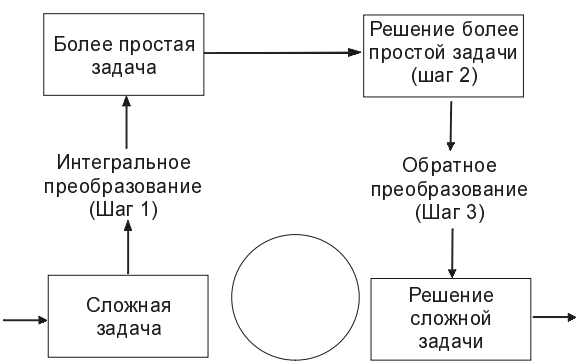


Рис. П2.1. Общая схема применения интегральных преобразований для решения УЧП

Применение интегральных преобразований для решения УЧП основано на возможности исключить в уравнении частную производную по одной из независимых переменных. Общая схема решения УЧП с использованием интегральных преобразований можно изобразить в виде схемы, представленной на рис. П2.1.

Прямое и обратное преобразования принято называть парой преобразований. Наиболее часто используемые на практике при решении УЧП пары интегральных преобразований представлены в табл. П2.1.

Таблица П2.1. *Некоторые пары интегральных преобразований*

Название преобразования	**Формула преобразования**
Синус-преобразование Фурье	$F_s[f] = F(\omega) = \frac{2}{\pi}\int_0^{\infty} f(t)\sin(\omega t)dt$
Обратное синус-преобразование Фурье	$F_s^{-1}[F] = f(t) = \frac{2}{\pi}\int_0^{\infty} F(\omega)\sin(\omega t)d\omega$
Косинус-преобразование Фурье	$F_c[f] = F(\omega) = \frac{2}{\pi}\int_0^{\infty} f(t)\cos(\omega t)dt$
Обратное косинус-преобразование Фурье	$F_c^{-1}[f] = f(t) = \frac{2}{\pi}\int_0^{\infty} F(\omega)\cos(\omega t)d\omega$
Преобразование Фурье	$F_s[f](\omega) = S_n(\omega) = \frac{2}{L}\int_0^{\infty} f(x)\sin\left(\frac{n\pi x}{L}\right)dx$
Обратное преобразование Фурье	$F_s^{-1}[f](\omega) = f(x) = \sum_{n=1}^{\infty} S_n(\omega)\sin\left(\frac{n\pi x}{L}\right)$
Конечное синус-преобразование Фурье	$F_s[f](\omega) = S_n(\omega) = \frac{2}{L}\int_0^{L} f(x)\sin\left(\frac{n\pi x}{L}\right)dx$
Обратное конечное синус-преобразование Фурье	$F_s^{-1}[f](\omega) = f(x) = \sum_{n=1}^{\infty} S_n(\omega)\sin\left(\frac{n\pi x}{L}\right)$
Конечное косинус-преобразование Фурье	$F_s[f](\omega) = C_n(\omega) = \frac{2}{L}\int_0^{L} f(x)\cos\left(\frac{n\pi x}{L}\right)dx$
Обратное конечное косинус-преобразование Фурье	$F_s^{-1}[f](\omega) = f(x) = \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n(\omega)\cos\left(\frac{n\pi x}{L}\right)$

Таблица П2.1. *(окончание)*

Название преобразования	Формула преобразования
Преобразование Лапласа	$L[f] = F(s) = \int_0^\infty f(t)e^{-st}\,dt$
Обратное преобразование Лапласа	$L^{-1}[F] = f(t) = \frac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} F(s)e^{st}\,ds$

П2.2.2. Решение краевой задачи УЧП с использованием синус-преобразования

Рассмотрим решение следующей краевой задачи УЧП

$$u_t = \alpha^2 u_{xx},\ 0 \le x \le \infty,\ 0 \le t \le \infty, \qquad (П2.25)$$

с начальными условиями

$$u(x,0) = 0,\ 0 \le x \le \infty \qquad (П2.26)$$

и граничными условиями

$$u(0,t) = A,\ 0 \le t \le \infty \qquad (П2.27)$$

методом синус-преобразования Фурье.

Шаг 1. Выполнение синус-преобразования уравнения (П2.25), начальных (П2.26) и граничных условий (П2.27).

Применим синус-преобразование к обеим частям уравнения (П2.25):

$$F_s[u_t] = \alpha^2 F_s[u_{xx}]. \qquad (П2.28)$$

Рассмотрим левую часть уравнения (П2.28). Так как синус-преобразование выполняется по переменной x, а интеграл $\int_0^\infty u_t(x,t)\sin(\omega x)dx$ равномерно сходящийся, частную производную по переменной t можно вынести за знак интеграла:

$$F_s[u_t] = \frac{2}{\pi}\int_0^\infty u_t(x,t)\sin(\omega x)\,dx = \frac{\partial}{\partial t}\left[\frac{2}{\pi}\int_0^\infty u(x,t)\sin(\omega x)\,dx\right]$$

$$= \frac{d}{dt}F_s[u] = \frac{d}{dt}U(t).$$

Отметим, что исходная функция u зависит от переменных x и t, а ее образ

$$F_s[u] = U(\omega, t)$$

зависит от переменных ω и t.

Так как в новой задаче переменная ω играет роль параметра, можно считать, что образы функций зависят только от одной переменной t:

$$F_s[u] = U(t).$$

Рассмотрим правую часть уравнения (П2.28). Дважды проинтегрировав по частям, получим:

$$F_s[u_{xx}] = \frac{2}{\pi}\int_0^\infty u_{xx}(x,t)\sin(\omega x)dx =$$

$$= \frac{2}{\pi}\left(u_x(x,t)\sin(\omega x)\Big|_0^\infty - \omega\int_0^\infty u_x(x,t)\cos(\omega x)\,dx \right) =$$

$$= -\frac{2}{\pi}\omega\int_0^\infty u_x(x,t)\cos(\omega x)dx = -\frac{2}{\pi}\omega\left(u(x,t)\cos(\omega x)\Big|_0^\infty - \omega\int_0^\infty u(x,t)\sin(\omega x)\,dx \right) =$$

$$= \frac{2}{\pi}\omega u(0,t) - \omega^2 F_s[u] = \frac{2}{\pi}\omega u(0,t) - \omega^2 U(t).$$

Таким образом, в результате синус-преобразования уравнение (П2.25) принимает следующий вид:

$$\frac{dU}{dt} = \alpha^2\left[-\omega^2 U(t) + \frac{2}{\pi}\omega u(0,t) \right]. \qquad (П2.29)$$

Подставляя в (П2.29) граничные условия (П2.27), получим обыкновенное дифференциальное уравнение

$$\frac{dU}{dt} = \alpha^2\left[-\omega^2 U(t) + \frac{2}{\pi}\omega A \right]. \qquad (П2.30)$$

Для нахождения начальных условий дифференциального уравнения (П2.30) применим синус преобразование к начальному условию (П2.26):

$$F_s[u(x,0)] = U(0) = 0. \qquad (П2.31)$$

Таким образом, вместо исходной смешанной задачи мы получили задачу Коши для обыкновенного дифференциального уравнения. На этом первый шаг решения УЧП методом синус-преобразования завершается.

Шаг 2. Решение задачи Коши дифференциального уравнения.

Запишем уравнение (П2.30) в следующем виде:

$$\frac{dU}{dt} + \omega^2\alpha^2 U = \frac{2A\omega\alpha^2}{\pi}. \qquad (П2.32)$$

Уравнение (П2.32) является неоднородным дифференциальным уравнением первого порядка. Его решение есть сумма решения однородного дифференциального уравнения

$$\frac{dU}{dt} + \omega^2 \alpha^2 U = 0 \qquad \text{(П2.33)}$$

и любого частного решения неоднородного уравнения, например,

$$U = \frac{2A}{\pi\omega}. \qquad \text{(П2.34)}$$

Уравнение (П2.33) является уравнением с разделяющимися переменными

$$\frac{dU}{U} = -\omega^2 \alpha^2 dt. \qquad \text{(П2.35)}$$

Проинтегрировав (П2.35), получим

$$\ln(U) = -\omega^2 \alpha^2 t + C, \qquad \text{(П2.36)}$$

где C — константа интегрирования.

Выполняя потенцирование, находим решение дифференциального уравнения (П2.33)

$$U(t) = C_1 e^{-\omega^2 \alpha^2 t}, \qquad \text{(П2.37)}$$

где $C_1 = e^C$.

Таким образом, общее решение неоднородного дифференциального уравнения (П2.32) имеет вид:

$$U(t) = C_1 e^{-\omega^2 \alpha^2 t} + \frac{2A}{\pi\omega}. \qquad \text{(П2.38)}$$

Константу интегрирования C_1 найдем из начальных условий (П2.31):

$$C_1 + \frac{2A}{\pi\omega} = 0. \qquad \text{(П2.39)}$$

Из (П2.39) найдем

$$C_1 = -\frac{A\omega\alpha^2}{\pi}. \qquad \text{(П2.40)}$$

Подставляя (П2.40) в (П2.38), находим решение задачи Коши неоднородного дифференциального уравнения (П2.32):

$$U(t) = \frac{2A}{\pi\omega}\left(1 - e^{-\omega^2 \alpha^2 t}\right). \qquad \text{(П2.41)}$$

Выражение (П2.41) есть результат применения синус преобразования к искомой функции $u(x,t)$.

Шаг 3. Выполнение обратного синус-преобразования.

Для этого вычислим интеграл

$$u(x,t) = F_s^{-1}[U] = \int_0^\infty \frac{2A}{\pi\omega}\left(1 - e^{-\omega^2\alpha^2 t}\right)\sin(\omega x)\, d\omega . \qquad (П2.42)$$

Так как

$$\int_0^\infty \frac{\sin(\omega x)}{\omega} d\omega = \frac{\pi}{2},$$

первый интеграл в (П2.42) равен

$$\int_0^\infty \frac{2A}{\pi}\frac{\sin(\omega x)}{\omega} d\omega = A . \qquad (П2.43)$$

Второй интеграл в (П2.42)

$$I_2 = \int_0^\infty \frac{2A}{\pi} e^{-\omega^2\alpha^2 t}\frac{\sin(\omega x)}{\omega} d\omega \qquad (П2.44)$$

является равномерно сходящимся, поэтому его можно продифференцировать по параметру x:

$$\frac{\partial I_2}{\partial x} = \int_0^\infty \frac{2A}{\pi} e^{-\omega^2\alpha^2 t}\cos(\omega x)\, d\omega . \qquad (П2.45)$$

Воспользовавшись табличным значением интеграла

$$\int_0^\infty e^{-\omega^2\alpha^2 t}\cos(\omega x)\, d\omega = \frac{\sqrt{\pi}}{2}\frac{e^{-\left(\frac{x}{2\alpha\sqrt{t}}\right)^2}}{\alpha\sqrt{t}},$$

для (П2.45) получим

$$\frac{\partial I_2}{\partial x} = \frac{A}{\sqrt{\pi}}\frac{e^{-\left(\frac{x}{2\alpha\sqrt{t}}\right)^2}}{\alpha\sqrt{t}} . \qquad (П2.46)$$

Проинтегрировав (П2.46) по переменной x, найдем выражение для интеграла (П2.44):

$$I_2 = \frac{A}{\sqrt{\pi}}\int_0^x \frac{e^{-\left(\frac{\xi}{2\alpha\sqrt{t}}\right)^2}}{\alpha\sqrt{t}} d\xi = \frac{A}{\sqrt{\pi}}\int_0^{\frac{x}{\alpha\sqrt{t}}} e^{-\left(\frac{\zeta}{2}\right)^2} d\zeta = \frac{A}{2}\operatorname{erf}\left(\frac{x}{\alpha\sqrt{t}}\right). \qquad (П2.47)$$

Подставляя (П2.43), (П2.47) в (П2.42), получим окончательное выражение для решения краевой задачи УЧП (П2.25):

$$u(t,x) = A\left(1 - 0.5erf\left(\frac{x}{2\alpha\sqrt{t}}\right)\right).$$

П2.2.2.3. Решение краевой задачи уравнения в частных производных с использованием преобразования Фурье

Рассмотрим решение следующей краевой задачи УЧП

$$u_t = \alpha^2 u_{xx},\ -\infty \le x \le \infty,\ 0 \le t \le \infty, \qquad (П2.48)$$

с начальными условиями

$$u(x,0) = \varphi(x),\ 0 \le x \le \infty \qquad (П2.49)$$

Шаг 1. Преобразование задачи.

Так как пространственная переменная x изменяется в пределах от $-\infty$ до $+\infty$, выполним преобразование Фурье по переменной x уравнения (П2.48):

$$F[u_t] = \alpha^2 F[u_{xx}],$$

и начальных условий (П2.49)

$$F[u(x,0)] = F[\varphi(x)].$$

Для нахождения фурье-образов производных, используя известное свойство преобразования Фурье:

$$F[u_x] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} u_x(x,t)e^{-i\xi x}dx = i\xi F[u],$$

$$F[u_{xx}] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} u_{xx}(x,t)e^{-i\xi x}dx = -\xi^2 F[u],$$

$$F[u_t] = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} u_t(x,t)e^{-i\xi x}dx = \frac{\partial}{\partial t}F[u],$$

получаем

$$\frac{dU}{dt} = -\alpha^2\xi^2 U(t), \qquad (П2.50)$$

где $U(t) \equiv F[u(x,t)]$,

$$U(0) = \Phi(\xi), \qquad (П2.51)$$

где $\Phi(\xi) = F\left[\varphi(x)\right]$.

Отметим, что функция $U(t)$ зависит не только от переменной t, но и от переменной ξ, которая в уравнении (П2.50) играет роль константы.

Шаг 2. Решение уравнения (П2.50).

Так как переменная ξ в уравнении (П2.50) играет роль константы, решение задачи Коши имеет вид:

$$U(t) = \Phi(\xi) e^{-\alpha^2 \xi^2 t}. \qquad (П2.51)$$

Шаг 3. Нахождение обратного преобразования

Искомое решение $u(x,t)$ находится по формуле

$$u(x,t) = F^{-1}\left[U(\xi,t)\right] = F^{-1}\left[\Phi(\xi) e^{-\alpha^2 \xi^2 t}\right]. \qquad (П2.52)$$

Для поиска явного вида функции $u(x,t)$ используем известную теорему о свертке двух функций, в соответствие с которой фурье-образ свертки двух функций $f(x)$ и $g(x)$

$$f(x) * g(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(\xi) g(x-\xi)\, d\xi = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x-\xi) g(\xi)\, d\xi$$

равен произведению их фурье-образов:

$$F\left[f * g\right] = F[f] F\left[g\right]. \qquad (П2.53)$$

Применяя к (П2.53) обратное преобразование Фурье, получим

$$f * g = F^{-1}\left[F\left[f\right] F\left[g\right]\right]. \qquad (П2.54)$$

Сравнивая (П2.53) и (П2.54), видим, что

$$F^{-1}\left[\Phi(x) e^{-\alpha^2 \xi^2 t}\right] = F^{-1}\left[\Phi(\xi)\right] * F^{-1}\left[e^{-\alpha^2 \xi^2 t}\right] = \varphi(\xi) * F^{-1}\left[e^{-\alpha^2 \xi^2 t}\right], \qquad (П2.55)$$

где фурье-образ функции $F^{-1}\left[e^{-\alpha^2 \xi^2 t}\right]$ равен:

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\alpha^2 \xi^2 t} e^{i\xi x} d\xi = \frac{1}{\sqrt{2}} \frac{e^{-\left(\frac{x}{2\alpha\sqrt{t}}\right)^2}}{\alpha\sqrt{t}}. \qquad (П2.56)$$

Подставляя (П2.56) в (П2.54) и используя определение свертки двух функций, окончательно получаем решение краевой задачи УЧП:

$$u(t,x) = \frac{1}{2\alpha\sqrt{\pi t}} \int_{-\infty}^{\infty} \varphi(\xi) e^{-\left(\frac{x-\xi}{2\alpha\sqrt{t}}\right)^2} d\xi.$$

П2.2.2.4. Решение краевой задачи УЧП методом преобразования Лапласа

Рассмотрим решение следующей краевой задачи УЧП

$$u_t = u_{xx},\ 0 \le x \le \infty,\ 0 \le t \le \infty, \tag{П2.57}$$

с начальными условиями

$$u(x,0) = u_0,\ 0 \le x \le \infty \tag{П2.58}$$

и граничными условиями

$$u_x(0,t) - u(0,t) = 0,\ 0 \le t < \infty. \tag{П2.59}$$

Шаг 1. Преобразование задачи.

Для нахождения решения данной краевой задачи будем использовать преобразование Лапласа по переменной *t*.

Примечание

Здесь можно выполнять преобразование по переменной *x*, так как ее диапазон изменения [0, ∞].

Найдем образы частных производных, входящих в уравнение (П2.57):

$$L[u_t] = \int_0^\infty u_t(x,t)e^{-st}dt = u(x,t)e^{-st}\Big|_0^\infty + s\int_0^\infty u(x,t)e^{-st}dt = sU(x,s) - u(x,0),$$

$$L[u_x] = \int_0^\infty u_x e^{-st}dt = \frac{\partial}{\partial x}U(x,s)$$

$$L[u_{xx}] = \int_0^\infty u_{xx}e^{-st}dt = \frac{\partial^2}{\partial x^2}U(x,s).$$

Подставляя полученные выражения в (П2.57), получаем обыкновенное дифференциальное уравнение

$$sU(x) - u_0 = \frac{d^2U}{dx^2},\ 0 \le x < \infty. \tag{П2.60}$$

Начальные условия для уравнения (П2.60) получаем преобразованием (П2.58):

$$\frac{dU}{dx}(0) = U(0). \tag{П2.61}$$

Шаг 2. Решение дифференциального уравнения (П2.60).

Дифференциальное уравнение (П2.60) является обыкновенным дифференциальным уравнением второго порядка. Для нахождения решения краевой

задачи для данного уравнения необходимо задать еще одно условие, которое достаточно очевидно из физических соображений: $U(x) \to 0$ при $x \to +\infty$.

Общее решение (П2.60) есть сумма общего решения однородного уравнения

$$\frac{d^2U}{dx^2} - sU(x) = 0$$

и частного решения неоднородного уравнения u_0/s:

$$U(x) = c_1 e^{\sqrt{s}x} + c_2 e^{-\sqrt{s}x} + \frac{u_0}{s}. \qquad (П2.61)$$

Из второго граничного условия следует, что $c_1 \equiv 0$. Константа c_2 является решением уравнения

$$-c_2\sqrt{s} = c_2 + \frac{u_0}{s},$$

полученного подстановкой (П2.61) в (П2.60),

$$c_2 = -\frac{u_0}{s\left(\sqrt{s}+1\right)}. \qquad (П2.62)$$

Подставляя (П2.62) в (П2.61), получаем окончательное выражение для $U(x)$:

$$U(s) = u_0\left(1 - \frac{1}{s\left(\sqrt{s}+1\right)}\right). \qquad (П2.63)$$

Шаг 3. Определение температурного поля $u(x,t)$.

Для определения температурного поля $u(x,t)$ необходимо вычислить следующий интеграл:

$$u(x,t) = L^{-1}\left[U(x,s)\right] = \int_{c-i\infty}^{c+i\infty} U(s)e^{st}ds = \int_{c-i\infty}^{c+i\infty}\left(\frac{1}{s} - \frac{1}{s\left(\sqrt{s}+1\right)}\right)e^{st}ds, \qquad (П2.64)$$

используя для этого, например, методы функций комплексного переменного.

Результат его вычисления известен и приводится в справочниках по специальным функциям:

$$u(x,t) = u_0 - u_0\left[\operatorname{erfc}\left(\frac{x}{2\sqrt{t}}\right) + \operatorname{erfc}\left(\sqrt{t} + \frac{x}{2\sqrt{t}}\right)e^{x+t}\right], \qquad (П2.65)$$

где

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-\xi^2} d\xi$$

— функция, дополнительная к интегралу вероятностей.

Выражение (П2.65) является решением краевой задачи УЧП (2.57).

П2.3. Метод преобразования зависимых переменных

Продемонстрируем метод преобразования координат на примере решения следующей краевой задачи:

$$u_t = \alpha^2 u_{xx} - \beta u\,,\ 0 \le x \le 1,\ 0 \le t < \infty, \qquad (П2.66)$$

— дифференциальное уравнение,

$$u(x,0) = \varphi(x)\,,\ 0 \le x \le 1, \qquad (П2.67)$$

— начальные условия,

$$\begin{cases} u(0,t) = 0, \\ u(1,t) = 0, \end{cases} \qquad (П2.68)$$

— граничные условия.

Основная идея метода состоит в преобразовании функции $u(x,t)$ в функцию $w(x,t)$, такую, что дифференциальное уравнение в частных производных для функции $w(x,t)$ будет проще исходного уравнения. К сожалению, не существует универсальных правил для выбора соответствующего преобразования. В этих условиях приходится использовать интуитивные представления о поведении решений УЧП. Например, уравнение (П2.66) описывает некоторый процесс распространения тепла вдоль стержня. При этом температура $u(x,t)$ в каждой точке стержня меняется за счет двух факторов:

- диффузии тепла вдоль стрежня (слагаемое $\alpha^2 u_{xx}$);
- перенос тепла через боковую поверхность стержня (слагаемое $-\beta u$).

Заметим, что при отсутствии диффузионного переноса тепла вдоль стержня ($\alpha = 0$), температура в каждой точке стержня будет уменьшаться до нуля по экспоненциальному закону

$$u(x_0,t) = u(x_0,0)e^{-\beta t}\,.$$

Данное обстоятельство позволяет предположить, что решение задачи можно искать в виде произведения двух сомножителей

$$u(x,t) = e^{-\beta t} w(x,t), \qquad (П2.69)$$

где $w(x,t)$ — распределение температуры обусловленное диффузией.

Подставив (П2.69) в (П2.66), продифференцировав и приведя подобные члены, получим

$$w_t = \alpha^2 w_{xx}. \qquad (П2.70)$$

Выполнив подстановку (П2.69) в начальные и граничные условия, получим:

$$w(x,0) = \varphi(x) \qquad (П2.71)$$

— начальные условия,

$$\begin{cases} w(0,t) = 0, \\ w(1,t) = 0, \end{cases} \qquad (П2.72)$$

— граничные условия.

Решение УЧП (П2.70) с начальными условиями (П2.71) и граничными условиями (П2.72) методом разделения переменных было получено в разделе П2.1:

$$w(t,x) = \sum_{n=1}^{\infty} a_n e^{-(n\pi\alpha)^2 t} \sin(n\pi x), \qquad (П2.73)$$

где

$$a_n = 2\int_0^1 \varphi(\xi)\sin(\pi n\xi)\, d\xi.$$

Так как функции $u(x,t)$ и $w(x,t)$ связаны преобразованием (П2.69), решение краевой задачи (П2.66)–(П2.68) имеет вид:

$$u(t,x) = e^{-\beta t} \sum_{n=1}^{\infty} a_n e^{-(n\pi\alpha)^2 t} \sin(n\pi x).$$

П2.4. Метод преобразования координат

Проиллюстрируем основную идею метода преобразования координат на примере преобразования дифференциального уравнения в частных производных второго порядка, зависящего от трех переменных x, y и z

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} + \frac{\partial^2 \psi}{\partial z^2} + (E - U(x,y,z))\psi = 0, \qquad (П2.74)$$

где E — некоторая константа,

$$U(x, y, z) = -\frac{1}{\sqrt{x^2 + y^2 + z^2}}. \qquad (П2.75)$$

Данное уравнение появляется в квантовой механике при описании движения электрического заряда в центрально-симметричном электрическом поле.

Перейдем в (П2.74), (П2.75) от декартовой (x, y, z) к параболической системе координат (ξ, η, φ) в соответствие со следующими формулами:

$$x = \sqrt{\xi\eta}\cos\varphi,$$

$$y = \sqrt{\xi\eta}\sin\varphi,$$

$$z = \frac{1}{2}(\xi - \eta),$$

$$r = \sqrt{x^2 + y^2 + z^2} = \frac{1}{2}(\xi + \eta),$$

где $\xi, \eta \in [0, \infty[$, $\varphi \in [0, 2\pi]$.

Поверхности $\xi = \text{const}$, $\varphi = \text{const}$ являются параболоидами вращения. Ось вращения параболоидов совпадает с осью Oz, а их фокусы находятся в начале координат. Справедливы обратные соотношения:

$$\xi = r + z,$$

$$\eta = r - z,$$

$$\varphi = \arctan\left(\frac{x}{y}\right).$$

Параболическая система координат является ортогональной. Квадрат элемента длины dl^2 в данной системе определяется выражением

$$dl^2 = \frac{\xi + \eta}{4\xi}d\xi^2 + \frac{\xi + \eta}{4\eta}d\eta^2 + \xi\eta d\varphi^2, \qquad (П2.76)$$

а элемент объема:

$$dV = \frac{1}{4}(\xi + \eta)d\xi d\eta d\varphi.$$

Из (П2.76) видно, что коэффициенты Ламе h_1, h_2, h_3 в параболической системе координат равны

$$h_1 = \frac{1}{2}\sqrt{\frac{\xi+\eta}{\xi}}, \qquad (П2.77)$$

$$h_2 = \frac{1}{2}\sqrt{\frac{\xi+\eta}{\eta}}, \qquad (П2.78)$$

$$h_3 = \sqrt{\xi\eta}\,. \qquad (П2.79)$$

Подставляя выражения (П2.77)–(П2.79) в общее выражение для оператора Лапласа в криволинейной системе координат

$$\Delta\varphi = \frac{1}{h_1 h_2 h_3}\left[\frac{\partial}{\partial q_1}\left(\frac{h_2 h_3}{h_1}\frac{\partial\varphi}{\partial q_1}\right)+\frac{\partial}{\partial q_2}\left(\frac{h_1 h_3}{h_2}\frac{\partial\varphi}{\partial q_2}\right)+\frac{\partial}{\partial q_3}\left(\frac{h_1 h_2}{h_3}\frac{\partial\varphi}{\partial q_3}\right)\right], \qquad (П2.80)$$

где $q_1 \equiv \xi$, $q_2 \equiv \eta$, $q_3 \equiv \varphi$, получим

$$\Delta = \frac{4}{\xi+\eta}\left[\frac{\partial}{\partial\xi}\left(\xi\frac{\partial}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\eta\frac{\partial}{\partial\eta}\right)\right]+\frac{1}{\xi\eta}\frac{\partial^2}{\partial\varphi^2}\,. \qquad (П2.81)$$

Выражение для функции $U(x, y, z)$ (П2.75) в параболической системе координат принимает вид:

$$U = -\frac{1}{r} = -\frac{2}{\xi+\eta}\,. \qquad (П2.82)$$

Подставляя (П2.81), (П2.82) в (П2.74), получаем уравнение в параболической системе координат:

$$\frac{4}{\xi+\eta}\left[\frac{\partial}{\partial\xi}\left(\xi\frac{\partial\psi}{\partial\xi}\right)+\frac{\partial}{\partial\eta}\left(\eta\frac{\partial\psi}{\partial\eta}\right)\right]+\frac{1}{\xi\eta}\frac{\partial^2\psi}{\partial^2\varphi}+2\left(E+\frac{2}{\xi+\eta}\right)\psi = 0\,. \qquad (П2.83)$$

Ищем решение данного уравнения в виде:

$$\psi(\xi,\eta,\varphi) = f_1(\xi)\,f_2(\eta)\,e^{im\varphi}\,. \qquad (П2.84)$$

Подставив (П2.84) в (П2.83) и сократив множитель $e^{im\varphi}$, получаем

$$\frac{4}{\xi+\eta}\left[\frac{d}{d\xi}\left(\xi\frac{df_1}{d\xi}\right)f_2+\frac{d}{d\eta}\left(\eta\frac{df_2}{d\eta}\right)f_1\right]-\frac{m^2}{\xi\eta}f_1 f_2+2\left(E+\frac{2}{\xi+\eta}\right)f_1 f_2 = 0 \qquad (П2.85)$$

Далее умножим уравнение (П2.85) на $\frac{\xi+\eta}{4}$

$$\left[\frac{d}{d\xi}\left(\xi\frac{df_1}{d\xi}\right)f_2+\frac{d}{d\eta}\left(\eta\frac{df_2}{d\eta}\right)f_1\right]-\frac{m^2}{4}\left(\frac{1}{\xi}+\frac{1}{\eta}\right)f_1 f_2+\left(\frac{E}{2}(\xi+\eta)+1\right)f_1 f_2 = 0$$

и перегруппируем его члены

$$\begin{matrix}\left[\frac{d}{d\xi}\left(\xi\frac{df_1}{d\xi}\right)-\frac{m^2 f_1}{4\xi}+\frac{E}{2}\xi f_1\right]f_2+\ldots\\ \left[\frac{d}{d\eta}\left(\eta\frac{df_2}{d\eta}\right)-\frac{m^2 f_2}{4\eta}+\frac{E}{2}\eta f_2\right]f_1+f_1 f_2=0\end{matrix}. \qquad (П2.86)$$

Введем две постоянные β_1, β_2, связанные условием

$$\beta_1+\beta_2=1. \qquad (П2.87)$$

Подставляя (П2.87) в (П2.86), получим

$$\begin{matrix}\left[\frac{d}{d\xi}\left(\xi\frac{df_1}{d\xi}\right)-\frac{m^2 f_1}{4\xi}+\frac{E}{2}\xi f_1+\beta_1 f_1\right]f_2+\ldots\\ \left[\frac{d}{d\eta}\left(\eta\frac{df_2}{d\eta}\right)-\frac{m^2 f_2}{4\eta}+\frac{E}{2}\eta f_2+\beta_2 f_2\right]f_1=0\end{matrix}. \qquad (П2.88)$$

Уравнение (П2.88) будет справедливым для всех $f_1, f_2 \neq 0$, если

$$\frac{d}{d\xi}\left(\xi\frac{df_1}{d\xi}\right)+\left[-\frac{m^2}{4\xi}+\frac{E}{2}\xi+\beta_1\right]f_1=0, \qquad (П2.89)$$

$$\frac{d}{d\eta}\left(\eta\frac{df_2}{d\eta}\right)+\left[-\frac{m^2}{4\eta}+\frac{E}{2}\eta+\beta_2\right]f_2=0. \qquad (П2.90)$$

Таким образом, в параболической системе координат исходное УЧП сводится к системе из двух обыкновенных дифференциальных уравнений, т. е. оказывается возможным разделить переменные. Так как рассмотрение методов решения уравнений типа (П2.90), (П2.91), выходит за рамки нашей книги, мы отсылаем заинтересованного читателя к [17, Т. 1].

П2.5. Метод разложения по собственным функциям

Продемонстрируем основную идею метода разложения по собственным функциям на примере неоднородной задачи УЧП

$$u_t=\alpha^2 u_{xx}+f(x,t), \quad 0\le x\le 1,\ 0\le t<\infty, \qquad (П2.91)$$

с начальными условиями

$$u(x,0)=\varphi(x) \qquad (П2.92)$$

и граничными условиями

$$\begin{cases}u(0,t)=0\\ u(1,t)=0\end{cases}. \qquad (П2.93)$$

Основная идея метода состоит в разложении плотности источника $f(x,t)$ в ряд

$$f(x,t) = f_1(t)X_1(x) + f_2(t)X_2(x) + \ldots + f_n(t)X_n(x) \qquad (П2.94)$$

по собственным функция $X_n(x)$ соответствующей однородной задачи:

$$u_t = \alpha^2 u_{xx},\ 0 \le x \le 1,\ 0 \le t < \infty \qquad (П2.95)$$

с начальными условиями (П2.92) и граничными условиями (П2.93). Решение данной краевой задачи, как было показано в разделе П2.1, сводится к нахождению решения обыкновенного дифференциального уравнения

$$X'' + \lambda^2 X = 0, \qquad (П2.96)$$

удовлетворяющего заданным граничным условиям

$$X(0) = 0, \qquad (П2.97)$$

$$X(1) = 0. \qquad (П2.98)$$

Задача нахождения решения обыкновенного дифференциального уравнения, удовлетворяющего заданным граничным условиям, называется задачей Штурма-Лиувилля, а функции, являющиеся решением данной задачи, — собственными функциями.

Шаг 1. Нахождение собственных функций задачи Штурма-Лиувилля.

Собственными функциями задачи Штурма-Лиувилля уравнения (П2.95) с краевыми условиями (П2.97), (П2.98) являются функции

$$X_n(x) = \sin(\pi n x),\ \ n = 1, 2, 3\ldots \qquad (П2.99)$$

Подставляя (П2.99) в (П2.94), получаем

$$f(x,t) = f_1(t)\sin(\pi x) + f_2(t)\sin(2\pi x) + \ldots + f_n(t)\sin(n\pi x) + \ldots \qquad (П2.100)$$

Для нахождения функции $f_n(t)$ умножим (П2.100) на $\sin(m\pi x)$ и проинтегрируем по координате x на отрезке [0, 1]. Учитывая, что

$$\int_0^1 \sin(m\pi x)\sin(n\pi x)\,dx = \begin{cases} 0, & m \ne n \\ \dfrac{1}{2}, & m = n \end{cases},$$

в результате получим

$$\int_0^1 f(x,t)\sin(\pi n x)\,dx = \sum_{m=1}^{\infty} f_n(t)\int_0^1 \sin(m\pi x)\sin(m\pi x)\,dx = \frac{1}{2} f_n(t). \qquad (П2.101)$$

Следовательно,

$$f(x,t)=\sum_{n=1}^{\infty} f_n(t)\sin(n\pi x). \qquad (П2.102)$$

Шаг 2. Нахождение отклика $X_n(x)T_n(t)$ на входное воздействие $f_n(t)T_n(t)$.

Подставляя (П2.102) в (П2.91), получим

$$u_t=\alpha^2 u_{xx}+\sum_{n=1}^{\infty} f_n(t)\sin(n\pi x). \qquad (П2.103)$$

Вид уравнения (П2.103) позволяет предположить, что общее решение системы, находящейся под воздействием суммы внешних воздействий, будет равно сумме откликов системы на каждое воздействие. Следовательно, общее решение задачи будем искать в следующем виде:

$$u(x,t)=\sum_{n=1}^{\infty} T_n(t)\sin(n\pi x). \qquad (П2.104)$$

Подставляя (П2.104) в (П2.103), получим

$$\sum_{n=1}^{\infty} T_n'(t)\sin(n\pi x)=-\alpha^2\sum_{n=1}^{\infty}(n\pi)^2 T_n(t)\sin(n\pi x)+\sum_{n=1}^{\infty} f_n(t)\sin(n\pi x). \qquad (П2.105)$$

Подставляя (П2.105) в граничные условия (П2.93), получаем

$$\sum_{n=1}^{\infty} T_n(t)\sin(0)\equiv 0,$$

$$\sum_{n=1}^{\infty} T_n(t)\sin(n\pi)\equiv 0.$$

Следовательно, функция $u(x,t)$, определяемая в соответствии с (П2.104), удовлетворяет заданным граничным условиям.

Подставляя (П2.105) в начальное условие (П2.92), получаем

$$\sum_{n=1}^{\infty} T_n(0)\sin(n\pi x)=\varphi(x). \qquad (П2.106)$$

Для нахождения $T_n(0)$ следует, аналогично тому, как это было сделано ранее, домножить (П2.106) на функцию $\sin(n\pi x)$ и проинтегрировать по переменной x на интервале [0, 1]. В результате получаем:

$$T_n(0)=2\int_0^1 \varphi(\xi)\sin(n\pi\xi)\,d\xi. \qquad (П2.107)$$

Запишем уравнение (П2.105) в следующем виде:

$$\sum_{n=1}^{\infty}\left[T_n' + (n\pi\alpha)^2 T_n - f_n(t)\right]\sin(n\pi x) = 0. \qquad (П2.108)$$

Так как уравнение (П2.108) справедливо при любом значении x, каждая функция $T_n(t)$ является решением дифференциального уравнения

$$T_n' + (n\pi\alpha)^2 T_n = f_n(t), \quad n = 1, 2, \ldots \qquad (П2.109)$$

Таким образом, для нахождения функций $T_n'(t)$ необходимо решить задачу Коши для каждого из уравнений (П2.109) с начальными условиями (П2.107).

Чтобы найти общее решение (П2.109), можно использовать метод вариации постоянных. Общее решение однородного уравнения

$$T_n' + (n\pi\alpha)^2 T_n = 0,$$

описывается функцией

$$T_n(t) = Ce^{-(n\pi\alpha)^2 t}. \qquad (П2.110)$$

Подставляем (П2.110) в (П2.109), считая C функцией, зависящей от переменной t. В результате получаем

$$C'e^{-(n\pi\alpha)^2 t} = f(t).$$

Откуда

$$C = \int_0^t f(\xi)e^{(n\pi\alpha)^2 \xi} d\xi + c, \qquad (П2.111)$$

где c — постоянная интегрирования, определяемая из начальных условий.

Подставляя (П2.111) в (П2.110), находим общее решение дифференциального уравнения (П2.109)

$$T_n(t) = ce^{-(n\pi\alpha)^2 t} + \int_0^t f(\xi)e^{-(n\pi\alpha)^2 (t-\xi)} d\xi. \qquad (П2.112)$$

Константу интегрирования c находим подстановкой (П2.112) в (П2.107):

$$c = a_n.$$

Таким образом, решение задачи Коши уравнения (П2.109) с начальным условием (П2.107) имеет вид:

$$T_n(t) = a_n e^{-(n\pi\alpha)^2 t} + \int_0^t f(\xi)e^{-(n\pi\alpha)^2 (t-\xi)} d\xi. \qquad (П2.113)$$

Анализ решения показывает, что температурный отклик состоит из двух частей: первая часть обусловлена начальными условиями, она определяет свободное остывание стержня, вторая часть обусловлена наличием внешнего источника температуры.

П2.6. Метод функций Грина

Краткое обсуждение основных идей метода функций Грина для решения краевой задачи УЧП, содержащего функции, зависящие от двух переменных, проведем на примере решения неоднородной задачи Дирихле уравнения Пуассона в круге единичного радиуса:

$$u_{rr} + \frac{1}{r}u_r + \frac{1}{r^2}u_{\theta\theta} = f(r,\theta), \quad 0 \le r < 1, \ 0 \le \theta < 2\pi \qquad (П2.114)$$

с граничными условиями

$$u(1,\theta) = 0, \ 0 \le \theta \le 2\pi. \qquad (П2.115)$$

Примечание

В связи с тем, что граница области имеет форму окружности, удобно решать данную задачу в цилиндрической системе координат.

В данной постановке уравнение (П2.114) имеет совершенно прозрачный физический смысл. В некоторой области задано распределение зарядов, описывающееся функцией $f(x,y)$, а также значение потенциала на некоторой кривой C, ограничивающей данную область. Требуется найти распределение потенциала внутри области.

Как известно, точечный электрический заряд q создает в окружающем его пространстве электрическое поле $\vec{E}$, напряженность которого определяется следующим выражением:

$$\vec{E} = \frac{q}{\left|\vec{r} - \vec{R}_0\right|^3}\left(\vec{r} - \vec{R}_0\right), \qquad (П2.116)$$

где $\vec{R}_0$ — радиус-вектор заряда, $\vec{r}$ — радиус-вектор точки наблюдения. В качестве количественной характеристики электрического поля также используется функция $\varphi(x,y)$, называемая потенциалом поля, которая связана с напряженностью поля следующим соотношением:

$$\vec{E} = -\vec{\nabla}\varphi(x,y) = -\left[\frac{\partial\varphi}{\partial x}\vec{e}_x + \frac{\partial\varphi}{\partial y}\vec{e}_y\right], \qquad (П2.117)$$

где $\vec{e}_x, \vec{e}_y$ — координатные орты.

Для электрического поля, создаваемого системой пространственно распределенных зарядов, справедлив принцип суперпозиции:

$$\vec{E} = \sum_{i=1}^{N} \frac{q_i}{\left|\vec{r} - \vec{R}_i\right|}\left(\vec{r} - \vec{R}_i\right), \qquad (П2.118)$$

для потенциалов, соответственно,

$$\varphi\left(\vec{r}\right) = \sum_{i=1}^{N} \varphi\left(\vec{r} - \vec{R}_i\right), \qquad (П2.119)$$

где $\vec{r}$ — радиус-вектор точки наблюдения, $\vec{R}_i$ — радиус-вектор i-го заряда, N — число зарядов системы.

Из (П2.118), (П2.119) ясен подход к нахождению решения задачи. Необходимо решить уравнение (П2.114) для случая, когда в правой части находится функция, описывающая точечный заряд. Затем вычислить потенциал, создаваемый системой зарядов, в соответствии с (П2.119).

Для вычисления потенциала электрического поля, создаваемого непрерывным распределением зарядов в круге единичного радиуса, в точке (r,θ) необходимо вычислить потенциал электрического поля, создаваемого зарядом, размещенным в точке (ρ,ϕ), $G(r,\theta,\rho,\phi)$, который обращается в ноль на границе. Затем следует просуммировать потенциалы полей каждого из зарядов по всей площади круга:

$$u(r,\theta) = \int_0^{2\pi}\int_0^{1} G(r,\theta,\rho,\phi) f(\rho,\phi)\,\rho d\rho d\phi\,. \qquad (П2.120)$$

Предваряя нахождение функции Грина точечного заряда, найдем потенциал двумерного электрического поля, создаваемого точечным зарядом q, расположенным в начале координат, используя для этого полярную систему координат (r,θ), координаты которой связаны с декартовой системой следующими соотношениями:

$$x = r\cos\theta\,,\ \ y = r\sin\theta\,,\ \ 0 \le \theta \le 2\pi\,, \qquad (П2.121)$$

обратно

$$r = \sqrt{x^2 + y^2}\,,\ \ \theta = \arctan\left(\frac{x}{y}\right). \qquad (П2.122)$$

Подставляя (П2.121), (П2.122) в (П2.116) и полагая $\vec{R} \equiv 0$, получаем

$$\vec{E} = \left(\frac{q\cos\theta}{r}, \frac{q\sin\theta}{r}\right). \qquad (П2.123)$$

Из (П2.123) видно, что абсолютное значение напряженности электрического поля в декартовой системе координат зависит только от

$$\left|\vec{E}\right| = \frac{q}{r}$$

и не зависит от азимутального угла θ. Следовательно, в полярной системе координат линиями равной напряженности являются окружности, а составляющая напряженности электрического поля $E_\theta \equiv 0$. С целью нахождения в полярной системе координат выражения для радиальной составляющей электрического поля $\vec{E}_r$ вычислим поток заряда через окружность радиуса *r*, который по закону сохранения должен равняться полному заряду, находящемуся в круге

$$\int_0^{2\pi} E_r r d\theta = q\,. \qquad \text{(П2.124)}$$

Так как $\vec{E}_r$ не зависит от азимутального угла, поток заряда через окружность радиуса *r* равен

$$2\pi E_r r = q\,,$$

откуда

$$E_r = \frac{q}{2\pi r}\,,$$

следовательно,

$$\vec{E} = \left(\vec{E}_r, \vec{E}_\theta\right) = \left(\frac{q}{2\pi r}, 0\right). \qquad \text{(П2.125)}$$

Запишем систему уравнений (П2.117) в полярной системе координат:

$$\vec{E}_r = -\frac{\partial \varphi}{\partial r}\vec{e}_r\,, \qquad \text{(П2.126)}$$

$$\vec{E}_\theta = -\frac{1}{r}\frac{\partial \varphi}{\partial \theta}\vec{e}_\theta\,. \qquad \text{(П2.127)}$$

Подставляя (П2.125) в (П2.126), (П2.127), получаем

$$\frac{q}{2\pi r} = -\frac{\partial \varphi}{\partial r}\,, \qquad \text{(П2.128)}$$

Проинтегрировав (П2.128), окончательно находим уравнение для потенциала точечного заряда в полярной системе координат

$$\varphi(r) = -\frac{q}{2\pi}\ln(r) = \frac{q}{2\pi}\ln\left(\frac{1}{r}\right).$$

Для единичного заряда потенциал принимает вид:

$$\varphi = -\frac{1}{2\pi}\ln(r) = \frac{1}{2\pi}\ln\left(\frac{1}{r}\right). \qquad (П2.129)$$

В случае точечного заряда, находящегося в точке (ρ,ϕ), под r в (П2.129) следует понимать расстояние между точкой расположения заряда и точкой наблюдения.

Единственный недостаток функции (П2.129) состоит в том, что она не удовлетворяет заданным граничным условиям (П2.115). Для его устранения разместим вне круга единичный точечный заряд противоположно знака так, чтобы обеспечить постоянство потенциала на границе области — окружности единичного радиуса. Обозначим координаты точки расположения положительного и отрицательного зарядов (x_0, y_0) и (x_1, y_1) соответственно. Найдем расстояния R, $\overline{R}$ от данных зарядов до текущей точки окружности единичного радиуса $(\cos(\vartheta), \sin(\vartheta))$:

$$R = \sqrt{\left(x_0 - \cos(\vartheta)\right)^2 + \left(y_0 - \sin(\vartheta)\right)^2} = \sqrt{r_0^2 + 1 - 2r_0\cos(\vartheta + \Delta\vartheta_0)}, \qquad (П2.130)$$

$$\overline{R} = \sqrt{\left(x_1 - \cos(\vartheta)\right)^2 + \left(y_1 - \sin(\vartheta)\right)^2} = \sqrt{r_1^2 + 1 - 2r_1\cos(\vartheta + \Delta\vartheta_1)}, \qquad (П2.131)$$

где

$$\Delta\vartheta_0 = \arctan\left(\frac{x_0}{y_0}\right),$$

$$\Delta\vartheta_1 = \arctan\left(\frac{x_1}{y_1}\right),$$

$$r_0 = \sqrt{x_0^2 + y_0^2},$$

$$r_1 = \sqrt{x_1^2 + y_1^2}.$$

Как очевидно из (П2.1.130), (П.1.131), условия равенства потенциала на окружности единичного радиуса выполняется, если

$$\frac{\overline{R}}{R} = \text{const}. \qquad (П2.132)$$

Условие (П2.132) выполняется, если

$$\Delta\vartheta_0 = \Delta\vartheta_1, \qquad (П2.133)$$

т. е. точки лежат на одном луче, проведенном из начала координат, и

$$r_1 = \frac{1}{r_0}. \qquad (П2.134)$$

Как очевидно, координаты точки расположения отрицательного заряда равны $(1/\rho, \varphi)$ в полярной системе координат.

Полный потенциал, создаваемый системой двух зарядов в точке с координатами (r, θ) (рис. П2.2), $u(r, \theta)$ равен

$$u(r, \theta) = \frac{1}{2\pi} \ln\left(\frac{1}{R}\right) - \frac{1}{2\pi} \ln\left(\frac{1}{\bar{R}}\right). \qquad (П2.135)$$

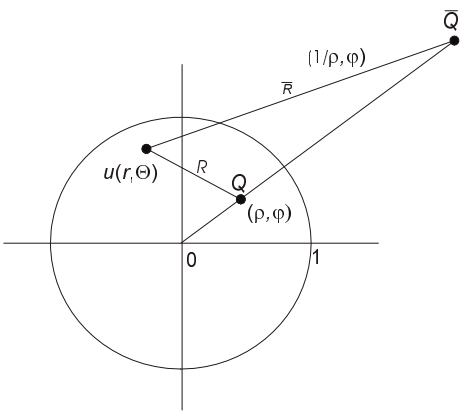


Рис. П2.2. Схема расположения зарядов разных знаков, обеспечивающих постоянный потенциал на окружности единичного радиуса

Подставляя в (П2.135) координаты произвольной точки окружности единичного радиуса, найдем, что потенциал на окружности $\varphi_{R=1}$ равен

$$\varphi_{R=1} = -\frac{1}{2\pi} \ln(\rho).$$

С учетом полученного результата функция Грина окончательно записывается в следующем виде:

$$G(r, \theta, \rho, \phi) = \frac{1}{2\pi}\left[\ln\left(\frac{1}{R}\right) - \ln\left(\frac{1}{\bar{R}}\right) + \ln(\rho)\right], \qquad (П2.136)$$

где первый член — потенциал, создаваемый положительным зарядом, второй член — потенциал, создаваемый отрицательным зарядом, третий член — потенциал, добавленный для вычитания постоянного потенциала,

$$R = \sqrt{r^2 - 2\rho r\cos(\theta - \phi) + \rho^2} \qquad (П2.137)$$

— расстояние от положительного заряда до точки наблюдения (рис. П2.2),

$$\overline{R} = \sqrt{r^2 - 2\frac{r}{\rho}\cos(\theta - \phi) + \frac{1}{\rho^2}} \qquad (П2.138)$$

— расстояние от отрицательного заряда до точки наблюдения (рис. П2.2).

Далее подставляя (П2.136) в (П2.120), окончательно получаем

$$u(r,\theta) = \frac{1}{2\pi}\int_0^{2\pi}\int_0^1 \ln\left(\frac{\rho\overline{R}}{R}\right) f(\rho,\varphi)\rho d\rho d\phi\,, \qquad (П2.139)$$

где $R, \overline{R}$ — определяются по формулам (П2.137), (П2.138).

Формула (П2.139) дает решение задачи Дирихле для уравнения Пуассона в круге единичного радиуса с помощью функции Грина данной задачи. При известной плотности распределения зарядов интеграл в (П2.139) можно вычислить, например, численно.

В общем случае функция Грина находится решением дифференциального уравнения, в котором в правой части функция $f(r,\theta)$ заменена функцией плотности единичного заряда, помещенного в некоторую произвольно выбранную точку (ρ,ϕ):

$$f_1(r,\theta,\rho,\phi) = \delta(r-\rho,\theta-\phi)\,,$$

где $\delta(r-\rho,\theta-\phi)$ — дельта функция Дирака, обладающая следующими свойствами:

$$\delta(r-\rho,\theta-\phi) = \frac{\delta(r-\rho)\delta(\theta-\phi)}{r} = \begin{cases} 0, & \text{если } r \neq \rho \wedge \theta \neq \phi \\ \infty, & \text{если } r = \rho \wedge \theta = \phi \end{cases},$$

$$\int_{-\infty}^{\infty} \delta(r-\rho,\theta-\phi)\rho d\rho d\phi = 1\,.$$

Приведем для справки функции Грина УЧП других типов.

❒ Уравнение теплопроводности ($u_t = -\alpha^2\Delta u$, Δ — оператор Лапласа):

$$G(\vec{r},t,\vec{r}',t') = \frac{\Theta(t-t')}{\left[4\pi\alpha^2(t-t')\right]^{-\frac{n}{2}}} e^{-\frac{(\vec{r}-\vec{r}')^2}{4\alpha^2(t-t')}},$$

где $\Theta(t-t')$ — функция Хевисайда

$$\Theta(x)=\begin{cases}0, & \text{если } x<0\\ 1, & \text{если } x\geq 0\end{cases}.$$

❒ Уравнение Гельмгольца ($\Delta u+k^2u=0$):

$$G_1(x,x')=\frac{1}{2ik}e^{ik|x-x'|},\ n=1,$$

n — размерность пространства,

$$G_2(\vec{r},\vec{r}')=\frac{i}{4}H_0^{(1)}\left(k|\vec{r}-\vec{r}'|\right),\ n=2,$$

где $H_0^{(1)}$ — функция Ханкеля,

$$G_3(r,\vec{r}')=\frac{e^{ik|\vec{r}-\vec{r}'|}}{4\pi|\vec{r}-\vec{r}'|},\ n=3.$$

❒ Волновое уравнение ($\frac{\partial^2 u}{\partial t^2}=\alpha^2\Delta u$):

$$G_1(x,t,x',t')=\frac{1}{2a}\Theta\left(\alpha(t-t')-|x-x'|\right),\ n=1,$$

$$G_2(\vec{r},t,\vec{r}',t')=\frac{1}{2\pi a}\left(\alpha^2(t-t')^2-|\vec{r}-\vec{r}'|^2\right)^{-\frac{1}{2}}\Theta\left(\alpha(t-t')-|\vec{r}-\vec{r}'|\right),\ n=2,$$

$$G_3(\vec{r},t,\vec{r}',t')=\frac{1}{2\pi a}\Theta(t-t')\delta\left(\alpha^2(t-t')^2-|\vec{r}-\vec{r}'|\right),\ n=3.$$

Отметим, что функция Грина волнового уравнения тождественно равна нулю при $t<t'$, поэтому ее называют запаздывающей.

П2.7. Метод интегральных уравнений

Проиллюстрируем основную идею метода интегральных уравнений на примере следующего УЧП

$$\left(\vec{\nabla}^2+k^2-U\right)\psi=0, \qquad (\text{П2.140})$$

где $\vec{\nabla}^2=\Delta$ — оператор Лапласа.

Переписав данное уравнение в форме

$$\left(\vec{\nabla}^2+k^2\right)\psi=U\psi, \qquad (\text{П2.141})$$

видим, что, следуя методу, описанному в предыдущем разделе, можем сразу написать общее решение (П2.141)

$$\psi(\vec{r}) = -\frac{1}{4\pi}\int G(\vec{r},\vec{r}_0)U(\vec{r}_0)\psi(\vec{r}_0)dV_0\ , \qquad (П2.142)$$

где удовлетворяющая граничным условиям функция $G(\vec{r},\vec{r}_0)$ является решением уравнения

$$\left(\vec{\nabla}^2 + k^2\right)G(\vec{r},\vec{r}_0) = 4\pi\delta(\vec{r}-\vec{r}_0)\,.$$

Проверим, что (П2.142) является решением УЧП (П2.140). Подставляя (П2.142) в левую часть (П2.140), имеем:

$$-\frac{1}{4\pi}\left(\vec{\nabla}^2 + k^2\right)\int G(\vec{r},\vec{r}_0)U(\vec{r}_0)\psi(\vec{r}_0)dV_0 =$$
$$= -\frac{1}{4\pi}\int\left(\vec{\nabla}^2 + k^2\right)G(\vec{r},\vec{r}_0)U(\vec{r}_0)\psi(\vec{r}_0)dV_0 =$$
$$= -\frac{1}{4\pi}\int -4\pi\delta(\vec{r}-\vec{r}_0)U(\vec{r}_0)\psi(\vec{r}_0)dV_0 = U(\vec{r})\psi(\vec{r}).$$

Сравнивая полученный результат с правой частью (П2.140), убеждаемся в их тождественности.

Анализ уравнения (П2.141) показывает, что искомая функция $\psi(\vec{r})$ находится в левой и правой части под интегралом. Уравнения данного типа (в которых искомая функция находится под знаком интеграла) называются интегральными уравнениями. Классификация интегральных уравнений и численные методы их решения описаны в *главе 11.*

Более наглядно метод интегральных уравнений можно продемонстрировать на примере задачи Штурма-Лиувилля для уравнения

$$\frac{d}{dz}\left[p\frac{d\psi}{dz}\right] + \left[q(z) + \lambda r(z)\right]\psi = 0 \qquad (П2.142)$$

с краевыми условиями Дирихле

$$\psi(0) = \psi(l) = 0\,. \qquad (П2.143)$$

Для того чтобы свести (П2.142) к интегральному уравнению, введем функцию Грина $G(z,z_0)$, удовлетворяющую уравнению

$$\frac{d}{dz}\left[p\frac{dG(z,z_0)}{dz}\right] + q(z)G(z,z_0) = -\delta(z-z_0)\,. \qquad (П2.144)$$

В соответствие с (П2.143) краевые условия для функции Грина $G(z,z_0)$ имеют вид

$$G(0,z_0)=G(l,z_0)=0\,. \qquad (П2.145)$$

Перенося в (П2.142) в правую часть член $\lambda r\psi$, получаем:

$$\frac{d}{dz}\left[p\frac{d\psi}{dz}\right]+q(z)\psi=-\lambda r(z)\psi\,. \qquad (П2.146)$$

Из (П2.146) видно, что, следуя общему подходу, нужно искать решение методом функций Грина в виде

$$\psi(z)=\lambda\int_0^l G(z,z_0)r(z_0)\psi(z_0)dz_0\,. \qquad (П2.147)$$

Перенося в (П2.142) $\lambda r\psi$ в правую часть, получаем

$$\psi(z)=\lambda\int_0^1 G(z,z_0)r(z_0)\psi(z_0)dz_0+\left[\psi(0)p(0)\frac{dG(z,z_0)}{dz_0}\right]_{z_0=0}- -\left[\psi(l)p(l)\frac{dG(z,z_0)}{dz_0}\right]_{z_0=l}. \qquad (П2.148)$$

Уравнение (П2.148) является интегральным уравнением, так как функция $\psi(x)$ удовлетворяет условиям Дирихле, то интегральное уравнение имеет вид (П2.147). При этом интегральное уравнение включает в себя все данные, относящиеся к задаче, и никакие дополнительные условия накладывать на функции $\psi(x)$ не нужно.

Приведем для справки интегральные уравнения, которым удовлетворяют классические ортогональные функции, их решения и соответствующие функции Грина:

❒ $\dfrac{d^2\psi}{dx^2}+\lambda\psi=0$, $\psi(0)=\psi(l)=0$,

$$\psi(z)=\lambda\int_0^l G(z,z_0)\psi(z_0)dz_0\,,$$

$$G(z,z_0)=\frac{1}{l}\begin{cases}z(l-z_0), z<z_0,\\ z(l-z), z>z_0.\end{cases}$$

Решения: $\psi(z)=\sin\left(\dfrac{n\pi z}{l}\right),\lambda=\left(\dfrac{n\pi}{l}\right)^2$, n — целое.

❒ $\frac{d}{dz}\left[\left(1-z^2\right)\frac{d\psi}{dz}\right]+\lambda\psi=0$, ψ конечна при $z=\pm1$.

$$\psi(z)=\lambda\int_{-1}^{1}G(z,z_0)\psi(z_0)\,dz_0-\frac{1}{2}\int_{-1}^{1}\psi(z_0)\,dz_0,$$

$$G(z,z_0)=\frac{1}{2}\begin{cases}\ln\left(\frac{1+z}{1-z_0}\right), z<z_0,\\ \ln\left(\frac{1+z_0}{1-z}\right), z>z_0.\end{cases}$$

Решения: полиномы Лежандра $P_n(z)$, $\lambda=n(n+1)$, n — целое.

❒ $\frac{1}{z}\frac{d}{dz}\left(z\frac{d\psi}{dz}\right)+\left(\lambda-\frac{n^2}{z^2}\right)\psi=0$, ψ конечна при $z=0,\infty$.

$$\psi(z)=\lambda\int_{0}^{l}G(z,z_0)\psi(z_0)\,dz_0,$$

$$G(z,z_0)=\frac{1}{2n}\begin{cases}\left(\frac{z}{z_0}\right)^n, z<z_0,\\ \left(\frac{z_0}{z}\right), z>z_0.\end{cases}$$

Решения: функции Бесселя $J_n\left(\sqrt{\lambda}z\right)$.

❒ $\frac{d^2\psi}{d^2z}+\left(\beta^2-\alpha^2z^2\right)\psi=0$

или

$\frac{d^2\psi}{d^2z}+\left(\lambda-\alpha-\alpha^2z^2\right)\psi=0$, $\lambda=\alpha^2+\beta^2$, ψ конечна при $z=\pm\infty$.

$$\psi(z)=\lambda\int_{0}^{l}G(z,z_0)\psi(z_0)\,dz_0,$$

$$G(z,z_0)=\sqrt{\frac{\alpha}{\pi}}\begin{cases}e^{\frac{\alpha z^2}{2}}\int_{-\infty}^{z}e^{-\alpha\xi^2}d\xi\, e^{\frac{\alpha z_0^2}{2}}\int_{z_0}^{\infty}e^{-\alpha\xi^2}d\xi, z<z_0,\\ e^{\frac{\alpha z_0^2}{2}}\int_{-\infty}^{z_0}e^{-\alpha\xi^2}d\xi\, e^{\frac{\alpha z^2}{2}}\int_{z_0}^{\infty}e^{-\alpha\xi^2}d\xi, z>z_0.\end{cases}$$

Решения: функции Эрмита $e^{-\frac{\alpha z^2}{2}} H_n\left(\sqrt{\alpha} z\right)$, $\lambda = 2(n+1)\alpha$, n — целое.

❒ $\frac{1}{z^2}\frac{d}{dz}\left(z^2 \frac{d\psi}{dz}\right) + \left(-\beta^2 + \frac{2\alpha}{z}\right)\psi = 0$, ψ конечна при $z = \pm\infty$.

- Параметр λ отождествляют с 2α:

$$\psi(z) = \lambda \int_0^l G(z, z_0) z_0 \psi(z_0) dz_0, \quad \lambda = 2\alpha,$$

$$G(z, z_0) = \frac{e^{-\beta(z - z_0)}}{2\beta z z_0}.$$

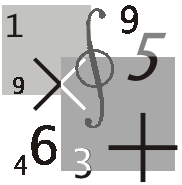
Решения: полиномы Лагерра $e^{-\beta z} L_n^1(2\beta z)$, $L_n(2\beta z)$, $\frac{\alpha}{\beta} - 1 = n$, n — целое.

- Параметр λ отождествляют с $\sqrt{\alpha^2 - \beta^2}$, и тогда эквивалентное уравнение имеет вид:

$$\frac{1}{z^2}\frac{d}{dz}\left(z^2 \frac{d\psi}{dz}\right) + \left(\lambda - \alpha^2 + \frac{2\alpha}{z}\right)\psi = 0, \quad \lambda = \alpha^2 - \beta^2.$$

$$\psi(z) = \lambda \int_0^l G(z, z_0) z_0^2 \psi(z_0) dz_0,$$

$$G(z, z_0) = e^{-\alpha(z + z_0)} \begin{cases} \int_{z_0}^{\infty} \frac{e^{2\alpha\xi}}{\xi^2} d\xi, z < z_0, \\ \int_{z}^{\infty} \frac{e^{2\alpha\xi}}{\xi^2} d\xi, z > z_0. \end{cases}$$



Приложение 3

Приближенные методы решения дифференциальных уравнений в частных производных

Точные решения уравнений в частных производных могут быть получены только для ограниченного круга задач. В этом случае приходится использовать те или иные приближенные методы нахождения решения УЧП. Такая необходимость возникает, например, при наличии поверхностных возмущений — отклонений граничной поверхности или граничных условий от граничной поверхности или граничных условий, допускающих получение точного решения.

В приложении рассмотрено два различных метода получения приближенных решений: *вариационный метод* и *метод теории возмущений*. Первый из них применим, когда поверхностные возмущения велики, второй — когда поверхностные возмущения являются малыми, поэтому решение может быть найдено в виде разложения по степеням малого параметра, характеризующего величину возмущения, главный член разложения является решением при отсутствии возмущений.

П3.1. Вариационный метод

Проиллюстрируем основную идею вариационных методов на примере решения задачи Дирихле эллиптического УЧП

$$\Delta u = f(x,y),\ 0 \le x \le 1,\quad 0 \le y \le 1 \qquad (П3.1)$$

$$u(x,y)=0 \qquad (П3.2)$$

на границе квадрата.

В основе вариационного метода решения данной задачи лежит известная теорема (теорема о минимуме энергии), утверждающая, что функция $u(x,y)$, являющаяся решением задачи Дирихле

$$\Delta u = f(x,y)$$

в области D,

$$u(x,y)=0$$

на границе D, минимизирует функционал

$$J[u]=\iint_D \left[u_x^2+u_y^2+2u(x,y)f(x,y)\right]dxdy, \qquad (П3.3)$$

имеющий смысл потенциальной энергии.

Таким образом, задача поиска решения задачи Дирихле (П3.1) сводится к задаче поиска функции, минимизирующей функционал (П3.3).

Для нахождения искомой функции $u(x,y)$ будем использовать метод Ритца, в соответствии с которым функция $u(x,y)$ ищется в виде линейных комбинаций функций (пробных функций), удовлетворяющих граничному условию (П3.2)

$$u(x,y)=\sum_{i=1}^{n} a_i\varphi_i(x), \qquad (П3.4)$$

n — число используемых функций.

Далее приведен типичный набор пробных функций, которые, как очевидно, удовлетворяют граничным условиям:

$$\varphi_1(x,y)=xy(1-x)(1-y),$$

$$\varphi_2(x,y)=x\varphi_1(x,y),$$

$$\varphi_3(x,y)=y\varphi_1(x,y),$$

$$\varphi_4(x,y)=x\varphi_2(x,y),$$

$$\varphi_5(x,y)=y\varphi_2(x,y)$$

$$\varphi_6(x,y)=x\varphi_3(x,y),$$

$$\varphi_6(x,y)=y\varphi_3(x,y)$$

...

Таким образом, первые четыре приближения функции $u(x, y)$ задаются следующими выражениями:

$$u_1(x,y) = a_1 xy(1-x)(1-y),$$

$$u_2(x,y) = xy(1-x)(1-y)[a_1 + a_2 x],$$

$$u_3(x,y) = xy(1-x)(1-y)[a_1 + a_2 x + a_3 y],$$

$$u_4(x,y) = xy(1-x)(1-y)[a_1 + a_2 x + a_3 y + a_4 x^2]$$

...

Подставляя (П3.4) в (П3.3) получим

$$J[u_n] = \int_0^1 \int_0^1 \left\{ \left[\sum_{j=1}^{n} a_j \frac{\partial \varphi_j}{\partial x} \right]^2 + \left[\sum_{j=1}^{n} a_j \frac{\partial \varphi_j}{\partial y} \right]^2 + 2 f(x,y) \sum_{j=1}^{n} a_J \varphi_J \right\} dxdy \qquad (П3.5)$$

Из (П3.5) видно, что функционал является функцией коэффициентов a_j, $j = 1... n$. Для нахождения минимума функционала (П3.3) необходимо найти коэффициенты a_j из условия равенства нулю частных производных $\frac{\partial J}{\partial a_j}$:

$$\frac{\partial J[u_n]}{\partial a_1} = 2 \int_0^1 \int_0^1 \left\{ \sum_{j=1}^{n} \left[\frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_1}{\partial x} + \frac{\partial \varphi_j}{\partial y} \frac{\partial \varphi_1}{\partial y} \right] a_j + f \varphi_1 \right\} dxdy = 0,$$

... (П3.6)

$$\frac{\partial J[u_n]}{\partial a_n} = 2 \int_0^1 \int_0^1 \left\{ \sum_{j=1}^{n} \left[\frac{\partial \varphi_j}{\partial x} \frac{\partial \varphi_n}{\partial x} + \frac{\partial \varphi_j}{\partial y} \frac{\partial \varphi_n}{\partial y} \right] a_j + f \varphi_n \right\} dxdy = 0.$$

Система линейных уравнений (П3.6) может быть записана в матричной форме

$$A\bar{a} = \bar{b},$$

где A — матрица размерности $n \times n$, элементы которой вычисляются по формуле

$$A_{ij} = \int_0^1 \int_0^1 \left[\frac{\partial \varphi_i}{\partial x} \frac{\partial \varphi_j}{\partial x} - \frac{\partial \varphi_i}{\partial y} \frac{\partial \varphi_j}{\partial y} \right] dxdy, \qquad (П3.7)$$

$\bar{a}$ — вектор-столбец неизвестных, $\bar{b}$ — вектор, компоненты которого вычисляются по формуле

$$b_i = -\int_0^1\int_0^1 f(x,y)\varphi_i(x,y)dxdy\,. \qquad (П3.8)$$

Решив данную систему уравнений относительно коэффициентов a_j, получаем приближенную минимизирующую функцию, а значит, и приближенное решение задачи Дирихле (П3.1), (П3.2).

П3.2. Методы теории возмущений

Теория возмущений позволяет получить приближенные решения, близкие к точным, если рассматриваемая задача близка к задаче, решение которой удается найти аналитически. При этом предполагается, что данные отличия не имеют сингулярного характера, поэтому изменение от точно разрешимой задачи до рассматриваемой можно осуществить непрерывным образом. Аналитически данное требование означает, что возмущение является непрерывной функцией параметра λ, определяющего величину возмущения. Если данное требование выполняется, то оказывается возможным получить формулы, описывающие изменение решения при изменении λ. При этом, чем больше λ отличается от нуля, тем большее число членов ряда необходимо удерживать, для получения ответа с нужной степенью точности.

П3.2.1. Применение метода последовательных приближений к решению обыкновенных нелинейных дифференциальных уравнений

Обсуждение основных идей методов теории возмущений начнем с рассмотрения примера использования данного метода для нахождения решения обыкновенного дифференциального уравнения второго порядка, описывающего движение нелинейного математического маятника

$$\ddot{x} + \omega_0^2 x = -\beta x^3\,, \qquad (П3.9)$$

где ω_0, β — некоторые постоянные. Считая член, стоящий в правой части (П3.9), малым (условие, когда данное приближение справедливо, будет описано далее), найдем его решение методом последовательных приближений:

$$x(t) = x_0 + \delta x\,, \qquad (П3.10)$$

где

$$x_0(t) = Ae^{i\omega_0 t}\,, \qquad (П3.11)$$

решение уравнения

$$\ddot{x} + \omega_0^2 x = 0 . \qquad (П3.12)$$

Подставляя (П3.10), (П3.11) в (П3.9), получаем:

$$\begin{aligned} &\ddot{x}_0 + \delta\ddot{x} + \omega_0^2 \left(x_0 + \delta x\right) = -\beta\left(x_0 + \delta x\right)^3 = \\ &= -\beta\left(x_0^3 + 3x_0^2\delta x + 3x_0\left(\delta x\right)^2 + \left(\delta x\right)^3\right) \approx -\beta\left(x_0^3 + 3\left|x_0\right|^2 \delta x\right). \end{aligned} \qquad (П3.13)$$

Учитывая, что x_0 удовлетворяет уравнению (П3.12), получаем

$$\delta\ddot{x} + \omega_0^2\delta x = -\beta\left(x_0^3 + 3\left|x_0\right|^2 \delta x\right). \qquad (П3.14)$$

В правой части (П3.14) имеется член, пропорциональный δx, который наиболее удобно присоединить ко второму члену, стоящему в правой части

$$\delta\ddot{x} + \left(\omega_0^2 + 3\beta\left|x_0\right|^2\right)\delta x = -3\beta x_0^3 . \qquad (П3.15)$$

Так как $\left|x_0\right|^2 = A^2$, уравнение (П3.15) записываем в виде

$$\delta\ddot{x} + \left(\omega_0^2 + 3\beta A^2\right)\delta x = -3\beta x_0^3 . \qquad (П3.16)$$

Из (П3.16) видно, что учет нелинейного члена привел к сдвигу частоты

$$\omega_0^2 \to \omega^2 = \omega_0^2 + 3\beta A^2 ,$$

поэтому при нахождении решения уравнения (П3.15) следует считать, что зависимость $x_0\left(t\right)$ имеет вид

$$x_0\left(t\right) = Ae^{i\omega t} . \qquad (П3.17)$$

Подставляя (П3.17) в (П3.15), получаем:

$$\delta\ddot{x} + \omega^2\delta x = -3\beta A^3 e^{3i\omega t} . \qquad (П3.18)$$

Уравнение (П3.18) является уравнением гармонического осциллятора, совершающего вынужденные колебания по действием внешней вынуждающей силы $-3\beta A^3 e^{3i\omega t}$.

Частный интеграл неоднородного уравнения ищем в виде

$$\delta x = ae^{3i\omega t} . \qquad (П3.19)$$

Подстановка (П3.19) в (П3.18) дает

$$a = \frac{3\beta A^3}{8\omega^2} ,$$

следовательно, частный интеграл неоднородного уравнения равен

$$\delta x = \frac{3\beta A^3}{8\omega^2} e^{3i\omega t},$$

а общее решение неоднородного уравнения (П3.18) имеет вид

$$\delta x = A_1 e^{i\omega t} + \frac{3\beta A^3}{8\omega^2} e^{3i\omega t}. \qquad (П3.20)$$

Подставляя (П3.20) в (П3.10) и считая, что поправка к зависимости $x(t)$ целиком обусловлена действием внешней вынуждающей силы ($A_1 = 0$), окончательно найдем:

$$x(t) = Ae^{i\omega t} + \frac{3\beta A^3}{8\omega^2} e^{3i\omega t}. \qquad (3.21)$$

Из (П3.21) очевиден критерий применимости метода последовательных приближений:

$$\frac{3\beta A^2}{8\omega^2} = \frac{3\beta A^2}{8\left(\omega_0^2 + 3\beta A^2\right)} << 1.$$

П3.2.1. Обычная формула теории возмущений для решения нелинейных дифференциальных уравнений

Рассмотрение обычной формулы теории возмущений на примере уравнения Шредингера, одномерно описывающего движение квантово-механической частицы в потенциале $\lambda V(x)$:

$$\frac{d^2\psi}{dx^2} + \left[k^2 - \lambda U(x)\right]\psi = 0, \qquad (П3.22)$$

где $k^2 = \frac{2m}{\hbar^2}E$, $U = \frac{2m}{\hbar^2}V$, m — масса частицы, E — энергия частицы, $\hbar$ — постоянная Планка.

Будем считать, что движение частицы ограничено в области $0 \le x \le L$ бесконечно высоким потенциальным барьером при $x = 0$ и при $x = L$, а потенциал $\lambda V(x)$ — можно рассматривать как возмущение.

Следовательно, функция $\psi(x)$ должна удовлетворять граничным условиям

$$\psi(0) = \psi(L) = 0.$$

Невозмущенная разрешимая задача содержит те же самые граничные условия и состоит в решении уравнения

$$\frac{d^2\varphi_n}{dx^2} + k_n^2\varphi_n = 0\,. \qquad (\text{П3.33})$$

Решения уравнения (П3.33) таковы:

$$\varphi_n = \sqrt{\frac{2}{L}}\sin\frac{n\pi x}{L},\ n \text{ — целое}, \qquad (\text{П3.34})$$

$$k_n = \left(\frac{n\pi}{L}\right)^2,$$

$$\int_0^L \varphi_n(x)\varphi_m(x)dx = \delta_{nm}$$

— условие ортогональности.

Интегральная формулировка уравнения (П3.22) может быть получена с помощью функции Грина, являющейся решением уравнения

$$\frac{d^2\psi}{dx^2}G_k(x,x_0) + k^2 G_k(x,x_0) = -\delta(x-x_0)\,, \qquad (\text{П3.35})$$

удовлетворяющей тем же граничным условиям, что и функция $\psi(x)$,

$$\psi(x) = -\lambda\int_0^L G_k(x,x_0)U(x_0)\psi(x_0)\,dx_0\,. \qquad (\text{П3.36})$$

Так как функции φ_n образуют полную систему базисных функций, функцию $G_k(x,x_0)$ можно разложить в ряд по φ_n:

$$G_k(x,x_0) = \sum_n A_n\varphi_n(x)\,. \qquad (\text{П3.37})$$

Подставляя (П3.37) в (П3.35), получаем

$$\sum_n A_n\left(k^2 - k_n^2\right)\varphi_n(x) = -\delta(x-x_0)\,. \qquad (\text{П3.38})$$

Умножив (П3.38) на $\varphi_n(x)$, проинтегрировав по объему и используя свойство ортонормированности базисных функций, получаем:

$$A_n = \frac{\varphi_n(x_0)}{k_n^2 - k^2},$$

следовательно,

$$G_k\left(x,x_0\right)=\sum_n\frac{\varphi_n\left(x\right)\varphi_n\left(x_0\right)}{k_n^2-k^2} \qquad (\text{П3.39})$$

— требуемое разложение.

В одномерном случае оказывается возможным записать функцию $G_k\left(x,x_0\right)$ в явном виде:

$$G_k\left(x,x_0\right)=\frac{1}{k\sin\left(kL\right)}\begin{cases}\sin\left(kx\right)\sin\left[k\left(L-x_0\right)\right], x\le x_0,\\ \sin\left(kx_0\right)\sin\left[k\left(L-x\right)\right], x\ge x_0.\end{cases} \qquad (\text{П3.40})$$

Подставляя в интегральное уравнение (П3.36) ряд (П3.39), получаем:

$$\psi\left(x\right)=\lambda\sum_p\frac{\int_0^L\varphi_p\left(x_0\right)U\left(x_0\right)\psi\left(x_0\right)}{k^2-k_p^2}\varphi_p\left(x\right). \qquad (\text{П3.41})$$

Оказывается удобным выделить из ряда (П3.41) член, к которому стремится ψ, когда $\lambda\to 0$. Пусть это будет φ_n. Обозначим соответствующее ψ через ψ_n, так что

$$\psi_n\xrightarrow[\lambda\to 0]{}\varphi_n.$$

Перепишем разложение функции $\psi\left(x\right)$ следующим образом:

$$\psi_n\left(x\right)=\varphi_n\left(x\right)+\lambda\sum_{p\ne n}\frac{\int_0^L\varphi_p\left(x_0\right)U\left(x_0\right)\psi\left(x_0\right)}{k^2-k_p^2}\varphi_p\left(x\right). \qquad (\text{П3.42})$$

Здесь функция $\psi_n\left(x\right)$ пронормирована так, чтобы коэффициент при $\varphi_n\left(x\right)$ был равен единице. Данная процедура всегда возможна, так как функция $\psi_n\left(x\right)$, умноженная на произвольную постоянную, по-прежнему остается решением уравнения (П3.22). Требование равенства коэффициента при $\varphi_n\left(x\right)$ приводит к условию

$$k^2=k_n^2+\lambda\int_0^L\varphi_n\left(x_0\right)U\left(x_0\right)\psi\left(x_0\right)dx_0\,. \qquad (\text{П3.43})$$

Для проверки что (П3.43) согласуется с (П3.22) и (П3.42), умножим (П3.22) на $\varphi_n(x)$ и проинтегрируем от 0 до L. В результате получим:

$$\int_0^L \varphi_n \frac{d^2\psi}{dx^2} dx + k^2 \int_0^L \psi\varphi_n dx - \lambda \int_0^L \varphi_n U \psi dx = 0 . \qquad (П3.44)$$

Первый член полученного выражения вычисляется двукратным интегрированием по частям:

$$\int_0^L \varphi_n \frac{d^2\psi}{dx^2} dx = \varphi_n \frac{d\psi}{dx}\Big|_0^L - \int_0^L \frac{d\varphi_n}{dx}\frac{d\psi}{dx} dx =$$

$$= -\left(\psi \frac{d\varphi_n}{dx}\Big|_0^L - \int_0^L \frac{d^2\varphi_n}{dx^2}\psi dx \right) = \int_0^L \frac{d^2\varphi_n}{dx^2}\psi dx .$$

Выразив $\frac{d^2\varphi_n}{dx^2}$ из уравнения (П3.33) и подставив в предыдущее выражение, получаем:

$$\int_0^L \varphi_n \frac{d^2\psi}{dx^2} dx = -k_n^2 \int_0^L \psi\varphi_n dx , \qquad (П3.45)$$

но для $\psi(x)$ справедливо разложение (П3.42), подставляя которое в (П3.45) и используя условия ортогональности, окончательно находим:

$$\int_0^L \varphi_n \frac{d^2\psi}{dx^2} dx = -k_n^2 . \qquad (П3.46)$$

Подставляя (П3.46) в (П3.44), получаем (П3.43).

Теперь можно применить метод последовательных приближений к уравнению (П3.42) и, подставляя полученные результаты в (П3.43), найти приближения для величины k^2. За нулевое приближение для ψ_n — $\psi_n^{(0)}$ примем $\varphi_n(x)$, далее подставив его в правую часть уравнения (П3.42) вместо $\psi(x)$, получим первое приближение:

$$\psi_n^{(1)} = \varphi_n + \lambda \sum_{p \neq n} \frac{U_{pn}}{k^2 - k_p^2} \varphi_p , \qquad (П3.47)$$

где

$$U_{pn} = \int_0^L \varphi_p(x_0) U(x_0) \varphi_n(x_0) dx_0 . \qquad (П3.48)$$

Второе приближение $\psi_n^{(2)}$ получается подстановкой $\psi_n^{(1)}$ в (П3.42):

$$\psi_n^{(2)} = \varphi_n + \lambda \sum_{n \neq p} \frac{U_{pn}}{k^2 - k_p^2} \varphi_p + \lambda^2 \sum_{pq \neq n} \frac{U_{pq} U_{qn}}{\left(k^2 - k_p^2\right)\left(k^2 - k_q^2\right)} \varphi_p \,. \qquad (П3.49)$$

Продолжая описанную процедуру можно показать, что

$$\psi_n^{(a)} = \varphi_n + \lambda \sum_{n \neq p} \frac{U_{pn}}{k^2 - k_p^2} \varphi_p + \ldots + \\ + \ldots + \lambda^a \sum_{pq \ldots \neq n} \frac{U_{pq} U_{qr} \ldots U_{zn}}{\left(k^2 - k_p^2\right)\left(k^2 - k_q^2\right) \ldots \left(k^2 - k_z^2\right)} \varphi_p \,. \qquad (П3.50)$$

Выражения (П3.49), (П3.50) содержат неизвестное k^2, которое можно определить подстановкой (П3.50) в (П3.43):

$$k^2 = k_n^2 + \lambda U_{nn} + \lambda^2 \sum_{p \neq n} \frac{U_{np} U_{pn}}{k^2 - k_p^2} \varphi_p + \ldots + \varphi_p + \ldots + \\ + \ldots + \lambda^a \sum_{pq \ldots \neq n} \frac{U_{np} U_{pq} U_{qr} U_{rs} \ldots U_{zn}}{\left(k^2 - k_p^2\right)\left(k^2 - k_q^2\right)\left(k^2 - k_r^2\right) \ldots \left(k^2 - k_z^2\right)} \,. \qquad (П3.51)$$

Уравнения (П3.51) можно решать методом последовательных приближений. Если обозначить через $\left(k^2\right)^{(a)}$ a-е приближение, то

$$\left(k^2\right)^{(1)} = k_n^2 + \lambda U_{nn} \,,$$

$$\left(k^2\right)^{(2)} = k_n^2 + \lambda U_{nn} + \lambda^2 \sum_{p \neq n} \frac{U_{np} U_{pn}}{k_n^2 - k_p^2} \,,$$

$$\left(k^2\right)^{(3)} = k_n^2 + \lambda U_{nn} + \lambda^2 \sum_{p \neq n} \frac{U_{np} U_{pn}}{k_n^2 + \lambda U_{nn} - k_p^2} + \lambda^3 \sum_{pq \neq n} \frac{U_{np} U_{pq} U_{qp}}{\left(k_n^2 - k_p^2\right)\left(k_n^2 - k_q^2\right)} \,,$$

$$\left(k^2\right)^{(4)} = k_n^2 + \lambda U_{nn} + \lambda^2 \sum_{p\neq n} \frac{U_{np}U_{pn}}{k_n^2 + \lambda U_{nn} + \lambda^2 \sum\limits_{q\neq n} \dfrac{U_{nq}U_{qn}}{k_n^2 - k_q^2} - k_p^2} +$$

$$+ \lambda^3 \sum_{pq\neq n} \frac{U_{np}U_{pq}U_{qp}}{\left(k_n^2 + \lambda U_{nn} - k_p^2\right)\left(k_n^2 + \lambda U_{nn} - k_q^2\right)} +$$

$$+ \lambda^4 \sum_{pqr\neq n} \frac{U_{np}U_{pq}U_{qr}U_{rn}}{\left(k_n^2 - k_p^2\right)\left(k_n^2 - k_q^2\right)\left(k_n^2 - k_r^2\right)}.$$

В общем случае

$$\left(k^2\right)^{(a)} = k_n^2 + \lambda U_{nn} + \lambda^2 \sum_{p\neq n} \frac{U_{np}U_{pn}}{\left(k^2\right)^{(a-2)} - k_p^2} +$$

$$+ \lambda^3 \sum_{pq\neq n} \frac{U_{np}U_{pq}U_{qn}}{\left[\left(k^2\right)^{(a-3)} - k_p^2\right]\left[\left(k^2\right)^{(a-3)} - k_q^2\right]} + \ldots + \qquad (\text{П3.52})$$

$$+ \lambda^a \sum_{pq\ldots\neq n} \frac{U_{np}U_{pq}U_{qr}U_{rs}\ldots U_{zn}}{\left(k_n^2 - k_p^2\right)\left(k_n^2 - k_q^2\right)\left(k_n^2 - k_r^2\right)\ldots\left(k_n^2 - k_z^2\right)}$$

Соответствующие им волновые функции таковы:

$$\psi_n^{(1)} = \varphi_n(x) + \lambda \sum_{p\neq n} \frac{U_{pn}}{k^2 - k_p^2} \varphi_p(x),$$

$$\psi_n^{(2)} = \varphi_n(x) + \lambda \sum_{n\neq p} \frac{U_{pn}}{k^2 - \lambda U_{nn} - k_p^2} \varphi_p(x) + \lambda^2 \sum_{pq\neq n} \frac{U_{pq}U_{qn}}{\left(k^2 - k_p^2\right)\left(k^2 - k_q^2\right)} \varphi_p(x),$$

$$\ldots$$

$$\psi_n^{(a)} = k_n^2 + \lambda \sum_{p\neq m} \frac{U_{pn}}{\left(k^2\right)^{(a-1)} - k_p^2} \varphi_p(x) +$$

$$+ \lambda^2 \sum_{p\neq n} \frac{U_{np}U_{pn}}{\left(\left(k^2\right)^{(a-2)} - k_p^2\right)\left(\left(k^2\right)^{(a-2)} - k_q^2\right)} \varphi_p(x) + \qquad (\text{П3.53})$$

$$+ \lambda^a \sum_{pq\ldots\neq n} \frac{U_{np}U_{pq}U_{qr}U_{rs}\ldots U_{zn}}{\left(k_n^2 - k_p^2\right)\left(k_n^2 - k_q^2\right)\left(k_n^2 - k_r^2\right)\ldots\left(k_n^2 - k_z^2\right)} \varphi_p(x)$$

Весьма важным для практического использования формул (П3.52), (П3.53) является вопрос о сходимости входящих в них рядов. Применительно к рядам $\psi^{(a)}$ нас интересует сходимость в среднем

$$\lim_{a\to\infty}\left[\psi_n(x)-\psi_n^{(a)}\right]^2=0,$$

для которой требуется сходимость ряда

$$\sum\left|A_p\right|^2,$$

составленного из сумм квадратов коэффициентов при φ_p в разложении $\psi^{(a)}=\sum A_p\varphi_p$.

Можно показать, что в терминах элементов U_{pn} сходимость в среднем для рядов (П3.53) будет иметь место, если

$$\left|U_{pn}\right|^2_{p\to\infty}\simeq k_p^{3-\varepsilon},\varepsilon>0.$$

Сходимость рядов (П3.53) для $\psi^{(a)}$ в среднем не означает сходимости рядов (П3.52) для $\left(k^2\right)^{(a)}$. Можно показать, что для обеспечения сходимости рядов (П3.52) функция $U(x)$ должна быть типа первой производной от дельта функции. Для использования описанного метода при невыполнении данного условия приходится предпринимать специальные меры для улучшения сходимости рядов (П3.53) с помощью методов, описанных в [17, Т. 2].

Формулы (П3.42), (П3.43), (П3.47)–(П3.52) применимы и к уравнениям относительно функций, зависящих от нескольких переменных. Например, если рассматривается движение квантовой частицы в трехмерном потенциале $U_0+\lambda U$, то ψ удовлетворяет уравнению

$$\Delta\psi+\left(k^2-U_0-\lambda U\right)\psi=0,$$

где невозмущенная задача с точным решением φ_n и собственным значением k_n состоит в решении уравнения

$$\Delta\varphi_n+\left(k_n^2-U_0\right)\varphi_n=0.$$

Движения двух частиц описывается уравнением

$$\Delta_1\psi+\Delta_2\psi+\left(k^2-U_0-\lambda U\right)\psi=0.$$

Здесь функция ψ зависит от шести переменных x_1,y_1,z_1 и x_2,y_2,z_2. Однако как в трехмерном, так и в шестимерном случаях возмущение создается чле-

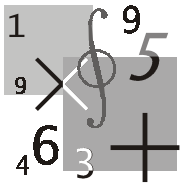
ном λU, поэтому единственное изменение, которое необходимо внести в полученные ранее формулы — изменить правило вычисления U_{pn}, теперь

$$U_{pn} = \int\ldots\int \varphi^*(x_1, x_2, \ldots, x_N) U(x_1, x_2, \ldots, x_N) \varphi(x_1, x_2, \ldots, x_N)\, dx_1 dx_2 \ldots dx_N ,$$

где φ^* — функция комплексно сопряженная функции φ, N — число измерений.

В терминах элементов U_{pn} сходимость ряда (П3.53) в среднем имеет место, если

$$\left|U_{pn}\right|^2_{p\to\infty} \simeq k_p^{4-N-\varepsilon}, \varepsilon > 0 .$$



Приложение 4

Метод обратной задачи рассеивания

Метод обратной задачи рассеивания позволяет находить решения некоторых нелинейных уравнений в частных производных. Первое уравнение, для которого было найдено решение данным методом, — уравнение Кордевега де Фриса (КдФ)

$$u_t - 6uu_x + u_{xxx} = 0,$$

описывающее распространение поверхностных волн в одном направлении в мелком канале. (Здесь под мелким каналом понимается канал, глубина которого значительно меньше длины волны.) Основной особенностью данного уравнения является наличие решений, представляющих собой локализованное в пространстве возмущение, перемещающееся с постоянной скоростью и не меняющее свою форму. Такое решение получило название *солитон* (уединенная волна).

В основе данного метода лежит представление исследуемого нелинейного дифференциального уравнения в виде условия совместимости системы линейных дифференциальных уравнений, получаемых с помощью преобразования Беклунда, представляющего собой способ замены переменных. Напомним, что заменой независимых и зависимых переменных называется произвольное преобразование вида

$$x' = f(x,t), t' = g(x,t)$$

и

$$\tilde{\psi} = F(\psi, x, t).$$

Замена зависимых переменных может иметь более сложный вид, например, $\tilde{\psi}_x = F(\psi, \psi_x, x, t)$. Отметим, что подобная замена не является заменой пере-

менных в полном смысле слова, так как функция $\tilde{\psi}$ может быть определена только с точностью до произвольной функции времени. В подобном случае в качестве преобразований используют некоторые соотношения, связывающие как пространственные, так и временные производные. Преобразованиями Беклунда называют замену переменных вида

$$\tilde{\psi} = F\left(\tilde{\psi}, \psi, \psi_x, \psi_t, \ldots, x, t\right), \tag{П4.1}$$

$$\tilde{\psi}_t = G\left(\tilde{\psi}, \psi, \psi_t, \psi_{tt}, \ldots, x, t\right). \tag{П4.2}$$

Преобразования (П4.1), (П4.2) не являются независимыми, так как $\tilde{\psi}_{xt} = \tilde{\psi}_{tx}$.

Рассмотрим преобразования Беклунда

$$\begin{aligned} u_x &= \frac{\varepsilon}{\sqrt{6\beta}}\left(\alpha\left(\psi - k^2\right) + \beta u^2\right), \\ u_t &= -\frac{\alpha\varepsilon}{\sqrt{6\beta}}\left(\frac{1}{3}\left(\psi + 2k^2\right)\left(\alpha\left(\psi - k^2\right) + \beta u^2\right) + \psi_{xx} + \left(\frac{2\beta}{3}\right)^{1/2} \varepsilon u \psi_x\right), \end{aligned} \tag{П4.3}$$

где $\varepsilon = \pm 1$, для уравнений

$$\begin{aligned} &\psi_t + \alpha\psi\psi_x + \psi_{xxx} = 0, \\ &u_t + \alpha k^2 u_x - \beta u^2 u_x + u_{xxx} = 0. \end{aligned} \tag{П4.4}$$

Условие полной интегрируемости уравнений (П4.3) требует, чтобы $u_{xt} = u_{tx}$. Несложные вычисления с использованием (П4.3) показывают, что

$$\begin{aligned} u_{xt} = \frac{\varepsilon}{\sqrt{6\beta}}\Bigg(&\alpha u_t - \left(\frac{2\beta}{3}\right)^{1/2} \varepsilon u \alpha\left(\frac{1}{3}\left(\psi + 2k^2\right)\left(\alpha\left(\psi - k^2\right) + \beta u^2\right) + \right. \\ &\left. + \psi_{xx} + \varepsilon\left(\frac{2\beta}{3}\right)^{1/2} u\psi_x\right)\Bigg), \end{aligned} \tag{П4.5}$$

$$\begin{aligned} v_{tx} = -\frac{\alpha\varepsilon}{\sqrt{6\beta}}\Bigg(&\psi_x\left(\alpha\left(\psi - k^2\right) + \beta u^2\right) + \frac{\alpha}{3}\left(\psi + 2k^2\right)\psi_x + \psi_{xxx} + \\ &+ \left(\frac{2\beta}{3}\right)^{1/2} \varepsilon u \psi_{xx} + \frac{\varepsilon}{\sqrt{6\beta}}\left(\alpha\left(\psi - k^2\right) + \beta u^2\right)\left(\frac{2\beta}{3} u\left(\psi + 2k^2\right) + \right. \\ &\left. + \varepsilon\left(\frac{2\beta}{3}\right)^{1/2} \psi_x\right)\Bigg). \end{aligned} \tag{П4.6}$$

Сравнивая (П4.5) и (П4.6), видим, что данные уравнения одинаковы, если функция ψ удовлетворяет уравнению КдФ. Преобразование уравнения КдФ в такое уравнение, для которого функция u является решением, можно получить, исключив из первого уравнения (П4.3) ψ и подставив его во второе уравнение (П4.4):

$$u_t + \alpha k^2 u_x - \beta u^2 u_x + u_{xxx} = 0 .$$

Таким образом, преобразования Беклунда действительно устроены так, что уравнение КдФ является их следствием. Верно также и обратное — любое из уравнений (П4.6) может быть получено из другого уравнения и уравнения КдФ.

Первое уравнение (П4.4) является уравнением КдФ, второе — модифицированным уравнением КдФ. Второе уравнение также имеет солитонные решения, которые связаны с солитонными решениями первого уравнения преобразованиями Беклунда (П4.3). Уравнения (П4.3) имеют вид уравнений Рикатти относительно функции u. Как известно, уравнения Рикатти могут быть линеаризованы с помощью подстановки Хопфа-Коула

$$u = -\varepsilon\sqrt{\frac{6}{\beta}}\frac{\varphi_x}{\varphi}, \qquad (П4.7)$$

выбор множителя $-\varepsilon\sqrt{\frac{6}{\beta}}$ обеспечивает исчезновение квадратичного члена в первом уравнении (П4.3).

Подставив (П4.7) в первое уравнение (П4.3) и считая $\varepsilon = 1$, получаем

$$\frac{\partial}{\partial x}\left(-\sqrt{\frac{6}{\beta}}\frac{\varphi_x}{\varphi}\right) = -\sqrt{\frac{6}{\beta}}\left(\frac{\varphi_{xx}}{\varphi} - \left(\frac{\varphi_x}{\varphi}\right)^2\right) =$$

$$= \frac{1}{\sqrt{6\beta}}\left(\alpha\left(\psi - k^2\right) + \beta\left(\sqrt{\frac{6}{\beta}}\frac{\varphi_x}{\varphi}\right)^2\right),$$

откуда, разрешив полученное уравнение относительно φ_{xx}, находим

$$\varphi_{xx} + \frac{\alpha}{6}\psi\varphi = \frac{\alpha k^2}{6}\varphi . \qquad (П4.8)$$

Подставляя (П4.7) во второе уравнение (П4.3) и поступая аналогично предыдущему, получаем

$$\varphi G_x - \varphi_x G = 0 , \qquad (П4.9)$$

где

$$G \equiv \varphi_t - \frac{\alpha}{6}\psi_x\varphi + \frac{\alpha}{3}\left(\psi + 2k^2\right)\varphi_x .$$

Интегрируя (П4.9), найдем

$$\varphi_t + \frac{\alpha}{3}\left(\psi + 2k^2\right)\varphi_x + \left(f(k,t) - \frac{\alpha}{6}\psi_x \right)\varphi = 0 . \qquad (П4.10)$$

Анализ уравнения (П4.8) показывает, что оно является уравнением Шредингера, описывающего рассеивание квантовой частицы на потенциале $\psi(x)$. Предполагая, что $\lim\limits_{x\to\pm\infty}\psi(x) = 0$, будем искать решение данного уравнения $\varphi_{\pm}(x,k)$ в виде падающих и отраженных волн, имеющих в пределе $x \to \pm\infty$ следующие асимптотики:

$$\lim_{x\to+\infty} e^{ikx}\varphi_+(x,k) = 1 ,$$

$$\lim_{x\to+\infty} e^{ikx}\varphi_-(x,k) = 1 .$$

В методе обратной задачи рассеивания набор решений $\varphi_+(k,x)$, $\varphi_+^*(k,x)$, $\varphi_-(k,x)$, $\varphi_-^*(k,x)$, где $*$ означает комплексное сопряжение, называется решениями Йоста. Подстановкой в (П4.8) легко убедиться в том, что уравнение (П4.8) эквивалентно следующим интегральным уравнениям

$$\varphi_-(x,k) = e^{-ikx} - k^{-1}\int_{-\infty}^{x}\psi(y)\sin\left[k(y-x)\right]\varphi_-(x,k)\,dy , \qquad (П4.11)$$

$$\varphi_+(x,k) = e^{-ikx} - k^{-1}\int_{-\infty}^{x}\psi(y)\sin\left[k(y-x)\right]\varphi_+(x,k)\,dy . \qquad (П4.12)$$

Уравнение типа (П4.11), (П4.12) называется интегральным уравнением Марченко.

Как очевидно, множество решений $\varphi_+(x,k), \varphi_+^*(x,k)$ или $\varphi_-(x,k), \varphi_-^*(x,k)$ является базисом рассматриваемого функционального пространства, поэтому функцию $\varphi_-(x,k)$ можно записать в виде разложения по базисным функциям $\left\{\varphi_+, \varphi_+^*\right\}$:

$$\varphi_-(x,k) = a(k)\varphi_+^*(x,k) + b(k)\varphi_+(x,k) , \qquad (П4.13)$$

где

$$a(k) = 1 - \frac{1}{2ik}\int_{-\infty}^{+\infty} \psi(y)\, e^{iky} \varphi_-(k, y)\, dy,$$
$$b(k) = \frac{1}{2ik}\int_{-\infty}^{+\infty} \psi(y)\, e^{iky} \varphi_-(k, y)\, dy. \qquad (П4.14)$$

Выражения (П4.13), (П4.14) позволяют определить матрицу рассеивания

$$S(k) = \begin{pmatrix} T_+(k) & R_-(k) \\ R_+(k) & T_-(k) \end{pmatrix}, \qquad (П4.15)$$

где

$$T_+(k) = T_-(k) = a^{-1}(k), \qquad (П4.16)$$

— коэффициент прохождения,

$$R_+(k) = R_-(k) = b(k)a^{-1}(k) \qquad (П4.17)$$

— коэффициент отражения.

Отметим первое основополагающее обстоятельство метода обратной задачи рассеивания: существует однозначное соответствие между данными рассеивания и формой потенциала рассеивания $\psi(x,t)$. Под данными рассеивания понимается матрица $S(k)$ для вещественных значений k и набор нормировочных постоянных и отрицательных собственных значений энергии дискретного спектра, соответствующих мнимым значениям k. Следовательно, по начальному значению функции $\psi(x,t_0)$ можно определить данные рассеивания. Если удастся связать закон эволюции данных рассеяния с эволюцией потенциала $\psi(x,t)$, удовлетворяющих уравнению КдФ, то с помощью интегральных уравнений (П4.11), (П4.12) по данным рассеивания можно восстановить в любой момент времени потенциал $\psi(x,t)$, решив тем самым задачу Коши для уравнения КдФ.

Второе основополагающее обстоятельство метода обратной задачи рассеивания состоит в том, что для семейства уравнений, связанных с уравнением КдФ различными преобразованиями Бэклунда, данные рассеивания удовлетворяют линейным дифференциальным уравнениям. Однако получение данных уравнений (подробную информацию можно найти в книге Абрамовица М., Сигура Х. "Соитоны и метод обратной задачи" — М.: Мир, 1987) оказывается весьма сложным и выходит за рамки нашей книги. В заключение приведем схему иллюстрирующую общую идею метода обратной задачи рассеивания (рис. П4.1).

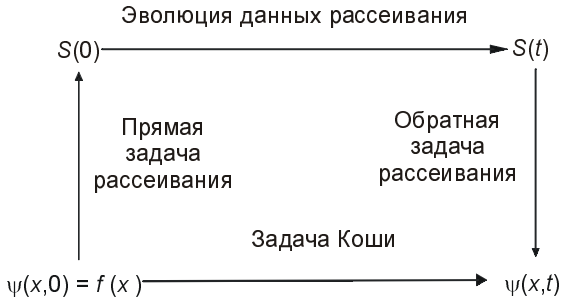


Рис. П4.1. Схема метода обратной задачи рассеивания

Сравнение рис. П4.1. и рис. П2.1 *(см. Приложение 2)* обнаруживает их принципиальное сходство. Данное обстоятельство не случайно, так как основной подход метода обратной задачи рассеивания во многом схож с методом интегральных преобразований.

Литература

1. Амосов А. А., Дубинский Ю. А., Копченова Н. А. Вычислительные методы для инженеров. — М.: Высшая школа, 1994.
2. Бахвалов Н. С., Жидков Н. П., Кобельков Г. М. Численные методы. — М.: Наука, 1987.
3. Березин Н. С., Жидков Н. П. Методы вычислений. — М.: В 2-х тт. — М.: Наука, 1960.
4. Васильев Ф. П. Численные методы решения экстремальных задач. — М.: Наука, 1988.
5. Воеводин В. В. Вычислительные основы линейной алгебры. — М.: Наука, 1977.
6. Волков Е. А. Численные методы. — М.: Наука, 1982.
7. Годунов С. К., Рябенький В.С. Разностные схемы. — М.: Наука, 1977.
8. Демидович Б. П., Марон И. А. Основы вычислительной математики. — М.: Наука, 1966.
9. Демидович Б. П., Марон И. А., Шувалова Э. З. Численные методы анализа. — М.: Наука, 1962.
10. Дьяченко В. Ф. Основные понятия вычислительной математики. — М.: Наука, 1977.
11. Дэннис Дж., Шнабель Р. Численные методы безусловной оптимизации и решения нелинейных уравнений. — М.: Мир, 1988.
12. Заварыкин В. М., Житомирский В. Г., Лапчик М. П. Численные методы. — М.: Просвещение, 1991.
13. Калиткин Н. Н. Численные методы. — М.: Наука, 1978.
14. Ландау Л. Д., Лифшиц Е. М. Механика. — М.: Наука, 2001.

15. Леснин В. В., Лисовец Ю. П. Основы методов оптимизации. — М.: Изд-во МАИ, 1995.

16. Марчук Г. И. Методы вычислительной математики. — М.: Наука, 1977.

17. Морс Ф. М., Фешбах Г. Методы теоретической физики. В 2-х тт. — М.: Изд-во иностранной литературы, 1958.

18. Ортега Дж., Пул У. Введение в численные методы решения дифференциальных уравнений. — М.: Наука, 1986.

19. Рябенький В. С. Введение в вычислительную математику. — М.: Наука, 1994.

20. Самарский А. А. Введение в численные методы. — М.: Наука, 1982.

21. Самарский А. А., Гулин А.В. Численные методы. — М.: Наука, 1989.

22. Самарский А. А., Николаев Е. С. Методы решения сеточных уравнений. — М.: Наука, 1978.

23. Тейлор Дж. Введение в теорию ошибок. — М.: Мир, 1985.

24. Тихонов А. Н., Костомаров Д. П. Вводные лекции по прикладной математике. — М.: Наука, 1984.

25. Турчак Л. И. Основы численных методов. — М.: Наука, 1987.

26. Шикин Е. В., Плис А. И. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователей. — М.: ДИАЛОГ–МИФИ, 1996.

27. Федоренко Р. П. Введение в вычислительную физику. — М.: Изд-во МФТИ, 1994.

28. Фарлоу С. Уравнения с частными производными для научных работников и инженеров. — М.: Мир, 1985.

29. Химмельбау Д. Прикладное нелинейное программирование. — М.: Мир, 1975.