ФГБОУ ВПО «Воронежский государственный технический университет»

И.Н. Пантелеев

СПЕЦГЛАВЫ ВЫСШЕЙ МАТЕМАТИКИ: ЧИСЛЕННЫЕ МЕТОДЫ

Утверждено Редакционно-издательским советом университета в качестве учебного пособия

Воронеж 2013

УДК 681.3.06(075)

Пантелеев И. Н. Спецглавы высшей математики: численные методы / И.Н. Пантелеев. Воронеж: ФГБОУ ВПО «Воронежский государственный технический университет», 2013. – 255 с.

Учебное пособие включает материал, необходимый для подготовки к практическим и лабораторным занятиям по курсу высшей математики во втором семестре. Изложение традиционных разделов численных методов имеет прикладную направленность. Для всех рассмотренных методов приведены программные реализации, а также соответствующие функции пакета MATLAB, приведено большое количество примеров, иллюстрирующих основные методы решения.

Издание соответствует требованиям федерального государственного образовательного стандарта высшего профессионального образования по направлению 280700.62 «Техносферная безопасность», профили «Защита в чрезвычайных ситуациях», «Безопасность жизнедеятельности в техносфере», «Защита окружающей среды», дисциплине высшая математика.

Учебное пособие подготовлено в электронном виде в текстовом редакторе Microsoft Word 2003 и содержатся в файле Vmfmm_ SpGChMet1.pdf.

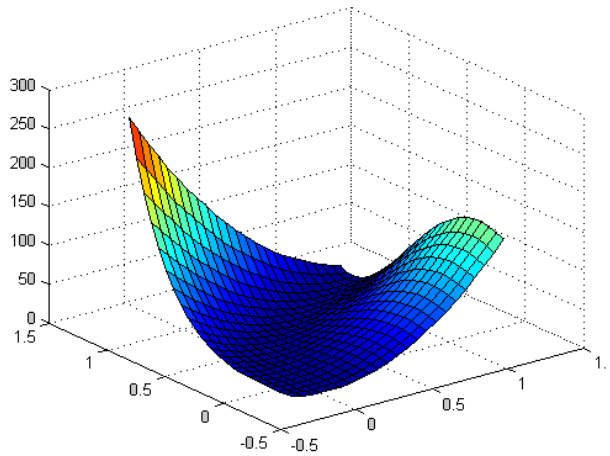
Табл. 6. Ил. 76. Библиогр.: 15 назв.

Рецензенты: кафедра физики Воронежского государственного университета инженерных технологий (зав. кафедрой д-р физ.-мат. наук, проф. Н.Н. Безрядин);
профессор Г.Е. Шунин

И.Н. Пантелеев

СПЕЦГЛАВЫ ВЫСШЕЙ МАТЕМАТИКИ: ЧИСЛЕННЫЕ МЕТОДЫ

Учебное пособие



Воронеж 2013

Учебное издание

Пантелеев Игорь Николаевич

СПЕЦГЛАВЫ ВЫСШЕЙ МАТЕМАТИКИ:
ЧИСЛЕННЫЕ МЕТОДЫ

В авторской редакции

Компьютерный набор И.Н. Пантелеева

Подписано к изданию 10.12.2013.
Объем данных 2516 кб

ФГБОУ ВПО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14

ВВЕДЕНИЕ

В настоящее время численные методы являются мощным математическим средством решения многих научно-технических проблем. Это связано как с невозможностью получить точное аналитическое решение, так и со стремительным развитием компьютерной техники. Существуют многочисленные стандартные программы и интегрированные пакеты прикладных программ. Однако научным и инженерно-техническим работникам важно понимать сущность основных численных методов и алгоритмов, поскольку зачастую интерпретация результатов расчетов нетривиальна и требует специальных знаний особенностей применяемых методов. При использовании численных методов необходимо большое внимание уделять структуре погрешностей при решении конкретных задач. Существуют четыре источника погрешностей, полученных в результате численного решения: математическая и физическая модели, исходные данные, приближенность метода и ошибки округления. Первые два источника приводят к так называемой неустранимой погрешности. Эта погрешность может присутствовать, даже если решение сформулированной задачи найдено точно. Погрешность метода возникает из-за того, что точный оператор и исходные данные, в частности начальные и краевые условия, заменяются по определенным правилам приближенными. Так, производные заменяются их разностными аналогами, интегралы – суммами, функции – специальными многочленами. Как правило, погрешность метода может быть оценена и поддается контролю. Погрешность округления возникает в связи с тем, что вычисления производятся с конечным числом значащих цифр. Поскольку на современных компьютерах число записывается, не менее чем с 10-12 десятичными знаками, то погрешность единичного округления порядка 10^{-10} – 10^{-12} обычно

пренебрежимо мала по сравнению с неустранимой погрешностью и погрешностью метода.

В последнее время компьютерная математика интенсивно развивается как передовое научное направление на стыке математики и информатики. ЭВМ и персональные компьютеры уже давно применяются для математических расчетов. В середине 90-х годов на смену алгоритмическим языкам программирования пришли специализированные системы компьютерной математики. Наибольшую известность получили системы Eureka, Mercury, Mathcad, Derive, Mathematika, Maple и др. Среди ряда современных систем особо выделяется MATLAB. Она прошла многолетний путь развития от узкоспециализированного матричного программного модуля до универсальной интегрированной системы, ориентированной на массовые персональные компьютеры класса IBM PC и Macintosh и рабочие станции UNIX. Система имеет мощные средства диалога, графики и комплексной визуализации. Система MATLAB предлагается разработчиками (корпорация MathWorks, Inc) как язык программирования высокого уровня для технических вычислений, расширяемым большим числом пакетов прикладных программ. Она вобрала в себя не только передовой опыт развития и компьютерной реализации численных методов, накопленный за последние три десятилетия, но и весь опыт становления математики за всю историю ее развития.

В данном пособии изложение традиционных разделов по курсу численных методов имеет прикладную направленность. Для каждого описанного вычислительного метода приведены его программные реализации, а также соответствующие функции 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.

I. ТЕОРИЯ ПОГРЕШНОСТЕЙ

Рассмотрим 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.

1.2. 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 погрешности 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. I.1), в которой требуется предсказать угол отклонения маятника от вертикали 0, начинающего движение в момент времени $t = t_0$

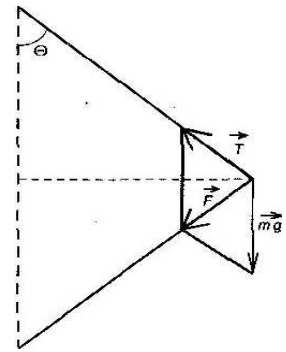


Рис. I.1. Модель математического маятника

Движение маятника может быть описано дифференциальным уравнением второго порядка:

$$l\frac{d^2\theta}{dt^2} + g\sin\theta + \mu\frac{d\theta}{dt} = 0 \qquad \text{(I.1)}$$

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 /, $g, \mu, t_0, y = e_y, \theta(t_0)$ известны с некоторыми погрешностями.
- Для решения уравнения (I.1), не имеющего аналитического решения, приходится использовать численный метод, вследствие чего возникает погрешность метода.
- Вычислительная погрешность, возникающая вследствие конечной точности представления чисел в компьютере.

1.3. Абсолютная и относительная погрешности. Формы записи данных

Определение 1. Если a — точное значение некоторой величины и a^* известное приближение к нему, то абсолютной погрешностью приближенного значения a называют некоторую величину $\Lambda(a^*)$, про которую известно, что

$$|a^* - a| \le \Lambda(a^*) \qquad \text{(I.2)}$$

Определение 2. Относительной погрешностью приближенного значения называют некоторую величину $\delta(a^*)$ про которую известно, что;

$$\left|\frac{a^* - a}{a^*}\right| \le \delta(a^*) \qquad \text{(I.3)}$$

Относительную погрешность часто выражают в процентах.

Определение 3. Значащими цифрами числа называют все цифры в его записи, начиная с первой ненулевой слева.

Пример 1.

$$(a^*) = 0.03045 \qquad (a^*) = 0.03045000$$

Примечание. Здесь цифры, записанные курсивом, значащие.

Определение 4. Значащую цифру называют верной, если модуль погрешности числа не превосходит единицы разряда, соответствующего этой цифре.

Пример 2.

$a^* = 0.03045 \qquad \Lambda(a^*) = 0.000003$

$a^* = 0.03045000 \qquad \Lambda(a^*) = 0.00000007$

Примечание. Здесь цифры, записанные курсивом, верные.

Определение 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 пределам 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 $\Lambda(a^*)$ принято также записывать в виде:

$$a = a^* \pm \Lambda(a^*) \qquad \text{(I.4)}$$

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

Пример 3.

$$a = 1.123 \pm 0.004$$

$$a = 1.123 \pm 4 \cdot 10^{-3}$$

$$1.123 - 0.004 \le a \le 10123 + 0.004$$

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

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

Пример 4.

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

1.4. Вычислительная погрешность

Далее для краткости будем обозначать абсолютную погрешность числа χ как e_x, относительную погрешность $-\delta_x$.

□ 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 $x = e_x$, $y = e_y$

• 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

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

• 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

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

• относительная погрешность

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

□ Погрешность функции зависящей от одной переменной.

• абсолютная погрешность:

$$f(x \pm e_x) = f(x) \pm f^1(x) \cdot e_x$$

$$\nabla f = f(x \pm e_x) - f(x) = f^1(x) e_x$$

• относительная погрешность

$$\frac{|\Delta f|}{|f|} = \frac{|f^1(x)|}{|f(x)|} e_x$$

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

1.5. Понятия о погрешности машинных вычислений

Для представления чисел в памяти компьютера применяют два способа.

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 специальном 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

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

где $\alpha \in \{0,1,\ldots,r-1\}$, то есть

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

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 $|a-a^*|$, 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 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.I.2).

Знак числа / Целая часть числа / Дробная часть числа

Рис. I.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:

$$\delta_1 = \frac{\Delta_1}{a_2} = 0.04\,\%,$$

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

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 $\left(r^{-1} \le |M| \le 1\right)$.

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.

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

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.I.3

Знак порядка	Порядок числа (т разрядов)	Знак мантиссы	Мантисса числа (/ разрядов)

Рис. I.3. Схема машинного слова запоминающего устройства с плавающей головкой

Числа $\pm r^{r^m}$ определяют границы допустимого числового диапазона. Относительная точность представления вещественных чисел равна

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

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, $r = 2$, $l = 40$, $m = 6$. Следовательно, точность представления чисел с плавающей запятой не хуже $2^{-39}\left(\approx 10^{-12}\right)$, граница машинного нуля $2^{-64}\left(\approx 10^{-19}\right)$, машинной бесконечности 2^{63}.

Пример 6. Рассмотрим запоминающее устройство, состоящее из $k = 8$ элементов и имеющее $r = 10\left(r = 0,1,\ldots 9\right)$. 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 $a_2 = 999123123$. 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. I.4.

+	0	1	+	1	1	2	3	1
	Порядок числа			Мантисса числа				
+	0	3	+	9	9	9	1	2

Рис. I.4. Представление чисел a_1, a_2 с плавающей запятой в запоминающем устройстве.

Абсолютные погрешности чисел равны:

$$\Delta_1 = (0.1123123 - 0.11231)\cdot 10^1 = (0.23\cdot 10^{-5})\cdot 10^1,$$
$$\Delta_2 = (0.999123123 - 0.99912)\cdot 10^3 = (0.31\cdot 10^{-5})\cdot 10^5.$$

Относительные погрешности чисел равны:

$$\delta_1 = \frac{\Delta_1}{a_1} = \frac{(0.32\cdot 10^{-5})\cdot 10^{-1}}{0.1123\cdot 10^1} = 2\cdot 10^{-3}\ \%,$$
$$\delta_2 = \frac{\Delta_2}{a_2} = \frac{(0.31\cdot 10^{-5})\cdot 10^3}{0.99912\cdot 10^3} = 3\cdot 10^{-3}\ \%.$$

II. РЕШЕНИЕ УРАВНЕНИЙ С ОДНОЙ ПЕРЕМЕННОЙ

2.1. Общие сведения и основные определения

Наиболее общий вид нелинейного уравнения:

$$F(x) = 0, \qquad \text{(II.1)}$$

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

Определение 1. Всякое число $\xi \in [a, b]$, обращающее функцию $F(x)$ в ноль, называется корнем уравнения (II.1).

Определение 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:

$$F(\xi) = F'(\xi) = \ldots = F^{(k-1)}(\xi) = 0 \qquad \text{(II.2)}$$

Определение 3. Однократный корень называется простым.

Определение 4. Уравнение $F(x) = 0$ и $G(x) = 0$ называются равносильными (эквивалентными), если множества решений данных уравнений совпадают.

Нелинейные уравнения с одной переменной подразделяются на алгебраические и трансцендентные.

Определение 5. Уравнение (II.1) называется алгебраическим, если функция F(x) является алгебраической.

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

$$P_n(x) = a_0 x^n + a_1 x^{n-1} + \ldots a_n, \qquad \text{(II.3)}$$

где $a_0, a_1, \ldots, a_n$ – действительные коэффициенты уравнения, x-неизвестное. Из алгебры известно, что всякое алгебраическое уравнение имеет, по крайней мере, один вещественный или два комплексно сопряженных корня.

Определение 6. Уравнение (II.1) называется трансцендентным, если функция $F(x)$ не является алгебраической.

Определение 7. Решить уравнение (II.1) означает:

1.Установить, имеет ли уравнения корни.

2. Определить число корней уравнения.

3. Найти значение корней уравнения с заданной точностью.

2.2. Отделение корней

Определение 8. Отделение корней — процедура нахождения отрезков, на которых уравнение (II.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)·F(b) < 0), то уравнение (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 (II.1) имеет на отрезке [a, b] единственный корень, причем функция F(x) на данном отрезке непрерывна.

Разделим отрезок [a, b] пополам точкой c = (a+b)/2. Если $F(x) \neq 0,$ 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.

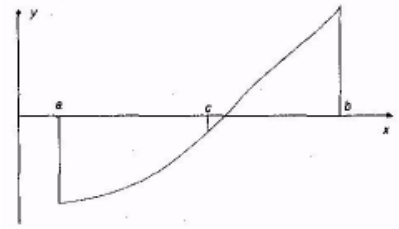


Рис. II.1 К объяснению метода половинного деления

Пример 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 II.1) содержащий описание функции $f(x) = x^4 - 11x^3 + x^2 + x + 0.1.$

Листинг II.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 II.2), содержащий описание функции, возвращающей значение корня уравнения *f(x) = 0* методом половинного деления.

Листинг II.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;
```

3. Постройте график функции на интервале [-1, 1] (рис.II.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:

```
>> Divs (“Func”, x1, x2, 10^-5)
ans =
     0.3942
```

5. Проверьте полученное значение корня:

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

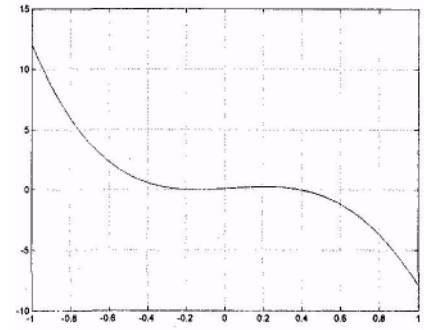



Рис. II.2. График функции $f(x) = x^4 - 11x^3 + x^2 + x + 0.1$.

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 II.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 fevel(f,c,(k))* fevel(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.

>> $[c\ L] = Div21("Func", x1, x2, 10\text{^}{-5});$

7. Визуализируйте зависимость значения корня от номера итерационного процесса. (Рис.II.3).

>> plot (c,"-0")

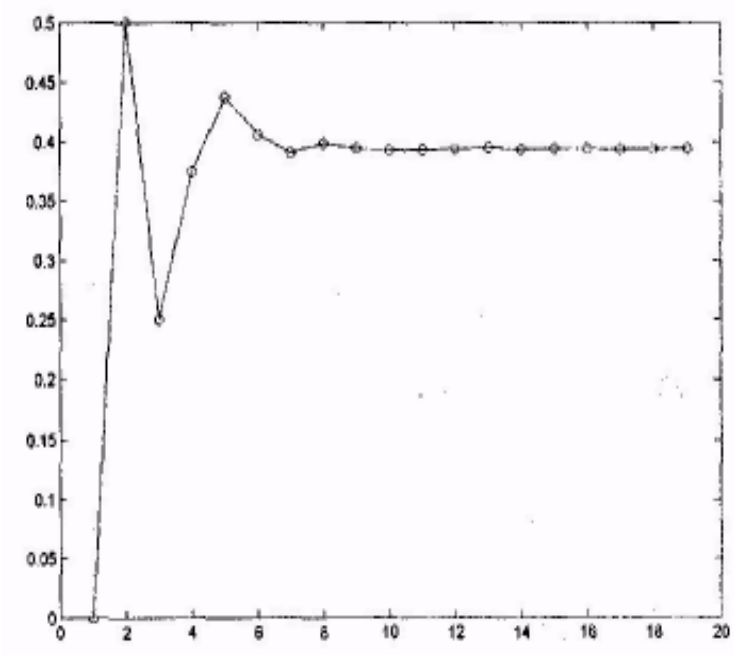


Рис.II.3. Зависимость значения корня от номера шага вычислительной процедуры

8. Визуализируйте зависимость длины отрезка, на котором ищется значение корня, от номера итерации. (Рис.II.4).

>> plot (L,"-0");

2.4. Метод простой итерации

Заменим уравнение (II.1) равносильным уравнением:

$$x = f(x) \qquad \text{(II.4)}$$

Пусть ξ — корень уравнения (II..4), а x_0 — полученное каким-либо способом нулевое приближение к корню ξ. Подставляя x_0 в правую часть уравнения (II.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

$$x_0, x_1 \ldots, x_n \ldots. \qquad \text{(II.5)}$$

называемую итерационной последовательностью.

Геометрическая интерпретация данного алгоритма представлена на рис.II.4. Итерационная последовательность, вообще говоря, может быть как сходящейся, так и расходящейся, что определяется видом функции $f(x)$.

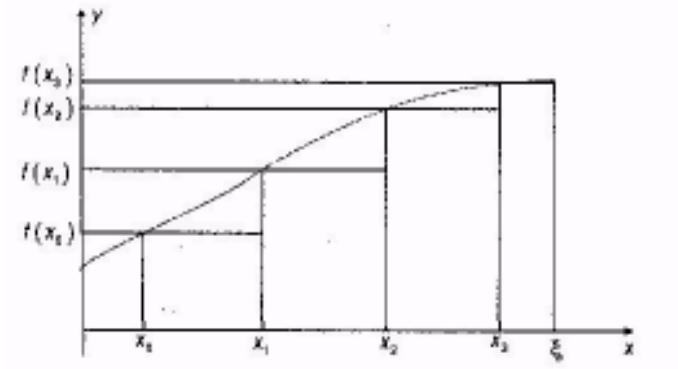


Рис.II.4. К объяснению метода простой итерации

Теорема II.1. Если функция $f(x)$ непрерывна, а последовательность (II.5) сходится, то предел последовательности (II.5) является корнем уравнения (II.4).

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

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} f(x_{n-1}) = f\left(\lim_{n-\infty} x_{n-1}\right) = f(\xi) \qquad \text{(II.6)}$$

Условие сходимости итерационного процесса определяется следующей теоремой.

Теорема II.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 (II.5) с любым начальным значением $x_0 \in [a,b]$. В силу условия 2 теоремы II.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 II.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:

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

$$|x_3 - x_2| \le q|x_2 - x_1| \le q^2|x_1 - x_0|, \qquad \text{(II.7)}$$

Рассмотрим ряд

$$x_0 + (x_1 - x_0) + (x_2 - x_1) + \ldots (x_n - x_{n-1}) + \ldots \quad \text{(II.8)}$$

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 (II.8) совпадает с n-ым членом итерационной последовательности (II.5), т. е.

$$S_{n-1} = x_n \qquad \text{(II.9)}$$

Сравним ряд (II.8) с рядом

$$|x_1 - x_0| + q|x_1 - x_0| + q^2|x_1 - x_0| + \ldots \quad \text{(II.10)}$$

Заметим, что в силу соотношения (II.7) абсолютные величины членов ряда (II.8) не превосходят соответствующих членов ряда (II.10). Но ряд (II.10) сходится как бесконечно убывающая геометрическая прогрессия ($q < 1$, по условию). Следовательно, и ряд (II.8) сходится, т. е. его частичная сумма (II.9)

имеет предел. Пусть $\xi = \lim_{n\to\infty} x_n$. В силу непрерывности функции f получаем (см. (II.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 (II.8) и (II.9), имеем

$$\xi - x_n = \xi - S_{n+1} = (x_{n-1} - x_n) + (x_{n+2} - x_{n-1}) + \dots \quad \text{(II.11)}$$

Сравним (II.11) с остатком ряда (II.9):

$$q^n |x_1 - x_0| + q^{n+1} |x_1 - x_0| + \dots \qquad \text{(II.12)}$$

Учитывая оценку (II.7), получаем

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

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

$$\Delta x_n \le \frac{q^n}{1-q} |x_1 - x_0| \qquad \text{(II.13)}$$

На практике удобнее использовать модификацию формулы (II.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

$$\Delta x_n \le \frac{q}{1-q} |x_n - x_{n-1}| \qquad \text{(II.14)}$$

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:

$$f(x) = x - mF(x) \qquad \text{(II.15)}$$

Функция *f(x)* должна удовлетворять условиям теоремы II.2. Дифференцируя (II.15), получим

$$f'(x) = 1 - mF'(x). \qquad \text{(II.16)}$$

Для выполнения условия 3 теоремы II.2 достаточно подобрать m, так чтобы для всех $x \in [a, b]$

$$|1 - mF'(x)| \le 1 \qquad \text{(II.17)}$$

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

1. Создать файл Func.m (Листинг II.4), содержащий описание функции

$$f(x) = x^4 - 11x^3 + x^2 + x + 0.1$$

Листинг II.4. Файл Func.m

Function $z = Func(x)$

z = x.^4 – 11 * x.^3 + x.^2 + x + 0.1;

2. Создайте файл Funcl.m (листингII.5), содержащий описание функции $f1(x, m, f) := x - mf(x)$.

Листинг II.5. Файл Func1.m

Function $z = Func1(x, m, f)$

z = x – m * Fevel(f, x);

3. Создайте файл Func2.m (листинг II.6), содержащий описание функции $f2(II.16)$.

Листинг II.6. Файл Func2.m

Function z = Func(x,m,f)

dx = 10^–7;

x1 = x + dx

tmpl = x – m * fevel(f, x);

tmp2 = x – m * feval(f, x);

z = abs((tmp2 – tmpl)/ dx);

4. Постройте графики функций f1,f2, (рис.II.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

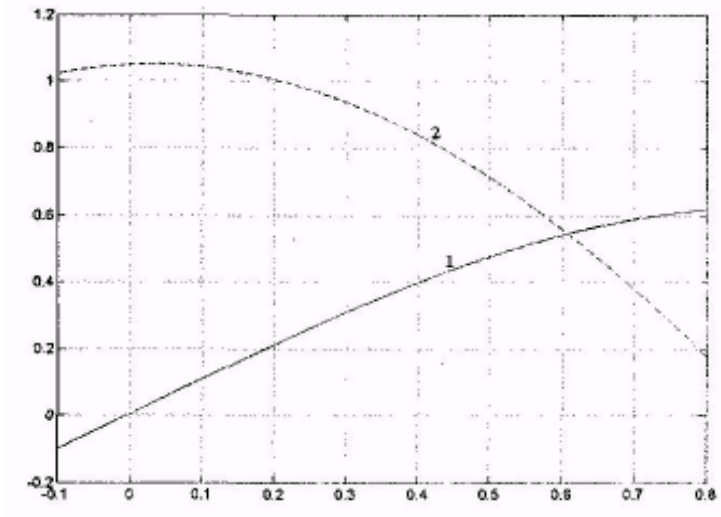


Рис.II.5. График функции $f(x) = x - mf(x) - 1$ и $f'(x) = 1 - mF(x) - 2$

Из рис. II.6 видно, что на интервале [0.21; 0.8] функция удовлетворяет условиям теоремы II.2.

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

Листинг II.7. Функция My_Iter.m

Funrtion $z = My\_Iter(t, x0, eps, q, m)$

$x(1) = x0$;

i=1

$while\ ads(x(i)) - Funcl(x(i), m, f) > q/(1-q) * eps$

$x(i+1) = Funcl(x(i), m, f);$

$i = i + 1;$

end;

$z = x;$

6. Задайте параметры итерационного процесса:

$>> q = 0.01;$

$>> eps = 10\text{^}{-}5;$

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

$>> z = My\_Itel('Func', x0, eps, q, m)$

8. Визуализируйте итерационный процесс (рис. II.7).

$>> plot(z, '-0');$

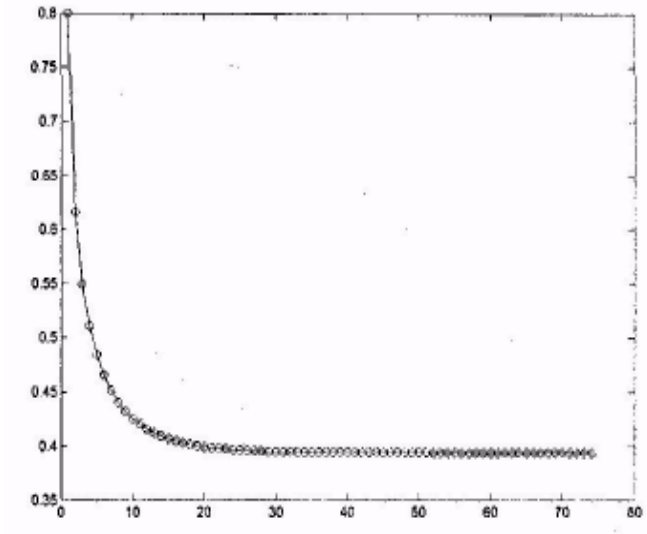


Рис.II.6. Зависимость значения корня уравнения от номера шага итерационного процесса

9. Выведите точное значение корня.

$>> Ni = lenth(z);$

$>> z(Ni)$

ans=

0.3942

10. Выведите значения функции.

>>Func$(z(Ni))$

ans =

-1.8185e-006

Для вычисления нулей функций, зависящих от одной переменной, 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:

$$x = fzero(fun, x0, optlone, P1, P2, \ldots)$$

$$[x, fval, exitflag, output] = fzero(\ldots)$$

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 f zero () 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 II.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:

$>> x = fzero('Func', 0.8, optimset('Tolx', 10^-2, 'disp', 'iter'))$

Funccount	X	£(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	0.86-1	-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
I8      0.437961     -0.157497      search
19      1.16204      -12.8248       search
20      0.288         0.215057      search
Looking for a     in the inter-  [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:

$>>[xfval] = fzero('Func',0.8,optimset('TolX',10^-5,'disp','iter'))$

Funcount	X	f(x)	Proce-
1	0.8	-3.6824	initial
2	0.77737	-3.32063	search
3	0.82262	-4.06625	search
4	0.768	-3.17712	search
5	0.832	-4.23184	search
6	0.75474	-2.98039	search
7	0.84525	-4.47271	search
8	0.736	-2.71444	search
9	0.864	-4.82695	search
10	0.70949	-2.36229	search
11	0.83051	-5.35561	search
12	0.672	-1.9106	search
13	0.928	-6.16014	search
14	0.618981	-1.35979	search
15	0.98101	-7.41582	search
16	0.544	-0.743367	search
17	1.056	-9.43876	search
18	0.43796	-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	interpola-
22	0.467237	-0.288821	interpola-
23	0.369452	0.0698667	interpola-
24	0.398884	0.0148167	interpola-
25	0.393735	0.0013607	interpola-
26	0.394168	,27096e-	interpola-
27	0.394188	,91718e-	interpola-

Zero found in the interval: [0.288, 1.1621.

x -0.3942

fval =

2.2710e-

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

$>> x = fzero('Func',3,optimset('disp','final'))$

zero fond 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 с-вектор строка, содержащая значение коэффициентов полинома:

$$c_1 x^n + c_2 x^{n-1} + \ldots + c_{n+1} x^0$$

ПримерII.4. Найти корень уравнения

$x^4 - 11x^3 + x^2 + x + 0.1 = 0$

используя функцию roots():

$>> c = [1,-11,1,1,01];$

>>roots(c)

ans=

10.8998

0.3942

-0.1470+0.04091

-0.1470-0.04091

III. МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

3.1. Общие сведения и основные определения

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

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &= b_2, \\ a_{m1}x_1 + a_{m2}x_2 + \ldots a_{mn}x_n &= b_m, \end{aligned} \qquad \text{(Ш.1)}$$

которая может быть записана в матричном виде:

$$A \cdot x = b, \qquad \text{(III.2)}$$

где А- прямоугольная матрица размерности $m \times n$;

$$A = \begin{Bmatrix} a_{11}, a_{12}, \ldots a_{1n} \\ a_{21}, a_{22}, \ldots a_{2n} \\ a_{m1}, a_{m2}, \ldots a_{mn} \end{Bmatrix}, \qquad \text{(Ш.3)}$$

x-вектор n—го порядка:

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

b-вектор порядка m:

$$b = \begin{Bmatrix} b_1 \\ b_2 \\ b_m \end{Bmatrix},$$

Определение III.1. Решение системы (III.1) называется такая упорядоченная совокупность чисел

$x_1 = c_1, x_2 = c_2, \ldots, x_n = c_n$, которая обращает все уравнения системы (III.1) в верные равенства.

Определение III.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 III.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

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n &= b_2, \\ a_{n1}x_1 + a_{n2}x_2 + \ldots a_{nn}x_n &= b_n, \end{aligned} \qquad \text{(Ш.4)}$$

при условии, что матрица $A \approx \left(a_{ij}\right)$ невырождена. Метод Гаусса состоит в преобразовании системы (III.4) последовательным исключением переменных к равносильной системе с треугольной матрицей

$$\begin{aligned} x_1 + \alpha_{12}x_n + \ldots \alpha_{1n}x_n &= \beta_1, \\ x_2 + \ldots \alpha_{2n}x_n &= \beta_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots & \\ x_n &= \beta_n, \end{aligned} \qquad \text{(III.5)}$$

Затем из системы (III.5) последовательно находят значения всех неизвестных $x_n, x_{n-l}, \ldots, x_l$.

Таким образом, процесс решения системы (III.4) распадается на два этапа:

1. Прямой ход — приведение системы (III.4) к треугольному виду.
2. Обратный ход — нахождение значений неизвестных переменных, в соответствие с (III.5).

Для реализации метода Гаусса в пакете MAT1AB необходимо:

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

Листинг III.1. Файл Exchenge.m

```
Function z = Exchenge(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 III.2), содержащий описание функции, возвращающей расширенную матрицу системы к диагональному виду.

Листинг III.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 = Exchage(c,i);
    end;
  for j = 0:N
      c(i, N + 1 – j) = c(i, N + 1 – j) / c(i,i);
  end;
  for  m = i + i : N
      alpha=c(m,i);
      for j = i:N+1
          c(m,j)=c(m,i) – 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 III.3), содержащий описание функции, возвращающей решение системы линейных уравнений методом Гаусса.

Листинг III.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	3	16	15

5. Задать вектор-столбец свободных членов:

$>> b = [10;20;30;40;50]$

$b =$

10.000
20.000
30.000
40.000
50.000

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=

10.000
20.000
30.000
40.000
50.000

3.3. Вычисление определителей

Решение системы (III.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:

$$x = \begin{Bmatrix} a_{11}; a_{12}; a_{1n} \\ a_{21}; a_{22}; a_{2n} \\ a_{n1}; a_{n2}; a_{nm} \end{Bmatrix}. \qquad \text{(III.6)}$$

Представим матрицу X в виде:

$$X = Y \cdot Z, \qquad \text{(III.7)}$$

где:

$$Y = \begin{Bmatrix} y_{11}; 0; \ \ldots \ 0 \\ y_{21}; y_{22}; \ldots 0 \\ y_{n1}; y_{n2}; \ldots y_{nm} \end{Bmatrix}, \qquad Z = \begin{Bmatrix} 1; z_{12}; \ldots z_{1n} \\ 0; 1; \ \ldots z_{2n} \\ 0; 0; \ \ldots 1 \end{Bmatrix} \qquad \text{(Ш.8)}$$

Известно, что определитель матрицы равен произведению определителей:

$$|X| = |Y| \cdot |Z|, \qquad \text{(Ш.9)}$$

но $|Z| = 1$, поэтому:

$$|X| = y_{11} \cdot y_{22} \cdot \ldots \cdot y_{nn}, \qquad \text{(III.10)}$$

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

$$y_{ij} = x_{i1}; y_{ij} = x_{ij} - \sum_{k=1}^{j-1} y_{ik} \cdot z_{kj}, \quad \text{при } i \geq j \geq 2, \qquad \text{(III.11)}$$

$$z_{ij} = \frac{x_{1j}}{y_{11}}; \ \ y_{ij} = \left(x_{ij} - \sum_{k=1}^{j-1} y_{ik} \cdot z_{kj}, \right) / y_{ii}, \quad \text{при } 1 < i \leq j, \ \text{(III.12)}$$

Далее приводится листинг файла Determinant.m, содержащий описание функции, которая возвращает значение определителя матрицы, вычисляемого в соответствие с (III.11), (III.12).

Листинг III.4. Файл Determinant.m.

```
Function z = Determinant(A)
P = 1;
C = 1;
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 i = 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) * 2(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, j);
```

```
        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 (III.4) в следующем виде:

$$\begin{aligned} x_1 &= \alpha_{11}x_1 + \alpha_{12}x_2 + \ldots \alpha_{1n}x_n + \beta_1, \\ x_2 &= \alpha_{21}x_1 + \alpha_{22}x_2 + \ldots \alpha_{2n}x_n + \beta_2, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n &= \alpha_{n1}x_1 + \alpha_{n2}x_2 + \ldots \alpha_{nn}x_n + \beta_n, \end{aligned} \qquad \text{(III.13)}$$

или сокращенно:

$$x_i = \sum_{j=1}^{n} \alpha_{ij}x_j + \beta_i, \qquad \text{(III.14)}$$

где $i = 1,2,\ldots,n.$

Правая часть (III.14) определяет отображение F.

$$F : y_i = \sum_{j=1}^{n} \alpha_{ij}x_j + \beta_i, i = 1,2,\ldots n, \qquad \text{(III.15)}$$

преобразующее точку $x = (y_1, y_2, \ldots, y_n)$, n-мерного векторного пространства в точку $y = (y_1, y_2, \ldots, y_n)$того же пространства. Используя систему (III.4) и выбрав начальную точку $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$, можно построить итерационную последовательность точек n-мерного пространства (аналогично методу простой итерации для скалярного уравнения $x = f(x)$).

$$x^{(0)}, x^{(1)}, \ldots, x^{(n)}, \ldots \qquad \text{(III.16)}$$

Оказывается, что при определенных условиях последовательность (III.16) сходится и ее предел является решением системы (III.13). Предваряя обсуждение данных условий, напомним необходимые сведения из математического анализа.

Определение III.4. Функцию *p(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 III.5. Множество с введенной в нем метрикой *p* становится метрическим пространством.

Определение III.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 III.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.

*ОпределениеIII.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:

$$\rho(Fx, Fy) \leq \alpha \cdot \rho(x,y), \qquad \text{(III.17)}$$

Определение III.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.

Теорема III.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

$$\rho\left(x, x^{(k)}\right) \leq \frac{\alpha^k}{1-\alpha} \rho\left(x^{(0)}, x^{(1)}\right), \qquad \text{(III.18)}$$

Приняв k-ое приближение за нулевое, из (III.18) получаем полезное для приложений неравенство:

$$\rho\left(x, x^{(k)}\right) \leq \frac{\alpha}{1-\alpha} \rho\left(x^{(k-1)}, x^{(k)}\right), \qquad \text{(III.19)}$$

Рассмотрим условия, при которых отображение (III.15) будет сжимающим. Как следует из определения (III.17), решение данного вопроса зависит от способа метризации данного пространства. Обычно при решении систем линейных уравнений используют одну из следующих метрик:

$$\rho_1(x,y) = \max_{1 \leq i \leq n} |x_i - y_i|, \qquad \text{(III.20)}$$

$$\rho_2(x,y) = \sum_{i=1}^{n} |x_i - y_i|, \qquad \text{(III.21)}$$

$$\rho_3(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \qquad \text{(III.22)}$$

Для того чтобы отображение *F,* заданное в метрическом пространстве уравнениями (III.15), было сжимающим, достаточно выполнения одного из следующих условий:

1. В пространстве с метрикой ρ_1:

$$\alpha = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |\alpha_{ij}| < 1. \qquad \text{(III.23)}$$

2. В пространстве с метрикой ρ_2 :

$$\alpha = \max_{1 \le j \le n} \sum_{i=1}^{n} \left|\alpha_{ij}\right| < 1. \qquad \text{(III.24)}$$

3. В пространстве с метрикой ρ_3 :

$$\alpha = \sqrt{\sum_{i}^{n} \sum_{j}^{n} \alpha_{ij}^2} < 1. \qquad \text{(III.25)}$$

Для установления момента прекращения итераций при достижении заданной точности может быть использована, например, формула, следующая из (III.19);

$$\rho\left(x^{(k-1)}, x^{(k)}\right) \le \varepsilon(1-\alpha)/\alpha. \qquad \text{(III.26)}$$

Алгоритм решения системы методом итераций реализуется следующим образом:

1. Приведите систему (III.4) к виду с преобладающими диагональными коэффициентами.
2. Разделите каждое уравнение на соответствующий диагональный коэффициент.
3. Проверьте выполнение условий (III.23)—(III.25).
4. Выберите метрику, для которой выполняется условие сходимости итерационного процесса.
5. Реализуйте итерационный процесс (обычно за начальное приближение берется столбец свободных членов).

Пример III.1. Преобразование системы уравнений к виду пригодному для итерационного процесса.

$$2.34x_1 - 4.21x_2 - 11.61x_3 = 14.41$$
$$8.04x_1 + 5.22x_2 + 0.27x_3 = -6.44$$
$$3.92x_1 - 7.99x_2 + 8.37x_3 = 55.56$$

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:

$$x_1 = -0.6492537x_2 - 0.033582x_3 - 0.800995$$
$$x_2 = 0.5131147x_1 - 0.2655737x_3 - 5.7352459$$
$$x_3 = 0.2015503x_1 - 0.36261864x_2 - 1.2411714$$

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

1. Создайте файлы Rol.m, Ro2.m, Ro3.m (листинги III.5-III.7), содержащие описания функций, возвращающих значение расстояния между точками в соответствующем метрическом пространстве.

Листинг III.5. Файл Ro1.m

Function $z = Ro1(x, y)$

$z = \max\left(abs\left(x - y\right)\right);$

Листинг III.6. Файл Ro2.m

Function $z = Ro2(x, y)$

$z = sum\left(abs\left(x - y\right)\right);$

Листинг III.7. Файл Ro3.m

Function $z = Ro3(x, y)$

$z = norm(x - y);$

2. Создайте файл Check.m (листинг III.8), содержащий описание функции, возвращающей значение коэффициентов *a* для каждого типа метрик.

Листинг III.8. Файл Check.m

Function $z = Check(A)$

$alpha1 = sum(ads(A),2);$

$z(1) = \max(alpha1);$

$alpha2 = sum(ads(A),1);$

$z(2) = \max(alpha2);$

$alpha3 = A.\hat{}2;$

$z(3) = sum(sum(alpha3))\hat{}0.5;$

3. Создайте файл LS_Iter.m (листинг III.9), содержащий описание функции, возвращающей решение системы линейных уравнений методом простой итерации.

Листинг III.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 выбранной метрикой

% eps - переменная, определяющая точность численного
% решения.

$delta = esp * (1 - aipha) / alpha;$

$N = size(A,1);$

$x0 = zeroz(N,1);$

$x1 = A * x0 + b;$

$Delta - feval(Ro, x0, x1);$

$while \quad Delta > delta$

$x0 = x1;$

$x1 = A * x0 + b;$

$Delta = feval(Ro, x0, x1);$

$end;$

$z = x1;$

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 *a* для каждого метрического пространства:

» alpha==Check (A)
alpha=
0.7787 1.0119 0.9636
7. Найдите решения системы линейных уравнений:

$>> LS\_Iter\left(A, b, 'Rol', ans(1), 10\wedge -6\right)$

$ans =$

2.2930
– 4.8155
0.9672

3.5. Метод Зейделя

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

$$y_i = \sum_{j=1}^{n} \alpha_{ij} x_j + \beta_i \; ; \qquad \text{(III.27)}$$

где: i=1,2,…..,n.

Из (III.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 $y_1, y_2, \ldots, y_{t-1}$:

$$\begin{aligned}
&y_1 = \sum_{j=1}^{n} \alpha_{1j} x_j + \beta_i, \\
&y_2 = \alpha_{21} y_1 + \sum_{j=2}^{n} \alpha_{2j} x_j + \beta_2, \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \qquad \text{(III.28)} \\
&y_i = \sum_{j=1}^{i\to 1} \alpha_{ij} y_j + \sum_{j=i}^{n} \alpha_{ij} x_j + \beta_i, \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
&y_n = \sum_{j=1}^{n\to 1} \alpha_{nj} y_j + \alpha_{nn} x_n + \beta_n.
\end{aligned}$$

Достаточные условия сходимости итерационного процесса (III.23)-(III.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 (III.2) на транспонированную матрицу системы A^T, получим равносильную систему:

$$C \cdot X = D \qquad \text{(III.29)}$$

где $C = A^T A, \quad D = A^T b.$

Система (III.29) называется нормальной системой уравнений. Нормальные системы уравнений обладают рядом свойств, среди которых можно выделить следующие:

- Матрица С коэффициентов при неизвестных нормальной системы является симметричной (т.е. $a_{ij} = a_{ji}, i, j = 1,2,\ldots,n$)

- Все элементы, стоящие на главной диагонали матрицы С, положительны: (т.е $a_{ij} > 0, i = 1,2,\ldots,n$).

Последнее свойство дает возможность “автоматически” приводить нормальную систему (III.29) к виду, пригодному для

итерационного процесса Зейделя:

$$x_i = \sum_{i \neq j} a_{ij} x_j + \beta_j, (i = 1,2,\ldots,n), \quad (III.30)$$

$$a_{ij} = -\frac{C_{ij}}{C_{ii}} (i \neq j), \quad (III.31)$$

$$\beta_i = \frac{d_i}{C_{ii}}, \quad (III.32)$$

Целесообразность привидения системы к нормальному виду и использование метода Зейделя вытекает из следующей теоремы:

Теорема III.2 Интерационный процесс метода Зейделя для приведенной системы (III.30), эквивалентной нормальной системе (III.29), всегда сходится к единственному решению этой системы при любом выборе начального приближения.

Таким образом, решение произвольной системы линейных уравнений вида (III.1) методом Зейделя реализуется в соответствии со следующим алгоритмом:

1. Ввод матрицы А 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 (III.30), (III.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 (листингIII.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 III.10. Файл Zeidel.m

```
function [zl,z2]=Zeidel(A,b,eps)
N=size(A,l)
% приведение системы к нормальному виду
C=A"A;
D=A'*b,
% приведение системы к виду, пригодному для итерационного
% процесса Зейделя
for i=l: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(ads(R2 – R1));
if _ S < eps
z1 = d1;
z2 = s;
return
end;
R1 = R2;
end;
```

2. Задать значение коэффициентов при неизвестных исходной системы линейных уравнений и столбец исходных 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( ).

```
>> zeidel(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 (/), деления слева (\), возведение в степень (^), транспонирование (‘), 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 ZeidelO, 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.

IV. МЕТОДЫ РЕШЕНИЯ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

На примере метода Ньютона и метода спуска продемонстрируем основные подходы к решению задачи нахождения корней системы нелинейных уравнений, опишем соответствующие алгоритмы и их программные реализации, а также обсудим использование соответствующих функций пакета MATLAB.

4.1. Векторная запись нелинейных систем. Метод простых итераций

Пусть требуется решить систему уравнений

$$\begin{cases} f_1(x_1, x_2, \dots x_n) = 0, \\ f_2(x_1, x_2, \dots x_n) = 0, \\ \dots \quad \dots \quad \dots \quad \dots \\ f_n(x_1, x_2, \dots x_n) = 0, \end{cases} \qquad \text{(IV.1)}$$

где $f_1.f_2...,f_n$ - заданные, вообще говоря, нелинейные вещественнозначные функции n вещественных переменных $x_1,x_2.....x_n$.
Введем обозначения:

$$\bar{x} \equiv \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_3 \end{pmatrix}, \quad F(\bar{x}) \equiv \begin{pmatrix} f_1(\bar{x}) \\ f_2(\bar{x}) \\ \vdots \\ f_n(\bar{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1, x_2, \dots x_n) \\ f_2(x_1, x_2, \dots x_n) \\ \vdots \\ f_n(x_1, x_2, \dots x_n) \end{pmatrix}, \quad \bar{0} \equiv \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Тогда систему (IV.1) можно заменить одним уравнением

$$F(\bar{x}) = \bar{0} \qquad \text{(IV.2)}$$

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 главе 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 ≥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 (IV.1) преобразована к следующей эквивалентной нелинейной системе:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n), \\ x_2 = \varphi_2(x_1, x_2, \dots x_n), \\ \qquad \dots \\ x_n = \varphi(x_1, x_2, \dots x_n) \end{cases} \qquad \text{(IV.3)}$$

или в компактной записи:

$$\bar{x} = \Phi(\bar{x}) = \begin{pmatrix} \varphi_1(\bar{x}) \\ \varphi_2(\bar{x}) \\ \dots \\ \varphi_n(\bar{x}) \end{pmatrix} = \begin{pmatrix} \varphi_1(x_1, x_2, \dots, x_n) \\ \varphi_2(x_1, x_2, \dots, x_n) \\ \dots \\ \varphi_n(x_1, x_2, \dots, x_n) \end{pmatrix}. \qquad \text{(IV.4)}$$

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

$$\bar{x}^{(k+1)} = \Phi(\bar{x}^{(k)}), \qquad \text{(IV.5)}$$

где k определяет метод простых итераций, для задачи (IV.3) и k= 0,1,2,...,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 IV.5), то при определенных условиях данная последовательность со скоростью геометрической прогрессии будет приближаться к вектору $\bar{x}^*$ — неподвижной точке отображения Φ(x).

Справедлива следующая теорема, которую мы приводим без доказательства.

Теорема IV.1. Пусть функция Φ(x) и замкнутое множество $\mathrm{M} \subseteq D(\Phi) \subseteq R_n$ таковы, что:

1. $\Phi(x) \in \mathrm{M}, \quad \forall \bar{x} \in \mathrm{M}.$

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

Тогда Φ(x) имеет в M единственную неподвижную точку $\bar{x}^*$; последовательность $\left(\bar{x}^{(k)}\right)$, определяемая (IV.5), сходится при любом $\bar{x}^{(0)} \in \mathrm{M}$ к $\bar{x}^*$; справедливы оценки:

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

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 IV.2. Пусть $\Phi(x)$ дифференцируема в замкнутом шаре $S(x^{(0)}, r) \subseteq D(\Phi)$, причем $\exists q \in (0;1)$: $\sup\limits_{x \in S}\left\|\Phi'(\bar{x})\right\| \le q$. Тогда если центр $\bar{x}^{(0)}$ и радиус r шара S таковы, что $\left\|\bar{x}^{(0)} - \Phi(\bar{x}^{(0)})\right\| \le r(1-q)$, то справедливо заключение теоремы IV.1 с M = S .

Запишем метод последовательных приближений (IV.5) в развернутом виде:

$$\begin{cases} x_1^{(k+1)} = \varphi_1\left(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}\right), \\ x_2^{(k+1)} = \varphi_2\left(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}\right), \\ \dots \\ x_n^{(k+1)} = \varphi_n\left(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}\right). \end{cases} \qquad \text{(IV.6)}$$

Сравнение (IV.6) с вычислительной формулой метода простой итерации решения систем линейных уравнений (III.13) обнаруживает их сходство. Учитывая, что в линейном случае, как правило, более эффективен метод Зейделя, а данном случае также может оказаться более эффективным его многомерный аналог, называемый методом покоординатных итераций:

$$\begin{cases} x_1^{(k+1)} = \varphi_1\left(x_1^{(k)}, x_2^{(k)}, \dots, x_{n-1}^{(k)}, x_n^{(k)}\right), \\ x_2^{(k+1)} = \varphi_2\left(x_1^{(k+1)}, x_2^{(k)}, \dots, x_{n-1}^{(k)}, x_n^{(k)}\right), \\ \dots \\ x_n^{(k+1)} = \varphi_n\left(x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{n-1}^{(k+1)}, x_n^{(k)}\right). \end{cases} \qquad \text{(IV.7)}$$

Заметим, что, как и для линейных систем, отдельные уравнения в (IV.7) неравноправны, т, е. перемена местами уравнений системы (IV.3) может изменить в некоторых пределах число итераций и вообще ситуацию со сходимостью последовательности итераций. Для того чтобы применить метод простых итераций (IV.6) или его "зейделеву" модификацию (IV.7) к исходной системе (IV.1), необходимо сначала тем или иным способом привести эту систему к виду (IV.3). Это можно сделать, например, умножив (IV.2) на неособенную nxn матрицу A и прибавив к обеим частям уравнения $-\mathrm{A}\cdot F(\bar{x})$ вектор неизвестных. Полученная система

$$\bar{x} = \bar{x} - \mathrm{A}\cdot F(\bar{x}) \qquad \text{(IV.8)}$$

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 (IV.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 (IV.2).

Тогда точный корень уравнения (IV.2) можно представить в виде:

$$\bar{x}^{*} = \bar{x}^{(k)} + \Delta\bar{x}^{(k)}, \qquad \text{(IV.9)}$$

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 (IV.9) в (IV.2), имеем

$$F\left(\bar{x}^{(k)} + \Delta\bar{x}^{(k)}\right) = 0. \qquad \text{(IV.10)}$$

Предполагая, что функция $F(\bar{x})$ непрерывно дифференцируема в некоторой выпуклой области, содержащей $\bar{x}$ и $\bar{x}^{(k)}$, разложим левую часть уравнения (IV.10) по степеням малого вектора $\Delta\bar{x}^{(k)}$, ограничиваясь линейными членами:

$$F\left(\bar{x}^{(k)} + \Delta\bar{x}^{(k)}\right) = F\left(\bar{x}^{(k)}\right) + F'\left(\bar{x}^{k}\right)\Delta\bar{x}^{(k)} \qquad \text{(IV.11)}$$

или в развернутом виде :

$$\begin{cases} f_1\left(x_1^{(k)} + \Delta x_1^{(k)}, \ldots, x_n^{(k)} + \Delta x_n^{(k)}\right) = f_1\left(x_1^{(k)}, \ldots, x_n^{(k)}\right) + \cdots \\ \qquad \cdots + \Delta x_1^{(k)}\dfrac{\partial f_1}{\partial x_1} + \Delta x_n^{(k)}\dfrac{\partial f_1}{\partial x_n} = 0 \\ \qquad\qquad \ldots\ldots\ldots \quad \ldots\ldots\ldots\ldots \quad \ldots\ldots\ldots\ldots \quad \ldots\ldots \\ f_n\left(x_1^{(k)} + \Delta x_1^{(k)}, \ldots, x_n^{(k)} + \Delta x_n^{(k)}\right) = f_n\left(x_1^{(k)}, \ldots, x_n^{(k)}\right) + \cdots \\ \qquad \cdots + \Delta x_1^{(k)}\dfrac{\partial f_n}{\partial x_1} + \Delta x_n^{(k)}\dfrac{\partial f_n}{\partial x_n} = 0. \end{cases} \qquad \text{(IV.12)}$$

Из формул (IV.11) и (IV.12) видно, что под производной $F'(\bar{x})$ следует понимать матрицу Якоби системы функций $f_1, f_2, \ldots, f_n$ относительно переменных $x_1, x_2, \ldots x_n$, т. е.

$$F'(\bar{x}) = W(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \dots \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \dots \frac{\partial f_2}{\partial x_n} \\ \dots \quad \dots \quad \dots \quad \dots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} \dots \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

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 (IV.12) может быть записана в следующем виде:

$$F(\bar{x}^{(k)}) + W(\bar{x}^{(k)})\Delta\bar{x}^{(k)} = 0. \qquad \text{(IV.13)}$$

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

$$\Delta\bar{x}^{(k)} = -W^{-1}(\bar{x}^{(k)})F(\bar{x}^{(k)}). \qquad \text{(IV.14)}$$

Отсюда видно, что метод Ньютона для решения системы (IV.1) состоит в построении итерационной последовательности:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - W^{-1}(\bar{x}^{(k)})F(\bar{x}^{(k)}), \qquad \text{(IV.15)}$$

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 IV.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

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

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

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

Причем

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

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

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

По формуле (IV.15) получаем первое приближение:

$$\bar{x}^{(1)}=\bar{x}^{(0)}-W^{-1}(\bar{x}^{(0)})F(\bar{x}^{(0)})=\begin{bmatrix}0.5\\ 0.5\\ 0.5\end{bmatrix}-\begin{bmatrix}\frac{3}{8} & \frac{1}{8} & \frac{1}{8}\\ \frac{7}{20} & \frac{1}{20} & -\frac{3}{20}\\ \frac{11}{40} & -\frac{7}{40} & \frac{1}{40}\end{bmatrix}\cdot\begin{bmatrix}-0.25\\ -1.25\\ -1.00\end{bmatrix}=$$

$$=\begin{bmatrix}0.5\\ 0.5\\ 0.5\end{bmatrix}+\begin{bmatrix}0.375\\ 0\\ -0.125\end{bmatrix}=\begin{bmatrix}0.875\\ 0.500\\ 0.375\end{bmatrix}.$$

Аналогично находятся дальнейшие приближения. Результаты вычислений приведены в таблице.

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; г = 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 само-

стоятельную 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

$$\begin{cases} f(x,y)=0 \\ g(x,y)=0 \end{cases} \qquad \text{(IV.16)}$$

Из функций f(x,y), g(x,y) системы (IV.16) образуем новую функцию.

$$\Phi(x,y)=f(x,y)^2+g(x,y)^2. \quad \text{(IV.17)}$$

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

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

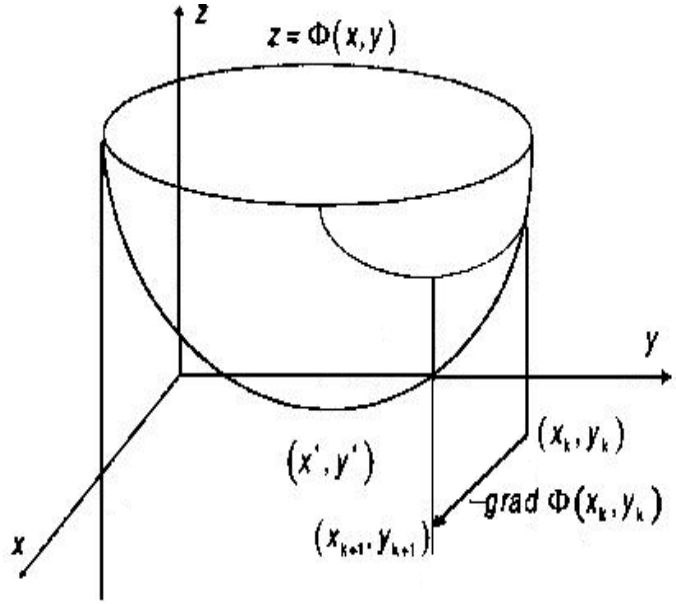


Рис. IV.1. Пространственная интерпретация метода наискорейшего спуска для функции (IV.17)

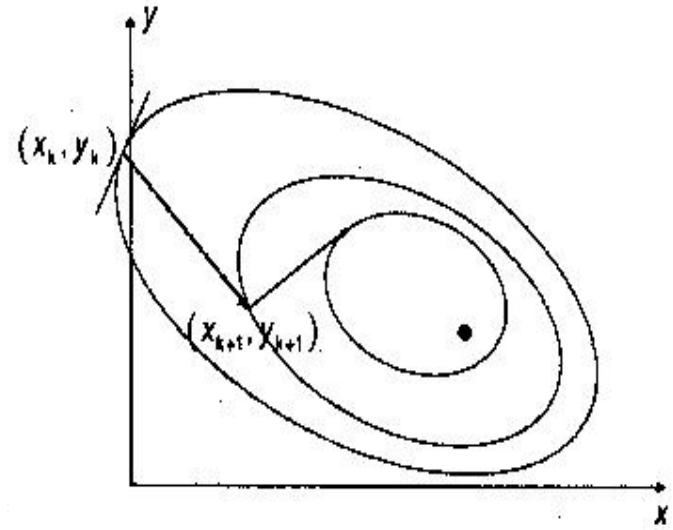


Рис. IV.2. Траектория наискорейшего спуска для функции (IV.17) в плоскости XOУ

Следовательно, если тем или иным способом удается получить точку (x*,y*), минимизирующую функцию Φ(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 (IV.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 Φ(x,y) — обычно получают по рекуррентной формуле

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \alpha_k \begin{pmatrix} p_k \\ q_k \end{pmatrix}, \qquad \text{(IV.18)}$$

где k =0,1,2,..., $(p_k,q_k)^r$ - вектор, определяющий направление минимизации, а α_k— скалярная величина, характеризующая величину шага минимизации (шаговый множитель). Учитывая геометрический смысл задачи минимизации функций двух переменных Φ(x, y) — "спуск на дно" поверхности z = Φ(x:,y) (рис. IV.1), итерационный метод (IV.18) можно назвать методом спуска, если вектор $(p_k,q_k)^T$ при каждом k является направлением спуска (т. е. существует такое a >0, что $\Phi(x_k + ap_k, y_k + \alpha q_k) < \Phi(x_k,y_k)$), и если множитель a_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 (IV.18) минимизации функции Φ(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 a_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

$$\begin{pmatrix} p_k \\ q_k \end{pmatrix} = -\overline{\nabla}\Phi(x_{k,}, y_k) = -\begin{pmatrix} \Phi'_x(x_k, y_k) \\ \Phi'_y(x_k, y_k) \end{pmatrix} \qquad \text{(IV.19)}$$

— антиградиент функции Φ(x,y). Таким образом, из семейств (IV.18) выделяем градиентный метод.

Оптимальный шаг в направлении антиградиента — это такой шаг, при котором значение $\Phi(x_{k+1},y_{k+1})$ наименьшее среди всех других значений Φ(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 (IV.19), если полагать в нем

$$a_k = \arg\min_{a>0} \Phi\left(x_k - a\Phi'_x(x_k, y_k),\ y_k - a\Phi'_y(x_k, y_k)\right). \qquad \text{(IV.20)}$$

Такой выбор шагового множителя, называемый исчерпывающим спуском, вместе с формулой (IV.19) определяет метод наискорейшего спуска.

Геометрическая интерпретация этого метода хорошо видна на рис. IV.1, IV.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 a_k не удается. Следовательно, приходится делать ставку на применение каких-либо численных методов одномерной минимизации и находить a_k в (IV.18) лишь приближенно.

Несмотря на то, что задача нахождения минимума функции одной переменной $\varphi_k(a) = \Phi\left(x_k - a\Phi'_x(x_k, y_k), y_k - a\Phi'_y(x_k, y_k)\right)$ намного проще, чем решаемая задача, применение тех или иных численных методов нахождения значений $a_k = \arg\min \varphi_k(a)$ 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 a_k, в которых учитывается специфика минимизируемых функций (типа сумм квадратов функций).

Зачастую успешной является такая стратегия градиентного метода, при которой шаговый множитель a_k в (IV.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 $|x^* - x_{k+1}| \le C|x^* - x_k|^p$, 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 квазиньютоновскими, 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.

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.

Для решения n-мерной системы (IV.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 IV.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 IV.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=l:N
  for  j=l: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=l:N;j=l:N;
» x(i)=Xmin+i*(Xmax-Xmin)/N;
» y(j)=Ymin+j*(Ymax-Ymin)/N;
```



```
» Ml=F_L4(x,y,0.5); M2=F_L4(x,y,l) ;M3=F_L4(x,y, 1.5) ;
» [X Y]=meshgrid(x,y);
» surfc(X,Y,Ml);  colormap gray
» surfc(X,Y,M2);  colorap gray
» surfc(X,Y,M3);  colormap gray
```

Из рис. IV.3-IV.5 видно, что при μ = 1 функция f(x,y,μ) представляет собой параболоид вращения, при μ>1 параболоид становится эллиптическим, "вытягиваясь" вдоль оси oX (при μ < 1 — вдоль оси oY).

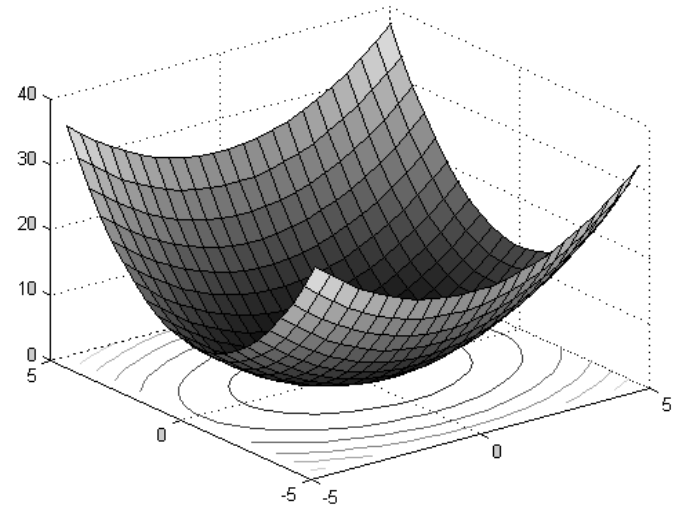


Рис. IV.3. Поверхность и карта линий уровня функции $f(x, y) = x^2 + 0.5 \cdot y^2$

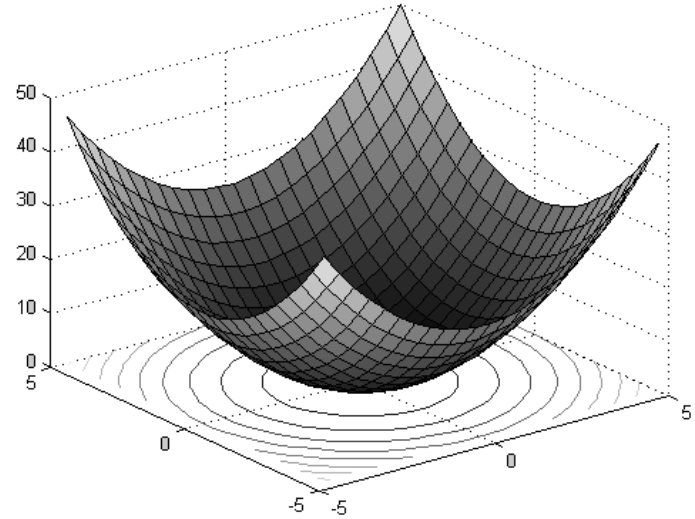


Рис. IV.4. Поверхность и карта линий уровня функции

$$f(x,y) = x^2 + y^2$$

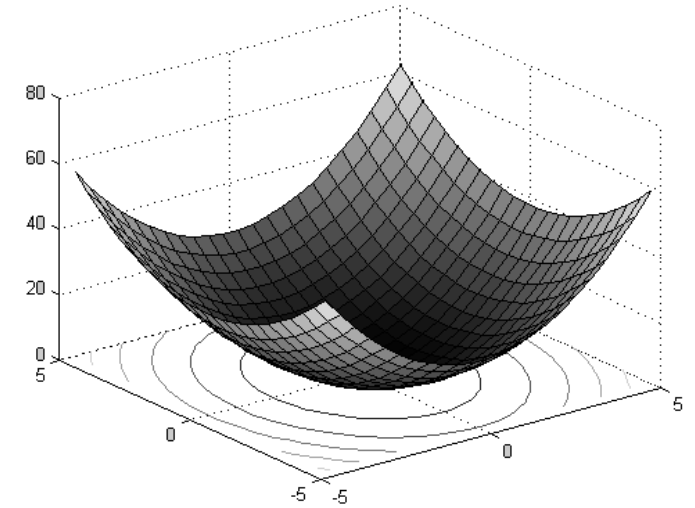


Рис. IV.5. Поверхность и карта линий уровня функции

$$f(x,y) = x^2 + 1.5 \cdot y^2$$

3. Создайте файл Dx_F_L4.m (листинг IV.2), содержащий описание функции, возвращающей значения частной производной, по переменной x.

Листинг IV.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=l:N
    for j=l:N
       z(i, j)=2*x(i);
       end;
end;
```

4. Создайте файл DyJF_L4.m (листинг IV.3), содержащий описание функции, возвращающей значения частной производной функции f(x,y,μ.) по переменной y.

Листинг IV.3. Файл DyJF_L4.m

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

5.Создайте файл L_Grad.m (листинг IV.4), содержащий описание функции, возвращающей длину градиента функции (x,y,μ).

Листинг IV.4. Файл L_Grad.m

```
function z=L_Grad(x,y,mu)
```

```
N=length(x) ;
z=zeros(N);
for i=l:N
  for j=l: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 IV.5), содержащий описание функции, возвращающей значение проекции на ось oX нормированного единичного вектора, противоположно направленного вектору градиента.

Листинг IV.5. Файл S_x.m

```
function z=S_x(x,y,mu);
N=length(x);
z=zeros(N) ;
i=l:N;
j=l: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 IV.6), содержащий описание функции, возвращающей значение проекции на ось o Y нормированного единичного вектора, противоположно направленного вектору градиента.

Листинг IV.6. Файл S_y.m

```
function z=S_y(x,y,rnu) ;
N=length(x) ;
z=zeros(N);
for i=l:N
```



```
    for j=l: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 IV.7), содержащий описание функции, возвращающей значение шага.

Листинг IV.7. Файл Lambda.m

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

9. Создайте файл MG.m (листинг IV.8), содержащий описание функции, возвращающей значения переменных x, y и соответствующее значение функции f(x,y,μ) на каждом шаге итерационного процесса.

Листинг IV.8. Файл MG.m

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

10. Найдите минимум функции f(x,y,μ) 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=l; alpha=l; garama=l; % 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. IV.6)
>> hold off;
>> plot(ff);  % зависимость значения функции f(x,y) от номера
              % итерации (рис. IV.7)
>> stairs(X,Y); % построение траектории итерационного
                % процесс;
                % на плоскости XoY (рис. IV.8) .
>> plot3(X,Y,ff) % построение траектории итерационного
             % процесса Б трехмерном пространстве(рис. IV.9)
```

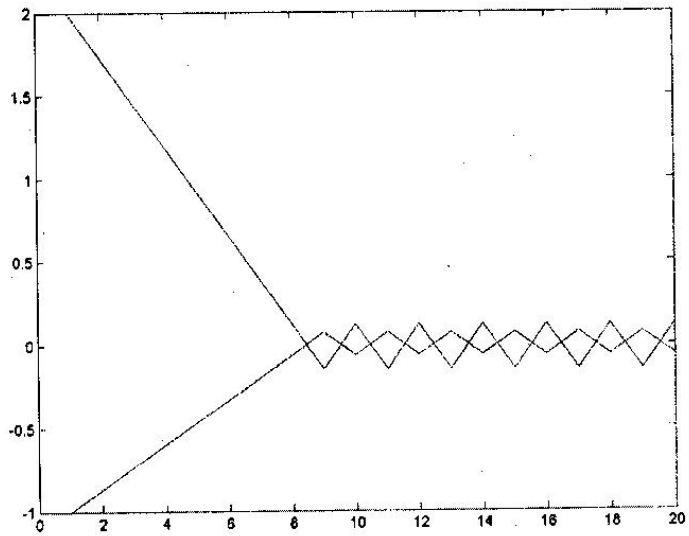


Рис. IV.6. Траектория решения системы нелинейных уравнений на плоскости XoY

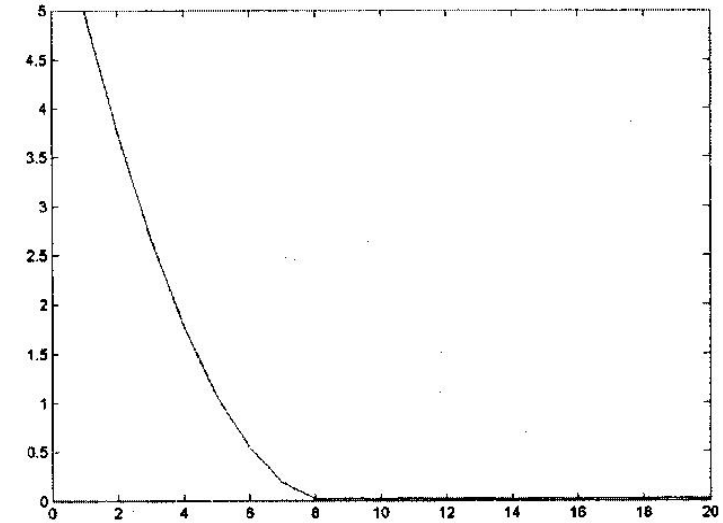


Рис. IV.7. Зависимость значения минимума функции от номера итерации

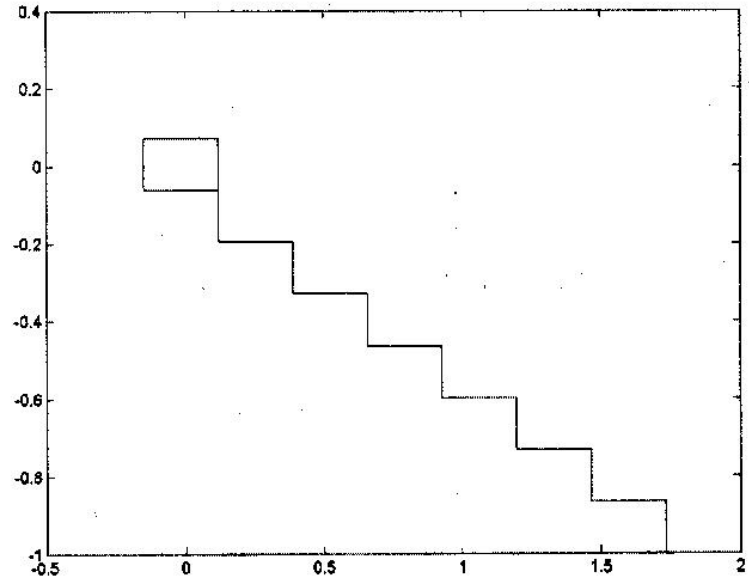


Рис. IV.8. Траектория итерационного процесса на плоскости XoY

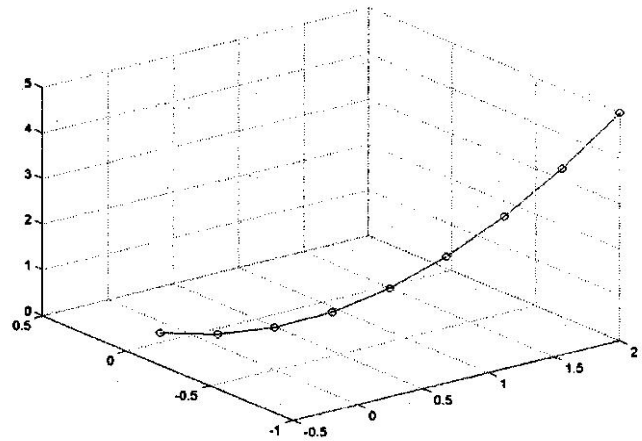


Рис. IV.9. Визуализация итерационного процесса в трехмерном пространстве

Пример IV.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:

$$f(x,y) = k_x x^4 + k_y y^4 + l_x x^3 + l_y y^3 + o_x x^2 + o_y y^2 + p_x x + p_y y + q +$$

$$k_{xy} x^4 y^4 + \cdots + l_{xy} x^3 y^3 + o_{xy} x^2 y^2 + p_{xy} xy$$

Форма функции определяется коэффициентами k_x, k_y, l_x, l_y ит.д.

1. Создайте файл F2_L4.m (листинг IV.9), содержащий описание функции, возвращающей значения функции/(x, y).

Листинг 4.9. Файл F2_L4.m

```
function z=F2_L4(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,O,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=l:N
    for j=l:N
z(i,j)=Kx*x(i) .^4+Ky*y(j) .^4+Lx*x(i) .^3+Ly*y(j) .^3+...
            Ox*x(i) .^2+Qy*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;0y=2;Py=0.6;Kxy=0.2;Pxy=0.1;
» Xmin=-l;Xmax=l;
» Ymin=-l;Ymax=l;
» N=23;
» x(i)=Xmin+(Xmax-Xmin) / (N-l) * (i-1) ;
» y(j)=Ymin+(Ymax-Ymin)/ (N-l)*(j-l) ;
» 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,z)
```

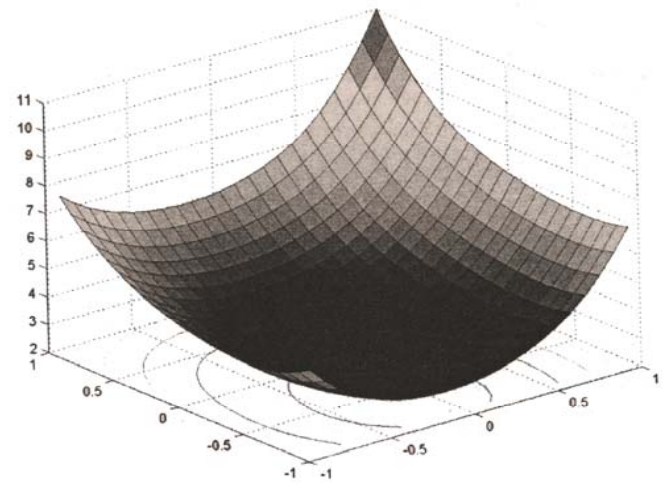


Рис. IV.10. Визуализация функции f (x, y)

— График функции, изображенной на рис. IV.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 IV.10) и g2_y.m (листинг IV.11), содержащие описание функций, возвращающих значения частных производных.

Листинг IV.10. Файлы 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=zero9(N);
for  i=l:N
 for  j=l: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*0xy*x(i) .*y(j) .^2+Pxy*y(j);
  end;
end;
```

Листинг IV.11. Файлы 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=l:N
  for  j=l: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 IV.12), содержащий описание скалярной функции, возвращающей значение длины градиента функции f(x,y).

Листинг IV.12. Файл L2_Grad.m

```
function z=L2_Grad(x,y,Lx,Ly,Qx,Px,Lxy,Oxy,Q,Kx,Ky,Qy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=l:N
  for  j=l:
z(i,j)= L2_Grad (x(i) ,y(j) ,Lx,Ly,Ox, Px,Lxy,Oxy,Q,Kx,Ky,Oy,Py,...
            Kxy.Pxy).^2+...
        L2_Grad (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 (листинг IV.13) и S2_y.m (листинг IV.14), содержащие описание функций, возвращающих координаты нормированного единичного вектора, сонаправленного с вектором, противоположным направлению вектора градиента.

Листинг IV.13. Файлы 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=l:N;
j = l: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 IV.14. Файлы S2_y.m

```
function    z=S2_y(x,y,Lx, Ly,0x, Px,Lxy,Oxy,Q,Kx,Ky, Oy, Py,Kxy, Pxy);
N=length(x) ;
z=zeros(N);
i=1:N;
j=1:N;
for i=l:N
       for j=l: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 IV.15), g2_xy.m (листинг IV.16), g2_yy.m (листинг IV.17), содержащие описание функций, возвращающих значения производных $f''_{xx}(x,y)$, $f''_{xy}(x,y)$, $f''_{yy}(x,y)$, соответственно.

Листинг IV.15. Файлы 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=l:N
  for j=l: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 IV.16. Файлы 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=l: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 IV.17. Файлы g2_yy.m

```
function
z=g2_yy(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,C,Kx,Ky,Oy,Py,Kxy,Pxy)
N=length(x);
z=zeros(M);
for  i=1:N
for j=l:N
    z(i,j)=l2*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 IV.18), содержащий описание функции, возвращающей значения коэффициента λ.

Листинг IV.18. Файл Lambda2.m

```
function z=Lambda2(x,y,Lx,Ly,Ox,Px,Lxy,Oxy,Q.,Kx,Ky,Qy,Py,Kxy,Pxy)
N=length(x);
z=zeros(N);
for i=l:N
  for j=l:N
al=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,Cy,Fy,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,Qy,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,BOty.Pty);
```

```
   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 IV.19), содержащий описание функции, возвращающей значения переменных x, y и соответствующее значение функции f(x,y,μ) на каждом шаге итерационного процесса.

Листинг IV.19. Файл MG2.m

```
function [X,Y,ff]=MG(xO,yO,nu,Lx,Ly,Ox,Px,Lxy,Oxy,...
….Q,Kx,Ky,Cy,Py,Kxy,Pxy)
X(1)=x0;
Y(l)=y0;
ff(l)=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-l)+Lambda2(X(i-l),Y{i-l),Lx,Ly,Ox, Px,Lxy,Oxy,Q,._
        Kx,Ky,Oy,Py,Kxy,Pxy)*s2_x(X(i-l),Y(i-l),Lx,Ly,Ox,Px,Lxy,Oxy,
                                Q,Kx,Ky,Oy,Py,Kxy,Pxy),
     Y(i)=Y(i-l)+Lambda2(X(i-l),Y(i-l),Lx,Ly,Ox, Px,Lxy,Oxy,Q,Kx,...
       Ky,Oy,Py,Kxy,Pxy)*s2_y(X(i-l),Y(i-l),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,μ) и визуализируйте итерационный процесс.

```
» xO=1;yO=0.5;  % начальное приложение
» nu«10;  % число шагов итерационного процесса
»[X У ff]=MG2(xO,yO,nu,Lx,Ly,Ox,Px,Lxy,Oxy,Q,Kx,Ky,Oy,PY, Kxy.Pxy);
» plot(X);  hold on;  plot(Y);  hold off % рис.  IV.11
» plot(ff);  % рис.  IV.12
» stairs(X,Y);  % рис.  IV.13
» plot3(X,Y,ff);  % рис.  IV.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. IV.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.

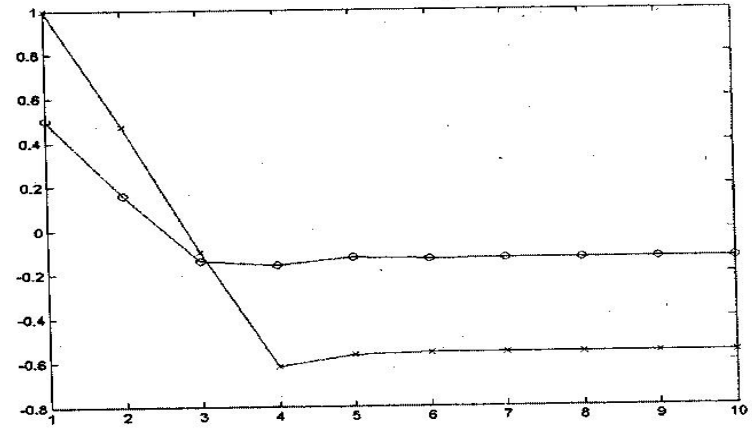


Рис. IV.11. Зависимость координат точки решения от номера итерации

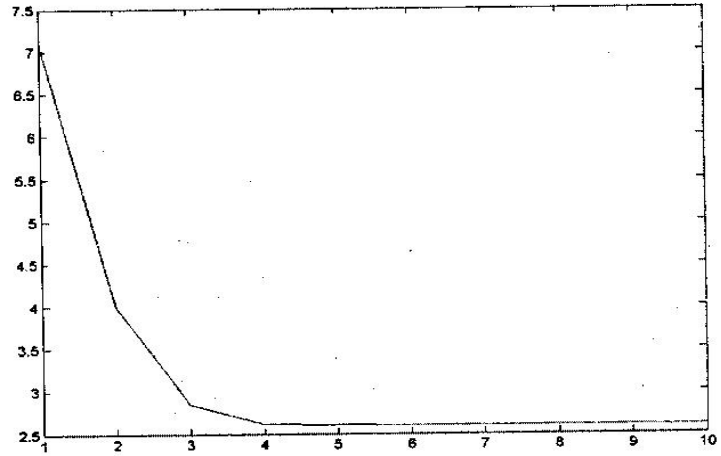


Рис. IV.12. Зависимость значения исследуемой функции от номера итерации

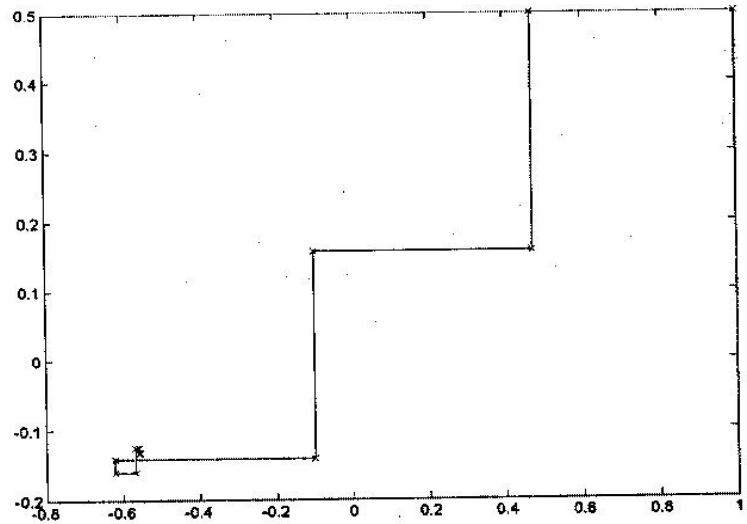


Рис. IV.13. Траектории итерационного процесса в плоскости XoY

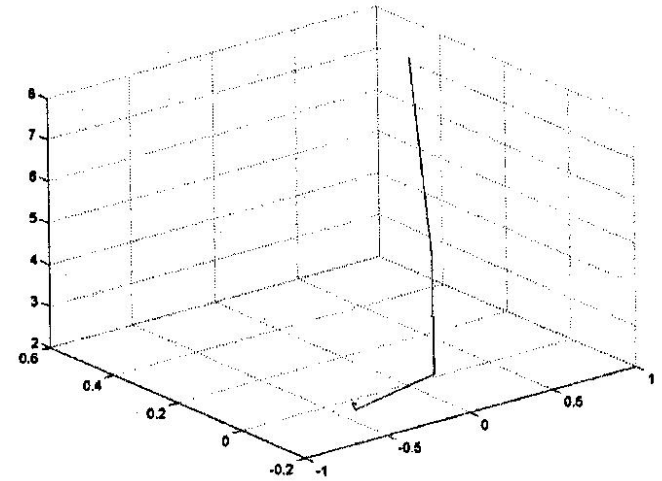


Рис. IV.14. Траектория итерационного процесса в пространстве

4.4. 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

$$F_{sq}\left(\bar{x}^{(k+1)}\right)=F\left(\bar{x}^{(k)}\right)+\left[\nabla f\left(\bar{x}^{(k)}\right)\right]^T\cdot\Delta\bar{x}^{(k)}+\ldots+$$
$$+\ldots+(1/2!)\cdot\left[\Delta\bar{x}^{(k)}\right]^T\cdot\vec{\nabla}^2F\left(\bar{x}^{(k)}\right)\cdot\Delta\bar{x}^{(k)}. \qquad \text{(IV.21)}$$

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

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

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 $\Delta\bar{x}^{(k)}$:

$$\nabla F_{sq}\left(\bar{x}^{(k+1)}\right)=\nabla F_{sq}\left(\bar{x}^{(k)}\right)+\vec{\nabla}^2F_{sq}\left(\bar{x}^{(k)}\right)\Delta\bar{x}^{(k)}=0. \qquad \text{(IV.22)}$$

Для учета фактических особенностей минимизируемой функции будем использовать в (IV.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 $\Delta\bar{x}^{(k)}$:

$$\Delta\bar{x}^{(k)}=\left[\nabla^2F\left(\bar{x}^{(k)}\right)\right]^{-1}\cdot\nabla F\left(\bar{x}^{(k)}\right). \qquad \text{(IV.23)}$$

Метод Ньютона реализуется следующей последовательностью действий:

1.Задается (произвольно) точка начального приближения $\bar{x}^0$.

2.Вычисляется а цикле по номеру итерации k=0,1,:

• значение вектора градиента $\vec{\nabla}F\left(\bar{x}\right)^{(k)}$ в точке $x=\bar{x}^{(k)}$;

• значение матрицы вторых производных $\vec{\nabla}^2F\left(\bar{x}\right)^{(k)}$;

• значение матрицы обратной матрице вторых производных $\vec{\nabla}^2F\left(\bar{x}\right)^{(k)^{-1}}$;

• значение шага $\Delta\bar{x}^{(k)}$ по формуле (IV.23);

• новое значение приближения $x^{(k+1)}$ по формуле (IV.19).

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

Оценка погрешности метода Ньютона дается формулой

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

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

<u>Достоинства метода Ньютона.</u>

• 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 $\overline{\nabla}^2 F(\bar{x})^{(k)}$ окажется плохо обусловленной.

Примечание.

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

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

$$\|A\|_2 = \max_{4\le l\le m}\left(\sum_{k=1}^{n}|a_{ik}|\right).$$

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 IV.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).

Создайте файл Rozenbrok.m (листинг IV.20), содержащий описание функции, возвращающей значения функции Розенброка.

Листинг IV.20. Файл Rozenbrok.m

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


2. Задайте координатную сетку.

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

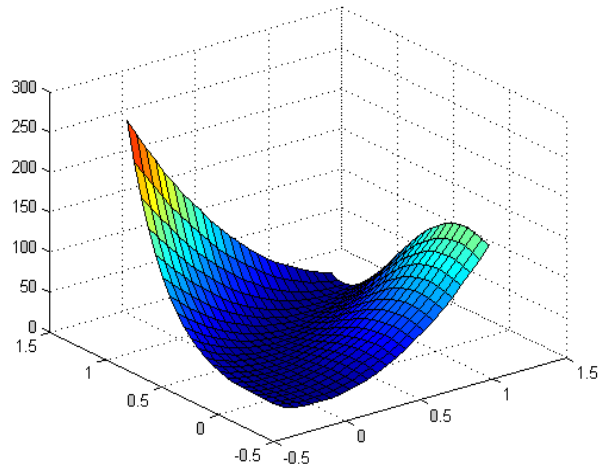
```
» [X Y]=meshgrid(x,y) ;
» surf(X,Y,Z); % визуализация поверхности, задаваемой функ-
цией % Розенброка (рис. IV.15)
» contour (X,Y, z, 53) ; % визуализация карты эквипотенциалей
                          % функции Розенброка (рис. IV.16)
```

5. Создайте файл R_x.m (листинг IV.21), содержащий описание функции, возвращающей значение первой производной по переменной x.

Листинг IV.21. Файл R_x.m

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

6. Создайте файл R_y.m (листинг IV.22), содержащий описание функции, возвращающей значение первой производной по переменной y.



Рие. IV.15. Поверхность, задаваемая функцией Розенброка

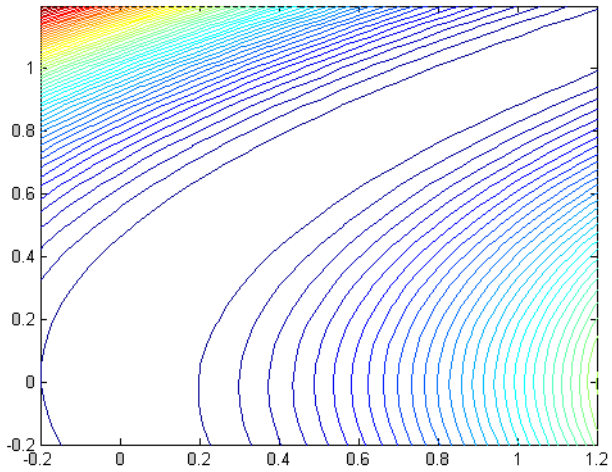


Рис. IV.16. Карта линий уровня функции Розенброка

Листинг IV.22. Файл R_y.m

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

7. Создайте файл R_xx.m (листинг IV.23), содержащий описание функции, возвращающей значение второй производной по переменной x.

Листинг IV.23. Файл R_xx.m

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

8. Создайте файл R_yy.m (листинг IV.24), содержащий описание функции, возвращающей значение второй производной по переменной y.

Листинг IV.24. Файл R_yy.m

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

9. Создайте файл R_xy.m (листинг IV.25), содержащий описание функции, возвращающей значение смешанной производной по переменным x, y.

Листинг IV.25. Файл R_xy.m

```
function z=R_xy(x,y)
N=length(x);
z=zeros(N);
for i=l:N
  for j=l:N
    z(i,j)=-400*x(i);
  end;
end;
```

10. Создайте файл Rg.m (листинг IV.26), содержащий описание функции, возвращающей значения обратной матрицы вторых производных на вектор-градиент.

Листинг IV.26. Файл Rg.m

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

11.Создайте файл RnO.m (листинг IV.27), содержащий описание функции, возвращающей значения переменных и соответствующих значений функции Розенброка.

Листинг IV.27. Файл RnO.m

```
function z=Rg(x,y)
A(l,l)=R_xx(x,y);
A(l,2)=R_xy(x,y);
A(2,l)=R_xy(x,y);
A(2,2)=R_yy(x,y);
B(l,l)=R_x(x,y);
```



```
B(2,l)=R_y(x,y);
z=A^-l*B;
```

12.Визуализируйте итерационный процесс (рис. IV.17-IV.19).

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

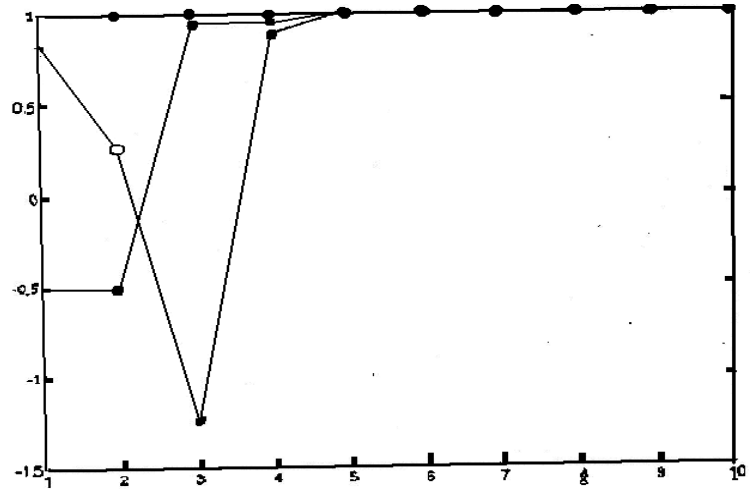


Рис. IV.17. Зависимость значений координат решения уравнения от номера итерации

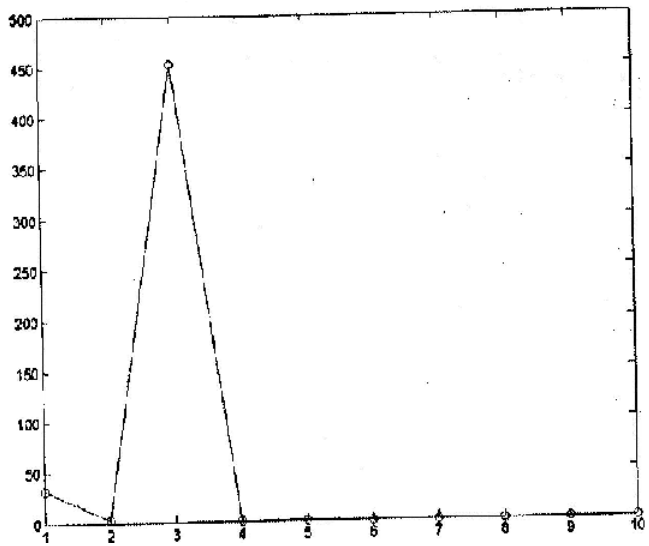


Рис. IV.18. Зависимость значения исследуемой функции от номера итерации

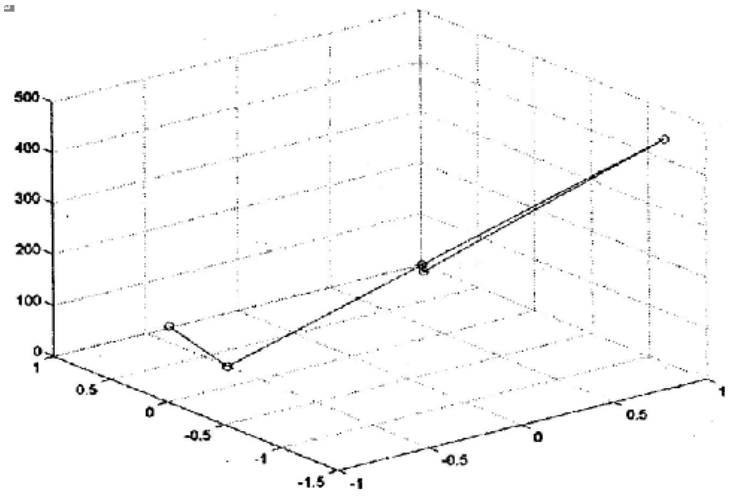


Рис. IV.19. Траектория итерационного процесса в пространстве

4.5. 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

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

где

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

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

Для этого необходимо выполнить следующую последовательность действий:

1. Создать файл fm.m (листинг IV.28), содержащий описание функции, возвращающей значения функции $\overline{F}(\overline{x})$.

Листинг IV.28. Файл fm.m

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

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( 'f solve' ) )
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 =
    l.0e-008 *
      0.5162
     -0.4888

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

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

fval =
    l.0e-008 *
     0.5162
    -0.4888

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

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

Iteration	Func-count	f(x)	Norm of step	First-order 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 Fml.m (листинг IV.29), содержащий описание функции, возвращающей одновременно значения функции $\overline{F}(\overline{x})$ и значения якобиана.

Листинг IV.29. Файл Fml.m

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

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('fml',z,options);

Iteration	Func-count	f(x)	Norm of step	First-order 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 IV.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 IV.30. Файл F_sq.m

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


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

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

V. ИНТЕРПОЛИРОВАНИЕ ФУНКЦИЙ

В данном рассмотрим основные методы решения задачи об интерполяции функции, заданной таблично (интерполяционный полином, полином Лагранжа, полином Ньютона, сплайн-интерполяция) и опишем соответствующие алгоритмы и их программные реализации, а также обсудим использование соответствующих функций пакета MATLAB.

5.1. Постановка задачи

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

Исходные данные в задаче интерполяции

X	Xo	X 1	. . .	Xп
f(x)	Уо	У1	. . .	Уп

Требуется получить значение функции *f*(x) для значения аргумента xє [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 = 0, 1,...,n)

$$F(x_0) = y_0, F(x_1) = y_1, \ldots, F(x_n) = y_n. \qquad (V.1)$$

В данном случае нахождение приближенной функции называется интерполированием, а точки x_0, x_1 ,..., xn называются узлами интерполяции.

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

$$P_n(x) = a_0x^n + a_1x^{n-1} + \ldots a_{n-1}x + a_n = \sum_{k=0}^{n} a_k n^{n-k} \qquad (V.2)$$

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

$$\sum_{k=0}^{n} a_k x_i^{n-k} y_i , \quad i = 0, 1, \ldots, n. \qquad (V.3)$$

Решив систему (V.3) относительно неизвестных a_0, a_1, … , a_n , находим значения этих неизвестных и, подставив в (V.2), находим аналитическое выражение аппроксимирующей функции.

Система (V.4) всегда имеет единственное решение, т. к. ее определитель

$$\begin{vmatrix} x_0^n & x_0^{n-1} & \ldots & 1 \\ x_1^n & x_1^{n-1} & \ldots & 1 \\ \vdots & \vdots & \ldots & \vdots \\ x_n^n & x_n^{n-1} & \ldots & 1 \end{vmatrix} , \qquad (V.4)$$

известный в алгебре как определитель Вандермонда, отличен от нуля. Следовательно, интерполяционный многочлен $P_n(x)$ существует и единственен.

Пример V.1. Решить, используя пакет MATLAB, задачу интерполяции с помощью полинома n-ой степени для функции *f* (x) = sin (x), заданной таблично в восьми точках на интервале [0, 2π].

1. Задайте табличные значения интерполируемой функции.

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

2. Визуализируйте табличную зависимость и истинные значения функции (рис. V.1).

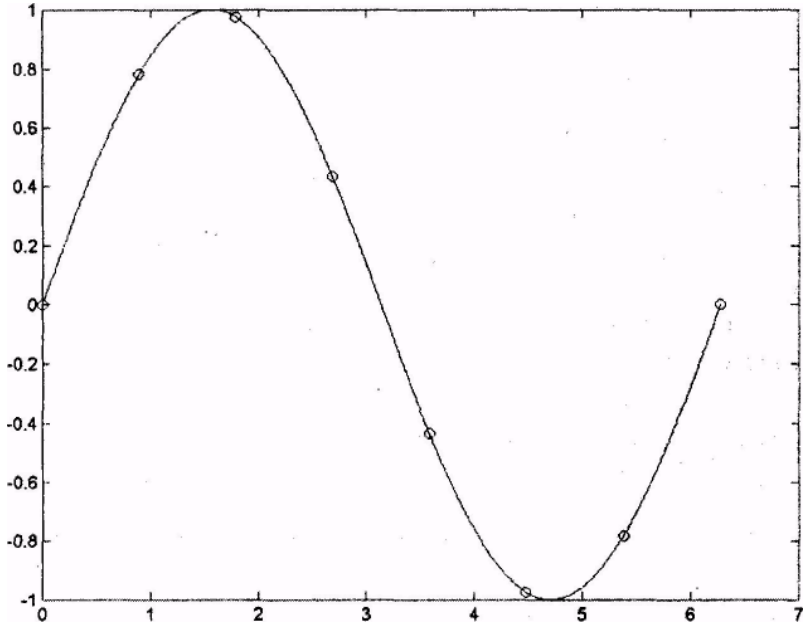


Рис. V.1. График функции f (x) = sin (x) и табличных значений, используемых для решения задачи интерполяции


```
» 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)
```

3. Создайте файл Pol.m (листингV.1), содержащий описание функции, возвращающей значения полинома (V.2).

Листинг V.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 Vandermortd.m (листинг V.2), содержащий описание функции, возвращающей значения элементов матрицы Вандермонда.

Листинг V.2. Файл Vandermond. m

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

5. Вычислите значения элементов матрицы Вандермонда.
» M=Vandennond(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. V.2).

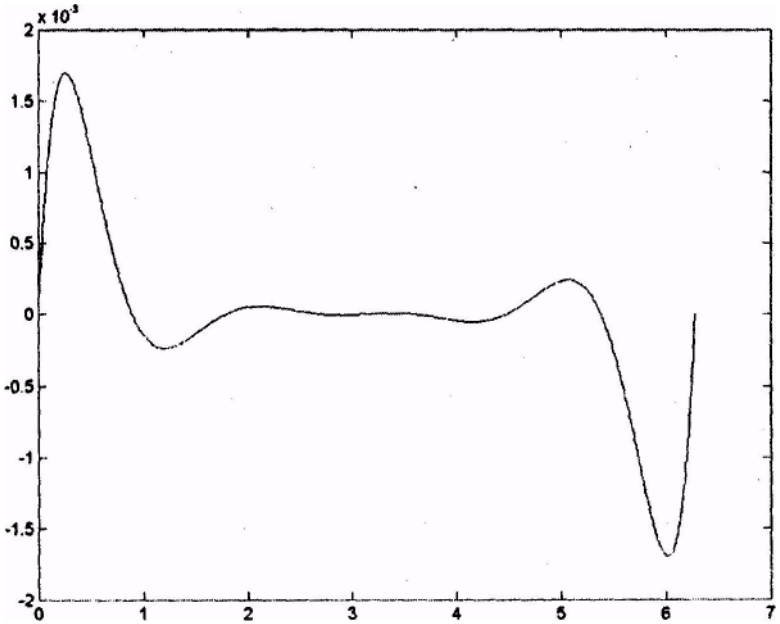


Рис. V.2. Погрешность аппроксимации функции f (x) = sin (x) полиномом 8-й степени.

5.2. Интерполяционный полином Лагранжа

Для функции, заданной табл. V.1, построим интерполяционный многочлен $L_n(x)$, степень которого не выше n и выполнены условия (V.1).
Будем искать $L_n(x)$ в виде

$$L_n(x) = l_o(x) + l_1(x) + \ldots + l_{n(x)}, \qquad (V.5)$$

где l_i – многочлен степени n, причем

$$l_i(x_k) = \begin{cases} y_i, \text{ если } i = к \\ 0, \text{ если } i \neq к \end{cases}. \qquad (V.6)$$

Очевидно, что требование (V.6) с учетом (V.5) обеспечивают выполнение условий (V.1).
Многочлены l_i,(x) составим следующим образом:

$$l_i(x) = c_i (x - x_0)(x - x_i) \cdot \ldots \cdot (x - x_{i-1})(x - x_{i+1}) \cdot \ldots \cdot (x - x_n), \quad (V.7)$$

где c_i— постоянный коэффициент, значение которого находится из первой части условия (V.6):

$$c_i = \frac{y_i}{(x_i - x_0) \cdot \ldots \cdot (x_i - x_{i-1})(x_i - x_{i+1}) \cdot \ldots \cdot (x_i - x_n)}, \qquad (V.8)$$

Подставив с, в (V.7) и далее в (V.5), окончательно получим:

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0) \cdot \ldots \cdot (x - x_{i-1}) \cdot (x - x_{i+1}) \cdot \ldots \cdot (x - x_n)}{(x_i - x_0) \cdot \ldots \cdot (x_i - x_{i-1})(x_i - x_{i+1}) \cdot \ldots \cdot (x_i - x_n)}. \quad (V.9)$$

Формула (V.9) решает поставленную задачу.
Пример V.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, 2π].
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 V.3), содержащий описание функции возвращающей значение многочлена l_i ,(x).

```
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=l;
for j=l: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 V.4), содержащий описание функции, возвращающей значения полинома Лагранжа.

Листинг V.4 . Файл Lagr. m

```
function z=Pol_I<agr(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=l: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. V.3).

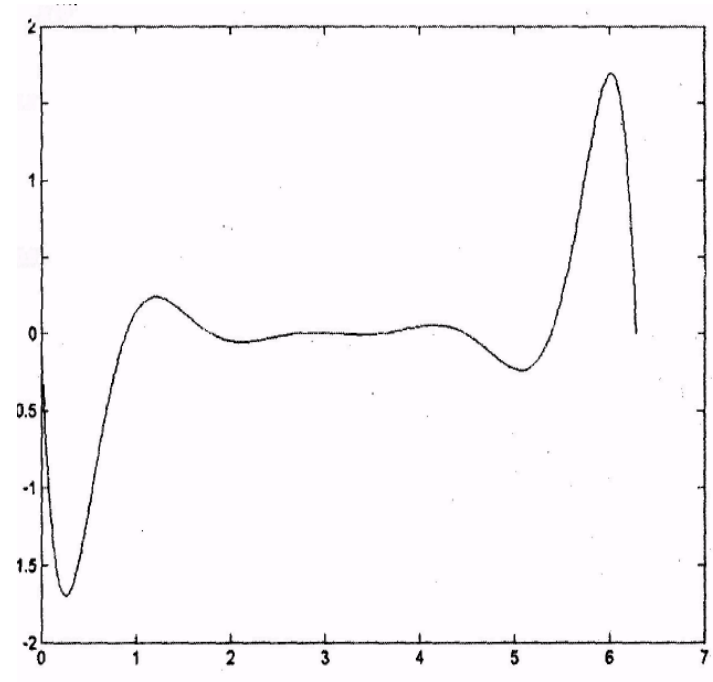


Рис. V.3. Погрешность аппроксимации функции f (x) = sin (x) полиномом Лагранжа

5.3. Интерполяционный полином Ньютона для равноотстоящих узлов

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

$$h = x_{i+1} - x_i \; (i = 1, 2, \ldots, n) \qquad \text{(V.10)}$$

5.3.1 . Конечные разности

Для функции, заданной табл. V.1 с постоянным шагом (V.10), определим разности между значениями функции в соседних узлах интерполяции:

$$\Delta y_i = y_{i+1} - y_i . \qquad \text{(V.11)}$$

Такое выражение называют конечными разностями первого порядка. Из конечных разностей первого порядка можно образовать конечные разности второго порядка:

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i = (y_{i+2} - y_{i+1}) - (y_{i+1} - y_i) = y_{i+2} - 2y_{i+1} + y_i . \qquad \text{(V.12)}$$

Аналогично получают выражение для конечных разностей третьего порядка:

$$\Delta^3 y_i = \Delta^2 y_{i+1} - \Delta^2 y_i = (y_{i+3} - 2y_{i+2} + y_{i+1}) - (y_{i+2} - 2y_{i+1} + y_i) = y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i . \qquad \text{(V.13)}$$

Методом математической индукции можно доказать, что

$$\Delta^k y_i = y_{i+k} - k y_{i+k-1} + \frac{k(k-1)}{2!} y_{i+k-2} - \ldots + (-1)^k y_i . \qquad \text{(V.14)}$$

5.3.2. Первая интерполяционная формула Ньютона

Будем искать интерполяционный полином в виде:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \ldots + \ldots + a_n(x - x_0) \cdot \ldots \cdot (x - x_{n-1}) \qquad \text{(V.15)}$$

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 (V.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:

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

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

$$y_2 = P_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = = a_0 + a_1 2h + a_2 2h^2 .$$

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 $a_к$ будет иметь вид

$$a_n = \frac{\Delta^n y_0}{n! h^2} . \qquad \text{(V.16)}$$

Подставляя (V.16) в выражение для многочлена (V.15), получаем

$$P_n(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta^2 y_0}{2!h^2}(x - x_0)(x - x_1) + \ldots + \\ + \ldots + \frac{\Delta^n y_0}{n!h}(x - x_0)\ldots(x - x_{n-1}) . \qquad \text{(V.17)}$$

Формула (V.17) называется первым интерполяционным полиномом Ньютона.

Пример V.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, 2π].

1. Создайте файл Newtonl.m (листинг V.5), содержащий описание функции, возвращающей локальное значение интерполяционного полинома Ньютона.

Листинг V.5. Файл Newton1. m

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



```
   r=l;
  for  i=l:k-l
    r=r.*(t-x(i));
  end;
  s=f(l,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=l:M
Z(j)=Newtonl(X(j),x,y); end;
```

5. Постройте разности между точными и интерполированными значениями функции (рис. V.4).

```
» plot(X,Y-Z)
```

Отдавая дань традициям преподавания численных методов в прошлом веке, приведем описание модификации формулы (V.17), применявшейся при ручных вычислениях. Положим

$\frac{x - x_0}{h} = t$, т.е. $x = x_0 + ht$, тогда:

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

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

. . .

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

Подставляя данные выражения в (V.17), окончательно получаем

$$P_n(x) = P_n(x_0 + th) = y_0 + t\Delta y_o + \frac{t(t-1)}{2!}\Delta^2 y_0 + \ldots + \\ + \ldots + \frac{t(t-1)\cdot\ldots\cdot(t-n+1)}{n!}\Delta^n y_0 \quad . \qquad \text{(V.18)}$$

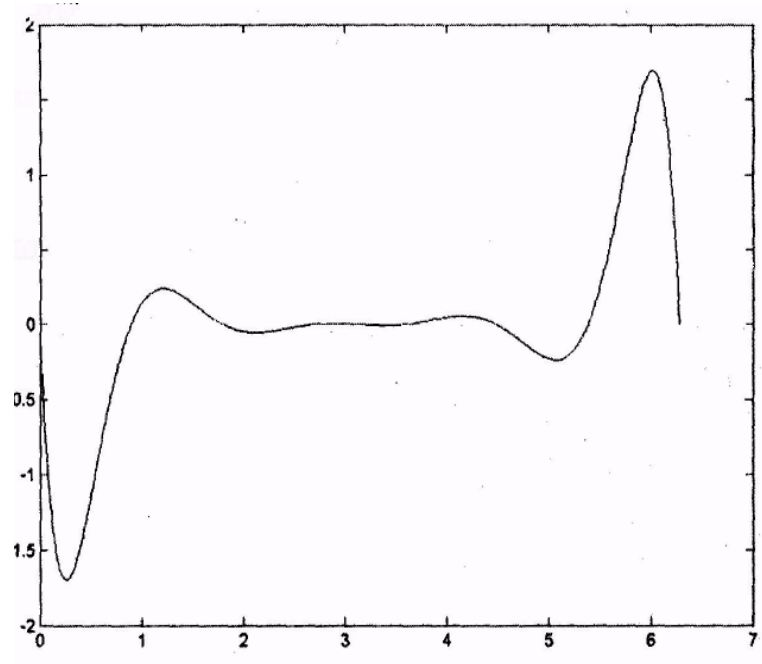


Рис. V.4. Разность между точными и интерполированными значениями функции с помощью первого полинома Ньютона

Формула (V.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:

$$P_n(x) = a_0 + a_1(x - x_n) + a_2(x - x_n)(x - x_{n-1}) + \ldots + \\ + \ldots + a_n(x - x_n)\cdot\ldots\cdot(x - x_1) . \qquad \text{(V.19)}$$

Коэффициенты полинома (V.19), находятся из условия совпадения значений функции и интерполяционного многочлена в узлах:

$$a_k = \frac{\Delta^k y_{n-k}}{k! n^k}. \qquad \text{(V.20)}$$

Подставив (V.20) в (V.19) и перейдя к переменной $t = \frac{x - x_n}{h}$, получим окончательный вид интерполяционной формулы Ньютона, используемой при ручных вычислениях:

$$P_n(x) = P_n(x_n + th) = y_n + t\Delta y_{n-1} + \frac{t(t+1)}{2!}\Delta^2 y_{n-2} + \ldots + \\ + \ldots + \frac{t(t+1)\cdot\ldots\cdot(t+n-1)}{n!}\Delta^n y_0 \qquad \text{(V.21)}$$

Пример V.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, 2π].

1. Создайте файл Newton2.m (листинг V.6), содержащий описание функции, возвращающей локальное значение интерполяционного полинома Ньютона.

Листинг V.6. Файл Newton2. m

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



```
for k=l:N
  r=l;
 for i=l:k-l
    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=l:M;
» X(j)=2*pi/(M-l)*(j-l);
» Y=sin(X);   % вычисление точных значений
```

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

```
» for k=l:M
Z(j)=Newtonl(X(j) ,x,y) ; end;
```

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

```
» plot(X,Y-Z)
```

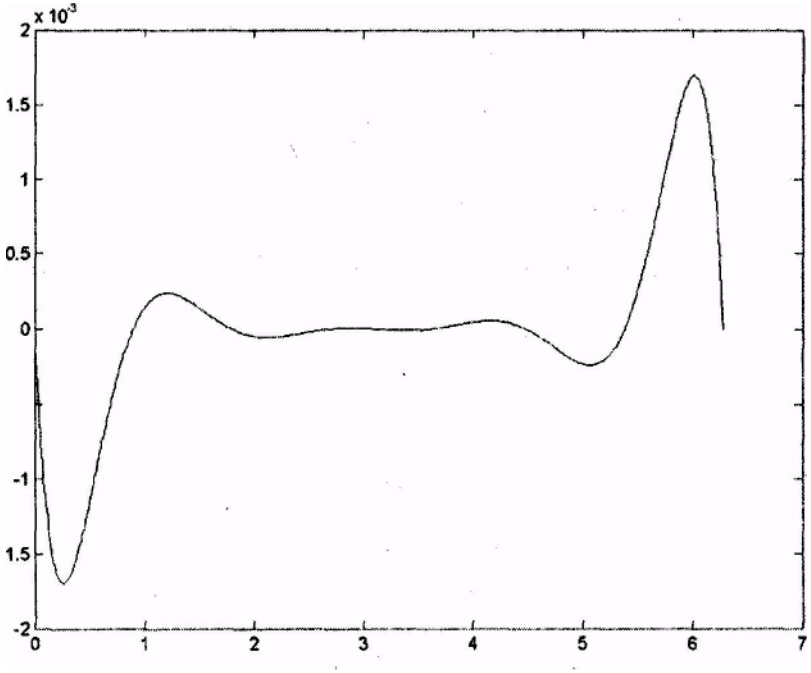


Рис. V.5. Разность между точными и интерполированными значениями функции с помощью второго полинома Ньютона

5.4. Погрешность интерполяции

Погрешность интерполяции полиномом Лагранжа оценивается по формуле:

$$\left|R_n(x)\right| \le \frac{M_{n+1}}{(n+1)!}\left|\Pi_{n+1}(x)\right|, \qquad \text{(V.22)}$$

где

$$M_{n+1} = \max_{x_0 \le x \le x_n}\left|f^{(n+1)}(x)\right| \quad , \qquad \text{(V.23)}$$

$$\Pi_{n+1}(x) = (x-x_0)(x-x_1)\cdot\ldots\cdot(x-x_n). \qquad \text{(V.24)}$$

Погрешность интерполяции полиномом Ньютона оценивается по формулам:

$$\left|R_n(x)\right| \le \frac{h^{n+1}}{(n+1)!}\left|t(t-1)(t-2)\cdot\ldots\cdot(t-n)\right|, \qquad \text{(V.25)}$$

$$\left|R_n(x)\right| \le \frac{h^{n+1}}{(n+1)!}\left|t(t+1)(t+2)\cdot\ldots\cdot(t+n)\right|, \qquad \text{(V.26)}$$

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, 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:

$$S(x) = a_i + b_i(x-x_{i-1}) + c_i(x-x_{i-1})^2 + d_i(x-x_{i-1})^3, \qquad \text{(V.27)}$$

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):

$$S(x_{i-1}) = y_{i-1} = a_i \ , \qquad \text{(V.28)}$$

$$S(x_i) = y_i = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 \qquad \text{(V.29)}$$

Число этих уравнений 2π 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 $S'(x-0),\ S'(x+0),\ S''(x-0),\ S'(x-0),\ S''(x+0)$ во внутреннем узле x_i .

Вычислив выражения для производных S'(x), S''(x) последовательным дифференцированием (V.27):

$$S'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2 \ , \qquad \text{(V.30)}$$

$$S''(x) = 2c_i + 6d_i(x - x_{i-1}) \ , \qquad \text{(V.31)}$$

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 $i = 0, 1, \ldots, n-1$.

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:

$$b_{i+1} = b_i + 2c_i h_i + 2d_i h_i^2 \ , \qquad \text{(V.32)}$$

$$c_{i+1} = c_i + 3d_i h_i \ , \qquad \text{(V.33)}$$

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

$$c_1 = 0 \ , \ c_n + 3d_n h_n = 0. \qquad \text{(V.34)}$$

Исключив из уравнений (V.28)-(V.33) я неизвестных a_t , получаем систему уравнений:

$$\begin{aligned} & b_i h_i - c_i h_i^2 - d_i h_i^3 = y_i - y_{i-1} \ , \\ & b_{i+1} - b_i - 2c_i h_i - 3d_i h_i^2 = 0 \ , \\ & c_{i+1} - c_i - 3d_i h_i = 0 \ , \qquad \text{(V.35)} \\ & c_1 = 0 \ , \\ & c_n + 3d_n h_n = 0 \ , \end{aligned}$$

где $i = 0, 1, \ldots, n-1$.

Система (V.35) состоит из 3n уравнений. Решив систему (V.35), получаем значения неизвестных b_i, c_i, d_i, определяющих совокупность всех формул для искомого интерполяционного сплайна

$$S_i(x) = y_{i-1} + b_{i(}(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3 \ , \qquad \text{(V.36)}$$

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 V.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 / (N-1)*(i-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. V.6, V.7).

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

5. Вычислите и визуализируйте значения первых производных сплайна (рис. V.8).

```
» m=1:M-1
» yyl(m) = (yy(m+l)-yy(m))/(2*pi/(M-l));
» plot(X(m),yyl(m))
```

6. Вычислите и визуализируйте значения вторых производных сплайна (рис. V.9).

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

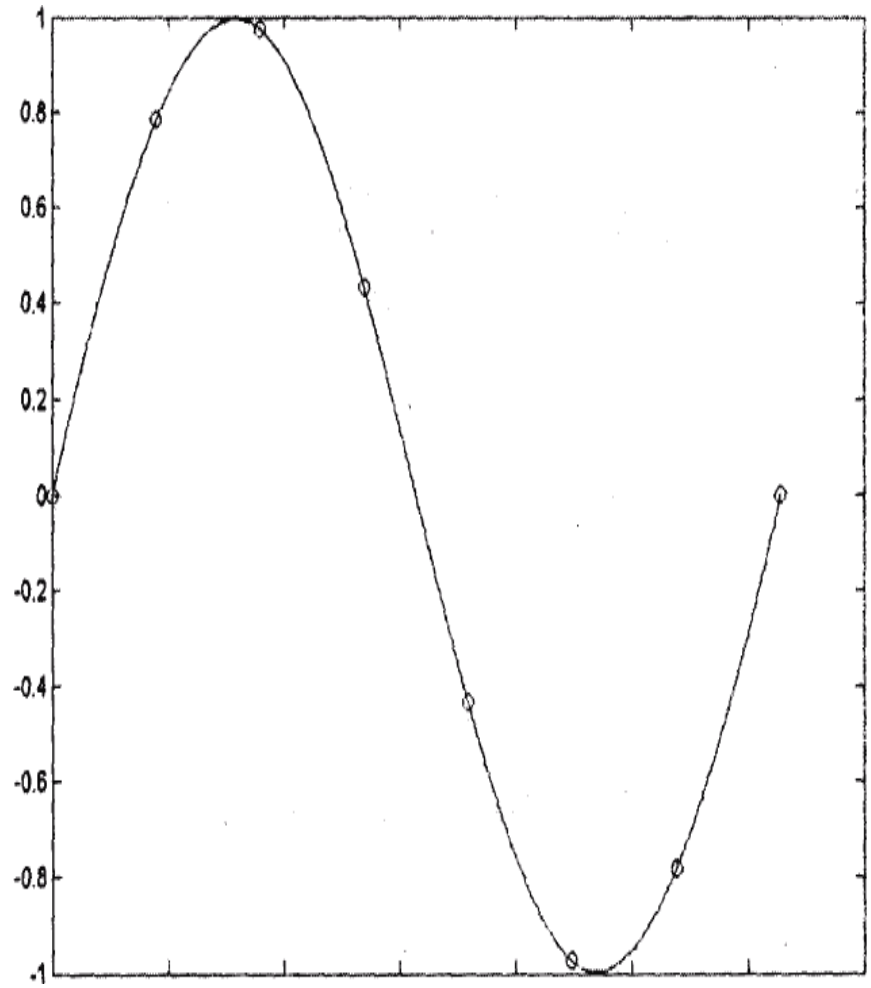



Рис. V.6. Визуализация исходных данных и результатов сплайн-интерполяции

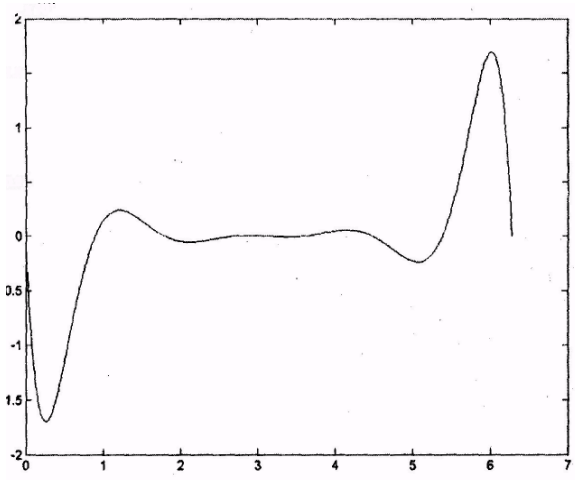


Рис. V.7. Разность между точными и интерполированными значениями функции $f(x) = \sin(x)$

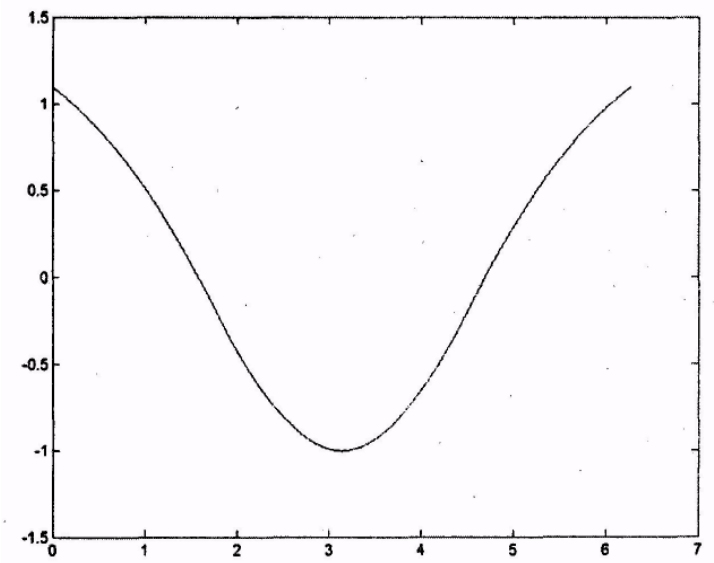


Рис. V.8. Первая производная, вычисленная по численным значениям, полученным сплайн-интерполяцией

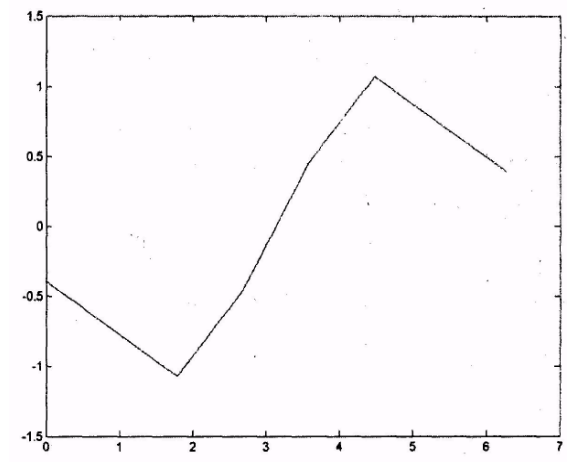


Рис. V.9. Вторая производная, вычисленная по численным значениям, полученным сплайн-интерполяцией

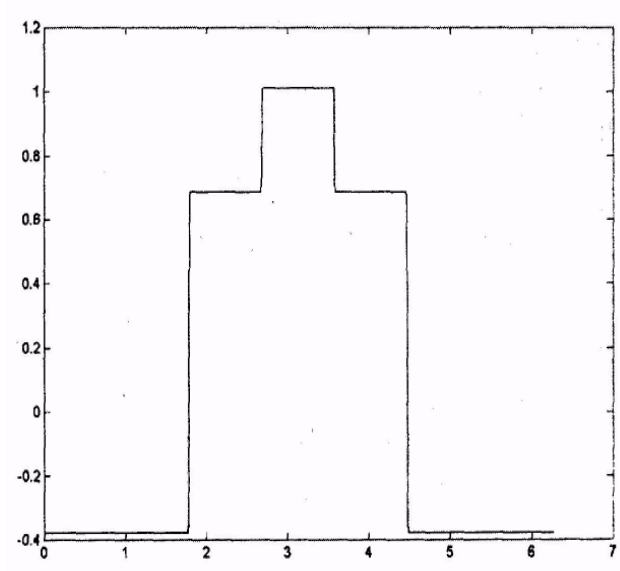


Рис. V.10. Третья производная, вычисленная по численным значениям, полученным сплайн-интерполяцией

7. Вычислите и визуализируйте значения третьих производных сплайнов (рис. V.10).
» yy3 (т) = (yy2 (m+1)-yy2 (m)) / (2*pi/ (M-1)) ; » plot(X(т),yy3);
» m=1:M-3;
Как видно из рис. V.7—V.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 interp 1 (), имеющая следующий синтаксис:
yi = interpi (x,y,xi) — линейная интерполяция табличных значений x, y в точках, абсциссы которых находятся в векторе xi.
yi = interpi (y,xi) — линейная интерполяция табличных значений y в точках, абсциссы которых находятся в векторе xi в предположении, что
x=l:length(y).
yi = interpi (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, 2π], используя встроенную функцию пакета MATLAB interpi ().
1. Задайте табличные значения интерполируемой функции.

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

2. Задайте значения абсцисс точек, в которых вычисляется значение интерполяционного полинома.
» M=1000;
» j=l:M,
» X(j)=2*pi/(M-l)*(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. V.11)
» yi=interpl(x,y,X);
» plot(x,y,'o[l],X,Y,X,yi)

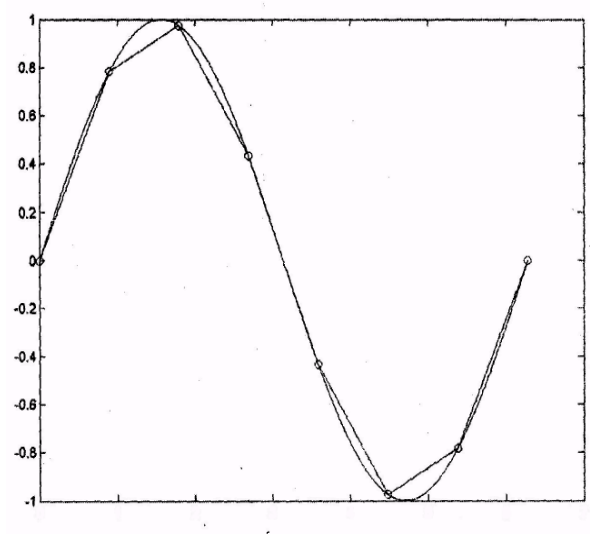


Рис. V.11. Исходные данные и результат линейной интерполяции

VI. ЧИСЛЕННОЕ ДИФФЕРЕНЦИРОВАНИЕ И ИНТЕГРИРОВАНИЕ

В данном разделе рассмотрим формулы численного дифференцирования и интегрирования (формулу прямоугольников, формулу трапеций, формулу Симпсона), погрешности данных формул, изложим основные идеи вычисления определенных интегралов методом Монте-Карло, обсудим использование соответствующих функций пакета MATLAB.

6.1. Численное дифференцирование функций, заданных аналитически

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

$$f'(x) = \lim_{\Delta \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \qquad \text{(VI.1)}$$

Переходя в (VI.1) от бесконечно малых разностей к конечным, получаем приближенную формулу численного дифференцирования:

$$f'(x) = \frac{f(x+\Delta x) - f()x}{\Delta x}. \qquad \text{(VI.2)}$$

Формула (VI.2) позволяет построить простой вычислительный алгоритм:

1. Задать значение точки, в которой вычисляется производная.
2. Задать значение приращения *Ax*.
3. Вычислить производную в соответствие с формулой (VI.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:

$$f(x+\Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \frac{f''(x)}{2!}(\Delta x)^2 + \ldots + \frac{f^{(2)}}{n!}(\Delta x)^n \ldots \text{(VI.3)}$$

Подставив (VI.3) в (VI.2) и приведя подобные члены, получим:

$$f'(x) \approx f'(x) + \frac{f'''(x)}{2!}\Delta x + \ldots \qquad \text{(VI.4)}$$

Из (VI.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 (VI.2) имеет первый порядок точности по Δx.

Можно вычислять производную, используя симметричную разностную схему:

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}. \qquad \text{(VI.5)}$$

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

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

$$-\frac{1}{2\Delta x}\left(f(x) - f'(x)\frac{\Delta x}{1!} + f''(x)\frac{(\Delta x)^2}{2!} - f'''(x)\frac{(\Delta x)^3}{3!} + \ldots \right). \qquad \text{(VI.6)}$$

Раскрыв в (VI.6) скобки и приведя подобные члены, получаем:

$$f'(x) \approx f'(x) + f''''(x)\frac{(\Delta x)^2}{3!} + \ldots \qquad \text{(VI.7)}$$

Из (VI.7) видно, что основной член погрешности равен $f'''(x)(\Delta x)^2/3!$, *т. к.* данный член $\sim(\Delta x)^2$, говорят, что формула (VI.5) имеет второй порядок точности по Δx.

Пример VI.1. Вычислить значения производной функции f(x) = $\sin(0.01x^2)$ 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=l:N-l;
» df(m)=(yf(m+1)-yf(m))/dx;
» plot(df) % визуализация производной функции (рис. VI.1)
» fl=inline('0.02*x.*cos(0.01*x.^2)'); % задание функции,
                                % описывающей  первую производную
» ya=feval(£l,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. VI.2)
```

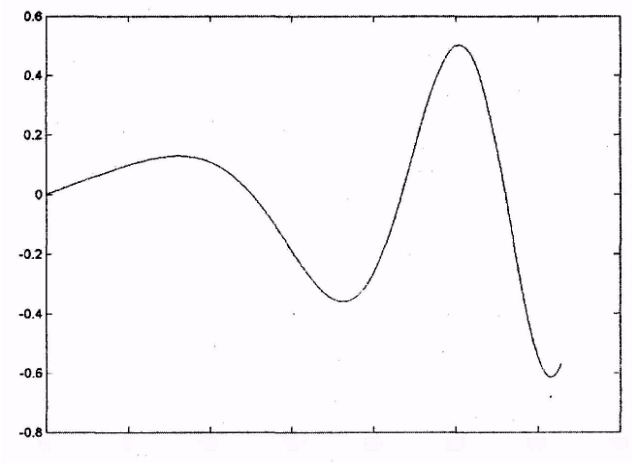


Рис. VI.1. График производной функции f (x) = sin ($0.01x^2$)

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 л-го порядка;

diff (x,n,dim) — возвращает конечные разности n-го порядка по столбцам (dim=i) или строкам матрицы.

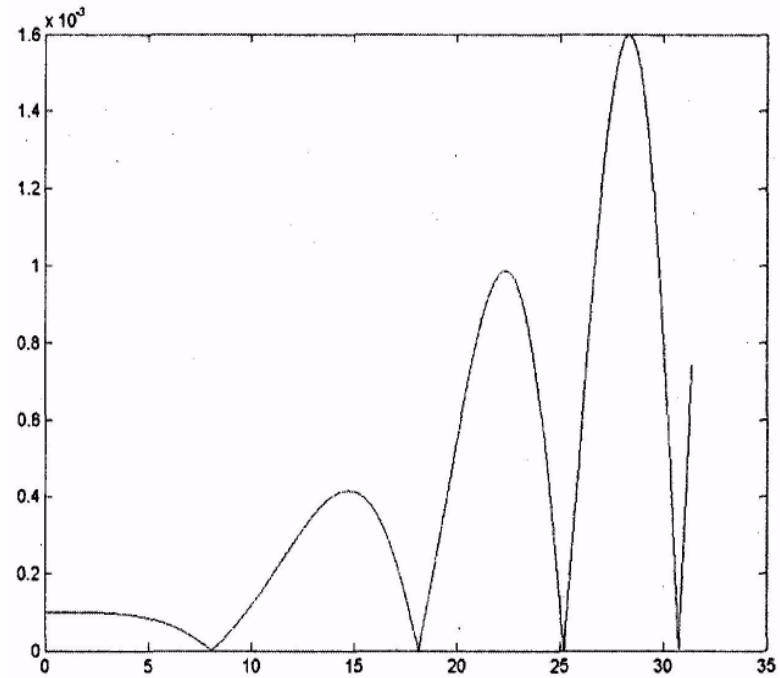


Рис. VI.2. Разность между точными и численными значениями производных функции *f* (x) = sin ($0.01x^2$)

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 главе V мы строили интерполяционный полином $F_n(x)$, продифференцировав который, можно получить значение производной

$$f'(x) = F_n'(x) \quad . \qquad \text{(VI.8)}$$

Полагая, что погрешность интерполирования определяется формулой

$$R_n(x) = f(x) - F_n(x) \quad , \qquad \text{(VI.9)}$$

находим оценку погрешности вычисления полинома

$$r_n(x) = f'(x) - F_n'(x) \quad . \qquad \text{(VI.10)}$$

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

$$F = \int_a^b f(x)dx \qquad \text{(VI.11)}$$

— есть площадь фигуры, ограниченная графиком функции f(x) и прямыми $x = a$, $x = b$ (рис. VI.3).

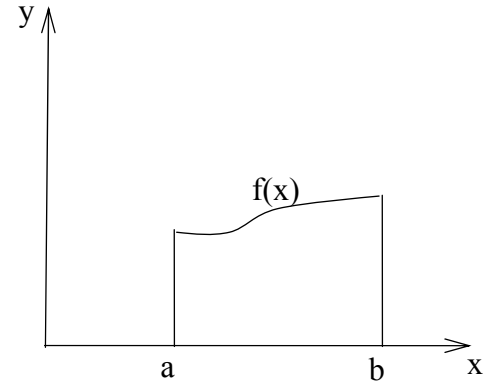


Рис. VI.3 К объяснению геометрического смысла определенного интеграла

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

$$\Delta x = \frac{b-a}{N} \, . \qquad \text{(VI.12)}$$

Тогда координата правого конца i-го отрезка определяется по формуле

$$x_i = x_0 + i\Delta x \, , \qquad \text{(VI.13)}$$

где $x_0 = a$, $i = 0, 1, \ldots, N$.

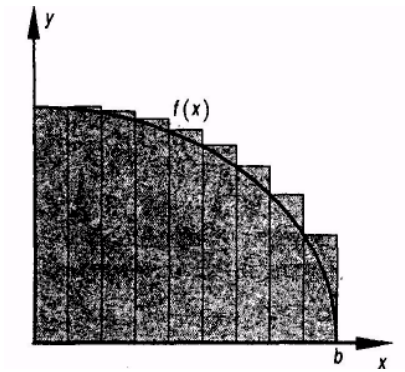


Рис. VI.4. Геометрическая интерпретация метода левых прямоугольников

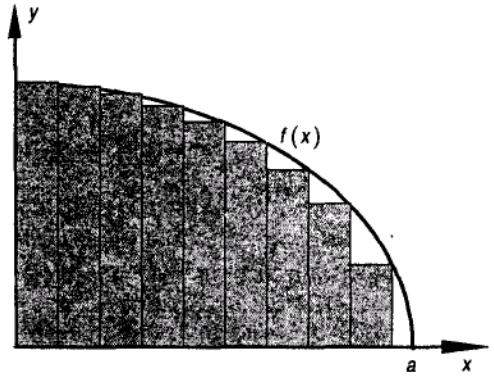


Рис. VI.5. Геометрическая интерпретация метода правых прямоугольников

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. VI.4) или в точке x_{i+1} (метод правых прямоугольников, рис. VI.5). Погрешность вычисления значения интеграла на каждом шаге показана на рисунках закрашенными фигурами.
Значение определенного интеграла вычисляется по формулам

$$F_L = \sum_{i=0}^{N-1} f(x_i)\Delta x \qquad \text{(VI.14)}$$

$$F_R = \sum_{i=1}^{N} f(x_i)\Delta x \qquad \text{(VI.15)}$$

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:

$$F_N = \sum_{i=0}^{N-1} \frac{1}{2}(f(x_{i+1}) + f(x_i))\Delta x = \left[\frac{1}{2}f(x_0) + \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2}f(x_N)\right]\Delta x. \qquad \text{(VI.16)}$$

Более высокая точность вычисления интегралов обеспечивается при использовании параболической интерполяции (полиномом второй степени) по трем соседним точкам:

$$y = ax^2 + bx + c\,. \qquad \text{(VI.17)}$$

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:

$$\begin{cases} y_0 = ax_0^2 + bx_0 + c, \\ y_1 = ax_1^2 + bx_1 + c, \\ y_2 = ax_2^2 + bx_2 + c, \end{cases} \qquad \text{(VI.18)}$$

относительно неизвестных a ,b , c.
Решив систему (VI.18) относительно неизвестных a, b, c любым известным методом (например, Крамера), подставив найденные выражения в (VI.17) и выполнив элементарные преобразования, получаем

$$y = y_0\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + y_1\frac{(x-x_0)(x-x_2)}{(x_1-x_2)(x_1-x_2)} + y_2\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}. \qquad \text{(VI.19)}$$

Отметим, что формула (VI.19) совпадает с интерполяционным полиномом Лагранжа, подробно рассмотренным в главе *V,* при интерполяции по трем точкам.

Площадь под параболой y = y(x) на интервале $[x_0, x_2]$ находится посредством элементарного интегрирования (VI.19):

$$F_0 = \frac{1}{3}(y_0 + 4y_1 + y_2)\Delta x, \qquad \text{(VI.20)}$$

где $\Delta x = x_1 - x_0 = x_2 - x_1$.

Искомый определенный интеграл находится как площадь всех параболических сегментов (формула Симпсона):

$$F_N = \frac{1}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + \right. \\ \left. + \ldots + 2f(x_{N-2}) + 4f(x_{N-1}) + f(x_N)\right]\Delta x . \qquad \text{(VI.21)}$$

Обратите внимание на то обстоятельство, что в формуле Симпсона N должно быть четным числом.

Пример VI.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=l:N;
» dx=(Xmax-Xmin)/(N-l); % 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;
» yl(m-l)=y(m) ;
» Fr=sum(yl) *dx
Fr =
1.0004
» Fr-1
ans =
```



```
    3.9284e-004
% вычисление интеграла по формуле левых прямоугольников
» m=l:N-l;
» yl (m) =y (m) ;
» Fl=sum(yl)*dx
Fl =
    0.9996
» Fl-1
ans =
     -3.9295e-004
% вычисление интеграла методом трапеций
» s=0;
    for i=2:N-l
       s=s+y[i];
    end;
    Ft=(0.5*y(l)+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-l
         if  i-2*ceil(i/2)= =0
             k=4;
         else
             k=2;
         end;
         s=s+k*y(i);
      end;
      Fs=(y(l)+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 ( ), guadl( ), 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, pi,p2,...) — возвращает значение интеграла от функции fun на на интервале [a, b] на каждом шаге итерационного процесса, pi, p2 — параметры, передаваемые в функцию fun;

[q,fcnt] = quadi(fun,a,b,...) — возвращает в переменную fcnt дополнительно к значению интеграла число выполненных итераций.

Функция guad1 ( ) возвращает значения интеграла, используя для вычислений метод Лоббато (Lobbato). Синтаксис этой функции аналогичен синтаксису функции quad ().

Пример VI.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 VI.4. Вычислить интеграл $\int_0^{\pi/2} \sin(x)dx \;\; dx$ 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 VI.5. Вычислить интеграл с переменным верхним пределом $\int_0^x \sin(\xi)d\xi$ на интервале $[0,\, 3\pi/2]$.

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. VI.6)
```

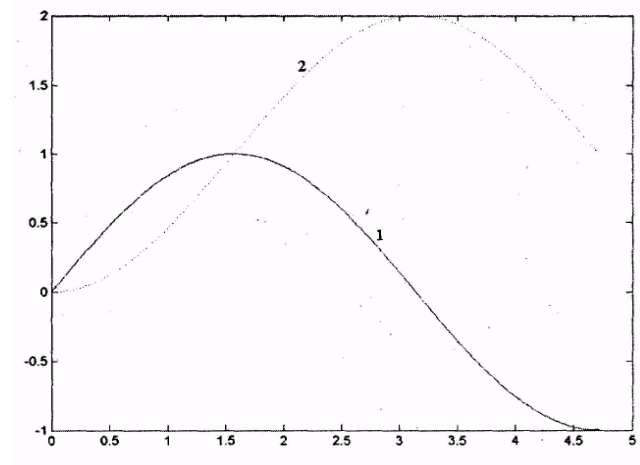



Рис. VI.6. Графики функций f(x)=sin (x) (1), $\varphi(x) = \int_0^x \sin(\xi)d\zeta$

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:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''(x_i)(x - x_i)^2 + \ldots \qquad \text{(VI.22)}$$

Тогда интеграл от данной функции на отрезке $[x_i\ , x_{i+1}]$ будет равен:

$$\int_{x_i}^{x_{i+1}} f(x)dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \ldots \quad \text{(VI.23)}$$

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 $f(x_i)\Delta x$:

$$\Delta_i = \left[\int_{x_i}^{x_{i+1}} f(x)dx\right] - f(x_i)\Delta x \approx \frac{1}{2} f'(x_i)(\Delta x)^2 \ . \qquad \text{(VI.24)}$$

Из (VI.24) видно, что основной член погрешности на каждом отрезке имеет порядок $(\Delta x)^2$ или в символической записи $0((\Delta x)^2)$. Поскольку полное число отрезков равно N, а $\Delta x = (b-a)/N$, то полная погрешность метода левых прямоугольников по порядку величины равна $N\Delta_i \approx N(\Delta x)^2 \approx 0(N^{-1})$. Аналогично можно показать, что погрешность метода правых прямоугольников также пропорциональна $0(N^{-2})$.

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:

$$\Delta_i = \left[\int_{x_i}^{x_{i+1}} f(x)dx\right] - 1/2\left[f(x_i) + f(x_{i+1})\right]\Delta x \qquad \text{(VI.25)}$$

Заменив в (VI.25) первый член выражением (VI.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 $\frac{1}{4} f''(x)(\Delta x)^3 \approx 0((\Delta x)^3) \approx 0(N^{-3})$. Следовательно, полная погрешность формулы трапеций на отрезке [a,b] по порядку величины равна $0(N^{-2})$.

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 $f^{IV}(x)(\Delta x)^4$. Следовательно, погрешность формулы Симпсона на отрезке $[x_i, x_{i+1}]$ пропорциональна $f^{IV}(x_i)(\Delta x)^5$., а полная погрешность на отрезке [a,b] по порядку величины составляет $0(N^{-4})$.

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:

$$f(x,y) = f(x_i, y_{i_i}) + f_x'(x_i, y_i)(x - x_i) + f_y'(x_i, y_i)(y - y_i) + \ldots, \qquad \text{(VI.26)}$$

где f_x', f_y' – частные производные по соответствующим переменным. Погрешность вычисления интеграла Δ_i равна

$$\Delta_i = \iint f(x,y)dxdy - f(x_i, y_i)\Delta x\Delta y. \qquad \text{(VI.27)}$$

Подставив (VI.26) в (VI.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 0(Δx), то погрешность интегрирования по прямоугольнику $x_i \le x \le x_{i+1}$, $y_i \le y \le y_{i+1}$ равна

$$\Delta_i \approx \frac{1}{2}\left[f_x'(x_i, y_i) + f_y'(x_i, y_i)\right](\Delta x)^3. \qquad \text{(VI.28)}$$

Из (VI.28) видно, что погрешность интегрирования по одному параллелепипеду составляет $0((\Delta x)^3)$. Так как имеется N параллелепипедов, полная погрешность по порядку величины равна $N(\Delta x)^3$. Однако в двумерном случае $N \sim \frac{1}{(\Delta x)^2}$, поэтому полная погрешность $\Delta_i \sim (\Delta x) \sim 0(N^{-1/2})$. Напомним, что в одномерном случае полная погрешность метода прямоугольников $\Delta_i \sim 0(N^{-1})$.

Аналогичные оценки для двумерных обобщений формул трапеций и Симпсона показывают, что они соответственно равны $0(N^{-1})$ и $0(N^{-2})$. Вообще можно показать, что если для одномерного случая погрешность составляет $0(N^{-\alpha})$, то в d-мерном случае она равна $0(N^{-\alpha/d})$.

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 (VI.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.

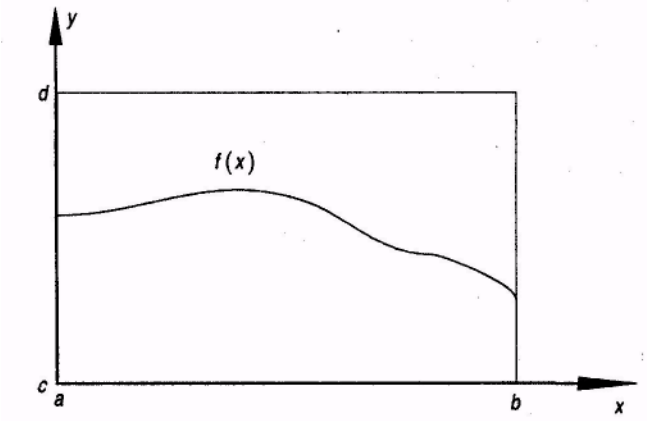


Рис. VI.7. К объяснению метода Монте-Карло

Представим себе прямоугольник высотой H и длиной (b - a), такой, что функция f(x) целиком лежит внутри данного прямоугольника (рис. VI.7).

Сгенерируем N пар случайных чисел, равномерно распределенных в данном прямоугольнике:

$$a \le x_i \le b \text{ , } 0 \le y_i \le H. \qquad \text{(VI.29)}$$

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:

$$F_N = A\frac{n_s}{N}, \qquad \text{(VI.30)}$$

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]:

$$F_N = (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i), \qquad \text{(VI.31)}$$

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 $0(N^{-1/2})$. Следовательно, для достаточно больших d интегрирование по методу Монте - Карло будет приводить к меньшим погрешностям при тех же значениях *N*.

Пример VI.6. Вычисление интеграла $\int_0^{\pi/2} \sin(x)dx$ методом Монте - Карло в пакете MATLAB.

Решение:

```
% задание координат вершит прямоугольника
» X min = 0;
» X max = pi/2;
» Y min = 0;
» Y max =1.5;
% генерация случайных координат
» N=2000;
» x =X min+ (X max - X min) *rand (N, 1) ;
» y =Y min+(Y max - Y min) *rand (N,l) ;
% подсчет числа точек, попавших под график функции
» s=0 ;
» for i =l:N
        if y(i)<=feval(f,x(i))
            s=s+l;
          end;
       end;
```



```
» s* (Xmax-Xmin) * (Ymax-Ymin) /N  % вычисление значения
% интеграла
ans =
    1.0261                                                    ;
% вычисление интеграла в соответствие с (VI.31)
» Fr=feval(f ,x) ;
» (Xmax-Xmin) /N*sum(Fr)
ans =
     1.0091
```

VII. МЕТОДЫ ОБРАБОТКИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

В данной главе изложим подход к решению задачи о среднеквадратичном приближении функции, заданной таблично, рассмотрим аппроксимацию элементарными функциями, линейными комбинациями элементарных функций и функциями произвольного вида, рассмотрим 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. VII.1).

Таблица VII.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

$$x_1, x_2, \ldots, x_n$$

совпадать с соответствующими значениями *f(x)* из табл. VII.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. VII.1).

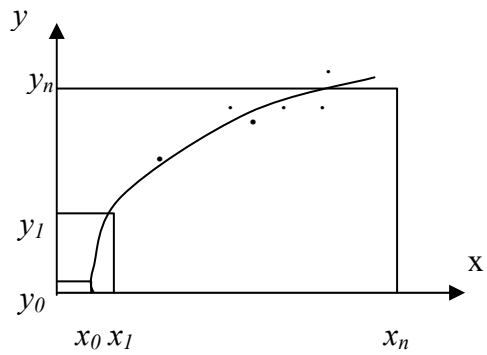


Рис. VII.1. К объяснению метода наименьших квадратов

Поставим задачу так, чтобы с самого начала обязательно учитывался характер исходной функции: найти функцию заданного вида

$$y = F(x), \qquad \text{(VII.1)}$$

которая в точках $x_1, x_2, \ldots, x_n$ принимает значения как можно более близкие к табличным значениям $y_1, y_2, \ldots, y_n$.

Следует отметить, что строгая функциональная зависимость для табл. VII.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 $x_1, x_2, \ldots, x_n$ имеет значения

$$\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_n. \qquad \text{(VII.2)}$$

Требование близости табличных значений $y_1, y_2, \ldots, y_n$ и значений (VII.2) можно истолковать таким образом. Будем рассматривать совокупность значений функции *f(x)* из табл. VII.1 и совокупность значений (VII.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 $M(y_1, y_2, ..., y_n)$ и $\bar{M}\left(\bar{y}_1, \bar{y}_2, ..., \bar{y}_n\right)$ было наименьшим. Воспользовавшись метрикой Евклидова пространства, приходим к требованию, чтобы величина

$$\sqrt{\left(y_1 - \bar{y}_2\right)^2 + \left(y_2 - \bar{y}_2\right)^2 + ... + \left(y_n - \bar{y}_n\right)^2}, \qquad \text{(VII.3)}$$

была наименьшей. Это равносильно следующему: сумма квадратов

$$\left(y_1 - \bar{y}_2\right)^2 + \left(y_2 - \bar{y}_2\right)^2 + \left(y_n - \bar{y}_n\right)^2 \qquad \text{(VII.4)}$$

должна быть наименьшей.

Приведем окончательную формулировку задачи приближения функции *f(x):* для функции *f*(*x*), заданной табл. VII.1, найти функцию *F(x)* определенного вида так, чтобы сумма квадратов (VII.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.)

$$y = \frac{1}{ax + b} \qquad y = ax + b$$

$$y = ax^2 + bx + c \qquad y = a \ln x + b$$

$$y = ax^m \qquad y = a\frac{1}{x} + b$$

$$y = ae^{mx} \qquad y = \frac{x}{ax + b}$$

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:

$$y = F(x, a, b, c). \qquad \text{(VII.5)}$$

Имеем:

$$F\left(x_i, a, b, c\right) = \bar{y}_i\,. \qquad \text{(VII.6)}$$

Сумма квадратов разностей соответствующих значений функций *f(x)* и *F(x)* имеет вид:

$$\sum_{i=1}^{n}\left(y_i - F(x_i, a, b, c)\right)^2 = \Phi(a, b, c). \qquad \text{(VII.7)}$$

Сумма является функцией *Φ(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:

$$\sum_{i=1}^{n}\left[y_i - F(x_i, a, b, c)\right] \cdot F_n'(x_i, a, b, c) = 0,$$

$$\sum_{i=1}^{n}\left[y_i - F(x_i,a,b,c)\right]\cdot F'_b(x_i,a,b,c) = 0, \qquad \text{(VII.8)}$$

$$\sum_{i=1}^{n}\left[y_i - F(x_i,a,b,c)\right]\cdot F'_c(x_i,a,b,c) = 0.$$

Решив систему (VII.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 (VII.8).

Значения разностей

$$y_i - F(x_i,a,b,c) = \varepsilon_i \qquad \text{(VII.9)}$$

называют отклонениями измеренных значений от вычисленных по формуле (VII.5).

Сумма квадратов отклонений

$$\sigma = \sum_i^n \varepsilon_i^2 \qquad \text{(VII.10)}$$

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 (VII.10) имеет наименьшее значение.

7.2. Нахождение приближающей функции в виде линейной функции и квадратичного трехчлена

Ищем приближающую функцию в виде:

$$F(x,a,b) = ax + b. \qquad \text{(VII.11)}$$

Находим частные производные

$$\frac{\partial \Phi}{\partial a} = x, \frac{\partial F}{\partial b} = 1. \qquad \text{(VII.12)}$$

Составляем систему вида (VII.8)

$$\sum\left(y_i - ax_i - b\right)x_i = 0,$$

$$\sum\left(y_i - ax_i - b\right) = 0.$$

Примечание.

Здесь и далее сумма берется по переменной i=1,2,... ,n.

Имеем:

$$\sum x_i y_i - a\sum x_i^2 - b\sum x_i = 0,$$

$$\sum y_i - a\sum x_i^2 - bn = 0. \qquad \text{(VII.13)}$$

Разделив каждое уравнение (VII.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:

$$M_{x^2}\cdot a + M_x\cdot b = M_{xy},$$
$$M_{x^2}\cdot a + b = M_y.$$

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:

$$\begin{pmatrix} a \\ b \end{pmatrix}=\begin{pmatrix} M_{x^2} & M_x \\ M_x & 1 \end{pmatrix}^{-1}\begin{pmatrix} M_{xy} \\ M_y \end{pmatrix}. \qquad \text{(VII.14)}$$

Вычислив значения параметров *a, b* в соответствие с (VII.14), получаем конкретные значения и, следовательно, конкретный вид линейной функции (VII.11).

В случае нахождения приближающей функции в форме квадратного трехчлена имеем:

$$F(x,a,b,c)=ax^2+bx+c. \qquad \text{(VII.15)}$$

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 (VII.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:

$$\sum y_i x_i^2-a\sum x_i^4-b\sum x_i^3-c\sum x_i^2=0,$$
$$\sum y_i x_i-a\sum x_i^3-b\sum x_i^2-c\sum x_i=0,$$
$$\sum y_i-a\sum x_i^2-b\sum x_i-c\,n=0.$$

Разделив каждое уравнение на n и перенеся члены, не содержащие неизвестные параметры в правую часть, получаем:

$$\left(\frac{1}{n}\sum x_i^4\right)a+\left(\frac{1}{n}\sum x_i^3\right)b+\left(\frac{1}{n}\sum x_i^2\right)c=\frac{1}{n}\sum y_i x_i^2,$$
$$\left(\frac{1}{n}\sum x_i^3\right)a+\left(\frac{1}{n}\sum x_i^2\right)b+\left(\frac{1}{n}\sum x_i\right)c=\frac{1}{n}\sum y_i x_i, \qquad \text{(VII.16)}$$
$$\left(\frac{1}{n}\sum x_i^2\right)a+\left(\frac{1}{n}\sum x_i\right)b+c=\frac{1}{n}\sum y_i.$$

Решив систему (VII.16) относительно неизвестных *a*, *b, c*, находим значения параметров приближающей функции.

Пример VII.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=l:N;
» Xmin=0;
» Xmax=10;
» x(i)=Xmin+(Xmax-Xmin)/(N-l)*(i-l);
» 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);
» yl=y+Yrnd'; % создание зашумленных данных
» plot(x,y,x,yl,'o'); % визуализация точной и зашумленной
                      % последовательностей (рис. VII.2)
```

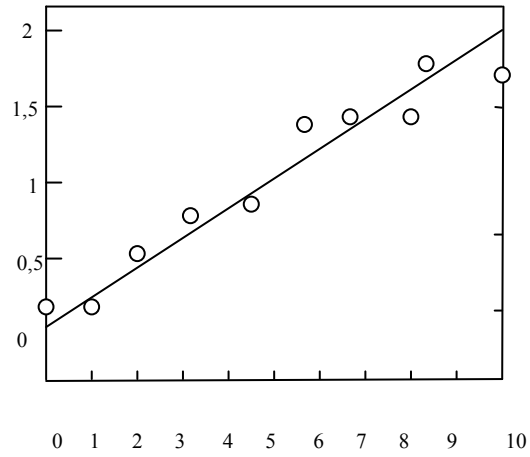


Рис. VII . 2. Исходные данные и точная зависимость y = 0.2x

```
% вычисление элементов матрицы M в (VII.14)
» tmp=x(i).^2;
» M(l,l)=l/N*sum(tmp);
» M(l,2)=l/N*sum(x);
» M(2,1)=M(1,2);
» M(2,2)=l;
% вычисление элементов вектора d
» d(l,l)=l/N*dot(x,yl);
» d(2,l)=l/N*sum(yl);
% решение системы линейных уравнений (VII.14)
» Coeff=A^-l*d;
Coeff =
    0.1941
    0.0993
» F=inline('a*x+b','a','b','x');
% задание аппроксимирующей функции
»tmpl(i)=feval(F,Coeff(1,1),Coeff(2,1),x(i));
% вычисление значений
% аппроксимирующей
% функции
```



```
% вычисление суммы квадратов отклонений
% при линейной аппроксимации
» tmp=tmpl-yl;
» dot(trap, tmp)
ans =
    0.0685
% аппроксимация исходных данных полиномом
% второй степени задание матрицы системы
% линейных уравнений в (VII.16)
» A(l,l)=l/N*sum(x.^4);
» A(1,2)=1/N*sum(x.^3);
» A(l,3)=l/N*sum(x.^2);
» A(2,1)=A(1,2);
» A(2,2)=A(1,3);
» A(2,3)=l/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(l,l)=l/N*dot(x.^2,yl);
» d(2,l)=l/N*dot(x,yl);
» d(3,l)=l/N*sum(yl) ;
% решение системы линейных уравнений(VII.16)
» Coeff=A^-l*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-yl; 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:

$$\begin{pmatrix} y1_1 \\ y1_2 \\ \vdots \\ y1_{10} \end{pmatrix} = \begin{pmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_{10} & t_{10}^2 \end{pmatrix} \times \begin{pmatrix} Coeff_0 \\ Coeff_1 \\ Coeff_2 \end{pmatrix}.$$

Решение данной системы уравнений, удовлетворяющее методу наименьших квадратов, находится с помощью оператора \:

$$\text{Coeff} = \text{A}\backslash\text{yl},$$

где

$$y1 = \begin{pmatrix} y1_1 \\ y1_2 \\ \vdots \\ y1_{10} \end{pmatrix}, \qquad A = \begin{pmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_{10} & t_{10}^2 \end{pmatrix}.$$

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\yl
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 VII. 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. VII. 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. VII.3;
```

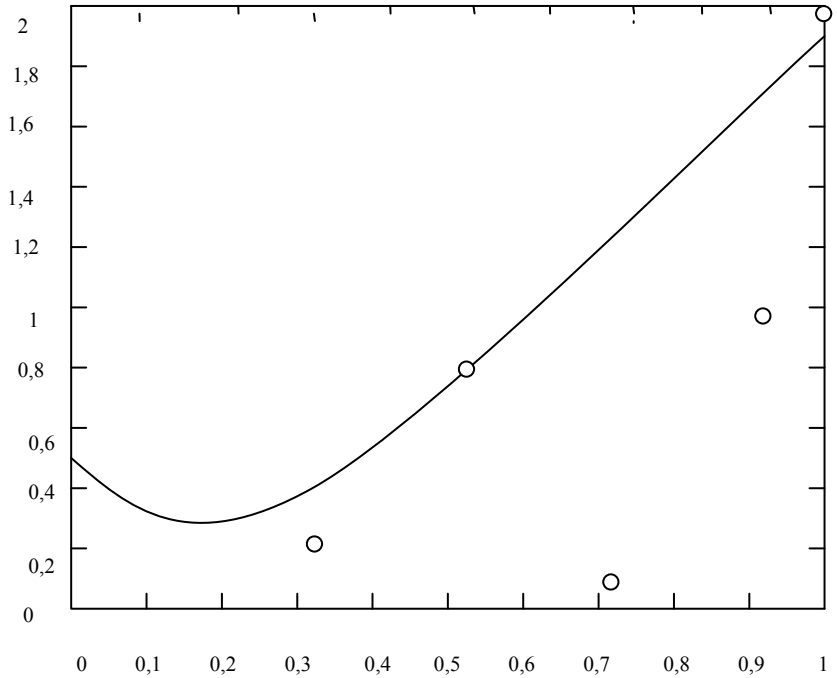


Рис. VII.3. Исходные данные и аппроксимирующая функция вида $F(x,a,b,c) = a \cdot x^2 + b \cdot x + c \cdot 1/(x + 1)$

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)*:

$$\min(f(x)) = f1(x)^2 + f_2(x)^2 + \ldots f(x)^2 + L,$$

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,PI,P2,...)
[x,resnorm,residual,exitflag,output,lambda,
jacobian]=lsqnonlin(...)
```

Он аналогичен синтаксису функции fsolve ( ), подробно описанной в разделе IV. Поэтому далее мы ограничимся только примером, демонстрирующим использование данной функции для нахождения параметров функции

$$F(x,a,b,c) = \exp(a + bx + cx^2).$$

Пример 7.3. Найти, используя метод наименьших квадратов, для данных, представленных в табл. VII.3, коэффициенты аппроксимирующей функции

$$F(x,a,b,c) = \exp(a + bx + cx^2).$$

Таблица VII.3

Исходные данные примера VII.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 VII.1), содержащий описание функции, возвращающей значения вектор-функции *f*x).

Листинг VII.1. Файл F77.m

```
function z=F77(Coeff,vx,vy)
k=l:length(vx);
z=vy-
exp(Coeff(1)+Coeff(2)*vx+Coeff(3)*vx.^2);
```

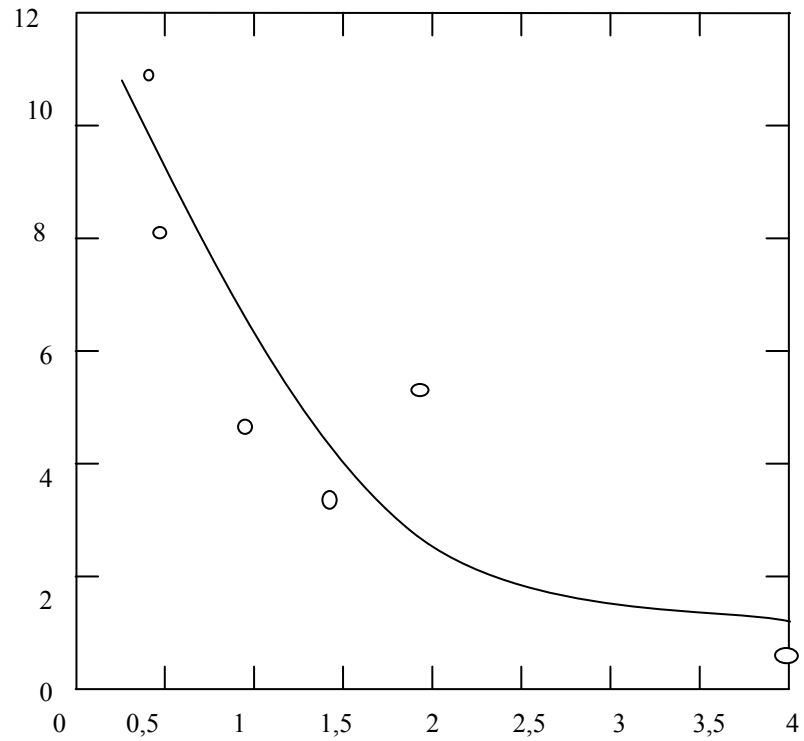



Рис. VII.4. Исходные данные и аппроксимирующая функция вида $F(x,a,b,c) = exp(a + bx + cx^2)$

2. Выполните следующую последовательность команд:

```
% задание исходных данных
» vx=[0.3;0.4;l;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=[l 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+Ь*x+c*x.^2)','x','a','b','c');
% задание
% аппроксимирующей функции
» X=vx(l):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=l:length(vx);
» j=1:length(X);
» plot(vx(i),vy(i),'o',X(j) ,Y(j))
% визуализация исходных данных и
% аппроксимирующей функции (рис. VII.4)
```


VIII. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Рассмотрим подходы и методы решения обыкновенных дифференциальных уравнений (методы Пикара, Эйлера, Эйлера-Коши, Рунге-Кутта), а также обсудим использование соответствующих функций пакета MATLAB.

8.1. Общие сведения и определения

Определение VIII.1. Дифференциальным уравнением 1-го порядка называют соотношение вида

$$F\left(x, y, \frac{dy}{dx}\right) = 0. \qquad \text{(VIII.1)}$$

Определение VIII.2. Дифференциальное уравнение вида

$$\frac{dy}{dx} = f(x, y), \qquad \text{(VIII.2)}$$

где *f(x,y)* — заданная функция двух переменных, называется дифференциальным уравнением первого порядка, разрешенным относительно производной.

Определение VIII.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 (VIII.1), (VIII.2), по определению получаем:

$$F\left(x, \varphi(x), \frac{d\varphi(x)}{dx}\right) = 0 \qquad \text{(VIII.3)}$$

$$\frac{d\varphi[x]}{dx} = f(x, \varphi(x)). \qquad \text{(VIII.4)}$$

Определение VIII.4. Соотношение (VIII.3) называется решением уравнения (VIII.4) в неявной форме (или интегралом уравнения (VIII.2)), если оно определяет y как функцию от x: $y = \varphi(x)$, которая есть решение уравнения (VIII .2).

Определение VIII.5. График решения $y = \varphi(x)$ уравнения (VIII.2) называется интегральной кривой данного уравнения.

Определение VIII.6. Проекция графика решения на ось ординат называется фазовой кривой или траекторией дифференциального уравнения.

Определение VIII.7. Задача о нахождении решения $y = \varphi(x)$ уравнения (VIII.2), удовлетворяющего начальному условию $\varphi(x_o)=y_o$, называется задачей Коши.

Определение VIII.*8.* Через каждую точку (x,y) из области определения уравнения (VIII.2) проведем прямую, тангенс угла которой к оси абсцисс равен $f(x,y)$. Данное семейство прямых называется *полем направлений, соответствующим уравнению* (VIII.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 (VIII.1), (VIII.2) обеспечивается теоремой Пикара.

Теорема Пикара. Если функция f определена и непрерывна в некоторой области G, определяемой неравенствами

$$|x - x_0| \le a, |y - y_0| \le b, \qquad \text{(VIII.5)}$$

и удовлетворяет в этой области условию Липшица по y:

$$|f(x, y_1) - f(x, y_2)| \le M|y_1 - y_2|, \qquad \text{(VIII.6)}$$

то на некотором отрезке $|x - x_o| \le h$, где h — положительное число, существует и притом только одно решение $y = y(x)$ уравнения (VIII.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

$$M = \max|f'_y(x)|. \qquad \text{(VIII.7)}$$

Определение VIII.9. Дифференциальным уравнением n-го порядка называют соотношение вида

$$F(x, y, y', y'', ..., y^{(k)}, ..., y^{(n)}) = 0, \qquad \text{(VIII.8)}$$

где x — независимая переменная, $y = y(x)$ — неизвестная функция аргумента x, $F(x, y, y', y'', ..., y^{(k)}, ..., y^{(n)})$ — заданная функция переменных $x, y, y', y'', ..., y^{(k)}, ..., y^{(n)}$.

Определение VIII.10. Задача о нахождении решения уравнения (VIII.8), удовлетворяющего начальным условиям

$$y(x_0) = y_0, y'(x_0) = y'_0, ..., y^{(n-1)}(x_0) = y_0^{(n-1)}, \qquad \text{(VIII.9)}$$

где y_0 , y'_0 , $y_0^{(n-1)}$, - заданные числа, называется задачей Коши для системы дифференциальных уравнений.

Разрешив дифференциальное уравнение (VIII.8) относительно производной $y^{(n)}$ и выполнив следующую замену переменных:

$$\begin{aligned} y' &= z_1, \\ y'' &= z_2, \\ &\dots \\ y^{(n-1)} &= z_{n-1}, \end{aligned} \qquad \text{(VIII.10)}$$

дифференциальное уравнение n-го порядка сводится к системе дифференциальных уравнений первого порядка:

$$\begin{aligned} y' &= z_1, \\ y'' &= z_2, \\ &\dots \\ y^{(n-1)} &= z_{n-1}, \\ z'_n &= f(x, y, z_1, z_2, ..., z_{n-1}). \end{aligned} \qquad \text{(VIII.11)}$$

Например, уравнение второго порядка

$$y'' = -\omega^2 y \qquad \text{(VIII.12)}$$

можно записать в виде двух уравнений

$$\begin{aligned} y' &= z, \\ z' &= -\omega^2 y. \end{aligned} \qquad \text{(VIII.13)}$$

Методы решений дифференциальных уравнений подразделяются на три основные группы:

- Аналитические методы решения.
- Графические методы.
- Численные методы.

8.2. Метод Пикара

Метод Пикара позволяет получить приближенное решение дифференциального уравнения (VIII.2) в виде функции, заданной аналитически.

Пусть в условиях теоремы существования требуется найти решения (VIII.2) с начальным условием $y_o = y(x_o)$. Запишем уравнение (VIII.1) в следующем эквивалентном виде:

$$dy = f(x, y)dx. \qquad \text{(VIII.14)}$$

Проинтегрируем обе части (VIII.14) от x_0 до x:

$$\int_{y_0}^{x} dy = \int_{x_0}^{x} f(x, y)dx. \qquad \text{(VIII.15)}$$

Вычислив интеграл в правой части, получим:

$$y(x) = y_0 + \int_{x_0}^{x} f(x, y)dx. \qquad \text{(VIII.16)}$$

Очевидно, что решение интегрального уравнения (VIII.16) будет удовлетворять дифференциальному уравнению (VIII.2) и начальному условию $y_o = y(x_o)$.

Действительно, при $x = x_0$ получим:

$$y(x_0) = y_0 + \int_{x_0}^{x} f(x, y)dx = y_0.$$

Интегральное уравнение (VIII.16) позволяет использовать

метод последовательных приближений. Положим $y = y_0$ и получим из (VIII.16) первое приближение:

$$y_1(x) = y_0 + \int_{x_0}^{x} f(x, y_0)dx. \qquad \text{(VIII.17)}$$

Интеграл, стоящий в правой части (VIII.17), содержит только переменную x, после нахождения этого интеграла будет получено аналитическое выражение приближения y_1 как функции переменной x. Заменим теперь в уравнении (VIII.16) y найденным значением, $y_1(x)$ и получим второе приближение:

$$y_2(x) = y_0 + \int_{x_0}^{x} f(x, y_1)\,dx \qquad \text{(VIII.18)}$$

и т. д.

В общем случае итерационная формула имеет вид:

$$y_n(x) = y_0 + \int_{x_0}^{x} f(x, y_{n-1})\,dx\,, \quad (n = 1, 2, ...). \qquad \text{(VIII.19)}$$

Последовательное применение формулы (VIII.19) дает последовательность функций:

$$y_1(x), y_2(x), ..., y_n(x), ... \qquad \text{(VIII.20)}$$

Так как функция f непрерывна в области G, то она ограничена в некоторой области $G' \subset G$, содержащей точку (x_o, y_0), т. е.

$$|f(x, y)| \le N. \qquad \text{(VIII.21)}$$

Применяя к уравнению (VIII.19) принцип сжимающих отображений, можно показать, что последовательность (VIII.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 (VIII.16), а, следовательно, и дифференциального уравнения (VIII.2) с начальными условиями $y_o = y(x_o)$. Это означает, что *к-й* член последовательности (VIII.20) является приближением к точному решению уравнения (VIII.2) с определенной степенью точности.

Оценка погрешности *к-го* приближения дается формулой:

$$\left|y(x)-y_k(x)\right| \le M^k N \frac{d^{k+1}}{(k+1)!}, \qquad \text{(VIII.22)}$$

где M — константа Липшица (IX.7), N — верхняя грань модуля функции f из неравенства (IX.21), а величина d для определения окрестности

$$|\, x\text{-}x_o \,| < d$$

вычисляется по формуле:

$$d = \min\left(a, \frac{b}{N}\right). \qquad \text{(VIII.23)}$$

8.3. Метод Эйлера

В основе метода Эйлера лежит идея графического построения решения дифференциальных уравнений (рис. VIII.1).

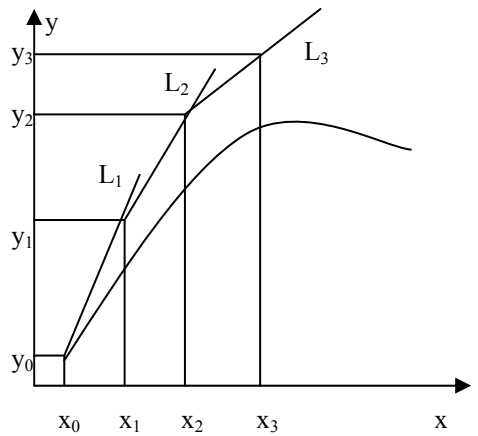


Рис. VIII.1. Графическая интерпретация метода Эйлера

Пусть дано уравнение (VIII.2) с начальным условием $y_0 = y(x_0)$. Выбрав достаточно малый шаг h, построим, начиная с точки x_0, систему равноотстоящих точек $x_i = x_0 + ih \quad (i = 0,1,2,...)$.

Вместо искомой интегральной кривой на отрезке $[x_o\ ,x_1]$ 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 (VIII.2), состоит в последовательном применении пары формул:

$$\Delta y_k = hf(x_k, y_k). \qquad \text{(VIII.24)}$$

$$y_{k+1} = y_k + \Delta y_k. \qquad \text{(VIII.25)}$$

Метод Эйлера, как видно из рис VIII.1, имеет погрешность. Найдем локальную погрешность, присутствующую на каждом шаге, которая определяется разностью между точным значением функции и соответствующим значением касательной. Для первого шага:

$$\Delta = y(x_1) - (y_0 + hf(x_0, y_0)) = y(x_0 + h) - (y_0 + hf(x_0, y_0)) =$$
$$y_0 + y'(x_0)h + y''(x_0)\frac{h^2}{2!} + \ldots - (y_0 + hf(x_0, y_0)) = y''(x_0)\frac{h^2}{2!} \qquad \text{(VIII.26)}$$

Из (VIII.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:

$$y''_{i+1} = y_i + h f(x_i, y_i),$$
$$y_{i+1} = y_i + h \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2}. \qquad \text{(VIII.27)}$$

Геометрически это означает, что определяется направление интегральной кривой в исходной точке (x_i, y_i) и во вспомогательной точке, $(x_{i+1}, y^{\bullet}_{i+1})$ а в качестве окончательного берется среднее значение этих направлений.

Пример VIII.*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 VIII.1), содержащий описание функции, возвращающей решение дифференциального уравнения методом Эйлера.

Листинг VIII.1. Файл Euler_g9.m

```
function [X,Y]=Euler_g9(y0,x0,xl,N)
dx=(xl-x0)/N;
x(1)=x0;
y(1)=y0;
for i=l:N
  x(i+l)=x(l)+dx*i;
  y(i+l)=y(i)+dx*F9(x(i));
end;
X=x;
Y=y;
function z=F9(x)
z=x.^2;
```

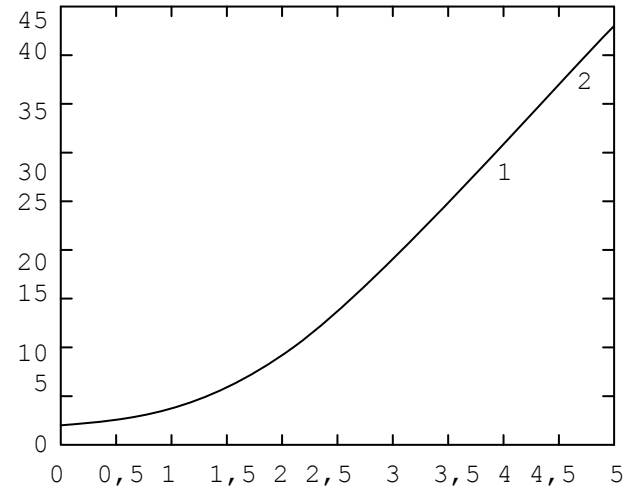



Рис. VIII.2. Визуализация точного (1) и численного (2) решений задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2$, $y(0) = 1{,}3$ методом Эйлера

• Выполните следующую последовательность команд:

```
» x0=0;   % левая граница отрезка интегрирования
» xl=5;   % правая граница отрезка интегрирования
» y0=1.3; % начальное условие
» N=50;   % число узлов разбиения отрезка интегрирования
» [X Y]=Euler_g9(y0,x0,xl,N); % нахождение численного
% решения задачи Коши
» i=l:length(X);
» Z(i)=yO+l/3*X(i).^3; % вычисление значений точного решения
```

» plot(X,Z,X,Y,':') % визуализация численного и точного решений
% (рис. VIII.2)
» plot(X,abs(Z-Y)) % визуализация разности между численным
% и точным решениями ДУ (рис. VIII.3)

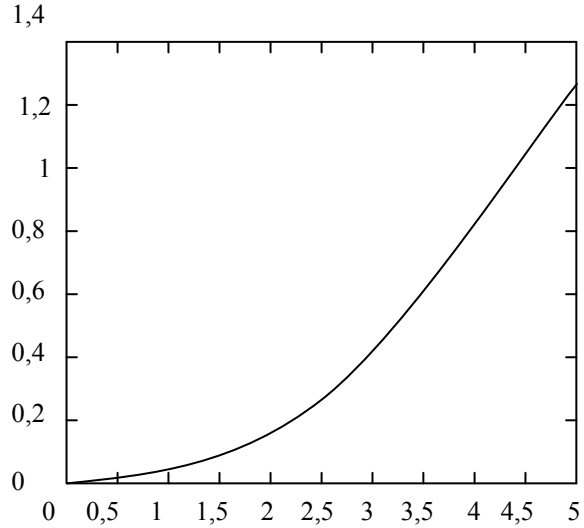


Рис. VIII.3. Разность между численным и точным решениями задачи Кош дифференциального уравнения $\frac{dy}{dx} = x^2, y(0) = 1{,}3$

2. Метод Эйлера-Коши.

- Создайте файл EulerKoshi.m (листинг VIII.2), содержащий описание функции, возвращающей решение дифференциального уравнения методом Эйлера-Коши.

Листинг_ VIII.2. Файл EulerKoshi.m

```
function  [X,Y]=EulerKoshi(y0,x0,xl,N)
dx=(xl-x0)/N;
x( l)=x0;
y( l)=y0;
for i=2:N
  x(i)=x(l)+dx*(i-l);
  Z=y(i-l)+dx*F9(x(i-l) ,y(i-l)) ;
```



```
  y(i)=y(i-l)+(F9(x(i-l),y(i-l))+F9(x(i),Z))*dx/2;
end;
X=x;
Y=y;
function z=F9(x,y); z=x.^2;
```

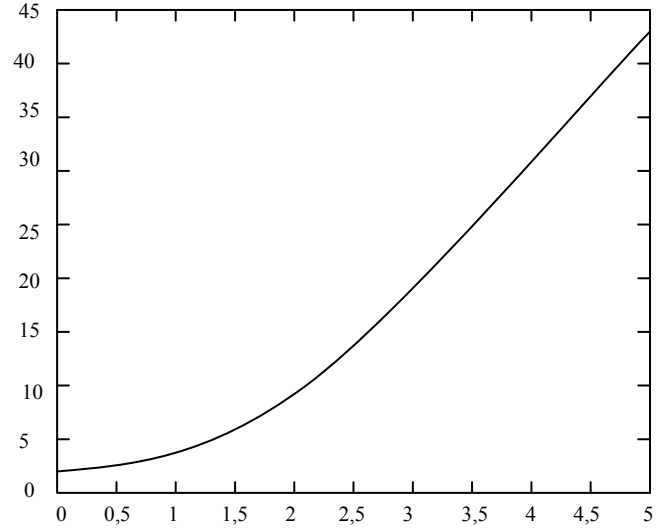


Рис. VIII.4. Визуализация точного и численного решений задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2, y(0) = 1{,}3$ методом Эйлера-Коши

- Выполните следующую последовательность команд:

```
» x0=0;%левая граница отрезка интегрирования
» xl=5;%правая граница отрезка интегрирования
» yO=1.3; % начальное условие
» N=50;% число узлов разбиения отрезка
% интегрирования
» [X Y]=EulerKoshi(y0,x0,xl,N);
   % нахождение численного решения задачи Коши
» i=l: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. VIII.4)
» plot(X,abs(Z-Y))
  % визуализация разности между численным и
  % точным решениями ДУ (рис. VIII.5)
```

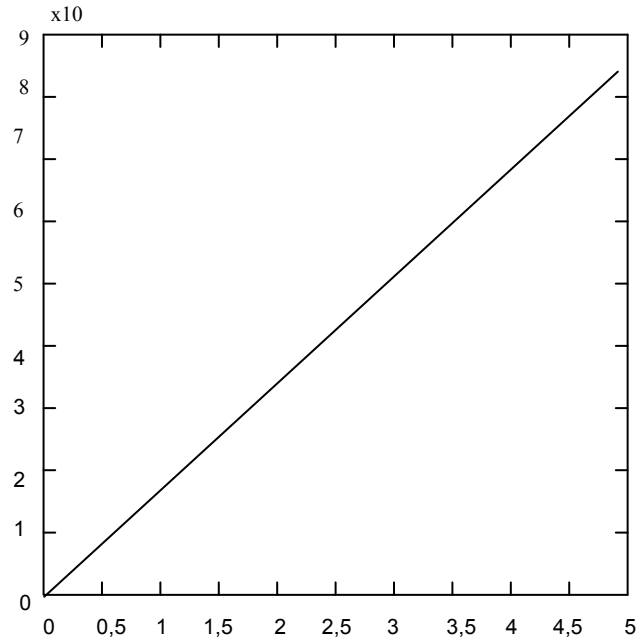


Рис. VIII.5. Разность между точным решением задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2, y(0) = 1{,}3$ и численным решением, полученным методом Эйлера-Коши

Из сравнения рис. VIII.3, VIII.5 видно, что погрешность, как и ожидалось, уменьшилась в 10^2 раз (h = 0.1).

8.4. 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:

$$\alpha_2, ..., \alpha_q;\ p_1, ..., p_q;\ \beta_{ij},\ 0 < j \le q.$$

Последовательно вычисляем:

$$k_1(h) = h \cdot f(x, y),$$
$$k_2(h) = h \cdot f(x + \alpha_2 \cdot h, y + \beta_{12} k_1(h)),$$
$$\dots$$
$$k_q(h) = h \cdot f(x + \alpha_q h, y + \beta_{q1} k_1(h) + ... \beta_{qq-1} k_{q-1}(h))$$

и полагаем:

$$y(x + h) = y(x) + \sum_{i=1}^{q} p_i k_i(h) = z(h). \qquad \text{(VIII.28)}$$

Рассмотрим вопрос о выборе параметров $\alpha_i, p_i, \beta_{ij}$. Обозначим

$$\varphi(h) = y(x + h) - z(h).$$

Будем предполагать, что

$$\varphi(0) = \varphi'(0) ... = \varphi^{(x)}(0) = 0,$$

а $\varphi^{(x+1)}(0) \neq 0$ для некоторой функции *f(x, y)*.

По формуле Тэйлора справедливо равенство

$$\varphi(h) = \sum_{i=0}^{s} \frac{\varphi^{(i)}(0)}{i!} h^i + \frac{\varphi^{(s+1)}(0h)}{(s+1)!} h^{s+1}, \qquad \text{(VIII.29)}$$

где $0 < \theta < 1$.

При q = 1 будем иметь:

$$\varphi(h) = y(x+h) - y(x) - p_1 \cdot h \cdot f(x,y),$$
$$\varphi(0) = 0,$$
$$\varphi'(0) = \left(y'(x+h) - p_1 f(x,y)\right)_{h=0},$$
$$\varphi''(h) = y''(x+h).$$

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 (VIII.28) получаются формулы (VIII.24), (VIII.25) метода Эйлера. Погрешность данного метода на шаге согласно (VIII.29) равна

$$\varphi(h) = \frac{\varphi''(x+h\theta)\cdot h^2}{2}.$$

Рассмотрим случай $q = 2$, тогда

$$\varphi(h) = y(x+h) - y(x) - p_1 h f(x,y) - p_2 h f\left(\bar{x}, \bar{y}\right),$$

где $\bar{x} = x + \alpha_2 h, \;\; y = y + \beta_{21} h f(x,y).$

Согласно исходному дифференциальному уравнению

$$y' = f,$$
$$y'' = f_x + \frac{df}{dy}\frac{dy}{dx} = f_x + f_y f(x,y),$$

$$y''' = f_{xx} + f_{xy} f + f\frac{df_y}{dx} + f_y \frac{df}{dx} =$$
$$= f_{xx} + f_{xy} f + f\left(f_{xy} + f_{yyf}\right) + f_y\left(f_x + f_y f\right) = \qquad \text{(VIII.30)}$$
$$= f_{xx} + 2 f_{xy} f + f_{yy} f^2 + f_y y''.$$

Вычисляя производные функции $\varphi(h)$ и, подставляя в выражения для $\varphi(h), \varphi'(h), \varphi''(h)$ значение $h = 0$, получаем:

$$\varphi(0) = 0,$$
$$\varphi'(0) = (1 - p_1 - p_2) f, \qquad \text{(VIII.31)}$$
$$\varphi''(0) = (1 - 2p_2\alpha_2) f_x + (1 - 2p_2\beta_{21}) f_y f.$$

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:

$$1 - p_1 - p_2 = 0,$$
$$1 - 2p_2\alpha_2 = 0, \qquad \text{(VIII.32)}$$
$$1 - 2p_2\beta_{21} = 0.$$

Задавая произвольно значения одного из параметров и определяя значения остальных параметров из системы (VIII.32), можно получать различные методы Рунге-Кутта с порядком погрешности $s = 2$. Например, при $p_1 = \frac{1}{2}$ из (VIII.32) получаем: $p_2 = \frac{1}{2}, \alpha_2 = 1, \beta_{21} = 1$

Для выбранных значений параметров формула (VIII.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 (VIII.27) метода Эйлера-Коши. Из (VIII.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:

$$\begin{aligned} k_1 &= h\cdot f(x,y),\\ k_2 &= h\cdot f\left(x+\frac{h}{2}, y+\frac{k_1}{2}\right),\\ k_3 &= h\cdot f\left(x+\frac{h}{2}, y+\frac{k_2}{2}\right),\\ k_4 &= h\cdot f(x+h, y+k_3),\\ \Delta y &= z(h)-y(x)=\frac{1}{6}(k_1+2k_2+2k_3+k_4). \end{aligned} \qquad \text{(VIII.33)}$$

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 углом 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 $\left(x_i+\frac{h}{2}, y_i+\frac{k_1}{2}\right)$. Наконец, из точки (x_i, y_i), сдвигаются в направлении, определяемом углом α_3, для которого

$$tg\alpha_3 = f\left(x_i+\frac{h}{2}, y_i+\frac{k_1}{2}\right),$$

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 (VIII.33). На этом окончательном направлении и выбирается очередная точка $(x_{i+1}, y_{i+1}) = (x_i+h, y_i+\Delta y)$

Пример VIII.2. Найти решение задачи Коши дифференциального уравнения

$$\frac{dy}{dx} = x^2,\ y(0)=1.3$$

методом Рунге-Кутта четвертого порядка.

1. Создайте файл RungeKutt4.m (листинг VIII.3), содержащий описание функции, возвращающей решение дифференциального уравнения методом Рунге-Кутта четвертого порядка.

Листинг VIII.3. Файл RungeKutt4,m

```
function [X,У]=RungeKutt4(y0,x0,xl,N)
dx=(xl-x0)/N;
x(l)=x0;
У(1)=У0;
for i=2:N
x(i)=x(l)+dx*(i-l);
kl=dx*F9(x(i-l),y(i-l));
k2=dx*F9(x(i-l)+dx/2,y(i-l)+kl/2);
```

```
k3=dx*F9(x(i-l)+dx/2,y(i-l)+k2/2);
k4=dx*F9(x(i-l)+dx,y(i-l)+k3);
y(i)=y(i-l)+l/6*(kl+2*k2+2*k3+k4); end; X=x; Y=y;
function z=F9(x,y) z=x.^2;
```

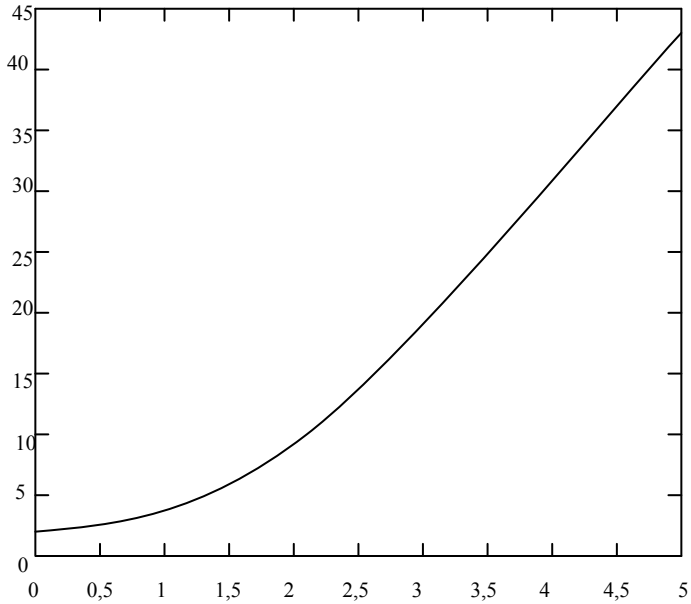


Рис. VIII.6. Визуализация точного решения задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2, y(0) = 1{,}3$ и численного решения, полученного методом Рунге-Кутта

2. Выполните следующую последовательность команд:

```
» x0=0;% левая граница отрезка интегрирования
» xl=5;%правая граница отрезка интегрирования
» yO=1.3;% начальное условие
» N=50; % число узлов разбиения отрезка
        % интегрирования
» [X Y]=RungeKutt4(y0,x0,xl,N);
```



```
  % нахождение численного решения задачи Коши
» i=l:length(X);
» Z(i)=y0+l/3*X(i).^3; % вычисление значений
                        % точного решения
» plot(X,Z,X,Y,':') % визуализация точного и
                    % численного решений
                    % (рис. VIII.6)
» plot(X,abs(Z-Y))% визуализация разности
% между численным и точным решениями ДУ
% (рис. VIII.7)
```

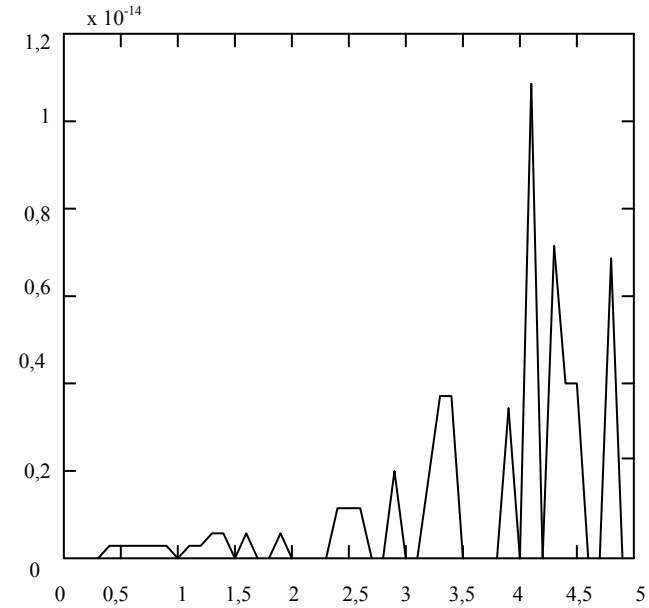


Рис. VIII.7. Разность между точным решением задачи Коши дифференциального уравнения $\frac{dy}{dx} = x^2, y(0) = 1{,}3$ и численным решением, полученным методом Рунге-Кутта

8.5. 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 $\bar{y}' = \bar{F}\left(t, \bar{y}\right)$.

Кроме того, решатели odel5s, ode23s, ode23t и ode23tb системы дифференциальных уравнений неявного вида

$$M\left(t, \bar{y}\right)\bar{y}' = \bar{F}\left(t, \bar{y}\right),$$

а также все решатели, кроме ode23s, могут находить решения уравнения вида $M\left(\bar{y}\right)\bar{y}' = \bar{F}\left(t,\ \bar{y}\right)$.

Пример VIII.3. Найти решение задачи Коши для дифференциального уравнения

$$\frac{dT}{dt} = -r\left(T - T_s\right), \qquad \text{(VIII.34)}$$

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 (VIII.34) сначала создайте файл Tempr.m (листинг VIII.4), содержащий определение функции, стоящей в правой части уравнения (VIII.1):

Листинг VIII.4. Файл Tempr.m

```
function Z=Tempr(t,T)
% определение функции, стоящей в правой части
% уравнения (VIII.1)
```

```
global Ts r
Z(l)=-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
» TO = 80     % задание начальной температуры
              % тела
» [t,T]=ode45('Tempr',[0:0.01:15],TO);
             % Tempr - имя файла, содержащего
             % определение функции,
% стоящей в правой части уравнения (VIII.1);
% [0:0.01:15] - вектор,
% определяющий интервал
% интегрирования,
% TO - 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. VIII.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',le-4,AbsTol',le-4);
```


предваряющая команду вызова функции решателя системы дифференциальных уравнений.

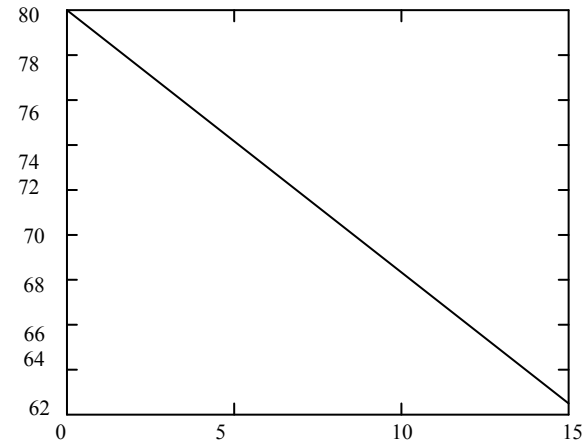


Рис. VIII.8. Численное решение уравнения теплопроводности, возвращенное функцией ode45

IX. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ В ЧАСТНЫХ ПРОИЗВОДНЫХ

Обсудим общие сведения и классификацию уравнений в частных производных (УЧП), рассмотрим явные и неявные разностные схемы для эллиптических, параболических и гиперболических уравнений и их программные реализации, продемонстрируем основные идеи использования метода Монте-Карло для решения УЧП.

9.1. Общие сведения и классификация уравнений в частных производных

Определение IX.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:

$$u_t = \frac{\partial u}{\partial t}, \quad S(t), \quad u_{xx} = \frac{\partial^2 u}{\partial x^2}.$$

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

$$Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu = G \quad , \qquad \text{(IX.1)}$$

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 (IX.1) называется *однородным,* если правая часть *C(x,y)* тождественно равна нулю для всех *x* и *y*. Если *C(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 (IX.1) — константы, то уравнение (IX.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).

9.2. 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)

$$u(1,\theta) = \sin\theta, \quad 0 \le \theta \le 2\pi.$$

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 заменяем выражением

$$f'(x) \cong \frac{f(x+h) - f(x)}{h}, \qquad \text{(IX.2)}$$

с точностью до членов порядка h аппроксимирующей производную в точке x.

Выражение, стоящее в правой части (IX.2), называется правой разностной производной.

В разложении Тейлора функции $f(x)$ можно заменить h на $-h$ и получить левую разностную производную:

$$f'(x) \cong \frac{f(x) - F(x-h)}{h}. \qquad \text{(IX.3)}$$

Складывая (IX.2) и (IX.3), получаем центральную разностную производную:

$$f'(x) \cong \frac{f(x+h) - f(x-h)}{2h}. \qquad \text{(IX.4)}$$

Поступая аналогично, можно получить центральную разностную производную — аппроксимацию второй производной:

$$f''(x) \cong \frac{1}{h^2}[f(x+h) - 2f(x) + f(x+h)]. \qquad \text{(IX.5)}$$

Теперь можно распространить понятие конечно-разностной аппроксимации на частные производные. Используя разложение функции двух переменных в ряд Тейлора

$$u(x+h, y) = u(x, y) + u_x(x, y)h + u_{xx}(x, y)\frac{h^2}{2!} + \ldots,$$

$$u(x-h, y) = u(x, y) - u_x(x, y)h + u_{xx}(x, y)\frac{h^2}{2!} - \ldots,$$

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. IX.1). Будем использовать следующие обозначения:

$$u(x_j, y_i) = u_{ij},$$

$$u(x_j, y_i + k) = u_{i+1,j},$$

$$u(x_j, y_i - k) = u_{i-1,j},$$

$$u(x_j - h, y_i) = u_{ij-1},$$

$$u(x_j + h, y_i) = u_{ij+1},$$

$$u_x(x_j, y_i) = \frac{1}{2h}(u_{i,j+1} - u_{i,j-1}),$$

$$u_y(x_j, y_i) = \frac{1}{2h}(u_{i+1,j} - u_{i-1,j}),$$

$$u_{xx}(x_j, y_i) = \frac{1}{h^2}(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}),$$

$$u_{yy}(x_j, y_i) = \frac{1}{k^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}).$$

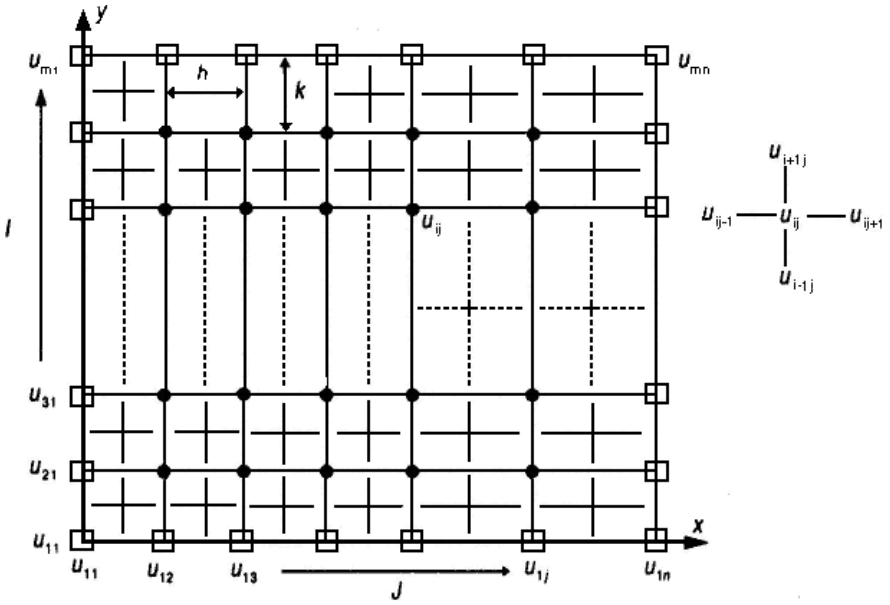


Рис. IX.1. Координатная сетка, используемая для решения УЧП эллиптического типа

Заменим частные производные в уравнении Лапласа их конечно-разностной аппроксимацией:

$$\frac{1}{h^2}(u_{i,j+1} - 2u_{i,j-1}) + \frac{1}{k^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) = 0. \qquad \text{(IX.6)}$$

Если k=h, то уравнение Лапласа приводится к виду

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij} = 0. \qquad \text{(IX.7)}$$

Разрешив (IX.7) относительно u_{ij}, получим

$$u_{ij} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}). \qquad \text{(IX.8)}$$

Отметим, что в (IX.8) все соотношения берутся для внутренних узлов сетки, так как значения функции на границе заданы. Соотношение (IX.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 (IX.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, $(I-1j)$-ом, $(i+1j)$-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 (IX.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 (IX.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, основанные на обсуждавшемся нами ранее методе Зейделя.

Пример IX.*1.* Найти в пакете MATLAB решение задачи Дирихле уравнения Лапласа

$$u_{xx} + u_{yy} = 0,$$

с граничными условиями

$$u(x,0) = 0, \quad u(1,y) = g_2(y), \quad u(x,1) = 0, \quad u(0,y) = g_4(y).$$

1. Создайте файл InterationL.m (листинг IX.1), содержащий описание функции, возвращающей численное решение уравнения Лапласа методом релаксаций (для получения

вычислительной схемы, обсуждение которой проводилось ранее, следует положить omega=1).

Листинг IX.1. Файл IterationL.m

```
function z=interationL(N,mega,Number_of_Interation,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+1j+1)); % релаксация
    end;
   end;
   if k==1
     q=phi;
   else
     q=cat (1,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 по левой границе
mu(i,N+1)= -10; % потенциал по правой границе
mu(1,j)=5; % потенциал по нижней границе
mu(N+1,j)=5; % потенциал по 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 К-го решения из общей %матрицы
решений
[C,h]=contour(x1,y1,A,17); % построение карты %эквипотенциальных
поверхностей К-го решения уравнения %Лапласа (рис. IX.2)
```

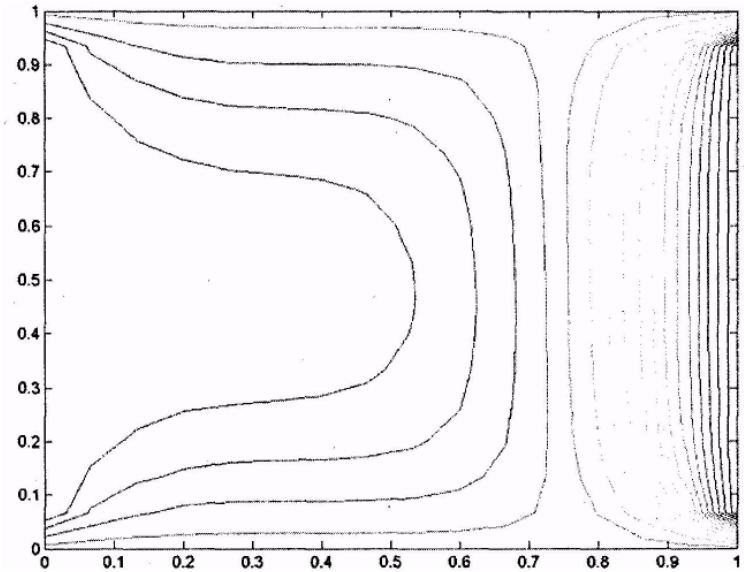


Рис. IX.2. Карта линий равного потенциала численного решения уравнения Лапласа

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 анимации 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 к, 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, 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 15x15 узлов (рис. IX.2). Из рис. IX.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 15x15 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. IX.3. Анализ карты эквипотенциальных поверхностей, представленных на рис. IX.2, IX.3, показывает, что, используя сплайн-интерполяцию, удалось устранить недостатки численного решения, проявляющиеся в наличии изломов линий равного потенциала.

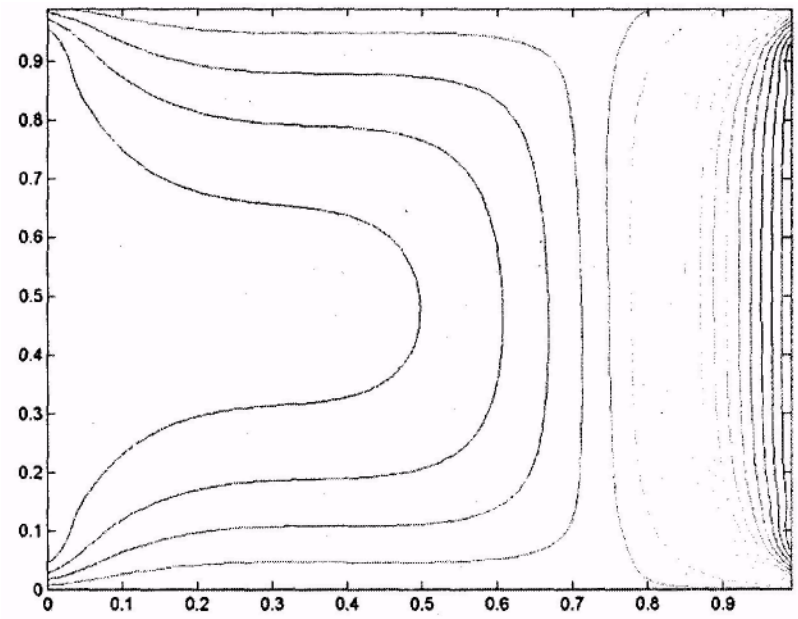


Рис. IX.3. Карта линий уровня интерполированного решения уравнения Лапласа

9.3. 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:

$$u_t = u_{xx}, \quad 0 < x < 1, \quad 0 < t < \infty, \qquad \text{(IX.9)}$$

с граничными условиями

$$\begin{cases} u = (0,t) = 1, & 0 < t < \infty, \\ u_x(1,t) = -[u(1,t) - g(t)], \end{cases}$$

с начальными условиями

$$u(x,0) = 0, \quad 0 \le x \le 1.$$

Рассматриваемое УЧП 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. IX.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 м,-,- 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:

$$u_t = \frac{1}{k}[u(x,t+k) - u(x,t)] = \frac{1}{k}[u_{i+1j} - u_{ij}],$$

$$u_{xx} = \frac{1}{h^2}[u(x+h,t) - 2u(x,t) + u(x-h,t)] = \frac{1}{h^2}[u_{ij-1} - 2u_{ij} + u_{ij-1}].$$

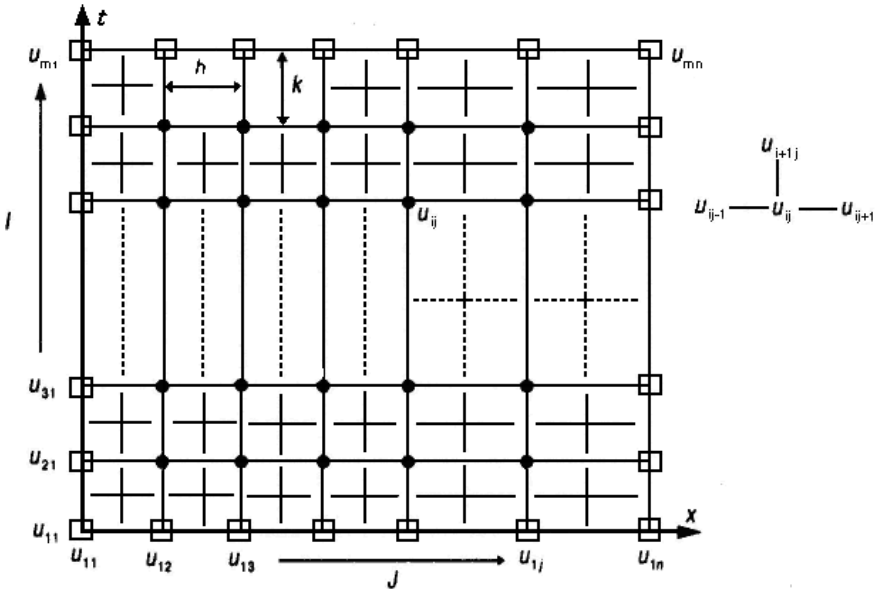


Рис. IX.4. Пространственно-временная сетка, используемая для решения одномерного уравнения теплопроводности

Подставим данные выражения в исходное уравнение $u_t = u_{xx}$ и разрешим получившееся уравнение относительно значений функции на верхнем временном слое. В результате получаем:

$$u_{i=1\,j} = u_{ij}\,\frac{k}{h^2}[u_{ij+1} - 2u_{ij} + u_{ij-1}]. \qquad \text{(IX.10)}$$

Формула (IX.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:

$$\frac{1}{h}[u_{in} - u_{in-1}] = -[u_{in} - g_i]. \qquad \text{(IX.11)}$$

Из (IX.11) находим:

$$u_{in} = \frac{u_{in-1} - hg_i}{1+h}. \qquad \text{(IX.12)}$$

Формулы (IX.10), (IX.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:

$$u_{2j} = u_{1j} + \frac{k}{h^2}[u_{1j+1} - 2u_{1j} + u_{1j-1}], \quad j = 2,3,\ldots,n-1 \qquad \text{(IX.13)}$$

2. Находим величину $u_{2л}$ по формуле (IX.12):

$$u_{2,n} = \frac{u_{2,n-1} + hg_2}{1+h}. \qquad \text{(IX.14)}$$

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 IX.2. Найти в пакете MATLAB решение краевой задачи уравнения теплопроводности (IX.8) с начальными условиями:

$$T(x,0) = \begin{cases} 0, \textit{если } x \neq 25 \\ 1, \textit{если } x = 25 \end{cases},$$

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
% соотношении 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. IX.5)
```

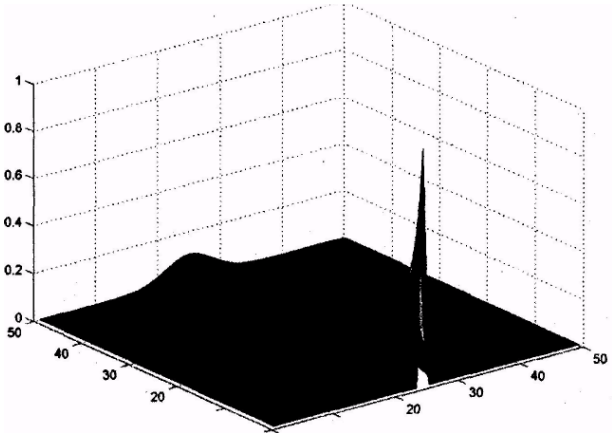


Рис. IX.5. Численное решение уравнения теплопроводности

Получим явную вычислительную схему для нахождения численного решения УЧП гиперболического типа, на примере волнового уравнения:

$$\frac{\partial^2 U}{\partial^2 x} = \frac{1}{\nu^2}\frac{\partial^2 U}{\partial t^2}. \qquad \text{(IX.15)}$$

Запишем уравнение (IX.15) в конечных разностях:

$$\frac{u_{ij+1} - 2u_{ij} + u_{ij-1}}{h^2} = \frac{1}{\nu^2}\frac{u_{i+1j} - 2u_{ij} + u_{i-1j}}{\tau^2}. \qquad \text{(IX.16)}$$

Полученное уравнение позволяет выразить значение функции u в момент времени t_{i+1} через значения функции в предыдущие моменты времени:

$$u_{i+1j} = \nu^2\left(\frac{\tau}{h}\right)^2\left(u_{ij+1} - 2u_{ij} + u_{ij-1}\right) + 2u_{ij} - u_{i-1j}. \qquad \text{(IX.17)}$$

Разностная схема (IX.17) устойчива, если $\tau \le h/\nu$.

Отметим, что для начала вычислений в соответствии с (IX.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. IX.6).

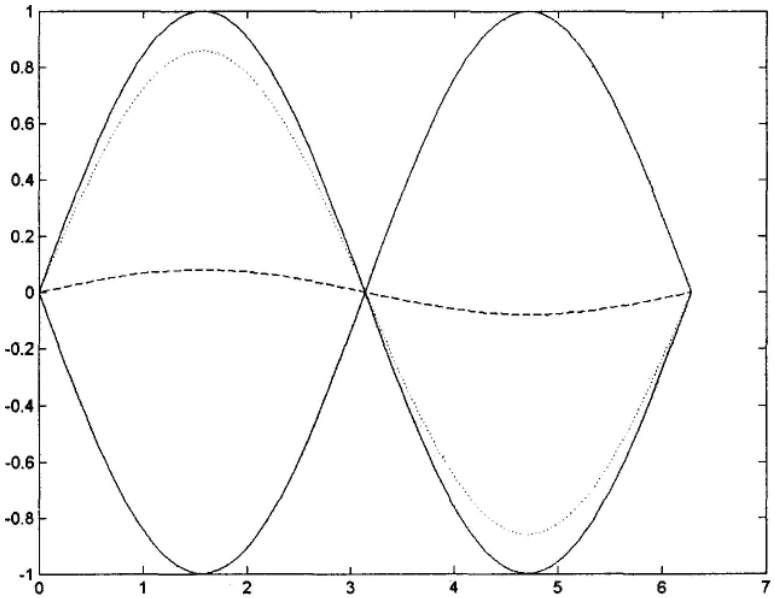


Рис. IX.6. Решение волнового уравнения в моменты времени t=0, 9, 24, 49

Пример IX.3. Найти в пакете MATLAB решение краевой задачи волнового уравнения (IX.14), в котором $\nu = 1, c$ начальными условиями:

$$U(x,0) = \sin(x),$$

где $x \in [0, 2\pi]$, и граничными условиями:

$$U(0,t) = 0, \quad U(2\pi,t) - 0$$

на временном интервале [0,59].

Решение:

```
>>Nt=60; % число шагов по времени
>>Nx=100; % число узлов по координате
% задание начальных и граничных условий
>> U(1,j)=sin(2*pi*(j-1)/(Nx-1)); % начальное условие
>> U(2,j)=U(1,j); % граничное условие
>> 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),'-')
```

9.4. Неявная разностная схема для уравнения параболического типа

Построим неявную разностную схему для уравнения теплопроводности. Рассмотрим задачу нахождения решения УЧП

$$u_t = u_{xx}, \quad 0 < x < 1, \quad 0 < t < \infty, \qquad \text{(IX.18)}$$

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}:

$$u_t(x,t) = \frac{1}{k}[u(x,t+k) - u(x,t)],$$

$$u_{xx}(x,t) = \frac{\lambda}{h^2}[u(x+h,t+k) - 2u(x,t+k) + u(x-h,t+k)] +$$

$$+\frac{1-\lambda}{h^2}[u(x+h,t) - 2u(x,t) + u(x-h,t),$$

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 (IX.18) получаем разностную задачу:

$$\frac{1}{k}(u_{i+1,j} - u_{i,j}) = \frac{\lambda}{h^2}(u_{i+1\,j+1} - 2u_{i+1\,j} + u_{i+1,j-1}) + \frac{1-\lambda}{h^2}(u_{ij+1} - 2u_{ij} + u_{ij-1}),$$

$$\begin{cases} u_{i,1} = 0, \\ u_{i,n} = 0, \end{cases} \quad i = 1,2,\ldots,m, \qquad \text{(IX.19)}$$

$$u_{1j} = 1, \quad j = 1,2,\ldots,n-1.$$

Перенесем все неизвестные значения u с верхнего временного слоя (с индексом $(i+1)$) в левую часть уравнения:

$$-\lambda r u_{i+j+1} + (1+2r\lambda)u_{i+1,\,j-1} = \\ = r(1-\lambda)\,u_{i,\,j+1} + [1-2r(1-\lambda)]u_{i\,j} + r(1-\lambda)u_{i\,j-1}\,, \qquad \text{(IX.20)}$$

где $r = k/h^2$.

Схема уравнения системы (IX.20).

	$j-1$	j	$j+1$
$i+1$	$-r\lambda$	$+1+1r\lambda$	$+\quad -r\lambda$
i	$r(1-\lambda)$	$+1-2r\lambda(1-\lambda)$	$+\quad r(1-\lambda)$

Если i фиксировано, а j изменяется от 2 до (n-1), соотношения (IX.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 дает таблица.

Обсудим более подробно алгоритм решения смешанной краевой задачи одномерного уравнения теплопроводности для случаев $\lambda = 0.5$ (схема Кранка-Никольсона), $h=\Delta=0.2$, $k=\Delta t = 0.08$ (при этом $r = k/h^2=2$). В данном случае сетка содержит 6 узлов вдоль оси x. В соответствии с вычислительной схемой (IX.20), двигаясь слева направо $j = 2, 3, 4, 5$ по первым двум слоям ($j=1$), получаем следующие четыре уравнения:

$$\begin{aligned} -u_{21} + 3u_{22} - u_{23} &= u_{11} - u_{12} + u_{13} = 1, \\ -u_{22} + 3u_{23} - u_{24} &= u_{12} - u_{13} + u_{14} = 1, \\ -u_{23} + 3u_{24} - u_{25} &= u_{13} - u_{14} + u_{15} = 1, \\ -u_{24} + 3u_{25} - u_{26} &= u_{14} - u_{15} + u_{16} = 1. \end{aligned} \qquad \text{(IX.21)}$$

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:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & . & . & . & . & 0 \\ a_1 & b_2 & c_2 & 0 & . & . & . & . & 0 \\ 0 & a_2 & b_3 & c_3 & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & c_{n-1} \\ . & . & . & . & . & . & . & a_{n-1} & b_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ . \\ . \\ . \\ d_n \end{bmatrix}.$$

Для нахождения ее решения преобразуем систему к эквивалентному виду:

$$\begin{bmatrix} 1 & c_1^* & 0 & 0 & . & . & . & . & 0 \\ 0 & 1 & c_2^* & 0 & . & . & . & . & 0 \\ 0 & 0 & 1 & c_3^* & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & c_{n-1}^* \\ . & . & . & . & . & . & . & a_{n-1} & b_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_n \end{bmatrix} = \begin{bmatrix} d_1^* \\ d_2^* \\ d_3^* \\ . \\ . \\ . \\ d_n^* \end{bmatrix},$$

где

$$c_1^* = c_1 / b_1 ,$$

$$c_{j+1}^* = \frac{c_{j+1}}{b_{j+1} - a_j c_j^*}, \quad j = 1,2,\ldots,n-2 ,$$

$$d_1^* = d_1 / b_1 ,$$

$$d_{j+1}^* = \frac{d_{j+1} - a_j d_j^*}{b_{j+1} - a_j c_j^*}.$$

Решение системы (IX.21) 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 $\lambda = 1$:

$$\begin{pmatrix} 1+2r & -r & & & \\ -r & 1+2r & -r & & \\ & \ddots & \ddots & \ddots & \\ & & r & 1+2r & -r \\ & & & -r & 1+2r \end{pmatrix} \begin{pmatrix} u_{i+1,2} \\ u_{i+1,3} \\ \vdots \\ u_{i+1,n-2} \\ u_{i+1,n-1} \end{pmatrix} = \begin{pmatrix} u_{i,1} + r \cdot u_{i,1} \\ u_{i,2} \\ \vdots \\ u_{i,n-2} \\ u_{i,n-1} + r \cdot u_{i,n} \end{pmatrix}, \quad (\text{IX.22})$$

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 IX.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 разности схемы (λ=1).

Для нахождения решения смешанной краевой задачи необходимо:

1. Создать файл U1.m (листинг IX.2), содержащий описание функции, возвращающей значения решения смешанной краевой задачи.

Листинг IX.2. Файл U1.m

```
function z=U1(u,A,r,Nt,Nx)
% u - матрица, содержащая начальные и граничные условия
% A – матрица системы (IX.22)
% r=k/h^2 – переменная, введенная в (IX.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 (IX.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
% визуализации решения смешанной краевой задачи (рис. IX.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).

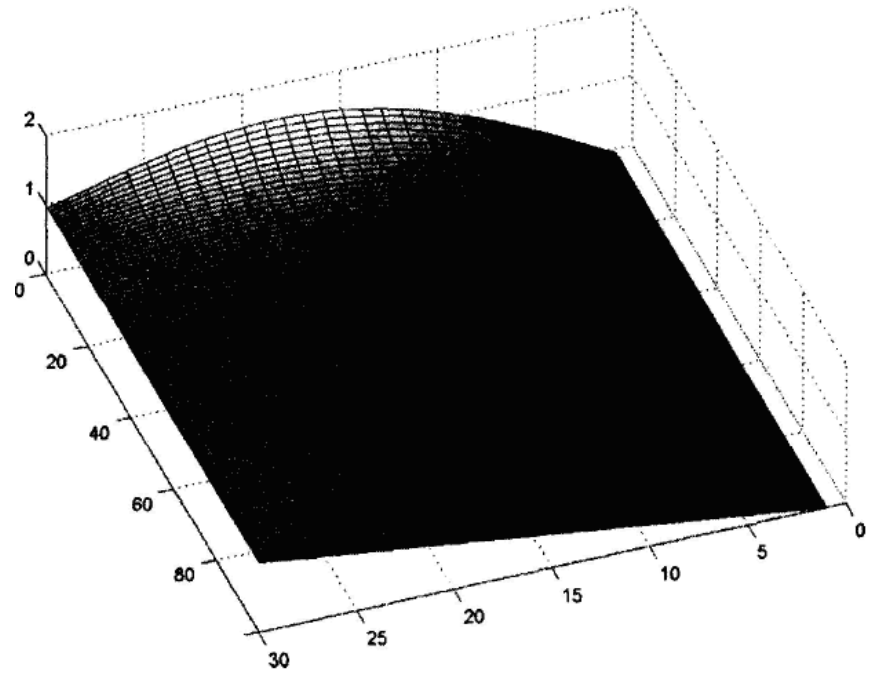


Рис. IX.7. Решение смешанной краевой задачи УЧП параболического типа

9.5. Решение уравнений с частными производными методом Монте-Карло

Рассмотрим решение задачи Дирихле уравнения Лапласа методом случайных блужданий:

$$u_{xx} + u_{yy} = 0,\ 0 < x < 1,\ 0 < y < 1.$$

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. IX.8)).

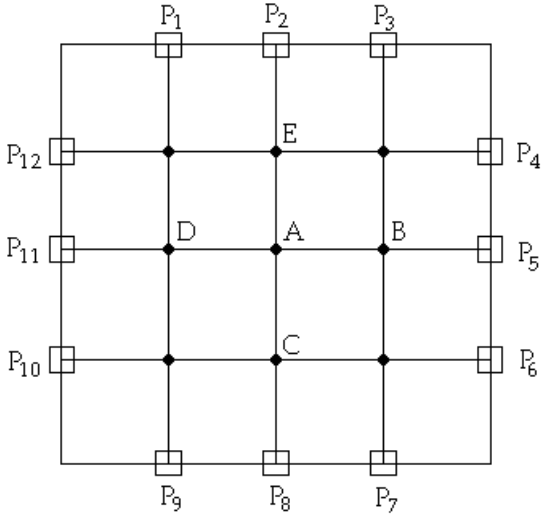


Рис. IX.8. Координатная сетка, используемая в методе Монте-Карло

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:

$$R(A) = g_1 P_A(p_1) + g_2 P_A(p_2) + \ldots + g_{12} P_A(p_{12}). \quad \text{(IX.23)}$$

Для чего надо играть в "Блуждающего пьяницу"?

Оказывается, что среднее вознаграждение (IX.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 *K(A)* будет средним арифметическим от средних вознаграждений для четырех соседних точек:

$$R(A) = \frac{1}{4}[R(B) + R(C) + R(D) + R(E)]. \quad \text{(IX.24)}$$

Итак, мы видим, что *R(A)* удовлетворяет уравнению (IX.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

$$u_{ij} = \frac{1}{4}[u_{i-1j} + u_{i+1j} + u_{ij-1} + u_{i,j+1}], \quad \text{(IX.25)}$$

где (i, j) – внутренняя точка,

$$u_{ij} = g_{ij}, \quad \text{(IX.26)}$$

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 IX.5. Решение методом случайных блужданий в пакете MATLAB уравнения Лапласа:

$$u_{xx} + u_{yy} = 0, \quad 0 < x < 1, \quad 0 < y < 1,$$
$$u(0, y) = 0, u(1, y) = 0, \quad u(x,0) = 5, u(x,1) = 0.$$

1. Создайте файл Моп1е_Каг1о.т (листинг IX.3), содержащий описание функции, возвращающей численное решение уравнения Лапласа.

Листинг IX.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;
```

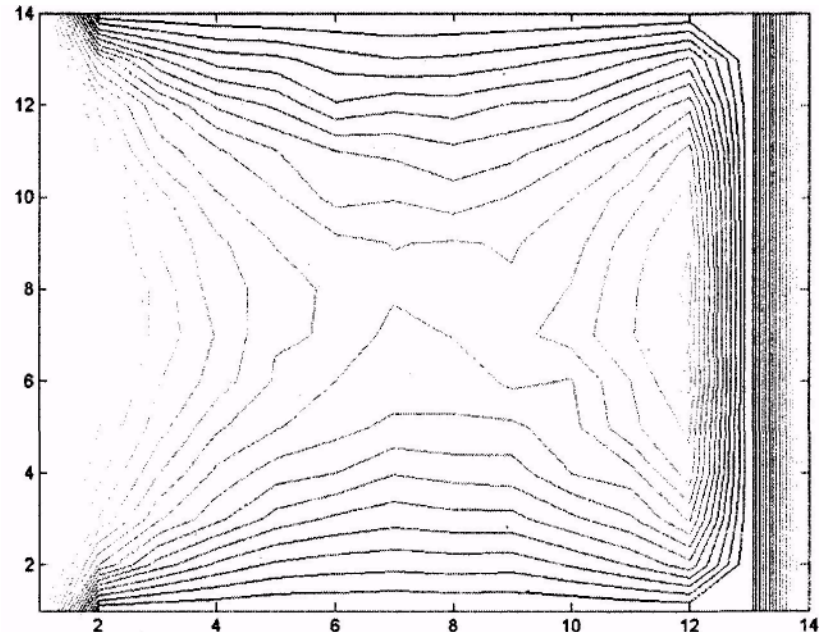


Рис. IX.9. Карта линий уровня решения уравнения Лапласа, полученного методом Монте-Карло

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. IX.9)
```


Для сравнения на рис IX.10 представлено решение уравнения Лапласа, полученное методом релаксаций.

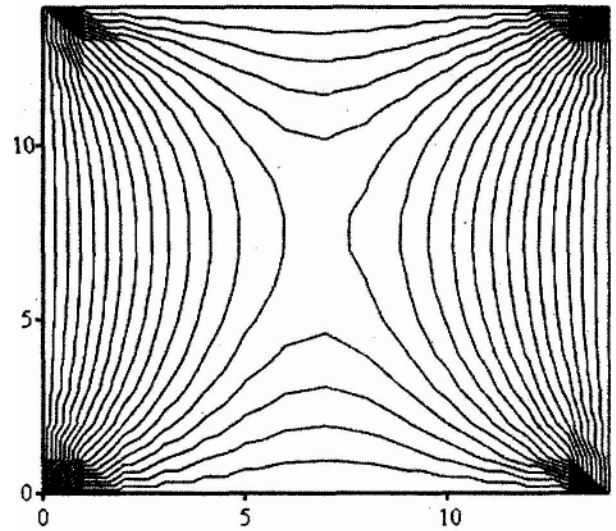


Рис. IX.10. Решение уравнения Лапласа, полученное методом релаксаций

Рассмотрим эллиптическую краевую задачу в квадрате

$$u_{xx} + (\sin x)u_{xx} = 0,\ 0 < x < \pi,\ 0 < x < \pi,$$

$$u(x, y) = g(x, y)$$

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

$$u_{ij} = \frac{u_{ij+1} + u_{i,j-1} - \sin x_j (u_{i+1j} + u_{i-1j})}{2(1 + \sin x_j)}. \qquad \text{(IX.27)}$$

Коэффициенты при u_{i+1j} ,u_{ij+1} ,u_{ij-1} ,u_{i-1j} в (IX.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 $\frac{1}{2(1+\sin x_j)}$,
- (*i, j*-1) с вероятностью $\frac{1}{2(1+\sin x_j)}$,
- (*i*+1,*j*) с вероятностью $\frac{\sin x_j}{2(1+\sin x_j)}$,
- (*i*-1,*j*) с вероятностью $\frac{\sin x_j}{2(1+\sin x_j)}$.

В остальном правила игры не меняются. Модификация для других задач может оказаться более хитроумной, однако основная идея метода не меняется.

X. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ

Обсудим общие сведения и классификацию интегральных уравнений, рассмотрим квадратурные методы решения интегральных уравнений Фредгольма и Вольтерра и их программные реализации.

10.1. Общие сведения об интегральных уравнениях

Определение X.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 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:

$$x(t) = \int_D K(t, s, x(s))\, ds + f(t), \qquad \text{(X.1)}$$

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.

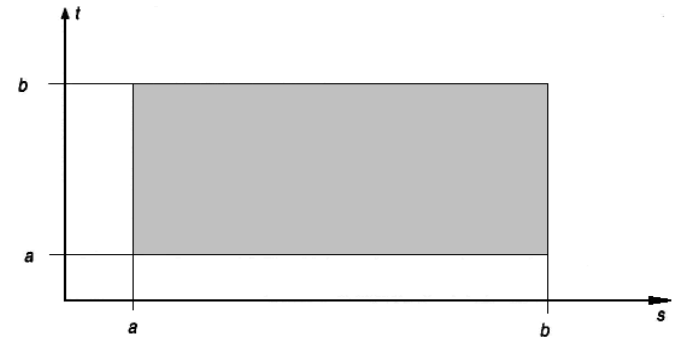
Далее мы ограничим рассмотрение одномерными линейными интегральными уравнениями, в которых функция $x(t)$ является функцией, зависящей от одной переменной, а область D —

отрезком конечной длины. В этих уравнениях подынтегральная функция *K(t,s,x(s))* представима в виде *Q*(*t,s,x(s)*).

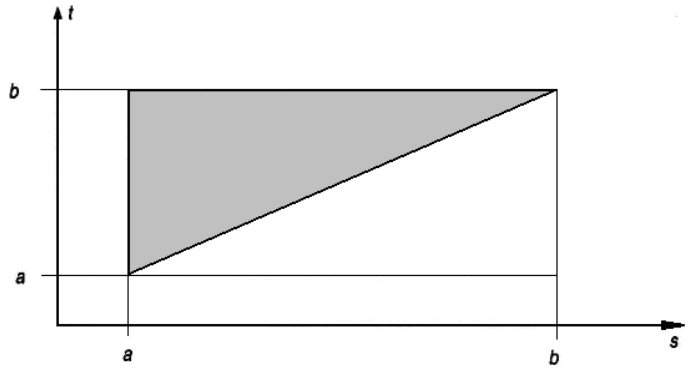
Классификация типов линейных интегральных уравнений проводится по виду верхней границы интеграла в (X.1): если верхняя граница интегрирования является постоянной, то уравнение называется *уравнением Фредгольма,* если переменной — *уравнением Вольтерра,* которые, в свою очередь, подразделяются на уравнения первого и второго рода. На практике наиболее широко применяются линейные интегральные уравнения второго рода:

- Фредгольма

$$x(t) = \lambda \int_a^b Q(t,s)x(s)ds + f(t); \qquad (X.2)$$



а



б

Рис. X.1. Области задания ядер Q(t,s) интегральных уравнений Фредгольма (а) и Вольтера (б)

- Вольтерра

$$x(t) = \lambda \int_a^t Q(t,s)x(s)ds + f(t), \qquad (X.3)$$

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. X. 1(а)), уравнения Вольтерра — в треугольнике $a \le a \le 1 \le b$ (рис. X.1(б)).

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 (X.1), (X.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

$$\int_a^b Q(t,s)x(s)ds = f(t); \qquad (X.4)$$

- Вольтерра

$$\int_a^t Q(t,s)x(s)ds = f(t). \qquad \text{(X.5)}$$

Эта задача относится к классу некорректных задач.

Уравнения первого и второго рода можно записать в общем виде, используя функцию *h(1),* тожественно равную нулю для уравнений первого рода и единице для уравнений второго рода:

$$h(t)x(t) = \lambda \int_D Q(t,s)x(s)ds + f(t). \qquad \text{(X.6)}$$

Когда функция *h(t)* обращается в ноль в некоторых точках прямоугольника интегрирования, уравнение (X.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 (X.5). Для уравнения Фредгольма второго рода (X.4) при тех же требованиях единственное непрерывное решение существует, например, при условии, что

$$|\lambda| < \frac{1}{C(b-a)}, \qquad \text{(X.7)}$$

где $C = \max_{t,s\in[a,b]} |Q(t,s)|$.

При снижении требований к гладкости возможных решений условие (X.7) ослабляется. Например, для функций, интегрируемых с квадратом, в роли достаточного условия фигурирует неравенство

$$|\lambda| < \frac{1}{\sqrt{\int_a^b \int_a^b Q^2(t,s)\,dt\,ds}}. \qquad \text{(X.8)}$$

Известны формулы (или совокупности формул) позволяющие найти точное решение *x*(*t*). Например, решение уравнения Вольтерра с λ = 1 и мультипликативным ядром

$$Q(t,s) = p(t)s(t) \qquad \text{(X.9)}$$

вычисляется по формуле

$$x(t) = \int_a^t R(t,s)f(s)ds + f(t), \qquad \text{(X.10)}$$

где

$$R(t,s) = p(t)q(s)e^{\int_s^t p(\xi)q(\xi)d\xi}. \qquad \text{(X.11)}$$

Решение уравнения Фредгольма с вырожденным ядром

$$Q(t,s) = \sum_{i=1}^m p_i(t)q_i(s) \qquad \text{(X.12)}$$

имеет вид

$$x(t) = \lambda \sum_{i=1}^m z_i p_i(t) + f(t), \qquad \text{(X.13)}$$

где числа z_i — решения системы линейных алгебраических уравнений:

$$\begin{cases} z_i - \lambda c_{11}z_1 - \lambda c_{12}z_2 - \ldots - \lambda c_{1m}z_m = d_1 \\ z_2 - \lambda c_{21}z_1 - \lambda c_{22}z_2 - \ldots - \lambda c_{2m}z_m = d_2 \\ \vdots \\ z_1 - \lambda s_{m1}z_1 - \lambda c_{m2}z_2 - \ldots - \lambda c_{mm}z_m = d_m \end{cases}, \qquad \text{(X.14)}$$

где $c_{ij} = \int_a^b q_i(s)p_i(s)ds,\ d_i = \int_a^b q_i(s)f(s)ds.$

Условие существования и единственности решения уравнения Фредгольма с вырожденным ядром, как очевидно, зависит от значения определителя D(λ) системы линейных алгебраических

уравнений (X.14), называемого *определителем Фредгольма.* Если D(λ)≠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.

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. Отметим, 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.

10.2. Квадратурный метод решения интегральных уравнений Фредгольма

Заменим определенный интеграл (X.4) его приближенным значением, вычисляемым с помощью конкретной квадратурной формулы

$$\int \varphi(s)ds \approx \sum_{j}^{n} A_j \varphi(s_j), \qquad \text{(X.15)}$$

где j=1n — номера узлов сетки по переменной, A_j — весовые коэффициенты квадратурной формулы.

Подставив правую часть приближенного равенства (X.15) с $\varphi(s) = Q(t,s)\cdot x(s)$ вместо интеграла в интегральное уравнение Фредгольма второго рода (X.4), получим

$$x(t) \approx \lambda \sum_{j=1}^{n} A_j Q(t, s_j) x(s_j) + f(t). \qquad \text{(X.16)}$$

Выражение (X.16) задает функцию, описывающую приближенное решение интегрального уравнения (X.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

$$x(t_i) \approx \lambda \sum_{j=1}^{n} A_j Q(t_i, s_j) x(s_j) + f(t_i). \qquad \text{(X.17)}$$

где $i = 1,n$.

Введем обозначения $Q_{ij} = Q(t_i, s_j)$, $f_i = f(t_i)$, $x_i = x(t_i)$ и запишем (X.17) в виде системы n линейных алгебраических уравнений с n неизвестными:

$$\begin{cases} (1-\lambda A_1 Q_{11})x_1 - \lambda A_2 Q_{12} x_1 - \ldots - \lambda A_n Q_{1n} x_n = f_1, \\ (1-\lambda A_2 Q_{21})x_1 - \lambda A_2 Q_{22} x_2 - \ldots - \lambda A_n Q_{2n} x_n = f_2, \\ \vdots \\ (1-\lambda A_n Q_{n1})x_1 - \lambda A_2 Q_{n2} x_2 - \ldots - \lambda A_n Q_{nn} x_n = f_n. \end{cases} \qquad \text{(X.18)}$$

для решения которой можно использовать любой из методов, подробно обсуждавшихся в главе *III.*

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 X.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 X.1), содержащий описание функции, возвращающей значения функции *Q(t,s)*.

Листинг X.1. Файл q11.m

```
function z=Q(t,s)
z=1./sqrt(t+s.^2)
```

2. Создайте файл F11.m (листинг X.2), содержащий описание функции, возвращающей значения функции *f*(*t*).

Листинг X.2. Файл f11.m

```
function z=f11(t)
z=sqrt(t+1)-sqrt(t+4)+t;
```

3. Создайте файл Solve_g11.m (листинг X.3), содержащий описание функции, возвращающей решение интегрального уравнения.

Листинг X.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
>> [N,X]=solve_g11(x1,x2,N,Lambda);
>> plot(X,Y)  % визуализация решения интегрального уравнения
              % (рис.X.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. 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.

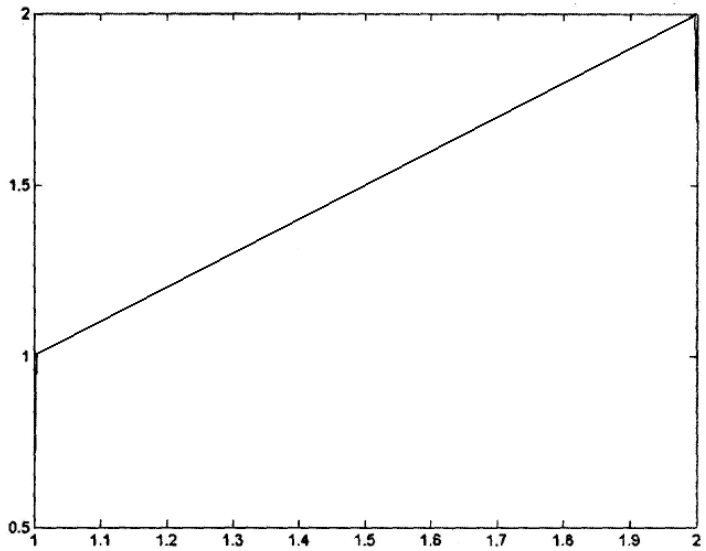


Рис. X.2. Численное решение интегрального уравнения $x(t)=\int_1^2 \frac{x(s)ds}{\sqrt{t+s^2}}+\sqrt{t+1}-\sqrt{t+4}+t.$

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. X.3, на котором представлено решение уравнения из примера X.1, полученное при использовании квадратурной формулы Симпсона. Далее приведен листинг файла Solve2_g11.m (листинг X.4), содержащего описание соответствующей функции.

Листинг X.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, Solve1_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);
>> plot(X,Y)  % визуализация решения интегрального уравнения
              % (рис. X.3)
```

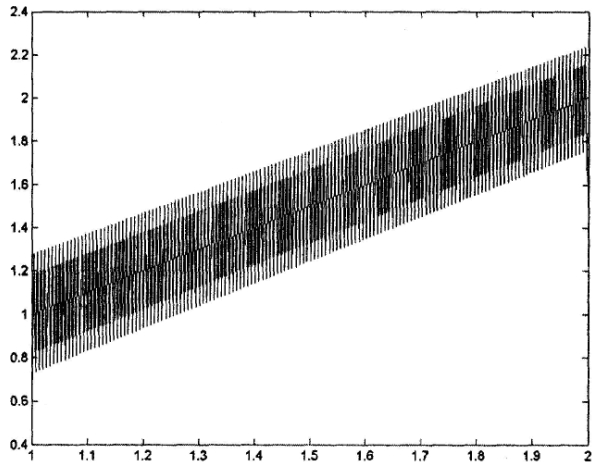


Рис. X.3. Численное решение интегрального уравнения $x(t)=\int_1^2 \frac{x(s)ds}{\sqrt{t+s^2}}+\sqrt{t+1}-\sqrt{t+4}+t$ при использовании формулы Симпсона

10.3. Квадратурный метод решения интегральных уравнений Вольтера

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

$$x(t)=\int_a^t Q(t,s)x(s)ds+f(t), \qquad \text{(X.19)}$$

где $t\in[a,\ b]$, можно положить $\lambda=1$.

Учитывая, что уравнение Вольтерра формально можно считать уравнением Фредгольма вида

$$x(t)=\int_a^t Q(t,s)x(s)ds+f(t) \qquad \text{(X.20)}$$

с ядром

$$K(t,s)=\begin{cases}Q(t,s)\ при\ a\le s\le t\le b\\ 0\ при\ a\le t\le s\le b\end{cases}, \qquad \text{(X.21)}$$

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 (X.17). Составим систему линейных алгебраических уравнений, аналогичную системе (X.18), которая в силу свойств ядра интегрального уравнения (X.21) вырождается в треугольную:

$$\begin{cases}(1-A_1Q_{11})x_1=f_1\\ -A_1Q_{21}x_1+(1-A_2Q_{22})x_2=f_2\\ \vdots\\ -A_1Q_{n1}x_1-A_2Q_{n2}x_2-\ldots+(1-A_nQ_{nn})x_n=f_n.\end{cases} \qquad \text{(X.22)}$$

Из (X.22) видно, что искомые значения x_1, x_2,…,x_n находятся последовательными вычислениями по следующим формулам:

$$x_1 = \frac{f_1}{1 - A_1 Q_{11}}, \qquad \text{(X.23)}$$

$$x_i = \frac{f_i + \sum_{j=1}^{i-1} A_j Q_{ij} x_j}{1 - A_i Q_{ii}}, \qquad \text{(X.24)}$$

Пример X.2. Решение в пакете MATLAB интегрального уравнения

$$x(t) = \int_3 t\cos^2(ts^3)x(s)ds + t^2 - \frac{1}{3}tg(t^4).$$

(Точное решение уравнения $x = t^2$.)

1. Создайте файл q11_2.m (листингX.5), содержащий описание функции, возвращающей значения подынтегральной функции.

Листинг X.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 X.6), содержащий описание функции, возвращающей значения функции *f*(*t*).

Листинг X.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 X.7), содержащий описание функции, возвращающей решение интегрального уравнения.

Листинг X.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);
>> plot(X,Y) % визуализация решения интегрального уравнения
                % (рис. X.4)
```

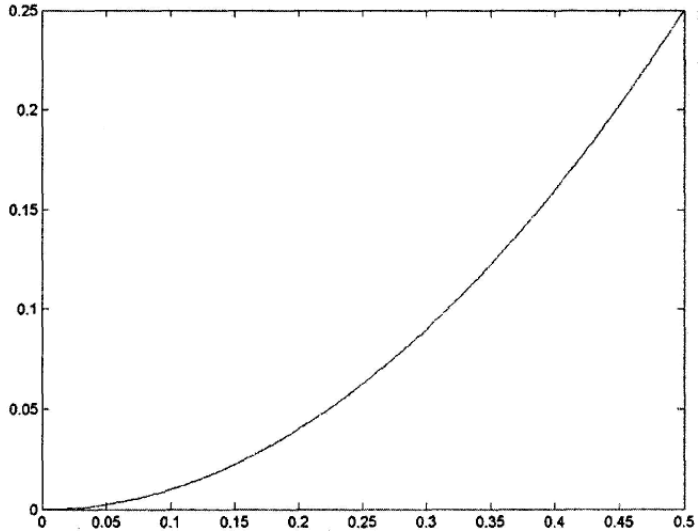


Рис. X.4. Численное решение интегрального уравнения

$$x(t) = \int_0^t t\cos^2(ts^3)x(s)ds + t^2 - \frac{1}{3}tg(t^4).$$

```
% 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 (X.5) при использовании метода средних прямоугольников. Решения уравнения будем находить в узлах временной сетки

$$t_1 = a + h,\ t_2 = t_1 + h, \ldots,\ t_i = t_{i-1} + h. \qquad \text{(X.25)}$$

Подставляя (X.25) в (X.4), получаем равенства

$$\int_a^{t_i} Q(T_1, s)x(s)ds = f(t_i). \qquad \text{(X.26)}$$

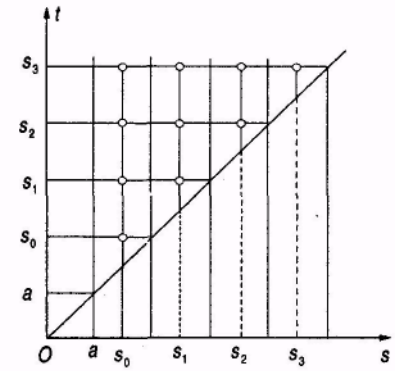


Рис. X.5. Пространственно-временная сетка, используемая для решения уравнения Вольтерра

Из (X.26) 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. X.5). Выбор данной сетки означает, что

$$x_1 \approx x(s_1), x_2 \approx x(s_2), \ldots, x_n \approx x(s_n). \qquad \text{(X.27)}$$

Учитывая выбор квадратурной формулы и условия (X.27), запишем (X.26) в следующем виде:

$$hQ(t_1, s_1)x_1 = f(t_1),$$
$$hQ(t_2, s_1)x_1 + hQ(t_2, s_2)x_2 = f(t_2),$$

$$hQ(t_3,s_1)x_1 + hQ(t_3,s_2)x_2 + hQ(t_3,s_3)x_3 = f(t_3), \quad \text{(X.28)}$$

…,

где $$s_i = t_i - \frac{h}{2}.$$

Из равенств (X.28) последовательно находим:

$$x_1 = \frac{f(t_1)}{hQ(t_1,s_1)}, \quad \text{(X.29)}$$

$$x_i = \frac{f(t_i) - h\sum_{i=1}^{i-1} Q(t_i,s_j)x_j}{hQ(t_i,s_i)}, \quad \text{(X.30)}$$

где $i = 2, 3,$

Пример X.4. Найти в пакете МАТЬАВ решение интегрального уравнения

$$\int_1^t (t^2 + s^2 + 1)x(s)ds = t^2 - \frac{1}{t},$$

используя систему узлов координатно-временной стеки, представленную на рис. X.5.

(Точное решение уравнения $x(t) = \frac{1}{t^2}$.)

1. Создайте файл F11_4.m (листинг X.8), содержащий описание функции, возвращающей значения функции f(*t*).

Листинг X.8. Файл F11_4.m

```
function z=F11_4(t)
z=t.^2-1./t;
```

2. Создайте файл Q11_4.m (листинг X.9), содержащий описание функции, возвращающей значения функции *Q(t,s).*

Листинг X.9. Файл Q11_4.m

```
function z=Q11_4(t,s)
z=t.^2+s.^2+1;
```

3. Создайте файл УоИеггаП.т (листинг X.10), содержащий описание функции, возвращающей решения интегрального уравнения Вольтерра.

Листинг X.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(1));
% Вычисление значений решения интегрального
уравнения в соответствие с (X.29),(X.30)
x(1)=F(1)/(h*q(1,1));
for m=2:N
   s=0;
   for j=2:m-1
     s=s+q(m,j)*m(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. X.6)
```

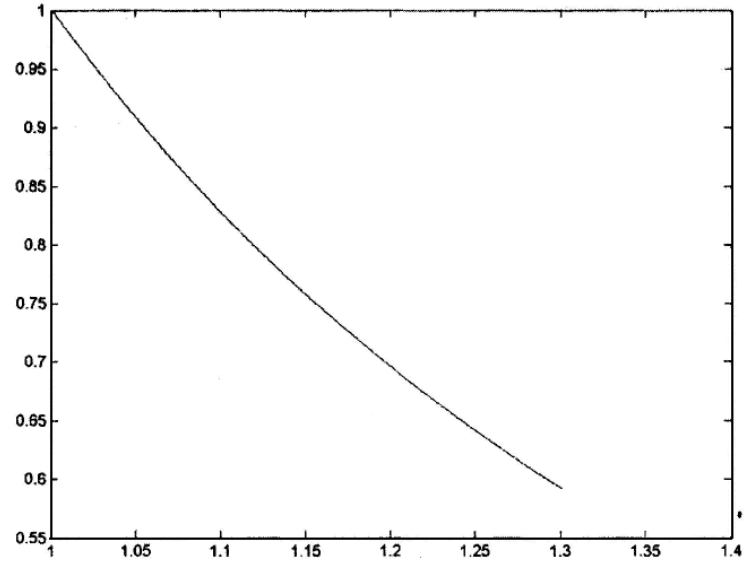


Рис. X.6. Численное решение интегрального уравнения $\int_1^t (t^2+s^2+1)x(s)ds = t^2 - \frac{1}{t}$, полученное на пространственно-временной сетке, представленной на рис X.5.

При использовании квадратурных формул замкнутого типа и совпадающей системы узлов возникает проблема вычисления значения $x_1=x(t_1)=x(s_1)=x(a)$, которая не возникала при решении уравнений Вольтерра второго рода. Действительно, при i=1 равенство (X.25) теряет всякий смысл.

Для нахождения начального значения x_1 продифференцируем уравнение (X.5) по t.

$$Q(t,t)x(t) + \int_a^t Q'(t,s)x(s)ds = f'(t). \qquad \text{(X.31)}$$

Положив в (X.31) $t= a$, получим

$$Q(a,a)x(a) = f'(a). \qquad \text{(X.32)}$$

Откуда следует, что

$$x(a) + \frac{f'(a)}{Q(a,a)},$$

следовательно,

$$x_1 = \frac{f'(a)}{Q_{11}}. \qquad \text{(X.33)}$$

При использовании квадратурной формулы трапеций далее получаем:

$$\frac{h}{2}Q_{21}x_1 + \frac{h}{2}Q_{22}x_2 = f_2 \Rightarrow x_2 = \frac{f_2 \dfrac{h}{2}Q_{21}x_1}{\dfrac{h}{2}Q_{22}},$$

$$\frac{h}{2}Q_{31}x_1 + \frac{h}{2}Q_{32}x_2 + \frac{h}{2}Q_{33}x_3 = f_3 \Rightarrow x_3 = \frac{f_3 \dfrac{h}{2}Q_{31}x_1 - hQ_{32}x_2}{\dfrac{h}{2}Q_{33}},$$

т. е. в общем случае при любому j= 2, 3,

$$x(s_j) \approx x_j = \frac{f_j - \dfrac{h}{2}Q_{j1}x_1 - h\sum_{k=2}^{j-1}Q_{jk}x_k}{\dfrac{h}{2}Q_{jj}}. \qquad \text{(X.34)}$$

Пример X.4. Найти в пакете MAT1AB решение интегрального уравнения

$$\int_1^t (t^2+s^2+1)x(s)ds = t^2 - \frac{1}{t},$$

используя замкнутые квадратурные формулы.

1. Создайте файл F11_4.m (листинг X.11), содержащий описание функции, возвращающей значения функции $f(t)$.

Листинг X.11. Файл F11_4.m

```
function z=F11_4(t)
z=t.^2-1./t;
```

2. Создайте файл dF11_4.m (листинг X.12), содержащий описание функции, возвращающей значения производной функции *f(t)*.

Листинг X.12. Файл dF11_4.m

```
function z=dF11_4(t)
z=2*t+1/t.^2;
```

3. Создайте файл Q11_4.m (листинг X.13), содержащий описание функции, возвращающей значения функции *Q(t,s)*.

Листинг X.13. Файл Q11_4.m

```
function z=Q11_4(t)
z=t.^2+s.^2+1;
```

4. Создайте файл Volterra11_2.m (листинг X.14), содержащий описание функции, возвращающей решения интегрального уравнения Вольтерра на совпадающей сетке.

Листинг X.14. Файл Volterra11_2.m

```
function [T,Z]=Volterra11(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 (X.33), (X.34)
F=F11_4(t);
F(1)=dF11_4(t(1));
x(1)=F(1)/(h*q(1,1));
for m=2:N
   s=0;
```



```
   for j=2:m-1
      s=s+q(m,j)*m(j);
    end;
    x(m) = (F(m) -h*s)/( h*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 2]=Volterra11_2 (t1,t2,N) ; % численного
% решения интегрального уравнения Вольтерра
>> plo1; (T,Z); % визуализация решения уравнения
% Вольтерра (рис. X.7)
```

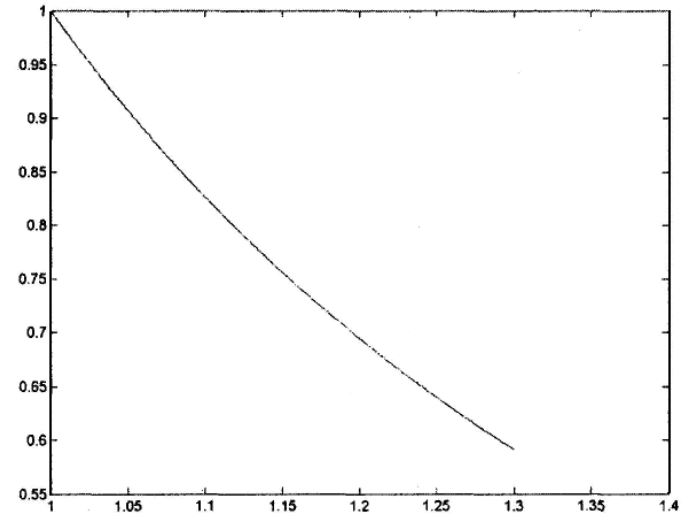


Рис. X.7. Численное решение интегрального уравнения $\int_1^t (t^2+s^2+1)x(s)ds = t^2 - \frac{1}{t}$, полученное при использовании замкнутых квадратурных формул

ЗАКЛЮЧЕНИЕ

Система MatLab представляет собой уникальный сплав универсальных программных и алгоритмических средств с широкой гаммой специализированных приложений. Входной язык и среда программирования MatLab очень близки к современным системам визуального программирования на базе универсальных алгоритмических языков типа C++, Java, Objekt Pascal. По ряду аспектов MatLab уступает указанным системам – режим интерпретации, небольшой запас визуальных компонентов. Однако с его библиотекой численных методов ни по объему, ни по качеству не может сравниться ни одна из систем программирования. На базе MatLab созданы многочисленные расширения, обеспечивающие моделирование и анализ систем в разнообразных сферах человеческой деятельности.

Многие учебные заведения у нас и за рубежом используют MatLab при подготовке специалистов широкого профиля. Для современного инженера и научно-технического работника MatLab является незаменимым инструментом моделирования и исследования различных прикладных систем, прежде всего за счет использования готовых решений. Но не менее важно научиться создавать новые приложения, используя программные и алгоритмические средства MatLab, а также возможность объединения модулей, разработанных в разных системах программирования. Использование системы MatLab в учебном процессе позволяет сблизить дисциплины основанные на математическом моделировании и численных методах.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Амосов А.А. Вычислительные методы для инженеров / А.А. Амосов, Ю.А. Дубинский, Н.А. Копчёнова. – М.: Высш. шк., 1994.

2. Бахвалов Н.С. Численные методы / Н.С. Бахвалов, Н.П. Жидков, Г.М. Кобельков. – М.: Наука, 1987.

3. Боглаев Ю.П. Вычислительная математика и программирование / Ю.П. Боглаев. – М.: Высш. шк., 1990.

4. Васильев Ф.П. Численные методы решения экстремальных задач / Ф.П. Васильев. – М.: Наука, 1998.

5. Волков Е.А. Численные методы / Е.А. Волков. – М.: Наука, 1982.

6. Демидович Б.П. Основы вычислительной математики / Б.П. Демидович, И.А. Марон. – М.: Наука, 1966.

7. Калиткин Н.Н. Численные методы / Н.Н. Калиткин. - М.: Наука, 1974.

8. Кетков Ю.Л. MATLAB 6.x: программирование численных методов / Ю.Л. Кетков, А.Ю. Кетков, М. Шульц. СПб.: БХВ-Петербург, 2004.

9. Марчук Г.И. Методы вычислительной математики / Г.И. Марчук. - М.: Наука, 1977.

10. Пирумов У.Г. Численные методы / У.Г. Пирумов. – М.: Дрофа, 2003.

11. Поршнев С.В. Вычислительная математика / С.В. Поршнев. - Курс лекций. – СПб.: БХВ-Петербург, 2004.

12. Самарский А.А. Введение в численные методы / А.А. Самарский. – М.: Наука, 1982.

13. Самарский А.А. Численные методы / А.А. Самарский, А.В. Гулин. – М.: Наука, 1989.

14. Тихонов А.Н. Вводные лекции по прикладной математике / А.Н. Тихонов, Д.П. Костромаров. М.: Наука, 1984.

15. Турчак Л.И. Основы численных методов / Л.И. Турчак, П.В. Плотников. – М.: ФИЗМАТЛИТ, 2005.

ОГЛАВЛЕНИЕ