Министерство образования и науки Российской Федерации
Амурский государственный университет

А.Г. Масловская, А.В. Павельчук

ЧИСЛЕННЫЕ МЕТОДЫ: ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ И РЕАЛИЗАЦИИ АЛГОРИТМОВ НА БАЗЕ ППП MATLAB

Учебное пособие

Благовещенск
2016

ББК 22.193 я73
М 31

Рекомендовано
учебно-методическим советом университета

Рецензенты:
А.Е. Ковтанюк, профессор каф. информатики и компьютерного моделирования школы естественных наук ДВФУ, д-р физ.-мат. наук;
С.В. Барышкинов, профессор каф. информатики и методики преподавания информатики БГПУ, д-р физ.-мат. наук

Масловская А.Г., Павельчук А.В.
М31 Численные методы: использование инструментальных средств и реализация алгоритмов на базе ППП Matlab. Учебное пособие / А.Г. Масловская, А.В. Павельчук. – Благовещенск: Амурский гос. ун-т, 2016.

В учебном пособии рассматриваются основные теоретические сведения о методах численного анализа, практические аспекты программной реализации вычислительных схем в среде Matlab, а также примеры использования инструментальных возможностей пакета прикладных программ для решения практических задач. Приводятся варианты индивидуальных заданий для лабораторных работ и вопросы для самоконтроля.

Учебное пособие предназначено для студентов, обучающихся по направлениям подготовки 09.03.01 «Информатика и вычислительная техника», 09.03.02 «Информационные системы и технологии», 24.03.01 «Ракетные комплексы и космонавтика» и по специальности 24.05.01 «Проектирование, производство и эксплуатация ракет и ракетно-космических комплексов», а также для студентов других направлений подготовки и специальностей, использующих методы вычислительной математики и инструментальные средства ППП Matlab для решения задач исследовательского и прикладного характера.

ББК 87 я73

В авторской редакции.

ВВЕДЕНИЕ

Современная инженерная практика все чаще представлена областями, в которых аналитические методы могут быть применены только для достаточного узкого класса задач. Поэтому численные методы приобрели исключительно важное значение, в том числе в связи с широким применением программных сред и развитием математических пакетов прикладных программ. В системе прикладного математического образования в вузе методы вычислений занимают важное место. Этот курс тесно связан с основными разделами математических дисциплин (линейная алгебра, математический анализ, дифференциальные уравнения, уравнения математической физики), а также дисциплин цикла «Программирование».

Материал, содержащийся в данном пособии, необходим для организации практических занятий, лабораторного практикума и самостоятельной работы студентов по следующим тематическим разделам: элементы теории погрешностей, численные методы решения нелинейных скалярных уравнений и систем, численные методы решения систем линейных алгебраических уравнений, основы теории интерполирования, методы численного дифференцирования и численного интегрирования, метод наименьших квадратов, численные методы решения обыкновенных дифференциальных уравнений в постановке задач Коши и краевых задач, уравнений с частными производными, интегральных уравнений. Особенностью пособия является его прикладная направленность. Здесь приводятся необходимые теоретические сведения, подробные алгоритмы типовых методов, примеры их программных реализаций, встроенные функции пакета Matlab, контрольные вопросы для самопроверки, а также индивидуальные задания к лабораторным работам. Студенты должны научиться применять на практике методы численного анализа, приобрести навыки программной реализации вычислительных алгоритмов, иметь четкое представление о способах решения задач с использованием встроенного инструментария ППП Matlab, анализировать границы применимости того или иного численного метода, интерпретировать полученные результаты, а также оценивать погрешность вычислений.

1 ПРАВИЛА ПРИБЛИЖЕННЫХ ВЫЧИСЛЕНИЙ И ОЦЕНКА ПОГРЕШНОСТЕЙ ПРИ ВЫЧИСЛЕНИЯХ

При реализации математических моделей явлений или процессов, а также при решении прикладных задач с использованием методов численного анализа неизбежным является возникновение погрешностей. Существует несколько подходов к введению классификаций погрешностей в зависимости от того, какой классификационный признак положен в основу. Для качественного представления возникающих при численном решении задачи погрешностей можно рассмотреть различные виды погрешности в зависимости от того, на каком этапе реализации схемы вычислительного эксперимента они могут возникнуть.

Погрешность задачи обусловлена приближенным характером исходных концептуальной и содержательной постановки задачи моделирования, обусловленным также тем, что свойством любой математической модели является ее неполнота – модель не тождественна объекту-оригиналу, а лишь отражает приоритетные для конкретного исследования основные принципы поведения данного объекта. Также данный вид погрешности может возникать в связи с тем, что на практике параметрами модели часто служат приближенные числа, полученные путем обработки экспериментальных данных, на основе априорного эмпирического анализа и пр. Как правило, погрешность задачи принято считать неустранимой или безусловной, тем не менее, можно полагать, что исследователь может на этапе формулировки научной гипотезы и введения математических соотношений для формализации явления или процесса может на нее существенно повлиять.

Погрешность метода связана со способом – численным методом, применяемым для решения сформулированной задачи. Теоретический анализ вычислительных методик, используемых для решений различного класса задач, позволяет оценить такие погрешности, управлять ими на основе варьирования параметров вычислительной схемы, а также указать границы применимости конкретного численного метода. Следует также понимать, что для решения одной и

той же задачи можно использовать методы, относящиеся к совершенно разным группам и, соответственно, характеризующиеся различными уровнями погрешностей. Поэтому погрешность метода относят к так называемой устранимой или условной погрешности.

Погрешность округлений или погрешность вычислений вызвана необходимостью выполнять арифметические операции над приближенными числами при реализации вычислительных схем и алгоритмов. При решении задач с использованием ЭВМ, как правило, исследователь имеет дело с приближенными числами, усеченными до определенного количества разрядов, зависящего от применяемой вычислительной техники.

Под ***полной погрешностью*** результата решения задачи или численной реализации математической модели понимают совокупность погрешностей всех трех видов.

1.1 Приближенные числа, их абсолютные и относительные погрешности

Для количественной оценки погрешностей, возникающих в процессе численного решения задачи, воспользуемся базовым подходом, а именно, введением погрешностей двух видов: абсолютной и относительной соответственно. Будем считать, что даны два числа: A – точное и a – приближенное значение числа A. Величина $\Delta a = |A - a|$ называется ***абсолютной погрешностью*** приближенного числа a, $\delta a = \dfrac{\Delta a}{|A|}$ – его ***относительной погрешностью*** (часто относительной погрешностью считают и величину: $\delta a = \dfrac{\Delta_a}{|A|}$).

Оценками или ***границами*** абсолютной и относительной погрешностей («***предельными погрешностями***») называют числа Δ_a и δ_a такие, что $\Delta_a \geq \Delta a$, $\delta_a \geq \delta a$. Поскольку часто истинные погрешности не могут быть оценены на практике, под абсолютной и относительной погрешностями понимают их верхние оценки.

При установлении погрешностей и работе с приближенными числами требуется руководствоваться определенным набором достаточно простых правил. Абсолютные погрешности записывают не более чем с двумя-тремя значащими цифрами, при подсчете значащих цифр не учитывают нулей, стоящих слева. В приближенном числе не следует сохранять те разряды, которые подвергаются округлению в его абсолютной погрешности. Относительная погрешность обычно выражается в процентах и ее принято записывать не более чем с двумя-тремя значащими цифрами.

Пример 1. В некоторую вычислительную машину можно ввести числа только с тремя значащими цифрами. С какой точностью можно ввести в нее число *e*? Найти абсолютную и относительную погрешности полученных приближенных чисел.

Решение. Число *e*=2.7182818…, полагаем, что с тремя верными цифрами число e можно записать в виде *e*≈2.71, тогда абсолютную погрешность можно оценить $\Delta_e = 0.0082$, а относительную $\delta_e = \frac{\Delta_e}{|e|} = 0.0030 = 0.3\%$.

Количество верных знаков числа отсчитывается от первой значащей цифры числа до первой значащей цифры его абсолютной погрешности. Количество верных знаков числа отсчитывается от первой значащей цифры числа до первой значащей цифры его абсолютной погрешности: например, число *a*=45.846923, с абсолютной погрешностью Δ_a=0.021 имеет три верных знака 4, 5, 8 и остальные знаки – сомнительные.

При записи результата, как правило, оставляют, кроме верных, один сомнительный знак, при записи промежуточных вычислений кроме верных оставляют два-три сомнительных знака.

Относительная погрешность приближенного числа связана с количеством его верных знаков. Ориентировочно можно считать, что наличие одного верного знака соответствует относительной погрешности порядка 10 %, двух верных знаков – погрешности порядка 1 % и т.д.

Если известно, что все знаки числа – верные, то нетрудно оценить его абсолютную погрешность, руководствуясь следующим правилом – погрешность равна половине единицы последнего разряда числа (если число округлено до m-го десятичного разряда, то их погрешности оцениваются величиной $0.5\cdot10^{-m}$).

1.2 Сложение и вычитание приближенных чисел

Абсолютная погрешность алгебраической суммы нескольких приближенных чисел равна сумме абсолютных погрешностей слагаемых: если $S=\sum_{i=1}^{n} a_i$, то

$$\Delta_S=\sum_{i=1}^{n}\Delta_{a_i}\ . \qquad (1.1)$$

При большом количестве слагаемых оценка абсолютной погрешности суммы по формуле (1.1) оказывается завышенной, т.к. обычно происходит частичная компенсация погрешностей разных знаков. Если известно, что все знаки приближенных чисел, составляющих сумму, верные и округлены до m-го разряда, то погрешность суммы можно оценить с использованием правила Чеботарева: $\Delta_S=\sqrt{3n}\cdot0.5\cdot10^{-m}$, где n – число слагаемых (правило используют при n>10).

Если среди слагаемых имеется одно число, абсолютная погрешность которого значительно превосходит абсолютные погрешности остальных слагаемых, то абсолютная погрешность суммы считается равной этой наибольшей погрешности. При этом в сумме следует сохранять столько десятичных знаков, сколько их в слагаемом с наибольшей погрешностью.

Пример 2. Найти сумму приближенных чисел 0.2542, 5.13513, 246.7, считая в них все знаки верными (т.е. абсолютная погрешность каждого слагаемого не превосходит половины единицы младшего оставленного разряда).

Решение. Наибольшую погрешность Δ=0.05 имеет число 246.7, поэтому можно считать, что погрешность суммы составляет 0.05. Т.к. количество слагаемых невелико, то в расчетах сохраняем только один запасной знак, т.е. ок-

ругляем слагаемые до 0.001: $0.254+5.135+246.7=252.089$, в окончательном результате запасной знак отбрасываем: 252.09. Полный учет погрешностей существенно не изменил бы результат: $\Delta_S = 0.00005+0.000005+0.05=0.050055$.

Для оценки относительной погрешности суммы необходимо воспользоваться результатом вычисления абсолютной погрешности:

$$\delta_S = \frac{\Delta_S}{|S|}. \qquad (1.2)$$

Относительная погрешность δ_S суммы нескольких чисел одного и того же знака заключена между наименьшей и наибольшей из относительных погрешностей слагаемых: $\min\delta_{a_k} \le \delta_S \le \max\delta_{a_k}$ $(a_k > 0, k = 1,2,...n)$. Относительная погрешность разности двух положительных чисел больше относительных погрешностей этих чисел.

Пример 3. Оценить относительную погрешность суммы из примера 2 и сравнить ее с относительными погрешностями слагаемых.

Решение. Относительная погрешность суммы равна $\delta_S = \frac{0.05}{252.089} = 0.000198 = 0.0198\%$. Относительные погрешности слагаемых составляют: $\frac{0.00005}{0.2542} = 0.0197\%$, $\frac{0.000005}{5.13513} = 0.000097\%$, $\frac{0.05}{246.7} = 0.02\%$.

Можно отметить, что согласно указанному выше замечанию, погрешность суммы заключена между наименьшей и наибольшей значениями относительных погрешностей слагаемых, наибольший вклад в погрешность результата вносит третье слагаемое.

1.3 Умножение и деление приближенных чисел

При умножении и делении приближенных чисел складываются их относительные погрешности: если $r = \frac{\prod_{i=1}^{n} a_i}{\prod_{j=1}^{m} b_j}$, то относительная погрешность этого выражения оценивается величиной:

$$\delta_r = \sum_{i=1}^{n} \delta_{a_i} + \sum_{j=1}^{m} \delta_{b_j} . \qquad (1.3)$$

При большом значении $n+m$ следует воспользоваться статистической оценкой: если все числа имеют примерно одинаковую относительную погрешность δ, то относительная погрешность выражения r принимается равной: $\delta_r = \sqrt{3(n+m)} \cdot \delta$ при $n+m>10$.

Если у одного из чисел относительная погрешность значительно превосходит относительные погрешности остальных чисел, то относительная погрешность выражения (1.3) считается равной этой наибольшей погрешности. При этом в результате следует сохранять столько десятичных знаков, сколько их в числе с наибольшей погрешностью.

Абсолютная погрешность выражения вида r вычисляется по его относительной погрешности:

$$\Delta_r = |r| \cdot \delta_r . \qquad (1.4)$$

Пример 4. Вычислить выражение $r = \dfrac{2.1 \cdot 123.4 \cdot 0.05849}{5.1278 \cdot 18.0639}$, считая, что все числа даны с верными знаками.

Решение. Наибольшую относительную погрешность имеет число 2.1, которое содержит всего два верных знака: $\delta_a = \dfrac{0.05}{2.1} = 0.0238 = 2.4\%$. Поэтому можно считать, что относительная погрешность результата составляет 2.4 %, т.е. результат содержит не более двух верных знаков. В расчетах сохраняем один запасной знак, округляя вес числа до трех знаков:

$$r = \frac{2.1 \cdot 123.4 \cdot 0.0585}{5.13 \cdot 18.1} = 0.163 .$$

Абсолютную погрешность вычисляем по его относительной погрешности и найденному численному значению:

$$\Delta_r = r\delta_r = 0.163 \cdot 0.024 = 0.0039 .$$

Округляя результат до верных знаков, отбрасываем запасной знак, запишем ответ: $r = 0.16$.

1.4 Погрешности вычисления значения функции

Функции одной переменной. Абсолютная погрешность дифференцируемой функции $y = f(x)$, вызываемая достаточно малой погрешностью аргумента Δ_x, оценивается величиной:

$$\Delta_y = |f'(x)|\Delta_x. \tag{1.5}$$

Если значения функции положительны, то для относительной погрешности имеется оценка:

$$\delta_y = \frac{|f'(x)|}{|f(x)|}\Delta_x = \left|[\ln f(x)]'\right|\Delta_x. \tag{1.6}$$

В частности, для основных элементарных функций получаются следующие правила.

1. Степенная функция $y = x^a$. Абсолютная погрешность: $\Delta_y = ax^{a-1}\Delta_x$. Относительная погрешность: $\delta_y = |a|\delta_x$.

2. Показательная функция $y = a^x$. Абсолютная погрешность: $\Delta_y = a^x \ln a \cdot \Delta_x$. Относительная погрешность: $\delta_y = \ln a \cdot \Delta_x$.

3. Логарифмическая функция $y = \ln x$. Абсолютная погрешность: $\Delta_y = \frac{1}{x}\cdot\Delta_x = \delta_x$. Относительная погрешность: $\delta_y = \frac{\delta_x}{|\ln x|}$.

Пример 5. Диаметр круга *d*=0.51 м измерен с точностью 1 см. Вычислить площадь круга.

Решение. Вычислим площадь круга, считая ее функцией диаметра: $S(d) = \frac{\pi}{4}d^2$. Для этого оценим абсолютную погрешность функции *S* с помощью (1.5): $\Delta_S = 2\frac{\pi}{4}d\cdot\Delta_d$, подставляя численные значения.

Чтобы не учитывать погрешность числа π, возьмем его, например, с пятью значащими цифрами.

Абсолютная погрешность результата составляет:

$$\Delta_S = \frac{3.1416}{2} 0.51 \cdot 0.01 = 0.0080 .$$

Тогда, сохраняя в ответе один сомнительный знак, вычислим площадь круга: $S(d) = \frac{3.142}{4} 0.51^2 \approx 0.204$ м^2.

Данную задачу можно решить, используя также правило расчета относительной погрешности произведения: $\delta_S = 2\delta_d$. Убедитесь самостоятельно в согласовании результатов, полученных с помощью разных подходов к вычислению абсолютной погрешности.

Функции нескольких переменных. Абсолютная погрешность дифференцируемой функции $y = f(x_1, x_2, ...x_n)$, вызываемая достаточно малыми погрешностями $\Delta_{x_1}, \Delta_{x_2}, ...\Delta_{x_n}$ аргументов, оценивается величиной:

$$\Delta_y = \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i} \right| \cdot \Delta_{x_i} . \qquad (1.7)$$

Если значения функции положительны, то для относительной погрешности имеет место оценка:

$$\delta_y = \sum_{i=1}^{n} \frac{1}{f} \left| \frac{\partial f}{\partial x_i} \right| \cdot \Delta_{x_i} . \qquad (1.8)$$

Пример 6. Вычислить значение функции $f = x_1 \cdot x_2^2 \cdot x_3^3$, если приближенные числа записаны со всеми верными знаками:

$x_1 = 12.7, \quad x_2 = 1.14, \quad x_3 = 5.478$.

Решение.

Оценим абсолютные и относительные погрешности аргументов функции нескольких переменных:

$\Delta_{x_1} = 0.05, \quad \Delta_{x_2} = 0.005, \quad \Delta_{x_3} = 0.0005$;

$$\delta_{x_1} = \frac{0.05}{12.7} = 0.39 \ \%, \quad \delta_y = \frac{0.005}{1.14} = 0.44 \ \%, \quad \delta_z = \frac{0.0005}{5.478} = 0.009 \ \%,$$

Тогда относительная погрешность функции может быть вычислена как:

$$\delta_f = \frac{1}{\left|x_1 \cdot x_2^2 \cdot x_3^3\right|}\left[x_2^2 \cdot x_3^3 \Delta_{x_1} + 2x_1 \cdot x_2 \cdot x_3^3 \Delta_{x_2} + 3x_1 \cdot x_2^2 \cdot x_3^2 \Delta_{x_3}\right] =$$

$$= \frac{\Delta_{x_1}}{|x_1|} + 2\frac{\Delta_{x_2}}{|x_2|} + 3\frac{\Delta_{x_3}}{|x_3|} = \delta_{x_1} + 2\delta_{x_2} + 3\delta_{x_3} = 1.3 \ \ \%.$$

Поэтому значение функции следует вычислять не более, чем с двумя верными знаками: $f = 2713.18 \approx 2.7 \cdot 10^3$.

1.5 Определение допустимой погрешности аргументов по допустимой погрешности функции

Данная задача имеет однозначное решение только для функции одной переменной $y = f(x)$, если эта функция дифференцируема и $f'(x) \neq 0$, то

$$\Delta_x = \frac{1}{|f'(x)|}\Delta_y . \qquad (1.9)$$

Для функции нескольких переменных $y = f(x_1, x_2, ...x_n)$ задача решается только при введении ограничений. Если значения всех аргументов можно одинаково легко определить с любой точностью, то обычно применяют *принцип одинаковых влияний*, считая, что в формуле (1.8) все слагаемые $\left|\frac{\partial f}{\partial x_i}\right| \cdot \Delta_{x_i}$ равны между собой, что дает формулу:

$$\Delta_{x_i} = \frac{\Delta_y}{n\left|\frac{\partial f}{\partial x_i}\right|}, \qquad i = 1,2...n . \qquad (1.10)$$

Пример 7. С какой точностью следует определить радиус основания *R* и высоту *H* цилиндрической банки, чтобы ее вместимость можно было определить с точностью до 1 %?

Решение. $V = \pi R^2 H$, поэтому относительная погрешность: $\delta_V = 2\delta_R + \delta_H$. Если можно обеспечить любую точность определения *R* и *H*, то можно воспользоваться принципом равных влияний, откуда на долю *R* и *H* приходится по 0.5 %. Таким образом, по принципу равных влияний надо определить радиус с относительной погрешностью 0.25 %, а высоту – 0.5 %. На практике

чаще встречаются такие случаи, когда радиус определяется с меньшей точностью, чем высота. Тогда, если радиус определяется с точностью вдвое меньшей, чем высота, полагаем $\delta_R = 2\delta_H$ и из условия $2\delta_R + \delta_H = 5\delta_H = 1\%$ находим $\delta_H = 0.2$ %, $\delta_R = 0.4$ %. Число π нужно выбирать с относительной погрешностью 0.01 % ($\pi = 3.145$), чтобы эту погрешность не приходилось учитывать в окончательном результате.

Контрольные вопросы

1. Какая погрешность считается неустранимой и почему?

2. Как определить верные знаки числа по его абсолютной и относительной погрешностям?

3. В чем заключается особенность определения погрешности результата сложения или вычитания приближенных чисел?

4. Какова особенность определения погрешности результата операций деления или умножения приближенных чисел?

5. Как определяется относительная погрешность вычисления значения функции?

Индивидуальные задания

1. Округляя следующие числа до трех значащих цифр, определить абсолютную и относительную погрешности полученных приближенных чисел.

Номер варианта	Число *A*	Число *B*	Номер варианта	Число *A*	Число *B*
1	2.1514	0.01205	11	5.6987	0.005214
2	3.13105	0.001314	12	23.563	1.0256987
3	– 7.236	0.05695	13	789.123	– 4.12369
4	5.123	0.012589	14	13.2589	0.02569
5	1.48963	0.014569	15	119.321	0.025896
6	2.1563	– 12.697852	16	12.356584	– 0.014552
7	0.000589	158.963	17	1.258932	2.3156
8	18.5623	– 1.256	18	2.12589	5.69784
9	5.256987	0.1352	19	– 0.1258	1963.4
10	0.25874	5236.89	20	0.258963	14789.1

2. Определите количество верных знаков в числе x, если известна его абсолютная погрешность.

Номер варианта	Число x	Абсолютная погрешность Δ_x	Номер варианта	Число x	Абсолютная погрешность Δ_x
1	0.394113	$0.25\cdot10^{-2}$	11	856.354	$0.1\cdot10^{-2}$
2	– 0.2113	$0.5\cdot10^{-2}$	12	1.09315	$0.15\cdot10^{-2}$
3	293.481	0.1	13	-32.285	$0.2\cdot10^{-2}$
4	38.2543	$0.27\cdot10^{-2}$	14	0.314563	$0.05\cdot10^{-2}$
5	0.1132	$0.1\cdot10^{-3}$	15	5.03312	$0.1\cdot10^{-2}$
6	0.12514	$0.15\cdot10^{-2}$	16	0.25836	$0.15\cdot10^{-2}$
7	0.25896	$0.1\cdot10^{-3}$	17	1.2587	$0.1\cdot10^{-2}$
8	1.258	$0.5\cdot10^{-2}$	18	1.5893	$0.75\cdot10^{-2}$
9	832.112	$0.1\cdot10^{-2}$	19	5.8932	$0.1\cdot10^{-3}$
10	125.366	$0.15\cdot10^{-2}$	20	0.01914	$0.3\cdot10^{-2}$

3. Найти сумму приближенных чисел $a+b+c$ и указать абсолютную и относительную погрешности результата.

Номер варианта	Число a	Число b	Число c	Номер варианта	Число a	Число b	Число c
1	0.145	321.22	78.2	11	23.56	6987.2	0.005214
2	145.26	0.22135	787.3	12	156.3	0.12563	1.25
3	456.35	12.0258	45.3	13	223.6	6.022548	0.524
4	12.3	0.1236	1.358	14	156.3	– 0.12	1.00525
5	13.6	478.2	0.001256	15	3.44	6.548	112.116
6	1.2	– 152.69	0.014872	16	1.25	8.593	4.563215
7	0.2589	2.5	14.589	17	5.52641	1.256	3.61
8	21.36	89.3	0.0125	18	4.5	12.4569	– 1.25634
9	1.11	5.8963	0.18985	19	0.12589	– 0.3654	0.25
10	25.963	1.456	8.395414	20	1.258	3.6	4.58896

4. Вычислить выражение, считая, что все цифры даны с верными знаками. Определить абсолютную и относительную погрешности результата.

Номер варианта	Выражение r	Номер варианта	Выражение r
1	$\dfrac{3.12\cdot346.9+0.05613}{1.178\cdot34.506}$	11	$\dfrac{0.14\cdot789.66+0.01236}{1.4893}$
2	$\dfrac{5.2+316.85\cdot0.012589}{17.25\cdot58.3}$	12	$\dfrac{12.36+10.236985\cdot0.012}{0.005\cdot0.258}$

3	$\frac{5{,}514 + 5.23 \cdot 0.58129}{145.26 \cdot 0.3698}$	13	$\frac{0.1236 + 789.3 \cdot 1.354}{1.4893 \cdot 23.5}$
4	$\frac{0.123 \cdot 2.36 + 256.34}{12.36 \cdot 5.6}$	14	$\frac{12.6 \cdot 0.0085 + 0.12}{10.5 \cdot 11.258}$
5	$\frac{(2.13 + 0.0123)23.5}{168.96312}$	15	$\frac{0.136 + 79.553 \cdot 1.3224}{1.493 \cdot 3.5}$
6	$\frac{(1.58963 + 12.5)5.86941}{115.589632}$	16	$\frac{16.6369 + 0.85 \cdot 0.1158}{10.2 \cdot 11.698}$
7	$\frac{1.25 + 4.8115 \cdot 16.36}{17.893254}$	17	$\frac{3.1152 + 25.66 \cdot 12.896325}{2.3 \cdot 0.00258}$
8	$\frac{2.1652 + 45.8963 \cdot 0.2}{158.654123}$	18	$\frac{1.0514 + 2.23 \cdot 0.529}{15.26 \cdot 0.38}$
9	$\frac{8.14 \cdot 7.6 + 0.23116}{1.48931}$	19	$\frac{1.14582 + 0.2596 \cdot 1.5}{1.125 \cdot 96.3}$
10	$\frac{5.5 + 1.1563 \cdot 0.552}{145.26 \cdot 0.111}$	20	$\frac{3.21101 + 1.524 \cdot 4.2}{2.26 \cdot 0.333}$

5. Для следующих функций вычислить значения при указанных значениях переменных. Указать абсолютную и относительную погрешности результатов, считая все знаки исходных данных верными.

Номер варианта	u	x	y	z
1	$\ln\left(x^2 + z \cdot y\right)$	0.9700	1.1321	0.1101
2	$\frac{x^2 + y}{z}$	3.280	23.105	0.126
3	$x^4 + 2y^3 + 3z^2$	0.12	5.36	0.1365
4	$x \cdot y \cdot z^2$	0.012	7.3000	1.487
5	$x^3 \cdot y^4 \cdot z^5$	1.2252	1.456	0.1578
6	$e^x + yz$	1.22	0.333	5.48702
7	$\ln\left(x \cdot y \cdot z\right)$	21.3113	0.025	0.2563
8	$x^3 \cdot y^2 \cdot z^3$	1.202	0.1300	7.960
9	$x + y \cdot z^2$	1.133	5.5000	0.33152
10	$x + y^2 \cdot z^2$	10.111	22.0553	8.11
11	$\frac{x^2 + y}{z}$	2.105	1.120	5.523
12	$z \cdot \ln\left(x^2 + y\right)$	1.5896	0.256	93.5100
13	$x^3 + 2y^2 + 5z$	1.583	0.0125	17.520

14	$5x^5 \cdot y \cdot z^2$	1.111	2.56901	1.140
15	$2x^2 \cdot y \cdot z^2$	0.250	0.0015	1.14895
16	$\cos(x \cdot y \cdot z)$	2.156	0.2589	16.000
17	$x^3 \cdot y^2 \cdot z^3$	5.6000	8.9900	14.25416
18	$x \cdot \sin(y \cdot z)$	15.201	18.256	0.12544
19	$x \cdot y^2 + z^2$	5.24130	0.1000	5.400
20	$8e^x + 2y \cdot z$	2.300	1.4593	4.3601

2 ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ НЕЛИНЕЙНЫХ СКАЛЯРНЫХ УРАВНЕНИЙ

Рассмотрим задачу приближенного нахождения корней уравнения вида:

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

где $f: R_1 \mapsto R_1$ – алгебраическая или трансцендентная функция. Аналитические методы решений скалярных уравнений вида (2.1) разработаны только для отдельных классов уравнений. В общем случае можно говорить лишь о приближенном вычислении корней уравнения (2.1), т.е. таких значений аргумента $x=\xi$, при которых равенство $f(\xi)=0$ истинно. При этом под близостью приближенного значения $\bar{x}$ к корню ξ уравнения (2.1) понимают выполнение неравенства:

$$|\xi-\bar{x}|<\varepsilon. \tag{2.2}$$

При малых $\varepsilon>0$ часто важно контролировать не абсолютную погрешность, а относительную: $\dfrac{|\xi-\bar{x}|}{|\bar{x}|}$.

При характеристике методов часто используют понятие скорости их сходимости. Близость приближенного решения задачи x_k, полученного после k-го шага итерационного метода, к точному решению ξ и аналогичную близость для приближенного решения, полученного после (k+1)-го шага связывает формула

$$|x_{k+1}-\xi|\le\beta|x_k-\xi|^{1+\alpha}. \tag{2.3}$$

В зависимости от значений α (α=0, 0<α<1, α=2) говорят о линейной, сверхлинейной и квадратичной сходимости. При большом числе итераций на скорость сходимости метода главное влияние оказывает значение α, но при малом их числе значение β может оказаться определяющим. Наконец, скорость сходимости метода и скорость его работы – не взаимозаменяемые понятия, поскольку при оценке скорости сходимости никак не учитывается количество вычислений на k-м шаге. Различные программные реализации одного и того же метода также будут различаться по скорости работы в зависимости от исполь-

зуемого языка программирования, вида транслятора и квалификации программиста.

2.1 Локализация корней

Нелинейная функция $f(x)$ в области определения $D(f) \subseteq R_1$ может иметь конечное или бесконечное количество нулей или не иметь их вообще. Большинство же методов нахождения корней требует знания промежутков, где имеется и притом единственный нуль функции. Таким образом, ставится подзадача существования и единственности, нахождения границ и локализации (изоляции, отделения) корней. Для функций общего вида нет универсальных способов решения поставленных подзадач. На практике используют следующие подходы.

Графический способ локализации корней. Если по геометрической интерпретации функции $f(x)$ можно однозначно представить ситуацию, касающуюся количества и расположения нулей, то локализацию производят, отделяя те промежутки на оси абсцисс, где график функции $y = f(x)$ пересекает ось Ox.

Если построение графика функции $y = f(x)$ вызывает затруднения, но исходное уравнение (2.1) очевидным образом представляется в виде $f_1(x) = f_2(x)$ и функции $f_1(x)$ и $f_2(x)$ таковы, что построение их графиков возможно, задача определения корней и областей их единственности решается отслеживанием точек пересечения этих графиков и выделением на оси абсцисс тех промежутков, которым принадлежат проекции данных точек.

Пример 1. Графическим методом установить промежутки локализации корней уравнения $x^2 - \cos x - 1 = 0$.

Представив уравнение в виде $x^2 - 1 = \cos x$. Создадим файл-сценарий, с помощью которого строим графики функций $y_1(x) = x^2 - 1$ и $y_2(x) = \cos x$:

```
x=-2:0.001:2;
y1=x.^2-1;
y2=cos(x);
plot(x,y1,'r-',x,y2,'b-');grid on
```

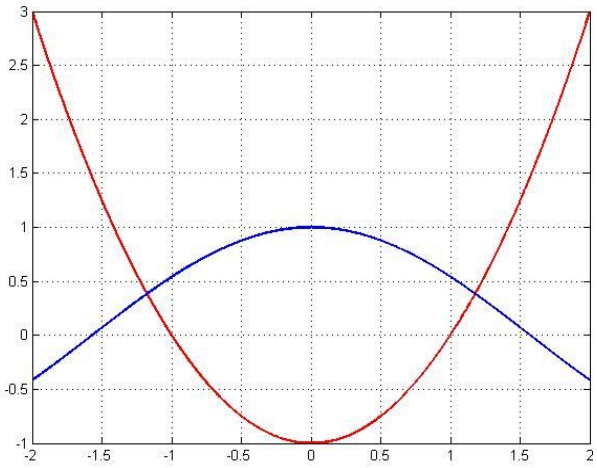


Рис. 2.1. Результат графической локализации корня уравнения.

Традиционный подход заключается в прямом построении графика функции $y = f(x)$:

```
x=-2:0.001:2;
y=x.^2-1-cos(x);
plot(x,y,'r-');grid on
```

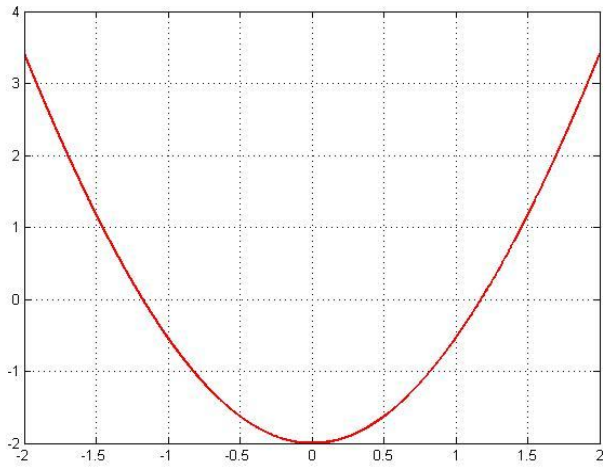


Рис. 2.2. Результат графической локализации корня уравнения.

Рассмотрение графиков позволяет сделать заключение, что данное уравнение имеет два корня: $\xi_1 \in [-1.5,-1]$ и $\xi_2 \in [1,1.5]$.

Аналитический способ отделения корней. Установить, что на данном отрезке $[a, b]$ (определенным, например, графическим способом) непрерывная

функция $f(x)$ имеет нуль можно, используя усиленную теорему Больцано-Кош, наложив дополнительное требование монотонности функции на этом отрезке.

Утверждение 1. Непрерывная строго монотонная функция $f(x)$ имеет и притом единственный нуль на отрезке $[a, b]$ тогда и только тогда, когда на его концах она принимает значения разных знаков.

Для того, чтобы заключить, что дифференцируемая функция $f(x)$ монотонна на данном отрезке $[a, b]$ необходимо убедиться в знакопостоянстве ее производной на всем отрезке.

Утверждение 2. Пусть $f \in C^1_{[a,b]}$. Тогда если $f'(x)$ не меняет знак на (a, b), то условие $f(a)\cdot f(b)<0$ является необходимым и достаточным для того, чтобы уравнение (2.1) имело и притом единственный корень на $[a,b]$.

Пример 2. Используя аналитический способ, убедиться, что установленные графическим способом отрезки локализации корней уравнения $x^2-\cos x-1=0$ удовлетворяют условиям утверждения 2.

Исследуем знаки значений функции на концах отрезков локализации корней:

	–1.5	–1	1	1.5
$f(x)$	+	–	–	+

Пусть $f(x)=x^2-\cos x-1$, тогда $f'(x)=2x+\sin x$. На отрезке [–1.5,–1] производная функции $y=f(x)$ всегда имеет отрицательное значение, на отрезке [1,1.5] – положительное. Поэтому на основании утверждения 2 можно заключить, что данное уравнение имеет и при том единственный корень на каждом из отрезков локализации.

В общем случае, если указанные процедуры затруднительны для анализа, всю область определения или какую-нибудь ее часть, вызывающую по тем или иным соображениям интерес, разбивают на отрезки точками x_i, расположенными на условно небольшом расстоянии h одна от другой. Вычислив значения во

всех этих точках, проверяют выполнение условия $f(x_{i-1})\cdot f(x_i)\le 0$. Если заранее известно число корней в исследуемой области, то, измельчая шаг h поиска, таким процессом можно либо их все локализовать, либо утверждать, что возможно наличие пар корней, не различимых с точностью $h = \varepsilon$.

2.2 Методы дихотомии. Метод половинного деления

Пусть функция $f(x)$ определена и непрерывна при всех $x\in[a,b]$ и на $[a,b]$ меняет знак. Согласно утверждению 2 уравнение (2.1) на (a, b) имеет единственный корень.

Выберем произвольную точку $c\in[a,b]$, $[a,b]$ – промежуток существования корня, c – пробная точка. Вычисление значения $f(c)$ приведет к какой-либо одной из следующих взаимоисключающих ситуаций, которые можно интерпретировать следующим образом:

1) $f(a)\cdot f(c)<0$, при этом корень находится на интервале (a, c),

2) $f(c)\cdot f(b)<0$, корень находится на интервале (c, b),

3) $f(c)=0$, точка c – искомый корень.

Таким образом, одно вычисление значения функции позволяет уменьшить промежуток $[a,b]$ существования корня. Для случая 1 требуется положить $b = c$, для случая 2 – $a = c$, в случае 3 – алгоритм заканчивает свою работу. Описанная процедура сужения промежутка существования нуля непрерывной функции далее повторяется циклически. Такой легко программируемый процесс объединяет семейство *методов дихотомии*.

Наиболее известным и достаточно простым частным случаем семейства етодов дихотомии является *метод половинного деления*, реализующий самый простой способ выбора пробной точки – деление промежутка существования корня пополам: $c=\dfrac{a+b}{2}$.

За один шаг метода половинного деления промежуток существования корня сокращается ровно вдвое. Поэтому если за k-е приближение этим методом к корню ξ уравнения (2.1) примем точку x_k, являющуюся серединой полу-

ченного на k-м шаге отрезка $[a_k, b_k]$ в результате последовательного сужения $[a,b]$, то получим:

$$|\xi - x_k| < \frac{b-a}{2^k}, \quad \forall k \in N. \tag{2.4}$$

Следовательно, метод имеет линейную сходимость со значением β=0.5 согласно оценке (2.3). Неравенство (2.4), являясь априорной оценкой абсолютной погрешности приближенного равенства $x_k \approx \xi$, дает возможность подсчитать число итераций метода половинного деления, достаточное для получения корня с заданной точностью. Для этого требуется найти наименьшее натуральное k, удовлетворяющее неравенству $\frac{b-a}{2^n} \le \varepsilon$. Однако в практике алгоритмизации этого и других методов решения нелинейных скалярных уравнений в качестве критерия останова вычислительного процесса часто используют упрощенные требования. Первый подход состоит в совместном использовании двух проверок: «по x» и «по y»:

$$|x_{k+1} - x_k| \le \varepsilon, \quad |f(x_{k+1})| \le \delta, \tag{2.5}$$

где ε, δ – априорно заданные точности вычислений. Таким образом, процесс работы вычислительной схемы завершается, как только достигнута заданная точность согласно (2.5). Второй подход заключается в использовании только второго соотношения в (2.5). Рисунок 2.3 дает геометрическую интерпретацию методу половинного деления.

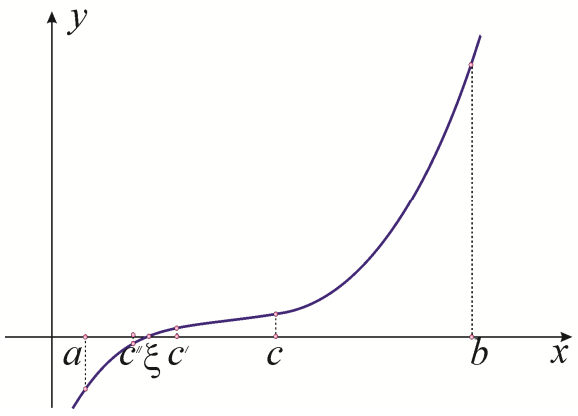


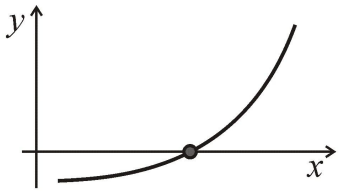
Рис. 2.3. Геометрическая интерпретация процедуры приближения к корню уравнения методом половинного деления.

2.3 Метод хорд

В семействе методов дихотомии можно достичь несколько лучших результатов, если отрезок $[a,b]$ делить не пополам, а пропорционально величинам ординат $f(a)$ и $f(b)$ графика данной функции $f(x)$. Это означает, что точку c есть смысл находить как абсциссу точки пересечения оси Ox с прямой, проходящей через точки $A(a, f(a))$ и $B(b, f(b))$, – с хордой AB дуги $A\xi B$. Из уравнения прямой, проходящей через точки A и B, находим:

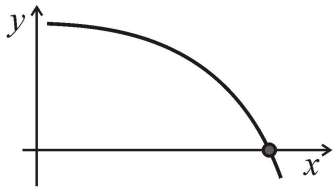
$$c = a - \frac{f(a)(b-a)}{f(b)-f(a)}. \qquad (2.6)$$

Метод, получающийся из метода дихотомии таким выбором пробной точки, называют *методом хорд* (методом секущих или методом линейной интерполяции). Для сходимости метода хорд потребуем выполнений условий утверждения 2, а также потребуем, чтобы на отрезке $[a, b]$ вторая производная $f''(x)$ сохраняла знак. Возможны четыре случая (рисунок 2.4).



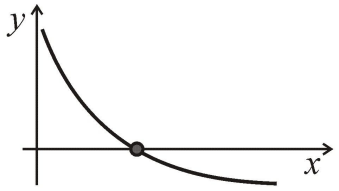
$f'(x) > 0, f''(x) > 0$ – функция возрастает, ее график вогнутый

а



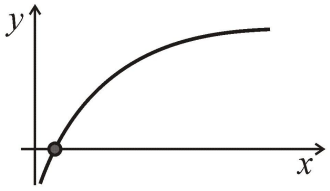
$f'(x) < 0, f''(x) < 0$ – функция убывающая, ее график выпуклый

б



$f'(x) < 0, f''(x) > 0$ – функция убывающая, ее график вогнутый

в



$f'(x) > 0, f''(x) < 0$ – функция возрастающая, ее график выпуклый

г

Рис. 2.4. Геометрическая иллюстрация локализации корня и возможностей поведения функции $f(x)$ в окрестности корня.

Тогда для одного из концов отрезка знаки значений функции и ее второй производной совпадают $f(x)\cdot f''(x)>0$. Назовем его неподвижной точкой. Если неподвижной точкой является левый конец отрезка *a*, то в качестве нулевого приближения выбирается правый конец отрезка $x_0=b$ и очередное приближение получается по формуле

$$x_{n+1}=x_n-\frac{f(x_n)}{f(x_n)-f(a)}(x_n-a),\quad n=0,1,2,\ldots, \tag{2.7}$$

а если неподвижной точкой оказывается *b*, то $x_0=a$ и

$$x_{n+1}=x_n-\frac{f(x_n)}{f(b)-f(x_n)}(b-x_n),\quad n=0,1,2,\ldots. \tag{2.8}$$

Для оценки точности приближения можно воспользоваться формулой

$$|x_n-\xi|\le\frac{|f(x_n)|}{m}\ \text{ или }\ |x_n-\xi|\le\frac{M-m}{m}|x_n-x_{n-1}|,$$

где $0<m\le|f'(x)|\le M,\ \ \forall x\in[a,b]]$. Если же на этом отрезке $M\le|f'(x)|\le 2m,\ \ \forall x\in[a,b]$, то $|x_n-\xi|\le|x_n-x_{n-1}|$. Метод обладает суперлинейной сходимостью со значениями параметров $\alpha\approx0.62,\ \beta\approx|M/2m|^{0.62}$ согласно (2.3). Рисунок 2.5 иллюстрирует геометрическую интерпретацию процедуры нахождения корня методом хорд.

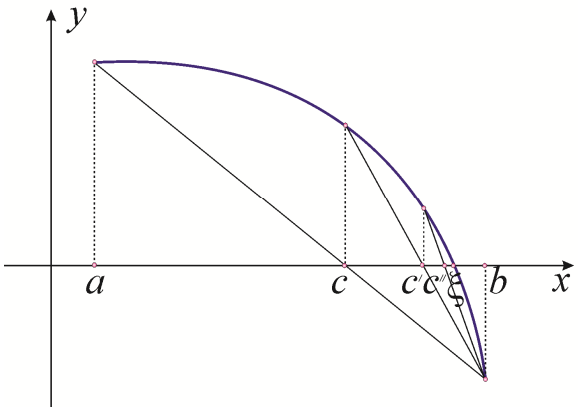


Рис. 2.5. Приближение к корню уравнения методом хорд (точка x_0 – неподвижная точка).

2.4 Метод Ньютона

Метод Ньютона или метод касательных является одним из широко распространенных итерационных методов решения нелинейных уравнений, что обусловлено с его идейной простотой, быстрой сходимостью и относительно несложной программной реализацией.

Метод касательных можно использовать в тех случаях, когда корень ξ уравнения (2.1) отделен на отрезке $[a,b]$, причем первая $f'(x)$ и вторая производные $f''(x)$ определены, непрерывны и сохраняют постоянные знаки на этом отрезке (четыре возможных варианта поведения функции в окрестности корня ξ показаны на рисунке 2.2).

Пусть функция $f(x)$ дважды дифференцируема на отрезке $[a,b]$, содержащем корень ξ уравнения (2.1), пусть $x_n \in [a,b]$ – уже известный член последовательности приближений к ξ. Для любого $x \in [a,b]$ можно записать формальное представление разложения функции $f(x)$ по формуле Тейлора:

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \frac{1}{2} f''(\theta_n)(x - x_n)^2 + \ldots \qquad (2.9)$$

Так как ξ – корень уравнения (2.1), то разложение (2.9) справедливо и для любого x, например, для $x = \xi$. Считая, что значение x_n достаточно близко к ξ (разность $(x - \xi)^2$ – достаточно мала), ограничимся двумя членами разложения в правой части выражения (2.9), при этом будет найдено новое приближение x_{n+1}. Предполагая, что, по крайней мере, на элементах последовательности x_n первая производная данной функции в ноль не обращается, итерационный процесс Ньютона определится в явном виде формулой:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0,1,2,\ldots \qquad (2.10)$$

Так же, как и в методе хорд, в качестве нулевого приближения корня выбирают один из концов отрезка $[a,b]$, а в качестве неподвижной точки – другой. Если $f(a) \cdot f''(a) > 0$, то $x_0 = a$ и b – неподвижная точка. В противном случае: $x_0 = b$ и a – неподвижная точка.

В полученном выражении (2.10) легко усмотреть уравнение касательной к кривой $f(x)$, проведенной в точке $(x_n, f(x_n))$. Отсюда геометрический смысл метода Ньютона: приближения к корню ξ совершаются по абсциссам точек пересечения касательных к графику данной функции, проводимых в точках, соответствующих предыдущим приближениям.

Рисунок 2.6 отражает геометрический смысл нахождение корня методом касательных.

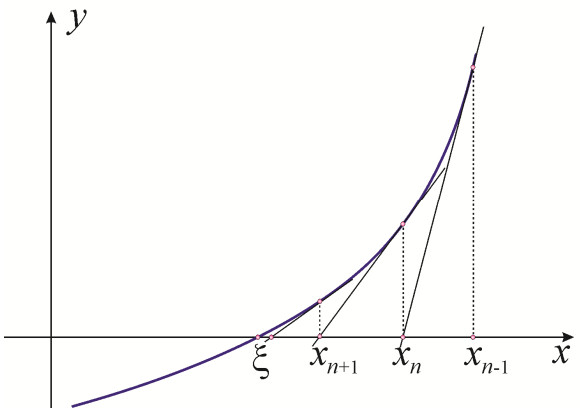


Рис. 2.6. Приближение к корню уравнения методом Ньютона.

Для оценки скорости сходимости метода Ньютона можно использовать следующее утверждение.

Утверждение 3. Пусть функция $f(x)$ удовлетворяет условиям

$$\begin{cases} |f'(x)| \geq m > 0, \\ |f''(x)| \leq M < \infty, \end{cases} \quad \forall x \in [a,b].$$

Тогда, если члены последовательности $\{x_n\}$, определяемые методом Ньютона (2.10), при любом фиксированном $n \in \mathrm{N}_0$ принадлежат отрезку $[a,b]$ и эта последовательность сходится к корню ξ уравнения (1.1), то справедливы неравенства:

$$\begin{aligned} |x_{n+1} - \xi| &\leq \frac{M}{2 \cdot m} |x_n - \xi|^2, \\ |x_{n+1} - \xi| &\leq \frac{M}{2 \cdot m} |x_{n+1} - x_n|^2. \end{aligned} \tag{2.11}$$

Метод обладает квадратичной сходимостью со значением $\beta \approx |M / 2m|^{0.62}$.

2.5 Комбинированный метод хорд и касательных

Пусть $f(a)\cdot f(b)<0$, а $f'(x)$ и $f''(x)$ определены, непрерывны и сохраняют постоянные знаки на отрезке $[a,b]$. Объединяя метод хорд и метод Ньютона, можно получить итерационный метод, на каждом шаге которого будут находиться значения корня ξ уравнения (2.1) по недостатку и по избытку. Отличие этого метода от вышеописанных состоит в том, что в итерационном процессе координаты точек концов отрезка сходимости находятся различными способами. В общем случае координаты концов отрезка на n+1-м шаге рассчитываются по формулам:

$$x_{n+1}=x_n-\frac{f(x_n)(\tilde{x}_n-x_n)}{f(\tilde{x}_n)-f(x_n)},\quad \tilde{x}_{n+1}=\tilde{x}_n-\frac{f(\tilde{x}_n)}{f'(\tilde{x}_n)} \tag{2.12}$$

Значения x_0 и $\tilde{x}_0$ выбираются в зависимости от знаков $f'(x)$ и $f''(x)$ на $[a,b]$:

1) если $f'(x)>0$ и $f''(x)>0$, то $x_0=a, \tilde{x}_0=b$;

2) если $f'(x)>0$ и $f''(x)<0$, то $x_0=b, \tilde{x}_0=a$;

3) если $f'(x)<0$ и $f''(x)>0$, то $x_0=a, \tilde{x}_0=b$;

4) если $f'(x)<0$ и $f''(x)<0$, то $x_0=b, \tilde{x}_0=a$.

Если заданная погрешность приближенного значения равна ε, то процесс сближения прекращается в тот момент, когда будет обнаружено, что $|\tilde{x}_{n+1}-x_{n+1}|<\varepsilon$. За приближенное значение корня лучше всего взять среднее арифметическое полученных последних значений $\xi\approx\frac{1}{2}(x_{n+1}+\tilde{x}_{n+1})$. Сходимость метода не хуже, чем сходимости методов, комбинацией которых он является.

2.6 Модификации метода Ньютона

Введение модификаций для формул, определяющих интеграционные вычислительные схемы, как правило, преследует две цели. Первая заключается в сокращении вычислительных затрат (как следствие, это может приводить к потере точности), вторая, напротив, направлена на повышение точности метода,

что, в свою очередь, может привести к усложнению вычислительного алгоритма.

Метод Ньютона-Шредера. Если заведомо известно число p – показатель кратности корня ξ, то для ускорения сходимости метода Ньютона в формулу (2.10) вводят корректирующий множитель p:

$$x_{n+1} = x_n - p \cdot \frac{f(x_n)}{f'(x_n)}, \quad n = 0,1,2,... \tag{2.13}$$

Такую модификацию называют *методом Ньютона – Шредера*.

Упрощенный метод Ньютона. Использование на каждом шаге одного и того же шагового множителя $\frac{1}{f'(x_0)}$ дает модифицированный или *упрощенный метод Ньютона*:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}, \quad n = 0,1,2,... \tag{2.14}$$

Этот метод имеет очевидную геометрическую интерпретацию: в начальной точке x_0 проводится касательная к графику функции $f(x)$, а во всех последующих точках проводятся прямые, параллельные этой касательной, как показано на рисунке 2.7.

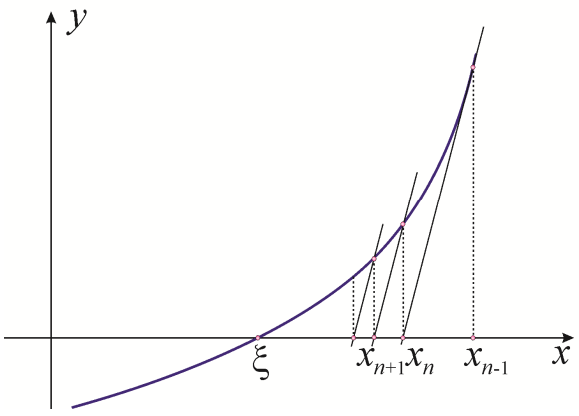


Рис. 2.7. Приближение к корню уравнения упрощенным методом Ньютона.

Разностный метод Ньютона. Заменяя производную, вычисленную в точке x_n, ее разностным аналогом, получим *разностный метод Ньютона*:

$$x_{n+1} = x_n - \frac{f(x_n)h_n}{f(x_n + h_n) - f(x_n)}, \quad n = 0,1,2,\ldots, \tag{2.15}$$

где h_n – конечное приращение аргумента на *n*-ой итерации.

Метод Стеффенсена основан на выборе $h_n = f(x_n)$ (это можно делать на стой стадии итерационного процесса, когда значения $|f(x_n)|$ уже достаточно малы), в следствие чего формула (2.15) преобразуется к виду:

$$x_{n+1} = x_n - \frac{f(x_n)^2}{f(x_n + f(x_n)) - f(x_n)}, \quad n = 0,1,2,\ldots . \tag{2.16}$$

Двушаговый метод Ньютона. Принимая $h_n = x_{n-1} - x_n$ в выражении (2.16), получим итерационный процесс, который носит название *двушагового метода Ньютона*:

$$x_{n+1} = x_n - \frac{f(x_n)(x_{n-1} - x_n)}{f(x_{n-1}) - f(x_n)}, \quad n = 1,2,\ldots . \tag{2.17}$$

2.7 Метод Чебышева третьего порядка

Пусть производные функции $f(x)$ первого, второго и третьего порядков непрерывны на отрезке $[a,b]$, причем $f'(x) \neq 0$ на $[a,b]$. Допустим также, что проведена процедура отделения корней и начальное приближение $x_0 \in [a,b]$, тогда итерационный процесс *метода Чебышева* задается соотношением:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f''(x_n) \cdot f^2(x_n)}{2 \cdot (f'(x_n))^3}, \quad n = 0,1,2,\ldots . \tag{2.18}$$

Данный метод обладает скоростью сходимости третьего порядка, оценка погрешности имеет вид

$$|\xi - x_n| \le \left(\alpha \cdot \frac{\beta^3}{3!} \cdot |\xi - x_0| \right)^{\frac{3^n - 1}{2}}, \tag{2.19}$$

где $\alpha = \max\limits_{x \in [a,b]} |F'''(f(x))|$, $\beta = \max\limits_{x \in [a,b]} |f'(x)|$, $F(f(x))$ – функция, обратная к $f(x)$ на отрезке $[a,b]$, имеет непрерывные производные первого, второго и третьего порядков.

2.8 Метод простой итерации

В основе метода простых итераций или метода последовательных приближений лежат следующие положения. Решаемое уравнение (2.1) заменим равносильным уравнением следующего вида:

$$x = g(x). \qquad (2.20)$$

Предположим также, что проведена процедура локализации корня ξ уравнения (2.1), в результате чего установлен отреок существования корня $[a, b]$ и x_0 – найденное каким-либо способом грубое приближение к корню ξ. Последовательно применяя рекуррентное соотношение $x_{k+1} = g(x_k)$, образуем итерационную последовательность:

$$x_0,\ \ x_1 = g(x_0),\ \ x_2 = g(x_1),\ \ \ldots x_n = g(x_{n-1}).$$

Процесс построения итерационной последовательности имеет простую геометрическую интерпретацию. Рисунки 2.8 и 2.9 отображают примеры сходящейся и расходящейся последовательностей.

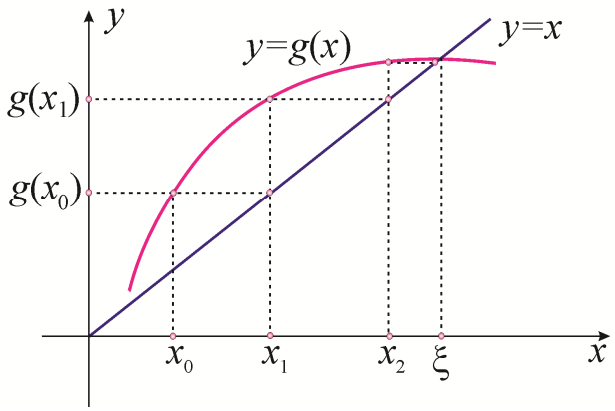


Рис. 2.8. Пример сходящейся последовательности.

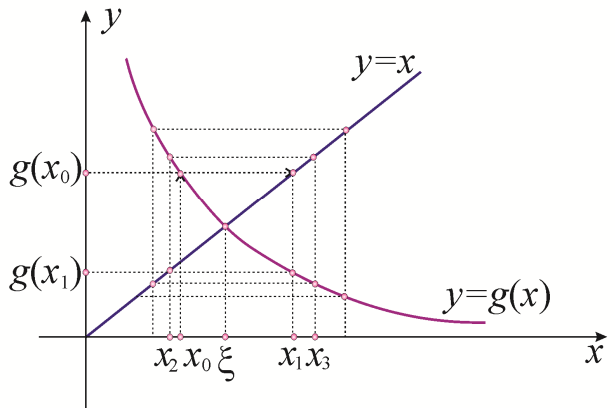


Рис. 2.9. Пример расходящейся последовательности.

Итерационный процесс сходится к значению ξ, такому, что $f(\xi)=0$, если интервал $[a,b]$ является интервалом изоляции корня уравнения (2.20) (т.е. внутри него не содержится других корней этого уравнения) и во всех точках этого интервала производная $g'(x)$ для любых $x \in [a,b]$ удовлетворяет неравенству:

$$|g'(x)| \le q, \qquad (2.21)$$

где константа $q: 0 < q < 1$.

При этом, если производная $g'(x)$ на отрезке $[a,b]$ положительна, то последовательные приближения $x_{k+1}=g(x_k)$ сходятся к корню ξ монотонно (рисунок 2.5), а если – отрицательна, последовательные приближения колеблются около корня ξ (рисунок 2.6). Это означает, что в случае положительной производной $g'(x)$ достаточно выбрать лишь начальное приближение x_0, принадлежащее интервалу $[a,b]$, и тогда все остальные приближения автоматически будут содержаться в этой окрестности. В противном случае для того, чтобы к интервалу $[a,b]$ принадлежали все приближения x_k (k =2,3,…), необходимо, чтобы к нему принадлежали x_0 и x_1, лежащие по разные стороны от корня ξ.

Погрешности метода на k-м и (k+1)-м шагах удовлетворяют неравенству $|x_{k+1}-\xi| \le q|x_k-\xi|$, т.е. метод имеет линейную сходимость со значением β=q согласно оценке (2.3). Процесс вычислений следует прекращать при выполнении условия

$$|x_{k+1}-x_k| \le \frac{1-q}{q}|x_{k+1}-x_k| \le \varepsilon,$$

где ε – заданная точность.

При использовании метода простых итераций основным моментом является выбор функции $g(x)$ в уравнении (2.20), эквивалентном исходному. Производная функции $g(x)$ должна удовлетворять условию сходимости процесса (2.20). При этом следует помнить, что скорость сходимости последовательности $\{x_k\}$ к корню ξ тем выше, чем меньше число q.

Преобразование исходного уравнения к виду (2.20) можно осуществить различными способами, при этом в одних случаях величина $|g'(x)|$ вблизи корня окажется малой, а в других – большой, что скажется на скорости сходимости метода.

Рассмотрим традиционные способы преобразования уравнения общего вида (2.1) к форме (2.20). Преобразуем (2.1) следующим образом:

$$x = x - \mu f(x), \qquad (2.22)$$

где μ – отличная от нуля константа.

Если $f'(x) > 0$, из условий

$$|1 - \mu f'(x)| < 1, \qquad -1 < 1 - \mu f'(x), \qquad 1 - \mu f'(x) < 1$$

получим: $\mu > 0, \quad \mu < \dfrac{2}{f'(x)}$.

Полагая $\mu = \dfrac{1}{\max\limits_{[a,b]} |f'(x)|} = \dfrac{1}{M}$, имеем итерационную формулу: $x_{n+1} = x_n - \dfrac{f(x_n)}{M}$. Аналогично, если $f'(x) < 0$, можно положить: $x_{n+1} = x_n + \dfrac{f(x_n)}{M}$.

2.9 Численные примеры. Реализация в пакете Matlab

Пример 3. Методом хорд найти корень уравнения $x^2 - \cos x - 1 = 0$, локализованный на отрезке $[1,1.5]$, с точностью 10^{-4}.

Далее приведен листинг исполняемого кода программы, решающей данную задачу.

```
%вычисление корня уравнения методом хорд
a=1; % левая граница отрезка локализации
b=1.5; % правая граница отрезка локализации
del=abs(fun(a));
ep=0.0001; %априорное задание точности
while del>ep
```

```
        c=a-fun(a)*(b-a)./(fun(b)-fun(a)); % вычисление текущего
приближения к корню
        if fun(a)*fun(c)<0 % организация процедуры сужения отрез-
ка локализации
            b=c;
        else a=c;
        del=abs(fun(c)); %проверка условия достижения точности
        end
    end
    disp(c)

    function z=fun(x)
    z=x.^2-cos(x)-1;
```

В рабочем окне программы представлен результат работы программы «Метод секущих». Для того чтобы увидеть больше значащих цифр, установите формат **`format long`**:

```
>> 1.176481610626940
```

При записи ответа оставляем четыре верных знака и не более одного-двух сомнительных: x=1.17648.

Для того, чтобы решить задачу определения приближенного значения корня средствами пакета Matlab можно воспользоваться следующим функционалом.

Функция **`fzero`** позволяет приближенно вычислить корень уравнения на некотором интервале или ближайший корень к заданному начальному приближению. В простейшем варианте **`fzero`** вызывается с двумя входными и одним выходным аргументом **`x=fzero('func_name',x0)`**, где **`func_name`** – имя файл-функции, вычисляющей левую часть уравнения, **`x0`** – начальное приближение к корню, **`x`** – найденное приближенное значение корня.

Пример 4. Найти приближенное значение корня уравнения $x^2-\cos x-1=0$ на отрезке $[1,1.5]$, используя встроенную функцию пакета Matlab **`fzero`**.

Вызов функции **fzero** имеет простой вид:

```
fzero('fun',[1 1.5])
```

Результат можно увидеть в окне команд. Алгоритм функции **fzero** находит корень уравнения с точностью **eps**.

```
ans =   1.176501939901832
>> eps
ans =    2.220446049250313e-016
```

Важной особенностью функции **fzero** является то, что она вычисляет только те корни, в которых функция меняет свой знак (например, при попытке вычисления корня уравнения $(x-1)^2=0$ с помощью этой же функции будет выведено сообщение об ошибке).

Инструментарий Optimization Toolbox ППП Matlab позволяет решать нелинейные уравнения с помощью функции **fsolve**. В общем случае вызов этой функции со всеми принятыми по умолчанию настройками имеет вид **x=fsolve('func_name',x0,optimset('fsolve'))**. Данная функция будет работать корректно и в случае определения кратных корней уравнения.

Пример 5. Найти приближенное значение корня уравнения $x^2-\cos x-1=0$ на отрезке $[1,1.5]$, используя встроенную функцию пакета Matlab **fsolve**.

Оформим вызов указанной функции:

```
fsolve('fun',1.5,optimset('fsolve')).
```

Результат отображается в окне команд:

```
Optimization terminated: first-order optimality is less than
options.TolFun.
ans =   1.176501969943108
```

Кроме того, для нахождения корней полинома в пакете Matlab предусмотрена встроенная функция **roots**, возвращающая вектор-столбец, компоненты которого являются корнями полинома. Входным параметром этой функции является вектор коэффициентов полинома, записанного в порядке убывания степеней: **roots(C)**.

Пример 6. Найти корень уравнения $x^3 + 8x^2 - 14x - 20 = 0$, используя функцию **roots(C)**.

Вызовем соответствующую функцию, определим для нее вектор входных параметров (формат данных – `format long`):

```
P=[1 8 -14 20];
roots(P)
```

Результат в окне команд:

```
ans =
 -9.663015735855133
  0.831507867927567 + 1.174028061895200i
  0.831507867927567 - 1.174028061895200i
```

Контрольные вопросы

1. В чем заключается задача изоляции корней?

2. В чем суть графического метода отделения корней? Какие свойства функции одной переменной используются для проверки правильности локализации корня и его единственности на отрезке?

3. Поясните геометрический смысл методов: половинного деления, секущих, касательных.

4. Назовите основную сущность итерационных методов.

5. Какова последовательность действий при решении уравнения методом простых итераций?

6. Почему в методе касательных начальное приближение $x_0 \in [a,b]$ целесообразно выбирать из условия $f(x_0) \cdot f''(x_0) > 0$?

7. Чему равны порядки сходимости рассмотренных методов?

8. Поясните, почему метод секущих можно считать частным случаем метода Ньютона, а метод Ньютона – частным случаем метода Чебышева.

Индивидуальные задания

1. Используя графический метод, найти область существования решения предложенного в варианте уравнения. На основании аналитического подхода удостоверится в правильности выбора отрезков локализации корней. Выбрать

для анализа только один промежуток существования корня такой, для которого вторая производная $f''(x)$ знакопостоянна.

2. Найти корень уравнения $f(x)=0$ в найденной области с точностью $\varepsilon = 10^{-4}$ указанными методами в соответствии с номером варианта. Выполнить программную реализацию алгоритмов указанных методов.

3. Определить число итераций каждого метода, которое требуется для достижения указанной точности.

4. Вычислить корни уравнения, используя возможности ППП Matlab. Сравнить решения, полученные всеми способами. Сделать вывод.

5. Оформить отчет по лабораторной работе.

Номер варианта	**Уравнение** $f(x)=0$	**Численные методы**
1	$0.5x^3 - \cos x + 1 = 0$	Метод половинного деления Метод Ньютона Метод простых итераций
2	$0.2x^3 - \sin x + 2 = 0$	Разностный метод Ньютона Метод Чебышева Метод простых итераций
3	$5\cos x - \sqrt{x} = 0$	Комбинированный метод хорд и касательных Метод Чебышева Метод простых итераций
4	$x^3 + 11 \cdot x^2 - 8 \cdot x - 25 = 0$	Метод половинного деления Двушаговый метод Ньютона Метод простых итераций
5	$2\sin x - \sqrt{x+2} = 0$	Метод хорд Упрощенный метод Ньютона Метод простых итераций
6	$2\sin x - 2^x + 1 = 0$	Метод хорд Метод Ньютона Метод простых итераций
7	$x^3 + x^2 + 3 \cdot x - 13 = 0$	Метод половинного деления Разностный метод Ньютона Метод простых итераций
8	$\ln(x+5) - \cos x - 1 = 0$	Комбинированный метод хорд и касательных Метод Ньютона Метод простых итераций

9	$x^3 + 3.5x^2 + 12 \cdot x + 19 = 0$	Метод хорд Метод Стеффенсена Метод простых итераций
10	$x^3 + 6x^2 - 17 \cdot x - 21 = 0$	Метод половинного деления Упрощенный метод Ньютона Метод простых итераций
11	$\ln(x+5) - \sin x - 1 = 0$	Упрощенный метод Ньютона Метод Ньютона Метод простых итераций
12	$8x^3 + 2x^2 + 3.1 \cdot x + 4 = 0$	Метод половинного деления Метод Чебышева Метод простых итераций
13	$2\cos x - \sqrt{x+5} + 3 = 0$	Комбинированный метод хорд и касательных Двушаговый метод Ньютона Метод простых итераций
14	$x^3 + 2x^2 + 11 \cdot x + 17 = 0$	Метод половинного деления Двушаговый метод Ньютона Метод простых итераций
15	$10\sin x - \sqrt{x+3} = 0$	Метод половинного деления Упрощенный метод Ньютона Метод простых итераций
16	$3x^3 + 3x^2 + 2.5 \cdot x + 10 = 0$	Метод Ньютона Метод Чебышева Метод простых итераций
17	$\ln(x+7) - 5\sin x = 0$	Метод хорд Разностный метод Ньютона Метод простых итераций
18	$2\sin x - 0.2x^2 + 2 = 0$	Комбинированный метод хорд и касательных Упрощенный метод Ньютона Метод простых итераций
19	$5\cos x + 0.5x = 0$	Метод Ньютона Метод Стеффенсена Метод простых итераций
20	$x^3 + 3 \cdot x^2 + 3 \cdot x + 4 = 0$	Метод половинного деления Метод Ньютона Метод простых итераций

3 РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

Преобразования, выполняемые при решении многих прикладных задач, приводят их математические постановки к системам линейных алгебраических уравнений (СЛАУ). К СЛАУ сводятся, например, вычислительные схемы, построенные на основе преимущественного числа численных методов для решения дифференциальных уравнений, уравнений с частными производными, интегральных уравнений. СЛАУ, возникающие в приложениях, часто отличаются достаточно большой размерностью и требовательностью к точности реализуемых процедур. Особая значимость данного раздела вычислительной математики привела к тому, что на практике развитие получили методы, относящие к различным группам – универсальные и специализированные. В настоящее время вычислительная математика обладает достаточно большим спектром численных алгоритмов решения СЛАУ. Практически все известные пакеты прикладных математических программ включают возможность решения задач в постановке СЛАУ. Для того, чтобы сделать выбор в пользу того или иного метода необходимо ориентироваться на специфику рассматриваемой задачи, понимать особенности алгоритмической реализации численных процедур, знать способы оценки точности полученных решений и границы применимости численных методов.

Методы решения алгебраических задач (решение СЛАУ, вычисление определителей, обращение матриц, задачи на собственные значения) можно разделить на два основных класса: прямые и итерационные.

Прямые методы позволяют найти решение задачи за конечное число арифметических операций, которое можно априорно определить. Если бы в практике расчетов не возникали бы ошибки округления, то полученные прямыми методами решения всегда были бы точными. Реализация методов этой группы как вычислительных процедур дает способ построения приближенного решения задачи, который во многих случаях оказывается эффективным в применении.

Методы ***итерационной*** группы позволяют построить итерационный процесс и найти решение задачи как предел бесконечной последовательности приближений, вычисляемых по унифицированной вычислительной схеме. Особенностью итерационных методов является то, что каждое новое приближение к решению определяется с использованием приближения, найденного на предыдущей итерации.

Будем рассматривать подходы к построению численных методов решения СЛАУ. Введем постановку задачи. Пусть дана система n линейных алгебраических уравнений с n неизвестными.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n. \end{cases} \qquad (3.1)$$

или в матрично-векторном виде:

$$AX = B, \qquad (3.1')$$

где B – вектор свободных коэффициентов, A – вещественная матрица $n \times n$ коэффициентов системы (3.1), X – вектор неизвестных. Эффективность способов решения системы (3.1) во многом зависит от структуры и свойств матрицы A. Будем считать, что средствами, известными из курса линейной алгебры, проведено исследование СЛАУ вида (3.1). В результате установлено, что система совместна и имеет единственное решение (в частности, $\det A \neq 0$).

Следует заметить, что при программной реализации численных методов решения алгебраических задач могут возникать серьезные проблемы, связанные как с большим количеством вычислений и большим объемом памяти, используемой для хранения необходимой информации, так и с неустойчивостью методов, приводящей к ошибкам округления в случае решения плохо обусловленных систем.

3.1 Прямые методы

В силу больших вычислительных затрат весьма привлекательные в теоретическом плане и используемые на практике (при небольшом числе неизвест-

ных n) формулы Крамера и метод обратной матрицы становятся трудно применимыми при их реализации в виде численных алгоритмов. Наиболее известным способом решения системы (3.1) является методы, объединенные под названием «метод Гаусса», суть которых заключается в реализации процедуры последовательного исключения неизвестных.

Метод Гаусса с выбором главного элемента. Будем поэтапно преобразовывать систему (3.1), исключая последовательно сначала x_1 из второго, третьего, …, n-го уравнений, затем x_2 из третьего, четвертого, …, n-го уравнений и т. д. на (n-1)-м этапе *прямого хода метода Гаусса* систему (3.1) приведем к треугольному виду:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ \ldots\ldots\ldots a_{22}^{(1)}x_2 + \ldots + a_{2n}^{(1)}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \end{cases} \qquad (3.2)$$

Коэффициенты этой системы могут быть получены из коэффициентов системы (3.1) последовательным пересчетом по формулам:

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} a_{kj}^{(k-1)}, \quad b_i^{(k)} = b_i^{(k-1)} - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} b_k^{(k-1)}, \qquad (3.3)$$

где k – номер этапа, который изменяется от 1 до n-1, индексы i и j – от k+1 до n; кроме того, по определению полагаем $a_{ij}^{(0)} = a_{ij}$, $b_i^{(0)} = b_i$.

Трапециевидная структура системы (3.2) позволяет последовательно одно за другим вычислять значения неизвестных, начиная с последнего. Этот процесс носит название *обратного хода метода Гаусса* и определяется формулой:

$$x_k = \frac{1}{a_{kk}^{(k-1)}} \left(b_k^{(k-1)} - \sum_{j=k+1}^{n} a_{kj}^{(k-1)} x_j \right), \quad k = n, n-1, \ldots, 2, 1 \qquad (3.4)$$

Принимая во внимание, что алгоритм метода Гаусса предусматривает цикличность выполняемых операций, а также то, что от хранения промежуточных результатов можно отказаться, перепишем алгоритм в упрощенном виде:

1) для $k = 1,2,..n-1,\ \ i = k+1,...n,\ \ t_{ik} = \dfrac{a_{ik}}{a_{kk}},\ \ b_i = b_i - t_{ik} b_k$;

2) для $j = k+1,...,n,\ \ a_{ij} = a_{ij} - t_{ik} a_{kj},\ \ x_n = \dfrac{b_n}{a_{nn}}$;

3) для $k = n-1,...,2,1,\ \ x_k = \dfrac{b_k - \sum_{j=k+1}^{n} a_{kj} x_j}{a_{kk}}$.

Таким образом, входными параметрами при реализации алгоритма будут являться квадратная матрица *A* и вектор свободных членов *B*. Далее алгоритм прямого метода требует выполнения трех вложенных циклов вычислений прямого входа и один цикл – обратного. В качестве выходных данных будет получен вектор решения *X*.

Поскольку вычисления на ЭВМ сопровождаются накоплением ошибок округления, то, анализируя формулы (3.3), можно заключить, что выполнение алгоритма может прекратиться или привести к неверным результатам, если знаменатели дробей на некотором этапе окажутся равными нулю или очень маленькими числами. Чтобы уменьшить влияние ошибок округления и исключить деление на нуль, на каждом этапе прямого хода уравнения системы обычно представляют так, чтобы деление производилось на наибольший по модулю в данном столбце элемент. Числа, на которые производится деление, называют *ведущими* или *главными элементами*, а сам метод Гаусса – *метод Гаусса с постолбцовым выбором главного элемента*. Устойчивость алгоритма к погрешностям исходных данных и результатов промежуточных вычислений можно еще больше усилить, если выполнять деление на каждом этапе на элемент, наибольший по модулю во всей матрице. Такая модификация метода Гаусса носит название *метода главных элементов*.

Метод прогонки. При реализации математических моделей в постановке дифференциальных или уравнений с частными производными на практике часто возникает необходимость в решении СЛАУ, матрицы которых являются

слабозаполненными, в частотности, имеют ленточную структуру – ненулевые элементы располагаются на главной диагонали и на нескольких побочных.

Введем в рассмотрение СЛАУ следующего вида:

$$b_i x_{i-1} + c_i x_i + d_i x_{i+1} = r_i, \quad i = 1,2,...n, \quad b_1 = 0, \quad d_n = 0. \tag{3.5}$$

Система (3.5) имеет трехдиагональную структуру:

$$\begin{bmatrix} c_1 & d_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ b_2 & c_2 & d_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & b_3 & c_3 & d_3 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & b_{n-1} & c_{n-1} & d_{n-1} \\ 0 & 0 & 0 & 0 & \dots & 0 & b_n & c_n \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \dots \\ r_{n-1} \\ r_n \end{bmatrix} \tag{3.6}$$

Предполагая, что существуют такие наборы чисел δ_i, λ_i, при которых

$$x_i = \delta_i x_{i+1} + \lambda_i, \tag{3.7}$$

подставляя (3.7) в (3.5), получим:

$$x_i = -\frac{d_i}{c_i + b_i \delta_{i-1}} x_{i+1} + \frac{r_i - b_i \lambda_{i-1}}{c_i + b_i \delta_{i-1}}. \tag{3.8}$$

Последнее имеет место, если при всех $i = 1,2,...,n$ выполняются рекуррентные соотношения:

$$\delta_i = -\frac{d_i}{c_i + b_i \delta_{i-1}}, \quad \lambda_i = \frac{r_i - b_i \lambda_{i-1}}{c_i + b_i \delta_{i-1}}. \tag{3.9}$$

Процесс вычисления в силу условия $b_1 = 0$ может быть начат со значений: $\delta_1 = -\frac{d_1}{c_1}$, $\lambda_1 = \frac{r_1}{c_1}$ и продолжен далее по формулам (3.9).

При $i = n$ в силу $d_n = 0$ получим $\delta_n = 0$ и

$$x_n = \lambda_n = \frac{r_n - b_n \lambda_{n-1}}{c_n + b_n \delta_{n-1}}, \tag{3.10}$$

где λ_{n-1}, δ_{n-1} – уже известные с предыдущего шага числа. Далее по формулам (3.7) последовательно находятся $x_{n-1,}x_{n-2},...,x_1$, $i = n-1, n-2,...,1$.

Описанное решение уравнений вида (3.5) носит название *метода прогонки*, который состоит из двух этапов: вычисление *прогоночных коэффициентов* по формулам (3.9) – *прямая прогонка* и вычисление неизвестных по формулам (3.10), (3.7) – *обратная прогонка*.

Для успешного применения метода прогонки требуется, чтобы в процессе вычислений не возникало ситуаций с делением на ноль и накоплением ошибки округления.

Прогонку называют *корректной*, если знаменатель прогоночных коэффициентов (3.9) не обращается в нуль, и *устойчивой*, если $|\delta_i| < 1$, $i \in \{1,2,...,n\}$.

Следующее утверждение устанавливает достаточные условия корректности и устойчивости метода прогонки.

Утверждение. Пусть коэффициенты b_i и d_i уравнения (3.5) при i = 2,3,.., n–1 отличны от нуля и пусть $|c_i| > |b_i| + |d_i| \quad \forall \quad i = 1,2,...,n.$ Тогда метод прогонки является корректным и устойчивым (т.е. $c_i + b_i\delta_{i-1} \neq 0$, $|\delta| < 1$).

Схема Халецкого. Рассмотрим систему линейных уравнений, записанную в матричном виде:

$$AX = D, \tag{3.11}$$

где $A = (a_{ij})$ – квадратная матрица, $i, j = 1,2,...,n$, и

$$X = \begin{pmatrix} x_1 \\ ... \\ x_n \end{pmatrix}, \quad D = \begin{pmatrix} a_{1,n+1} \\ ... \\ a_{n,n+1} \end{pmatrix}$$

– векторы-столбцы.

Представим матрицу A в виде произведения $A=B\cdot C$, где

$$C = \begin{pmatrix} 1 & c_{12} & \dots & c_{1n} \\ 0 & 1 & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}, \; B = \begin{pmatrix} b_{11} & 0 & \dots & 0 \\ b_{21} & b_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}$$

– соответственно верхнетреугольная матрица с единичной диагональю и нижнетреугольная матрица. Тогда элементы b_{ij} и c_{ij} будут определяться по формулам:

$$\begin{cases} b_{i1} = a_{i1}, \\ b_{ij} = a_{ij} - \sum\limits_{k=1}^{j-1} b_{ik} c_{kj} \;\; (i \geq j > 1) \end{cases}$$

и

$$\begin{cases} c_{1j} = \dfrac{a_{1j}}{b_{11}}, \\ c_{ij} = \dfrac{1}{b_{ii}} \left(a_{ij} - \sum\limits_{k=1}^{i-1} b_{ik} c_{kj} \right) \;\; (1 < i < j). \end{cases}$$

Отсюда искомый вектор X может быть вычислен из цепочки уравнений

$$B \cdot Y = D, \;\; C \cdot X = Y. \qquad (3.12)$$

Так как матрицы B и C треугольные, то системы (3.12) легко решаются, а именно:

$$\begin{cases} y_1 = \dfrac{a_{1,n+1}}{b_{11}}, \\ y_i = \dfrac{1}{b_{ii}} \left(a_{in+1} - \sum\limits_{k=1}^{i-1} b_{ik} y_k \right) \;\; (i > 1) \end{cases} \qquad (3.13)$$

и

$$\begin{cases} x_n = y_n, \\ x_i = y_i - \sum\limits_{k=i+1}^{n} c_{ik} x_k \;\; (i < n). \end{cases} \qquad (3.14)$$

Из формул (3.14) видно, что числа y_i выгодно вычислять вместе с коэффициентами c_{ij}. Эта схема вычислений называется *схемой Халецкого*. В схеме применяется обычный контроль с помощью сумм.

Метод квадратных корней используется для решения линейной системы (3.11), у которой матрица A симметрическая, т.е.

$a_{ij} = a_{ji}$, $i, j = 1,2,\ldots,n$.

Он является более экономным и удобным по сравнению с методами решения систем общего вида, рассмотренными ранее.

Решение системы осуществляется в два этапа.

Прямой ход. Представим матрицу A в виде произведения двух взаимно транспонированных треугольных матриц:

$$A = T^T \cdot T,$$

где

$$T = \begin{pmatrix} t_{11} & t_{12} & \ldots & t_{1n} \\ 0 & t_{22} & \ldots & t_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & t_{nn} \end{pmatrix}, \quad T^T = \begin{pmatrix} t_{11} & 0 & \ldots & 0 \\ t_{12} & t_{22} & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ t_{1n} & t_{2n} & \ldots & t_{nn} \end{pmatrix}.$$

Перемножая матрицы T^T и T и приравнивая к матрице A, получим следующие формулы для определения t_{ij}:

$$\begin{cases} t_{11} = \sqrt{a_{11}}, \; t_{1j} = \dfrac{a_{1j}}{t_{11}} \;\; (j > 1), \\ t_{ii} = \sqrt{a_{ii} - \sum\limits_{k=1}^{i-1} t_{ki}^2} \;\; (1 < i \le n), \\ t_{ij} = \dfrac{a_{ij} - \sum\limits_{k=1}^{i-1} t_{ki} t_{kj}}{t_{ii}} \;\; (i < j), \\ t_{ij} = 0 \;\; (i > j). \end{cases}$$

После того, как матрица A найдена, систему (3.11) заменяем двумя эквивалентными ей системами с треугольными матрицами

$$T^T \cdot Y = D, \quad T \cdot X = Y. \tag{3.15}$$

Обратный ход. Записываем в развернутом виде системы (3.15):

$$\begin{cases} t_{11}y_1 = b_1, \\ t_{12}y_1 + t_{22}y_2 = b_2, \\ \qquad\qquad \dots \\ t_{1n}y_1 + t_{2n}y_2 + \dots + t_{nn}y_n = b_n, \end{cases}$$

$$\begin{cases} t_{11}x_1 + t_{12}x_2 + \dots + t_{1n}x_n = y_1, \\ \qquad t_{22}x_2 + \dots + t_{2n}x_n = y_2, \\ \qquad\qquad \dots \\ \qquad\qquad t_{nn}x_n = y_n. \end{cases}$$

Отсюда последовательно находим

$$y_1 = \frac{b_1}{t_{11}}, \quad y_i = \frac{b_i - \sum\limits_{k=1}^{i-1} t_{ki}y_k}{t_{ii}} \quad (i > 1),$$

$$x_n = \frac{y_n}{t_{nn}}, \quad x_i = \frac{y_i - \sum\limits_{k=i+1}^{n} t_{ik}x_k}{t_{ii}} \quad (i < n).$$

При вычислениях применяется обычный контроль с помощью сумм, причем при составлении суммы учитываются все коэффициенты соответствующей строки.

Заметим, что при действительных a_{ij} могут получиться чисто мнимые t_{ij}. Метод применим и в этом случае.

Метод квадратных корней дает большой выигрыш во времени по сравнению с рассмотренными ранее методами, так как, во-первых, существенно уменьшает число умножений и делений, во-вторых, позволяет накапливать сумму произведений без записи промежуточных результатов.

Итерационное уточнение корней уравнений. Как уже указывалось ранее, решение СЛАУ прямыми методами может привести к неудовлетворительному результату в силу низкой точности. В этом случае для уточнения полученных значений корней уравнений можно применить итерационный процесс.

Итак, пусть найдено приближенное решение $x^{(0)}$ СЛАУ вида (3.1). Тогда, полагая точное решение $x = x^{(0)} + \delta^{(1)}$, где $\delta^{(1)}$ – вектор поправки, будем иметь

уравнение $A\left(x^{(0)}+\delta^{(1)}\right)=B$ или $A\delta^{(1)}=r^{(1)}$, где $r^{(1)}=B-Ax^{(0)}$ носит название *вектора невязки* для приближенного решения $x^{(0)}$. Таким образом, чтобы найти вектор $\delta^{(1)}$, нужно решить линейную систему $A\delta^{(1)}=r^{(1)}$ с прежней матрицей A и новым вектором свободных членов $r^{(1)}$. Полученный в результате этого вектор поправок складывается с вектором приближенного решения $x^{(1)}=x^{(0)}+\delta^{(1)}$.

Процесс уточнения корней может быть продолжен: $r^{(k)}=B-Ax^{(k-1)}$, $A\delta^{(k)}=r^{(k)}$, $x^{(k)}=x^{(k-1)}+\delta^{(k)}$. Его можно вести до тех пор, пока компоненты векторов $x^{(k)},x^{(k-1)}$ не совпадут в пределах заданной точности. На практике обычно ограничиваются одной-двумя итерациями.

3.2 Итерационные методы

Во многих практических задачах приходится решать СЛАУ, размерность которых достаточно велика. В ряде случае оптимальными в применении и наиболее эффективными могут оказаться методы приближенного решения, принадлежащие итерационной группе.

Метод простых итераций. Запишем систему (3.1) в следующем эквивалентном виде:

$$\begin{cases} x_1 = d_{11}x_1 + d_{12}x_2 + \ldots + d_{1n}x_n + c_1, \\ x_2 = d_{21}x_1 + d_{22}x_2 + \ldots + d_{2n}x_n + c_2, \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n = d_{n1}x_1 + d_{n2}x_2 + \ldots + d_{nn}x_n + c_n, \end{cases} \tag{3.16}$$

или в матрично-векторной форме:

$$X = D \cdot X + C. \tag{3.17}$$

Итерационный процесс, определяемый рекуррентным равенством

$$X^{(k)} = D \cdot X^{(k-1)} + C,$$

где $k = 1,2,...$, носит название *метода простых итераций*.

Начиная с некоторого произвольного вектора $X^{(0)} = \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \\ \dots \\ x_n^{(0)} \end{pmatrix}$, на каждой *k*-ой итерации соответствующее *k*-е приближение определяется через значение вектора *X* на *k*–1-ой итерации:

$$\begin{cases} x_1^{(k)} = d_{11}x_1^{(k-1)} + d_{12}x_2^{(k-1)} + \ldots + d_{1n}x_n^{(k-1)} + c_1, \\ x_2^{(k)} = d_{21}x_1^{(k-1)} + d_{22}x_2^{(k-1)} + \ldots + d_{2n}x_n^{(k-1)} + c_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k)} = d_{n1}x_1^{(k-1)} + d_{n2}x_2^{(k-1)} + \ldots + d_{nn}x_n^{(k-1)} + c_n. \end{cases} \qquad (3.18)$$

Доказано, что итерационный процесс сходится к точному решению системы (3.1), если модули диагональных элементов каждой строки матрицы *A* превышают сумму модулей недиагональных элементов этой же строки или аналогичное условие выполняется для столбцов матрицы

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \ \ (i = 1,\ldots,n), \qquad |a_{ii}| > \sum_{j \neq i} |a_{ij}| \ \ (j = 1,\ldots,n).$$

Для элементов матрицы *D* условия сходимости могут быть сформулированы следующим образом: итерационная последовательность точек *n*-мерного пространства

$$X^{(0)}, X^{(1)}, \ \ldots, \ X^{(k)}, \ \ldots$$

сходится и ее предел является решением системы (3.1) при выполнении одного из условий

$$\sum_{j=1}^{n} |d_{ij}| \le \lambda_i < 1, \ \ i = \overline{1,n}, \qquad (3.19)$$

$$\sum_{i=1}^{n} |d_{ij}| \le \gamma_j < 1, \ \ j = \overline{1,n}. \qquad (3.20)$$

Оценка погрешности приближенного решения $X^{(k)}$ может быть вычислена с помощью формул, если выполнено условие (3.19):

$$\max \left| X_i - X_i^{(k)} \right| \le \frac{\lambda}{1-\lambda} \max \left| X_i^{(k)} - X_i^{(k-1)} \right|, \ \ i = \overline{1,n}, \qquad (3.21)$$

где $\lambda = \max\limits_{i=\overline{1,3}} \lambda_i$, или, если выполнено условие (3.20):

$$\sum_{i=1}^{n} \left| X_i - X_i^{(k)} \right| \le \frac{\gamma}{1-\gamma} \sum_{i=1}^{n} \left| X_i^{(k)} - X_i^{(k-1)} \right|, \tag{3.22}$$

где $\gamma = \max\limits_{j=\overline{1,3}} \gamma_j$

Для установления момента прекращения итераций при достижении заданной точности ε может быть использована формула

$$\max \left| X_i^{(k)} - X_i^{(k-1)} \right| \le \frac{1-\lambda}{\lambda} \varepsilon . \tag{3.23}$$

Метод Зейделя. Под *методом Зейделя* обычно понимается такая модификация метода простых итераций, при которой на каждом шаге итерационного процесса при вычислении k-го приближения переменной x_i учитываются уже найденные ранее k-е приближения неизвестных $x_1, x_2, \ldots, x_{i-1}$. Таким образом, вычисления производятся по формулам:

$$\begin{cases} x_1^{(k)} = d_{11} x_1^{(k-1)} + d_{12} x_2^{(k-1)} + \ldots + d_{1n} x_n^{(k-1)} + c_1, \\ x_2^{(k)} = d_{21} x_1^{(k)} + d_{22} x_2^{(k-1)} + \ldots + d_{2n} x_n^{(k-1)} + c_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k)} = d_{n1} x_1^{(k)} + d_{n2} x_2^{(k)} + \ldots + d_{n,n-1} x_{n-1}^{(k)} + d_{nn} x_n^{(k-1)} + c_n. \end{cases} \tag{3.24}$$

В более компактном виде итерационный процесс метода Зейделя можно записать следующим образом:

$$x_i^{(k)} = \sum_{j=1}^{i-1} d_{ij} \cdot x_j^{(k)} + \sum_{j=i}^{n} d_{ij} \cdot x_j^{(k-1)} + c_i, \quad k = 1,2,\ldots.$$

Условия сходимости и оценки погрешности метода простых итераций остаются верными и для метода Зейделя. Оба метода имеют линейную сходимость. Обычно метод Зейделя дает лучшую сходимость по сравнению с методом простых итераций, хотя имеются и обратные примеры.

Существует возможность автоматического преобразования исходной системы к виду, обеспечивающему сходимость итерационного процесса метода Зейделя.

Для этого умножим левую и правую части уравнения (3.1′) на A^T, получим равносильную систему:

$$W \cdot X = P, \quad W = A^T \cdot A, \quad P = A^T \cdot B. \tag{3.25}$$

Система уравнений (3.25) называется *нормальной* и обладает свойствами: матрица W – симметрическая; все элементы главной диагонали матрицы W положительны.

Указанные свойства позволяют приводить систему (3.25) к виду, пригодному для итерационного процесса Зейделя:

$$x_i = \sum_{i \neq j} d_{ij} x_j + \beta_j, \; i = \overline{1,n}, \quad d_{ij} = -\frac{w_{ij}}{w_{ii}}, i \neq j, \quad \beta_i = \frac{p_i}{w_{ii}}, \; i = \overline{1,n}. \tag{3.26}$$

Метод Якоби. Вернемся к рассмотрению задачи (3.1). После выяснения условия, которому должна удовлетворять матрица коэффициентов приведенной системы (3.17) для сходимости МПИ (3.18), следует осуществить приведение системы (3.1) к виду (3.17) так, чтобы это условие выполнялось. Рассмотрим один из способов такого приведения, достаточно эффективный в определенных случаях.

Представим матрицу A системы (3.1) в виде

$$A = L + G + R,$$

где G – диагональная, а L и R – соответственно левая и правая строго треугольные (т.е. с нулевой диагональю) матрицы. Тогда система (3.1) может быть записана в виде:

$$L \cdot X + G \cdot X + R \cdot X = B \tag{3.27}$$

и если на диагонали исходной матрицы нет нулей, то эквивалентной (3.1) задачей вида (3.17) будет

$$X = -G^{-1} \cdot (L + R) \cdot X + G^{-1} \cdot B, \tag{3.28}$$

т.е. в (3.17) и (3.18) следует положить

$$D = -G^{-1} \cdot (L + R), \quad C = G^{-1} \cdot B.$$

Основанный на таком приведении системы (3.1) к виду (3.17) метод простых итераций (3.18) называют *методом Якоби.* В векторно-матричных обозначениях он определяется формулой

$$X^{(k+1)} = -G^{-1} \cdot (L+R) \cdot X^{(k)} + G^{-1} \cdot B, \quad k = 0,1,2,\ldots . \tag{3.29}$$

Заметим, что обратной матрицей к матрице $G = (a_{ii})_{i=1}^{n}$ является диагональная матрица G^{-1} с элементами диагонали $g_{ii} = 1/a_{ii}$. Поэтому, чтобы записать метод Якоби (3.29) решения системы (3.1) в развернутом виде, достаточно выразить в каждом уравнении неизвестные, соответствующие диагональным элементам матрицы коэффициентов *A*, через остальные неизвестные:

$$\begin{cases} x_1 = -(a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n - b_1)/a_{11}, \\ x_2 = -(a_{21}x_1 + a_{23}x_3 + \ldots + a_{2n}x_n - b_2)/a_{22}, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = -(a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{n,n-1}x_{n-1} - b_n)/a_{nn}. \end{cases} \tag{3.30}$$

Далее реализуем итерационный процесс (3.29), воспользуюсь алгебраической записью системы (3.30). В уравнениях системы (3.30) припишем индексы, соответствующие номерам приближений (т.е. *k*=0,1,2,...):

$$\begin{cases} x_1^{(k+1)} = -(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \ldots + a_{1n}x_n^{(k)} - b_1)/a_{11}, \\ x_2^{(k+1)} = -(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + \ldots + a_{2n}x_n^{(k)} - b_2)/a_{22}, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k+1)} = -(a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \ldots + a_{n,n-1}x_{n-1}^{(k)} - b_n)/a_{nn}. \end{cases} \tag{3.31}$$

Одной из модификаций метода Якоби (3.31) является метод, реализующий итерационный процесс согласно процедуре метода Зейделя (3.24), при этом, для расчета значений неизвестных на текущей итерации используются уже найденные значения на этой же итерации:

$$\begin{cases} x_1^{(k+1)} = -(a_{12}x_2^{(k)} + a_{13}x_3^{(k)} + \ldots + a_{1n}x_n^{(k)} - b_1)/a_{11}, \\ x_2^{(k+1)} = -(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \ldots + a_{2n}x_n^{(k)} - b_2)/a_{22}, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_n^{(k+1)} = -(a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + \ldots + a_{n,n-1}x_{n-1}^{(k+1)} - b_n)/a_{nn}. \end{cases} \tag{3.32}$$

где $k = 0,1,2,...,\ x^{(0)} = \left(x_1^{(0)},...,x_n^{(0)}\right)^T$ – заданный вектор начальных приближений.

Достаточные условия сходимости (3.19) – (3.20) методов простой итерации и Зейделя являются достаточными условиями сходимости для метода Якоби (3.31) и метода Якоби в модификации Зейделя (3.32).

3.3 Численные примеры. Реализация в пакете Matlab

Пример 1. Методом простой итерации найти решение системы уравнений

$$\begin{cases} 11x_1 + 2.3x_2 - 2.5x_3 = 1.4 \\ 3.1x_1 + 11x_2 - 2.8x_3 = 8.9 \\ 3.3x_1 - 0.9x_2 + 11x_3 = 6.7 \end{cases} \qquad (3.33)$$

с точностью $\varepsilon = 10^{-4}$.

1. Для реализации метода простой итерации приведем систему (3.33) к виду (3.16):

$$\begin{cases} x_1 = -0.1x_1 - 0.23x_2 + 0.25x_3 + 0.14 \\ x_2 = -0.31x_1 - 0.1x_2 + 0.28x_3 + 0.89 \\ x_3 = -0.33x_1 + 0.09x_2 - 0.1x_3 + 0.67 \end{cases}$$

2. Проверим выполнение достаточного условия сходимости (3.19):

$\lambda_i < 1,$

$\lambda = \max\{0.58;0.69;0.52\} = 0.69.$

3. В качестве начального приближения выберем вектор свободных коэффициентов в преобразованной СЛАУ:

$X^T = (0.14;0.89;0.67).$

4. Реализуем итерационный процесс в Matlab, создав следующий скрипт-файл:

```
D=[-0.1 -0.23 0.25; -0.31 -0.1 0.28; -0.33 0.09 -0.1]; %ини-
циализация матрицы D
C=[0.14; 0.89; 0.67]; % инициализация вектора C
X0=C; % задание начального приближения
eps=10^-4; % задание требуемой точности
```

```
del=100; % задание (для входа в цикл) текущего значения точности вычислений
lya=0.69; % задание числа лямбда
k=0; % число итераций
while del>eps
    X1=D*X0+C;
    del=(lya/(1-lya)).*max(abs(X1-X0));
    X0=X1;
    k=k+1;
end
X1 % вывод решения
k % вывод значения числа итераций, требуемых для достижения точности
```

5. Решение СЛАУ (3.33), полученное с заданной точностью ε, показано на экране:

```
X1 =
   0.078255395008189
   0.956020604174063
   0.663839699276815
k =
     8
```

Таким образом, для достижения точности $\varepsilon = 10^{-4}$ потребовалось сделать 8 шагов методом простых итераций.

При записи ответа оставляем четыре верных знака и не более одного-двух сомнительных:

$x_1 = 0.07825,$

$x_2 = 0.95602,$

$x_3 = 0.66384.$

Для решения систем линейных уравнений средствами пакета Matlab применяются операторы возведения в степень и умножения, действие которых определяется правилами линейной алгебры.

Пример 2. Используя встроенные функции пакета Matlab, найдите решение системы линейных уравнений (3.33).

Решение системы (34) средствами пакета Matlab может быть записано следующим образом:

```
A=[11 2.3 -2.5; 3.1 11 -2.8; 3.3 -0.9 11];
B=[1.4; 8.9; 6.7];
Xex=A^-1*B
ksi=max(abs(A*Xex-B))
```

Результат вычисления значений вектора X и невязки найденного решения показан на экране:

```
Xex =
   0.078250365205257
   0.956014763436902
   0.663835189265079

ksi =
    1.776356839400251e-015
```

Контрольные вопросы

1. В чем заключается основное преимущество метода Гаусса с выбором главного элемента? Почему схемы Гаусса с выбором главного элемента дают более точный результат, нежели простая схема Гаусса?

2. Для каких специфических систем линейных алгебраических уравнений применим метод прогонки? На чем основана более высокая эффективность метода прогонки по сравнению с методом Гаусса?

3. В чем заключаются этапы прямой и обратной прогонки?

4. Какие существуют способы приведения исходной матрицы к виду, пригодному для решения методом простой итерации?

5. Каким образом может быть выбран вектор начальных приближений?

6. Назовите достаточное условие сходимости итерационных методов.

7. В чем заключается преимущество метода Зейделя при программировании?

Индивидуальные задания

Дана система линейных алгебраических уравнений $AX = B$.

1. Реализовать алгоритм указанного прямого метода решения предложенной системы в ППП Matlab. Вычислить невязку полученного решения.

2. Привести систему уравнений к виду, пригодному для итерационного процесса указанного метода, реализовать метод в пакете Matlab и решить систему уравнений с точностью $\varepsilon = 10^{-4}$ и исходными данными, приведенными в таблице. За сколько шагов достигается заданная точность в каждом из методов?

3. Найти решение этой же системы, используя инструментальные средства пакета Matlab.

4. Сравните найденное решение с полученными при реализации прямого и итерационного методов, а также использовании встроенного функционала пакета Matlab.

5. Оформить отчет по лабораторной работе.

Номер варианта	**Значения матрицы *A* и вектора *B***	**Методы**
1	$A = \begin{pmatrix} 9.1 & -1.2 & -2 \\ 3 & 10.5 & 4.3 \\ -1.01 & 4.5 & 11.5 \end{pmatrix}, B = \begin{pmatrix} 9.07 \\ 3.21 \\ -8.25 \end{pmatrix}$	1. Метод Гаусса с выбором главного элемента. 2. Метод Зейделя.
2	$A = \begin{pmatrix} 20.7 & -5.9 & 4.1 & -1.1 \\ -5.9 & 21.2 & -4.7 & -0.5 \\ 4.1 & -4.7 & 29.8 & 3.3 \\ -1.1 & -0.5 & 3.3 & 22.9 \end{pmatrix}, B = \begin{pmatrix} 7.12 \\ 8.07 \\ 7.1 \\ 6.8 \end{pmatrix}$	1. Метод квадратных корней. 2. Метод простой итерации.
3	$A = \begin{pmatrix} 10.39 & 0.25 & 0 & 0 \\ 2.25 & 6.28 & 0.98 & 0 \\ 0 & 1.32 & 9.31 & 1.04 \\ 0 & 0 & 3.35 & 12.28 \end{pmatrix}, B = \begin{pmatrix} 4.21 \\ 6.47 \\ 2.38 \\ 10.48 \end{pmatrix}$	1. Метод прогонки 2. Метод Якоби.

4	$A=\begin{pmatrix}22.34 & -4.21 & -1.61\\ 5.04 & 17.22 & 0.27\\ 1.92 & -2.99 & 18.37\end{pmatrix}$, $B=\begin{pmatrix}14.41\\ -6.44\\ 55.56\end{pmatrix}$	1. Схема Халецкого. 2. Метод Якоби в модификации Зейделя.
5	$A=\begin{pmatrix}5.18 & 0.25 & -0.44\\ -0.42 & 7.35 & 1.12\\ 1.14 & 0.12 & 12.83\end{pmatrix}$, $B=\begin{pmatrix}1.15\\ 0.86\\ 0.68\end{pmatrix}$	1. Метод Гаусса. 2. Метод простых итераций.
6	$A=\begin{pmatrix}11.54 & 2.22 & 1.21\\ 2.22 & 21.06 & -5.34\\ 1.21 & -5.34 & 16.45\end{pmatrix}$, $B=\begin{pmatrix}-9.93\\ -2.46\\ 6.81\end{pmatrix}$	1. Метод квадратных корней. 2. Метод простой итерации.
7	$A=\begin{pmatrix}17.51 & -1.12 & 4.64\\ -1.25 & 15.62 & 1.85\\ -0.53 & -3.5 & 15.31\end{pmatrix}$, $B=\begin{pmatrix}-1.05\\ -14.46\\ -17.73\end{pmatrix}$	1. Схема Халецкого. 2. Метод Якоби.
8	$A=\begin{pmatrix}18.2 & 1.4 & 0 & 0\\ -1.6 & 19.4 & -7.7 & 0\\ 0 & 1.9 & 21.5 & 4.8\\ 0 & 0 & 2.7 & 18.9\end{pmatrix}$, $B=\begin{pmatrix}32.76\\ 54.39\\ 59.18\\ -71.95\end{pmatrix}$	1. Метод прогонки 2. Метод Зейделя.
9	$A=\begin{pmatrix}22.42 & 4.21 & 1.85\\ 2.31 & 31.49 & 1.52\\ 3.49 & 4.85 & 28.72\end{pmatrix}$, $B=\begin{pmatrix}30.24\\ 4.095\\ 42.81\end{pmatrix}$	1. Метод Гаусса с выбором главного элемента. 2. Метод Якоби в модификации Зейделя.
10	$A=\begin{pmatrix}1.63 & 0.07 & 0.2\\ 0.07 & 2.44 & 0.3\\ 0.2 & 0.3 & 3.71\end{pmatrix}$, $B=\begin{pmatrix}1.34\\ 2.32\\ -1.42\end{pmatrix}$	1. Метод квадратных корней. 2. Метод простых итераций.
11	$A=\begin{pmatrix}2.11 & 0.78 & 0.45\\ -0.17 & 4.44 & 0.32\\ 2.14 & -1.3 & 15.05\end{pmatrix}$, $B=\begin{pmatrix}2.12\\ 5.69\\ -4.19\end{pmatrix}$	1. Метод Гаусса. 2. Метод простых итераций.
12	$A=\begin{pmatrix}21.4 & 5.1 & 0 & 0\\ -2.11 & 32.9 & 4.7 & 0\\ 0 & 1.9 & 21.5 & -3.2\\ 0 & 0 & 0.87 & 18.9\end{pmatrix}$, $B=\begin{pmatrix}32.7\\ 54.3\\ 59.1\\ -71\end{pmatrix}$	1. Метод прогонки. 2. Метод Зейделя.

13	$A=\begin{pmatrix}54.63 & 1.25 & 9.52\\ 4.01 & 19.63 & -1.52\\ -3.02 & 5.23 & 26.33\end{pmatrix}$, $B=\begin{pmatrix}1.44\\ 2.03\\ 18.96\end{pmatrix}$	1. Метод Гаусса с выбором главного элемента. 2. Метод Якоби.
14	$A=\begin{pmatrix}18.2 & 4.1 & 0 & 0\\ 5.4 & 21.5 & 2.14 & 0\\ 0 & -3.2 & 18.5 & 1.4\\ 0 & 0 & -1.11 & 11.6\end{pmatrix}$, $B=\begin{pmatrix}1.4\\ 9.8\\ 3.2\\ -4\end{pmatrix}$	1. Метод прогонки 2. Метод простой итерации.
15	$A=\begin{pmatrix}33.22 & -0.14 & 10.22\\ 4.56 & 22.14 & -1.58\\ 1.02 & 4.16 & 25.98\end{pmatrix}$, $B=\begin{pmatrix}5.63\\ 5.18\\ 9.44\end{pmatrix}$	1. Метод Гаусса. 2. Метод Зейделя.
16	$A=\begin{pmatrix}5.46 & 1.11 & 3.99\\ 1.11 & 18.52 & -2.36\\ 3.99 & -2.36 & 52.33\end{pmatrix}$, $B=\begin{pmatrix}11.25\\ 56.33\\ -14.2\end{pmatrix}$	1. Метод квадратных корней. 2. Метод простых итераций.
17	$A=\begin{pmatrix}10.2 & 1.4 & 0 & 0\\ -2.1 & 14 & 3.4 & 0\\ 0 & 2.3 & 11 & -1.5\\ 0 & 0 & 5 & 32\end{pmatrix}$, $B=\begin{pmatrix}18\\ 21\\ -9\\ 5\end{pmatrix}$	1. Метод прогонки 2. Метод Зейделя.
18	$A=\begin{pmatrix}1.5 & 5.12 & 4.64\\ 6.18 & -15.62 & 4.11\\ 4.5 & -3.5 & 18.91\end{pmatrix}$, $B=\begin{pmatrix}10.05\\ 24.4\\ 37.7\end{pmatrix}$	1. Схема Халецкого. 2. Метод Якоби.
19	$A=\begin{pmatrix}2.7 & 0.5 & 0.1 & -1.1\\ 0.5 & -4.7 & -2.2 & -0.5\\ 0.1 & -2.2 & 9.8 & 0.3\\ -1.1 & -0.5 & 0.3 & 5.9\end{pmatrix}$, $B=\begin{pmatrix}8\\ 9\\ 11\\ 2.1\end{pmatrix}$	1. Метод квадратных корней. 2. Метод простой итерации.
20	$A=\begin{pmatrix}12 & 2.69 & 0 & 0\\ -1.12 & 14 & -2.11 & 0\\ 0 & 1.65 & -15 & 3.25\\ 0 & 0 & 2.22 & 13\end{pmatrix}$, $B=\begin{pmatrix}4.21\\ 6.47\\ 2.38\\ 10.48\end{pmatrix}$	1. Метод прогонки 2. Метод Якоби.

4 РЕШЕНИЕ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ

Многие прикладные задачи сводятся к решению систем алгебраических или трансцендентных уравнений, которые в общем случае являются нелинейными. Поскольку аналитические методы решения систем нелинейных уравнений разработаны в существенно узком классе задач, большое значение на практике приобретают методы численного анализа.

Пусть требуется решить систему уравнений

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

где f_1, f_2, …, f_n – заданные нелинейные вещественнозначные функции n вещественных переменных $x_1, x_2, \ldots, x_n$.

Введем n – мерные векторы

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}, \quad F(X) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \ldots \\ f_n(x) \end{pmatrix}, \quad \overline{0} = \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

и систему (4.1) запишем в матрично-векторном виде:

$$F(X) = 0. \tag{4.1′}$$

Исходную задачу (4.1′) можно рассматривать как задачу о нулях нелинейного отображения $F : R_n \mapsto R_n$.

4.1 Метод простых итераций. Метод покоординатных итераций

Пусть система (1) преобразована к виду:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \ldots, x_n), \\ x_2 = \varphi_2(x_1, x_2, \ldots, x_n), \\ \ldots\ldots\ldots\ldots \\ x_n = \varphi_n(x_1, x_2, \ldots, x_n) \end{cases} \tag{4.2}$$

или:

$$X = \Phi(X). \qquad (4.2')$$

Рекуррентное равенство:

$$X^{(k+1)} = \Phi\left(X^{(k)}\right), \quad k = 0,1,2,..., \qquad (4.3)$$

определяет *метод простых итераций* для задачи (4.1) – (4.1′).

Если начать построение последовательности $X^{(k)}$ с некоторого вектора $X^{(0)}$ и продолжать по формуле (4.3), то при определенных условиях эта последовательность будет приближаться к вектору X^* – неподвижной точке отображения $\Phi(X)$: $R_n \to R_n$. Условия сходимости метода определяет следующее утверждение.

Утверждение. Пусть функция $\Phi(X)$ и замкнутое множество $M \subseteq D(\Phi) \subseteq R_n$ таковы, что:

1) $\Phi(X) \in M \quad \forall X \in M$;

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

Тогда $\Phi(X)$ имеет в M единственную неподвижную точку X^*, последовательность $X^{(k)}$, определяемая методом простых итераций (4.3), при любом $X^{(0)} \in M$ сходится к X^* и справедливы оценки:

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

При решении систем нелинейных уравнений, также как и в случае решения систем линейных алгебраических уравнений, можно ввести в рассмотрение модификацию метода простых итераций – аналог метода Зейделя, который в данном случае принято называть *методом покоординатных итераций*:

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

где $k = 0,1,2,\ldots$.

Оценка q имеет вид: $q = \max\limits_{X \in D} \|J(X)\|$ (норма матрицы Якоби не превосходит единицы для любого вектора $X \in D$), где $J(X)$ – матрица Якоби:

$$F'(X) = J(X) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}.$$

Для определённости и согласованности в качестве нормы вектора X и соответствующей нормы матрицы Якоби $J(X)$ для функций $\varphi_1(x_1, x_2)$ и $\varphi_2(x_1, x_2)$ выберем следующие нормы (представим на примере системы двух уравнений с двумя неизвестными):

$$\|X\| = \max_{X \in D} \{|x_1|; |x_2|\}, \ \|J\| = \max_{X \in D} \left\{ \left|\frac{\partial \varphi_1}{\partial x_1}\right| + \left|\frac{\partial \varphi_1}{\partial x_2}\right|; \left|\frac{\partial \varphi_2}{\partial x_1}\right| + \left|\frac{\partial \varphi_2}{\partial x_2}\right| \right\}.$$

Следует отметить, что при решении методом итераций системы нелинейных уравнений (4.1) в качестве области D можно считать множество точек вблизи точки пересечения кривых, определяемых уравнениями $f_1(x_1, x_2) = 0$ и $f_2(x_1, x_2) = 0$.

Рассмотрим способ преобразования системы уравнений (4.1) к виду (4.2). Преобразуем систему (4.1) так, чтобы в окрестности начальной точки $X^{(0)}$, близкой к искомому решению, выполнялось условие сходимости:

$\|J(X)\| < 1$.

Представим правые части уравнений системы (4.2) для случая двух переменных в виде:

$$\begin{cases} \varphi_1(x_1, x_2) = x_1 + \lambda_{11} f_1(x_1, x_2) + \lambda_{12} f_2(x_1, x_2), \\ \varphi_2(x_1, x_2) = x_2 + \lambda_{21} f_1(x_1, x_2) + \lambda_{22} f_2(x_1, x_2). \end{cases} \quad (4.4)$$

Последняя система будет равносильна исходной системе при условии, что числовая матрица λ_{ij} невырожденная, т.е.

$$\begin{vmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{vmatrix} \neq 0.$$

Для вычисления элементов матрицы λ_{ij} предположим, что норма матрицы Якоби для функций $\varphi_1(x_1, x_2)$ и $\varphi_2(x_1, x_2)$ равна нулю в точке $X^{(0)}$. Тогда, в силу непрерывности элементов матрицы Якоби, существует такая окрестность точки $X^{(0)}$, где $\|J(X)\| < 1$. Равенство $\|J(X)\| = 0$ означает, что матрица $J\left(X^{(0)}\right)$ – нулевая, т.е.

$$\left.\frac{\partial \varphi_1}{\partial x_1}\right|_{X^{(0)}} = 0, \left.\frac{\partial \varphi_1}{\partial x_2}\right|_{X^{(0)}} = 0, \left.\frac{\partial \varphi_2}{\partial x_1}\right|_{X^{(0)}} = 0, \left.\frac{\partial \varphi_2}{\partial x_2}\right|_{X^{(0)}} = 0.$$

Значения параметров $\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22}$ вычислим, используя последнее требование, решая последовательно две системы линейных уравнений:

$$\begin{cases} 1 + \lambda_{11}\dfrac{\partial f_1}{\partial x_1} + \lambda_{12}\dfrac{\partial f_2}{\partial x_1} = 0, \\ \lambda_{11}\dfrac{\partial f_1}{\partial x_2} + \lambda_{12}\dfrac{\partial f_2}{\partial x_2} = 0, \end{cases} \quad \begin{cases} \lambda_{21}\dfrac{\partial f_1}{\partial x_1} + \lambda_{22}\dfrac{\partial f_2}{\partial x_1} = 0, \\ 1 + \lambda_{21}\dfrac{\partial f_1}{\partial x_2} + \lambda_{22}\dfrac{\partial f_2}{\partial x_2} = 0. \end{cases} \tag{4.5}$$

4.2 Метод Ньютона и его модификации

Для нахождения приближенных решений системы уравнений (4.1) можно формально записать итерационный процесс:

$$X^{(k+1)} = X^{(k)} - A_k \cdot F\left(X^{(k)}\right), \quad k = 0,1,2,..., \tag{4.6}$$

определяющий большое семейство методов с матричными параметрами A_k. Положим $A_k = \left[F'\left(X^{(k)}\right)\right]^{-1}$, где $F'(X) = J(X)$ – матрица Якоби вектор-функции $F(X)$.

Таким образом получим явную формулу *метода Ньютона*:

$$X^{(k+1)} = X^{(k)} - \left(J\left(X^{(k)}\right)\right)^{-1} \cdot F\left(X^{(k)}\right), \quad k = 0,1,2,... \tag{4.7}$$

Перепишем формулу (4.7) в неявном виде:

$$F'\left(X^{(k)}\right)\left(X^{(k+1)} - X^{(k)}\right) = -F\left(X^{(k)}\right), \quad k = 0,1,2,... \tag{4.8}$$

Сравнивая (4.8) с формальным разложением $F(X)$ в ряд Тейлора:

$$F(X)=F\left(X^{(k)}\right)+F'\left(X^{(k)}\right)\left(X-X^{(k)}\right)+\frac{1}{2!}F''\left(X^{(k)}\right)\left(X-X^{(k)}\right)^2+\ldots, \qquad (4.9)$$

можно заключить, что последовательность $X^{(k)}$ в методе Ньютона получается пошаговой линеаризацией соответствующих нелинейных уравнений системы (4.1). При достаточной гладкости $F(X)$ и достаточно хорошем начальном приближении $X^{(0)}$ сходимость порождаемой методом Ньютона последовательности $X^{(k)}$ к решению X^* будет квадратичной и в многомерном случае.

Необходимость решения *n*-линейных задач, а также обращение матрицы производных на каждом шаге проводит к росту вычислительных затрат. Пути уменьшения таких затрат приводят к различным модификациям метода Ньютона.

Если матрицу Якоби вычислить и обратить всего один раз – в начальной точке $X^{(0)}$, получим *модифицированный метод Ньютона*:

$$X^{(k+1)}=X^{(k)}-\left(J\left(X^{(0)}\right)\right)^{-1}F\left(X^{(k)}\right), \quad k=0,1,2,... \qquad (4.10)$$

Этот метод требует значительно меньше вычислительных затрат на один итерационный шаг, но итераций при этом может потребоваться значительно больше для достижения заданной точности, метод имеет скорость сходимости геометрической прогрессии.

Вычисление и обращение матриц Якоби не на каждом шаге, а через несколько шагов приводят к *рекурсивному двушаговому* ступенчатому *процессу*:

$$\begin{cases} Z^{(k)}=X^{(k)}-A_kF\left(X^{(k)}\right), \\ X^{(k+1)}=Z^{(k)}-A_kF\left(Z^{(k)}\right). \end{cases} \qquad (4.11)$$

На базе метода Ньютона можно построить близкий к нему по поведению итерационный процесс, не требующий вычисления производных. Заменим частные производные в матрице Якоби разностными отношениями, подставив в формулу (4.7) матрицу $J\left(X^{(k)},h^{(k)}\right)^{-1}$, где

$$J(X,h)=\left(\frac{f_i\left(x_1,\ldots,x_j+h_j,\ldots,x_n\right)-f_i\left(x_1,\ldots,x_j,\ldots,x_n\right)}{h_j}\right)_{i,j=1}^{n}.$$

При удачном задании последовательности малых векторов $h^{(k)} = \left(h_1^{(k)}, h_2^{(k)}, .., h_n^{(k)}\right)$ получим *разностный метод Ньютона*:

$$X^{(k+1)} = X^{(k)} - \left[J\left(X^{(k)}, h^{(k)}\right)\right]^{-1} \cdot F\left(X^{(k)}\right), \tag{4.12}$$

имеющий сверхлинейную скорость сходимости.

4.3 Метод Брауна

В отличие от пошаговой линеаризации векторной функции $F(X)$, приведшей к методу Ньютона (4.7), *метод Брауна* предполагает проведение на каждом итерационном шаге поочередной линеаризации компонент вектор-функции $F(X)$. Приведем расчетные формулы метода Брауна в двумерном случае. Пусть требуется найти решение системы уравнений:

$$\begin{cases} f(x, y) = 0 \\ g(x, y) = 0 \end{cases} \tag{4.13}$$

и пусть уже получены приближения x_k, y_k. Итерационный процесс метода Брауна определяется соотношениями:

$$\begin{aligned} \hat{x}_k &= x_k - \frac{f(x_k, y_k)}{f'_x(x_k, y_k)} \\ q_k &= \frac{g(\hat{x}_k, y_k) \cdot f'_x(x_k, y_k)}{f'_x(x_k, y_k) \cdot g'_y(\hat{x}_k, y_k) - f'_y(x_k, y_k) \cdot g'_x(\hat{x}_k, y_k)} \\ p_k &= \frac{f(x_k, y_k) - q_k \cdot f'_y(x_k, y_k)}{f'_x(x_k, y_k)} \end{aligned} \tag{4.14}$$

$$x_{k+1} = x_k - p_k, \quad y_{k+1} = y_k - q_k.$$

4.4 Метод скорейшего спуска

Локальных характер сходимости всех рассмотренных выше методов затрудняет их применение в случаях, когда имеются проблемы с выбором хороших начальных приближений. В этом случае на помощь могут прийти численные методы оптимизации.

В этом методе решение системы (4.1) сводится к задаче отыскания минимумов функции

$$\Phi(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} f_i^2(x_1, x_2, \ldots, x_n). \qquad (4.15)$$

Так как функция (4.15) неотрицательная, то найдется точка $X^* = (x_1^*, x_2^*, \ldots x_n^*)$ такая, что

$$\Phi(x_1, x_2, \ldots, x_n) \geq \Phi(x_1^*, x_2^*, \ldots, x_n^*) \geq 0, \quad \forall (x_1, x_2, \ldots, x_n) \in R^n.$$

Следовательно, если удается найти точку X^*, минимизирующую функцию (4.15), и если при этом $\min\limits_{(x_1, x_2, \ldots, x_n) \in R^n} \Phi(x_1, x_2, \ldots, x_n) = \Phi(x_1^*, x_2^*, \ldots, x_n^*) = 0$, то точка X^* – истинное решение системы (4.1). Последовательность точек $X^{(k)}$ – приближений к точке X^* вычисляется по формуле:

$$X^{(k+1)} = X^{(k)} + \lambda_k \cdot \Phi(X^{(k)}), \qquad (4.16)$$

где $\Phi(X^{(k)})$ – вектор, определяющий направление минимизации, λ_k – скалярная величина, характеризующая величину шага минимизации (шаговый множитель). Учитывая геометрический смысл задачи минимизации функций двух переменных $\Phi(\overline{X})$ – «спуск на дно» поверхности $z = \Phi(\overline{X})$, итерационный метод (4.16) можно назвать методом спуска, $\Phi(X^{(k)})$ при каждом *k* является направлением спуска, и если множитель λ_k подбирается так, чтобы выполнялось условие релаксации $\Phi(X^{(k+1)}) < \Phi(X^{(k)})$, означающее переход на каждой итерации в точку с меньшим значением минимизируемой функции.

Таким образом, при построении численного метода вида (4.16) минимизации функции $\Phi(\overline{X})$ следует ответить на два главных вопроса: как выбирать направление спуска $\Phi(X^{(k)})$ и как регулировать длину шага в выбранном направлении с помощью скалярного параметра – шагового множителя λ_k.

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

$$\Phi_x\left(X^{(k)}\right) = -\operatorname{grad}\Phi\left(x^{(k)}\right) = -\left(\frac{\partial\Phi\left(X^{(k)}\right)}{\partial x_1} \quad \frac{\partial\Phi\left(X^{(k)}\right)}{\partial x_2} \ \ldots \ \frac{\partial\Phi\left(X^{(k)}\right)}{\partial x_n}\right)^T$$

– антиградиент функции $\Phi(\bar{x})$. Таким образом, из семейства методов (4.16) выделяем *градиентный метод*

$$X^{(k+1)} = X^{(k)} + \lambda_k \cdot \Phi_x\left(X^{(k)}\right).$$

Оптимальный шаг в направлении антиградиента – это такой шаг, при котором значение $\Phi\left(X^{(k+1)}\right)$ наименьшее среди всех других значений $\Phi(\overline{X})$ в этом фиксированном направлении, т.е. λ_k – находится из условия минимума функции:

$$\psi_k(\lambda_k) = \Phi\left(X^{(k)} - \lambda_k \cdot \Phi_x\left(X^{(k)}\right)\right).$$

Рассмотрим пример функции двух переменных. Разложим $\Phi\left(x^{(k)} - \lambda\Phi'_x, y^{(k)} - \lambda\Phi'_y\right)$ в ряд Тейлора в окрестности точки $\left(x^{(k)}, y^{(k)}\right)$:

$$\Phi\left(x^{(k)} - \lambda\,\Phi'_x, y^{(k)} - \lambda\,\Phi'_y\right) = \Phi\left(x^{(k)}, y^{(k)}\right) - \lambda\,\Phi'_x\Phi'_x - \lambda\,\Phi'_y\Phi'_y +$$
$$+\frac{1}{2}\left[(\lambda\,\Phi'_x)^2\,\Phi''_{xx} + 2\,\lambda\,\Phi'_x\lambda\,\Phi'_y\,\Phi''_{xy} + (\lambda\,\Phi'_y)^2\,\Phi''_{yy}\right]$$

Тогда

$$\frac{\partial\Phi}{\partial\lambda} = -\left((\Phi'_x)^2 + (\Phi'_y)^2\right) + \lambda\left[(\Phi'_x)^2\,\Phi''_{xx} + 2\,\Phi'_x\,\Phi'_y\,\Phi''_{xy} + (\Phi'_y)^2\,\Phi''_{yy}\right],$$

учитывая условие минимума:

$$\frac{\partial\Phi}{\partial\lambda} = 0,$$

имеем

$$\lambda = \frac{(\Phi'_x)^2 + (\Phi'_y)^2}{(\Phi'_x)^2\,\Phi''_{xx} + 2\,\Phi'_x\,\Phi'_y\,\Phi''_{xy} + (\Phi'_y)^2\,\Phi''_{yy}}.$$

Рисунок 4.1 иллюстрирует алгоритм нахождение решения системы уравнений методом наискорейшего спуска.

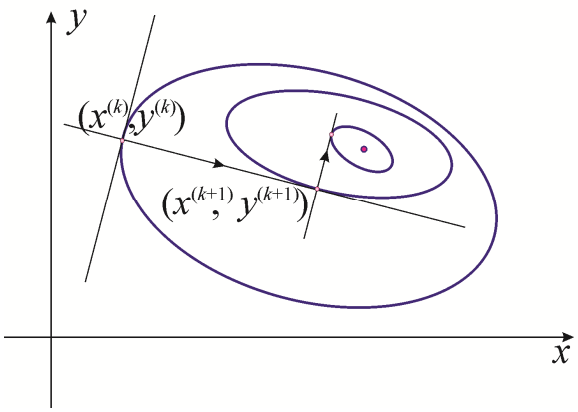


Рис. 4.1. Схематическое изображение траектории наискорейшего спуска для системы вида (4.13).

Главное достоинство градиентных методов решения нелинейных систем – глобальная сходимость. 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 уравнений. Однако этот метод обладает и недостатком – медленной сходимостью. Поэтому на практике часто используют построение *гибридных алгоритмов*, которые начинали бы поиск решения искомого решения нелинейной системы глобально сходящимся градиентным методом, а затем производили бы уточнение каким-либо быстросходящимся методом, например, методом Ньютона.

4.5. Численные примеры. Реализация в пакете Matlab

Пример 1. Привести систему линейных уравнений к виду, пригодному для метода простой итерации:

$$\begin{cases} y - 2x^2 = 0 \\ y - 2x^3 = 0.25 \end{cases}.$$

Используем представление приведенной системы с использованием (4.4). Для этого вычислим коэффициенты $\lambda_{11}, \lambda_{12}, \lambda_{21}, \lambda_{22}$. Вычислим производные, установив с помощью графической локализации корня начальное приближение $x^{(0)} = 1, y^{(0)} = 1.5$: $\left.\frac{\partial f_1}{\partial x}\right|_{X^{(0)}} = -4x\big|_{X^{(0)}} = -4$, $\left.\frac{\partial f_1}{\partial y}\right|_{X^{(0)}} = 1$, $\left.\frac{\partial f_2}{\partial x}\right|_{X^{(0)}} = -6x^2\big|_{X^{(0)}} = -6$,

$$\left.\frac{\partial f_2}{\partial y}\right|_{X^{(0)}} = 1.$$

Решая СЛАУ вида (4.5):

$$\begin{cases} 1-4\lambda_{11}-6\lambda_{12}=0 \\ \lambda_{11}+\lambda_{12}=0 \end{cases} \quad \begin{cases} -4\lambda_{21}-6\lambda_{22}=0 \\ 1+\lambda_{21}+\lambda_{22}=0 \end{cases},$$

Приходим к следующей формуле, пригодной для запуска итерационного процесса:

$$\begin{cases} x^{(k+1)} = x^{(k)} - 0.5\left(y^{(k)} - 2\left(x^{(k)}\right)^2\right) + 0.5\left(y^{(k)} - 2\left(x^{(k)}\right)^3 - 0.25\right), \\ y^{(k+1)} = y^{(k)} - 3\left(y^{(k)} - 2\left(x^{(k)}\right)^2\right) + 2\left(y^{(k)} - 2\left(x^{(k)}\right)^3 - 0.25\right). \end{cases}$$

Пример 2. Решить систему нелинейных уравнений

$$\begin{cases} y - 0.5x^2 = 0.2 \\ y - 0.5\cos x = 3 \end{cases} \qquad (4.17)$$

методом скорейшего спуска.

Для системы (4.17) запишем явный вид функции (4.15)

$$\Phi(x,y) = \left(y - 0.5x^2 - 0.2\right)^2 + \left(y - 0.5\cos x - 3\right)^2. \qquad (4.18)$$

Проведем программную реализацию алгоритма поиска экстремума функции (4.18) в пакете Matlab.

1. Построим график функции (4.18):

```
[X,Y]=meshgrid(0:0.1:3,0:0.1:3);
P=(Y-0.5*X.^2-0.2).^2+(Y-0.5*cos(X)-3).^2;
surf(X,Y,P)
```

2. Создадим файл, содержащий описание пользовательской функции **Phi**, возвращающей значение (4.18) в узлах координатной сетки.

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

```
            z(i,j)=(y(j)-0.5*x(i).^2-0.2).^2+(y(j)-0.5*cos(x(i))-
3).^2;
        end
    end
```

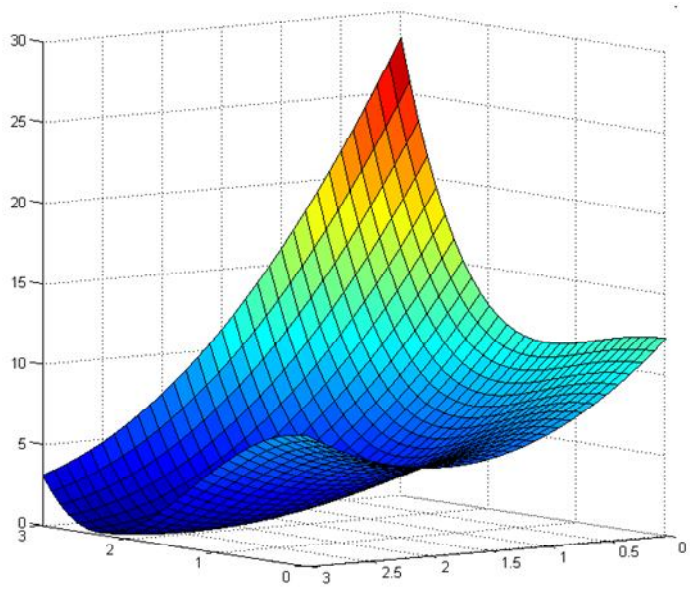


Рис. 4.2. График функции (4.17).

3. Создадим файлы, содержащие описание функций, возвращающих значения частных производных. Функция Phi_dx для вычисления частной производной по переменной *x*:

```
    function z=Phi_dx(x,y)
    N=length(x);
    z=zeros(N);
    for i=1:N
        for j=1:N
            z(i,j)=2*(y(j)-0.5*x(i).^2-0.2)*(-x(i))+2*(y(j)-
0.5*cos(x(i))-3)*(0.5*sin(x(i)));
        end
    end
```

Функция Phi_dy для вычисления частной производной по переменной *y*:

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

```
    z=zeros(N);
    for i=1:N
        for j=1:N
            z(i,j)=2*(y(j)-0.5*x(i).^2-0.2)+2*(y(j)-
0.5*cos(x(i))-3);
        end
    end
```

4. Файл L_Grad.m содержит описание функции, которая возвращает значение длины градиента функции (4.18).

```
    function z=Phi_grad(x,y)
    N=length(x);
    z=zeros(N);
    for i=1:N
        for j=1:N
            z(i,j)=Phi_dx(x(i),y(j)).^2+Phi_dy(x(i),y(j)).^2;
            z(i,j)=sqrt(z(i,j));
        end
    end
```

5. Файлы Unit_gradx.m и Unit_grady.m содержат функции, возвращающие координаты нормированного единичного вектора, сонаправленного с вектором, противоположным направлению вектора градиента:

```
    function z=Unit_gradx(x,y)
    N=length(x);
    z=zeros(N);
    for i=1:N
        for j=1:N
            z(i,j)=-Phi_dx(x(i),y(j))./Phi_grad(x(i),y(j));
        end
    end

    function z=Unit_grady(x,y)
    N=length(x);
    z=zeros(N);
    for i=1:N
```

```
        for j=1:N
            z(i,j)=-Phi_dy(x(i),y(j))./Phi_grad(x(i),y(j));
        end
    end
```

6. Создадим файлы Phi_ddx.m, Phi_ddy.m, Phi_ddxy.m, которые содержат описание вторых частных производных функции $\Phi(x,y)$: $\frac{\partial^2\Phi(x,y)}{\partial x^2}$, $\frac{\partial^2\Phi(x,y)}{\partial x\partial y}$, $\frac{\partial^2\Phi(x,y)}{\partial y^2}$ соответственно:

```
    function z=Phi_ddx(x,y)
    N=length(x);
    z=zeros(N);
    for i=1:N
        for j=1:N
            z(i,j)=-2*y(j)+3*x(i).^2+0.4+y(j).*cos(x(i))-
0.5*cos(2*x(i))-3*cos(x(i));
        end
    end

    function z=Phi_ddy(x,y)
    N=length(x);
    z=zeros(N);
    for i=1:N
        for j=1:N
            z(i,j)=4;
        end
    end

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

```
            z(i,j)=-2*x(i)+sin(x(i));
        end
    end
```

7. Функция Lambda.m возвращает значения коэффициента λ из уравнения (4.16):

```
    function z=Lyamda(x,y)
    N=length(x);
    z=zeros(N);
    for i=1:N
        for j=1:N
            a1=Phi_dx(x(i),y(j)).*Unit_gradx(x(i),y(j));
            a2=Phi_dy(x(i),y(j)).*Unit_grady(x(i),y(j));
            b1=Phi_ddx(x(i),y(j)).*Unit_gradx(x(i),y(j)).^2;
    b2=2*Phi_ddxy(x(i),y(j)).*Unit_gradx(x(i),y(j)).*Unit_grady(x
(i),y(j));
            b3=Phi_ddy(x(i),y(j)).*Unit_grady(x(i),y(j)).^2;
            z(i,j)=-(a1+a2)./(b1 +b2+b3);
        end
    end
```

8. Создадим файл GradMethod.m, содержащий описание функции, которая возвращает значения переменных x, y и соответствующее значение функции $\Phi(x, y)$ на каждом шаге итерационного процесса:

```
    function [X,Y,Fun]=GradMethod(x0,y0,Q)
    X(1)=x0;
    Y(1)=y0;
    Fun(1)=Phi(x0,y0)
    for i=2:Q
       X(i)=X(i-1)+Lyamda(X(i-1),Y(i-1)).*Unit_gradx(X(i-1),Y(i-
1));
```

```
        Y(i)=Y(i-1)+Lyamda(X(i-1),Y(i-1)).*Unit_grady(X(i-1),Y(i-
1));
        Fun(i)=Phi(X(i),Y(i));
    end
```

9. Для уточнения начального приближения проведем геометрическую локализацию решения:

```
x=0:0.001:3;
y1=0.5*x.^2+0.2;
y2=0.5*cos(x)+3;
plot(x,y1,x,y2);grid on
```

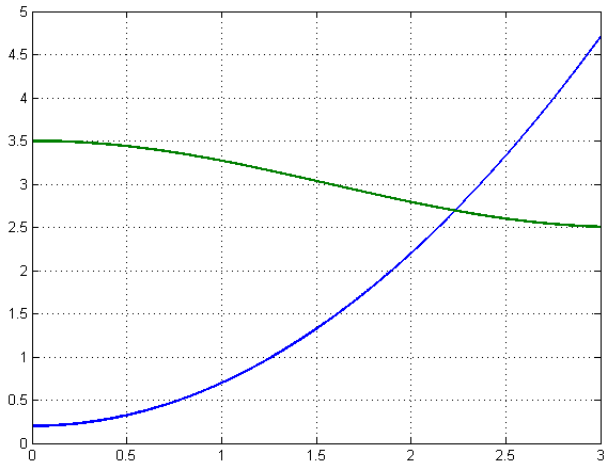


Рис.4.3. Геометрическая локализация
решения системы нелинейных уравнений (4.17)

Далее найдем решение исходной системы и визуализируем итерационный процесс:

```
x0=2;
y0=3;
Q=30;
[X,Y,Fun]=GradMethod(x0,y0,Q);
figure(2)
plot(X); grid on; hold on
plot(Y); grid on; hold on
```

Результат решения системы нелинейных уравнений (4.17) методом скорейшего спуска приведен в следующем окне:

```
    ans =
```

```
    2.232782044126464
ans =
    2.692657828286775
```

Зависимость координат точки решения исходной системы уравнений от номера итерации представлена на графике:

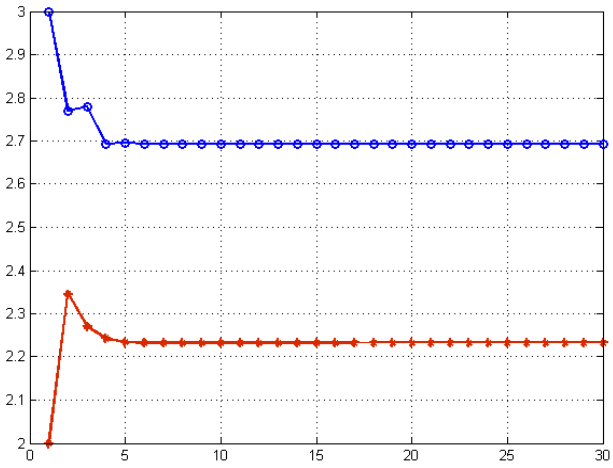


Рис. 4.4. Геометрическая интерпретация сходимости итерационного процесса.

Траектория итерационного процесса в пространстве представлена на рисунке:

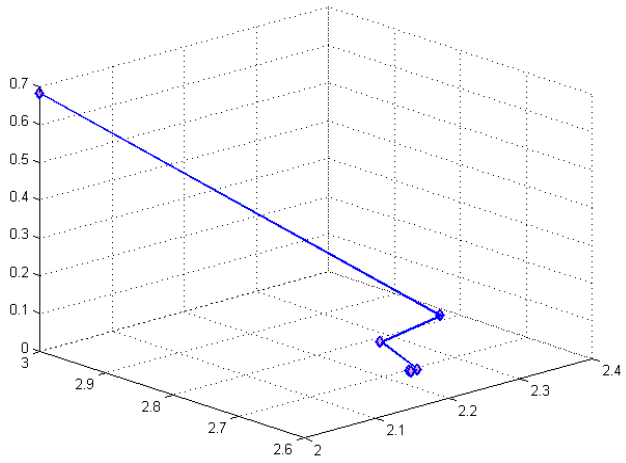


Рис.4.5. Траектория итерационного процесса в пространстве.

Для решения систем нелинейных уравнений средствами пакета Matlab применяется функция **`fsolve(),`** возвращающая вектор-столбец, компоненты которого являются корнями системы.

Пример 2. Решить систему уравнений (4.17) с помощью встроенной функции Matlab.

Файл NonlinSys.m содержит описание системы уравнений:

```
function z=NonlinSys(x)
z(1,1)=x(2)-0.5*x(1).^2-0.2;
z(2,1)=x(2)-0.5*cos(x(1))-3;
```

Найдем решение системы с помощью встроенного инструментария:

```
>> z(1,1)=2;
>> z(2,1)=3;
>> X=fsolve('NonlinSys',z,optimset('fsolve'))
Optimization terminated: first-order optimality is less than
options.TolFun.
X =
   2.232782044137628
   2.692657828269482
```

Определим по вектору невязки точность, с которой определяет решение встроенная функция пакета:

```
>> NonlinSys(X)
ans =
  1.0e-010 *
  -0.422198387362016
  -0.128892452266882
```

Для сравнения проведем расчет вектора невязки, полученного подстановкой решения, найденного метод скорейшего спуска:

```
W=[X(Q),Y(Q)];
NonlinSys(W)
```

Результат в окне команд:

```
ans =
  1.0e-015 *


   0.610622663543836
   0.444089209850063
```

Контрольные вопросы

1. Поясните геометрический смысл методов спуска.

2. Каким образом выбираются начальные приближения в рассмотренных методах?

3. Как выбор начального приближения влияет на сходимость метода Ньютона?

4. Почему рассмотренные методы являются итерационными?

5. Что произойдет, если в окрестности решения нелинейной системы (4.13) функция (4.15) будет иметь несколько минимумов?

Индивидуальные задания

1. Найти решение системы уравнений с точностью $\varepsilon = 10^{-3}$ указанными методами. Для определения начального приближения использовать графический способ локализации корней.

2. Найти решение системы нелинейных уравнений, используя возможности пакета Matlab.

3. Определить число итераций, оценить погрешности решения, сравнить результаты, полученные различными методами.

4. Оформить отчет по лабораторной работе.

Номер варианта	**Система уравнений**	**Методы**
1	$\begin{cases} y - \sin x = 0 \\ x^2 + y^2 = 4 \end{cases}$	1. Метод градиентного спуска. 2. Метод простых итераций.
2	$\begin{cases} y^2 + 4\exp x = 16 \\ x^2 + y^2 = 2 \end{cases}$	1. Метод покоординатных итераций. 2. Метод градиентного спуска.
3	$\begin{cases} y - \exp(0.5x) + 0.5 = 0 \\ x^2 + y^2 = 4 \end{cases}$	1. Метод градиентного спуска. 2. Метод Ньютона.
4	$\begin{cases} y - 1.5\sqrt{x} = 0 \\ x^2 + y^2 = 9 \end{cases}$	1. Метод градиентного спуска. 2. Модифицированный метод Ньютона.

5	$\begin{cases} y - 1.4\exp x + 0.25 = 0 \\ x^2 + y^2 = 9 \end{cases}$	1. Метод градиентного спуска. 2. Рекурсивный метод Ньютона.
6	$\begin{cases} y - 3\sqrt{x} = 1.8 \\ y - x^3 = 0.1 \end{cases}$	1. Метод градиентного спуска. 2. Разностный метод Ньютона.
7	$\begin{cases} y - 3\cos x = 0 \\ x^2 + y^2 = 6 \end{cases}$	1. Метод простых итераций. 2. Метод Брауна.
8	$\begin{cases} y - 2x^2 = 0 \\ y - 2x^3 = 0.25 \end{cases}$	1. Метод Ньютона. 2. Метод Брауна.
9	$\begin{cases} y - 0.5\sqrt{x} = 2.3 \\ x^2 + y^2 = 9 \end{cases}$	1. Метод покоординатных итераций. 2. Метод Брауна.
10	$\begin{cases} y - 0.25\cos x = 3 \\ y - 0.5x^3 = 0.2 \end{cases}$	1. Разностный метод Ньютона. 2. Метод покоординатных итераций.
11	$\begin{cases} y - \exp(0.2x) + 0.7 = 0 \\ x^2 + y^2 = 4 \end{cases}$	1. Метод простых итераций. 2. Метод Брауна.
12	$\begin{cases} y - 3\sqrt{x} = 1.8 \\ y - x^3 = 0.1 \end{cases}$	1. Метод Ньютона. 2. Метод градиентного спуска.
13	$\begin{cases} y - 0.8\exp x + 0.4 = 0 \\ x^2 + y^2 = 16 \end{cases}$	1. Метод градиентного спуска. 2. Метод Брауна.
14	$\begin{cases} y - 0.5\cos x = 1.5 \\ y - 0.25x^3 = 0.28 \end{cases}$	1. Модифицированный метод Ньютона. 2. Метод покоординатных итераций.
15	$\begin{cases} y - 0.3\sqrt{x} = 0.2 \\ x^2 + y^2 = 1 \end{cases}$	1. Метод градиентного спуска. 2. Метод Ньютона.
16	$\begin{cases} y - \sin x = 2 \\ y - 0.25x^3 = 0.31 \end{cases}$	1. Метод Ньютона. 2. Метод Брауна.
17	$\begin{cases} y - 3\sqrt{x} = 1.8 \\ x^2 + y^2 = 25 \end{cases}$	1. Метод покоординатных итераций. 2. Метод Ньютона.

18	$\begin{cases} y - \cos x = 0 \\ y - 0.2x^3 = 0.25 \end{cases}$	1. Модифицированный метод Ньютона. 2. Метод Брауна.
19	$\begin{cases} 4\exp x + y^2 = 8 \\ x^2 + y^2 = 1 \end{cases}$	1. Метод градиентного спуска. 2. Метод простых итераций.
20	$\begin{cases} y - \sin^2 x = 0 \\ x^2 + y^2 = 1 \end{cases}$	1. Метод градиентного спуска. 2. Метод Ньютона.

5 ИНТЕРПОЛИРОВАНИЕ ФУНКЦИЙ

5.1 Постановка задачи интерполирования функций

В основе многих численных методов лежит замена одной функции $f(x)$ другой функцией $\varphi(x)$, близкой к $f(x)$ и обладающей рядом свойств, позволяющими легко производить над ней аналитические или вычислительные операции. Такая замена называется *аппроксимацией* или *приближением функции* $f(x)$ функцией $\varphi(x)$. Задача аппроксимации функции возникает, в частности, если используется табличный способ задания функции. Наиболее распространенным способом решения задачи аппроксимации основан на использовании многочленов. Наряду с многочленами применяют ряды Фурье, экспоненциальные, логарифмические и другие элементарные функции. Для оценки «близости» функций выбирают тот или иной критерий согласия. Эти критерии основаны на способах измерения расстояния между функциями.

Геометрический смысл процедуры интерполирования функции состоит в том, чтобы найти кривую $y = \varphi(x)$ некоторого определенного типа, проходящую через заданную систему точек (x_i, y_i), $i = \overline{0,n}$.

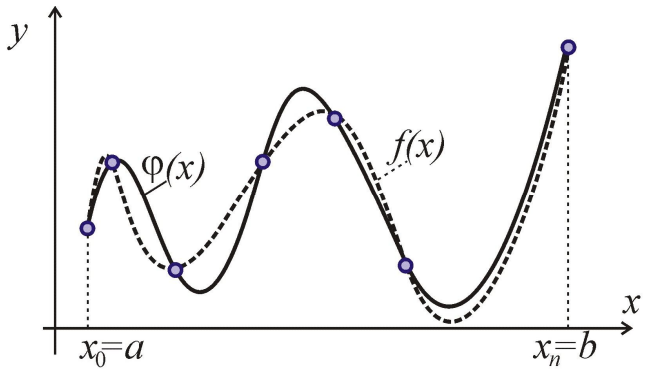


Рис. 5.1. Геометрическая интерпретация интерполирования функции.

Пусть функция $f(x)$ задана множеством своих значений для дискретного набора точек:

x_0	x_1	…	x_n
f_0	f_1	…	f_n

Здесь $f_i = f(x_i)$. Задача интерполяции имеет следующую формулировку: требуется найти интерполяционный многочлен $P(x) = P_n(x)$ степени не выше *n*, значения которого в узлах интерполяции x_i совпадают со значениями данной функции $P(x_i) = f_i$.

5.2 Интерполяционный полином Лагранжа

Для функции $f(x)$, заданной таблицей, построим интерполяционный многочлен $L_n(x)$, степень которого не выше *n*, в следующем виде:

$$L_n(x) = l_0(x) + l_1(x) + \ldots + l_n(x),$$

где $l_i(x)$ – многочлен степени *n*, причем

$$l_i(x_k) = \begin{cases} f_i, \text{ если } i = k, \\ 0, \text{ если } i \neq k. \end{cases} \qquad (5.1)$$

Многочлены $l_i(x)$ составим следующим образом

$$l_i(x) = c_i(x - x_0)(x - x_1)\cdot\ldots\cdot(x - x_{i-1})(x - x_{i+1})\cdot\ldots\cdot(x - x_n), \qquad (5.2)$$

где c_i – постоянный коэффициент, значение которого находится из первой части условия (5.1)

$$c_i = \frac{f_i}{(x_i - x_0)(x_i - x_1)\cdot\ldots\cdot(x_i - x_{i-1})(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}.$$

Тогда *интерполяционный многочлен Лагранжа* имеет вид

$$L_n(x) = \sum_{i=1}^{n} f_i \frac{(x - x_0)(x - x_1)\cdot\ldots\cdot(x - x_{i-1})(x - x_{i+1})\cdot\ldots\cdot(x - x_n)}{(x_i - x_0)(x_i - x_1)\cdot\ldots\cdot(x_i - x_{i-1})(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}. \qquad (5.3)$$

5.3 Интерполяционные формулы Ньютона

Формулы Ньютона предназначены для таблиц с равноотстоящими узлами. Будем считать, интерполируемая функция *f*(*x*) задана своими значениями $f_0, f_1, \ldots f_n$ а системе *равноотстоящих узлов* $x_0, x_1, \ldots x_n$ т.е. таких, что любой узел x_i, этой *сетки* можно представить в виде:

$$x_i = x_0 + ih,$$

где $i = \overline{0,n}$, $h>0$ – некоторая постоянная величина, называемая *шагом сетки* (таблицы).

Введем понятие конечной разности, для этого определим разности между значениями функции в узлах интерполяции. Конечная разность первого порядка имеет вид: $\Delta f_i = f_{i+1} - f_i$.

Конечная разность второго порядка задается выражением:

$$\Delta^2 f_i = \Delta f_{i+1} - \Delta f_i = f_{i+2} - 2f_{i+1} + f_i.$$

Конечная разность *n*-го порядка вычисляется по формуле:

$$\Delta^k f_i = f_{i+k} - k \cdot f_{i+k-1} + \frac{k(k-1)}{2!} f_{i+k-2} - \ldots + (-1)^k f_i.$$

Для функции, заданной таблицей своих значений, конечные разности разных порядков удобно помещать в одну общую таблицу с узлами и значениями функции (последние можно интерпретировать как конечные разности нулевого порядка). Эту общую таблицу называют *таблицей конечных разностей.*

Пример 1. Составить таблицу конечных разностей для табулированной функции $f(x)$:

x_i	0.1	0.2	0.3	0.4	0.5
f_i	8.52	6.32	5.14	2.31	1.05

Используя общую концепцию построения конечных разностей, запишем итоговую таблицу:

x_i	f_i	Δf_i	$\Delta^2 f_i$	$\Delta^3 f_i$	$\Delta^4 f_i$
0.1	8.52	- 2.2	1.02	- 2.67	5.89
0.2	6.32	- 1.18	- 1.65	3.22	
0.3	5.14	- 2.83	1.57		
0.4	2.31	- 1.26			
0.5	1.05				

Первая интерполяционная формула Ньютона используется для интерполирования и экстраполирования в точках *x*, близких к началу таблицы – узлу x_0, и имеет следующий вид:

$$P_n(x) = f_0 + q \cdot \Delta f_0 + \frac{q(q-1)}{2!}\Delta^2 f_0 + \ldots + \frac{q(q-1)\cdot\ldots\cdot(q-n+1)}{n!}\Delta^n f_0, \qquad (5.4)$$

где $q = \frac{x - x_0}{h}$, число *n* выбирают так, чтобы конечные разности *n*-го порядка были практически постоянными.

Когда значение аргумент находится ближе к концу отрезка интерполяции x_n, используется *вторая интерполяционная формула Ньютона*:

$$P_n(x) = f_n + q \cdot \Delta f_{n-1} + \frac{q(q+1)}{2!}\Delta^2 f_{n-2} + \ldots + \frac{q(q+1)\cdot\ldots\cdot(q+n-1)}{n!}\Delta^n f_0, \quad (5.5)$$

где $q = \frac{x - x_n}{h}$.

5.4 Центральные интерполяционные формулы

Наряду с выведенными специально для начала и конца таблицы первой и второй интерполяционными формулами Ньютона, имеется еще несколько формул, рассчитанных на их применение в центральной части таблицы и потому называемых *центральными интерполяционными формулами.* Прежде, чем определять эти формулы, введем понятие центральных разностей.

Будем считать, что узел x_0 расположен в середине таблицы, и нумерация остальных узлов производится, начиная с x_0 с использованием как положительных, так и отрицательных индексов, т.е. считаем $x_i = x_0 + ih$, где $i = 0,\pm1,\pm2,\ldots$.

Первая интерполяционная формула Гаусса, предназначенная для интерполирования в середине таблицы представима в виде:

$$f(x) \approx \overline{P}(x_0 + qh) = y_0 + q \cdot \Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_{-1} + \frac{(q+1)q(q-1)}{3!}\Delta^3 y_{-1} + \\ + \frac{(q+1)q(q-1)(q-2)}{4!}\Delta^4 y_{-2} + \ldots \qquad (5.6)$$

где $q = \dfrac{x - x_0}{h}$.

Записанные слагаемые легко дополнить следующими, если знать, что в этой формуле используются нижние центральные разности все возрастающих порядков.

Вторая интерполяционная формула Гаусса, использующая верхние центральные разности имеет вид:

$$f(x) \approx \overline{P}(x_0 + qh) = y_0 + q \cdot \Delta y_{-1} + \frac{q(q+1)}{2!}\Delta^2 y_{-1} + \frac{(q+1)q(q-1)}{3!}\Delta^3 y_{-2} + \\ + \frac{(q-1)q(q+1)(q+2)}{4!}\Delta^4 y_{-2} + \ldots \tag{5.7}$$

Интерполяционные формулы Гаусса служат полуфабрикатами для получения более симметричных, использующих все центральные разности интерполяционных формул.

Так, полусумма первого и второго интерполяционных многочленов Гаусса после преобразований приводит к формуле

$$f(x) \approx P_S(x_0 + qh) = y_0 + q\frac{\cdot \Delta y_{-1} + \cdot \Delta y_0}{2} + \frac{q^2}{2!}\Delta^2 y_{-1} + \\ + \frac{q(q^2-1)}{3!}\frac{\Delta^3 y_{-1} + \Delta^3 y_{-2}}{2} + \frac{q(q^2-1)}{4!}\Delta^4 y_{-2} + \ldots, \tag{5.8}$$

называемой *интерполяционной формулой Стирлинга.*

Если же взять полусумму второго интерполяционного многочлена Гаусса и такого же многочлена, но с нижними индексами, увеличенными на единицу (т.е. с базовой точкой x_1 вместо x_0), можно получить *интерполяционую формулу Бесселя*:

$$f(x) \approx P_B(x_0 + qh) = \frac{y_0 + y_1}{2} + \left(q - \frac{1}{2}\right)\Delta y_0 + \frac{q(q-1)}{2!}\frac{\Delta^2 y_{-1} + \Delta^2 y_0}{2} + \\ + \frac{q(q-1)\left(q - \frac{1}{2}\right)}{3!}\Delta^3 y_{-1} + \frac{q(q-1)(q+1)(q-2)}{4!}\frac{\Delta^4 y_{-2} + \Delta^4 y_{-1}}{2} + \ldots \tag{5.9}$$

5.5 Сплайн-интерполяция

В тех случаях, когда промежуток, на котором нужно подменить функцию $f(x)$ функцией $\varphi(x)$, велик, и отсутствуют основания считать данную функцию $f(x)$ достаточно гладкой, нет смысла пытаться повышать качество ее полиномиальной аппроксимации за счет использования в роли $\varphi(x)$ многочленов высоких степеней. Более перспективным в этих условиях является применение *кусочно-полиномиальной аппроксимации*, предполагающей, что аппроксимирующая функция $\varphi(x)$ составляется из отдельных многочленов, как правило, одинаковой небольшой степени, определенных каждый на своей части отрезка $[a,b]$.

Использование низких степеней многочленов, составляющих $\varphi(x)$, позволяет легко находить их коэффициенты как из интерполяционных, так и из иных условий. Так, если заданы значения y_i функции $y = f(x)$ на системе узлов x_i таких, что $a \le x_0 < x_1 < ... < x_n \le b$ и требуется аппроксимировать $f(x)$ *кусочно-линейной функцией* $\varphi(x)$, исходя из условий интерполяции $\varphi(x_i) = y_i, i = 0,1,...n$, то, выбирая функцию $\varphi(x)$ в виде

$$\varphi(x) = \begin{cases} a_1 x + b_1 \quad при \quad x \in [x_0, x_1], \\ a_2 x + b_2 \quad при \quad x \in [x_1, x_2], \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_n x + b_n \quad при \quad x \in [x_{n-1}, x_n] \end{cases}$$

для нахождения *n* пар ее коэффициентов a_k, b_k имеем систему из 2*n* линейных уравнений:

$$\begin{cases} \begin{cases} a_1 x_0 + b_1 = y_0 \\ a_1 x_1 + b_1 = y_1 \end{cases}, \\ \begin{cases} a_2 x_1 + b_2 = y_1 \\ a_2 x_2 + b_2 = y_2 \end{cases}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \begin{cases} a_n x_{n-1} + b_{n1} = y_{n-1} \\ a_n x_n + b_n = y_n \end{cases} \end{cases} \qquad (5.10)$$

Причем, каждая пара соседних уравнений системы (5.10), имеющих коэффициенты с одинаковыми индексами, не связана с остальными и может решаться отдельно.

Аналогично, каждое звено *кусочно-квадратичной функции* (при $n = 2m$ в) определяется тройкой коэффициентов a_k, b_k, c_k которые могут быть найдены последовательным решением (при $k = \overline{1,m}$) трехмерных линейных систем, соответствующим выставленным интерполяционным условиям. Фактически, в рассмотренных случаях речь идет о последовательной линейной интерполяции по перемещаемым вдоль отрезка парам соседних точек разбиения и о последовательной квадратичной интерполяции по тройкам таких точек.

При большом количестве узлов интерполяции приходится использовать полиномы высокой степени. Можно этого избежать, разбив отрезок интерполяции на несколько частей и построив на каждой части свой интерполяционный многочлен. Существенный недостаток такого интерполирования состоит в том, что в точках сшивки разных интерполяционных полиномов их первая производная будет разрывной, поэтому для решения задачи кусочно-линейной интерполяции используют особый вид кусочно-полиномиальной интерполяции – сплайн-интерполяцию.

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

Пусть на отрезке $[a,b]$ задана упорядоченная система несовпадающих точек x_k, $k = \overline{0,n}$.

Определение. *Сплайном* $S_m(x)$ называется определенная на $[a,b]$ функция, принадлежащая классу $C^l_{[a,b]}$ l раз непрерывно дифференцируемых функций, такая, что на каждом промежутке $[x_{k-1}, x_k]$, $k = \overline{2,n}$ – это многочлен n-й степени. Разность $d = m - l$ между степенью сплайна m и показателем его гладкости l называется *дефектом* сплайна.

Если сплайн $S_m(x)$ строится по некоторой функции $f(x)$ так, чтобы выполнялись условия $S_m(x_i) = f(x_i)$, то такой сплайн называется *интерполяционным сплайном* для функции $f(x)$ при этом *узлы сплайна* x_k, вообще говоря, могут не совпадать с узлами интерполяции x_i.

Тривиальные примеры интерполяционных сплайнов: кусочно-линейная функция $\varphi(x)$, определенная с параметрами a_k, b_k очевидно, является интерполяционным сплайном степени 1 дефекта 1, а кусочно-квадратичная функция есть интерполяционный сплайн степени 2 дефекта 2.

Совпадение дефекта сплайна с его степенью обеспечивает просто непрерывность сплайна. Интерес представляет построение сплайнов с большей гладкостью, т. е. с малым дефектом. Такие сплайны являют собой дальнейшее совершенствование идеи кусочно-полиномиальной аппроксимации.

Наиболее известным и широко применяемым интерполяционным сплайном является сплайн степени 3 дефекта 1. Положим, что узлы сплайна $a = x_0, x_1, ...x_n = b$ одновременно служат узлами интерполяции, т.е. в них известны значения функции $f_k = f(x_k),\ \ k = 0,1,...n$.

Определение. Кубическим сплайном дефекта 1, интерполирующим на отрезке $[a,b]$ данную функцию $f(x)$, называется функция

$$g(x) := \left\{ g_k(x) = a_k + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3, x \in [x_{k-1}, x_k] \right\}_{k=1}^{n}, \qquad (5.11)$$

удовлетворяющая совокупности условий: *а*) $g(x_k) = f_k$ – условие интерполяции в узлах сплайна; *б*) $g(x) \in C^2_{[a,b]}$ – двойная непрерывная дифференцируемость; *в*) $g''(a) = g''(b) = 0$ краевые условия.

Определенный таким образом сплайн называют еще *естественным* или *чертежным сплайном*. Несложно убедиться, что определяемая условиями *а*) – *в*) функция (11), представляющая собой кубический *n*-звенник с гладким сопряжением звеньев.

Замечание. Краевые условия в определении могут быть заменены на другие. Например, можно наложить дополнительные условия на первую производную функ-

ции $g(x)$ в точках a и b. В таком случае кубический сплайн, оставаясь интерполяционным дефекта 1, утрачивает свойство быть естественным. Для построения по данной функции $f(x)$ интерполирующего ее сплайна нужно найти $4n$ его коэффициентов a_k, b_k, c_k, d_k ($k = 1, 2, ..., n$).

5.6 Численные примеры. Реализация в пакете Matlab

Пример 2. Решить задачу интерполяции с помощью многочлена Лагранжа для функции $f(x)$, заданной таблично в примере 1:

x_i	0.1	0.2	0.3	0.4	0.5
f_i	8.52	6.32	5.14	2.31	1.05

1. Создадим функцию bas_fun.m, возвращающую значение многочлена $l_i(x)$:

```
function l=bas_fun(x,i,X,Y)
%x - абсцисса точек интерполяции
%i - номер узла
%X - вектор значений абсцисс узлов интерполирования
%Y - вектор значений ординат узлов интерполирования
N=length(X);
l=1;
for j=1:N
  if not(i==j)
      l=l*(x-X(j))./(X(i)-X(j));
  end
end
l=l*Y(i);
```

2. Создадим файл-функцию Lagr.m, возвращающий значение полинома Лагранжа:

```
function L=Lagr(x,X,Y)
N=length(X);
L=0;
for i=1:N
  L=L+bas_fun(x,i,X,Y);
```

```
end
```

3. В скрипт-файле зададим табулированную функцию и вычислим значения полинома Лагранжа:

```
x=[0.1 0.2 0.3 0.4 0.5];
y=[8.52 6.32 5.14 2.31 1.05];
M=length(x);
N=21; %задаем число точек интерполирования
h=(x(M)-x(1))./(N-1); % рассчитываем шаг интерполирования
%вычисляем координаты промежуточных точек
for j=1:N
    X(j)=x(1)+(j-1)*h;
end
% значение полинома Лагранжа в промежуточных точках
for j=1:N
    Yp(j)=Lagr(X(j),x,y);
end
plot(x,y,'ro',X,Yp,'b-'); grid on;
```

4. Значения табличной функции и интерполированные значения функции представлены на графике:

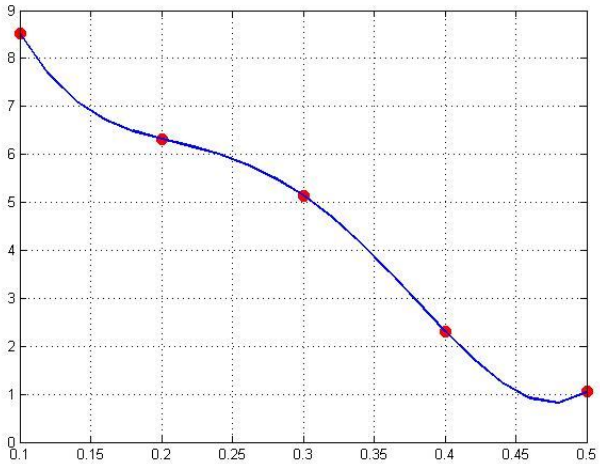


Рис. 5.1. Результат интерполирования функции с помощью полинома Лагранжа

Пример 3. Решить задачу интерполяции для функции, заданной таблично в примере 2, средствами пакета Matlab.

Для решения задачи одномерной интерполяции в пакете Matlab используется функция **`interp1()`**, которая реализует один из способов интерполирования: **`'nearest'`** – интерполяция по соседним элементам; **`'linear'`** – линейная интерполяция (применяется по умолчанию, если способ интерполирования не задан); **`'spline'`** – интерполяция кубическими сплайнами; **`'pchip'`** – интерполяция кубическими эрмитовыми сплайнами.

Для демонстрации использования функционала Matlab для решения задачи интерполирования продолжим запись предыдущего скрипт-файла:

```
Ylin=interp1(x,y,X,'linear');
Ysp=interp1(x,y,X,'spline');
Yne=interp1(x,y,X,'nearest');
Ypc=interp1(x,y,X,'pchip');
figure (2)
plot(x,y,'o',X,Ylin,X,Ysp,X,Yne,X,Ypc); grid on;
```

Исходные данные и результат всех видов интерполяции представлены на рисунке 5.2.

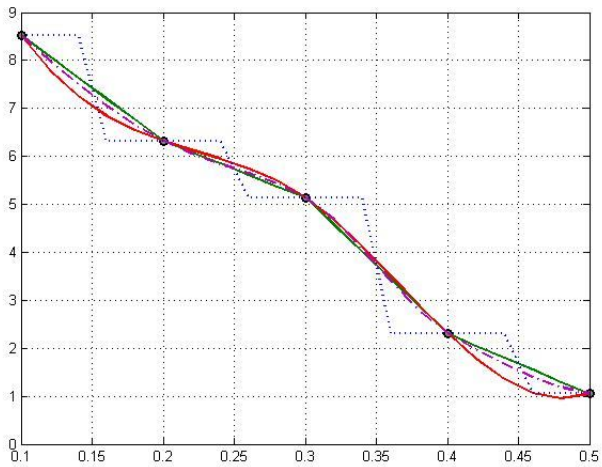


Рис. 5.2. Результат интерполирования функции
с помощью встроенной функции пакета Matlab **`interp1`**.

Альтернативным подходом может оказаться использование функции пакета Matlab **spline()**. Покажем решение задачи с помощью кубических сплайнов:

```
figure (3)
Yspline=spline(x,y,X);
plot(x,y,'o',X,Yspline); grid on;
```

Визуализация исходных данных и результатов кубической сплайн-интерполяции показана на графике.

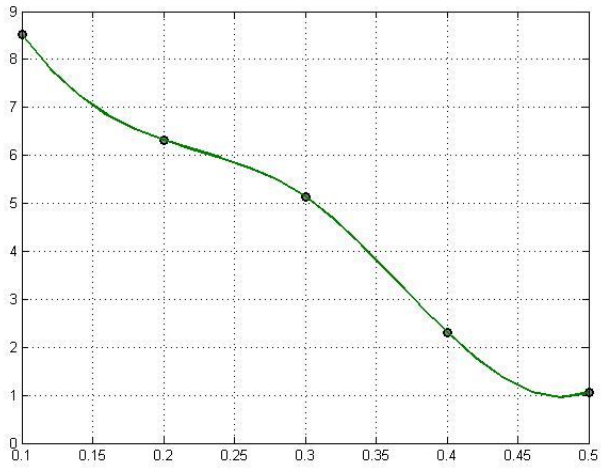


Рис. 5.2. Результат интерполирования функции
с помощью встроенной функции пакета Matlab **`spline`**.

Контрольные вопросы

1. В каких практических случаях может потребоваться аппроксимация функции?

2. В какой форме строится интерполяционный многочлен Лагранжа?

3. Как используется при выводе формулы Лагранжа требование совпадения его значений со значениями исходной функции в узлах?

4. Какие точки называются узлами интерполяции? Какие узлы называются равноотстоящими?

5. При решении каких задач используются интерполяционные формулы Ньютона, Гаусса, Бесселя, Стирлинга?

6. В каких случаях применяется сплайн-интерполяция? Какой недостаток «кусочного» интерполирования с помощью многочленов Лагранжа или Ньютона устраняется при интерполяции сплайнами?

Индивидуальные задания

1. По заданной таблице значений функции составить формулу интерполяционного многочлена Лагранжа. Построить его график, отметить узловые точки.

2. Пользуясь указанными интерполяционными формулами вычислить значение функции для указанных значений аргумента.

3. Решить задачу интерполяции средствами пакета Matlab. Привести графическое представление результатов сплайн-интерполяции.

3. Оформить отчет по лабораторной работе.

Номер варианта	x	$f(x)$	**Значения аргумента**	**Методы интерполяции**
1, 11	1.50	0.51183	*1 вариант:* $x = 1.509$ *11 вариант*: $x = 1.504$	Первая интерполяционная формула Ньютона
	1.51	0.50624		
	1.52	0.50064		
	1.53	0.49503		
	1.54	0.48940		
	1.55	0.48376		
	1.56	0.47811	*1 вариант:* $x = 1.553$ *11 вариант:* $x = 1.545$	Интерполяционная формула Стирлинга
	1.57	0.47245		
	1.58	0.46678		
	1.59	0.46110		
	1.60	0.45540		
2, 12	0.00	0.28081	*2 вариант:* $x = 0.495$ *12 вариант*: $x = 0.471$	Вторая интерполяционная формула Ньютона
	0.05	0.31270		
	0.10	0.34549		
	0.15	0.37904		
	0.20	0.41318		
	0.25	0.44774		
	0.30	0.48255	*2 вариант:* $x = 0.249$ *12 вариант*: $x = 0.253$	Интерполяционная формула Бесселя
	0.35	0.51745		
	0.40	0.55226		
	0.45	0.58682		
	0.50	0.62096		
3, 13	1.0	0.5652	*3 вариант:* $x = 1.053$ *13 вариант*: $x = 1.091$	Первая интерполяционная формула Ньютона
	1.1	0.6375		
	1.2	0.7147		
	1.3	0.7973		
	1.4	0.8861		
	1.5	0.9817		

	1.6	1.0848	*3 вариант: x* = 1.450 *13 вариант*: *x* = 1.523	Первая интерполяционная формула Гаусса
	1.7	1.1964		
	1.8	1.3172		
	1.9	1.4482		
	2.0	1.5906		
4, 14	0.50	1.6487	*4 вариант: x* =0.594 *14 вариант*: *x* = 0.598	Вторая интерполяционная формула Ньютона
	0.51	1.6653		
	0.52	1.6820		
	0.53	1.6989		
	0.54	1.7160		
	0.55	1.7333		
	0.56	1.7507	*4 вариант: x* =0.548 *14 вариант*: *x* = 0.555	Интерполяционная формула Стирлинга
	0.57	1.7683		
	0.58	1.7860		
	0.59	1.8040		
	0.60	1.8221		
5, 15	0.10	3.63004	*5 вариант: x* =0.110 *15 вариант*: *x* = 0.254	Первая интерполяционная формула Ньютона
	0.35	3.75680		
	0.60	3.88933		
	0.85	4.03258		
	1.10	4.19310		
	1.35	4.38042		
	1.60	4.60963	*5 вариант: x* =1.456 *15 вариант*: *x* = 1.554	Вторая интерполяционная формула Гаусса
	1.85	4.90697		
	2.10	5.32331		
	2.35	5.97322		
	2.60	7.18210		
6, 16	0.50	1.6487	*6 вариант: x* =0.594 *16 вариант*: *x* = 0.598	Вторая интерполяционная формула Ньютона
	0.51	1.6653		
	0.52	1.6820		
	0.53	1.6989		
	0.54	1.7160		
	0.55	1.7333		
	0.56	1.7507	*6 вариант: x* =0.555 *16 вариант*: *x* = 0.562	Интерполяционная формула Бесселя
	0.57	1.7683		
	0.58	1.7860		
	0.59	1.8040		
	0.60	1.8221		
7, 17	1.1	0.89121	*7 вариант: x* =1.102 *17 вариант*: *x* = 1.186	Первая интерполяционная формула Ньютона
	1.2	0.93204		
	1.3	0.96356		
	1.4	0.98545		
	1.5	0.99749		

	1.6	0.99957		
	1.7	0.99166	*7 вариант: x* =1.655 *17 вариант*: *x* = 1.684	Интерполяционная формула Стирлинга
	1.8	0.97385		
	1.9	0.94630		
	2.0	0.90930		
	2.1	0.86321		
8, 18	0.10	3.63004	*8 вариант: x* =2.588 *18 вариант*: *x* = 2.486	Вторая интерполяционная формула Ньютона
	0.35	3.75680		
	0.60	3.88933		
	0.85	4.03258		
	1.10	4.19310		
	1.35	4.38042		
	1.60	4.60963	*8 вариант: x* =1.542 *18 вариант*: *x* = 1.455	Вторая интерполяционная формула Гаусса
	1.85	4.90697		
	2.10	5.32331		
	2.35	5.97322		
	2.60	7.18210		
9, 19	1.50	0.51183	*9 вариант: x* =1.501 *19 вариант*: *x* = 1.508	Первая интерполяционная формула Ньютона
	1.51	0.50624		
	1.52	0.50064		
	1.53	0.49503		
	1.54	0.48940		
	1.55	0.48376		
	1.56	0.47811	*9 вариант: x* =1.555 *19 вариант*: *x* = 1.562	Интерполяционная формула Бесселя
	1.57	0.47245		
	1.58	0.46678		
	1.59	0.46110		
	1.60	0.45540		
10, 20	0.00	0.28081	*10 вариант: x* =0.488 *20 вариант*: *x* = 0.465	Вторая интерполяционная формула Ньютона
	0.05	0.31270		
	0.10	0.34549		
	0.15	0.37904		
	0.20	0.41318		
	0.25	0.44774		
	0.30	0.48255	*10 вариант: x* =0.255 *20 вариант*: *x* = 0.302	Первая интерполяционная формула Гаусса
	0.35	0.51745		
	0.40	0.55226		
	0.45	0.58682		
	0.50	0.62096		

6 ОБРАБОТКА ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ

Пусть в результате измерений получена таблица некоторой зависимости $y = f(x)$:

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 состоит в построении интерполяционного многочлена, значения которого в узлах интерполяции x_i совпадают со значениями данной функции $f(x_i) = f_i$. Если значения функции $y = f(x)$ известны с некоторой погрешностью, то требование совпадения значений в узлах интерполяции не оправдано, поскольку оно не означает совпадение характеров исходной и интерполирующей функции. Поэтому поставим задачу следующим образом – найти функцию вида $\varphi(x)$, которая в точках $x_1, x_2, \ldots, x_n$ принимает значения, близкие к табличным значениям $y_1, y_2, \ldots, y_n$.

Предположим, что приближающая функция $\varphi(x)$ в точках $x_1, x_2, \ldots, x_n$ имеет значения $\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_n$. Тогда нужно найти функцию $\varphi(x)$ определенного вида так, чтобы сумма квадратов отклонений значений табулированной функции y_i от аппроксимирующей $\bar{y}_i$

$$(y_1 - \bar{y}_1)^2 + (y_2 - \bar{y}_2)^2 + \ldots + (y_n - \bar{y}_n)^2 \qquad (6.1)$$

была наименьшей (рисунок 6.1).

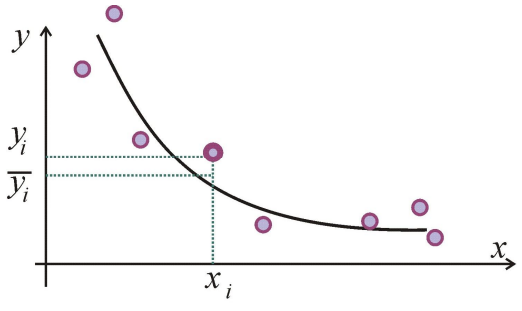


Рис. 6.1. Геометрическая интерпретация метода наименьших квадратов.

6.1 Нахождение приближающей функции в виде линейной

Пусть приближающая функция имеет вид

$$\varphi(x) = ax + b, \tag{6.2}$$

где a, b – параметры.

Составим сумму вида (6.1) для этого случая:

$$\Phi(a,b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2 \to \min . \tag{6.3}$$

Функция $\Phi(a,b)$ является неотрицательной квадратичной, поэтому в некоторой области она имеет единственную точку минимума (a^*, b^*), удовлетворяющую условиям: $\frac{\partial \Phi}{\partial a} = 0$, $\frac{\partial \Phi}{\partial b} = 0$, т.е. системе уравнений:

$$\begin{cases} \sum_{i=1}^{n} (y_i - ax_i - b)x_i = 0, \\ \sum_{i=1}^{n} (y_i - ax_i - b) = 0. \end{cases} \tag{6.4}$$

Запишем СЛАУ с неизвестными a и b в общепринятом виде:

$$\begin{cases} \left(\sum_{i=1}^{n} x_i^2\right) \cdot a + \left(\sum_{i=1}^{n} x_i\right) \cdot b = \sum_{i=1}^{n} x_i y_i , \\ \left(\sum_{i=1}^{n} x_i\right) \cdot a + n \cdot b = \sum_{i=1}^{n} y_i . \end{cases} \tag{6.5}$$

Вычислив значения параметров a, b, получаем конкретный вид функции (6.2).

В зависимости от характера табличных данных, изучаемого с помощью их изображения в соответствующей системе координат, при обработке экспериментальных данных часто используют иные семейства двухпараметрических функций. Существует возможность с помощью подходящего преобразования переменных получить линейную зависимость и использовать метод (6.5) для следующих семейств функций, показанных в таблице 6.1.

Таблица 6.1 – Замена переменных при линеаризации представителей семейства двухпараметрических функций

Функция $y = f(x)$	Линеаризованная форма	Замена переменных и постоянных для $Y = AX + B$
$y = \frac{a}{x} + b$	$y = a \cdot \frac{1}{x} + b$	$X = \frac{1}{x},\ Y = y,\ A = a,\ B = b$
$y = \frac{d}{x + c}$	$y = -\frac{1}{c} \cdot xy + \frac{d}{c}$	$X = xy,\ Y = y,$ $A = -\frac{1}{c},\quad B = \frac{d}{c}$
$y = \frac{1}{ax + b}$	$\frac{1}{y} = ax + b$	$X = x,\ Y = \frac{1}{y}, A = a,\ B = b$
$y = \frac{x}{ax + b}$	$\frac{1}{y} = a + b\frac{1}{x}$	$X = \frac{1}{x},\ Y = \frac{1}{y}, A = a,\ B = b$
$y = a \cdot \ln(x) + b$	$y = a \cdot \ln(x) + b$	$X = \ln(x),\ Y = y,\ A = a,\ B = b$
$y = c \cdot e^{ax}$	$\ln(y) = ax + \ln(c)$	$X = x,\ Y = \ln(y), A = a,\ B = \ln(c)$
$y = c \cdot x^a$	$\ln(y) = a \cdot \ln(x) + \ln(c)$	$X = \ln(x), Y = \ln(y), A = a, B = \ln(c)$
$y = \frac{1}{(ax + b)^2}$	$\frac{1}{\sqrt{y}} = ax + b$	$X = x,\ Y = \frac{1}{\sqrt{y}}, A = a,\ B = b$
$y = \frac{c \cdot x}{e^{dx}}$	$\ln\left(\frac{y}{x}\right) = -dx + \ln(c)$	$X = x, Y = \ln\left(\frac{y}{x}\right), A = -d, B = \ln(c)$
$y = \frac{1}{1 + c \cdot e^{ax}}$	$\ln\left(\frac{1}{y} - 1\right) = ax + \ln(c)$	$X = x,\ Y = \ln\left(\frac{1}{y} - 1\right), A = a,\ B = \ln(c)$

6.2 Нахождение приближающей функции в виде квадратного трехчлена

Приближающая функция имеет вид:

$$\varphi(x) = ax^2 + bx + c, \tag{6.6}$$

где a, b, c – параметры.

Составим сумму вида (6.1) как функцию $\Phi(a,b)$ для этого случая:

$$\Phi(a,b) = \sum_{i=1}^{n} \left(y_i - ax_i^2 - bx_i - c\right)^2 \to \min . \tag{6.7}$$

Функция $\Phi(a,b)$ является неотрицательной квадратичной, поэтому в некоторой области она имеет единственную точку минимума (a^*,b^*), удовлетворяющую условиям: $\frac{\partial\Phi}{\partial a}=0,\quad \frac{\partial\Phi}{\partial b}=0$, т.е. системе уравнений:

$$\begin{cases}\sum\limits_{i=1}^{n}\left(y_i-ax_i^2-bx_i-c\right)x_i^2=0,\\ \sum\limits_{i=1}^{n}\left(y_i-ax_i^2-bx_i-c\right)x_i=0,\\ \sum\limits_{i=1}^{n}\left(y_i-ax_i^2-bx_i-c\right)=0.\end{cases} \tag{6.8}$$

Запишем СЛАУ с неизвестными *a*, *b* и *c* в общепринятом виде:

$$\begin{cases}\left(\sum\limits_{i=1}^{n}x_i^4\right)\cdot a+\left(\sum\limits_{i=1}^{n}x_i^3\right)\cdot b+\left(\sum\limits_{i=1}^{n}x_i^2\right)\cdot c=\sum\limits_{i=1}^{n}x_i^2y_i,\\ \left(\sum\limits_{i=1}^{n}x_i^3\right)\cdot a+\left(\sum\limits_{i=1}^{n}x_i^2\right)\cdot b+\left(\sum\limits_{i=1}^{n}x_i\right)\cdot c=\sum\limits_{i=1}^{n}x_iy_i,\\ \left(\sum\limits_{i=1}^{n}x_i^2\right)\cdot a+\left(\sum\limits_{i=1}^{n}x_i\right)\cdot b+n\cdot c=\sum\limits_{i=1}^{n}y_i.\end{cases} \tag{6.9}$$

Решив систему относительно неизвестных *a*, *b*, *c*, находим значения параметров приближающей функции (6.6).

6.3 Численные примеры. Реализация в пакете Matlab

Пример 1. Даны табличные значения функциональной зависимости. Найти коэффициенты квадратичной аппроксимирующей функции (6.6).

x	0	1	2	3	4	5	6	7	8	9	10
y	- 0.27	0.43	2.10	4.83	7.74	12.42	12.15	15.43	24.07	32.06	35.98

Создадим скрипт-файл, реализующий метод наименьших квадратов:

```
x=0:1:10;
n=length(x);
y=[-0.27 0.43 2.1 4.83 7.74 14.42 12.15 15.43 24.07 32.06
35.98];
A(1,1)=sum(x.^4);
A(1,2)=sum(x.^3);
```

```
A(1,3)=sum(x.^2);
A(2,1)=A(1,2);
A(2,2)=A(1,3);
A(2,3)=sum(x);
A(3,1)=A(1,3);
A(3,2)=A(2,3);
A(3,3)=n;
B(1,1)=dot(x.^2,y);
B(2,1)=dot(x,y);
B(3,1)=sum(y);
C=A^-1*B
xpr=0:0.001:10;
a=C(1,1);
b=C(2,1);
c=C(3,1);
ypr=a*xpr.^2+b*xpr+c;
plot(x,y,'o',xpr,ypr); grid on
```

На рисунке 6.2 представлены исходные данные и аппроксимирующая функция.

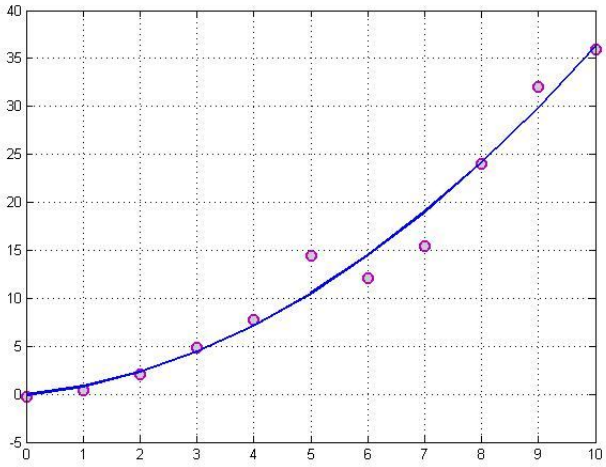


Рис. 6.2. Аппроксимация квадратичной зависимостью табулированной функции.

В результате работы программы получаем значение параметров приближающей функции (6.6):

```
C =

    0.302622377622376

    0.603685314685308

   -0.070209790209802
```

Пример 2. Используя табличные значения примера 1, найти параметры квадратичной аппроксимирующей функции (6.6) с помощью встроенной функции Matlab.

Для решения задачи обобщенной нелинейной регрессии в пакете Matlab имеется функция **`lsqnonlin()`**, возвращающая решение задачи нахождения точки минимума функции:

$$\min_x\left(f(x)\right)=f_1^2(x)+f_2^2(x)+\ldots+f_n^2(x)+L,$$

где $f(x)$ – вектор-функция, x – столбец искомых переменных, L – искомая константа.

Сначала создадим файл-функцию fun_mnk.m, которая возвращает значение вектор-функции $f(x)$:

```
function z=fun_mnk(C,x,y)
N=length(x);
i=1:N
z(i)=y(i)-(C(1)*x(i).^2+C(2)*x(i)+C(3));
```

Далее проведем вычисления значения аппроксимирующей функции с помощью функции `lsqnonlin()`,и визуализируем исходные данные и аппроксимирующую функцию:

```
x=0:1:10;
n=length(x);
y=[-0.27 0.43 2.1 4.83 7.74 14.42 12.15 15.43 24.07 32.06
35.98];
z=[0 0 0];
C=lsqnonlin('fun_mnk',z,[],[],[],x,y);
a=C(1);
b=C(2);
C=C(3);
```

```
xpr=0:0.001:10;
ypr=a*xpr.^2+b*xpr+c;
plot(x,y,'s',xpr,ypr); grid on
```

Параметры приближающей функции:

```
C =
   0.302622377622379   0.603685314685307  -0.070209790209813
```

График исходных данных и приближающей функции:

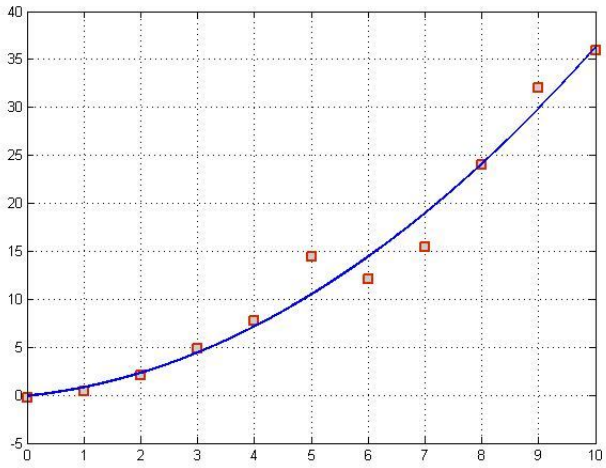


Рис. 6.3. Аппроксимация квадратичной зависимостью табулированной функции с помощью встроенной функции пакета Matlab.

Приближение табличной функции одной переменной полиномом заданного порядка по методу наименьших квадратов может быть реализовано в пакете Matlab с помощью функции **`polyfit(x,y,n)`**, в которой первые два входных аргумента являются векторами со значениями абсцисс и ординат табличной функции, а третий – требуемой степенью полинома.

Пример 3. Приблизить функцию, заданную таблицей полиномами четвертой, пятой и шестой степени и вывести график, отражающий характер приближений.

x_i	0.1	0.3	0.4	0.5	0.7	0.8	1.5
y_i	-3.7	-4.9	-2.0	0.1	0.8	2.5	3.6

Реализация поставленной задачи приведена в следующем скрипт-файле:

```
x=[0.1 0.3 0.4 0.5 0.7 0.8 1.5];
```

```
y=[-3.7 -4.9 -2.0 0.1 0.8 2.5 3.6];
xx=0.1:0.001:1.5;
y4=polyfit(x,y,4);
y5=polyfit(x,y,5);
y6=polyfit(x,y,6);
P4=polyval(y4,xx);
P5=polyval(y5,xx);
P6=polyval(y6,xx);
plot(x,y,'d',xx,P4,xx,P5,xx,P6); grid on
```

Результат аппроксимации представлен в графическом окне (рисунок 6.4).

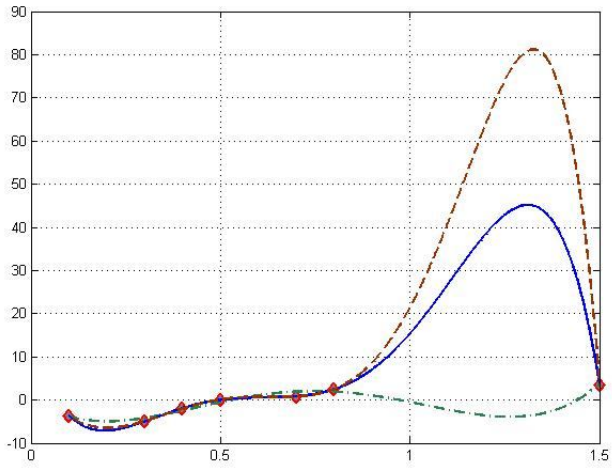


Рис. 6.4. Аппроксимация табулированной функции
с помощью встроенной функции пакета Matlab.

Контрольные вопросы

1. В чем суть приближения таблично заданной функции по методу наименьших квадратов? Чем этот метод отличается от метода интерполяции?

2. Какие элементарные функции чаще всего используются в качестве приближающих?

3. Как можно добиться повышения качества приближения?

4. Какая ошибка является среднеквадратической?

5. Какое из двух приближений одной и той же таблично заданной функции считается лучшим?

6. Каким образом построение приближающих функций в виде различных элементарных функций сводится к случаю линейной функции?

7. За счет чего возникает полиномиальное раскачивание?

Индивидуальные задания

1. С помощью метода наименьших квадратов найти значения параметров приближающей линейной функции и нелинейной функции (выбор нелинейной функции осуществить самостоятельно, исходя из эмпирического анализа данных).

2. Найти аппроксимирующую функцию средствами пакета Matlab. Рассчитать значения накопленных отклонений. Сравнить с результатами, полученными методом наименьших квадратов. Сделать вывод.

3. Построить графики исходных данных и аппроксимирующих функций.

4. Оформить отчет по лабораторной работе.

Номер варианта	**Исходные данные**		**Номер варианта**	**Исходные данные**		**Номер варианта**	**Исходные данные**	
	x	*y*		*x*	*y*		*x*	*y*
1	0	1.5	2	1.5	-0.72	3	2	1.11
	1	2.5		2.5	0.33		5	0.45
	2	3.5		3.5	1.09		8	0.19
	3	5		4	1.28		11	0.07
	4	7.5		5	1.85		16	0.02
4	1	0.6	5	1	0.03	6	1	2.85
	2	1.9		2	0.28		4	7.35
	3	4.3		3	0.34		9	36.50
	4	7.6		4	0.55		10	51.02
	5	12.6		5	0.78		16	343
7	-1	6.62	8	1.6	3.36	9	0	0
	0	3.94		1.8	3.35		1	4.11
	1	2.17		2.1	3.19		2	16.38
	2	1.35		2.3	2.79		5	102.6
	3	0.89		2.9	2.67		7	201
10	-1	13.45	11	2.4	1.30	12	0	0.42
	0	3.01		2.9	1.14		1	0.34
	1	0.67		3.1	1.24		2	0.32
	2	0.15		3.6	1.16		5	0.21
	5	0.05		4.1	1.09		7	0.19

Номер варианта	Исходные данные		Номер варианта	Исходные данные		Номер варианта	Исходные данные	
	x	*y*		*x*	*y*		*x*	*y*
13	1	5.51	14	0.1	0.90	15	-1	6.19
	1.5	4.72		0.4	0.71		0	5.56
	1.9	4.54		0.7	0.95		1	5.05
	2.2	4.33		1.3	0.92		2	4.36
	2.5	4.02		1.9	1.14		5	3.41
16	5	1.5	17	3	-0.73	18	2	2.65
	7	2.5		5	0.36		5	0.95
	11	3.5		7	1.07		8	0.37
	16	5		8	1.25		11	0.12
	21	7.5		10	1.84		16	0.02
19	1	5.55	20	1	0.23			
	2	4.39		2	0.74			
	3	4.09		3	0.48			
	4	4.01		4	0.75			
	5	3.88		5	0.98			

7 ЧИСЛЕННОЕ ДИФФЕРЕНЦИРОВАНИЕ И ИНТЕГРИРОВАНИЕ

Методики дифференцирования и интегрирования имеют важное применение при разработке алгоритмов многих прикладных задач, в т.ч. задач, сводящихся к решениям дифференциальных уравнений и уравнениям в частных производных. К сожалению, аналитические методы не всегда оказываются применимыми. Численные методы дифференцирования приходят на помощь в следующих случаях: аналитический вид функции не задан, возможно сильное усложнение функции при ее аналитическом дифференцировании или требуется получить значения производных с помощью однотипных вычислительных процессов без привлечения аналитических выкладок. Задачи численного интегрирования возникают каждый раз, когда необходимо провести интегрирование функций, для которых в общем случае первообразная среди элементарных функций может и не существовать.

7.1 Численное дифференцирование

Численное дифференцирование, т.е. нахождение производных заданной функции $y = f(x)$ в заданных точках, в отличие от численного интегрирования, можно считать не столь актуальной задачей, поскольку принципиальные трудности с аналитическим нахождением производных отсутствуют. Такая необходимость может возникнуть, например, при необходимости дифференцировании функции, заданной таблично.

Для получения формул численного дифференцирования можно использовать полиномиальную интерполяцию. Если известны значения данной функции $y_i = f(x_i)$ в точках $x_i = x_0 + ih \ \ (i = 0,1,...,n)$ при некотором $h > 0$, то можно найти конечные разности $\Delta^k y_i$ и записать интерполяционный полином, например первый многочлен Ньютона $P_n(x)$:

$$P_n(x) = f_0 + q \cdot \Delta f_0 + \frac{q(q-1)}{2!}\Delta^2 f_0 + \ldots + \frac{q(q-1)\cdot\ldots\cdot(q-n+1)}{n!}\Delta^n f_0 \ ,$$

где $q = \dfrac{x - x_0}{h}$.

Дифференцируя приближенное равенство $f(x) \approx P_n(x)$, можно построить формулы приближенного дифференцирования различной точности в зависимости от степени *n* используемого интерполяционного полинома. Получим конечно-разностную формулу численного дифференцирования:

$$f'(x) \approx q'_x \left[P_n(x_0 + qh)\right]'_q = \frac{1}{h}\left(\Delta y_0 + \frac{2q-1}{2}\Delta^2 y_0 + \frac{3q^2 - 6q + 2}{6}\Delta^3 y_0 + \right.$$

$$\left. + \frac{4q^3 - 18q^2 + 22q - 6}{24}\Delta^4 y_0 \right). \qquad (7.1)$$

Рассмотрим частные случаи формул численного дифференцирования. Фиксируя степень *n*, лежащего в основе интерполяционного многочлена, имеем (для некоторых $\delta > 0$, определяющих промежуток экстраполяции приемлемого качества):

на основе линейной интерполяции

$$f'(x) \approx \frac{\Delta y_0}{h} \text{ для } x \in (x_0 - \delta, x_1 + \delta); \qquad (7.2)$$

на основе квадратичной интерполяции

$$f'(x) \approx \frac{1}{h}\left(\Delta y_0 + \frac{2q-1}{2}\Delta^2 y_0 \right) \text{ для } x \in (x_0 - \delta, x_2 + \delta); \qquad (7.3)$$

на основе кубической интерполяции

$$f'(x) \approx \frac{1}{h}\left(\Delta y_0 + \frac{2q-1}{2}\Delta^2 y_0 + \frac{3q^2 - 6q + 2}{6}\Delta^3 y_0 \right) \text{ для } x \in (x_0 - \delta, x_3 + \delta). \qquad (7.4)$$

Однако в вычислительной практике удобнее оказываются выражения формул численного дифференцирования через значения функции в узловых точках. Учитывая, что точкам x_0, x_1, x_2 соответствуют значения $q = 0,1,2$ и раскрывая конечные разности через значения y_i, $i = 0,1,2$, имеем:

при $n = 1$ из (7.2)

$$f'(x_0) \approx y'_0 = \frac{y_1 - y_0}{h},$$

$$f'(x_1) \approx y'_1 = \frac{y_1 - y_0}{h};$$

при $n = 2$ из (7.3)

$$f'(x_0) \approx y'_0 = \frac{1}{2h}(-3y_0 + 4y_1 - y_2),$$

$$f'(x_1) \approx y'_1 = \frac{1}{2h}(y_2 - y_0),$$

$$f'(x_2) \approx y'_2 = \frac{1}{2h}(y_0 - 4y_1 + 3y_2).$$

При необходимости этот ряд формул может быть продолжен с помощью общей формулы (7.1). Повторное дифференцирование приближенного равенства (7.1), приводит к конечно-разностной формуле вычисления второй производной:

$$f''(x) \approx \frac{1}{h^2}\left(\Delta^2 y_0 + (q-1)\Delta^3 y_0 + \frac{1}{12}(6q^2 - 18q + 11)\Delta^4 y_0 + \ldots\right). \qquad (7.5)$$

Из формулы общего вида (7.5) также можно получить наиболее употребительные частные случаи, представляющие собой выражения вторых производных через значения функции в узловых точках. В частности, в точке x_1 имеем приближенное равенство:

$$f''(x_1) \approx y''_1 = \frac{y_2 - 2y_1 + y_0}{h^2}. \qquad (7.6)$$

7.2 Численное интегрирование

Если функция $f(x)$ непрерывна на отрезке $[a,b]$ и известна ее первообразная $F(x)$, то определенный интеграл от этой функции в пределах от *a* до *b* может быть вычислен по формуле Ньютона-Лейбница

$$\int_a^b f(x)dx = F(b) - F(a), \qquad (7.7)$$

где $F'(x) = f(x)$.

Но часто возникают ситуации, когда вычислить интеграл можно только с помощью численных методов:

1) $F(x)$ не выражается через элементарные функции;

2) $F(x)$ существует и выражается через элементарные функции, но ее сложно найти;

3) найдена $F(x)$, но сложно вычислить ее значение;

4) $f(x)$ задана таблично или графически.

Таким образом, ставится задача о приближенном вычислении интеграла $\int_a^b f(x)dx$. Обычный прием состоит в том, что данную функцию $f(x)$ на рассматриваемом отрезке $[a,b]$ заменяют интерполирующей функцией $P_n(x)$ простого вида, а затем приближенно полагают: $\int_a^b f(x)dx \approx \int_a^b P_n(x)dx$.

Метод прямоугольников. Геометрический смысл определенного интеграла $F = \int_a^b f(x)dx$ – площадь фигуры, ограниченной графиком функции $f(x)$ и прямыми: $x = a,\ x = b$. Разделим отрезок $[a,b]$ на n равных отрезков длиной Δx: $\Delta x = \dfrac{b-a}{n}$. Координата правого конца i-го отрезка определяется по формуле $x_i = x_0 + i \cdot \Delta x,$ где $x_0 = a$, $i = \overline{0,n}$.

Простейшая оценка площади кривой может быть получена как сумма площадей прямоугольников, одна из сторон которого равна длине отрезка $[x_i, x_{i+1}]$, а высота равна значению $f(x_i)$ – *метод левых прямоугольников* (рисунок 7.1), или $f(x_{i+1})$ – *метод правых прямоугольников* (рисунок 7.2).

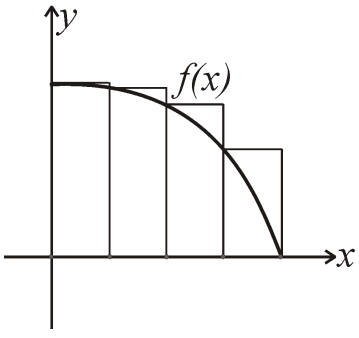


Рис. 7.1. Геометрическая интерпретация метода левых прямоугольников.

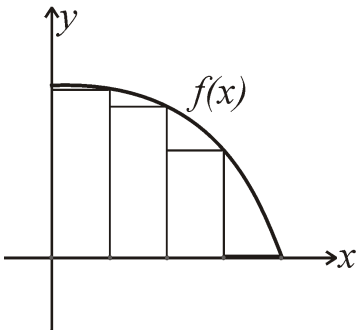


Рис. 7.2. Геометрическая интерпретация метода правых прямоугольников.

Тогда для левых прямоугольников определенный интеграл вычисляется по формуле

$$F_L = \sum_{i=0}^{n-1} f(x_i) \cdot \Delta x, \tag{7.8}$$

а для правых прямоугольников – по формуле

$$F_R = \sum_{i=1}^{n} f(x_i) \cdot \Delta x. \tag{7.9}$$

Погрешность метода левых и правых прямоугольников пропорциональна n^{-1}.

Метод трапеций. Заменим функцию на каждом интервале $[x_i, x_{i+1}]$, $i = \overline{0, n-1}$ отрезком прямой, проходящей через точки $(x_i, f(x_i))$ и $(x_{i+1}, f(x_{i+1}))$. Тогда фигура, ограниченная графиком функции и прямыми $x = x_i$, $x = x_{i+1}$, является трапецией. Тогда определенный интеграл определяется как сумма площадей всех трапеций по формуле:

$$F_n = \frac{1}{2}\sum_{i=0}^{n-1}\left(f(x_{i+1}) + f(x_i)\right) \cdot \Delta x = \left[\frac{1}{2} f(a) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2} f(b)\right] \cdot \Delta x. \tag{7.10}$$

Полная погрешность формулы трапеций на отрезке $[a, b]$ по порядку величины равна $\mathrm{O}(n^{-2})$.

Метод Симпсона. Формула получается при использовании параболической интерполяции по трем соседним точкам

$$y = ax^2 + bx + c. \tag{7.11}$$

Для нахождения параметров a, b, c полинома, проходящего через три точки (x_0, y_0), (x_1, y_1), (x_2, y_2), решаем систему линейных уравнений

$$\begin{cases} y_0 = ax_0^2 + bx_0 + c, \\ y_1 = ax_1^2 + bx_1 + c, \\ y_2 = ax_2^2 + bx_2 + c. \end{cases} \tag{7.12}$$

Решив систему (7.12), подставим найденные значения в (7.11) и получим

$$y = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_0 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}. \tag{7.13}$$

Интегрируя (7.13) на отрезке $[x_0, x_2]$, находим

$$F_0 = \frac{1}{3}(y_0 + 4y_1 + y_2) \cdot \Delta x,$$

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

Искомый определенный интеграл находится как площадь всех параболических сегментов:

$$F_n = \frac{1}{3}[f(a) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(b)] \cdot \Delta x.$$

В формуле Симпсона число n должно быть четным.

Полная погрешность формулы Симпсона на отрезке $[a, b]$ по порядку величины составляет $O(n^{-4})$.

Метод трех восьмых.

$$\begin{aligned} F_n = \frac{3 \cdot \Delta x}{8}[f(a) + f(x_{3m}) + 2(f(x_3) + f(x_6) + \ldots + f(x_{3m-3})) + \\ + 3(f(x_1) + f(x_2) + f(x_4) + f(x_5) + \ldots + f(x_{3m-2}) + f(x_{3m-1}))], \end{aligned} \tag{7.14}$$

где $\Delta x = \dfrac{b-a}{n} = \dfrac{b-a}{3m}$.

В формуле (7.14) число *n* должно быть равно $3m$.

Значение полной погрешности правила трех восьмых по порядку величины совпадает со значением полной погрешности формулы Симпсона.

Заметим, что оценка погрешности вычисления интеграла от функции, зависящей от d переменных, определяется следующим образом: если для одномерного случая погрешность составляет $\mathrm{O}\left(n^{-\alpha}\right)$, то в d-мерном случае она равна $\mathrm{O}\left(n^{-\frac{\alpha}{d}}\right)$.

Квадратурные формулы Гаусса. В квадратурных формулах Гаусса

$$\int_a^b f(t)dt = \sum_{i=1}^{n} A_i f(t_i) + R_n(f)$$

коэффициенты A_i и абсциссы t_i (i=1, 2, …, n) подбираются так, чтобы формула была точной для всех многочленов наивысшей возможной степени N.

Доказано, что такие числа A_i, t_i определяются однозначно при $N = 2n-1$. В таблице приведены значения абсцисс t_i и коэффициентов A_i, а также формулы остаточных членов $R_n(f)$ при $n = 4,5,7$.

n	t_i	A_i	$R_n(f)$
4	$-t_1 = t_4 = 0.861136312$ $-t_2 = t_3 = 0.339981044$	$A_1 = A_4 = 0.347854845$ $A_2 = A_3 = 0.652145155$	$R_4(f) \approx 2.88 \cdot 10^{-7} f^{(8)}(\xi)$ $-1 < \xi < 1$
5	$-t_1 = t_5 = 0.906179846$ $-t_2 = t_4 = 0.538469310$ $t_3 = 0$	$A_1 = A_5 = 0.236926885$ $A_2 = A_4 = 0.478628670$ $A_3 = 0.568888889$	$R_5(f) \approx 8.08 \cdot 10^{-4} f^{(10)}(\xi)$ $-1 < \xi < 1$
7	$-t_1 = t_7 = 0.949107912$ $-t_2 = t_6 = 0.741531186$ $-t_3 = t_5 = 0.405845151$ $t_4 = 0$	$A_1 = A_7 = 0.129484966$ $A_2 = A_6 = 0.279705391$ $A_3 = A_5 = 0.381830051$ $A_4 = 0.417959184$	$R_7(f) \approx 2.13 \cdot 10^{-15} f^{(14)}(\xi)$ $-1 < \xi < 1$

Неудобство применения квадратурной формулы Гаусса состоит в том, что абсциссы t_i и коэффициентов A_i, вообще говоря, иррациональные числа. Этот недостаток искупается ее высокой точностью при сравнительно малом числе узлов интегрирования. В тех случаях, когда подынтегральная функция сложна и на вычисление ее значений в каждом узле интегрирования требуется много времени, применение формулы Гаусса особенно выгодно.

Получить оценку погрешности результата, используя формулу остаточного члена, для формул Гаусса удается очень редко, так как это связано с вычислением производных высоких порядков от подынтегральной функции. В настоящее время разработаны практически более удобные методы, позволяющие осуществить контроль точности.

При вычислении интеграла $\int_a^b f(x)dx$ следует сделать замену переменной

$$x = \frac{b+a}{2} + \frac{b-a}{2}t\,.$$

Тогда формула Гаусса будет иметь вид $\int_a^b f(x)dx = \frac{b-a}{2}\sum_{i=1}^{n} A_i f(x_i) + R_n^*(f)$, где

$$x_i = \frac{b+a}{2} + \frac{b-a}{2}t_i,\ R_n^*(f) = \left(\frac{b-a}{2}\right)^{2n+1} R_n(f).$$

Численное интегрирование методом Монте-Карло. Пусть существует прямоугольник высотой H и длиной $(b-a)$, такой, что функция $f(x)$ целиком лежит внутри прямоугольника. Сгенерируем *n* пар случайных чисел, равномерно распределенных в данном прямоугольнике: $a \le x_i \le b,\ \ 0 \le y_i \le H$.

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. Оценка интеграла может быть получена по формуле

$$F_n = A\frac{n_s}{n}, \qquad (7.15)$$

где n_s – число точек, удовлетворяющих условию $y_i \le f(x_i)$, A – площадь прямоугольника.

Можно вычислить определенный интеграл как среднее значение функции $f(x)$ на отрезке $[a,b]$:

$$F_n = \frac{(b-a)}{n}\sum_{i=1}^{n} f(x_i), \qquad (7.16)$$

где x_i – последовательность случайных чисел с равномерным законом распределения на отрезке $[a,b]$.

В отличие от рассмотренных выше методов, погрешность метода Монте-Карло не зависит от размерности подынтегральной функции d и составляет $O\left(n^{-0.5}\right)$. Поэтому при достаточно больших d интегрирование по методу Монте-Карло приводит к меньшим погрешностям при тех же значениях n.

7.3 Численные примеры. Реализация в Matlab

Для аппроксимации производных конечными разностями в пакете Matlab существует функция **`diff()`**:

diff(x) – возвращает конечные разности, вычисленные по смежным элементам вектора x;

diff(x, n) – возвращает конечные разности n-го порядка, вычисленные по смежным элементам вектора x.

Пример 1. Вычислить значение производной функции на отрезке $[0,6]$.

$$f(x)=x^3-\sin(3x)-1. \qquad (7.17)$$

Листинг программы, дифференцирующей заданную функцию (7.17) численно, аналитически и с помощью встроенного инструментария Matlab, приведен далее.

```
dx=0.1; % задание шага координтаной сетки
x=0:dx:6; %вычисление координат узлов
f=funI(x); % вычисление значения функции
i=1:length(x)-1;
dif_f(i)=(f(i+1)-f(i))./dx; % аппроксимация производной правой разностью
difML=diff(f)./dx; % аппроксимация производной функции с спомощью встреонного инструментария Matlab
dif_f_ex=fun_dif_I(x);% сравнение с аналитически найденной производной функции f(x)
figure (1)
plot(x(i),dif_f(i),'ro',x,dif_f_ex,'b-',x(i),difML(i),'gs');grid on
figure (2)
plot(x(i),dif_f(i)-dif_f_ex(i)); grid on
```

```
function z=fun_dif_I(x)
z=3*x.^2-3*cos(3*x);
function z=funI(x)
z=x.^3-sin(3*x)-1;
```

Визуализация производной функции (7.17), разности между численными и аналитическими значениями первой производной с шагом сетки $dx = 0.1$ и $dx = 0.001$ изображены на рисунках 7.3, 7.4, 7.5.

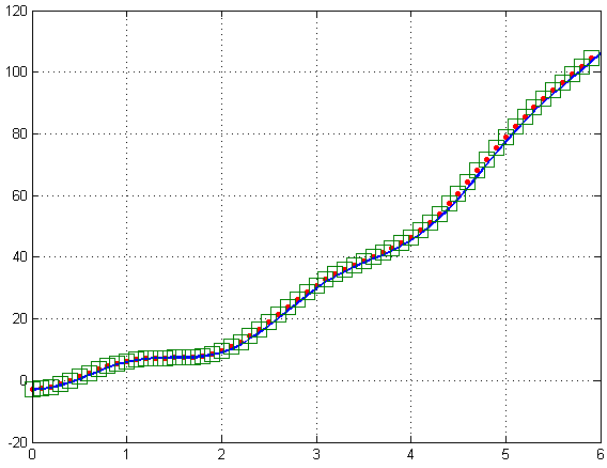


Рис. 7.3. Визуализация производной функции (7.17) (сплошная кривая – производная, вычисленная аналитически; точечный массив (маркер «o») – численная аппроксимация производной с шагом сетки $dx = 0.1$; точечный массив (маркер «□») –аппроксимация производной с помощью инструментария пакета).

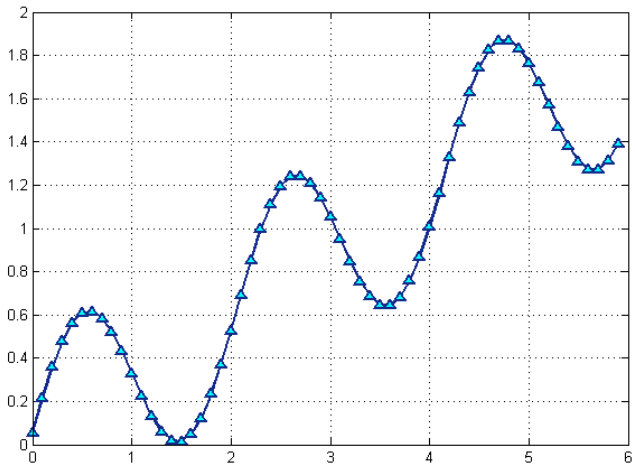


Рис. 7.4. Разность между численными и аналитическими значениями первой производной с шагом сетки $dx = 0.01$.

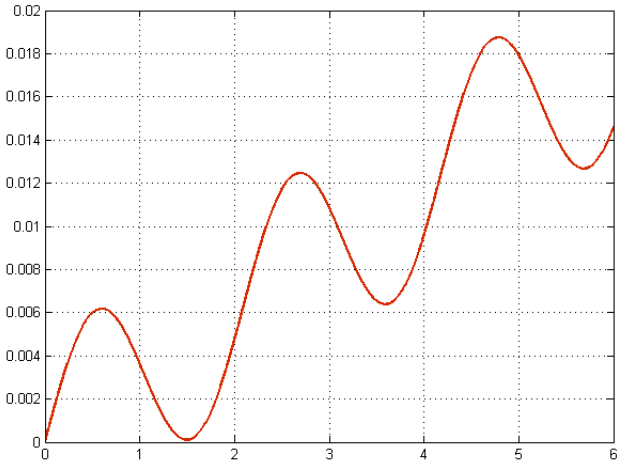


Рис. 7.5. Разность между численными и аналитическими значениями производной с шагом сетки $dx = 0.001$.

Для вычисления значений определенных интегралов в пакете Matlab применяются функции **`quad(), quad1(), trapz().`**

Функция **`quad(fun,a,b)`** – возвращает значение интеграла от функции *fun* на отрезке $[a,b]$, при вычислении используется метод Симпсона.

Функция **`quad1(fun,a,b)`** – возвращает значение интеграла от функции *fun* на отрезке $[a,b]$, используя для вычисления метод Лоббато.

Функция **`trapz(y)`** – возвращает значение определенного интеграла в предположении, что $x = 1 : length(y)$.

Функция **`trapz(x,y)`** – возвращает значение определенного интеграла на отрезке $[x(1), x(n)]$.

Пример 2. Вычислить значение определенного интеграла

$$\int_0^6 \left(x^3 - 1 - \sin(3x)\right) dx, \qquad (7.18)$$

используя метод правых прямоугольников и встроенные возможности пакета Matlab.

Для выполнения задания оформим листинг программы:

```
xmin=0; % задание ниженего предела
xmax=6; %задание верхнего предела
dx=0.001; %задание шага интегрирования
```

```
x=0:dx:6; %вычисление координат узлов
N=length(x);
f=funI(x); % вычисление значения функции
S=0;
for j=2:N
    S=S+f(j);
end
I=S*dx % вычисление значение интеграла методом правых прямо-
угольников
```

Результатом работы программы является значения интеграла, полученные методом правых прямоугольников и с помощью встроенной функции пакета:

```
I = 317.9951568146258 (метод правых прямоугольников)
ans = 317.8867813210021 (значение, полученное с помощью
встроенной функции пакета)
```

Можно отметить, что интеграл (7.18) может быть вычислен аналитически:

$$\int_0^6 \left(x^3 - 1 - \sin(3x)\right)dx = \left(\frac{x^4}{4} - x + \frac{1}{3}\cos(3x)\right)\Bigg|_0^6 \approx 317.8867.$$

Поэтому, в условиях данного примера, ошибка вычисления методом правых прямоугольников составляет 0.108, а ошибка вычисления интеграла с помощью встроенной функции пакета – 0.0000091.

Пример 2. Вычислить значение определенного интеграла (7.18) методом Монте-Карло.

Решение оформим в виде следующего листинга:

```
xmin=0;
xmax=6;
ymin=funI(xmin);
ymax=funI(xmax);
N=1000; %процесс генерации случайных координат
x1=xmin+(xmax-xmin)*rand(N,1);
y1=ymin+(ymax-ymin)*rand(N,1);
```

```
S=0;
for i=1:N
    if y1(i)<=funI(x1(i))
        S=S+1;
    end
end
IMK=S*(xmax-xmin)*(ymax-ymin)./N
Iex=(6^4/4-6+cos(3*6)/3)-1/3; % точное значение интеграла
Iex-IMK
```

В результате получаем значение интеграла, вычисленное по формуле (7.15), а также значение ошибки:

```
IMK =  308.2199
ans =    9.6669
```

Контрольные вопросы

1. Каким образом можно повысить точность численного дифференцирования?

2. В чем основные преимущества формулы трапеций по сравнению с методом прямоугольников?

3. Как выбирается шаг интегрирования?

4. Чему равен порядок погрешности формулы Симпсона для двумерной подынтегральной функции?

5. Какие условия обязательно должны выполняться в методе трех восьмых и почему?

Индивидуальные задания

1. Найдите значение первой и второй производных функции, заданной таблично, в указанных точках.

Номер варианта	***x***	***f*(*x*)**	**Значения аргумента**	**Номер варианта**	***x***	***f*(*x*)**	**Значения аргумента**
1, 2, 3	1.50	0.51183	*1 вариант* $x = 1.50$ $x = 1.52$	4, 5, 6	0.50	1.6487	*4 вариант* $x = 0.59$ $x = 0.61$
	1.51	0.50624			0.51	1.6653	
	1.52	0.50064			0.52	1.6820	
	1.53	0.49503			0.53	1.6989	
	1.54	0.48940	*2 вариант*		0.54	1.7160	*5 вариант*

	1.55	0.48376	$x = 1.61$ $x = 1.55$		0.55	1.7333	$x = 0.50$ $x = 0.54$
	1.56	0.47811			0.56	1.7507	
	1.57	0.47245			0.57	1.7683	
	1.58	0.46678	*3 вариант* $x = 1.60$ $x = 1.58$		0.58	1.7860	*6 вариант* $x = 0.50$ $x = 0.57$
	1.59	0.46110			0.59	1.8040	
	1.60	0.45540			0.60	1.8221	
	1.61	0.45321			0.61	1.8514	
7, 8, 9	0.00	0.28081	*7 вариант* $x = 0.00$ $x = 0.30$	10, 11, 12	1.1	0.89121	*10 вариант* $x = 1.1$ $x = 1.6$
	0.05	0.31270			1.2	0.93204	
	0.10	0.34549			1.3	0.96356	
	0.15	0.37904			1.4	0.98545	
	0.20	0.41318	*8 вариант* $x = 0.05$ $x = 0.50$		1.5	0.99749	*11 вариант* $x = 2.0$ $x = 2.2$
	0.25	0.44774			1.6	0.99957	
	0.30	0.48255			1.7	0.99166	
	0.35	0.51745			1.8	0.97385	
	0.40	0.55226	*9 вариант* $x = 0.25$ $x = 0.51$		1.9	0.94630	*12 вариант* $x = 1.4$ $x = 2.2$
	0.45	0.58682			2.0	0.90930	
	0.50	0.62096			2.1	0.86321	
	0.51	0.75263			2.2	0.75412	
13, 14, 15	1.0	0.5652	*13 вариант* $x = 1.0$ $x = 1.5$	16, 17, 18	0.10	3.63004	*16 вариант* $x = 0.10$ $x = 1.60$
	1.1	0.6375			0.35	3.75680	
	1.2	0.7147			0.60	3.88933	
	1.3	0.7973			0.85	4.03258	
	1.4	0.8861	*14 вариант* $x = 1.9$ $x = 2.1$		1.10	4.19310	*17 вариант* $x = 2.35$ $x = 2.85$
	1.5	0.9817			1.35	4.38042	
	1.6	1.0848			1.60	4.60963	
	1.7	1.1964			1.85	4.90697	
	1.8	1.3172	*15 вариант* $x = 1.2$ $x = 2.1$		2.10	5.32331	*18 вариант* $x = 0.85$ $x = 2.85$
	1.9	1.4482			2.35	5.97322	
	2.0	1.5906			2.60	7.18210	
	2.1	1.6231			2.85	8.13625	
19, 20	0.50	1.6487	*19 вариант* $x = 0.50$ $x = 0.57$				
	0.51	1.6653					
	0.52	1.6820					
	0.53	1.6989					
	0.54	1.7160	*20 вариант* $x = 0.55$ $x = 0.61$				
	0.55	1.7333					
	0.56	1.7507					
	0.57	1.7683					
	0.58	1.7860					
	0.59	1.8040					
	0.60	1.8221					
	0.61	1.8526					

2. Вычислить приближенное значение определенного интеграла указанными методами. Вычислить определенный интеграл с помощью встроенной функции ППП Matlab. Сравнить полученные значения интеграла с точным.

Номер варианта	**Интеграл**	**Пределы интегрирования** $[a, b]$	**Метод**
1	$\int_a^b \frac{\ln^2(x)}{x} dx$	[1, e]	1. Формула трапеций. 2. Квадратурная формула Гаусса.
2		[2, e^2]	
3	$\int_a^b \sqrt{4-x^2}\, dx$	[0, 2]	1. Формула трех восьмых. 2. Метод Монте-Карло.
4		[0, 1]	
5	$\int_a^b \frac{x}{1+x^4} dx$	[0, 1]	1. Формула трапеций. 2. Метод Монте-Карло.
6		[-1, 3]	
7	$\int_a^b \frac{\sin^2(x)}{\cos(x)} dx$	[0, π/6]	1. Формула трех восьмых. 2. Квадратурная формула Гаусса.
8		[0, π/3]	
9	$\int_a^b x \cdot \mathrm{arctg}(x) dx$	[-1, 1]	1. Формула трапеций. 2. Квадратурная формула Гаусса.
10		[-2, 2]	
11	$\int_a^b \frac{dx}{x^2+x}$	[1, 3]	1. Формула трех восьмых. 2. Метод Монте-Карло.
12		[1, 5]	
13	$\int_a^b \cos(\ln(x)) dx$	[1, exp(π/2)]	1. Формула трапеций. 2. Метод Монте-Карло.
14		[1, exp(3π/2)]	
15	$\int_a^b x^3 \sqrt{x^2-1} dx$	[2, 6]	1. Формула трех восьмых. 2. Квадратурная формула Гаусса.
16		[2, 4]	
17	$\int_a^b \frac{dx}{e^x-1}$	[ln(2), 2ln(2)]	1. Формула трапеций. 2. Квадратурная формула Гаусса.
18		[ln(2), 6ln(2)]	
19	$\int_a^b \frac{2 \cdot \arcsin(x)}{\sqrt{1-x^2}} dx$	[-0.25, 0.25]	1. Формула трех восьмых. 2. Метод Монте-Карло.
20		[-0.5, 0.5]	

3. Оформить отчет по лабораторной работе.

8 ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ КОШИ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Рассмотрим обыкновенное дифференциальное уравнение первого порядка следующего общего вида:

$$y' = f(x, y), \quad x \in [x_0, b] \tag{8.1}$$

с начальным условием:

$$y(x_0) = y_0, \tag{8.2}$$

где $f(x, y)$ – некоторая заданная, в общем случае, нелинейная функция двух переменных.

Задача (8.1) – (8.2) называется *задачей с начальными условиями* или *задачей Коши.* Пусть для задачи (8.1) – (8.2) выполнены требования, обеспечивающие существование и единственность ее решения $y = y(x)$ на отрезке $[x_0, b]$, кроме того, искомое решение обладает той или иной степенью гладкости.

Аналитические методы решения задач вида (8.1) – (8.2) применимы только для специальных типов уравнений. Используемые на практике приближенные методы решения задач Коши для обыкновенных дифференциальных уравнений можно условно разделить на три основные группы: приближенно-аналитические методы, графические методы и численные (сеточные) методы.

Приближенно-аналитические методы включают такие, которые позволяют находить решение $y = y(x)$ в виде некоторой аппроксимирующей функции $\varphi(x)$. Примером такого метода является, например, метод степенных рядов, реализация которого основана на представлении искомой функции отрезком ряда Тейлора с учетом заданной точности, где коэффициенты определяются последовательным дифференцированием самого уравнения. К этой группе методов относят также методы неопределенных коэффициентов и последовательных приближений.

Графические методы основаны на приближенном представлении искомого решения $y = y(x)$ на промежутке $[x_0, b]$ в виде графика, который можно строить по тем или иным правилам, связанными с графической интерпретацией задачи.

Численные методы предполагают получение набора приближенных значений y_i искомого решения $y(x)$ на некоторой сетке аргументов $x_i \in [x_0, b]$.

8.1 Метод Пикара

Проинтегрируем обе части уравнения (8.1) в границах от x_0 до x:

$$\int_{x_0}^{x} y'(t)dt = \int_{x_0}^{x} f(t, y(t))dt .$$

Отсюда, с учетом условия (8.2), получим:

$$y(x) = y_0 + \int_{x_0}^{x} f(t, y(t))dt . \qquad (8.3)$$

Применяя к последнему интегральному уравнению метод простых итераций:

$$y_{n+1} = \varphi(y_n), \quad n = 0,1,2,...,$$

и выбирая в качестве начальной функции $y(x_0) = y_0$, по формуле (8.3) при n=0 находим первое приближение:

$$y_1(x) = y_0 + \int_{x_0}^{x} f(t, y_0)dt ,$$

подстановка которого в (8.3) при $n = 1$ дает второе приближение, и т.д. Таким образом, получим *метод последовательных приближений* или *метод Пикара*:

$$y_{n+1}(x) = y_0 + \int_{x_0}^{x} f(t, y_n(t))dt, \quad n = 0,1,2,... \quad y_0(x) \equiv y_0 . \qquad (8.4)$$

Оценка погрешности k-го приближения имеет вид:

$$|y(x) - y_k(x)| \le M^k \cdot N \frac{d^{k+1}}{(k+1)!}, \qquad (8.5)$$

где $M = \max\left|f'_y(x)\right|$ – константа Липшица, $\left|f(x,y)\right| \le N$ – верхняя грань модуля функции $f(x,y)$, d – величина, определяющая окрестность $|x - x_0| \le d$, $d = \min\left(a, \frac{b}{N}\right)$.

8.2 Метод Эйлера

Пусть требуется построить таблицу приближенных значений y_i решения $y = y(x)$ задачи Коши (8.1) – (8.2) в расчетных точках $x_i = x_0 + ih,\ \ i = \overline{1,n}$ с расчетным шагом $h = \frac{b - x_0}{n}$:

x	x_0	x_1	…	$x_n = b$
y	y_0	y_1	…	$y_n \approx y(b)$

Для построения метода Эйлера можно использовать различные подходы. Рассмотрим идею графического построения решения дифференциальных уравнений (рисунок 8.1).

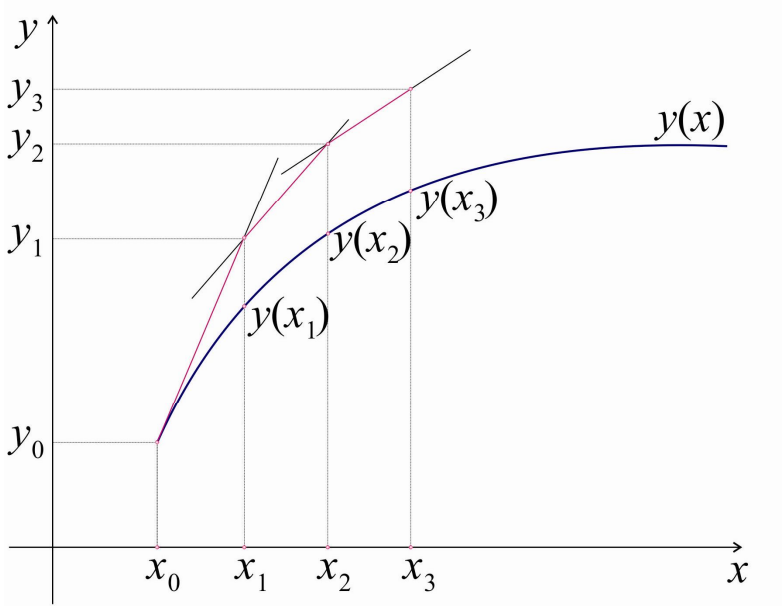


Рис. 8.1. Геометрическая интерпретация метода Эйлера.

Запишем уравнение касательной к кривой $y = y(x)$ в точке (x_0, y_0):

$$y = y_0 + f(x_0, y_0)(x - x_0). \qquad (8.6)$$

При достаточно малом шаге h ордината $y_1 = y_0 + h \cdot f(x_0, y_0)$ этой касательной (при подстановке значения $x_1 = x_0 + h$) по непрерывности должна мало отличаться от ординаты $y(x_1)$ решения $y(x)$ задачи (8.1) – (8.2). Следовательно, точка (x_1, y_1) пересечения касательной (8.6) с прямой $x = x_1$ может быть приближенно принята за начальную точку. Через эту точку снова проведем прямую $y = y_1 + f(x_1, y_1)(x - x_1)$, которая уже приближенно отражает поведение касательной к $y = y(x)$ в точке $(x_1, y(x_1))$. Пересекая эту касательную с прямой $x = x_2$, получив приближенное значение $y(x_2)$ и т.д., строим итерационный процесс *метода Эйлера*:

$$y_{i+1} = y_i + h \cdot f(x_i, y_i), \quad i = 0,1,2,...,n, \qquad (8.7)$$

который приближенно график решения $y = y(x)$ представляет в виде ломанной, составленный из отрезков приближенных касательных.

Соотношение (8.1) можно получить и аналитически. Линеаризуя решение в окрестности начальной точки по формуле Тейлора, получим:

$$y(x) = y(x_0) + y'(x_0)(x - x_0) + \frac{y''(\xi)}{2!}(x - x_0)^2,$$

откуда легко получить и саму формулу метода Эйлера (7) для n последовательных шагов и ее остаточный член: $r_k(h) = \frac{y''(\xi_k)}{2} h^2$, где ξ_k – некоторая точка интервала (x_0, x_k). Остаточный член характеризует локальную ошибку метода Эйлера – $O(h^2)$, совершаемую на каждом шаге. После $n \sim \frac{1}{h}$ шагов глобальная ошибка будет $O(h)$, т.е. метод Эйлера относится к методам первого порядка.

Другим подходом к получению формулы (8.7) является аппроксимация производной в выражении (8.1) правыми разностями.

Известны различные модификации метода Эйлера, которые направлены на уточнение направления перехода из точки (x_i, y_i) в точку (x_{i+1}, y_{i+1}). Например, в *методе Эйлера-Коши* используется следующий порядок вычисления:

$$y^*_{i+1} = y_i + h \cdot f(x_i, y_i), \quad y_{i+1} = y_i + h \frac{f(x_i, y_i) + f(x_{i+1}, y^*_{i+1})}{2}. \tag{8.8}$$

Метод второго порядка точности можно получить, используя разложение в ряд Тейлора функции $y(x)$ в окрестности некоторой точки x_i:

$$y(x_{i+1}) = y(x_i) + h \cdot y'(x_i) + \frac{1}{2!} h^2 \cdot y''(x_i) + O(h^3). \tag{8.9}$$

Дифференцируя (8.1) по формуле полной производной

$$y''(x) = f'_x(x, y) + f'_y(x, y) \cdot y'$$

найдем приближенное значение второй производной:

$$y''(x_i) = f'_x(x_i, y(x_i)) + f'_y(x_i, y(x_i)) \cdot f(x_i, y(x_i)) =$$
$$= f'_x(x_i, y_i) + f'_y(x_i, y_i) \cdot f(x_i, y_i).$$

Подставляя последнее выражение в (8.9), получим *исправленный метод Эйлера*:

$$y(x_{i+1}) = y_i + h\left[f(x_i, y_i) + \frac{h}{2}\left(f'_x(x_i, y_i) + f'_y(x_i, y_i) \cdot f(x_i, y_i)\right)\right]. \tag{8.10}$$

8.3 Метод Рунге-Кутты

Идея построения явных *методов Рунге-Кутты p-го порядка* заключается в получении приближений к значениям $f(x_{i+1})$ по формуле вида:

$$y_{i+1} = y_i + h \cdot \varphi(x_i, y_i, h), \tag{8.11}$$

где $\varphi(x, y, h)$ – некоторая функция, приближающая отрезок ряда Тейлора до *p*-го порядка и не содержащая частных производных функции $f(x, y)$.

Полагая в (8.11), что $\varphi(x, y, h) \equiv f(x, y)$, приходим к методу Эйлера, который можно считать частным случаем метода Рунге-Кутты при $p = 1$.

Рассмотрим построение методов Рунге-Кутты для $p = 2$. Пусть функция $\varphi(x, y, h)$ имеет следующую структуру:

$$\varphi(x,y,h)=c_1 f(x,y)+c_2 f(x+a\cdot h, y+b\cdot h\cdot f(x,y)).$$

Параметры c_1, c_2, a, b будем подбирать так, чтобы формула (8.11) определяла метод второго порядка, т.е. чтобы максимальная локальная ошибка составляла величину $O(h^3)$. Разложим функцию двух переменных $f(x+a\cdot h, y+b\cdot h\cdot f(x,y))$ по формуле Тейлора, ограничиваясь линейными членами:

$$f(x+a\cdot h, y+b\cdot h\cdot f(x,y)) = f(x,y)+f'_x(x,y)\cdot ah + f'_y(x,y)\cdot b\cdot h\cdot f(x,y)+O(h^2).$$

Подстановка этого выражения в (8.11) дает:

$$y_{i+1} = y_i + h\left[(c_1+c_2)\cdot f(x_i,y_i)+h\cdot\left(c_2\cdot a\cdot f'_x(x_i,y_i)+c_2\cdot b\cdot f'_y(x_i,y_i)\cdot f(x_i,y_i)\right)\right]+ \\ +O(h^3)$$

Сравнение последнего выражения с тейлоровским квадратичным представлением решения $y(x)$ (8.10) с точностью до $O(h^3)$ требует выполнения совокупности условий:

$$\begin{cases} c_1+c_2=1, \\ c_2\cdot a=0.5, \\ c_2\cdot b=0.5, \end{cases}$$

считая c_2 свободным параметром α, получим *однопараметрическое семейство методов Рунге-Кутты второго порядка*:

$$y_{i+1} = y_i + h\left[(1-\alpha)f(x_i,y_i)+\alpha f\left(x_i+\frac{h}{2\alpha}, y_i+\frac{h}{2\alpha}f(x_i,y_i)\right)\right]+O(h^3) \quad (8.12)$$

Данный метод является *двухэтапным* по количеству вычислений на каждом шаге.

Анализ метода Рунге-Кутты второго порядка позволяет представить, в какой форме следует конструировать *метод Рунге-Кутты произвольного порядка*. Используется запись, состоящая из следующей последовательности формул, параметры c_k, a_k, b_{kj} в которых подбираются так, чтобы получаемое значение y_{i+1} совпадало со значением разложения $y(x_{i+1})$ по формуле Тейлора с погрешностью $O(h^{p+1})$:

$$\begin{cases} \theta_1^i = f(x_i, y_i), \\ \theta_k^i = f\left(x_i + a_k \cdot h, y_i + h \sum\limits_{j=1}^{k-1} b_{kj} \cdot \theta_j^i \right), \quad k = 2,3,..., p, \\ y_{i+1} = y_i + h \sum\limits_{k=1}^{p} c_k \cdot \theta_k^i. \end{cases} \tag{8.13}$$

Наиболее употребительным частным случаем семейства методов (8.13) является *метод Рунге-Кутты четвертого порядка*:

$$\begin{cases} \theta_1^i = f(x_i, y_i), \\ \theta_2^i = f\left(x_i + \dfrac{h}{2}, y_i + \dfrac{h}{2}\theta_1^i \right), \\ \theta_3^i = f\left(x_i + \dfrac{h}{2}, y_i + \dfrac{h}{2}\theta_2^i \right), \\ \theta_4^i = f\left(x_i + h, y_i + h\theta_3^i \right), \\ y_{i+1} = y_i + \dfrac{h}{6}\left(\theta_1^i + 2\theta_2^i + 2\theta_3^i + \theta_4^i \right). \end{cases} \tag{8.14}$$

8.4 Линейные многошаговые методы решения задач Коши для обыкновенных дифференциальных уравнений

Будем рассматривать построение решения начальной задачи (8.1) с начальным условием (8.2). Пусть $x_i = x_0 + ih$ – система равноотстоящих узлов. Для вычисления очередного значения $y_{i+1} \approx y(x_{i+1})$ воспользуемся интерполяционным подходом. Проинтегрируем (8.1) в пределах $[x_i, x_{i+1}]$

$$y(x_{i+1}) = y(x_i) + \int\limits_{x_i}^{x_{i+1}} f(x, y(x))dx \tag{8.15}$$

Подстановка интерполяционных многочленов в равенство (8.15) приводит к формулам вычисления очередного значения $y_{i+1} \approx y(x_{i+1})$. В результате применения к полученным интегралам формул Ньютона-Лейбница получаем два семейства методов, называемых многошаговыми методами Адамса.

Экстраполяционные формулы Адамса. Чтобы выполнить указанную подстановку сделаем замену переменой $q = \dfrac{x - x_i}{h}$:

$$\int_{x_i}^{x_{i+1}} P_k(x)dx = h\int_0^1 P_k(x_i + qh)dq,$$

где

$$P_k(x) = P_k(x_i + qh) = fi + q\Delta f_{i-1} + \frac{q(q+1)}{2!}\Delta^2 f_{i-2} + \frac{q(q+1)(q+2)}{3!}\Delta^3 f_{i-3} + .$$

$$.. + \frac{q(q+1)...(q+k-1)}{k!}\Delta^k f_{i-k}.$$

Тогда $y_{i+1} = y_i + h\left(f_i + \dfrac{1}{2}\Delta f_{i-1} + \dfrac{5}{12}\Delta^2 f_{i-2} + \dfrac{3}{8}\Delta^3 f_{i-3} + ...\right).$ (8.16)

При фиксации значений $k = 0,1,2...$ задается степень интерполяционного многочлена и число слагаемых в (8.16).

При $k = 0$ $y_{i+1} = y_i + hf_i,$ (8.17)

При $k = 1$ $y_{i+1} = y_i + \dfrac{h}{2}(3f_i - f_{i-1}),$ (8.18)

При $k = 2$ $y_{i+1} = y_i + \dfrac{h}{12}(23f_i - 16f_{i-1} + 5f_{i-2}).$ (8.19)

Формулы (8.17) – (8.19) определяют методы Адамса-Башфорта первого, второго, третьего порядка точности. В общем случае, для k+1 раз непрерывно дифференцируемой функции шаговая ошибка определяется величиной $O\left(h^{k+1}\right)$. Следовательно, локальная погрешность метода типа будет составлять величину $h\int_0^1 R_k(x_i + qh)dq = O\left(h^{k+2}\right)$, а глобальная – величину $O\left(h^{k+1}\right)$.

Метод Адамса, порождаемый интерполяционной формулой k-ой степени, является методом (k+1)-го порядка точности относительно шага h.

Интерполяционные формулы Адамса. Для вычисления очередного приближения в интеграле

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} \widetilde{P}_k(x)dx,$$

сделаем замену $x = x_{i+1} + qh$ и подставляем в него выражение $\widetilde{P}_k(x)$, определяемого формулой:

$$\widetilde{P}_k(x) = \widetilde{P}_k(x_{i+1} + qh) = f_{i+1} + q\Delta f_i + \frac{q(q+1)}{2!}\Delta^2 f_{i-1} + \frac{q(q+1)(q+2)}{3!}\Delta^3 f_{i-2} + \ldots$$
$$\ldots + \frac{q(q+1)\ldots(q+k-1)}{k!}\Delta^k f_{i-k+1}.$$

Откуда получим интерполяционную формулу Адамса-Моултона

$$y_{i+1} = y_i + h\left(f_{i+1} - \frac{1}{2}\Delta f_i - \frac{1}{12}\Delta^2 f_{i-1} - \frac{1}{24}\Delta^3 f_{i-2} + \ldots\right) \tag{8.20}$$

Частные формулы имеют вид:

при $k = 0 \Rightarrow y_{i+1} = y_i + hf_{i+1}$, (8.21)

при $k = 1 \Rightarrow y_{i+1} = y_i + \frac{h}{2}(f_{i+1} + f_i)$, (8.22)

при $k = 2 \Rightarrow y_{i+1} = y_i + \frac{h}{12}(5f_{i+1} + 8f_i - f_{i-1})$, (8.23)

Уравнению (8.21) соответствует неявный метод Эйлера.

Предикт-корректорные методы Адамса. Под данным методом понимается совместное применение явных и неявных методов одинакового или смежных порядков.

Метод Адамса первого порядка:

$$\begin{cases} \breve{y}_{i+1} = y_i + hf(x_i, y_i), \\ y_{i+1} = y_i + hf(x_{i+1}, \breve{y}_{i+1}). \end{cases} \tag{8.24}$$

Второго порядка:

$$\begin{cases} \breve{y}_{i+1} = y_i + \frac{h}{2}(3f(x_i, y_i) - f(x_{i-1}, y_{i-1})), \\ y_{i+1} = y_i + \frac{h}{2}(f(x_{i+1}, \breve{y}_{i+1}) + f(x_i, y_i)). \end{cases} \tag{8.25}$$

Третьего порядка:

$$\begin{cases} \breve{y}_{i+1} = y_i + \dfrac{h}{12}\left(23f(x_i, y_i) - 16f(x_{i-1}, y_{i-1}) + 5f(x_{i-2}, y_{i-2})\right), \\ y_{i+1} = y_i + \dfrac{h}{12}\left(5f(x_{i+1}, \breve{y}_{i+1}) + 8f(x_i, y_i) - f(x_{i-1}, y_{i-1})\right). \end{cases} \tag{8.26}$$

8.5 Численные примеры. Реализация в пакете Matlab

Пример 1. Найти решение задачи Коши дифференциального уравнения

$$xy' = x \cdot \sin(x) \cdot y, \quad y(1) = 2 \tag{8.27}$$

методом Рунге-Кутты четвертого порядка.

Создадим функцию `func.m`, содержащую значение $f(x, y)$ – правой части дифференциального уравнения, разрешенного относительно производной:

```
function z=func(x,y)
z=sin(x)*y;
```

Создадим файл `RKmeth.m`, возвращающий решение дифференциального уравнения методом Рунге-Кутты четвертого порядка:

```
dx=0.001;
x=1:dx:5;
N=length(x);
y=zeros(N,1);
y(1)=2;
for i=2:N
    teta1=func(x(i-1),y(i-1));
    teta2=func(x(i-1)+dx/2,y(i-1)+teta1/2);
    teta3=func(x(i-1)+dx/2,y(i-1)+teta2/2);
    teta4=func(x(i-1)+dx,y(i-1)+teta3);
    y(i)=y(i-1)+(1/6)*dx*(teta1+2*teta2+2*teta3+teta4);
end
plot(x,y);
grid on
```

Построим график приближенного решения:

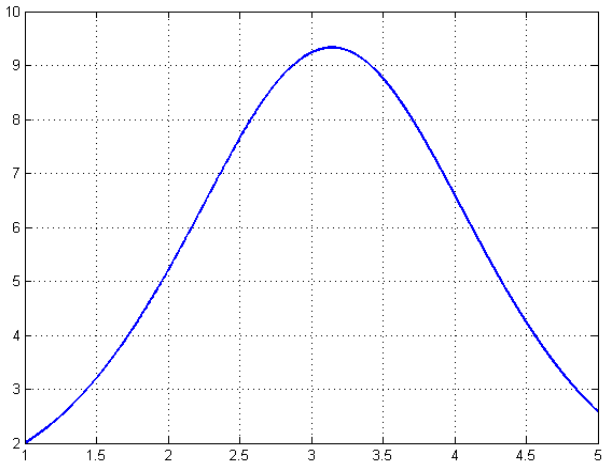


Рис.8.2. Геометрическая интерпретация решения для уравнения (8.27)

Решение задачи Коши для дифференциальных уравнений методом Рунге-Кутты 4 порядка реализовано в пакете Matlab в виде функции **`ode45()`**. Описание других функций, реализующих метод Рунге-Кутты и другие методы, а также решатели систем дифференциальных уравнений можно найти, используя справочную систему Matlab.

Пример 2. Найти решение задачи (8.27), используя встроенные функции пакета Matlab.

Построим решение в Matlab, используя функцию ode45.

```
[X,Y]=ode45('func',[1:dx:5],2);
plot(X,Y); grid on
```

Входными данными для функции `ode45` являются: имя файла, содержащего определение функции, стоящей в правой части уравнения (8.1), вектор, определяющий интервал интегрирования, начальное условие. Результат построения графика функции $y = f(x)$ аналогичен приведенному выше для точного и приближенного решений.

Контрольные вопросы

1. В чем заключается основная особенность метода Пикара?

2. Поясните геометрический смысл метода Эйлера для решения обыкновенных дифференциальных уравнений.

3. Что можно сказать о динамике погрешности в пошаговом методе Эйлера?

4. Как определяется порядок точности метода Рунге-Кутты?

5. В чем состоят принципиальные различия между одношаговыми и многошаговыми методами?

6. Чем отличаются явные и неявные многошаговые методы?

7. Каким образом можно организовать начальный этап работы при использовании многошаговых методов?

Индивидуальные задания

1. Применяя указанные в варианте методы, реализовать алгоритмы численного решения дифференциального уравнения с начальным условием на отрезке $[a,b]$ с произвольно заданным шагом. Для старта многошаговых методов предварительно использовать метод Рунге-Кутты 4-го порядка.

2. Решить предложенную задачу Коши, используя встроенные функции пакета Matlab.

3. Решить задачу, используя метод Пикара.

4. Осуществить графический вывод всех полученных результатов на одни координатные оси.

5. Оформить отчет по лабораторной работе.

Вариант	**Задача Коши**	**Отрезок** $[a,b]$	**Методы**
1	$y' = 2x - 3y,$ $y(0) = 1$	$[0,\ \ 0.5]$	Метод Эйлера, метод Рунге-Кутты, метод Адамса-Башфорта III порядка
2	$xy' = 5x - 2y,$ $y(1) = 1$	$[1,\ \ 1.5]$	Метод Эйлера-Коши, метод Рунге-Кутты II порядка, предикт-корректорная схема метода Адамса I порядка
3	$x^2 y' = 3x - 5y,$ $y(1) = 1$	$[1,\ \ 1.5]$	Метод Эйлера, метод Рунге-Кутты IV порядка, метод Адамса-Башфорта II порядка

4	$y' = 0.2y + \frac{0.8}{x},$ $y(1) = 2$	[1, 1.3]	Метод Эйлера, метод Рунге-Кутты IV порядка, метод Адамса-Моултона II порядка
5	$y' = \frac{0.2}{x^2} - \frac{y}{x} - 0.8y,$ $y(1) = 0.5$	[1, 1.2]	Метод Эйлера-Коши, метод Рунге-Кутты II порядка, предикт-корректорная схема метода Адамса III порядка
6	$y' = 1.6y^2 + \frac{0.4}{x^2},$ $y(1) = 1$	[1, 1.1]	Метод Эйлера, метод Рунге-Кутты IV порядка, предикт-корректорная схема метода Адамса II порядка
7	$y' = \frac{0.9}{x^2} - \frac{y}{x} - 1.6y^2,$ $y(1) = 0.75$	[1, 1.2]	Метод Эйлера, метод Рунге-Кутты IV порядка, метод Адамса-Башфорта III порядка
8	$y' = 0.9y^2 + \frac{1.6}{x^2},$ $y(1) = 1$	[1, 1.4]	Метод Эйлера, метод Рунге-Кутты II порядка, метод Адамса-Моултона III порядка
9	$xy' = xy^2 + 1,$ $y(1) = 1$	[1, 1.3]	Неявный метод Эйлера, метод Рунге-Кутты IV порядка, метод Адамса-Башфорта II порядка
10	$y' = y^2 + \frac{1}{x^2},$ $y(1) = 1$	[1, 1.2]	Метод Эйлера-Коши, метод Рунге-Кутты IV порядка, предикт-корректорная схема метода Адамса I порядка
11	$y' = y^2 + x^2,$ $y(0) = 0$	[0, 1.3]	Метод Эйлера, метод Рунге-Кутты II порядка, метод Адамса-Моултона II порядка
12	$y' = 0.25y^2 + \frac{2}{x^2},$ $y(1) = 1$	[1, 1.5]	Исправленный метод Эйлера, метод Рунге-Кутты II порядка, предикт-корректорная схема метода Адамса II порядка
13	$xy' = 2.2y + 5.2x^2,$ $y(1) = 1$	[1, 1.2]	Метод Эйлера-Коши, метод Рунге-Кутты II порядка, предикт-корректорная схема метода Адамса III порядка
14	$x^2y' = 3.3yx + 1.8x,$ $y(1) = 1.5$	[1, 1.4]	Метод Эйлера, метод Рунге-Кутты II порядка, предикт-корректорная схема метода Адамса III порядка
15	$y' = 10.6\frac{y}{x^2} + \frac{1}{x},$ $y(1) = 3$	[1, 1.35]	Неявный метод Эйлера, метод Рунге-Кутты IV порядка, метод Адамса-Башфорта III порядка
16	$xy' = 0.6x - 5y^2,$ $y(1) = 1.5$	[1, 1.5]	Неявный метод Эйлера, метод Рунге-Кутты II порядка, метод Адамса-Башфорта II порядка

17	$5xy' = \frac{2x^2}{\exp(y)},$ $y(1) = 1$	[1, 1.2]	Исправленный метод Эйлера, метод Рунге-Кутты IM порядка, метод Адамса-Башфорта III порядка
18	$y' = y\sin(x),$ $y(0) = 2$	[0, 5]	Метод Эйлера, метод Рунге-Кутты IV порядка, предикт-корректорная схема метода Адамса II порядка
19	$y' = y \cdot \cos(x),$ $y(0) = 1$	[0, 5]	Метод Эйлера, метод Рунге-Кутты II порядка, метод Адамса-Моултона III порядка
20	$2xy' = 1.8y + y^2,$ $y(1) = 1.5$	[1, 1.2]	Неявный метод Эйлера, метод Рунге-Кутты IV порядка, метод Адамса-Башфорта II порядка

9 МЕТОДЫ ПРИБЛИЖЕННОГО РЕШЕНИЯ КРАЕВЫХ ЗАДАЧ ДЛЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Будем рассматривать двухточечные краевые задачи для ОДУ второго порядка. Такие задачи имеют общий вид:

$$F(x, y, y', y'') = 0, \quad x \in [a, b], \tag{9.1}$$

$$\varphi_1(y(a), y'(a)) = 0, \quad \varphi_2(y(b), y'(b)) = 0,$$

где F, φ_1, φ_2 – заданные функции определенной гладкости.

Наиболее употребительны и хорошо изучены линейные краевые задачи, т.е. задачи вида (9.1), где F, φ_1, φ_2 – линейные функции. Для определенности будем полагать, что объектом изучения является линейная краевая задача:

$$L[y] = y'' + p(x)y' + q(x)y = f(x), \quad x \in [a, b] \tag{9.2}$$

$$l_a[y] = \alpha_0 y(a) + \alpha_1 y'(a) = A, \tag{9.3}$$

$$l_b[y] = \beta_0 y(b) + \beta_1 y'(b) = B, \tag{9.4}$$

где коэффициенты краевых условий не обращаются в ноль одновременно, а функции $p(x)$, $q(x)$ выбраны таким образом, что данная задача имеет единственное решение в заданном функциональном пространстве. Краевые условия (9.3)-(9.4) определяют в общем случае смешанную краевую задачу, при $\alpha_1 = \beta_1 = 0$ первую краевую задачу, при $\alpha_0 = \beta_0 = 0$ – вторую краевую задачу.

Аналитическое решение краевых задач вызывает большие трудности по сравнению с решением задачи Коши. Отсюда и большое разнообразие численных методов решения таких задач. По типу представления результатов решения задачи все методы можно разделить на приближенно-аналитические, дающие решение краевой задачи в виде некоторой функции и численные или сеточные методы, дающие совокупность решений в узлах выбранной сетки.

9.1 Метод конечных разностей

Идея метода конечных разностей решения краевых задач весьма проста и видна из самого названия: вместо производных в дифференциальном уравнении

используются их конечно-разностные аппроксимации. В отличие от методов решения начальных задач, где последовательно, шаг за шагом находится каркас приближенного решения, для краевой дифференциальной задачи возникает необходимость решать алгебраическую систему. В последнем случае обычно стараются удовлетворить двум требованиям: получить хорошее качество аппроксимации и обеспечить эффективное и устойчивое решение получающихся при этом алгебраических систем.

Рассмотрим наиболее типичные конечно-разностные аппроксимации.

Введем на отрезке $[a,b]$ сетку с шагом h: $h = \dfrac{b-a}{n}$:

$$\omega_h = \{x_i | x_i = x_0 + ih;\ \ i = 0,1,...,n;\ \ x_0 = a, x_n = b\}.$$

На этой сетке определим сеточные функции:

$$p_i := p(x_i),\ \ q_i = q(x_i),\ \ f_i = f(x_i), \tag{9.5}$$

отвечающие функциональным коэффициентам дифференциального уравнения (9.2).

Через y_i будем обозначать i-ую компоненту приближенного решения задачи (9.2)-(9.4).

Фиксируя в уравнении (9.2) $x = x_i$ с учетом введенных обозначений (9.5) получим уравнение:

$$y_i'' + p_i y_i' + q_i y_i = f_i, \tag{9.6}$$

где $i = \overline{0,n}$ по числу узлов сетки, y_i'', y_i', y_i – значение решения и его производных в i-ом узле.

В каждом внутреннем узле сетки ω_h значения производных аппроксимируются конечно-разностными отношениями по симметричным формулам второго порядка точности:

$$y_i' = \frac{y_{i+1} - y_{i-1}}{2h} + O(h^2),$$

$$y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + O(h^2).$$

В результате подстановки последних в уравнение (9.6) при $i=\overline{1,n-1}$ и приведения подобных членов получаем стандартное трехточечное разностное уравнение второго порядка:

$$\left(1+\frac{h}{2}p_i\right)y_{i+1}-\left(2-h^2q_i\right)y_i+\left(1-\frac{h}{2}p_i\right)y_{i-1}=h^2f_i, i=1,2...n-1. \qquad (9.7)$$

Выражение (9.7) представляет собой систему (n-1) линейных алгебраических уравнений с трехдиагональной матрицей коэффициентов, в то время как всего неизвестных (n+1): $y_0, y_1, ...y_n$. Два недостающих уравнения системы получим из краевых условий (9.3)-(9.4) данной задачи, для этого воспользуемся несимметричными аппроксимациями первого и второго порядков точности:

$$y'(a)=\frac{y(x_1)-y(x_0)}{h}+O(h)=\frac{-3y(x_0)+4y(x_1)-y(x_2)}{2h}+O\left(h^2\right),$$

$$y'(b)=\frac{y(x_n)-y(x_{n-1})}{h}+O(h)=\frac{y(x_{n-2})-4y(x_{n-1})+3y(x_n)}{2h}+O\left(h^2\right).$$

При аппроксимации производных разностями первого порядка получим два недостающих уравнения для системы (9.7) в виде:

$$\alpha_0y(x_0)+\alpha_1\frac{y(x_1)-y(x_0)}{h}=A \Leftrightarrow (h\alpha_0-\alpha_1)y_0+\alpha_1y_1=Ah, \qquad (9.8)$$

$$\beta_0y(x_n)+\beta_1\frac{y(x_n)-y(x_{n-1})}{h}=B \Leftrightarrow -\beta_1y_{n-1}+(h\beta_0-\beta_1)y_n=Ah. \qquad (9.9)$$

При аппроксимации производных разностями второго порядка уравнения будут иметь вид:

$$\alpha_0y(x_0)+\alpha_1\frac{-3y(x_0)+4y(x_1)-y(x_2)}{2h}=A \Leftrightarrow$$
$$(2h\alpha_0-3\alpha_1)y_0+4\alpha_1y_1-\alpha_1y_2=2Ah, \qquad (9.10)$$

$$\beta_0y(x_n)+\beta_1\frac{3y(x_n)-4y(x_{n-1})+y(x_{n-2})}{2h}=B \Leftrightarrow$$
$$-\beta_1y_{n-2}-4\beta_1y_{n-1}+(2h\beta_0+3\beta_1)y_n=2Bh. \qquad (9.11)$$

Сравнивая два варианта аппроксимации краевых условий можно заметить, что СЛАУ, образованная уравнением (9.8), системой (9.7) и уравнением (9.9) имеет трехдиагональную матрицу коэффициентов и к решению такой сис-

темы можно применить высокоэффективный метод прогонки. Во втором случае соответствующая методу прогонки структура еще должна быть создана, для чего нужно из уравнения (9.10) с помощью уравнения, получающегося из (9.7) при $i=1$, исключить неизвестное y_2, а из (9.11) с помощью уравнения, получающегося из (9.7) при $i=n-1$ исключить y_{n-2}. Усложнения, сопутствующие реализации второго варианта, компенсируются тем, что полученная в итоге система имеет второй порядок точности.

9.2 Метод коллокации

Будем искать приближенное решение линейной краевой задачи (9.2)-(9.4) в виде функции:

$$y_n(x)=\varphi_0(x)+\sum_{i=1}^{n}c_i\cdot\varphi_i(x), \tag{9.12}$$

где определяемые на отрезке $[a,b]$ базисные функции $\varphi_i(x), i=\overline{1,n}$ и дополнительная функция $\varphi_0(x)$ должны быть дважды дифференцируемыми и попарно линейно независимыми. Кроме того, функция $\varphi_0(x)$ должна удовлетворять краевым условиям (9.3)-(9.4), а функции $\varphi_i(x), i=\overline{1,n}$ – соответствующим однородным краевым условиям:

$$\begin{cases}\alpha_0\varphi_i(a)+\alpha_1\varphi_i'(a)=0,\\ \beta_0\varphi_i(b)+\beta_1\varphi_i'(b)=0\end{cases}\qquad \forall i\in\{1,2...n\} \tag{9.13}$$

В таком случае функция $y_n(x)$, определяемая выражением (9.12) при любых значениях коэффициентов c_i удовлетворяет краевым условиям (9.3)-(9.4).

В методе коллокации коэффициенты c_i подбираются так, чтобы в узлах коллокации $x_i: a<x_1<x_2<...<x_n<b$ значения $y_n(x_i)$с приближенного решения были согласованы с точными значениями $y(x_i)$. Согласование в узлах коллокации проводится подстановкой $y_n(x)$ в уравнение (9.2). Тогда имеем равенство:

$$y_n''(x_i)+p(x_i)y_n'(x_i)+q(x_i)y_n(x_i)=f(x_i). \tag{9.14}$$

Продифференцировав дважды функцию $y_n(x)$ в представлении (9.12) и подставив результат в (9.14), получим:

$$\sum_{j=1}^{n} c_j \varphi_j''(x_i) + p_i \sum_{j=1}^{n} c_j \varphi_j'(x_i) + q_i \sum_{j=1}^{n} c_j \varphi_j(x_i) =$$
$$= f_i - \varphi_0''(x_i) - p_i \varphi_0'(x_i) + q_i \varphi_0'(x_i),$$

где p_i, q_i, f_i соответствуют обозначениям сеточных функций. Положим

$$a_{ij} = \varphi_j''(x_i) + p_i \varphi_j'(x_i) + q_i \varphi_j'(x_i),$$

$$b_i = f_i - \varphi_0''(x_i) - p_i \varphi_0'(x_i) + q_i \varphi_0'(x_i).$$

Тогда (33) приобретает стандартный вид линейной алгебраической системы:

$$\sum_{j=1}^{b} a_{ij} c_j = b_i, i = \overline{1,n}$$

Решив эту систему каким-либо стандартным способом относительно c_i и подставив найденные значения в (9.12), получаем приближенное решение $y_n(x)$.

Успех применения метода коллокации (как и других аналитико-приближенных методов) сильно зависит от удачного выбора базисных функций. Чаще всего выбор базисных функций опирается на априорные или эмпирические сведения о решении. В отсутствие таковых, можно предложить набор базисных функций вида:

$$\varphi_0(x) = \delta + \gamma x,$$

коэффициенты которой подбирают таким образом, чтобы она удовлетворяла неоднородным краевым условиям (9.3)-(9.4):

$$\begin{cases} \alpha_0 \delta + (\alpha_0 a + \alpha_1)\gamma = A, \\ \beta_0 \delta + (\beta_0 b + \beta_1)\gamma = B. \end{cases}$$

Функции $\varphi_i(x),\ i = \overline{1,n}$ в случае $\alpha_1 = 0$ можно взять однопараметрическими вида:

$$\varphi_i(x) = \gamma_i (x-a)^i + (x-a)^{i+1} \qquad (9.15)$$

или в общем случае

$$\varphi_i(x)=\gamma_i(x-a)^{i+1}+(x-a)^{i+2}. \qquad (9.16)$$

Очевидно, что при любых γ_i удовлетворяют первому из равенств (9.13), а при фиксированном γ_i и второму.

Для (9.15) $\gamma_i=-\dfrac{\beta_0(b-a)^2+(i+1)\beta_1(b-a)}{\beta_0(b-a)+i\beta_1}$.

Для (9.16) $\gamma_i=-\dfrac{\beta_0(b-a)^2+(i+2)\beta_1(b-a)}{\beta_0(b-a)+(i+1)\beta_1}$.

Следовательно, можно рассчитывать, что с такими базисными функциями при найденных методом коллокации коэффициентах c_i определенная посредством (9.12) функция $y_n(x)$ будет удовлетворять краевым условиям и может служить приближенным решением данной краевой задачи.

Проблема формального выбора базисных функций упрощается в случае, когда в задаче (9.2)-(9.4) фигурируют однородные краевые условия:

$y(a)=0,\ \ y(b)=0$.

В этом случае $\varphi_0(x)=0$, а в роли $\varphi_i(x),\ \ i=\overline{1,n}$ могут выступать, например, функции:

$\varphi_i(x)=(x-a)^i(b-x)$ или $\varphi_i(x)=\sin\dfrac{i(x-a)}{b-a}\pi$.

К этому случаю легко свести более общий случай неоднородных краевых условий первого рода:

$y(a)=A, y(b)=B$.

С этой целью достаточно сделать линейную замену: $y=u+v$, где

$$v=A+\frac{B-A}{b-a}(x-a).$$

Дважды дифференцируя эту функцию и подставляя результат в уравнение (9.2) приходим к краевой задаче с однородными краевыми условиями относительно переменной *u*.

$$u''+p(x)u'+q(x)u=f(x)-\frac{B-A}{b-a}p(x)-vq(x),\ \ u(a)=0,\ \ u(b)=0,\ \ x\in[a,b].$$

9.3 Метод Галеркина

Метод Галеркина относится к группе проекционных методов. Пусть L – некоторый линейный оператор, действующий в гильбертовом пространстве H, т.е. в полном нормированном пространстве со скалярным произведением. Ставится задача приближенного решения операторного уравнения:

$$Ly = f, \tag{9.17}$$

т.е. задача отыскания приближения к неизвестному элементу $y \in H$, соответствующему заданному элементу $f \in H$. Пусть $\{\varphi_i\}_{i=1}^{\infty}$ – некоторая полная замкнутая система линейно независимых элементов из H. Ее n первых элементов $\varphi_1,...\varphi_n$ выделяют в H конечномерное подпространство H_n, в котором и ищется приближенное решение уравнения (9.17):

$$y_n = \sum_{i=1}^{n} c_i \varphi_i . \tag{9.18}$$

Тогда к равенству $f = Ly_n + (f - Ly_n)$ можно применить теории гильбертовых пространств: Любой элемент гильбертова пространства может быть представлен в виде суммы определенного элемента подпространства (проекции данного элемента на подпространство) и определенного элемента пространства, ортогонального к выбранному подпространству. Принадлежность элемента $f - Ly_n$ ортогональному к H_n подпространству ${H_n}^{\perp}$ означает, что он ортогонален каждому элементу φ_i, входящему в базис пространства H_n. Таким образом, при $i = \overline{1,n}$ имеем:

$$(f - Ly_n) \perp \varphi_i \Leftrightarrow (f - Ly_n, \varphi_i) = 0 \Leftrightarrow (Ly_n, \varphi_i) = (f, \varphi_i).$$

Подставляя сюда выражение (9.18) и пользуясь свойствами скалярного произведения, получаем:

$$\left(L\sum_{j=1}^{n} c_j \varphi_j, \varphi_i \right) = (f, \varphi_i) \Leftrightarrow \sum_{j=1}^{n} (L\varphi_j, \varphi_i) c_j = (f, \varphi_i) \tag{9.19}$$

Метод Галеркина приближенного решения операторного уравнения сводится к нахождению коэффициентов $c_1,...c_n$ линейной комбинации некоторых,

задаваемых определенным образом линейно независимых функций, называемых координатными $\varphi_1,...\varphi_n$ так, чтобы эти коэффициенты удовлетворяли линейной системе:

$$\sum_{j=1}^{n} a_{ij} c_j = d_i, \quad i = \overline{1,n}, \tag{9.20}$$

где $a_{ij} = (L\varphi_j, \varphi_i), d_i = (f, \varphi_i)$.

Рассмотрим решение краевой задачи (9.2)-(9.4). Введем в рассмотрение также гильбертово пространство $L_2[a,b]$ функций, интегрируемых на отрезке $[a,b]$. Скалярное произведение определяется равенством:

$$(u,v) = \int_a^b u(x), v(x) dx .$$

Будем искать решение в виде приближающей функции (9.12), которая должна удовлетворять краевым условиям:

$$l_a[\varphi_0] = A, l_b[\varphi_0] = B, l_a[\varphi_i] = 0, l_b[\varphi_i] = 0, \forall i \in \overline{1,n} .$$

Подставляя $y_n(x)$ в уравнение (9.2):

$$L\left[\varphi_0 + \sum_{i=1}^{n} c_i \varphi_i \right] = f \Leftrightarrow L\left[\sum_{i=1}^{n} c_i \varphi_i \right] = f - L[\varphi_0].$$

Такое операторное уравнение имеет вид:

$$\left(L \sum_{j=1}^{n} c_j \varphi_j, \varphi_i \right) = (f - L\varphi_0, \varphi_i).$$

Поэтому и выражение, определяющее d_i изменится: $d_i = (f - L[\varphi_0], \varphi_i)$.

Система линейных алгебраических уравнений, определяющих коэффициенты $c_1,...c_n$ будет иметь общий вид:

$$\sum_{j=1}^{n} c_j \int_a^b \varphi_i(x) L[\varphi_j] dx = \int_a^b [f(x) - L[\varphi_0]] \varphi_i(x) dx, \quad i = \overline{1,n} .$$

Согласно определению скалярного произведения найдем коэффициенты d_i:

$$d_i = (f - L[\varphi_0], \varphi_i) = \int_a^b [f(x) - \varphi_0''(x) - p(x)\varphi_0'(x) - q(x)\varphi_0(x)]\varphi_i(x)dx.$$

Также определим коэффициенты a_{ij} СЛАУ (9.20), используя определение скалярного произведения:

$$a_{ij} = (L\varphi_j, \varphi_i) = \varphi_i(b)\varphi_j'(b) - \varphi_i(a)\varphi_j'(a) - \int_a^b \varphi_j'(x)\varphi_i'(x)dx +$$

$$+ \int_a^b p(x)\varphi_j'(x)\varphi_i(x)dx + \int_a^b q(x)\varphi_j(x)\varphi_i(x)dx.$$

9.4 Схема решения граничных задач в Matlab

Рассмотрим решение граничных задач на примере обыкновенного дифференциального уравнения второго порядка (9.2), разрешенного относительно производной второго порядка. Требуется найти функцию *y*(*x*), удовлетворяющую на отрезке [*a*, *b*] дифференциальному уравнению

$$y'' = f(x, y, y')$$

и граничным условиям

$$\alpha \cdot y(a) + \beta \cdot y'(a) = A,$$

$$\gamma \cdot y(b) + \delta \cdot y'(b) = B.$$

Здесь α, β, γ, δ, *A*, *B* — заданные числа.

Решение этой задачи состоит из следующих этапов.

1. Преобразование дифференциального уравнения второго порядка к системе двух уравнений первого порядка.

2. Написание функции для вычисления правой части системы.

3. Написание функции, определяющей граничные условия.

4. Формирование начального приближения при помощи специальной функции **bvpinit**.

5. Вызов солвера **bvp4c** для решения граничной задачи.

6. Визуализация результата.

Первые два этапа выполняются практически так же, как и при решении задачи Коши. Введение вспомогательных функций $y_1(x)$ и $y_2(x)$ приводит к системе уравнений первого порядка относительно них:

$$\begin{cases} y_1' = y_2 \\ y_2' = f(x, y_1, y_2) \end{cases}.$$

Функция правой части системы зависит от x и вектора y, состоящего из двух компонент, $y(1)$ соответствует $y_1(x)$, а $y(2) - y_2(x)$, и программируется так же, как при решении задачи Коши.

Синтаксис ППП Matlab требует, чтобы при постановка граничных условий была преобразована так, чтобы в правых частях стояли нули:

$\alpha \cdot y(a) + \beta \cdot y'(a) - A = 0$, $\gamma \cdot y(b) + \delta \cdot y'(b) - B = 0$.

Функция, описывающая граничные условия, зависит от двух аргументов – векторов *ya* и *yb* и имеет следующую структуру:

```
function z=bound(ya,yb)
z=[a*ya(1)+b*ya(2)-A;g*yb(1)+d*yb(2)-B];
```

Вместо *a*, *b*, *g*, *d*, *A* и *B* следует подставить заданные числа.

Солвер **bvp4c** основан на методе конечных разностей, т.е. получающееся решение есть векторы значений неизвестных функций в точках отрезка (в узлах сетки). Выбор начальной сетки и приближения может оказать влияние на решение, выдаваемое солвером **bvp4c**. Для задания начальной сетки и приближения предназначена функция **bvpinit**, обращение к которой в самом простом случае выглядит следующим образом:

initsol = bvpinit (начальная сетка, начальное приближение к решению).

Начальная сетка определяется вектором координат узлов, упорядоченных по возрастанию и принадлежащих отрезку [*a, b*]. Если имеется априорная информация о решении, то разумно среди точек начальной сетки указать те, в которых решение сильно изменяется. Формирование равномерной сетки целесообразно производить функцией **linspace**:

X = linspace(a, b, n),

возвращающей вектор x из n равноотстоящих узлов между a и b, включая границы. Заданная сетка модифицируется солвером в процессе решения для обеспечения требуемой точности. Постоянное начальное приближение задается вектором из двух элементов для функций $y_1(x)$ и $y_2(x)$. Начальное приближение может зависеть от x, в этом случае требуется запрограммировать функцию, которая по заданному x возвращает вектор из двух компонент со значениями $y_1(x), y_2(x)$, и поместить указатель на нее во втором входном аргументе **bvpinit.** В результате работы **bvpinit** генерируется структура **initsol** с информацией о начальном приближении.

После определения начального приближения вызывается солвер **bvp4c**, входными аргументами которого являются указатели на функции правой части системы и граничных условий, начальное приближение и, при необходимости, управляющая структура с параметрами вычислительного процесса. Управляющая структура формируется при помощи функции **bvpset**. Солвер **bvp4c** возвращает единственный выходной аргумент – структуру с информацией о расчетной сетке, значения искомой функции и ее производной.

Для вычисления значений приближенного решения в произвольных точках отрезка следует применить функцию **deval**.

9.5 Численные примеры. Реализация в пакете Matlab

Требуется решить граничную задачу для обыкновенного дифференциального уравнения второго порядка

$$y'' = -\left(x^2 - 1\right)\sin x,\ y(0) = 0,\ \ y'(6) + 2y(6) = 1. \qquad (9.21)$$

при помощи солвера **bvp4c**.

Система дифференциальных уравнений первого порядка, соответствующая исходному уравнению, и граничные условия для нее:

$$\begin{cases} y_1' = y_2, \\ y_2' = -\sin x\left(x^2 - 1\right), \end{cases} \quad y_1(0) = 0,\ \ y_2(6) + 2y_1(6) - 1 = 0. \qquad (9.22)$$

Как было указано ранее, функция для граничных условий имеет два входных аргумента. Каждый аргумент является вектором значений неизвест-

ных функций $y_1(x), y_2(x)$ в начальной и конечной точке промежутка. Поэтому пользовательская функция граничных условий должна определяться аналогично функции bound:

```
function z=bound(ya,yb)
z=[ya(1);yb(1)+2*yb(2)-1];
```

Кроме того, определим вектор-функцию правой части системы дифференциальных уравнений (9.22):

```
function f=vec_rside(x,y)
f=[y(2); -sin(x).*(x.^2-1)];
```

Решение граничной задачи можно оформить в виде скрипт-файла, в котором необходимо:

1) при помощи **bvpinit** задать начальную сетку на отрезке [0, 6], например, из 10 узлов, и постоянное начальное приближение: $y_1(x)=1, y_2(x)=0$;

2) вызвать солвер **bvp4c** и получить приближенное решение;

3) отобразить графически приближенное решение, извлекая нужные компоненты из структуры, возвращаемой солвером.

Текст скрипта boundary_problemODE приведен далее:

```
%формирование сетки и задание начального приближения для неизвестной функции
meshinit=linspace(0,6,30);
yinit=[1 0];
initsol=bvpinit(meshinit,yinit);
%вызов солвера
sol=bvp4c('vec_rside','bound',initsol); %поля структуры sol:
% sol.x - содержит координаты сетки
% sol.y - матрица, влючающая векторы:
% sol.y(1,:) - значения функции y1
% sol.y(2,:) - значения функции y2
% вывод графика функции
plot(sol.x,sol.y(1,:),'r-'); grid on
```

Результат показан на рисунке 9.1.

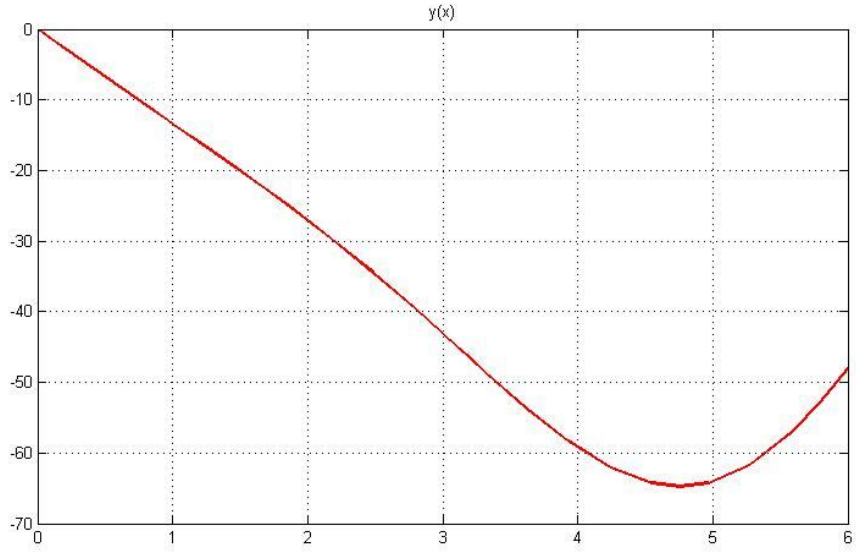


Рис. 9.1. Геометрическая интерпретация решения для уравнения (9.22)

Контрольные вопросы

1. В чем заключается суть метода конечных разностей?

2. В чем состоит особенность подбора коэффициентов c_i в методе коллокации?

3. Почему метод Галеркина относится к группе проекционных методов?

Индивидуальные задания

1. Применяя указанные методы, реализовать алгоритм численного решения дифференциального уравнения с краевыми условиями на отрезке $[a,b]$.

Задание 1. Используя метод конечных разностей, составить решение краевой задачи для обыкновенного дифференциального уравнения с точностью $\varepsilon = 10^{-3}$ с выбранным шагом.

Задание 2. Используя метод коллокации и метод Галеркина, найти решения дифференциальных уравнений с граничными условиями.

2. Решить предложенные задачи, используя встроенные функции пакета Matlab. Вывести графические представления решений на одни оси (для каждого задания).

3. Оформить отчет по лабораторной работе.

Номер Вариант	Задание 1	Задание 2
1	$y'' + \frac{y'}{x} + 2y = x,$ $y(0.7) = 0.5,$ $2y(1) + 3y'(1) = 1.2$	$y'' - 2xy' + 2y = x,$ $y(0) = 0, \quad y'(1) = 1$
2	$y'' + y' + y = x + 1,$ $y'(0.5) = 1.2,$ $y(0.8) - 2y'(0.8) = 1$	$y'' + y = \sin x,$ $y(0) = 0, \quad y\left(\frac{\pi}{2}\right) = 1$
3	$y'' - xy' + 2y = x + 1,$ $y(0.9) - 0.5y'(0.9) = 2,$ $y(1.2) = 1$	$y'' + 4y' + 4y = 8,$ $y(-1) = 0, \quad y(1) = 0$
4	$y'' - 2y' - \frac{y}{x} = 3,$ $y(0.2) = 2,$ $0.5y(0.5) - y'(0.5) = 1$	$y'' + 3y' + y = 1,$ $y(0) = 0, \ y(1) = 1$
5	$y'' + xy' + y = x + 1,$ $y(0.5) - 2y'(0.5) = 1,$ $y'(0.8) = 1.2$	$y'' - 2xy' + 2y = x^2 - 1,$ $y(0) = 0, \ y(1) = 2$
6	$y'' - 0.5y' + 3y = 2x^2,$ $y(1) + 2y'(1) = 0.6,$ $y(2) = 0$	$y'' - 3xy' + 2y = x + 1,$ $y(0) = 1, \quad y(1) = 0$
7	$y'' + 1.5y' - xy = 0.5,$ $2y(1.3) - y'(1.3) = 1,$ $y(1.6) = 3$	$y'' + 2xy' + 2y = x - 1,$ $y(0) = 0, \quad y(1) = 1$
8	$y'' + 2xy' - y = 0.4,$ $2y(0.3) + y'(0.3) = 1,$ $y'(0.6) = 2$	$y'' + xy' + y = 2x,$ $y(0) = 0, \quad y(1) = 0$
9	$y'' - 0.5y' + y = 2,$ $y(0.4) = 1.2,$ $y(0.7) + 2y'(0.7) = 1.4$	$y'' + (1 + x^2)y' = 1,$ $y(-1) = 0, \quad y(1) = 1$
10	$y'' + \frac{2}{x}y' - 3y = 2,$ $y'(0.8)) = 1.5,$ $2y(1.1) + y'(1.1) = 3$	$y'' + \frac{1}{x^2}y' + y = 0,$ $y(1) = 1, \quad y(2) = 1$

11	$y'' + 2y' - 3y = x^2 + 1,$ $y(0.5) = 0,$ $y(1) + y'(1) = 4$	$4y'' - xy' + y = \exp(x),$ $y(0) = 0, \quad y(1) = 5$
12	$y'' - y' + 2y = x - 5,$ $y'(0.2) = 1,$ $5y(1) + 3y'(1) = 0$	$y'' + 2y = \cos x,$ $y(0) = 0, \quad y\left(\frac{3\pi}{2}\right) = 1$
13	$y'' + xy' + 3y = x^3 + 8,$ $y(0.5) + y'(0.5) = 1,$ $y(1) = 5$	$y'' - 5y' + 6y = 1,$ $y(-2) = 0, \quad y(2) = 0$
14	$y'' + 3y' - \frac{y}{x^2} = x,$ $y(1) = 3,$ $0.5y(2) + y'(2) = 1$	$y'' - 4y' + y = 2,$ $y(0) = 0, \ y(1) = 2$
15	$y'' - x^2y' + y = x^2 - 4,$ $2y(0.5) + y'(0.5) = 1.5,$ $y'(1.5) = 2.5$	$y'' + xy' + 2y = x + 5,$ $y(0) = 0, \ y(1) = 2$
16	$y'' - xy' + 3xy = \exp(x),$ $y(1) - 4y'(1) = 1,$ $y(2) = 0$	$y'' + xy' + 5y = x^2 + 1,$ $y(0) = 1, \quad y(1) = 0$
17	$y'' + 5y' - x^2y = 0.5,$ $y(1) + 9y'(1) = 1,$ $y(1.5) = 5$	$y'' - xy' + y = (x-1)\sin x,$ $y(0) = 0, \quad y(1) = 1$
18	$y'' + x^2y' - xy = 1,$ $2y(0.2) - y'(0.2) = 1,$ $y'(1) = 2$	$y'' + x^2y' + 3xy = 2x,$ $y(-1) = 0, \quad y(1) = 0$
19	$y'' - 2xy' + y = 2x - 1,$ $y(0.3) = 1.5,$ $y(0.9) + 2y'(0.9) = 1$	$y'' + (1 + x^2)y' = \cos x,$ $y(-1) = 0, \quad y(1) = 1$
20	$y'' - xy' + 2y = 2,$ $y'(0)) = 1,$ $2y(1.3) - y'(1.3) = 5$	$y'' + y' + \frac{1}{x}y = 0,$ $y(1) = 1, \quad y(2) = 0$

10 ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ КРАЕВЫХ ЗАДАЧ ДЛЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ЧАСТНЫМИ ПРОИЗВОДНЫМИ

Будем рассматривать приближенные методы решения некоторых задач для дифференциальных уравнений с частными производными второго порядка с двумя независимыми переменными, которые в общем случае имеют вид:

$$F\left(x, y, U, U_x, U_y, U_{xx}, U_{xy}, U_{yy}\right) = 0. \qquad (10.1)$$

Решением уравнения (10.1) будем считать функцию $U(x, y)$, график решения представляет собой поверхность в пространстве $Oxyu$.

Уравнение (10.1) называется линейным, если оно первой степени относительно искомой функции и всех ее производных и не содержит их произведений:

$$A\frac{\partial^2 U}{\partial x^2} + 2B\frac{\partial^2 U}{\partial x \partial y} + C\frac{\partial^2 U}{\partial y^2} + a\frac{\partial U}{\partial x} + b\frac{\partial U}{\partial y} + cU = F(x, y). \qquad (10.2)$$

Если коэффициенты A, B, C, a, b не зависят от x, y, то уравнение (10.2) называют линейным дифференциальным уравнением с постоянными коэффициентами.

Рассмотрим классификацию уравнений с частными производными. Пусть $D = AC - B^2$ – дискриминант уравнения (10.2):

$D > 0$ – эллиптическое уравнение,

$D = 0$ – параболическое уравнение,

$D < 0$ – гиперболическое уравнение.

Простейшее уравнение эллиптического типа $\Delta U = 0$ – уравнение Лапласа, $\Delta U = f(x, y)$ – уравнение Пуассона.

10.1 Краевые задачи для уравнений эллиптического типа

Исследование стационарных процессов различной физической природы (колебания, теплопроводность и др.) часто приводят к уравнениям эллиптического типа

$$L[U] = \Delta U + aU_x + bU_y + cU = F(x,y), \qquad (10.3)$$

где $a = a(x,y)$, $b = b(x,y)$, $c = c(x,y)$, $F = F(x,y)$ – непрерывные функции.

Для этих уравнений обычно ставятся только краевые задачи, так как задача Коши может быть некорректной.

Рассмотрим наиболее часто встречающиеся краевые задачи.

Первая краевая задача. На контуре Г, ограничивающем область G задана непрерывная функция $\varphi(P) = \varphi(x,y)$. Требуется найти функцию $U(x,y)$, удовлетворяющую внутри области G уравнению (10.3) и принимающую на границе заданные значения:

$$L[U(P)] = F(P), \quad P \in G; \quad U(P) = \varphi(P), \quad P \in \Gamma.$$

Вторая краевая задача. На контуре Г, ограничивающем область G, задана непрерывная функция $\varphi_1(P)$. Требуется найти такую функцию $U(x,y)$, удовлетворяющую внутри G уравнению (10.3), нормальная производная которой на Г принимает заданные значения $\varphi_1(P)$:

$$L[U(P)] = F(P), \quad P \in G; \quad \frac{\partial U(P)}{\partial n} = \varphi_1(P), \quad P \in \Gamma.$$

Третья краевая задача. На контуре Г, ограничивающем область G, задана непрерывная функция $\psi(P) = \psi(x,y)$. Требуется найти такую функцию $U(x,y)$, удовлетворяющую внутри G уравнению (10.3), чтобы:

$$L[U(P)] = F(P), \quad P \in G;$$

$$\alpha_0 U(P) + \alpha_1 \frac{\partial U(P)}{\partial n} = \psi(P), \quad P \in \Gamma, \quad |\alpha_0| + |\alpha_1| \neq 0.$$

Если область ограничена, то задачу называют внутренней, иначе – внешней. Для уравнения Лапласа первая краевая задача называется задачей Дирихле, вторая – Неймана и третья – смешанной краевой задачей.

10.2 Метод конечных разностей. Конечно-разностные аппроксимации производных

Метод конечных разностей или метод сеток является одним из самых распространенных в настоящее время методов численного решения уравнений с частными производными. В его основе лежит идея замены непрерывного ре-

шения набором дискретных значений в узлах сеточной области. При этом производные в исходном дифференциальном уравнении заменяются конечно-разностными соотношениями.

Ограничимся рассмотрением случая двух независимых переменных. Построим множество равноотстоящих точек сетки $x_i = x_0 + ih,\ 0 \le i \le N$ и $y_j = y_0 + jk,\ 0 \le j \le M$. После этого область, на которой ищется решение, покрывается прямоугольной конечно-разностной сеткой: через каждую точку x_i проводятся прямые, параллельные *OY*, а через каждую y_j – прямые, параллельные оси *OX*.

Проводя аналогию с одномерным случаем, можно получить конечно-разностные аппроксимации производных, используя формулу Тейлора для функции двух переменных. Аппроксимация первых производных имеет вид:

$$\frac{\partial U}{\partial x}\left(x_i, y_j\right) \approx \frac{U_{i+1,j} - U_{i,j}}{h} + O(h) \approx \frac{U_{i,j} - U_{i-1,j}}{h} + O(h) \approx \frac{U_{i+1,j} - U_{i-1,j}}{2h} + O\left(h^2\right),$$

аналогично для $\frac{\partial U}{\partial y}\left(x_i, y_j\right)$. Аппроксимации вторых производных:

$$\frac{\partial^2 U}{\partial x^2}\left(x_i, y_j\right) \approx \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2} + O\left(h^2\right), \tag{10.4}$$

аналогично для $\frac{\partial^2 U}{\partial y^2}\left(x_i, y_j\right) \approx \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{k^2} + O\left(k^2\right).$ (10.5)

Тогда погрешность аппроксимации оператора Лапласа:

$$\left|\omega(x,y)\right| \le M\,\frac{h^2 + k^2}{12},\quad M = \max\left(\left|\frac{\partial^4 U}{\partial x^4}\right|, \left|\frac{\partial^4 U}{\partial y^4}\right|\right).$$

10.3 Уравнения Лапласа и Пуассона в конечных разностях

Получим конечно-разностное уравнение, соответствующее уравнению Пуассона:

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = -f(x,y). \tag{10.6}$$

При этом пользуются различными схемами. *Первая схема – пятиточечный шаблон, схема «крест»* (рисунок 10.1).

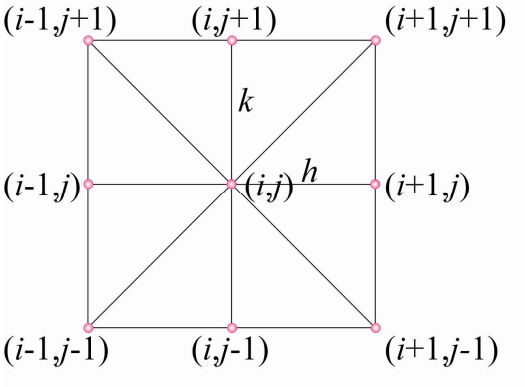


Рис. 10.1. Пятиточечный конечно-разностный шаблон.

Строим аппроксимацию для узла (i, j), используя полученные соотношения (10.4) и (10.5):

$$\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{k^2} = -f(x_i, y_j)$$

Уравнению Лапласа приближенно соответствует уравнение в конечных разностях (значение функции в каждом узле определяется как среднее арифметическое значений в остальных узлах) для случая квадратной сетки:

$$U(x_i, y_j) = \frac{1}{4}\left[U(x_{i-1}, y_j) + U(x_{i+1}, y_j) + U(x_i, y_{j+1}) + U(x_i, y_{j-1})\right]. \qquad (10.7)$$

Вторая схема – пятиточечный шаблон, схема модифицированный «крест». Приведем аппроксимацию по этой схеме для уравнения Лапласа:

$$U(x_i, y_j) = \frac{1}{4}\left[U(x_{i-1}, y_{j-1}) + U(x_{i-1}, y_{j+1}) + U(x_{i+1}, y_{j+1}) + U(x_{i+1}, y_{j-1})\right].$$

10.4 Решение краевых задач для уравнений эллиптического типа методом сеток

Идея метода сеток для приближенного решения краевых задач для двумерных дифференциальных уравнений заключается в следующем:

1. В плоской области G, в которой разыскивается решение, строится сеточная область G_h, состоящая из одинаковых ячеек и приближающая данную область G (рисунок 10.2).

2. Заданное дифференциальное уравнение в узлах построенной сетки заменяется соответствующим конечно-разностным уравнением.

3. На основании граничных условий устанавливаются значения искомого решения в граничных узлах области Γ_h (либо связываются системой соотношений с внутренними узлами области G).

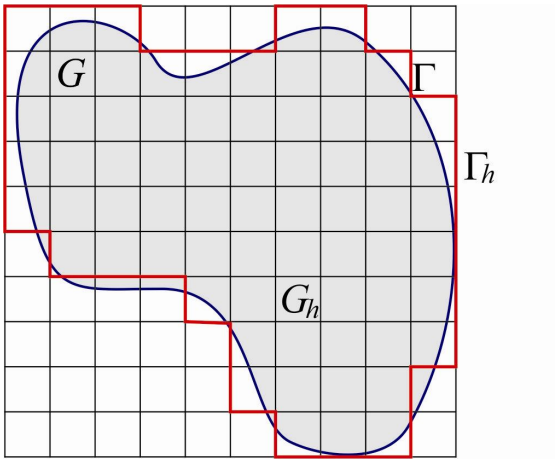


Рис. 10.2. Покрытие расчетной области сеткой.

Решив полученную систему конечно-разностное уравнений, находим значения искомой функции в узлах сеточной области. Покажем построение решения методом сеток задачи Дирихле для уравнения Лапласа:

$$\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial y^2}=0,\ \ (x,y)\in G,\ \ U(P)=\varphi(P),\ \ P\in\Gamma,$$

где $\varphi(P)=\varphi(x,y)$ – заданная непрерывная функция.

Для простоты рассмотрим случай квадратной сетки. Будем считать, что область ограничена простым кусочно-гладким контуром Г. Выбрав шаг h, построим квадратную сетку: $x_i=x_0+ih, y_j=y_0+jh,\ \ i,j=0,\pm1,\pm2,\ldots.$ с таким расчетом, чтобы узлы (x_i,y_j) сетки S_h или принадлежали области G или отстояли от границы Г на расстоянии меньшем, чем h.

Точки (узлы) сетки S_h называются соседними, если они удалены на расстояние, равное шагу h. Узел, принадлежащий внутренней части области G, называется внутренним (узлы типа 1 на рисунке 10.3), узел, примыкающий к границе – граничным (узлы типа 2 и 3). При этом, если граничный узел имеет со-

седний внутренний узел, он называется граничным узлом первого рода (узел типа 1), в противном случае граничный узел называется граничным узлом второго рода (узел типа 3). Внутренние и граничные узлы первого рода называют расчетными точками. Граничные узлы второго рода не входят в вычисление и могут быть изъяты из сетки. Будем считать, что множество S_h является связным.

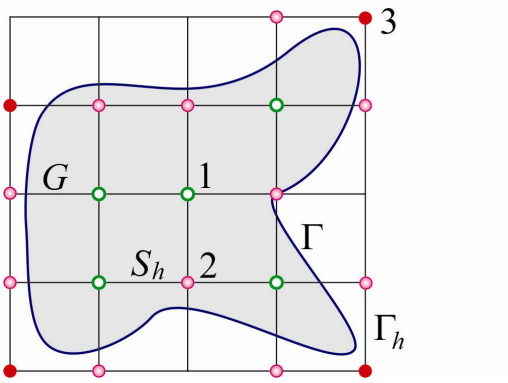


Рис. 10.3. Покрытие расчетной области сеткой (с указанием типа узлов).

Значение искомой функции в точках (x_i, y_j) обозначим U_{ij}. Следуя общей схеме (10.7), получим разностные уравнения. В граничных узлах первого рода сетки полагаем:

$$U(B) = \varphi(B), \qquad (10.8)$$

где B – ближайшая точка границы Г.

Система (10.6)-(10.7) является неоднородной линейной системой, причем число неизвестных (внутренних узлов) равно числу уравнений. Система (10.7)-(10.8) всегда совместна и имеет единственное решение. Чтобы доказать это достаточно убедиться в том, что соответствующая однородная система имеет единственное нулевое решение. Одним из эффективных методов решения подобных систем с трехдиагональной матрицей коэффициентов является метод прогонки. Применение численных процедур позволяет получить решение в узлах сеточной области.

При решении задачи Неймана или смешанной краевой задачи потребуется аппроксимация производных искомой функции и в краевых условиях, что

даст дополнительные уравнения, связывающие все узлы вместе с (10.7) в единую систему алгебраических уравнений.

Отдельно можно выделить группу итерационных методов, которые также можно эффективно использовать для решения уравнений Лапласа и Пуассона. Рассмотрим идею простейшего из итерационных методов – *метода простой релаксации*. Используя конечно-разностную аппроксимацию типа (10.7) для двумерного уравнения Пуассона, перепишем его конечно-разностный аналог в следующей постановке, удовлетворяющей виду итерационного процесса (при $h = k$ для всех внутренних узлов расчетной области):

$$U_{i,j} = U_{i,j} + \frac{\omega}{4}\left(\left[U_{i-1,j} + U_{i+1,j} - 4U_{i,j} + U_{i,j-1} + U_{i,j+1}\right] - f_{i,j}h^2\right)$$

или $U_{i,j} = U_{i,j} + \omega r_{i,j}$,

где ω – параметр релаксации, $1 \le \omega < 2$. Итерации выполняются до тех пор, пока норма невязки не будет находится в пределах заданной точности: $\|r\| < \varepsilon$.

Погрешность приближенного решения, полученного разностным методом, складывается из трех погрешностей: погрешности аппроксимации исходного уравнения, погрешности аппроксимации краевых условий, погрешности, получаемой в результате приближенного решения полученной системы разностных уравнений.

10.5 Решение краевых задач для криволинейных областей

Если граница области Г криволинейна помимо изложенного подхода, состоящего в «грубой» аппроксимации гладкой границы кусочным контуром Γ_h, существуют несколько вычислительных подходов, позволяющих улучшить аппроксимацию в указанных узлах области. Один из таких подходов заключается в модернизации конечно-разностных выражений для узлов, близлежащих к границе. Предположим, что для некоторой задачи граничная кривая пересекает прямоугольную конечно-разностную сетку в точках P и V, как показано на рисунке 10.4. Введем обозначения $AB = AC = h$, $AD = AE = k$, пусть

$AP = \mu x,\ \ AV = \lambda y,\ 0 < \lambda, \mu < 1$. Используя формулу Тейлора получаем аппроксимации: $\left.\dfrac{\partial U}{\partial x}\right|_A \approx \dfrac{\mu^2 U|_B - U|_P - (\mu^2 - 1)U|_A}{\mu(\mu - 1)h},\ O(h^2),$

$$\left.\frac{\partial^2 U}{\partial x^2}\right|_A \approx 2\frac{\mu U|_B + U|_P - (\mu + 1)U|_A}{\mu(\mu + 1)h^2},\ O(h).$$

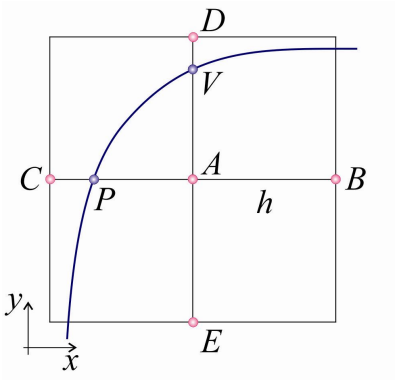


Рис. 10.4. Покрытие криволинейной области сеткой.

Таким образом, аппроксимация второй производной в окрестности границы хуже, чем во внутренних узлах сетки. Аналогичным образом получаются аппроксимации для $\left.\dfrac{\partial U}{\partial y}\right|_A$ и $\left.\dfrac{\partial^2 U}{\partial y^2}\right|_A$.

Второй подход связан с интерполяцией на граничных узлах и традиционно используется при решением краевой задачи Дирихле с помощью процесса Либмана. Процесс Либмана стоится как итерационный с одновременным исправлением граничных значений. Выбрав начальные приближения $U_{(i,j)}^{(0)}$ последовательно строят приближения $U_{(i,j)}^{(k)}$ для внутренних узлов (x_i, y_j) сетки

S_h: $U_{ij}^{(k)} = \dfrac{1}{4}\left[U_{i-1,j}^{(k-1)} + U_{i+1,j}^{(k-1)} + U_{i,j-1}^{(k-1)} + U_{i,j+1}^{(k-1)}\right]$, k=1,2,…

Что касается граничных узлов сетки, то значения функции в этих узлах последовательно исправляем по формулам линейной интерполяции:

$$U^{(0)}(A)=U(P)=\varphi(P),$$
$$U^{(k)}(A)=U(P)+\frac{U^{(k-1)}(B)-U(A)}{h+\delta}\delta,\quad k=1,2,..,$$

где P – ближайшая к точке A точка границы Г, B – ближайший к A внутренний узел сетки.

Геометрическая интерпретация показана на рисунке 10.5. При этом $\delta>0$, если A – внутренняя точка области G и $\delta<0$, если A – внешняя точка области G. Если узел A лежит на границе Г, то $\delta=0$. На практике после некоторого шага можно считать, что $U^{(k)}(A)$ остаются неизменными.

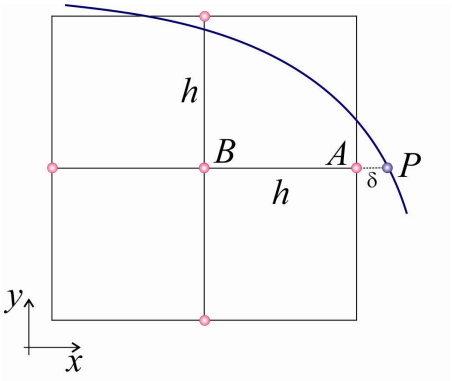


Рис. 10.5. Покрытие криволинейной области сеткой.

За начальные значения $U^{(0)}_{(i,j)}$ теоретически можно взять любую систему чисел. Однако в силу принципа максимума для значений искомой функции $U(x,y)$ должны быть выполнены неравенства: $\min\limits_{\Gamma}\varphi(P)\le U_{ij}\le\max\limits_{\Gamma}\varphi(P)$. Практически для выбора начального приближения решают задачу Дирихле на крупной сетке, а затем найденные значения используют для решения на более мелкой сетке. Для любого шага сетки процесс Либмана независимо от выбора начальных приближений сходится, т.е. существует $\lim\limits_{k\to\infty}U^{(k)}_{ij}=U_{ij}$, причем погрешность приближенного решения имеет порядок $O(h^2)$. Для практического применения вычислений заготавливают достаточное количество вычислительных шаблонов, каждая ячейка которых соответствует узлу сетки. Если уточнение граничных значений не производится, то данный вычислительный шаблон содержит ячейки только для внутренних узлов.

Для оценки точности на практике используют схему с двойным пересчетом соответственно с шагами h и $2h$. Если соответствующие результаты совпадают с заданной точностью, то результат принимаем, в противном случае продолжают схему двойного пересчета.

10.6 Метод сеток для уравнений параболического типа

В качестве уравнения параболического типа рассмотрим одномерное уравнение теплопроводности для однородного стержня $0 \le x \le l$:

$$\frac{\partial U}{\partial t} = a^2 \frac{\partial^2 U}{\partial x^2},$$

где $U(x)$ – температура, t – время, $a^2 = 1$ – нормированный коэффициент температуропроводности.

Пусть задано распределение температуры в начальный момент времени (начальное условие) $U(x,0) = f(x)$ и заданы изменения температуры в зависимости от времени на концах стержня (граничные условия первого рода): $u(0,t) = \varphi(t),\ \ u(l,t) = \psi(t).$

В системе координат $\{x,t\}$ в полуполосе $t \ge 0,\ \ 0 \le x \le l$ построим прямоугольную сетку: $x = ih, t = jk, i = \overline{0,n}$, где $h = \frac{l}{n}, k = \sigma h^2$ – шаги вдоль соответствующих осей. Принимая обозначения $x_i = ih, t_j = jk, U_{i,j} = U(x_i, t_j)$, заменим исходное уравнение конечно-разностным: $\frac{U_{i,j+1} - U_{i,j}}{\sigma h^2} = \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2}$.

Или $U_{i,j+1} = \sigma U_{i-1,j} + (1 - 2\sigma) U_{i,j} + \sigma U_{i+1,j}$. (10.9)

Из рассмотрения формулы (10.9) ясно, что, зная значения функции $U(x,t)$ в точках j-го слоя с помощью этой формулы можно вычислить значения $U(x,t)$ в точках следующего $(j+1)$го слоя. Если в вычислении участвуют четыре соседних узла, реализуется так называемая *явная схема*.

Исходя из начального условия находим значение температуры в первой полуполосе, а затем последовательно вычисляем значения на последующих

слоях $U(x_i,t_1),U(x_i,t_2)$.... через значения в предыдущих, используя для граничных узлов краевые условия.

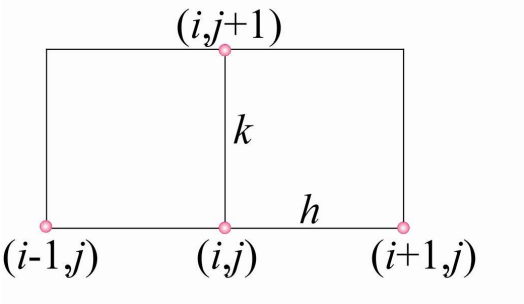


Рис. 10.6. Явная схема.

Значение $\sigma = 1/6$ является для схемы 1 наилучшим. Так как при таком выборе σ для погрешности $R_h(U)$ получаем оценку $R_h(U) = O\left(h^4\right)$, тогда как при другом выборе числа σ имеем $R_h(U) = O\left(h^2\right)$. При использовании конечно-разностной схемы (10.9) для решения уравнения теплопроводности возникает вопрос об устойчивости такой схемы. Доказано, что схема будет устойчивой, если σ удовлетворяет условию: $0 < \sigma \le \frac{1}{2}$. Для устойчивости явной разностной схемы шаги по времени и координате должны быть различными, причем выбор шага по координате накладывает определенные ограничения на выбор временной координаты (условия Куранта, Фридриха и Леви).

Неявная схема, открытая Кранком и Николсоном основана на численных приближениях для решения уравнения (10.9) в точке $\left(x, t + k/2\right)$, которая находится между рядами решетки. Приближение, используемое для $U_t\left(x, t + k/2\right)$, получено по формуле центрированной разности:

$$U_t\left(x, t + k/2\right) = \frac{U\left(x, t + k\right) - U\left(x, t\right)}{k} + O\left(k^2\right).$$

Используемое для $U_{xx}\left(x, t + k/2\right)$ приближение является средним значением приближений $U_{xx}\left(x, t\right)$ и $U_{xx}\left(x, t + k\right)$, которое имеет точность порядка $O\left(h^2\right)$. Схема Кранка-Николсона показана на рисунке 10.7.

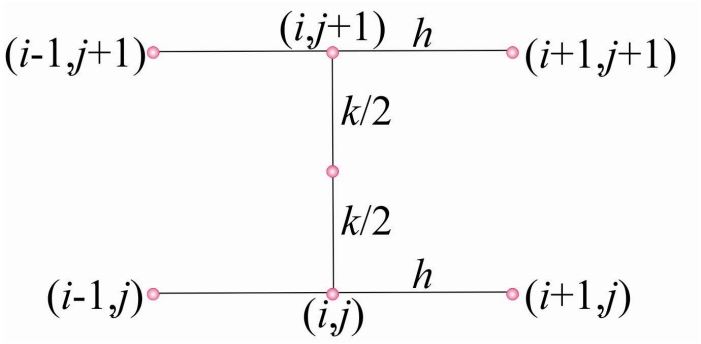


Рис. 10.7. Схема Кранка-Николсона.

Конечно-разностное уравнение имеет вид:

$$\frac{U_{i,j+1}-U_{i,j}}{k}=\frac{U_{i-1,j+1}-2U_{i,j+1}+U_{i+1,j+1}+U_{i-1,j}-2U_{i,j}+U_{i+1,j}}{2h^2}.$$

Полученная СЛАУ может быть также эффективно решена методом прогонки.

10.7 Метод сеток для уравнений гиперболического типа

Остановимся на простейшем уравнении гиперболического типа – уравнении свободных колебаний однородной ограниченной струны:

$$\frac{\partial^2 U}{\partial t^2}=a^2\frac{\partial^2 U}{\partial x^2}$$

с начальными и граничными условиями:

$U(x,0)=f(x),\ \ U_t(x,0)=F(x),\ \ 0\le x\le l,\ U(0,t)=\varphi(t),\ \ U(l,t)=\psi(t),\ \ 0\le t<\infty.$

В системе координат $\{x,t\}$ покроем полуполосу $t\ge 0,\ \ 0\le x\le l$ прямоугольной сеткой: $x=ih,\ \ t=jk,\ \ i=\overline{0,n}$, где h,k – шаги вдоль соответствующих осей. Заменим исходное уравнение конечно-разностным:

$$\frac{U_{i,j+1}-2U_{i,j}+U_{i,j-1}}{k^2}=a^2\frac{U_{i+1,j}-2U_{i,j}+U_{i-1,j}}{h^2}.$$

При $k=\dfrac{h}{a}$ уравнение упрощается: $U_{i,j+1}=U_{i+1,j}+U_{i-1,j}-U_{i,j-1}$.

Для начала вычислительного процесса необходимо знать значение температуры на двух предыдущих узлах $j=1, j=0$, на нулевом слое используем на-

чальное условие, а для нахождения недостающих уравнений в системе найдем значение $U(x,t)$ на фиктивном слое с номером $j=-1$.

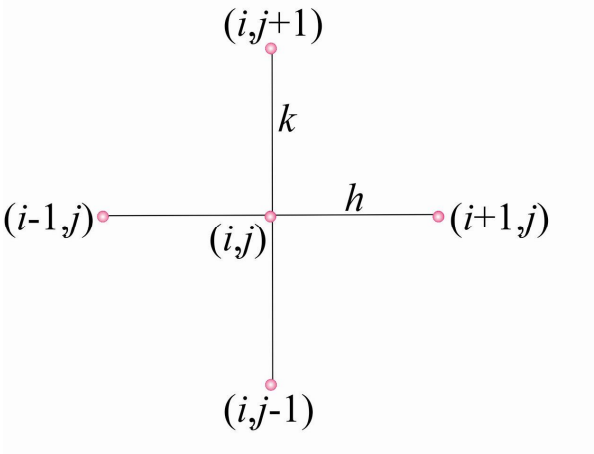


Рис. 10.8. Пятиточечная схема.

Для этого заменим производную во втором начальном условии конечно-разностным отношением:

$$\frac{U_{i,-1}-U_{i,0}}{-k}=F_i \Rightarrow U_{i,-1}=U_{i,0}-kF_i.$$

10.8 Решение граничных задач для дифференциальных уравнений в частных производных с использованием среды PDETOOL Matlab

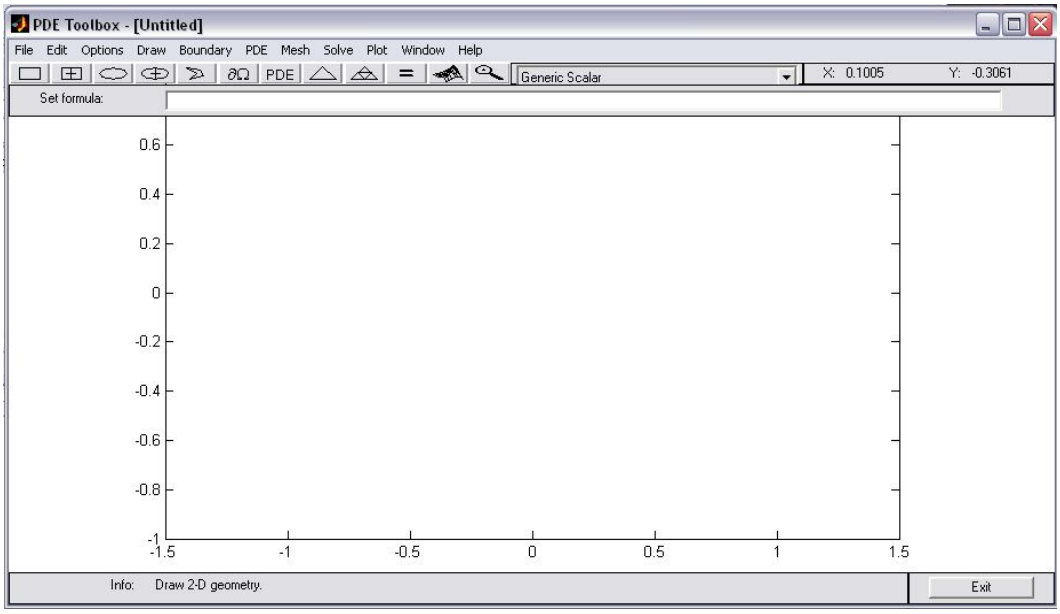
Большинство задач механики, теплопроводности, течения жидкости, электростатики и электродинамики сводится к дифференциальным уравнениям в частных производных в областях сложной формы. Partial Differential Equations Toolbox (PDE Toolbox), предназначен для решения граничных задач для дифференциальных уравнений в частных производных в двумерных областях методом конечных элементов.

Среда pdetool позволяет задать геометрию области, тип и коэффициенты дифференциального уравнения, граничные и начальные условия, произвести разбиение области на конечные элементы, решить получающуюся систему линейных уравнений и визуализировать результат.

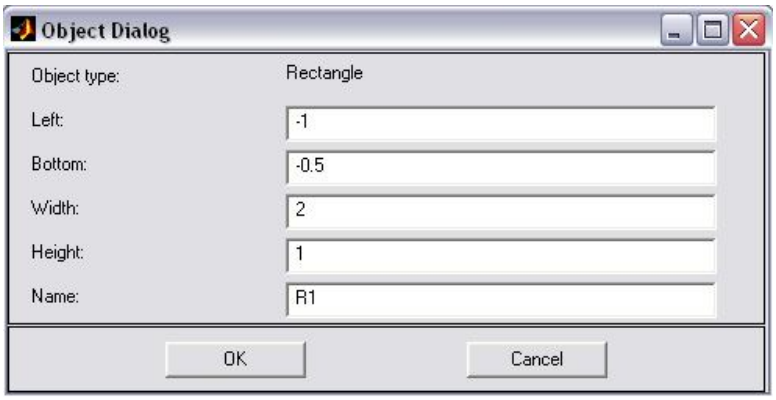
Рассмотрим основы работы в pdetool на примере задачи теплопроводности в простой области. Требуется найти распределение температуры T в прямо-

угольной области с расположенным в центре отверстием. Правая и левая границы прямоугольной области теплоизолированы. Внутри области нет источников тепла и коэффициент теплопроводности k равен 200. Распределение температуры описывается дифференциальным уравнением $k\nabla \cdot \nabla T = 0$, граничные условия на правой и левой границе задают нулевой поток тепла (n – вектор нормали к границе) $n \cdot k\nabla T = 0$, на верхней и нижней границе $T = 600$, а на окружности $T = 500$. Размерности единиц нормированы.

Начнем работу с выполнения команды *pdetool* в командном окне.



В появившемся окне PDE Toolbox среды pdetool сконструируем область. Выберем в меню Draw пункт Rectangle/square и построим прямоугольник. Нарисуем мышью требуемую прямоугольную область на осях от угла, затем двойным щелчком мыши вызовем диалоговое окно Object Dialog, в котором зададим координаты нижнего левого угла прямоугольника, а также его ширину и высоту.

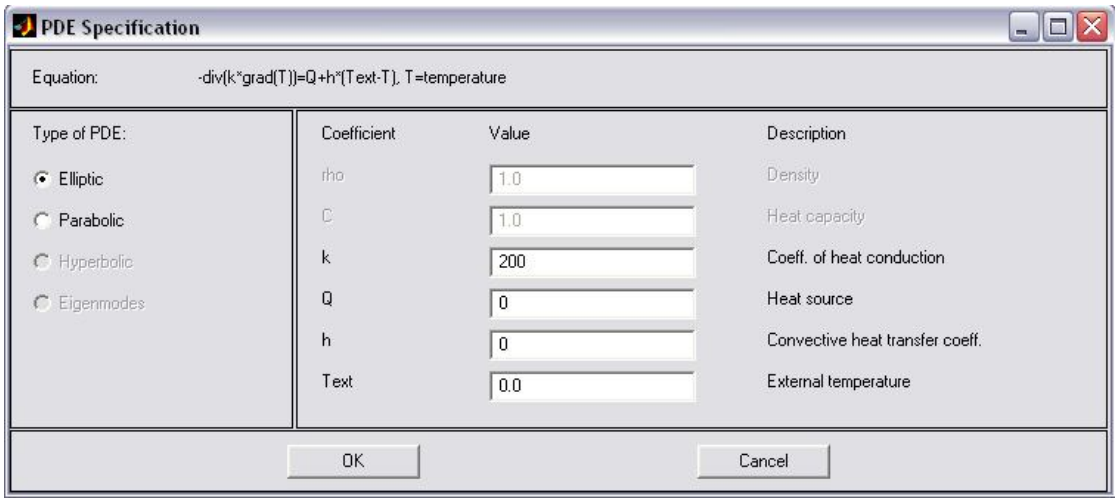


Выберем в меню Draw пункт Ellipse/circle и построим круг с центром в точке (0,0) и радиусом 0.25.

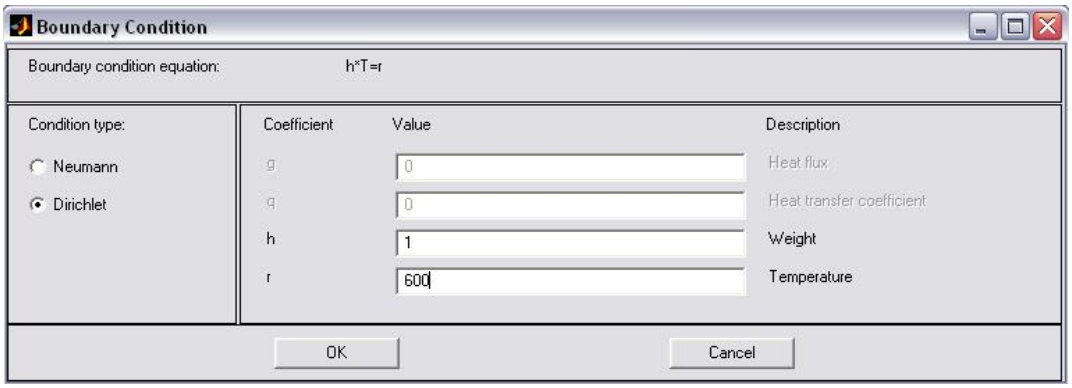
Следующий этап состоит в определении взаимосвязи между примитивами, образующими область, – круг должен быть удален из прямоугольника. Связь между примитивами определяется в строке Set Formula среды pdetool. Знак плюс означает объединение объектов, а минус – вычитание.

Зададим коэффициенты уравнения и граничные условия. Меню Options содержит подменю Application, которое позволяет задать тип решаемой задачи. Пункт Heat Transfer соответствует задаче о распределении тепла. Установим режим дифференциального уравнения, выбрав пункт PDE Mode в меню PDE.

Перейдем к пункту PDE Specification… меню PDE, появится диалоговое окно PDE Specification. Левая панель служит для выбора типа уравнения, Elliptic соответствует задаче о стационарном распределении тепла, описываемой эллиптическим дифференциальным уравнением, а Parabolic – нестационарному случаю. Установим значения коэффициентов:



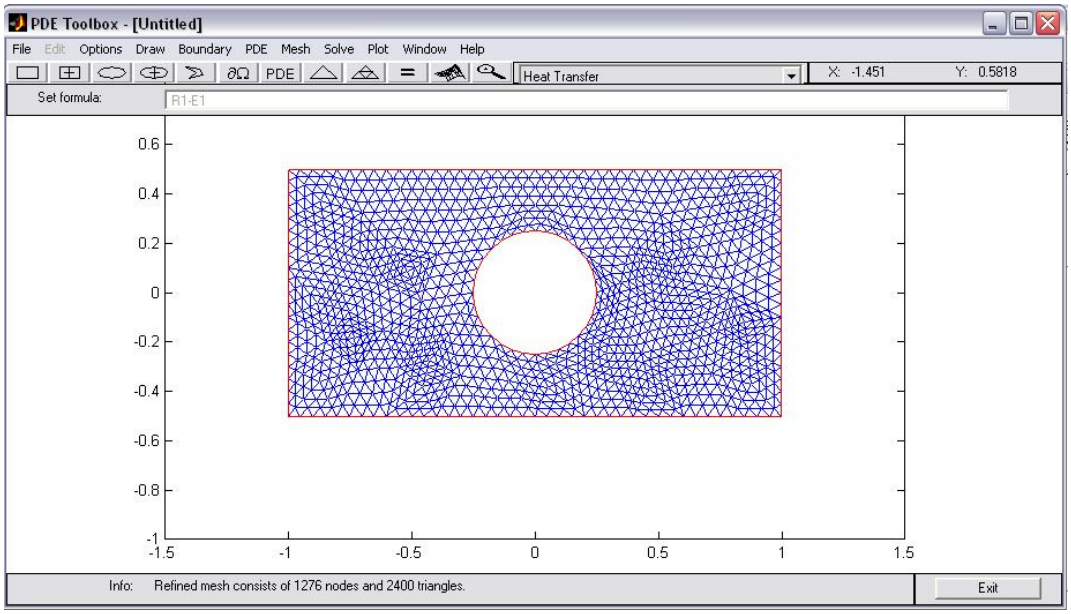
Перейдем к заданию граничных условий. Выберем пункт Boundary Mode меню Boundary. Сделаем текущей верхнюю границу прямоугольника и выберете в меню Boundary пункт Specify Boundary Condition…, появилось диалоговое окно Boundary Condition, предназначенное для выбора типа граничного условия. Определим коэффициенты граничного условия.



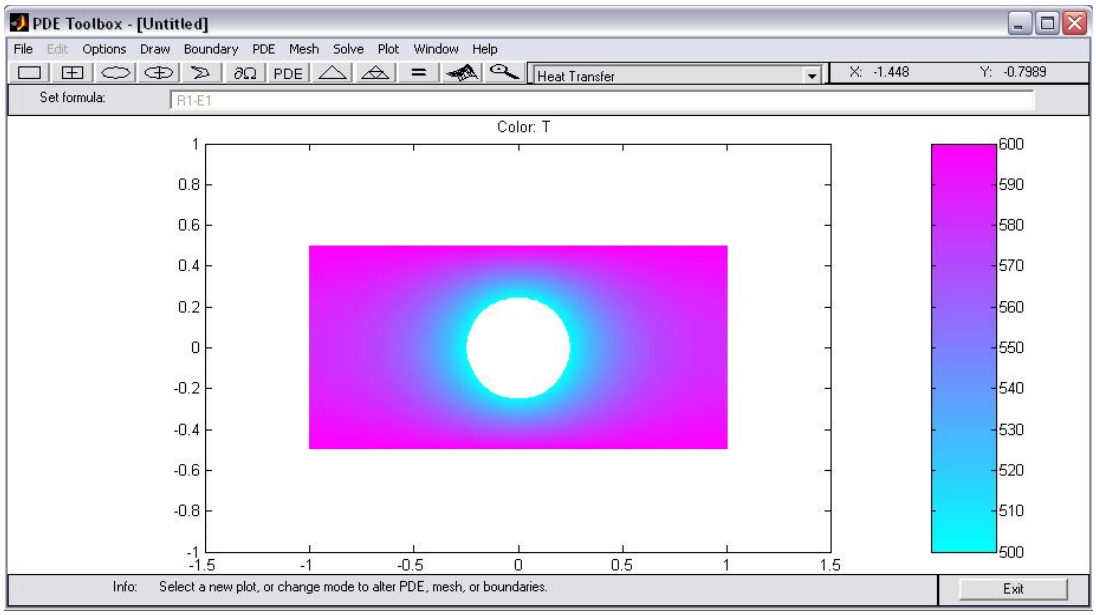
Аналогично определим остальные граничные условия.

Уравнение и граничные условия определены, следующим этапом является решение и визуализация результата.

Перейдем в режим триангуляции, выбрав пункт Mesh Mode, область разбивается на достаточно крупные треугольные элементы. Для получения решения с приемлемой точностью, следует уменьшить шаг разбиения области, выбрав пункт Refine Mesh.



Решение задачи на расчетной сетке производится выбором пункта Solve PDE меню Solve. Найденное распределение температуры отображается в окне среды pdetool контурным графиком с цветовой заливкой, рядом с которым расположен столбик с информацией о соответствии цвета значению температуры.



Для графического вывода решения в виде трехмерного (3D) изображения следует выбрать пункт Parameters… в меню Plot. В появившемся окне редактирования параметров изображения внести в соответствующие поля данные и активизировать кнопку Plot. В результате на экран будет выведен график, как показано на рисунке 10.9.

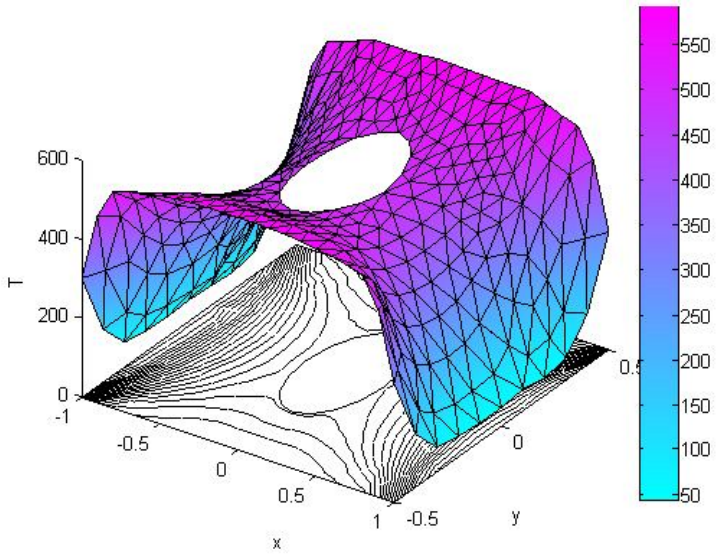


Рис. 10.9. Графическое представление решения.

10.9 Моделирование нестационарных процессов с использованием PDE tool Matlab

Среда PDE tool Matlab предоставляет возможность решения нестационарных прикладных задач, сформулированных в постановке уравнений с частными производными параболического и гиперболического типов, общая запись кото-

рых в обозначениях приложения выглядит соответственно следующим образом:

$$d\frac{\partial U}{\partial t} - \nabla\left(c\nabla U\right) + aU = f \text{ и } d\frac{\partial^2 U}{\partial t^2} - \nabla\left(c\nabla U\right) + aU = f .$$

Параметры d, c, a, f могут зависеть как от переменных x и y, так и от времени t.

Для инициализации уравнения требуется перейти к диалоговому окну **PDE Specification** и выбрать соответствующий тип уравнения. Нестационарная задача требует определения начального условия, для этого надо выбрать **Parameters** в меню **Solve**. В открывшемся диалоговом окне **Solve Parameters** можно установить вектор моментов времени, для которых производится расчет, и начальное распределение температуры (которое может быть функцией координат).

Граничные условия задаются как условия Дирихле или Неймана также, как и для задачи стационарного типа.

PDE tool Matlab позволяет искать решение только для ограниченной расчетной области. Можно визуализировать решение в определенный момент времени, либо проследить эволюцию процесса в отдельном графическом окне. Для реализации последнего требуется установить в диалоговом окне **Plot Selection** флаг для **Animation** и воспользоваться кнопкой **Options** для определения параметров анимированных результатов. В диалоговом окне **Animation Options** требуется задать число кадров в секунду **Animation Rate (fps)** и число повторов анимации **Number of repeats**.

10.10 Численные примеры. Реализация в пакете Matlab

Пример 1. Найти решение задачи Дирихле уравнения Пуассона $U_{xx} + U_{yy} = 1$, с граничными условиями:

$$U(x,0) = 0,\ U(1,y) = 0,\ U(x,1) = 0,\ U(0,y) = 0.$$

Создадим файл, содержащий описание функции, возвращающей численной решение уравнения Пуассона методом релаксации:

```
function [z,r]=relax_meth(N,h,Omega,It,U,C)
i=1:N+1;   x(i)=(i-1)*h; %вычисление координат узлов сетки
j=1:N+1;   y(j)=(j-1)*h; %вычисление координат узлов сетки
r=zeros(N,N);
for k=1:It % итерационный цикл
    for j=2:N    % прохождение по узлам сетки
        for i=2:N
           r(i,j)=(U(i+1,j)+U(i-1,j)+U(i,j+1)+U(i,j-1)
-4*U(i,j))-(h.^2)*C;
           U(i,j)=U(i,j)+(Omega/4)*r(i,j);
        end;
    end;
end;
z=U;
%вычисление матрицы невязки r(i,j) численного решения
for j=2:N
    for i=2:N
        r(i,j)=(U(i+1,j)+U(i-1,j)+U(i,j+1)+U(i,j-1)
-4*U(i,j))-(h.^2)*C;
    end;
end;
```

Выполним следующую последовательность команд:

```
clc, clear all
N=50;
a=5;
b=5;
C=1;
U=zeros(N+1,N+1); %инициализация массива U
% задание граничных условий
i=1:N+1;  j=1:N+1;
U(i,1)=0;% левая граница
U(i,N+1)=0;% правая граница
U(1,j)=0;% нижняя граница
U(N+1,j)=0;% верхняя граница
Omega=1.5; % параметр релаксации
```

```
It=300; % число итераций
h=(a-0)./N; %шаг сетки = 0.1
[z,r]=relax_meth(N,h,Omega,It,U,C);% решение уравнения Пуассона
normr=max(max(abs(r))) % оценка нормы невязки
x=0:h:a;
y=0:h:b;
[x1 y1]=meshgrid(x,y);
meshc(x1,y1,z); colormap(hsv) % построение графика поверхности
```

В графическом окне (рисунок 10.10) показано численное решение уравнения Пуассона. Результат вычисления невязки на экране:

```
normr =   4.6814e-004
```

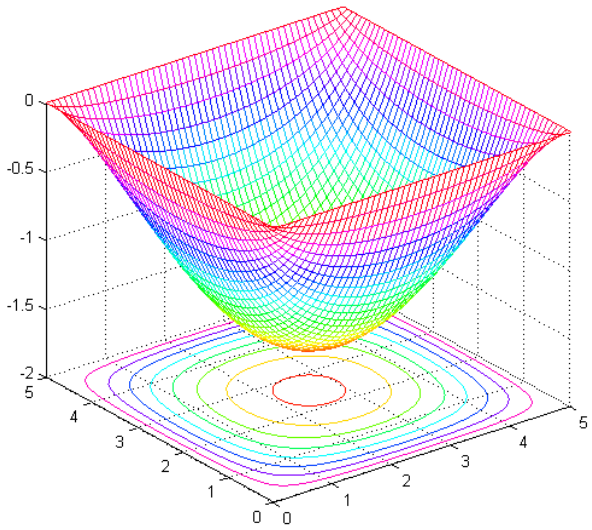


Рис. 10.10. Графическое представление решения задачи.

Пример 2. Найти решение смешанной краевой задачи уравнения в частных производных: $U_t = U_{xx}$, с начальными условиями

$$U(0,x) = \sin(\pi x) + x,\ 0 \le x \le 1,$$

и граничными условиями $U(t,0) = 0,\ U(t,1) = 10$ с использованием неявной разностной схемы.

Для нахождения решения смешанной краевой задачи создадим файл, содержащий описание функции, возвращающей значения искомой функции:

```
function z=Ul(U,A,r,Nt,Nx)
% U - матрица, содержащая начальные и граничные условия
% A - матрица системы
% r=k/(h*h)
% Nt - число шагов по времени
% Nx - число узлов по координате
z=U;
U1=U(1:1,2:Nx-1);
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
```

Выполним следующую последовательность команд:

```
Nx=30; % число узлов по координате x
Nt=100; % число шагов по времени
j=1:Nx; % задание начальных условий
U(1,j)=sin(pi*(j-1)/(Nx-1))+(Nx-1);
% задание граничных условий
alpha=0;
beta=10;
i=1:Nt;
U(i,1)=alpha;
U(i,Nx)=beta;
r=5;
for k=1:Nx-2
```

```
    A(k,k)=1+2*r; % создание матрицы коэффициентов системы линей-
ных уравнений
end;
for m=2:Nx-2
    A(m-1,m)=-r;
    A(m,m-1)=-r;
end;
z=Ul(U,A,r,Nt,Nx);
mesh(z);
view(160,70);
```

Решение смешанной краевой задачи уравнения в частных производных параболического типа показано в графическом окне.

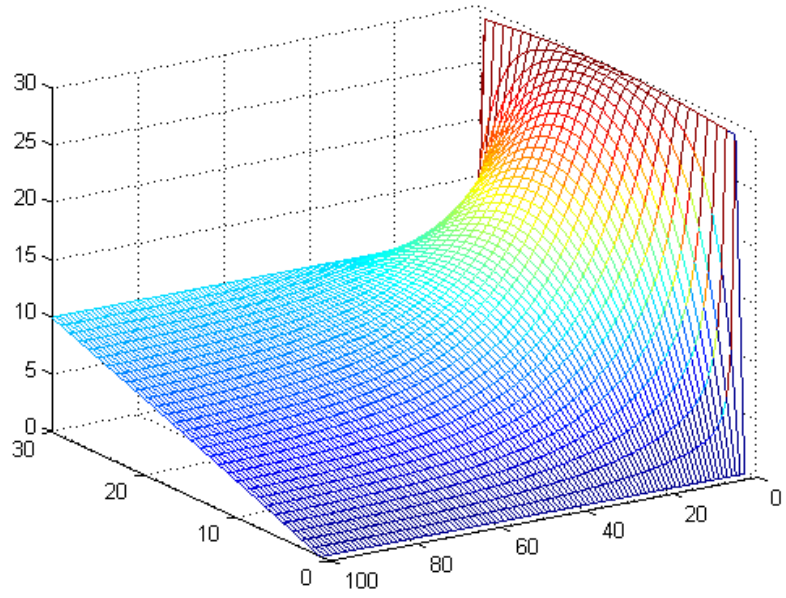


Рис. 10.11. Графическое представление решения.

Контрольные вопросы

1. Какие узлы называются граничными узлами первого и второго рода?

2. Из чего складывается погрешность приближенного решения, полученного разностным методом?

3. Выполнение какого условия необходимо для устойчивости явной разностной схемы для уравнений параболического типа?

4. В чем состоит особенность применения метода сеток для уравнений гиперболического типа?

Индивидуальные задания

1. Применяя метод сеток с шагом h=1/4 найти решение уравнения Лапласа $\Delta U = 0$ в квадрате с вершинами $A(0,0)$, $B(0,1)$, $C(1,1)$, $D(1,0)$. Краевые условия приведены в таблице:

Вариант	$U\big\vert_{AB}$	U_{BC}	$U\big\vert_{CD}$	$U\big\vert_{AD}$
1	$30y$	$30(1-x^2)$	0	0
2	$30y$	$3\cos\frac{\pi x}{2}$	$30\cos\frac{\pi y}{2}$	0
3	$50y(1-y^2)$	0	0	$50\sin\pi x$
4	$20y$	20	$20y^2$	$50x(1-x)$
5	0	$50x(1-x)$	$50y(1-y^2)$	$50x(1-x)$
6	$30\sin\pi y$	$20x$	$20y$	$30x(1-x)$
7	$30(1-y)$	$20\sqrt{x}$	$20y$	$30(1-x)$
8	$50\sin\pi y$	$30\sqrt{x}$	$30y^2$	$50\sin\pi x$
9	$40y^2$	40	40	$40\sin\frac{\pi x}{2}$
10	$50y$	$50(1-x^2)$	100	$60x$
11	$10y(1-y^2)$	$90(1-x^2)$	$30\cos\frac{\pi y}{2}$	0
12	0	$10\cos\frac{\pi x}{2}$	$20y^2$	$50\sin\pi x$
13	$60\sin\pi y$	0	$10\cos\frac{\pi x}{2}$	$10x(1-x)$
14	$10y$	30	$20y$	$\sin\pi x$
15	$20y^2$	$x(1-x)$	90	$50x(1-x)$
16		$40x$	$50y(1-y^2)$	$30x(1-x)$
17	$55y(1-y^2)$	$10\sqrt{x}$	$20y$	$40\sin\frac{\pi x}{2}$
18	$60y^2$	$70\sqrt{x}$	$30y^2$	$50\sin\pi x$
19	10	10	$10\cos\frac{\pi x}{2}$	$30(1-x)$
20	$20\sin\pi y$	$10\left(1-x^2\right)$	10	x

2. Разностным методом с шагом h найти решение уравнения Лапласа в области G при указанных краевых условиях. Решение конечно-разностной системы получить методом Либмана с уточнением граничных значений.

Вариант	Шаг	Область	Краевые условия
1	h=1	$x^2+y^2\le 16$	$U\big\|_{\Gamma}=x^2+2y^2$
2	h=1	$x^2+y^2\le 16$	$U\big\|_{\Gamma}=x^2y^2+10$
3	h=1/2	$x^2+y^2\le 4$	$U\big\|_{\Gamma}=4x^2\cdot\|y\|$
4	h=1/4	$x^2+y^2\le 1$	$U\big\|_{\tilde{A}}=1{,}5\cdot x^2\cdot\|2\cdot y\|$
5	h=1/3	$x^2+y^2\le 1$	$U\big\|_{\Gamma}=4\|x\|\cdot\|y\|+2.5$
6	h=1/4	$x^2+y^2\le 1$	$U\big\|_{\Gamma}=0.5x^2\cdot\|y\|$
7	h=1/2	$x^2+y^2\le 4$	$U\big\|_{\Gamma}=2x^2\cdot\|y\|$
8	h=1	$x^2+y^2\le 25$	$U\big\|_{\Gamma}=4x^2\cdot\|y\|+8$
9	h=1/3	$x^2+y^2\le 1$	$U\big\|_{\Gamma}=4x^2\cdot y^2+2.5$
10	h=1/4	$x^2+y^2\le 1$	$U\big\|_{\Gamma}=x^2y^2+3\|x\|$
11	h=1/4	$x^2+y^2\le 25$	$U\big\|_{\Gamma}=4x^2\cdot\|y\|$
12	h=1	$x^2+y^2\le 16$	$U\big\|_{\Gamma}=6x^2\cdot\|4\cdot y\|$
13	h=1/2	$x^2+y^2\le 49$	$U\big\|_{\Gamma}=x^2+2y^2$
14	h=1/4	$x^2+y^2\le 1$	$U\big\|_{\Gamma}=x^2\cdot\|y\|$
15	h=1/3	$x^2+y^2\le 4$	$U\big\|_{\Gamma}=2\cdot x^2\cdot\|2\cdot y\|$
16	h=1/4	$x^2+y^2\le 16$	$U\big\|_{\Gamma}=2x^2\cdot\|y\|$
17	h=1/2	$x^2+y^2\le 25$	$U\big\|_{\Gamma}=x^2y^2+10$
18	h=1	$x^2+y^2\le 49$	$U\big\|_{\Gamma}=4x^2\cdot y^2+2.5$
19	h=1/3	$x^2+y^2\le 16$	$U\big\|_{\Gamma}=x^2y^2+3\|x\|$
20	h=1/4	$x^2+y^2\le 49$	$U\big\|_{\Gamma}=4\|x\|\cdot\|y\|+2.5$

3. Используя метод сеток, составить решение смешанной задачи для дифференциального уравнения параболического типа $\dfrac{\partial U}{\partial t}=\dfrac{\partial^2 U}{\partial x^2}$ при заданных начальных и граничных условиях:

$$U(x,0)=f(x),\ U(0,t)=q(t),\ U(0.6,t)=p(t),\ x\in[0,0.6].$$

Решение задачи выполнить при $h = 0.1$ для $t \in [0,0.01]$ с четырьмя десятичными знаками.

Вариант	$f(x)$	$q(t)$	$p(t)$
1	$x(x-1)+0.2$	$3t+1$	$2t$
2	$\sin(x+0.66)$	0.4652	$0.165+4t$
3	$\cos(2x-0.22)$	$t+0.52$	0.65
4	$\sin(2.63-x)$	$5t-0.25$	0.141
5	$x(x-1)-0.48$	$0.46521-0.1t$	0.68
6	$\sin(2x)$	$3(t-0.12)$	$3t+1$
7	$3x(x-1)$	$0.08+2t$	0.36
8	$2x(x-1)+0.14$	0.122	$3t+1$
9	$\cos(2x+0.33)$	$t-0.47$	0.121
10	$3x(x+1)$	$0.165+4t$	0.157
11	$x(x-1)+0.5$	$0.46521-0.1t$	0.36
12	$\sin(x+0.35)$	0.155	$6t$
13	$\cos(x-0.22)$	$t+0.52$	0.65
14	$\sin(1.63-x)$	0.4652	0.121
15	$x(x-1)$	$3t+1$	0.68
16	$\sin(3x)$	$3(t-0.12)$	0.157
17	$5x(x-1)$	$0.08+2t$	$0.165+4t$
18	$2x(x-1)+0.5$	$0.165+4t$	$3t+1$
19	$\cos(2x)$	$5t-0.25$	0.141
20	$x(x+1.5)$	$t-0.47$	$3t+1$

4. Определить значение искомой функции, удовлетворяющей уравнению Лапласа $\Delta U = 0$, внутри двумерной области (рис. 10.12) с указанными в варианте краевыми условиями. Решение уравнения найти, используя, встроенный инструментарий ППП Matlab PDE tool. Выполнить графическое представление и анализ с полученных результатов.

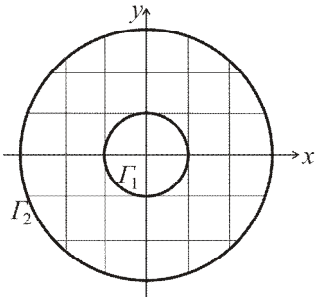


Рис. 10.12. Геометрия расчетной области.

Вариант	Область	Краевые условия
1	$4 \le x^2 + y^2 \le 36$	$U\|_{\Gamma_2} = x^2 + 2y^2,\ U\|_{\Gamma_1} = 3$
2		$U\|_{\Gamma_2} = x^2 \cdot \|y\| + 5,\ U\|_{\Gamma_1} = 0$
3		$U\|_{\Gamma_1} = 100,\ \left.\frac{\partial U}{\partial n}\right\|_{\Gamma_2} = -10$
4	$2 \le x^2 + y^2 \le 16$	$U\|_{\Gamma_1} = 4,\ U\|_{\Gamma_2} = (x+1)^2 \cdot (y+1)^2$
5		$U\|_{\Gamma_1} = 20,\ U\|_{\Gamma_2} = \|x\| \cdot \|y\| + 2x^2y^2$
6		$\left.\frac{\partial U}{\partial n}\right\|_{\Gamma_1} = 2,\ U\|_{\Gamma_2} = 50$
7	$3 \le x^2 + y^2 \le 25$	$U\|_{\Gamma_1} = 0,\ U\|_{\Gamma_2} = (x-1)^2 \cdot (y-1)^2$
8		$U\|_{\Gamma_1} = 10,\ U\|_{\Gamma_2} = \|x\| \cdot \|y\| + 25$
9		$U\|_{\Gamma_1} = 20,\ \left.\frac{\partial U}{\partial n}\right\|_{\Gamma_2} = -1$
10	$9 \le x^2 + y^2 \le 100$	$U\|_{\Gamma_1} = 18,\ U\|_{\Gamma_2} = \|x\| \cdot \|y\| + 4x^2y^2$
11		$U\|_{\Gamma_1} = 2x + y^2,\ U\|_{\Gamma_2} = 2x$
12		$\left.\frac{\partial U}{\partial n}\right\|_{\Gamma_1} = -3,\ U\|_{\Gamma_2} = 10$
13	$25 \le x^2 + y^2 \le 81$	$U\|_{\Gamma_1} = x + y^2, U\|_{\Gamma_2} = 6x$
14		$U\|_{\Gamma_1} = 4\sin(xy),\ U\|_{\Gamma_2} = \|x\| \cdot \|y\|$
15		$U\|_{\Gamma_1} = 5,\ \left.\frac{\partial U}{\partial n}\right\|_{\Gamma_2} = 2$
16		$\left.\frac{\partial U}{\partial n}\right\|_{\Gamma_1} = 1,\ U\|_{\Gamma_2} = x^2 \cdot \|y\| + 1$
17	$25 \le x^2 + y^2 \le 36$	$U\|_{\Gamma_1} = 0,\ U\|_{\Gamma_2} = \|x\| \cdot \|y\| + 16$
18		$U\|_{\Gamma_1} = 9,\ U\|_{\Gamma_2} = (x-1)^2 \cdot (y-1)^2$
19		$U\|_{\Gamma_2} = x^2 + 3y^2,\ U\|_{\Gamma_1} = 5$
20		$U\|_{\Gamma_1} = 3,\ U\|_{\Gamma_2} = (x+1)^2 \cdot (y+1)^2$

5. Оформить отчет по лабораторной работе.

11 ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ

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.

Определение. *Интегральным уравнением* называется уравнение относительно неизвестной функции, содержащейся под знаком интеграла.

Интегральное уравнение в достаточно общем виде можно представить в следующей форме:

$$x(t) = \int_D K(t, s, x(s))ds + f(t), \qquad (11.1)$$

где D – некоторая область n-мерного пространства, x – неизвестная функция, f – известная функция, K – функция относительно x (линейная или нелинейная).

Далее мы ограничимся рассмотрением одномерных линейных интегральных уравнений, 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 $Q(t, s) \cdot x(s)$.

Классификация типов линейных интегральных уравнений проводится по виду верхней границы интеграла в (11.1): если верхняя граница интегрирования является постоянной, то уравнение называется *уравнением Фредгольма*, если переменной – *уравнением Вольтера*, которые, в свою очередь, подразделяются на уравнения первого и второго рода. На практике наиболее широко применяются линейные интегральные уравнения второго рода:

Фредгольма:

$$x(t) = \lambda \int_a^b Q(t,s)x(s)ds + f(t) \qquad (11.2)$$

и Вольтерра:

$$x(t) = \lambda \int_a^t Q(t,s)x(s)ds + f(t), \qquad (11.3)$$

где $f(t)$ – известная функция, $x(t)$ – решение уравнения, $Q(t,s)$ – ядро интегрального уравнения.

Ядро интегрального уравнения Фредгольма определяется на множестве точек квадрата $[a,b]\times[a,b]$, уравнение Вольтерра – в треугольнике $a \le s \le t \le b$.

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 отдельного рассмотрения.

Дополнительный множитель λ, который может быть отнесен к интегральному ядру, в (11.2), (11.3) введен для придания уравнениям более общего вида. Имеются теоремы, устанавливающие существование решений интегральных уравнений при различных значениях λ, которые доказываются подобно тому, как это делается в теории линейных дифференциальных уравнений, через рассмотрение соответствующих однородных уравнений ($f(t) = 0$).

Значительно более сложной задачей оказывается необходимость доказательства существования, единственности и непрерывной зависимости решений от функции $f(t)$ для интегральных уравнений первого рода:

Фредгольма

$$\int_a^b Q(t,s)x(s)ds = f(t); \qquad (11.4)$$

Вольтерра

$$\int_a^t Q(t,s)x(s)ds = f(t). \qquad (11.5)$$

Эта задача относится к классу некорректных задач.

Уравнения первого и второго рода можно записать в общем виде, используя функцию $h(t)$, тождественно равную нулю для уравнения первого рода и единице для уравнений второго рода:

$$h(t)x(t) = \lambda \int_D Q(t,s)x(s)ds + f(t). \qquad (11.6)$$

Когда функция $h(t)$ обращается в ноль в некоторых точках прямоугольника интегрирования, уравнение (11.6) относится к интегральным уравнениям третьего рода. Уравнения данного типа встречаются в приложениях значительно реже, чем уравнения первых двух типов, и значительно менее изучены.

Многие используемые на практике интегральные уравнения имеют ядро, зависящее только от разности $(t-s)$. Интегральные уравнения с данным типом ядра называются *уравнениями с разностным ядром*. Примером таких уравнений является уравнение, получаемое в задаче Абеля.

Если $Q(t,s)$ и $f(t)$ – непрерывные функции, то при любых значениях параметра λ существует единственно непрерывное решение уравнения Вольтерра второго рода (11.3). Для уравнения Фредгольма второго рода (11.2) при тех же требованиях единственное непрерывное решение существует, например, при условии, что

$$|\lambda| < \frac{1}{C(b-a)}, \qquad (11.7)$$

где $C = \max\limits_{t,s\in[a,b]} |Q(t,s)|$.

При снижении требований к гладкости возможных решений условие (11.7) ослабляется. Например, для функций, интегрируемых с квадратом, в роли достаточного условия фигурирует неравенство

$$|\lambda| < \frac{1}{\sqrt{\int_a^b \int_a^b Q^2(t,s)dtds}}.$$

Известны формулы (или совокупность формул), позволяющие найти точное решение $x(t)$. Например, решение уравнения Вольтерра с $\lambda = 1$ и мультипликативным ядром

$$Q(t,s) = p(t)s(t)$$

вычисляется по формуле

$$x(t) = \int_a^t R(t,s)x(s)ds + f(t),$$

где

$$R(t,s) = p(t)q(s)e^{\int_s^t p(\xi)q(\xi)d\xi}.$$

Решение уравнения Фредгольма с вырожденным ядром

$$Q(t,s) = \sum_{i=1}^{m} p_i(t)q_i(s)$$

имеет вид

$$x(t) = \lambda \sum_{i=1}^{m} z_i p_i(t) + f(t),$$

где числа z_i – решения системы линейных алгебраических уравнений:

$$\begin{cases} z_1 - \lambda c_{11}z_1 - \lambda c_{12}z_2 - \ldots - \lambda c_{1m}z_m = d_1 \\ z_2 - \lambda c_{21}z_1 - \lambda c_{22}z_2 - \ldots - \lambda c_{2m}z_m = d_2 \\ \ldots \\ z_m - \lambda c_{m1}z_1 - \lambda c_{m2}z_2 - \ldots - \lambda c_{mm}z_m = d_m \end{cases}, \qquad (11.8)$$

где $c_{ij} = \int_a^b q_i(s)p_j(s)ds$, $d_i = \int_a^b q_i(s)f(s)ds$.

Условие существования и единственности решения уравнения Фредгольма с вырожденным ядром, как очевидно, зависит от значения определителя $D(\lambda)$ системы линейных алгебраических уравнений (11.8), называемого *определителем Фредгольма*. Если $D(\lambda) \neq 0$, то решение существует и единственно.

Наличие методов нахождения точного решения интегрального уравнения с вырожденным ядром позволяет построить приближенный метод, в основе которого лежит замена одного уравнения другим, ядро которого вырождено и в некотором смысле близко к ядру исходного уравнения. Данная замена ядра опирается на различные способы локальной аппроксимации функций, зависящих от двух переменных. Помимо метода замены ядра на вырожденный извес-

тен ряд других приближенно-аналитических методов решения интегральных уравнений, например, метод последовательных приближений, метод моментов и другие.

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 позволяет 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.

11.1 Квадратурный метод решения интегральных уравнений Фредгольма

Заменим определенный интеграл (11.2) его приближенным значением, вычисляемым с помощью конкретной квадратурной формулы

$$\int_a^b \varphi(s)ds = \sum_{j=1}^{n} A_j \varphi(s_j), \qquad (11.9)$$

где $j = \overline{1,n}$ – номера узлов сетки по переменной, A_j – весовые коэффициенты квадратурной формулы.

Представив правую часть приближенного равенства (11.9) с $\varphi(s) = Q(t,s)x(s)$ вместо интеграла в интегральное уравнение Фредгольма второго рода (11.2), получим

$$x(t) \approx \lambda \sum_{j=1}^{n} A_j Q(t, s_j) x(s_j) + f(t). \qquad (11.10)$$

Выражение (11.10) задает функцию, описывающую приближенное решение интегрального уравнения (11.2). Введем на отрезке $[a,b]$ дискретную вре-

менную сетку $t_1, t_2, ..., t_n$, узлы которой совпадают с узлами сетки $s_1, s_2, ..., s_n$. Для каждого момента времени t_j выполняется равенство

$$x(t_i) \approx \lambda \sum_{j=1}^{n} A_j Q(t_i, s_j) x(s_j) + f(t_i), \tag{11.11}$$

где $i = \overline{1,n}$.

Введем обозначения $Q_{ij} = Q(t_i, s_j)$, $f_i = f(t_i)$, $x_i = x(t_i)$ и запишем (11.11) в виде системы линейных алгебраических уравнений с *n* неизвестными:

$$\begin{cases} (1 - \lambda A_1 Q_{11}) x_1 - \lambda A_2 Q_{12} x_2 - ... - \lambda A_n Q_{1n} x_n = f_1, \\ -\lambda A_1 Q_{21} x_1 + (1 - \lambda A_2 Q_{22} x_2) - ... - \lambda A_n Q_{2n} x_n = f_2, \\ \qquad\qquad\qquad\qquad ... \\ -\lambda A_1 Q_{n1} x_1 - \lambda A_2 Q_{n2} x_2 - ... + (1 - \lambda A_n Q_{nn} x_n) = f_n. \end{cases} \tag{11.12}$$

Для решения которой можно использовать любой из методов решения систем линейных алгебраических уравнений.

Таким образом, нахождение решения уравнения Фредгольма второго рода осуществляется в соответствии 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.

Точность численного решения интегрального уравнения зависит от нескольких факторов: применяемой квадратурной формулы, числа узлов временной сетки, свойств функции $Q(t,s)$. На практике часто используют 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.

Важно понимать, что необходимо согласовать выбор конкретной квадратурной формулы (порядок ее точности) со степенью гладкости ядра интегрального уравнения. Если ядро и свободный член оказывается недостаточно гладкими, то для вычисления интеграла не следует применять высокоточные квадратуры, а лучше ограничиться такими формулами как формулы трапеций и прямоугольников.

11.2 Квадратурный метод решения интегральных уравнений Вольтерра

Так как линейные интегральные уравнения Вольтерра в отличие от уравнения Фредгольма имеют единственное непрерывное решение при любых значениях параметра λ, при нахождении численного решения уравнения

$$x(t) = \int_a^t Q(t,s)x(s)ds + f(t),$$

где $t \in [a,b]$, можно положить $\lambda = 1$.

Учитывая, что уравнение Вольтерра формально можно считать уравнением Фредгольма вида

$$x(t) = \int_a^t Q(t,s)x(s)ds + f(t)$$

с ядром

$$K(t,s) = \begin{cases} Q(t,s) & при\ a \le s \le t \le b, \\ 0 & при\ a \le t \le s \le b, \end{cases} \qquad (11.13)$$

для нахождения решения рассматриваемого уравнения воспользуемся результатами предыдущего параграфа.

Введем в рассмотрение временную сетку $s_j \in [a,b]$, состоящую из n узлов, и выберем конкретную квадратурную формулу с весами A_j, тогда приближенное решение интегрального уравнения принимает вид (11.11). Составим систему линейных алгебраических уравнений, аналогичную схеме (11.12), ко-

торая в силу свойств ядра интегрального уравнения (11.13) вырождается в треугольную:

$$\begin{cases} (1-\lambda A_1 Q_{11})x_1 = f_1, \\ -\lambda A_1 Q_{21}x_1 + (1-\lambda A_2 Q_{22}x_2) = f_2, \\ \dots \\ -\lambda A_1 Q_{n1}x_1 - \lambda A_2 Q_{n2}x_2 - \dots + (1-\lambda A_n Q_{nn}x_n) = f_n. \end{cases} \tag{11.14}$$

Из (11.14) видно, что искомые значения $x_1, x_2, \dots, x_n$ находятся последовательными вычислениями по следующим формулам:

$$x_1 = \frac{f_1}{1 - A_1 Q_{11}},$$

$$x_i = \frac{f_i + \sum\limits_{j=1}^{i-1} A_j Q_{ij} x_j}{1 - A_i Q_{ii}},$$

где $i = \overline{2,n}$.

Получим расчетные формулы для решения уравнения Вольтерра первого рода (11.5) при использовании метода средних прямоугольников. Решения уравнения будем находить в узлах временной сетки

$$t_1 = a + h,\ t_2 = t_1 + h,\ \dots,\ t_i = t_{i-1} + h. \tag{11.15}$$

Подставляя (11.15) в (11.5), получаем равенства

$$\int\limits_a^{t_i} Q(t_i, s)x(s)ds = f(t_i). \tag{11.16}$$

Из (11.16) 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 $[t_{i-1}, t_i]$. Выбор данной сетки означает, что

$$x_1 \approx x(s_1),\ x_2 \approx x(s_2),\dots,\ x_n \approx x(s_n). \tag{11.17}$$

Учитывая выбор квадратурной формулы и условия (11.17), запишем (11.16) в следующем виде:

$$
\begin{aligned}
hQ(t_1,s_1)x_1 &= f(t_1),\\
hQ(t_2,s_1)x_1 + hQ(t_2,s_2)x_2 &= f(t_2),\\
hQ(t_3,s_1)x_1 + hQ(t_3,s_2)x_2 + hQ(t_3,s_3)x_3 &= f(t_3),\\
&\ldots,
\end{aligned}
\tag{11.18}
$$

где $s_i = t_i - \dfrac{h}{2}$.

Из равенств (11.18) последовательно находим:

$$x_1 = \frac{f(t_1)}{hQ(t_1,s_1)},$$

$$x_i = \frac{f(t_i) - h\sum\limits_{j=1}^{i-1} Q(t_i,s_j)x_j}{hQ(t_i,s_i)},$$

где $i = \overline{2,n}$.

При использовании квадратурных формул замкнутого типа и совпадающей системы узлов возникает проблема вычисления значения $x_1 \approx x(t_1) = x(s_1) = x(a)$, которая не возникала при решении уравнений Вольтерра второго рода. Действительно, при $i = 1$ равенство (11.16) теряет всякий смысл.

Для нахождения начального значения x_1 продифференцируем уравнение (11.5) по *t*:

$$Q(t,t)x(t) + \int_a^t Q'_t(t,s)x(s)ds = f'(t). \tag{11.19}$$

Положив в (11.19) $t = a$, получим

$$Q(a,a)x(a) = f'(a).$$

Откуда следует, что $x(a) = \dfrac{f'(a)}{Q(a,a)}$, следовательно, $x_1 = \dfrac{f'(a)}{Q_{11}}$.

При использовании квадратурной формулы трапеций далее получаем:

$$\frac{h}{2}Q_{21}x_1 + \frac{h}{2}Q_{22}x_2 = f_2 \;\Rightarrow\; x_2 = \frac{f_2 - \dfrac{h}{2}Q_{21}x_1}{\dfrac{h}{2}Q_{22}},$$

$$\frac{h}{2}Q_{31}x_1 + hQ_{32}x_2 + \frac{h}{2}Q_{33}x_3 = f_3 \;\Rightarrow\; x_3 = \frac{f_3 - \frac{h}{2}Q_{31}x_1 - hQ_{32}x_2}{\frac{h}{2}Q_{33}},$$

т.е. в общем случае при любом $j = 2,3,...,$

$$x(s_j) \approx x_j = \frac{f_j - \frac{h}{2}Q_{j1}x_1 - h\sum_{k=2}^{j-1}Q_{jk}x_k}{\frac{h}{2}Q_{jj}}.$$

11.3 Численные примеры. Реализация в пакете Matlab

Пример 1. Найти решение интегрального уравнения

$$x(t) = \int_1^2 \frac{x(s)ds}{\sqrt{t^2+s^2}} + \sqrt{t+1} + t^2.$$

Создадим файл, содержащий описание функции, возвращающей значение подынтегральной функции $Q(t,s)$:

```
function z=Q(t,s)
z=1./sqrt(t.^2+s.^2);
```

Создадим файл, содержащий описание функции, возвращающей значение функции $f(t)$:

```
function z=f(t)
z=sqrt(t+1)+t.^2;
```

Создадим скрипт-файл, возвращающей решение интегрального уравнения:

```
%задание пределов интегрирования
a=1;
b=2;
% задание числа разбиений
N=200;
% задание коэффициента
ly=1;
h=(b-a)/(N-1); % расчет шага сетки
for i=1:N
```

```
        t(i)=a+h*(i-1);
    end
    s=t; %инициализация временной сетки
    %задание коэффициентов для квадратурной формулы метода трапе-
ций (вычисление интеграла)
    A(1)=0.5;
    for j=1:N-1
        A(j)=1;
    end
    A(N)=0.5;
    % вычисление значений функции Q(t,s) в узлах сеточной области
    for i=1:N
        for j=1:N
            q(i,j)=Q(t(i),s(j));
        end
    end
    %вычислениезначений функции f(t) в узлах сеточной области
    F=f(t);
    for i=1:N
        for j=1:N
            if i==j
                S(i,j)=1-ly*A(i)*q(i,j)*h;
            else
                S(i,j)=-ly*A(i)*q(i,j)*h;
            end
        end
    end
    x=S^-1*F';
    plot(t,x);
    grid on
```

Визуализация численного решения интегрального уравнения представлена в графическом окне:

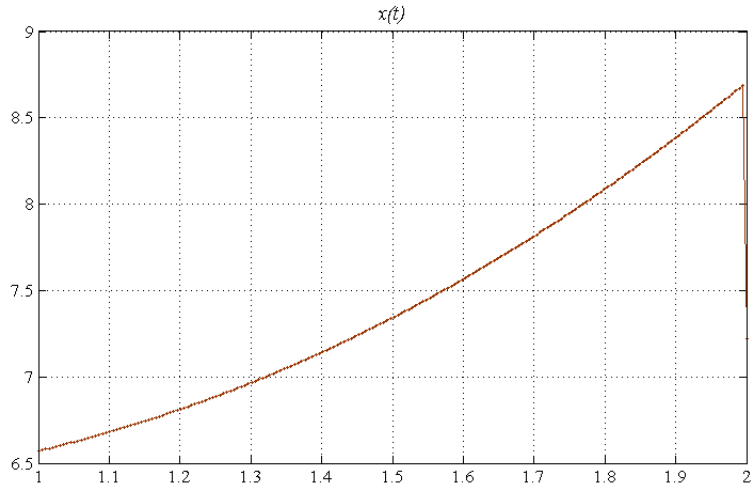


Рис. 11.1. Графическая интерпретация решения интегрального уравнения.

Контрольные вопросы

1. Для какого класса интегральных уравнений применим метод квадратур?

2. От каких факторов зависит точность численного решения интегрального уравнения?

3. В чем заключается принцип контроля точности численного решения – принцип Рунге?

Индивидуальные задания

1. Согласно варианту найти решение интегрального уравнения Фредгольма второго рода, реализовав в пакете Matlab квадратурный метод:

Номер варианта	**Уравнение Фредгольма второго рода**
1	$x(t) = \int_0^1 ts^2 x(s)ds + t$
2	$x(t) = \int_0^{\pi} \cos(t+s)x(s)ds + 1$
3	$x(t) = 4\int_0^1 t^2 s^2 x(s)ds + t$
4	$x(t) = \frac{1}{2}\int_0^1 ts\, x(s)ds + \frac{5}{6}t$

5	$x(t)=\int_0^1 (ts+s)x(s)ds+t$
6	$x(t)=-\int_0^1 t\left(e^{ts}-1\right)x(s)ds+e^t-t$
7	$x(t)=-\int_0^{\pi}\sin(ts)x(s)ds+t$
8	$x(t)=\int_0^{\pi}\left(t^2+s\right)\cos s\cdot x(s)ds+\sin t$
9	$x(t)=\int_0^{2\pi}\sin t\cdot\cos s\cdot x(s)ds+\cos 2t$
10	$x(t)=\int_0^{2\pi}\sin(t+s)x(s)ds+t$
11	$x(t)=\int_0^2 t^2 sx(s)ds+\sqrt{t}+t^2$
12	$x(t)=\int_0^{\frac{\pi}{2}}\sin(t+s)x(s)ds+t$
13	$x(t)=3\int_0^5 ts^2x(s)ds+\sqrt{t}$
14	$x(t)=\frac{1}{2}\int_0^1 ts\,x(s)ds+\frac{5}{6}t$
15	$x(t)=\int_1^2\left(t^2s+s\right)x(s)ds+\sqrt{t}+t$
16	$x(t)=\int_1^4\left(e^{ts}+2\right)x(s)ds+2e^t-\sqrt{t}$
17	$x(t)=\int_0^{\frac{\pi}{2}}\cos(ts)x(s)ds+t^2$
18	$x(t)=\int_0^{2\pi}(t+s)\sin s\cdot x(s)ds+\cos t$
19	$x(t)=\int_0^{\pi}\sin t\cdot x(s)ds+t\sqrt{t}$
20	$x(t)=\int_0^{\pi}\sin\left(t^2+s\right)x(s)ds+t^3$

2. Оформить отчет по лабораторной работе.

ПРИЛОЖЕНИЕ

ОСНОВНЫЕ ПРИЕМЫ РАБОТЫ С ПАКЕТОМ MATLAB

Matlab – интерактивный матрично-ориентированный пакет, предназначенный для выполнения научных и инженерных расчетов. Пакет содержит обширную библиотеку программ по численным методам, использует двух- и трехмерную графику, а также форматы языков высокого уровня. Техника записи циклов, использование логических отношений, вызов подпрограмм и редактирование находят непосредственное применение в среде пакета Matlab.

Найти дополнительную информацию о командах, опциях и примерах можно, воспользовавшись справкой в диалоговом режиме, литературой и прилагаемой к пакету документацией.

1 Работа в командном окне

1.1 Вход в систему Matlab. Доступ к справочной информации

При работе в операционных системах Windows 9*x*, Windows 2000, Windows XP вход в пакет Matlab осуществляется простым кликом левой кнопки мыши по соответствующей иконке. В версиях пакета 5.0 и выше имеется собственный встроенный редактор текста. Командное окно пакета находится в одном окне, а редактор текста – в другом.

Существуют следующие способы получить информацию о функциях пакета Matlab в процессе работы.

1. Команда **`help`**

Команда **`help <имя_группы>`** выведет на экран список функций, размещенных в этой группе с краткими пояснениями. Например, команда

```
>> help elmat
```

выведет список функций, предназначенных для создания и работы с матрицами специального вида. Ввод команды с именем определенной функции выдаст на экран описание этой функции.

2. Команда **lookfor**

Эта команда позволяет выполнить поиск m-функции по ключевому слову, при этом анализируется первая строка комментария и она же выводится на экран, если в ней встретилось ключевое слово. Например, в системе Matlab нет m-функции с именем `inverse` и поэтому на команду

```
>> help inverse
```

ответом будет

`inverse.m not found`. Однако команда `lookfor inverse` покажет все совпадения, найденные в справочных файлах.

3. Меню **Help.**

4. Обращение к Web-серверу фирмы The Mathworks.

1.2 Арифметические операции. Встроенные скалярные функции. Операторы присваивания. Форматы вывода данных

Matlab можно использовать как обыкновенный калькулятор, который выполняет основные арифметические операции: + (сложение), – (вычитание), * (умножение), / (деление), ^ (возведение в степень) и содержит набор констант (pi, e, i).

Пример.

```
>> (2+3*pi)/2
ans =
     5.7124
```

Для более сложных вычислений в пакете предусмотрены дополнительные операции. Кратко перечислим основные функции, имеющиеся в пакете. Описание других можно найти, используя справку в диалоговом режиме.

Функция	Описание	Функция	Описание
abs()	абсолютное значение числа	**cos()**	$\cos(x)$
sin()	$\sin(x)$	**tan()**	$\mathrm{tg}(x)$
exp()	e^x	**log()**	$\ln(x)$ – натуральный лога-

			рифм
acos()	arccos(x)	**sqrt()**	$\sqrt{x}$ – извлечение кв. корня
atan()	arctg(x)	**round()**	Округление числа до ближайшего целого
ceil()	Наибольшее целое для данного действительного числа	**floor()**	Наименьшее целое число для данного действительного числа
rem()	Остаток от деления двух чисел	**sign()**	Знак числа (1,если знак + и –1, если знак –)

Matlab является интерпретирующим языком непосредственных вычислений. Операторы пакета обычно имеют форму:

>> имя_переменной=выражение

или просто

>> выражение

Выражение формируется из операторов, функций и имен переменных. Если имя переменной и знак равенства в левой части отсутствуют, автоматически генерируется переменная **ans**, которой присваивается результат вычислений. Matlab различает строчные и прописные буквы в именах команд, функций и переменных!

Например, для присвоения выражениям различных имен используется знак равенства.

```
>> s=3-floor(exp(3.3))
s  =
   -24
```

Точка с запятой позволяет задать несколько выражений и вывод значений результата после нажатия клавиши <Enter> не производится:

```
>> d=sin(5.6);          Задание d
>> a=cos(6.5);          Задание a
>> a+d                  Вывод на экран a+d
ans =
```

```
    0.3453
```

Поскольку все вычисления в Matlab выполняются с двоичной точностью, формат вывода может изменяться с помощью следующих команд:

Название формата	Описание формата представления числа
`format short`	С фиксированной точкой и 4 знаками после точки (по умолчанию)
`format long`	С фиксированной точкой и 14 знаками после точки
`format short e`	Научная нотация с 4 десятичными знаками
`format long e`	Научная нотация с 15 десятичными знаками
`format short g`	С фиксированной точкой и 3 знаками после точки
`format long g`	Научная нотация с 13 десятичными знаками

Например,

```
>> format long
>> 3*cos(1.2)
ans =
    1.08707326343002
```

1.3 Создание матриц. Операции с матрицами и массивами. Векторные и матричные функции

Matlab работает с одним видом объектов – числовыми прямоугольными матрицами, элементами которых могут быть в общем случае комплексные числа. Все переменные представляют собой матрицы, матрицы 1×1 интерпретируются как скаляры, матрицы с одной строкой или одним столбцом – как вектора. В системе Matlab матрицы могут быть созданы разными способами:

1. введены явно с помощью списка элементов;
2. сгенерированы встроенными операторами или функциями;
3. созданы в m-файлах;
4. загружены из внешнего файла данных.

Например, в результате выполнения оператора

```
>> A=[1 2 3;3 2 1;4 5 6]
```

или

```
>> A=[1 2 3
3 2 1
4 5 6]
```

Matlab создает матрицу 3×3 и присваивает ее значение переменной A.

```
A =
1  2  3
3  2  1
4  5  6
```

В пакет Matlab встроен ряд функций, позволяющих создавать матрицы специального вида. Приведем некоторые из них:

1. функция **`rand(n)`** создает матрицу размера n×n (функция **`rand(m,n)`** – матрицу размера *m*×*n*), каждый элемент которой – случайное число с равномерным законом распределения в диапазоне [0,1];

2. функция **`magic(n)`** создает матрицу размера *n*×*n*, которая является магическим квадратом;

3. функция **`zeros(m,n)`** создает нулевую матрицу размера *m*×*n*;

4. функция **`ones(m,n)`** создает матрицу размера *m*×*n*, каждый элемент которой равен единице;

5. функция **`diag(x)`** создает матрицу, у которой на главной диагонали стоят элементы вектора *x*;

6. функция **`diag(A)`** создает матрицу, у которой на главной диагонали стоят диагональные элементы матрицы *A*.

Ссылки на отдельные элементы матриц и векторов осуществляются с помощью индексов в круглых скобках.

Приведем несколько примеров задания матриц, выделения элементов и подматриц.

```
>> Z=zeros(4,5); % Создание нулевой матрицы размера 4×5.
>> O=ones(3,2); % Создание единичной матрицы размера 3×2.
```

```
>> X=0:0.5:2 % Вывод на экран вектора X размера 1×5.
X =
  0  0.5000  1.0000  1.5000  2.0000
>> Y=sin(X); % Создание вектора Y размера 1×5, каждый элемент которого является sin от X.
>> A(2,3)
ans =
     1 % Выделение элемента (2,3) из матрицы A.
>> A(8) % Альтернативное выделение элемента(2,3)
ans = % как 8-го по счету, если записать матрицу в
     1 % один вектор-столбец.
>> A(1:2,1:3) % Выделение подматриц A.
ans =
     1   2   3
     3   2   1
>> A([1 2],[1 3])
ans =
     1   3
     3   1
>> A(2,2)=round(tan(8)) % Замена элемента матрицы A.
ans =
     1   2   3
     3  -7   1
     4   5   6
>> A(:,2) % Выделение 2-го столбца.
ans =
     2
    -7
     5
>> A(2,:) % Выделение 2-й строки.
ans =
     3  -7   1
```

В пакете Matlab доступны следующие матричные операции: + (сложение), – (вычитание), * (умножение), ^ (возведение в степень, применима только для квадратных матриц!), ’ (транспонирование), / (правое деление), \ (левое деление). Если размерность матриц не соответствует выполняемой операции, то система генерирует сообщение об ошибке. Приведенные операции могут стать поэлементными, если перед ними поставить точку. Приведем примеры использования матричных операций.

```
>> D=[1 2;4 8];
>> C=D'
C =
   1   4
   2   8
>> 5*(D*C)^2
ans =
    2125   8500
    8500   3400
>> C^2 % Произведение матриц C*C.
ans =
    9   36
   18   72
>> C.^2 % Квадрат каждого элемента матрицы C.
ans =
    1   16
    4   64
>> sin(D./2)
ans =
    0.4794   0.8415
    0.9093   0.7568
```

Часто возникает необходимость использования векторных или матричных функций.

Аргументами векторных функций являются векторы (строки или столбцы). Если в качестве аргумента указана матрица размера m×n, то данная функция действует постолбцово, т.е. результатом действия является вектор-

столбец, каждый элемент которого – результат действия этой функции на соответствующий столбец. Например, максимальный элемент прямоугольной матрицы находится с помощью команды **max(max(A))**. Наиболее употребительными матричными функциями являются:

inv(X) – обратная матрица,

det(X) – определитель матрицы,

size(X) – размерность матрицы,

norm(X) – норма вектора или матрицы,

rank(X) – ранг матрицы,

cond(X) – число обусловленности.

2 Циклы. Условные операторы и операторы отношения

Операторы управления Matlab и операторы отношения при использовании работают также, как и в большинстве языков программирования.

Оператор цикла **for** в общем виде записывается как:

```
for <переменная_цикла>=<выражение_цикла>
<Выполняемые_операторы>
end
```

Оператор цикла **while** записывается следующим образом:

```
while <условие>
<Выполняемые_операторы>
end
```

Условный оператор **if** имеет следующий синтаксис:

```
if <условие>
<Выполняемые_операторы1>
else
<Выполняемые_операторы2>
end
```

В пакете Matlab возможно также множественное ветвление.

В пакете Matlab используются следующие операторы отношений и логические операторы:

==	равно	~	не
~=	не равно	&	и
<	меньше	|	или
>	больше		
<=	меньше или равно		
>=	больше или равно		

Приведем несколько примеров использования циклов и условий.

Пример. Найдем значение 5!.

```
>> a=1;
>> for i=1:5
a=a*i;
end
>> a
a =
  120
```

Пример. Для различных значений *x*, полученных делением на 3 чисел от 0 до 10, выведем значения квадратов и кубов этих чисел.

```
>> n=10;
>> k=0;
>> while k<=n
x=k/3;
disp([x x^2 x^3])
k=k+1;
end
```

Пример. В зависимости от *s* присвоим переменной *a* разные значения.

```
>> s=10;
>> if (s<0)|(s>10)
a=s^2+5;
else
a=0;
end
>> a
a =
  0
```

3 М – файлы

Matlab может последовательно выполнять последовательность операторов, записанных в файл на диске. Имена таких файлов имеют вид <имя>.m и такие файлы называются m-файлами. Большая часть работы в Matlab заключается в создании, редактировании и выполнении таких m-файлов. Существуют два типа m-файлов: файлы-программы (сценарии) и файлы-функции.

3.1 Файлы-программы

Файлы-программы состоят из последовательности обычных операторов пакета Matlab. Если m-файл с таким сценарием имеет имя, например, start.m, то команда `start`, введенная в командной строке, вызовет выполнение последовательности операторов этого файла. Переменные в программе являются глобальными и могут изменить значения переменных с теми же именами в командном окне. В таких файлах легко исправить ошибки, внутри файла можно ссылаться на другие m-файлы и рекурсивно на этот же файл.

Приведем пример программы, которая подсчитывает число отрицательных элементов массива, сохраненной под именем script.m:

```
S=input('Введите массив S'); % команда input для запроса мас-
сива у пользователя
N=length(S); % считываем размерность массива S
count=0; % инициализация счетчика
for i=1:N
    if S(i)>0
        count=count+1; % подсчет числа отрицательных элемен-
тов
    end
end
disp(count); % команда disp для вывода на экран
```

Вызов в командной строке

```
>> script
```

приведет к выполнению операторов программы:

```
>> script
```

```
Введите массив S    [3 -7 5 -8 0 -42]
     3
```

3.2 Файлы-функции

Файлы-функции дают возможность расширять Matlab, поскольку определенные пользователем функции имеют тот же статус, что и другие функции Matlab. Переменные по умолчанию являются локальными и не влияют на имена и значения переменных в текущей рабочей области Matlab.

Общий вид задания:

```
function [x1,x2,…]=name(y1,y2,…)
<тело функции>
```

Первая строка функции содержит объявление выходных аргументов, имени функции и входные аргументов, без этой строки файл является программой, но не функцией.

Совпадение имени функции и имени файла является обязательным условием в Matlab. Символ % указывает на то, что вся строка после него является комментарием и игнорируется при выполнении программы.

Приведем пример функции funpr.m, которая производит подсчет количества элементов массива, кратных заданному числу и вычисляет сумму таких элементов:

```
function [n,S]=funpr(P,d)
N=length(P); % определение размерности массива P
n=0; % обнуление счетчика
S=0; % обнуление суммы
for i=1:N
    if rem(P(i),d)==0 % поиск элементов массива, кратных d
        n=n+1;        % подсчет количества элементов
        S=S+P(i);     % подсчет суммы
    end
end
```

Вызов в командной строке (после задания массива P):

```
>> [n,S]=funpr([2 -9 8 5 0 -6],2)
n = 4
```

```
S = 4
```

присвоит переменным *n* и *S* значения выходных параметров.

Для вычисления значения функции, имя которой передается в функцию через строковую переменную, часто используется функция **`feval`**, аргументами которой являются имя функции и координата точки, в которой вычисляется значение данной функции.

Приведем пример использования функции **`feval`**. Зададим свою функцию `funpr2` изменений аргумента:

```
function z=funpr2(x)
z=x.^3-3*x.^2+2*x+3;
```

И пусть, например, требуется обратиться к этой функции в другом файле-функции, вычисляющей значение функции в середине указанного отрезка:

```
function z=aver(funpr2,a,b)
c=(a+b)/2;
z=feval(funpr2,c)
```

Результат обращения к функции `aver` будет иметь вид:

```
>> aver('funpr2',-2,2)
z =  3
ans = 3
```

3.3 Текстовые строки. Сообщения об ошибках

Текстовые строки вводятся в Matlab в виде текста в одинарных кавычках: **`m='Mistake'`**. Вывод текстовой строки осуществляется при помощи оператора **`disp`**. Оператор `disp(m)` выведет на экран сообщение:

```
Mistake
```

Сообщение об ошибках лучше выводить при помощи функции error, так как после обращения к данной функции выполнение m-файла будет прекращено. Для выхода из цикла (и из программы) можно воспользоваться командой **`break`**. В m-файле запрос на ввод данных можно организовать при помощи оператора **`input`**. При вводе оператора `m=input('введите размерность массива')` на экран выводится запрос и выполнение работы

файла приостанавливается до ввода требуемых данных с клавиатуры. После нажатия клавиши <Enter> данные присваиваются переменной m.

4 Графики

Пакет Matlab позволяет строить двух- и трехмерные графики кривых и поверхностей. Дополнительные возможности и описание графиков можно найти, используя справку в диалоговом режиме.

4.1 Построение двумерных графиков

Команду `plot` используют для построения графика двумерной функции. Следующий пример иллюстрирует, как построить графики функций $y = \sin(x) + x^2$ и $y = 10\sin^2(x)$ на отрезке $\left[-\frac{3\pi}{2}, \frac{3\pi}{2}\right]$:

```
>> x=-3*pi/2:0.01:3*pi/2;
>> y1=sin(x)+x.^2;
>> y2=10*(sin(x)).^2;
>> plot(x,y1,x,y2)
>> grid on
```

В первой строке программы задается область с шагом 0.01. В следующих двух строках задаются две функции. Первые три строки заканчиваются точкой с запятой – это предотвращает вывод векторов *x*, *y*, *z* на экран. Команда **`plot`** строит графики сплошной линией (рисунок П.1).

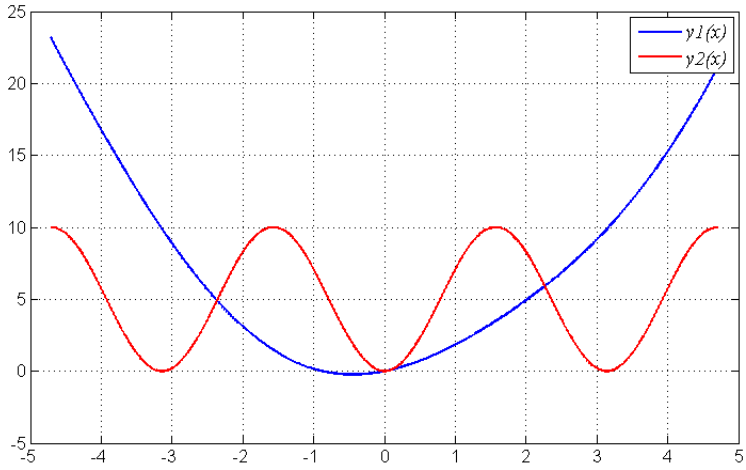


Рис. П.1. Графики двух функций, построенные в одной системе координат.

Команда **plot** позволяет легко задать стиль (тип маркера и тип линий) и цвет линий, например,

```
>> plot(x,f,'k-',x,g,'k:')
```

осуществляет построение первого графика сплошной черной линией, а второго – черной пунктирной.

В таблице приведены возможные значения аргументов с указанием результата.

Цвет		**Тип маркера**		**Тип линии**	
y	желтый	·	точка	-	сплошная
m	розовый	o	кружок	:	пунктирная
c	голубой	×	крестик	-.	штрихпунктирная
r	красный	+	знак «плюс»	--	штриховая
g	зеленый	*	звездочка		
b	синий	s	квадрат		
w	белый	d	ромб		
k	черный	v,∧,<,>,∨	треугольники		
		p,h	звезды		

Удобство оформления графиков зависит от дополнительных элементов оформления: координатной сетки, подписей к осям, заголовка и легенды. Сетка наносится командой **grid on**, функции **xlabel, ylabel** служат для размещения подписей к осям, **title** – для заголовка, **legend** – сопровождение легендой.

4.2 Графики функций двух переменных

Для отображения функции двух переменных следует:

1. сгенерировать матрицы с координатами узлов сетки на прямоугольной области определения функции;

2. вычислить функцию в узлах сетки и записать полученные значения в матрицу;

3. использовать одну из встроенных функций пакета.

Сетка генерируется функцией **meshgrid**, вызываемой с двумя входными аргументами:

```
>> [X,Y]=meshgrid(-2:0.1:2, -2:0.1:2);
>> Z=sin(X.^2).*cos(Y.^2).*(1-X.^2).*(1-Y.^2);
```

Функция **surf** строит каркасную поверхность графика функции и заливает каждую клетку определенным цветом, зависящим от значения функции в точках, соответствующим углам клетки (рисунок П.2):

```
>> surf(X,Y,Z)
```

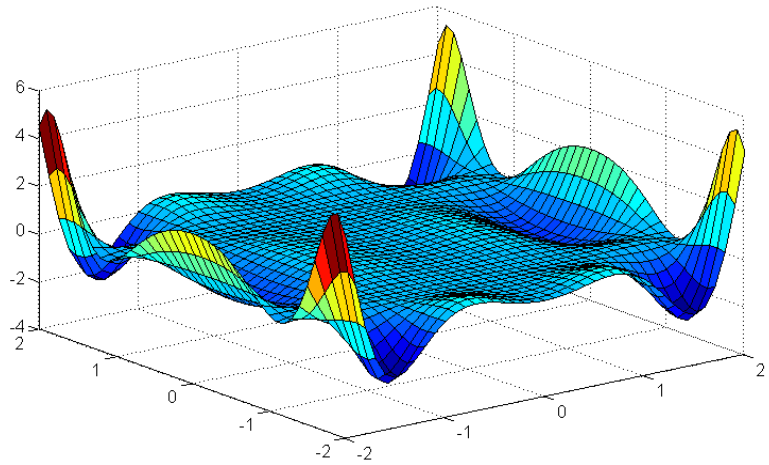


Рис. П.2. Пример визуализации графика функции двух переменных с помощью команды `surf`.

Команда **shading flat** позволяет убрать каркасные линии, а команда **shading interp** позволяет получить поверхность плавно-залитую цветом. Команда **colorbar** выводит рядом с графиком цветовую шкалу. Линии уровня можно построить с помощью команд **meshc, surfc**, которые имеют те же входные аргументы, что и `mesh, surf`.

4.3 Анимированные графики

При изучении движения точки на плоскости или в пространстве полезно не только построить траекторию точки, но и следить за движением точки по этой траектории. Для построения анимированных графиков применяются функции **comet** и **comet3**.

```
>> t=[0:0.001:10];
>> x=sin(t)./(t+1);
>> y=cos(t)./(t+1);
```

```
>> comet(x,y)
```

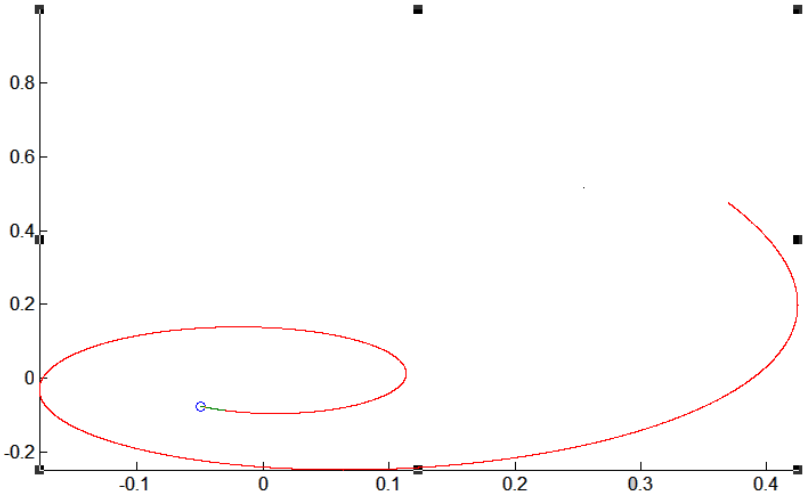


Рис.П.3. Визуализация анимированного графика (последний момент времени).

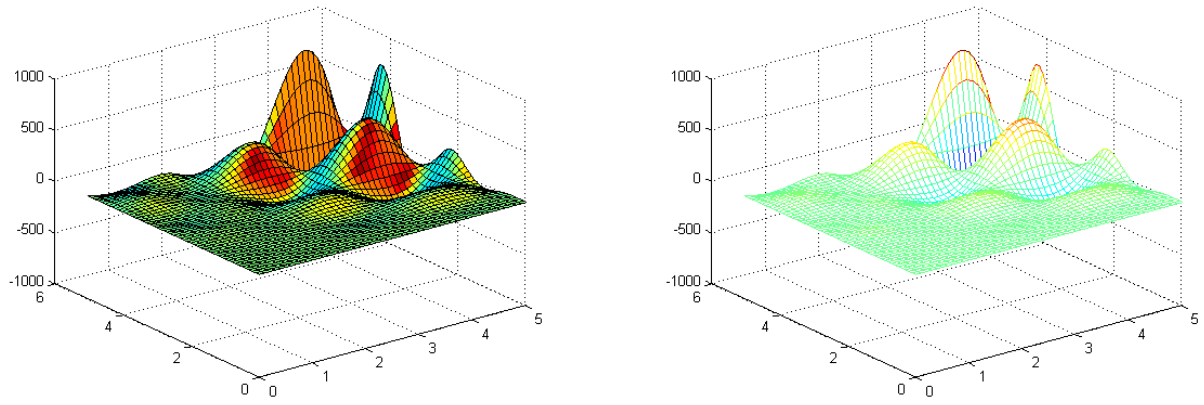
4.4 Работа с несколькими графиками

Обычно графики выводятся в специальное графическое окно с заголовком **Figure 1**. При следующем построении графика новый график выводится в то же самое окно. Matlab представляет возможность вывода: каждого графика в свое окно, нескольких графиков в одно окно, в пределах одного окна нескольких графиков, каждого в своих осях.

Следующая последовательность команд построит графики функций в отдельных окнах **Figure1** и **Figure2**, причем последнее окно будет текущим и все последующие изменения будут соответственно отражаться именно на этом окне. Чтобы сделать текущим другое окно следует два раза щелкнуть по нему мышкой и вернуться к продолжению команд:

```
>> Z=3*sin(2*X).*cos(3*Y).*(1-X.^2).*Y.^2;
>> figure
>> surfl(X,Y,Z)
>> figure
>> mesh(X,Y,Z)
```

Результат на экране (рисунок П.4):



а *б*

Рис. П.4. Графический вывод функции двух переменных в различные графические окна (*а* – с помощью команды `surfl`, *б* – с помощью команды `mesh`).

Matlab позволяет разместить в графическом окне несколько осей и вывести на них различные графики. Для это можно воспользоваться функцией **`subplot`**, которая располагает оси в виде матрицы с тремя параметрами: число подграфиков по вертикали, горизонтали, номер подграфика. Следующий пример иллюстрирует использование этой функции.

```
>> [X,Y]=meshgrid(-1:0.05:1,0:0.05:1);
>> Z=4*sin(2*pi*X).*cos(1.5*Y*pi).*(1-X.^2).*Y.*(1-Y);
>> subplot(2,2,1)
>> mesh(X,Y,Z)
>> title('mesh')
>> subplot(2,2,2)
>> surf(X,Y,Z)
>> title('surf')
>> subplot(2,2,3)
>> meshc(X,Y,Z)
>> title('meshc')
>> subplot(2,2,4)
>> surfl(X,Y,Z)
>> shading interp
>> title('surfl')
>> colormap (cool)
```

Результат показан на экране (рисунок П. 5.):

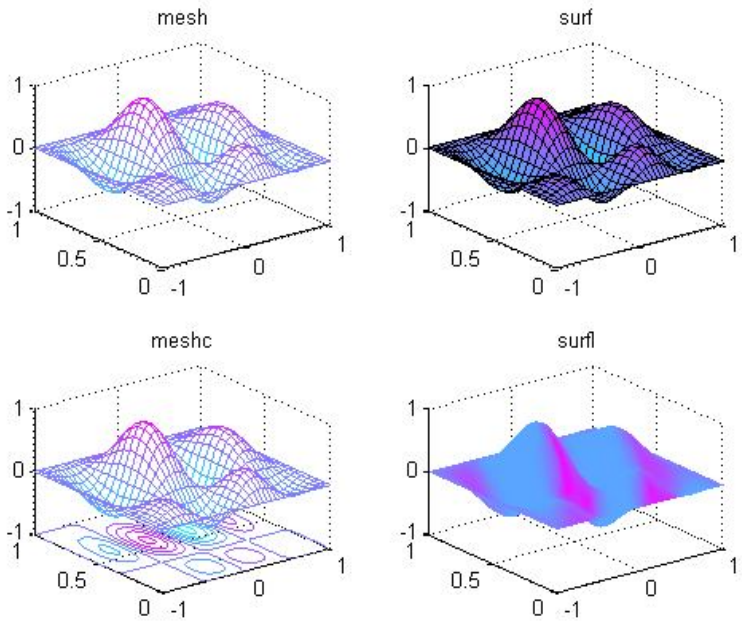


Рис. П.5. Вывод функции двух переменных на одно графическое окно (с помощью различных функций).

5 Диаграммы и гистограммы

Наглядным способом представления векторных и матричных данных являются диаграммы и гистограммы. Значение элемента вектора пропорционально высоте столбика диаграммы или площади сектора диаграммы. Гистограммы используются для получения информации о распределении данных по заданным интервалам.

Отображение вектора в виде столбчатой диаграммы осуществляется функцией **bar**.

```
>> data=[2.2 2.7 3.2 3.7 2.5 3.3 3.1 1.5 1.2 1.2];
>> bar(data)
```

Результат на экране (рисунок П.6).

Разметку горизонтальной оси можно задать вектором с возрастающими значениями, что учитывается в первом аргументе функции **bar**. Выбор ширины столбцов осуществляется заданием третьего дополнительного аргумента. Например,

```
>> t=-1:0.1:1;
>> x=sin(t).*exp(t);
```

```
>> bar(t,x,1.0)
```

Результат на экране (рисунок П.7).

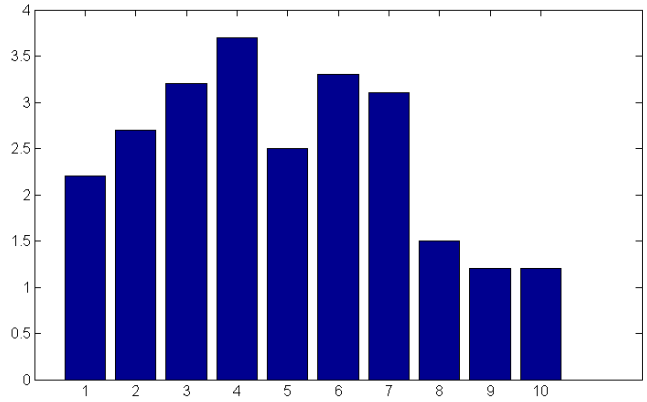


Рис. П.6. Визуализация столбчатой диаграммы.

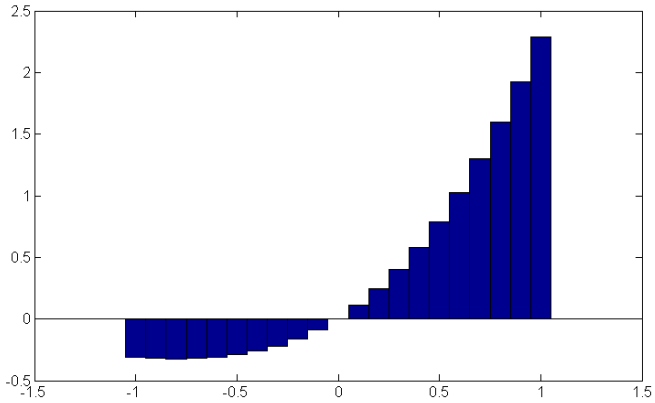


Рис. П.7. Визуализация столбчатой диаграммы с учетом вариации параметров.

Для учета вклада каждого из элементов вектора в общую сумму его элементов оказываются полезными круговые диаграммы:

```
>> data=[1.2 1.7 2.2 2.4 2.5 1.3 1.1 0.2];
>> pie(data)
```

Результат на экране (рисунок П.8).

Визуализация векторных данных может быть осуществлена при помощи **pie3** и **bar3**, которые строят трехмерные круговые и столбчатые диаграммы.

Для получения наглядного представления о распределении данных служит функция **hist**.

```
>> data=randn(100000,1);
```

```
>> hist(data,50)
```

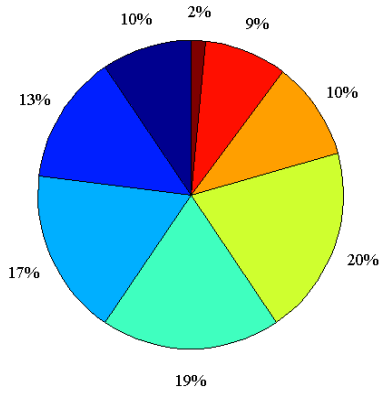


Рис. П.8. Визуализация круговой диаграммы.

Данная последовательность команд заполняет вектор `data` числами, распределенными по нормальному закону, разбивает интервал, которому они принадлежат, на пятьдесят равных частей и строят гистограмму попадания в каждый из интервалов (рисунок П.9).

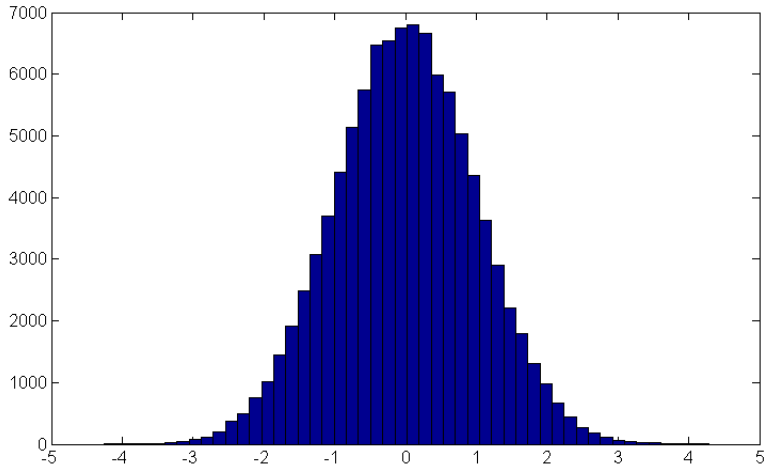


Рис. П.9. Визуализация гистограммы.

Предположим в матрице DATA, состоящей из четырех строк и трех столбцов, содержатся результаты измерений трех величин за четыре момента времени. Для построения столбчатой диаграммы используется команда **`bar`**:

```
>> DATA=[1.3 1.5 1.05;2.7 2.5 3.1;2 2.9 2.3;5.2 5.7 5.1];
>> bar(DATA)
```

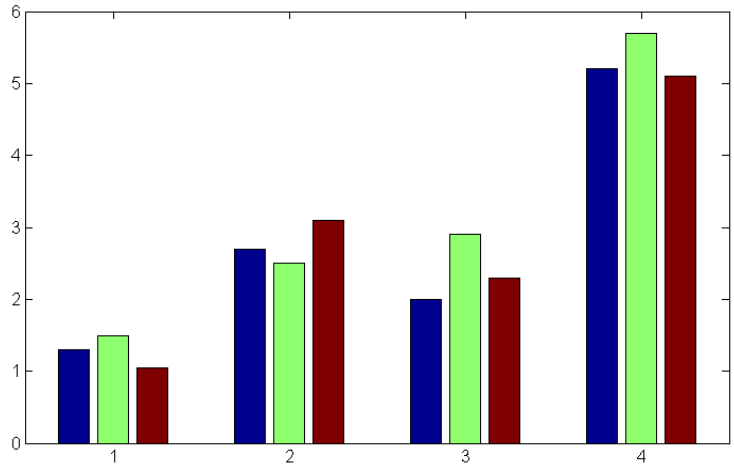


Рис. П.10. Визуализация столбчатой диаграммы (при наличии нескольких измерений).

Индивидуальные задания

№ варианта	Задание 1
1	Вычислить минимальный элемент произвольно введенного массива C и его номер среди элементов $C_i<0$.
2	Для матрицы $A(N,M)$ заменить элементы выше главной диагонали и на ней: отрицательные на ноль, положительные – на 1.
3	Вычислить минимальный элемент произвольно введенного массива C и его порядковый номер среди элементов $C_i>0$.
4	Для матрицы $A(N,M)$ заменить элементы ниже главной диагонали и на ней: отрицательные на 2, положительные – на -2.
5	Вычислить среднее геометрическое элементов произвольно введенного массива Y таких, что $Y_i>0$.
6	Для матрицы $A(N,M)$ найти максимальный и минимальный элемент из диагональных и поменять их местами.
7	Вычислить максимальный элемент произвольно введенного массива C и его порядковый номер среди элементов $C_i>0$.
8	Для матрицы $A(N,M)$ поменять первый столбец с последним, второй с предпоследним и т. д.
9	Вычислить среднее арифметическое элементов произвольно

	введенного массива S таких, что $S_i>0$.
10	Для матрицы $A(N,M)$ найти максимальный и минимальный элемент из элементов, расположенных под главной диагональю и поменять их местами.
11	Вычислить сумму и количество элементов произвольно введенного массива M таких, что $M_i>0$.
12	Для матрицы $A(N,M)$ вычислить сумму и количество положительных элементов, кратных 3.
13	Вычислить среднее арифметическое элементов произвольно введенного массива S таких, что $0<S_i<10$.
14	Для матрицы $A(N,M)$ вычислить сумму и количество положительных элементов, кратных 3.
15	Найти максимальный и минимальный элемент произвольно введенного массива и поменять их местами.
16	В каждой строке матрицы $A(N,M)$ вычислить среднее геометрическое элементов этой строки таких, что $0<a_{i,j}<10$.
17	Вычислить минимальный элемент произвольно введенного массива C и его порядковый номер среди элементов $C_i>0$.
18	Для матрицы $A(N,M)$ поменять все элементы главной диагонали на противоположные по знаку.
19	Вычислить среднее арифметическое элементов произвольно введенного массива L таких, что $L_i>0$.
20	Для матрицы $A(N,M)$ найти максимальный и минимальный элемент из диагональных и поменять их местами.

Задание 2. Построить графики функций одной переменной на отрезке $[0.01,2\pi]$ $f(x)=a\dfrac{\sin bx}{x}$, $g(x)=c\cdot e^{-dx}\cdot\cos fx$. Выведите графики различными способами: *а*) в отдельные графические окна, *б*) в одно окно на одни оси, *в*) в

одно окно на отдельные оси. Дайте заголовки, разместите подписи, легенду, используйте различные цвета, стили линий и типы маркеров, нанесите сетку.

№ варианта	Параметры	№ варианта	Параметры
1, 11	$a=1, b=1, c=1,$ $d=1, f=1$	2, 12	$a=0.5, b=2, c=1,$ $d=0.5, f=2$
3, 13	$a=0.5, b=1.5, c=0.5,$ $d=1, f=1$	4, 14	$a=1.8, b=1, c=1.2,$ $d=1, f=1$
5, 15	$a=1.9, b=1, c=0.8,$ $d=1, f=1$	6, 16	$a=1, b=2, c=1,$ $d=1, f=2$
7, 17	$a=0.5, b=1.3, c=2,$ $d=1, f=1.6$	8, 18	$a=3, b=2, c=1.2,$ $d=0.9, f=2$
9, 19	$a=1.6, b=1.5, c=5,$ $d=1, f=1.8$	10, 20	$a=1, b=3, c=0.9,$ $d=4, f=1.4$

Задание 3. Визуализируйте график функции двух переменных на прямоугольной области определения: $z(x,y)=\left(\sin ax^2+\cos by^2\right)^c$, $x\in[-1,1], y\in[-1,1]$. Выведите графики двумя различными способами (по выбору) в одно графическое окно. Параметры a,b,c взять из таблицы к заданию 2.

Задание 4. Фирма, занимающаяся реализацией сезонного товара, имеет следующие средние доходности $S=(50,45,51,55,79,90,110,100,92,76,79,88)$. Каждый элемент вектора S соответствует прибыли, полученной фирмой за соответствующий календарный месяц. Построить столбчатую и круговую диаграммы, интерпретирующие модельное распределение для данного примера (диаграммы вывести в одно графическое окно).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ануфриев И.Е., Смирнов А.Б., Смирнова Е.Н. Matlab 7. – СПб.: БХВ – Петербург, 2005. – 1104 с.

2. Бахвалов С.В., Жидков Н. П., Кобельков Г.М. Численные методы. – М.: Бином. Лаборатория знаний, 2011.– 640 с.

3. Вержбицкий В.М. Численные методы. (Линейная алгебра и нелинейные уравнения).– М.: Книга по требованию, 2005. – 432 с.

4. Вержбицкий В.М. Численные методы. (Математический анализ и обыкновенные дифференциальные уравнения): Учебное пособие для вузов/В.М. Вержбицкий. – 2-е изд., испр. - М. : ООО «Издательский дом «ОНИКС 21 век», 2005. – 400 с.

5. Демидович Б.П., Марон И.А. Основы вычислительной математики. – СПб.: Лань, 2007. – 664с.

6. Иглин С.П. Математические расчеты на базе Matlab. – СПб.: БХВ-Петербург, 2005.– 640 с.

7. Кетков Ю., Кетков А., Шульц М. MATLAB 7: программирование, численные методы. -СПб.: БХВ - Петербург, 2005. – 752 с.

8. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах. – Учебное пособие. 3-е изд., стер. – СПб.: Лань, 2009. – 368 с.

9. Лапчик, М.П. Численные методы: учеб. пособие для вузов/ М.П. Лапчик,. М.И. Рагулина, Е.К. Хеннер – М.: Академия , 2009. – 384 с.

10. Мэтьюз Д.Г., Куртис Ф.Д. Численные методы. Использование Matlab / пер. с англ. – М.: Вильямс, 2001. – 720 с.

11. Петров И.Б., Лобанов А.И. Лекции по вычислительной математике. – М.: Бином. Лаборатория знаний Интуит, 2010. – 523 с.

12. Самарский А.А. Введение в численные методы: Учеб. пособие. – СПб.: Лань, 2009. – 288 с.

СОДЕРЖАНИЕ

Анна Геннадьевна Масловская,
профессор кафедры математического анализа и моделирования АмГУ,
д-р физ.-мат. наук, доцент;

Анна Владимировна Павельчук,
старший преподаватель кафедры общей математики и информатики АмГУ

Численные методы: использование инструментальных средств и реализация алгоритмов на базе ППП Matlab. *Учебное пособие.*

Заказ 742.