

Алексеев Е. Р., Чеснокова О. В.

Решение задач вычислительной математики в пакетах Mathcad 12, MATLAB 7, Maple 9

- ✓ Алгоритмы вычислительной математики
- ✓ Задачи линейной алгебры
- ✓ Обработка экспериментальных данных
- ✓ Нелинейные уравнения и системы
Численное интегрирование и
дифференцирование
- ✓ Обыкновенные
дифференциальные уравнения
- ✓ Уравнения в частных
производных
- ✓ Интегральные уравнения
- ✓ Программирование
численных методов
в математических
пакетах
- ✓ Решение
задач
вычислительной
математики в
Mathcad,
MATLAB, Maple

САМОУЧИТЕЛЬ



004.9
32.973.26-018.2
А47

Подписано в печать 26.02.2006. Формат 70×90¹/₁₆. Гарнитура «Баскервиль». Печать офсетная. Усл. печ. л. 35,96. Тираж 3000 экз. Зак. № 6405

Алексеев Е. Р., Чеснокова О. В.

Решение задач вычислительной математики в пакетах Mathcad 12, MATLAB 7, Maple 9/Алексеев Е. Р., Чеснокова О. В. – М. : НТ Пресс, 2006. – 496 с. : ил. – (Самоучитель).

ISBN 5-477-00208-5

Книга адресована читателям, желающим ознакомиться с численными методами решения прикладных задач и их реализацией в современных математических пакетах Mathcad 12, MATLAB 7, Maple 9.

В доступной форме описаны вычислительные методы решения задач линейной алгебры, обработки результатов эксперимента, интегрирования и дифференцирования. Подробно рассмотрены способы решения нелинейных уравнений и систем. Особое внимание уделено методам решения обыкновенных дифференциальных уравнений, интегральных уравнений и разностным схемам решения уравнений математической физики.

Особенностью книги является наличие блок-схем и программ всех рассматриваемых вычислительных методов. Кроме того, в книге рассмотрены возможности популярных математических пакетов для решения задач вычислительной математики.

Для студентов, аспирантов и преподавателей технических вузов, инженеров, научно-технических работников и всех, применяющих численные методы при решении прикладных задач.

УДК 004.9
ББК 32.973.26-018.2

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельца авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно остается, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможный ущерб любого вида, связанный с применением содержащихся здесь сведений.

Все торговые знаки, упомянутые в настоящем издании, зарегистрированы. Случайное неправильное использование или пропуск торгового знака или названия его законного владельца не должно рассматриваться как нарушение прав собственности.

© Алексеев Е. Р., Чеснокова О. В., 2006
© Оформление «НТ Пресс», 2006

Оглавление

Предисловие	9
Введение	9
Глава 1 ▼	
Решение задач линейной алгебры	11
1.1. Основные определения алгебры матриц	11
1.2. Решение систем линейных уравнений	13
1.2.1. Метод Гаусса	15
1.2.2. Вычисление обратной матрицы методом Гаусса	20
1.2.3. Вычисление определителя методом Гаусса	24
1.2.4. Метод квадратных корней	25
1.2.5. Схема Халецкого	27
1.2.6. Методы LU- и QR-разложения	29
1.2.7. Метод ортогонализации	32
1.2.8. Метод прогонки	34
1.2.9. Метод итерации	36
1.2.10. Метод Зейделя	38
1.2.11. Метод релаксации	41
1.2.12. Метод градиента	43
1.3. Собственные значения и собственные векторы	45
1.3.1. Метод скалярных произведений	46
1.3.2. Метод итерации	48

1.4. Решение задач линейной алгебры в Mathcad	50
1.4.1. Задачи алгебры матриц	53
1.4.2. Решение систем линейных уравнений	58
1.5.3. Собственные значения, собственные векторы, нормы, числа обусловленности матриц	67
1.4.3. Реализация численных методов решения систем линейных уравнений	73
1.5. Решение задач линейной алгебры с использованием Maple	77
1.6. Возможности MATLAB для решения задач линейной алгебры	85
1.7. Советы по применению математических пакетов при решении задач линейной алгебры	107
1.8. Задачи для самостоятельного решения	107

Глава 2 ▾

Интерполяция функций	110
2.1. Интерполирование функций	110
2.1.1. Канонический полином	111
2.1.2. Полином Ньютона	113
2.1.3. Полином Лагранжа	115
2.1.4. Интерполяция сплайнами	117
2.1.5. Линейная интерполяция	122
2.2. Решение задач интерполяции в Mathcad	124
2.3. Реализация интерполяции в MATLAB	128
2.4. Решение задачи интерполяции средствами Maple	134
2.5. Советы по использованию математических пакетов при решении задач интерполяции	137

Глава 3 ▾

Метод наименьших квадратов	138
3.1. Подбор параметров экспериментальной зависимости методом наименьших квадратов	138
3.1.1. Подбор коэффициентов линейной зависимости	140
3.1.2. Подбор коэффициентов полинома k-й степени	141
3.1.3. Подбор коэффициентов функции $Y = ax^b \cdot e^{cx}$	142
3.1.4. Функции, приводимые к линейной	143
3.1.5. Уравнение регрессии и коэффициент корреляции	145
3.1.6. Криволинейная корреляция	146
3.2. Решение задач аппроксимации в Mathcad	147
3.3. Реализация метода наименьших квадратов в MATLAB	155

3.4. Решение задач аппроксимации средствами Maple	160
3.5. Советы по применению математических пакетов при подборе зависимостей методом наименьших квадратов	163
3.6. Задания для самостоятельного решения	164

Глава 4 ▼

Методы решения нелинейных уравнений	167
4.1. Локализация (отделение) корней	168
4.2. Метод половинного деления (дихотомии)	170
4.3. Метод хорд (пропорциональных частей)	172
4.4. Метод касательных (метод Ньютона)	174
4.5. Комбинированный метод	177
4.6. Метод простой итерации (последовательных приближений)	180
4.7. Вычисление комплексных корней	183
4.8. Методы решения алгебраических уравнений	183
4.8.1. Вычисление значений многочлена	184
4.8.2. Деление многочлена на линейный множитель	184
4.8.3. Вычисление производных многочлена	186
4.8.4. Метод спуска	188
4.8.5. Метод парабол	191
4.9. Решение нелинейных уравнений в Mathcad	194
4.9.1. Алгебраические уравнения	194
4.9.2. Трансцендентные уравнения	199
4.9.3. Реализация численных методов решения нелинейных уравнений ...	202
4.10. Использование Maple для решения нелинейных уравнений	206
4.10.1. Алгебраические уравнения	207
4.10.2. Трансцендентные уравнения	208
4.10.3. Реализация численных методов решения алгебраических уравнений	211
4.11. Возможности MATLAB для решения нелинейных уравнений	215
4.12. Советы по использованию математических пакетов при решении нелинейных уравнений	217
4.12. Задания для самостоятельного решения	218

Глава 5 ▼

Приближенное решение систем нелинейных уравнений	219
5.1. Метод Ньютона	219
5.2. Метод итерации	222

5.3. Метод градиента	224
5.4. Решение нелинейных уравнений в Mathcad	227
5.5. Использование Maple для решения нелинейных уравнений	238
5.6. Возможности MATLAB для решения нелинейных уравнений	241
5.7. Советы по использованию математических пакетов при решении нелинейных уравнений	246
5.8. Задания для самостоятельного решения	247

Глава 6 ▼

Численное интегрирование и дифференцирование функций

249

6.1. Методы численного интегрирования	249
6.1.1. Интегрирование по методу прямоугольников	250
6.1.2. Интегрирование по методу трапеций	253
6.1.3. Интегрирование по методу Симпсона	255
6.1.4. Квадратурные формулы Ньютона–Котеса	256
6.1.5. Квадратурные формулы Чебышева	260
6.1.6. Квадратурные формулы Гаусса	263
6.1.7. Вычисление интеграла с заданной точностью	265
6.2. Численное дифференцирование	266
6.3. Интегрирование и дифференцирование в Mathcad	268
6.4. Вычисление интегралов и производных в MATLAB	276
6.5. Возможности Maple для интегрирования и дифференцирования ..	281
6.6. Советы по использованию математических пакетов при интегрировании и дифференцировании	286
6.7. Задания для самостоятельного решения	287

Глава 7 ▼

Решение обыкновенных дифференциальных уравнений и систем

289

7.1. Общие сведения о дифференциальных уравнениях	289
7.2. Численные методы решения дифференциальных уравнений и их реализация в MATLAB	291
7.2.1. Решение дифференциальных уравнений методом Эйлера	292
7.2.2. Решение дифференциальных уравнений при помощи модифицированного метода Эйлера	296
7.2.3. Решение дифференциальных уравнений методами Рунге–Кутты ...	299
7.2.4. Решение дифференциальных уравнений методом прогноза-коррекции Адамса	306

7.2.5.	Решение дифференциальных уравнений методом Милна	309
7.2.6.	Решение систем дифференциальных уравнений	313
7.3.	Встроенные функции MATLAB для решения дифференциальных уравнений	314
7.4.	Решение дифференциальных уравнений в Mathcad	317
7.4.1.	Встроенные функции Mathcad для решения обыкновенных уравнений и систем	317
7.4.2.	Решение дифференциальных уравнений первого порядка	321
7.4.3.	Решение систем дифференциальных уравнений	326
7.4.4.	Решение жестких задач	328
7.4.5.	Решение дифференциальных уравнений высших порядков	333
7.4.6.	Решение двухточечных краевых задач в Mathcad	335
7.5.	Решение дифференциальных уравнений в Maple	337
7.5.1.	Решение дифференциальных уравнений	337
7.5.2.	Решение систем обыкновенных дифференциальных уравнений	338
7.6.	Советы по применению математических пакетов при решении дифференциальных уравнений	340
7.7.	Задания для самостоятельного решения	341

Глава 8 ▼

Решение дифференциальных уравнений в частных

производных	343

8.1.	Общие сведения о дифференциальных уравнениях в частных производных	343
8.2.	Решение уравнений в частных производных методом разделения переменных	346
8.3.	Использование метода сеток для решения уравнения в частных производных и его реализация в Mathcad	351
8.3.1.	Использование метода сеток для решения параболических уравнений в Mathcad	352
8.3.2.	Использование метода сеток для решения гиперболических уравнений в Mathcad	363
8.3.3.	Использование метода сеток для решения эллиптических уравнений в Mathcad	366
8.4.	Использование встроенных функций Mathcad для решения уравнений в частных производных	369
8.4.1.	Решение параболических уравнений	369
8.4.2.	Решение гиперболических уравнений	373
8.4.3.	Решение эллиптических уравнений	376

8.5. Использование MATLAB для решения уравнений в частных производных	378
8.5.1. Решение эллиптических уравнений	379
8.5.2. Решение гиперболических и параболических уравнений	400
8.6. Использование Maple для решения уравнений математической физики	404
8.7. Советы по использованию математических пакетов при решении дифференциальных уравнений в частных производных	405
8.8. Задания для самостоятельного решения	406

Глава 9 ▼

Решение интегральных уравнений	408
9.1. Общие сведения об интегральных уравнениях	408
9.2. Численные методы решения интегральных уравнений	410
9.2.1. Использование метода конечных сумм при решении интегральных уравнений Фредгольма	411
9.2.2. Использование метода конечных сумм при решении интегральных уравнений Вольтерра	413
9.3. Решение интегральных уравнений в Mathcad	419
9.4. Решение интегральных уравнений в MATLAB	421
9.5. Решение интегральных уравнений в Maple	423
9.6. Советы по применению математических пакетов при решении интегральных уравнений	424
9.7. Задания для самостоятельного решения	424

Приложение 1 ▼ Введение в Mathcad	425

Приложение 2 ▼ Введение в MATLAB	452

Приложение 3 ▼ Введение в Maple	466

Заключение	485

Список используемой литературы	486

Предметный указатель	488

Предисловие

Цель этой книги – дать читателю-непрофессионалу в области математики инструмент для решения задач вычислительной математики, возникающих в его практической деятельности. Наряду с изложением теоретического материала и блок-схем алгоритмов вычислительной математики приводится большое количество примеров решения задач в популярных математических пакетах.

Все примеры проверены авторами в Mathcad 12, MATLAB 7, Maple 9. Если читатель обнаружит ошибки, просим сообщить по адресу teacher@teacher.dn-ua.com.

Введение

Существует большое количество книг по вычислительной математике, ориентированных на математиков-профессионалов [5, 9, 10, 11, 16], и еще большее количество книг по математическим пакетам [1, 4, 8, 12–14], авторы которых считают, что читатель хотя бы в общих чертах знаком с курсом вычислительной математики, и поэтому описывают только возможности пакетов. Данная книга отличается от остальных тем, что в ней доступным языком описаны очень сложные алгоритмы вычислительной математики. Ее авторы попытались соединить учебник по численным методам с решением вычислительных задач в популярных пакетах.

Авторы надеются, что читатель обладает навыками пользователя персонального компьютера под управлением Windows, знает высшую математику в объеме курса технического вуза. Основы работы в Mathcad, MATLAB и Maple кратко описаны в приложениях.

Особенностью этой книги является наличие блок-схем алгоритмов и программ для каждого описанного в книге численного метода. Также книга содержит большое количество практических примеров решения практических задач с использованием математических пакетов.

Книга состоит из девяти глав и трех приложений. Каждая глава содержит численные методы, блок-схемы алгоритмов и возможности пакетов для решения вычислительных задач. Подпрограммы, реализующие методы решения задач, приведены на встроенных языках программирования одного из математических пакетов Mathcad, MATLAB или Maple. В конце каждой главы приведены советы авторов по применению того или иного пакета для решения рассматриваемых задач.

Первая глава посвящена численным методам решения задач линейной алгебры. В частности, подробно рассмотрены способы решения систем линейных

алгебраических уравнений, вычисления определителя и обратной матрицы. Особое внимание уделено алгоритмам нахождения собственных значений и векторов.

Во второй главе рассматривается обработка результатов эксперимента с помощью полиномиальной и сплайн-интерполяции.

Третья глава содержит описание метода наименьших квадратов и его применение для обработки экспериментальных данных.

В четвертой главе рассказывается о методах решения нелинейных уравнений. Особое внимание уделено нахождению действительных и комплексных корней алгебраических уравнений итерационным методом, а также рассматривается случай комплексных корней в трансцендентных уравнениях.

Численные методы, описанные в пятой главе, позволяют решать системы нелинейных уравнений.

Шестая глава посвящена методам численного интегрирования и дифференцирования.

В седьмой главе представлено много методов решения обыкновенных дифференциальных уравнений: одношаговые методы Рунге–Кутты (в том числе и методы автоматического подбора шага интегрирования), многошаговые методы прогноза-коррекции Адамса и Милна.

В восьмой главе описаны метод разделения переменных и метод сеток решения дифференциальных уравнений в частных производных. Подробно изложены возможности последних версий пакета Mathcad для решения уравнений математической физики. Кроме того, в этой главе рассмотрена графическая среда *pdetool* для решения дифференциальных уравнений в частных производных методом конечных элементов в MATLAB.

Девятая глава описывает метод конечных сумм для решения интегральных уравнений.

Все главы снабжены заданиями для самостоятельного решения, что позволит использовать книгу в учебном процессе.

Соглашения, принятые в книге

- Функции и имена переменных выделены моноширинным шрифтом;
- определения, новые понятия – *курсивом*;
- названия пунктов меню, сочетания клавиш, элементы программы – **полужирным шрифтом**;
- последовательность команд, выполняемых из меню, отмечается стрелочкой, например **File ➤ Save** (Файл ➤ Сохранить);
- клавиши, нажимаемые одновременно, записываются со знаком + (плюс), например **Ctrl+S**.



Здесь содержатся важные советы и примечания, относящиеся к данной теме. Они позволят лучше понять информацию, изложенную в тексте.

Глава 1

Решение задач линейной алгебры

В этой главе речь идет об основных методах решения задач линейной алгебры. Приведено подробное математическое описание и даны блок-схемы решения систем линейных алгебраических уравнений, вычисления определителей и обратных матриц, а также нахождения собственных значений и собственных векторов.

1.1. Основные определения алгебры матриц

Напомним основные определения алгебры матриц. Если $m \cdot n$ выражений расставлены в прямоугольной таблице из m строк и n столбцов, то говорят о *матрице* размера $m \times n$. Выражения a_{ij} называют элементами матрицы. Элементы a_{ii} ($i = 1, \dots, n$), стоящие в таблице на линии, проходящей из левого верхнего угла в правый нижний угол квадрата $n \times n$, образуют *главную диагональ матрицы*. Матрица размером $m \times n$ ($m \neq n$) называется *прямоугольной*. В случае если $m = n$, то матрицу называют *квадратной матрицей* порядка n . В частности, матрица типа $1 \times n$ — это *вектор-строка*, а матрица размером $m \times 1$ является *вектором-столбцом*. Число (скаляр) можно рассматривать как матрицу типа 1×1 .

Квадратная матрица $A = \{a_{ij}\}$ размером $n \times n$ называется:

- *нулевой*, если все ее элементы равны нулю: $A = \{a_{ij} = 0\}$;
- *верхней треугольной*, если все элементы, расположенные ниже главной диагонали, равны нулю: $A = \{a_{ij} = 0, \text{ для всех } i > j\}$;

- *нижней треугольной*, если все элементы, расположенные выше главной диагонали, равны нулю: $A = \{a_{ij} = 0, \text{ для всех } i < j\}$;
- *диагональной*, если все элементы, кроме элементов главной диагонали, равны нулю: $A = \{a_{ij} = 0, \text{ для всех } i \neq j\}$;
- *единичной*, если элементы главной диагонали равны единице, а все остальные – нулю: $A = \{a_{ij} = 0, \text{ для всех } i \neq j \text{ и } a_{ij} = 1, \text{ для всех } i = j\}$;

С квадратной матрицей связано понятие *определителя*, или *детерминанта*. Определителем матрицы A является число $\det A$, или Δ , вычисляемое по правилу:

$$\det A = \sum_{(i_1, i_2, \dots, i_n)} (-1)^\lambda a_{1i_1} a_{2i_2} \dots a_{ni_n},$$

где сумма распределена на всевозможные перестановки $(i_1, i_2, \dots, i_n)$ элементов $1, 2, \dots, n$ и, следовательно, содержит $n!$ слагаемых, причем $\lambda = 0$, если перестановка четная, и $\lambda = 1$, если перестановка нечетная. Квадратная матрица называется *невырожденной*, если ее определитель отличен от нуля. В противном случае матрица называется *вырожденной*, или *сингулярной*.

С матрицами можно проводить операции сравнения, сложения и умножения. Две матрицы $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$ считаются *равными*, если они одного типа, то есть имеют одинаковое число строк и столбцов, и соответствующие элементы их равны $\{a_{ij}\} = \{b_{ij}\}$. *Суммой двух матриц* $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$ одинакового типа называется матрица $C = \{c_{ij}\}$ того же типа, элементы которой равны сумме соответствующих элементов матриц $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$, то есть $\{c_{ij}\} = \{a_{ij} + b_{ij}\}$. Разность матриц определяется аналогично. *Произведением числа a на матрицу* $A = \{a_{ij}\}$ (или произведением матрицы на число) называется матрица, элементы которой получены умножением всех элементов матрицы A на число a , то есть $A = aA = A\{a \cdot a_{ij}\}$. *Произведением матриц* $A = \{a_{ij}\}$ размерностью $m \times n$ и $B = \{b_{ij}\}$ размерностью $p \times s$ является матрица C размерностью $m \times s$, каждый элемент которой можно представить формулой $\{c_{ij}\} = \{a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}, i = 1, \dots, m, j = 1, \dots, s\}$. Таким образом, произведение матриц $A \cdot B$ имеет смысл тогда и только тогда, когда количество строк матрицы A совпадает с количеством столбцов матрицы B . Кроме того, произведение двух матриц не обладает переместительным законом, то есть $A \cdot B \neq B \cdot A$. В тех случаях, когда $A \cdot B = B \cdot A$, матрицы A и B называются *перестановочными*.

Если в матрице $A = \{a_{ij}\}$ размерностью $m \times n$ заменить строки соответствующими столбцами, то полученная *транспонированная* матрица $A^T = \{a_{ji}\}$. В частности, для вектора-строки транспонированной матрицей является вектор-столбец.

Обратной матрицей по отношению к данной матрице $A = \{a_{ij}\}$ размерностью $n \times n$ называется матрица $A^{-1} = \{A_{ij}\}$ того же типа, которая, будучи умноженной

Система линейных уравнений может не иметь ни одного решения, и тогда она называется *несовместной*. Например, в системе

$$\begin{aligned}x_1 + x_2 &= 1, \\x_1 + x_2 &= 3\end{aligned}$$

левые части уравнений совпадают, а правые различны, поэтому никакие значения x_1 и x_2 не могут удовлетворить обоим уравнениям сразу.

Если же система линейных уравнений обладает решением, то она называется *совместной*. Совместная система называется *определенной*, если она обладает одним-единственным решением, и *неопределенной*, если решений больше, чем одно. Так, система

$$\begin{aligned}x_1 + 2x_2 &= 7, \\x_1 + x_2 &= 6\end{aligned}$$

определена и имеет единственное решение: $x_1 = 5$ и $x_2 = 1$, а система уравнений

$$\begin{aligned}3x_1 - x_2 &= 1, \\6x_1 - 2x_2 &= 2\end{aligned}$$

не определена, так как имеет бесконечное множество решений вида $x_1 = k$ и $x_2 = 3k - 1$, где число k произвольно.

Совокупность всех решений неопределенной системы уравнений называется ее *общим решением*, а какое-то одно конкретное решение – *частным*. Частное решение, полученное из общего при нулевых значениях свободных переменных, называется *базисным*.

При определении совместности систем уравнений важную роль играет понятие ранга матрицы. Пусть дана матрица A размером $n \times m$. Вычеркиванием некоторых строк или столбцов из нее можно получить квадратные матрицы k -го порядка, определители которых называются *минорами* порядка k матрицы A . Наивысший порядок не равных нулю миноров матрицы A называют *рангом* матрицы и обозначают $r(A)$. Из определения вытекает, что $r(A) \leq \min(n, m)$, $r(A) = 0$, только если матрица нулевая и $r(A) = n$ для невырожденной матрицы n -го порядка. При элементарных преобразованиях (перестановке строк матрицы, умножении строк на число, отличное от нуля, и сложении строк) ранг матрицы не изменяется. Итак, если речь идет об исследовании системы на совместность, следует помнить, что система n линейных уравнений с m неизвестными:

- несовместна, если $r(A|b) > r(A)$;
- совместна, если $r(A|b) = r(A)$, причем при $r(A|b) = r(A) = m$ имеет единственное решение, а при $r(A|b) = r(A) < m$ имеет бесконечно много решений.

1.2.1. Метод Гаусса

Пусть дана система n линейных алгебраических уравнений (СЛАУ) с n неизвестными

[illegible]

Обозначим через

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

матрицу коэффициентов системы (1.1), через

$$b = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}$$

столбец ее свободных членов и через

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

столбец из неизвестных (искомый вектор). Тогда систему (1.1) можно записать в матричном виде: $A \cdot x = b$ и составить ее расширенную матрицу $(A|b)$:

$$A' = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{pmatrix}. \quad (1.2)$$

Первый этап решения системы уравнений (1.1), называемый *прямым ходом метода Гаусса*, заключается в приведении расширенной матрицы (1.2) к *треугольному виду*. Это означает, что все элементы матрицы (1.2) ниже главной диагонали должны быть равны нулю:

$$A' = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} & b_1 \\ 0 & a_{22} & a_{23} & \dots & a_{2n} & b_2 \\ 0 & 0 & a_{33} & \dots & a_{3n} & b_3 \\ 0 & 0 & 0 & \dots & a_{4n} & b_4 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & a_{nn} & b_n \end{pmatrix}. \quad (1.3)$$

Для формирования первого столбца матрицы (1.3) необходимо из каждой строки (начиная со второй) вычесть первую, умноженную на некоторое число M .

В общем виде этот процесс можно записать так:

2-я строка = 2-я строка – $M \times$ 1-я строка,

3-я строка = 3-я строка – $M \times$ 1-я строка,

.....,

i -я строка = i -я строка – $M \times$ 1-я строка,

.....,

n -я строка = n -я строка – $M \times$ 1-я строка.

Понятно, что преобразование элементов второй строки будет происходить по формулам:

$$a_{21} = a_{21} - M \cdot a_{11}, \quad a_{22} = a_{22} - M \cdot a_{12}, \quad \dots \quad a_{2i} = a_{2i} - M \cdot a_{1i}, \quad \dots \quad a_{2n} = a_{2n} - M \cdot a_{1n},$$

$$b_2 = b_2 - M \cdot b_1.$$

Так как целью данных преобразований является обнуление первого элемента строки, то M выбирается из условия:

$$a_{21} - M \cdot a_{11} = 0,$$

Следовательно,

$$M = \frac{a_{21}}{a_{11}}.$$

Элементы третьей строки и коэффициент M можно рассчитать аналогично:

$$a_{31} = a_{31} - M \cdot a_{11}, a_{32} = a_{32} - M \cdot a_{12}, \dots a_{3i} = a_{3i} - M \cdot a_{1i}, \dots a_{3n} = a_{3n} - M \cdot a_{1n},$$

$$b_3 = b_3 - M \cdot b_1$$

и

$$a_{31} - M \cdot a_{11} = 0 \Rightarrow M = \frac{a_{31}}{a_{11}}.$$

Таким образом, преобразование элементов i -й строки будет происходить следующим образом:

$$a_{i1} = a_{i1} - M \cdot a_{11}, a_{i2} = a_{i2} - M \cdot a_{12}, \dots a_{ii} = a_{ii} - M \cdot a_{1i}, \dots a_{in} = a_{in} - M \cdot a_{1n},$$

$$b_i = b_i - M \cdot b_1.$$

Коэффициент M для i -й строки выбирается из условия

$$a_{i1} - M \cdot a_{11} = 0$$

и равен

$$M = \frac{a_{i1}}{a_{11}}.$$

После проведения подобных преобразований для всех строк матрица (1.2) примет вид:

$$A' = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ 0 & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & a_{n2} & \dots & a_{nn} & b_n \end{pmatrix}.$$

Очевидно, что если повторить описанный выше алгоритм для следующих столбцов матрицы (1.2), причем начинать преобразовывать второй столбец с третьего элемента, третий столбец – с четвертого и т.д., то в результате будет получена матрица (1.3). Алгоритм этого процесса изображен на рис. 1.1.

Заметим, что если в матрице (1.2) на главной диагонали встретится элемент a_{kk} , равный нулю, то расчет коэффициента

$$M = \frac{a_{ik}}{a_{kk}}$$

для k -й строки будет невозможен. Избежать деления на ноль можно, избавившись от нулевых элементов на главной диагонали. Для этого перед обнулением элементов в k -м столбце необходимо найти в нем максимальный по модулю

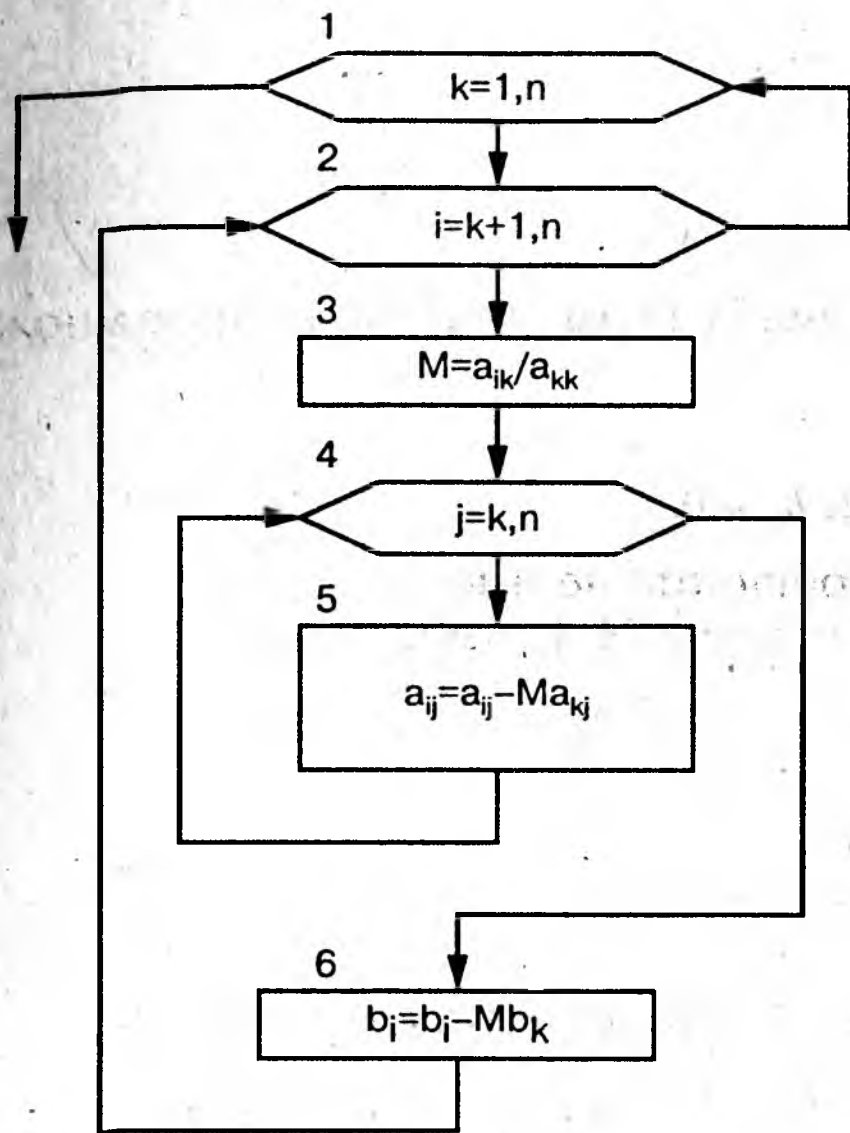


Рис. 1.1 ▼ Блок-схема алгоритма преобразования расширенной матрицы к треугольному виду

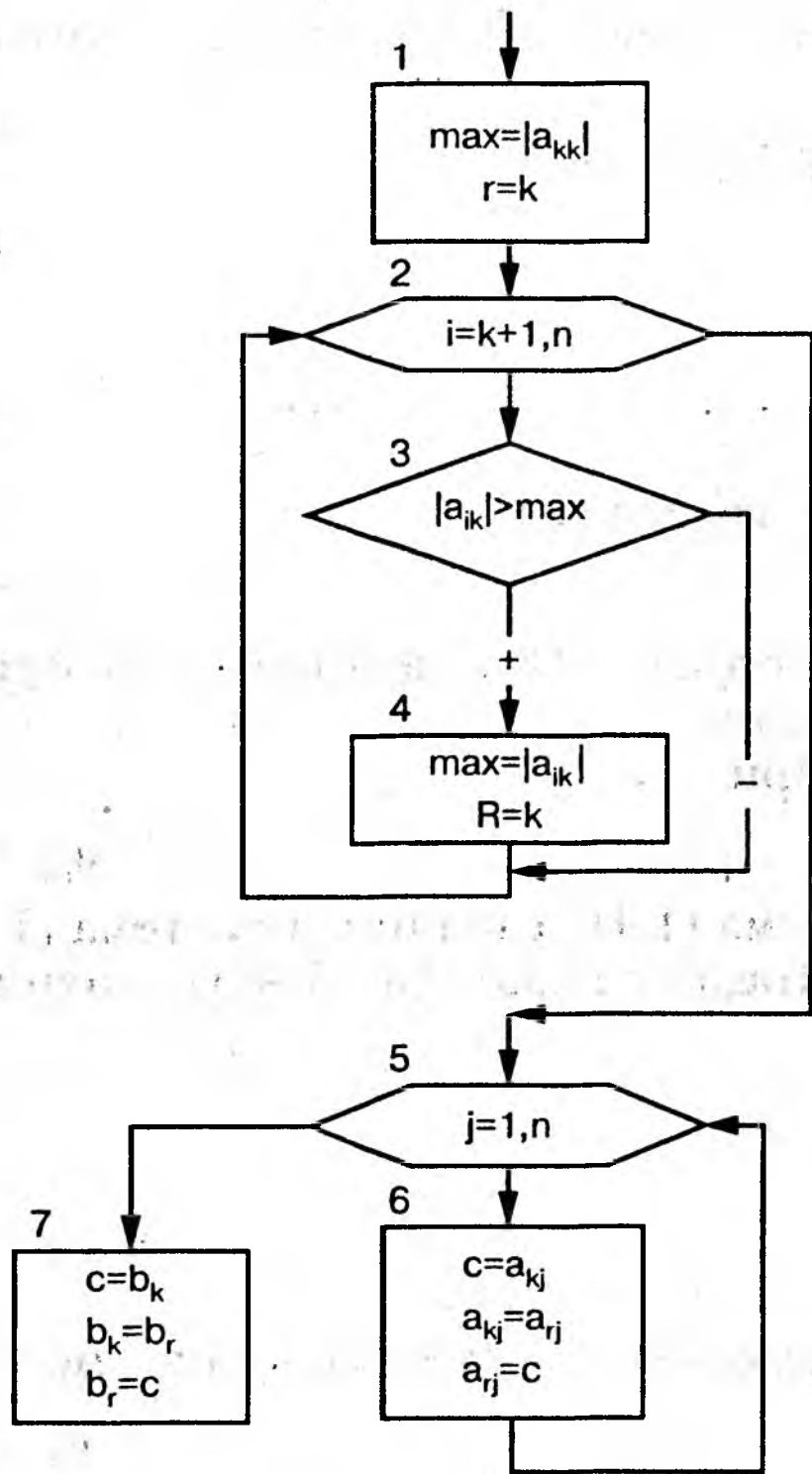


Рис. 1.2 ▼ Блок-схема алгоритма перестановки строк расширенной матрицы

элемент, запомнить номер строки, в которой он находится, и поменять ее местами с k -й. Алгоритм, отображающий эти преобразования, приведен на рис. 1.2.

В результате выполнения прямого хода метода Гаусса матрица (1.2) преобразуется в матрицу (1.3), а система уравнений (1.1) будет иметь следующий вид:

$$\left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1, \\ a_{22}x_2 + a_{13}x_3 + \dots + a_{2n}x_n = b_2, \\ a_{13}x_3 + \dots + a_{2n}x_n = b_3, \\ \dots, \\ a_{nn}x_n = b_n. \end{array} \right. \quad (1.4)$$

Решение системы (1.4) называют *обратным ходом метода Гаусса*.

Последнее n -е уравнение системы (1.4) имеет вид:

$$a_{nn} x_n = b_n.$$

Тогда если

$$a_{nn} \neq 0.$$

то

$$x_n = \frac{b_n}{a_{nn}}.$$

В случае если

$$a_{nn} = 0 \text{ и } b_n = 0,$$

то система (1.4), а следовательно, и система (1.1) имеют бесконечное множество решений.

При

$$a_{nn} = 0 \text{ и } b_n \neq 0,$$

система (1.4), а значит, и система (1.1) решения не имеет.

Предпоследнее $(n-1)$ -е уравнение системы (1.4) имеет вид:

$$a_{n-1, n-1} x_{n-1} + a_{n-1, n} x_n = b_{n-1}.$$

Значит,

$$x_{n-1} = \frac{b_{n-1} - a_{n-1, n} x_n}{a_{n-1, n-1}}.$$

Следующее $(n-2)$ -е уравнение системы (1.4) будет выглядеть так:

$$a_{n-2, n-2} x_{n-2} + a_{n-2, n-1} x_{n-1} + a_{n-2, n} x_n = b_{n-2}.$$

Отсюда имеем

$$x_{n-2} = \frac{b_{n-2} - a_{n-2, n-1} x_{n-1} - a_{n-2, n} x_n}{a_{n-2, n-2}}$$

или

$$x_{n-2} = \frac{b_{n-2} - (a_{n-2, n-1} x_{n-1} + a_{n-2, n} x_n)}{a_{n-2, n-2}} = \frac{b_{n-2} - \sum_{j=n-1}^n a_{n-2, j} x_j}{a_{n-2, n-2}}.$$

Таким образом, формула для вычисления i -го значения x будет иметь вид:

$$x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij} x_j}{a_{ii}}.$$

Алгоритм, реализующий обратный ход метода Гаусса, представлен в виде блок-схемы на рис. 1.3.

Объединив блок-схемы, изображенные на рис. 1.1, 1.2 и 1.3, получим общую блок-схему метода Гаусса (рис. 1.4). Блоки 2-6 содержат последовательный ввод

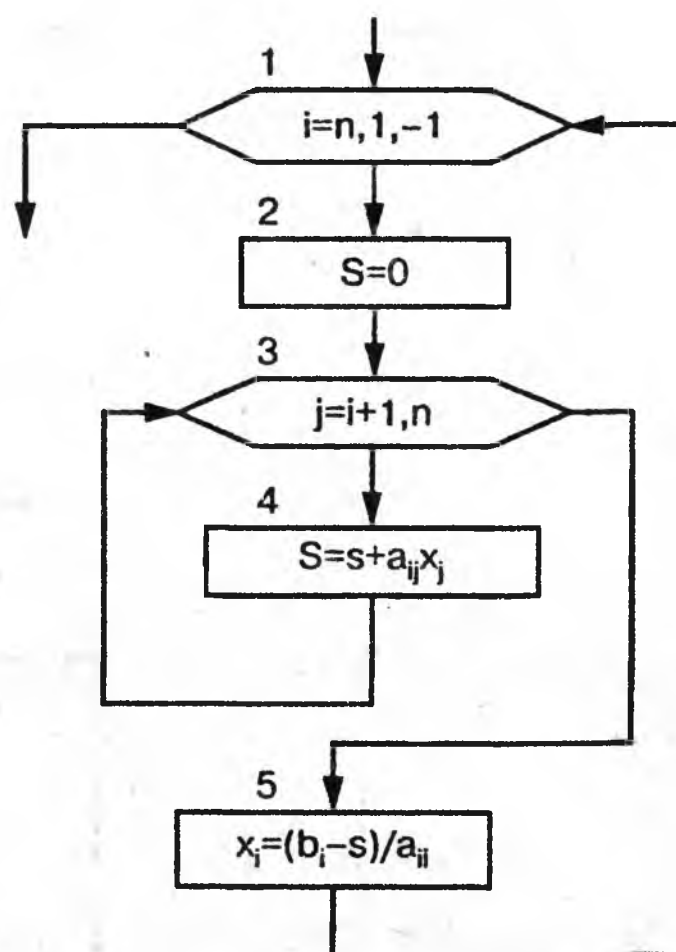


Рис. 1.3 ▼ Блок-схема алгоритма обратного хода метода Гаусса

данных, где n – это размерность системы линейных алгебраических уравнений, а сама система задается в виде матрицы коэффициентов при неизвестных A и вектора свободных коэффициентов b . Блоки 7–19 предусматривают прямой ход метода Гаусса, а блоки 20–29 – обратный. Для вывода результатов предусмотрено несколько блоков вывода. Если результат проверки условий 20 и 21 положительный, то выдается сообщение о том, что система имеет бесконечное множество решений (блок 22). Если условие 20 выполняется, а 21 – нет, то появляется сообщение о том, что система не имеет решений (блок 23). Сами же решения системы уравнений, представленные вектором x , вычисляются (блоки 24–28) и выводятся на печать (блок 29) только в случае невыполнения условия 20.

С программой, соответствующей этому алгоритму, можно ознакомиться в задаче 1.29.

1.2.2. Вычисление обратной матрицы методом Гаусса

Один из методов вычисления *обратной матрицы* основан на решении систем линейных алгебраических уравнений. Пусть задана некоторая матрица A :

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}. \quad (1.5)$$

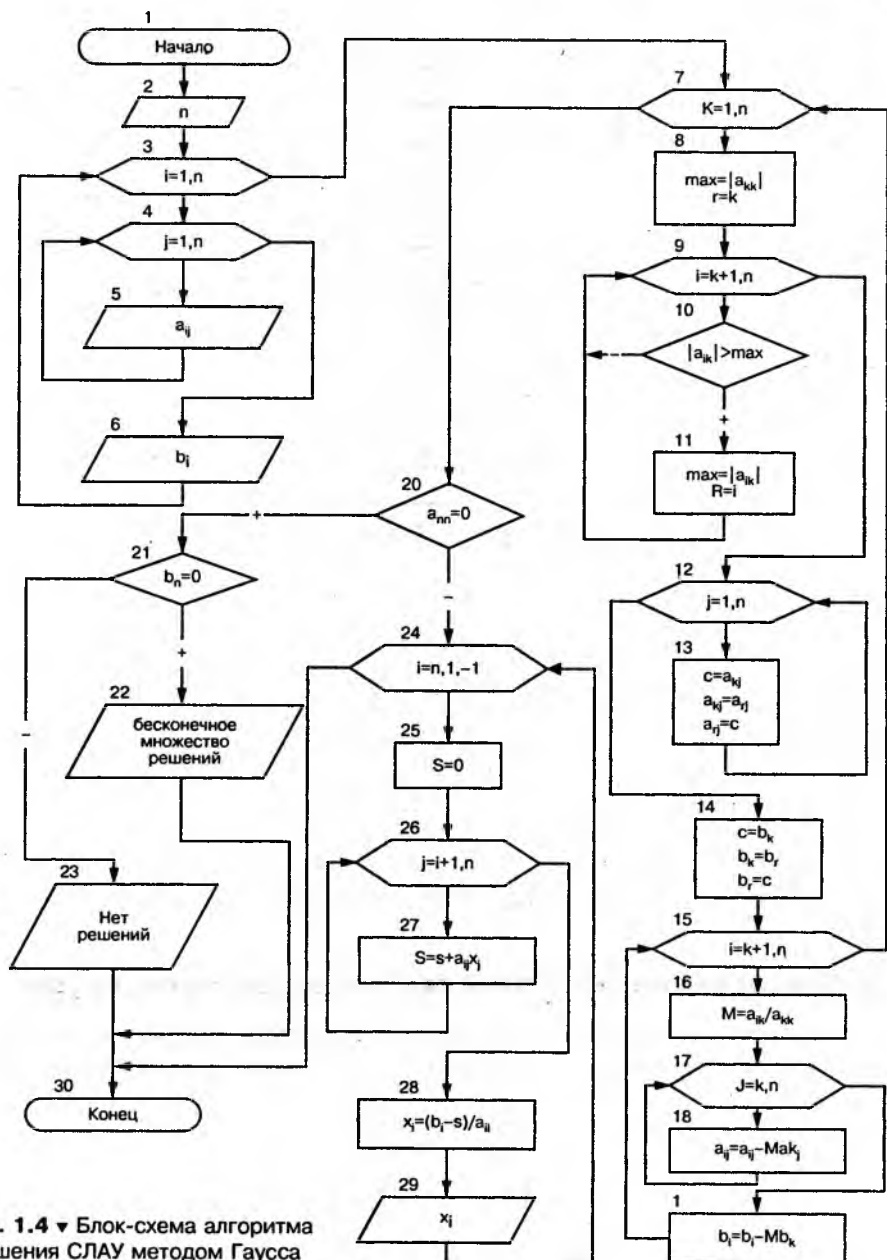


Рис. 1.4 ▼ Блок-схема алгоритма решения СЛАУ методом Гаусса

Необходимо найти матрицу A^{-1} , которая является обратной к матрице A :

$$A^{-1} = \begin{pmatrix} y_{11} & y_{12} & \dots & y_{1n} \\ y_{21} & y_{22} & \dots & y_{2n} \\ y_{31} & y_{32} & \dots & y_{3n} \\ \dots & \dots & \dots & \dots \\ y_{n1} & y_{n2} & \dots & y_{nn} \end{pmatrix}. \quad (1.6)$$

Матрица (1.6) будет обратной к матрице (1.5), если выполняется соотношение

$$A \cdot A^{-1} = E,$$

где E – это единичная матрица. Или более подробно:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ a_{31} & a_{32} & \dots & a_{3n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \times \begin{pmatrix} y_{11} & y_{12} & \dots & y_{1n} \\ y_{21} & y_{22} & \dots & y_{2n} \\ y_{31} & y_{32} & \dots & y_{3n} \\ \dots & \dots & \dots & \dots \\ y_{n1} & y_{n2} & \dots & y_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}. \quad (1.7)$$

Результат перемножения матриц из соотношения (1.7) можно представить поэлементно в виде n -го числа систем линейных уравнений. Умножение матрицы (1.5) на первый столбец матрицы (1.6) даст первый столбец единичной матрицы:

$$\begin{cases} a_{11} \cdot y_{11} + a_{12} \cdot y_{21} + \dots + a_{1n} \cdot y_{n1} = 1, \\ a_{21} \cdot y_{11} + a_{22} \cdot y_{21} + \dots + a_{2n} \cdot y_{n1} = 0, \\ \dots \\ a_{n1} \cdot y_{11} + a_{n2} \cdot y_{21} + \dots + a_{nn} \cdot y_{n1} = 0. \end{cases}$$

Система, полученная в результате умножения матрицы (1.5) на i -й столбец матрицы (1.6), будет выглядеть следующим образом:

$$\left. \begin{aligned} a_{11} \cdot y_{1i} + a_{12} \cdot y_{2i} + \dots + a_{1n} \cdot y_{ni} &= 0, \\ a_{21} \cdot y_{1i} + a_{22} \cdot y_{2i} + \dots + a_{2n} \cdot y_{ni} &= 0, \\ \dots \\ a_{i1} \cdot y_{1i} + a_{i2} \cdot y_{2i} + \dots + a_{in} \cdot y_{ni} &= 1, \\ \dots \\ a_{n1} \cdot y_{1i} + a_{n2} \cdot y_{2i} + \dots + a_{nn} \cdot y_{ni} &= 0. \end{aligned} \right\}$$

Понятно, что n -я система будет иметь вид:

$$\left. \begin{aligned} a_{11} \cdot y_{1n} + a_{12} \cdot y_{2n} + \dots + a_{1n} \cdot y_{nn} &= 0, \\ a_{21} \cdot y_{1n} + a_{22} \cdot y_{2n} + \dots + a_{2n} \cdot y_{nn} &= 0, \\ \dots &\dots \dots \\ a_{n1} \cdot y_{1n} + a_{n2} \cdot y_{2n} + \dots + a_{nn} \cdot y_{nn} &= 1. \end{aligned} \right\}$$

Решением каждой из приведенных выше систем будет i -й столбец обратной матрицы. Количество систем равно размерности обратной матрицы. Для отыскания решений систем линейных алгебраических уравнений можно воспользоваться методом Гаусса.

Описанный алгоритм представлен в виде блок-схемы на рис. 1.5. Блоки 2–5 отражают формирование столбца единичной матрицы. Если условие 3 выполняется и элемент находится на главной диагонали, то он равен единице, все остальные элементы нулевые. В блоке 6 происходит вызов подпрограммы для решения системы уравнений методом Гаусса. В качестве параметров в эту подпрограмму передается исходная матрица A , сформированный в пунктах 2–5

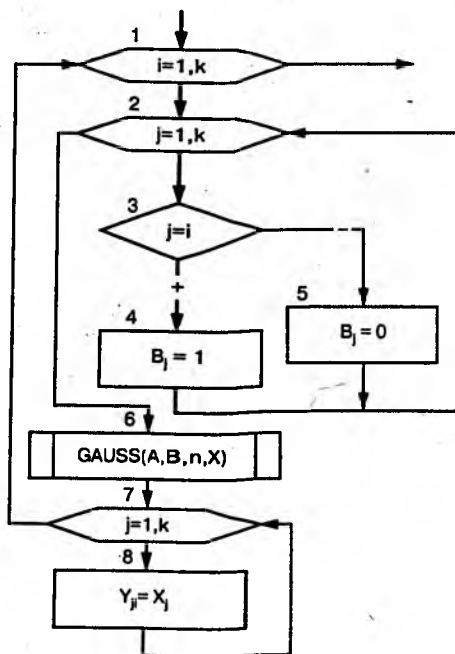


Рис. 1.5 ▼ Блок-схема алгоритма вычисления обратной матрицы

вектор свободных коэффициентов b , размерность системы n . Вектор x будет решением i -ой системы уравнений и, следовательно, i -ым столбцом искомой матрицы Y .

1.2.3. Вычисление определителя методом Гаусса

Пусть задана матрица (1.2), необходимо вычислить ее *определитель*. Для этого матрицу следует привести к треугольному виду (1.3), а затем воспользоваться свойством, известным из курса линейной алгебры, которое гласит, что определитель треугольной матрицы равен произведению ее диагональных элементов:

$$\det A = \prod_{i=1}^n a_{ii}.$$

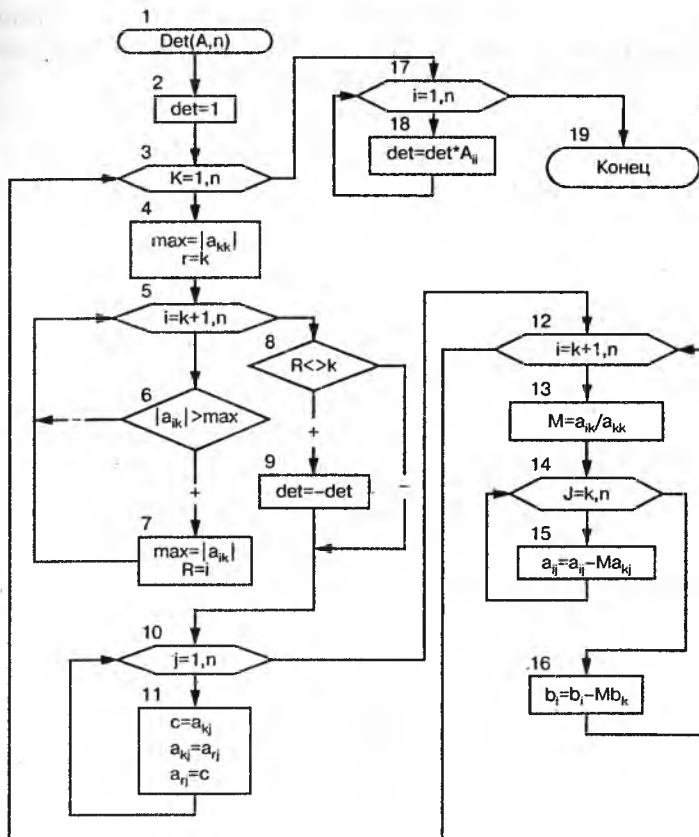


Рис. 1.6 ▼ Блок-схема алгоритма вычисления определителя

Приведение матрицы (1.2) к виду (1.3) можно осуществить с помощью прямого хода метода Гаусса. Алгоритм вычисления определителя матрицы, изображенный в виде блок-схемы на рис. 1.6, представляет собой алгоритм прямого хода метода Гаусса, в процессе выполнения которого проводится перестановка строк матрицы. Эта операция приводит к смене знака определителя. В блок-схеме момент смены знака отражен в блоках 8–9. В блоке 8 определяется, будут ли строки меняться местами, и если ответ утвердительный, то в блоке 9 происходит смена знака определителя. В блоках 17–18 выполняется непосредственное вычисление определителя путем перемножения диагональных элементов преобразованной матрицы.

1.2.4. Метод квадратных корней

Опишем еще один *точный способ* решения линейных систем, который называется *методом квадратных корней*. Пусть дана система линейных алгебраических уравнений (1.1). Запишем ее в матричной форме:

$$A \cdot x = b, \quad (1.8)$$

где $A = \{a_{ij}\}$ – симметрическая матрица, то есть $A^T = \{a_{ij}\} = A$. Тогда матрицу A можно представить в виде произведения двух транспонированных между собой треугольных матриц

$$A = T^T \times T, \quad (1.9)$$

где

$$T = \begin{pmatrix} t_{11} & t_{12} & \dots & t_{1n} \\ 0 & t_{22} & \dots & t_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & t_{nn} \end{pmatrix} \text{ и } T^T = \begin{pmatrix} t_{11} & 0 & \dots & 0 \\ t_{12} & t_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ t_{1n} & t_{2n} & \dots & t_{nn} \end{pmatrix}.$$

Перемножение матриц T и T^T дает следующие уравнения:

$$\begin{cases} t_{1i} \cdot t_{1j} + t_{2i} \cdot t_{2j} + \dots + t_{ii} \cdot t_{ij} = a_{ij}, & (i < j), \\ t_{ii}^2 + t_{2i}^2 + \dots + t_{ii}^2 = a_{ii}. \end{cases}$$

Отсюда последовательно находим:

$$\begin{cases} t_{11} = \sqrt{a_{11}}, \quad t_{1j} = \frac{a_{1j}}{t_{11}}, \quad (i > 1), \\ t_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} t_{ki}^2}, \quad (1 < i \leq n), \\ t_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} t_{ki} \cdot t_{kj}}{t_{ii}}, \quad (i < j), \\ t_{ij} = 0, \quad (i > j). \end{cases} \quad (1.10)$$

Система (1.8) имеет определенное единственное решение, если $t_{ii} \neq 0$ ($i = 1, 2, \dots, n$), так как тогда $\det A = \det T^T \cdot \det T = (\det T)^2 = (t_{11} t_{22} \dots t_{nn})^2 \neq 0$.

В случае если соотношение (1.9) выполняется, уравнение (1.8) эквивалентно двум уравнениям:

$$T^T \cdot y = b \text{ и } T \cdot x = y,$$

или

$$\begin{cases} t_{11} \cdot y_1 = b_1, \\ t_{12} \cdot y_1 + t_{22} \cdot y_2 = b_2, \\ \dots \\ t_{1n} \cdot y_1 + t_{2n} \cdot y_n + \dots + t_{nn} \cdot y_n = b_n \end{cases} \quad (1.11)$$

и

$$\begin{cases} t_{11} \cdot x_1 + t_{12} \cdot x_2 + \dots + t_{1n} \cdot x_n = y_1, \\ t_{22} \cdot x_2 + \dots + t_{2n} \cdot x_n = y_2, \\ \dots \\ t_{nn} \cdot x_n = y_n. \end{cases} \quad (1.12)$$

Из (1.11) и (1.12) последовательно находим:

$$\begin{cases} y_1 = \frac{b_1}{t_{11}}, \\ y_i = \frac{b_i - \sum_{k=1}^{i-1} t_{ki} \cdot y_k}{t_{ii}}, \quad (i > 1) \end{cases} \quad (1.13)$$

и

$$\begin{cases} x_n = \frac{y_n}{t_{nn}}, \\ x_i = \frac{y_i - \sum_{k=i+1}^n t_{ik} \cdot x_k}{t_{ii}}, \quad (i < n). \end{cases} \quad (1.14)$$

На практике метод квадратных корней применяют так. *Прямой ход* с помощью формул (1.10) и (1.13) последовательно вычисляются коэффициенты t_{ij} и y_i ($i = 1, 2, \dots, n$), а затем *обратным ходом* с помощью формулы (1.14) находятся неизвестные x_i ($i = n, n-1, \dots, 1$).

В общем случае метод квадратных корней применим и для вычисления комплексных корней системы (1.1), так как если для некоторой m -й строки выполняется условие $(t_{mm})^2 < 0$, то соответствующие элементы t_{mj} – мнимые. Блок-схема метода квадратных корней приведена на рис. 1.7.

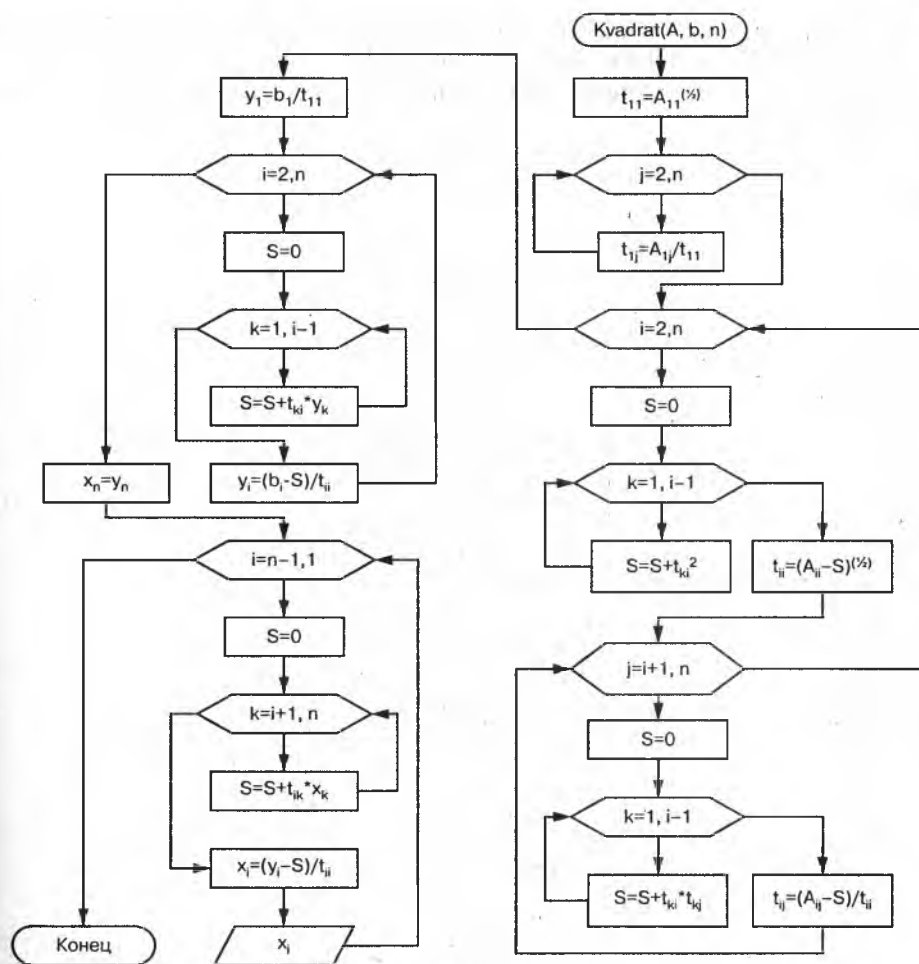


Рис. 1.7 ▼ Блок-схема метода квадратных корней

1.2.5. Схема Халецкого

Метод, называемый *схемой Халецкого*, также относится к *точным методам* решения систем линейных уравнений. Идея метода заключается в следующем [11]. Представим систему (1.1) в матричном виде

$$A \cdot x = b, \quad (1.15)$$

где $A = \{a_{ij}\}$ – квадратная матрица порядка n .

Представим матрицу A в виде $A = D \cdot C$, где $D = \{d_{ij}\}$ – нижняя треугольная матрица, а $C = \{c_{ij}\}$ – верхняя треугольная матрица с единичной диагональю:

$$D = \begin{pmatrix} d_{11} & 0 & \dots & 0 \\ d_{21} & d_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ d_{n1} & d_{n2} & \dots & d_{nn} \end{pmatrix}, \quad C = \begin{pmatrix} 1 & c_{12} & \dots & c_{1n} \\ 0 & 1 & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Тогда элементы d_{ij} и c_{ij} определяются по формулам:

$$\begin{cases} d_{11} = a_{11}, \\ d_{ij} = a_{ij} - \sum_{k=1}^{i-1} d_{ik} c_{kj}, \quad (i \geq j > 1) \end{cases} \quad (1.16)$$

и

$$\begin{cases} c_{1j} = \frac{a_{1j}}{d_{11}}, \\ c_{ij} = \frac{1}{d_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} d_{ik} c_{kj} \right), \quad (1 < i < j). \end{cases} \quad (1.17)$$

Следовательно, вектор x можно вычислить из системы

$$D \cdot y = b, \quad C \cdot x = y, \quad (1.18)$$

причем она легко решается, так как матрицы D и C – треугольные:

$$\begin{cases} y_1 = \frac{b_1}{d_{11}}, \\ y_i = \frac{1}{d_{ii}} \left(b_i - \sum_{k=1}^{i-1} d_{ik} y_k \right), \quad (i > 1) \end{cases} \quad (1.19)$$

и

$$\begin{cases} x_n = y_n, \\ x_i = y_i - \sum_{k=i+1}^n c_{ik} x_k, \quad (i < n). \end{cases} \quad (1.20)$$

Алгоритм схемы Халецкого довольно прост и изображен на рис. 1.8.

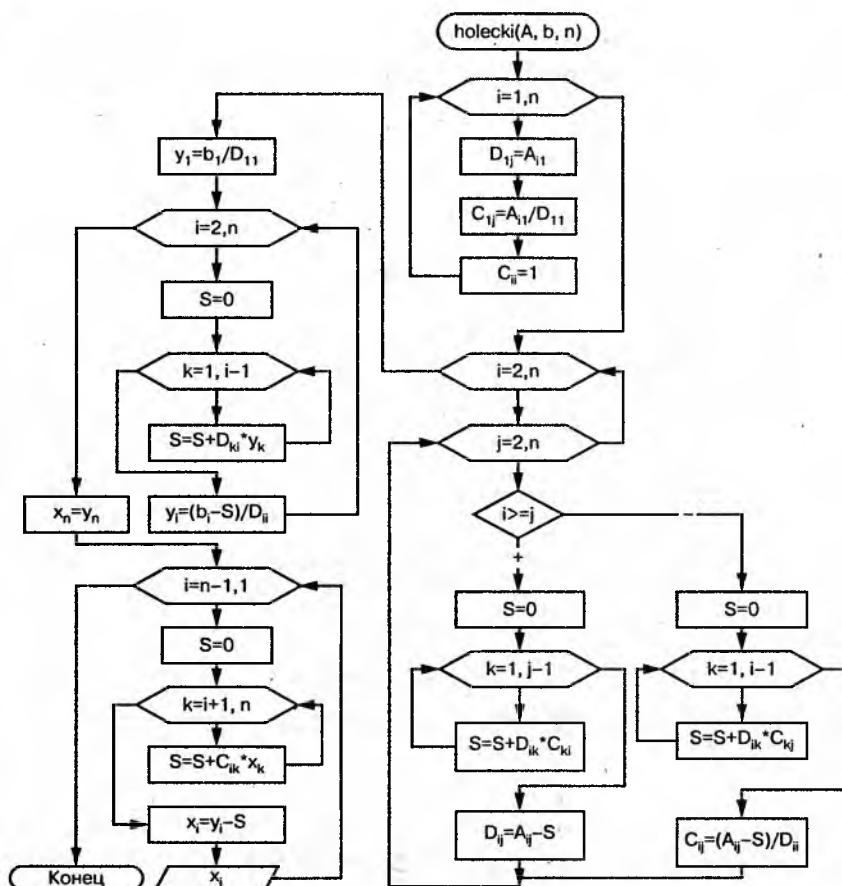


Рис. 1.8 ▼ Алгоритм схемы Халецкого

1.2.6. Методы LU- и QR-разложения

Изложим еще два *точных способа* решения линейных систем. Решение системы линейных алгебраических уравнений с помощью *LU-разложения* заключается в следующем. Пусть дана система

$$A \cdot x = b. \quad (1.21)$$

Представим матрицу $A = \{a_{ij}\}$ в виде произведения треугольных матриц $L = \{l_{ij}\}$ и $U = \{u_{ij}\}$:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ l_{n1} & l_{n2} & \dots & 1 \end{pmatrix} \times \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{nn} \end{pmatrix},$$

где элементы l_{ij} и u_{ij} определяются по формулам:

$$\begin{cases} u_{ij} = a_{ij}, (j = 1, 2, \dots, n), \\ u_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}, (i \leq j). \end{cases} \quad (1.22)$$

и

$$\begin{cases} l_{i1} = \frac{a_{i1}}{u_{11}}, (i = 2, \dots, n), \\ l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right), (i > j). \end{cases} \quad (1.23)$$

Формулы (1.22) и (1.23) дают нам LU-разложение матрицы A . Итак, если $A = L \cdot U$, то можно записать уравнение, эквивалентное (1.21):

$$L \cdot U \cdot x = b. \quad (1.24)$$

Введем вектор вспомогательных переменных $y = (y_1, y_2, \dots, y_n)^T$ и представим уравнение (1.24) в виде системы

$$\begin{cases} L \cdot y = b, \\ U \cdot x = y. \end{cases}$$

Таким образом, решение системы (1.1) с квадратной матрицей коэффициентов сводится к решению двух систем с треугольными матрицами коэффициентов:

$$\begin{cases} y_1 = b_1, \\ l_{21} \cdot y_1 + y_2 = b_2, \\ \dots \dots \dots \\ l_{n1} \cdot y_1 + l_{n2} \cdot y_2 + \dots + l_{n,n-1} \cdot y_{n-1} + y_n = b_n \end{cases} \quad (1.25)$$

и

$$\begin{cases} u_{11} \cdot x_1 + u_{12} \cdot x_2 + \dots + u_{1n} \cdot x_n = y_1, \\ u_{22} \cdot x_2 + \dots + u_{2n} \cdot x_n = y_2, \\ \dots \dots \dots \\ u_{nn} \cdot x_n = y_n. \end{cases} \quad (1.26)$$

Из (1.25) и (1.26) понятно, что все y_i могут быть вычислены по формуле:

$$y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k, \quad (i = 1, 2, \dots, n), \quad (1.27)$$

а для всех x_i истинно равенство

$$x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{k=i+1}^n u_{ik} x_k \right), \quad (i = n, n-1, \dots, 2, 1). \quad (1.28)$$

Итак, решить систему (1.1) методом LU-разложения¹ можно, выполнив следующие действия:

1. Определить элементы матриц L и U по формулам (1.22) и (1.23).

3. Получить решение заданной системы по формулам (1.27) и (1.28).

Алгоритм решения системы линейных алгебраических уравнений с помощью LU-разложения приведен на рис. 1.9.

Метод QR-разложения можно считать более сложной версией LU-разложения, так как в нем главную матрицу A линейной системы (1.21) также представляют

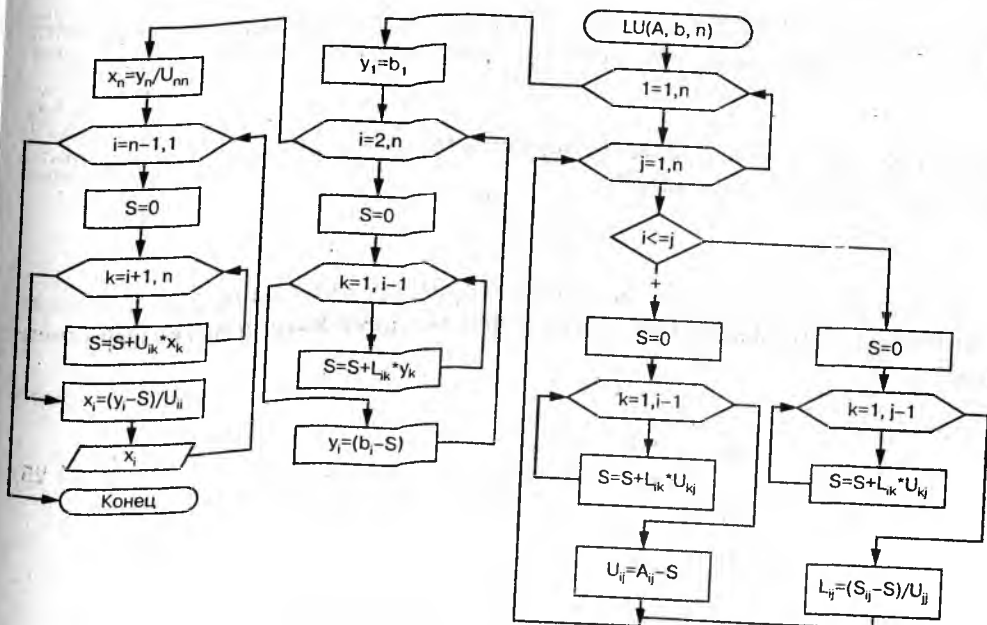


Рис. 1.9 ▽ Алгоритм метода LU-разложения

¹ Иногда применяют термин LU-факторизация или LU-декомпозиция.

в виде произведения двух матриц – ортогональной матрицы Q и верхней треугольной матрицы R . Далее поступают так же, как и в случае LU-разложения, то есть вводят дополнительные переменные y и записывают систему уравнений:

$$Q \cdot R \cdot x = b \Rightarrow (Q \cdot y = b \text{ и } R \cdot x = y).$$

Следовательно, решение системы уравнений (1.21) сводится к решению двух систем; матрица коэффициентов первой – ортогональная, второй – верхняя треугольная. А поскольку ортогональная матрица обладает свойством $Q^T = Q^{-1}$, то можно вычислить по формуле $y = Q^T \cdot b$, что значительно проще, чем решать уравнение $y = Q^{-1} \cdot b$. Вычислить x из второй системы также не составит труда, это можно сделать, например, по формулам (1.28), заменив элементы матрицы U на R (пример программирования см. в задаче 1.24).

Однако сложность этого метода заключается в вычислении ортогональной матрицы. В следующем параграфе подробно рассматривается эта проблема.

1.2.7. Метод ортогонализации

Рассмотрим еще один *точный метод* решения систем линейных алгебраических уравнений, который называют *методом ортогонализации*. Заключается он в том, что систему (1.1) ортонормируют, то есть приводят ее к виду:

$$Q \cdot x = d, \quad (1.29)$$

где Q – ортогональная матрица. Для этого разделим первое уравнение системы (1.1) на выражение

$$\sqrt{\sum_{k=1}^n a_{1k}^2}.$$

Получим

$$q_{1j} = \frac{a_{1j}}{\sqrt{\sum_{k=1}^n a_{1k}^2}}, \quad d_1 = \frac{b_1}{\sqrt{\sum_{k=1}^n a_{1k}^2}}, \quad \sum_{k=1}^n q_{1k}^2 = 1, \quad (j = 1, 2, \dots, n). \quad (1.30)$$

Тогда для второго уравнения системы (1.29) имеем:

$$\left. \begin{aligned} q_{2j} &= \frac{q'_{2j}}{\sqrt{\sum_{k=1}^n q'^2_{2k}}}, & d_2 &= \frac{d'_2}{\sqrt{\sum_{k=1}^n q'^2_{2k}}}, \quad (j = 1, 2, \dots, n), \\ q'_{2j} &= a_{2j} - q_{1j} \cdot \sum_{k=1}^n a_{2k} \cdot q_{1k}, & d'_2 &= b_2 - d_1 \cdot \sum_{k=1}^n a_{2k} \cdot q_{1k}. \end{aligned} \right\}$$

Следовательно, уравнение с номером i примет вид:

$$\left. \begin{aligned} q_{ij} &= \frac{q'_{ij}}{\sqrt{\sum_{k=1}^n q_{ik}^2}}, \quad d_i = \frac{d'_i}{\sqrt{\sum_{k=1}^n q_{ik}^2}}, \quad (i = 2, 3, \dots, n; j = 1, 2, \dots, n), \\ q'_{ij} &= a_{ij} - q_{1j} \cdot \sum_{k=1}^n a_{1k} \cdot q_{ik} - q_{2j} \cdot \sum_{k=1}^n a_{2k} \cdot q_{ik} - \dots - q_{i-1,j} \cdot \sum_{k=1}^n a_{i-1,k} \cdot q_{ik}, \\ d'_2 &= b_2 - d_1 \cdot \sum_{k=1}^n a_{1k} \cdot q_{2k} - d_2 \cdot \sum_{k=1}^n a_{2k} \cdot q_{2k} - \dots - d_{i-1} \cdot \sum_{k=1}^n a_{i-1,k} \cdot q_{2k}. \end{aligned} \right\} \quad (1.31)$$

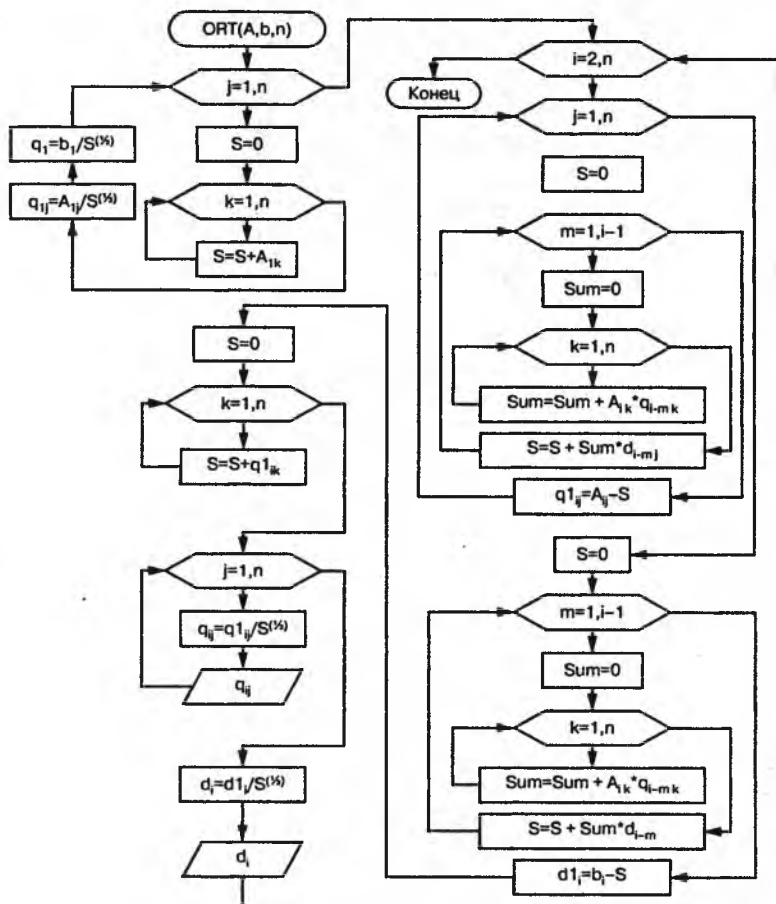


Рис. 1.10 ▼ Алгоритм метода ортогонализации

Если система (1.1) совместна, то, воспользовавшись формулами (1.30) и (1.31), можно прийти к новой системе (1.29), где матрица Q будет ортогональной, а это значит, что она обладает свойством

$$Q^T \cdot Q = E,$$

где E – единичная матрица. Значит, решение системы (1.29) можно записать в виде

$$Q^T \cdot Q \cdot x = Q^T \cdot d,$$

или

$$x = Q^T \cdot d. \quad (1.32)$$

Рис. 1.10 содержит алгоритм (пример программирования см. в задаче 1.23) решения линейной системы методом ортогонализации по формулам (1.30), (1.31) и (1.32). Заметим, что этим алгоритмом можно воспользоваться, если необходимо решить задачу вычисления ортогональной матрицы Q для матрицы A . Понятно, что в этом случае расчет вектора d теряет смысл.

1.2.8. Метод прогонки

На практике часто встречаются линейные системы вида

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

которые называют *системами с трехдиагональной матрицей коэффициентов*. Каждое уравнение такой системы связывает три неизвестных, расположенных особым образом, и может быть записано в общем виде так:

$$b_i \cdot x_{i-1} + c_i \cdot x_i + d_i \cdot x_{i+1} = r_i, \quad (1.33)$$

где $i = 1, 2, \dots, n$, $b_1 = 0$, $d_n = 0$. Такие уравнения называют *трехточечными разностными уравнениями второго порядка*. Для решения систем (1.33) существует специальный метод, называемый *методом прогонки*. Заключается он в следующем [9]. Предположим, что существуют такие наборы чисел δ_i и λ_i ($i = 1, 2, \dots, n$), для которых

$$x_i = \delta_i \cdot x_{i+1} + \lambda_i. \quad (1.34)$$

Уменьшим индекс в уравнении (1.34) и подставим его в (1.33), получим

$$b_i \cdot \delta_{i-1} \cdot x_i + b_i \cdot \lambda_{i-1} + c_i \cdot x_i + d_i \cdot x_{i+1} = r_i,$$

откуда имеем

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

Следовательно, уравнение (1.34) имеет место, если для всех $i = 1, 2, \dots, n$ выполняются рекуррентные соотношения:

$$\delta_i = -\frac{d_i}{c_i + b_i \cdot \delta_{i-1}}, \quad \lambda_i = \frac{r_i - b_i \cdot \lambda_{i-1}}{c_i + b_i \cdot \delta_{i-1}}. \quad (1.35)$$

Нетрудно заметить что, так как $b_1 = 0$, то процесс вычисления δ_i и λ_i по формулам (1.35) можно начать со значений

$$\delta_1 = -\frac{d_1}{c_1}, \quad \lambda_1 = \frac{r_1}{c_1}$$

и продолжать для $i = 1, 2, \dots, n$. В случае, когда $i = n$ и $d_n = 0$, получим $d_n = 0$, то есть для $i = n$ будем иметь

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

где δ_{i-1} и λ_{i-1} были вычислены на предыдущем шаге. Далее по формулам (1.35) последовательно находятся значения $x_{n-1}, x_{n-2}, \dots, x_1$ при $i = n-1, n-2, \dots, 1$.

Итак, алгоритм метода прогонки (рис. 1.11) сводится к вычислениям по трем формулам:

1. Находят прогоночные коэффициенты δ_i и λ_i по формулам (1.35). Этот процесс называют *прямой прогонкой*.

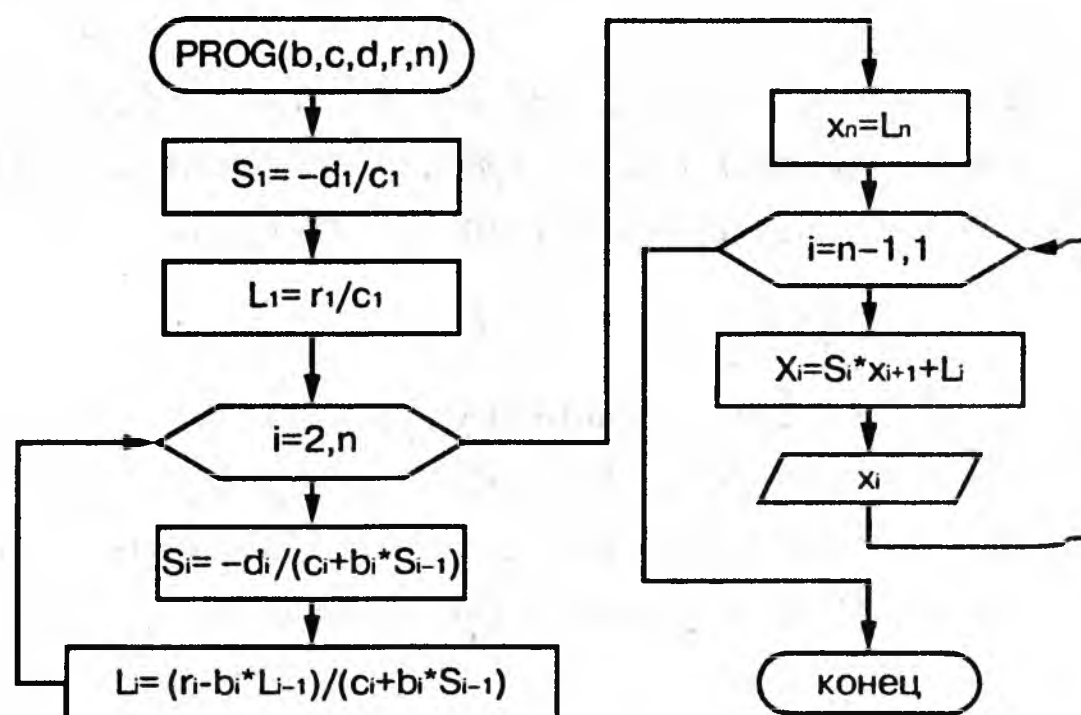


Рис. 1.11 ▼ Алгоритм метода прогонки

2. Определяют неизвестные по формуле (1.34), выполняя тем самым *обратную прогонку*.

Пример программирования приведен в задаче 1.25.

1.2.9. Метод итерации

Перейдем к рассмотрению *приближенных методов* решения систем линейных алгебраических уравнений. Один из них – это *метод итерации*. Изложим его суть.

Пусть дана линейная система

[illegible]

Разрешим первое уравнение системы (1.36) относительно x_1 , второе – относительно x_2 и так далее. Получим систему, эквивалентную данной:

[illegible]

где

$$\begin{aligned} \beta_i &= \frac{b_i}{a_{ii}}, \quad \alpha_{ij} = -\frac{a_{ij}}{a_{ii}}, \quad i \neq j; \\ \alpha_{ij} &= 0, \quad i = j; \\ i &= 1, 2, \dots, n, \quad j = 1, 2, \dots, n. \end{aligned} \quad (1.38)$$

Запишем систему (1.37) в матричном виде

$$x = \beta + \alpha \cdot x, \quad (1.39)$$

где

$$\alpha = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \dots & \dots & \dots & \dots \\ \alpha_{n1} & \alpha_{n2} & \dots & \alpha_{nn} \end{pmatrix}, \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_n \end{pmatrix}.$$

Для решения системы (1.39) методом итерации необходимо выбрать нулевое приближение. Пусть это будет столбец свободных коэффициентов:

$$x^0 = \beta, \quad (1.40)$$

тогда первое приближение будет иметь вид:

$$x^1 = \beta + \alpha \cdot x^0,$$

а второе будет таким:

$$x^2 = \beta + \alpha \cdot x^1,$$

и так далее: $(k + 1)$ -е приближение будем вычислять по формуле:

$$x^{k+1} = \beta + \alpha \cdot x^k, \quad (k = 1, 2, \dots)$$

или в развернутом виде:

$$x_i^k = \beta_i + \sum_{j=1}^n \alpha_{ij} \cdot x_j^k, \quad (i = 1, 2, \dots, n; \quad k = 1, 2, \dots). \quad (1.41)$$

Вообще говоря, если последовательность приближений $x^0, x^1, x^2, \dots, x^k, \dots$ имеет предел

$$x = \lim_{k \rightarrow \infty} x^k,$$

то этот предел и является решением системы (1.39).

При практическом применении метода итерации вычисления прекращают, когда выполняется условие

$$\max(|x - x^k|) \leq \epsilon, \quad (1.42)$$

где ϵ – заданная точность вычислений.

Приведем без доказательства *условие сходимости* метода итерации [11]: для существования единственного решения системы (1.36) и сходимости метода простых итераций достаточно выполнения условия

$$\sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| < |a_{ii}|, \quad i = 1, 2, \dots, n, \quad (1.43)$$

то есть чтобы модули диагональных коэффициентов для каждого уравнения системы были больше суммы модулей всех остальных коэффициентов, не считая свободных.

Итак, алгоритм метода простой итерации (рис. 1.12) заключается в следующем:

1. Проверка условий (1.43), если они не выполняются, то работа алгоритма завершена, иначе переходим к п.2.
2. Формирование матрицы α и массива β по формулам (1.38).
3. Формирование начального приближения x^0 по (1.40).
4. Расчет нового приближения x^1 по (1.41).
5. Если условие (1.42) выполняется и максимальная ошибка вычислений меньше заданного числа ϵ , то решение x^1 найдено, иначе переходим к п.4.



Метод Зейделя – это *приближенный метод* решения систем уравнений, представляющий собой модификацию метода простой итерации. Основная идея [11] этого метода заключается в том, что при вычислении $(k + 1)$ -го приближения неизвестной x_i учитываются уже вычисленные к этому времени $(k + 1)$ -е предшествующих неизвестных $x_1, x_2, \dots, x_{i-1}$.

Пусть дана приведенная линейная система

$$x_i = \beta_i + \sum_{j=1}^n \alpha_{ij} \cdot x_j \quad (i = 1, 2, \dots, n). \quad (1.44)$$

Выберем в качестве начального приближения корней вектор

$$x_1^0, x_2^0, \dots, x_n^0 \quad (1.45)$$

Тогда, согласно идее метода Зейделя, предполагая, что k -е приближения корней известны, будем строить $(k+1)$ -е приближения по следующим формулам:

$$\left\{ \begin{array}{l} x_1^{k+1} = \beta_1 + \sum_{j=1}^n \alpha_{1j} \cdot x_j^k; \\ x_2^{k+1} = \beta_2 + \alpha_{21} \cdot x_1^{k+1} + \sum_{j=2}^n \alpha_{2j} \cdot x_j^k; \\ \dots\dots\dots \\ x_i^{k+1} = \beta_i + \sum_{j=1}^{i-1} \alpha_{ij} \cdot x_j^{k+1} + \sum_{j=i}^n \alpha_{ij} \cdot x_j^k; \\ \dots\dots\dots \\ x_n^{k+1} = \beta_n + \sum_{j=1}^{n-1} \alpha_{nj} \cdot x_j^{k+1} + \sum_{j=n}^n \alpha_{nn} \cdot x_n^k, \quad (k = 0, 1, 2, \dots). \end{array} \right. \quad (1.46)$$

Возникает вопрос о сходимости метода. Заметим, что условия (1.43) применимы и для метода Зейделя, кроме того, если линейная система (1.36) – нормальная, то процесс Зейделя для эквивалентной ей приведенной системы сходится всегда. Напомним, что линейная система (1.36) называется *нормальной*, если матрица коэффициентов $A = \{a_{ij}\}$ – симметрическая, то есть $a_{ij} = a_{ji}$ и соответствующая квадратичная форма

$$u = \sum_{i=1}^n \sum_{j=1}^n a_{ij} \cdot x_i \cdot x_j$$

положительно определена.

Возникает проблема, как от системы (1.36) перейти к системе (1.39) таким образом, чтобы выполнялись все условия сходимости. Выполним следующие преобразования:

$$A^T \cdot A \cdot x = A^T \cdot b.$$

Пусть

$$F = A^T \cdot A \quad \text{и} \quad H = A^T \cdot b, \quad (1.47)$$

тогда система

$$F \cdot x = H \quad (1.48)$$

эквивалентна системе $A \cdot x = b$ и обладает следующими свойствами:

- ▶ матрица F симметрична, то есть $f_{ij} = f_{ji}$;
- ▶ все элементы главной диагонали матрицы F положительны: $f_{ii} > 0$;
- ▶ соответствующая квадратичная форма является положительно определенной.

Теперь, если систему (1.48) привести к специальному виду:

$$x_i = \sum_{j=1}^n \alpha_{ij} \cdot x_j + \beta_i, \quad (1.49)$$

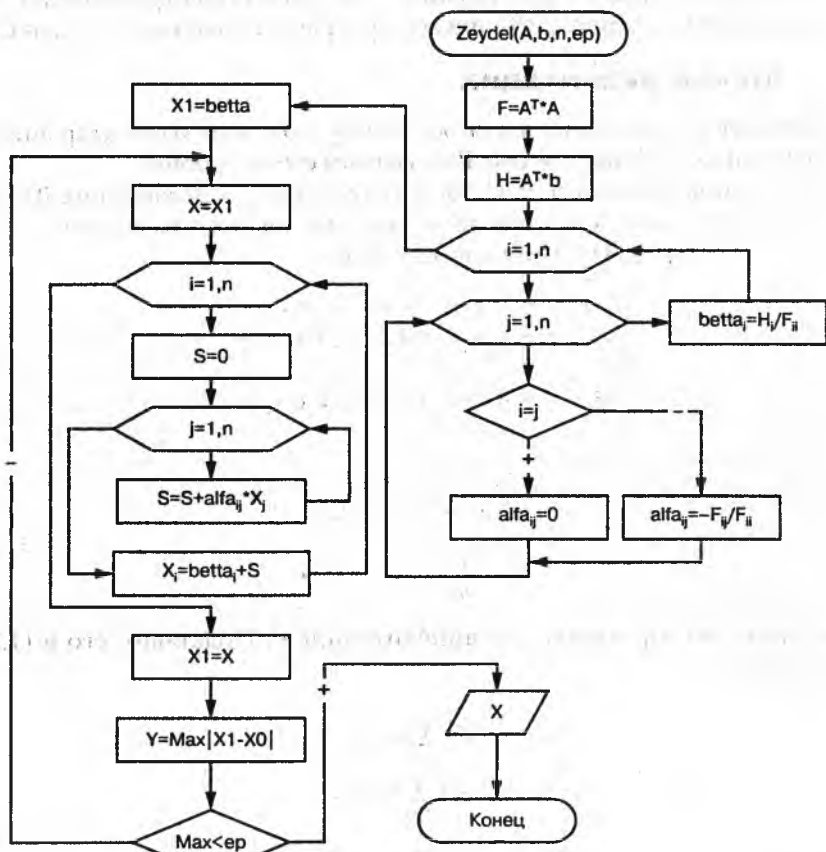


Рис. 1.13 ▼ Алгоритм метода Зейделя

где

$$\begin{aligned}\beta_i &= \frac{b_i}{f_{ii}}, \\ \alpha_{ij} &= -\frac{f_{ij}}{f_{ii}}, \quad (i \neq j), \\ \alpha_{ij} &= 0, \quad (i = j),\end{aligned}\tag{1.50}$$

то она будет сходиться к единственному решению при любом начальном приближении.

Алгоритм решения системы линейных алгебраических уравнений по методу Зейделя приведен на рис. 1.13 (пример программирования см. в задаче 1.26).

1.2.11. Метод релаксации

Метод релаксации, или, иначе, *метод ослабления*, – это еще один итерационный способ решения линейных систем. Рассмотрим его подробнее.

Пусть дана линейная система (1.36). Подготовим ее к релаксации. Для этого перенесем свободные члены налево и разделим первое уравнение на $(-a_{11})$, второе – на $(-a_{22})$ и т.д. Получим систему вида:

$$\begin{cases} -x_1 + d_{12} \cdot x_2 + \dots + d_{1n} \cdot x_n + c_1 = 0, \\ d_{21} \cdot x_1 - x_2 + \dots + d_{2n} \cdot x_n + c_2 = 0, \\ \dots \dots \dots \\ d_{n1} \cdot x_1 + d_{n2} \cdot x_2 + \dots - x_n + c_n = 0, \end{cases}\tag{1.51}$$

где

$$\begin{aligned}d_{ij} &= -\frac{a_{ij}}{a_{ii}}, \quad (i \neq j), \\ c_i &= \frac{b_i}{a_{ii}}.\end{aligned}\tag{1.52}$$

Пусть задан вектор начальных приближений x^0 . Подставим его в (1.51) и получим невязки:

$$\begin{cases} R_1^0 = c_1 - x_1^0 + \sum_{j=2}^n b_{1j} \cdot x_j^0, \\ R_2^0 = c_2 - x_2^0 + \sum_{j=1}^{n-1} b_{2j} \cdot x_j^0, \\ \dots \dots \dots \\ R_n^0 = c_n - x_n^0 + \sum_{j=1}^{n-1} b_{nj} \cdot x_j^0, \end{cases}\tag{1.53}$$

Если одной из неизвестных x_k^0 дать приращение Δx_k^0 , то соответствующая невязка R_k^0 уменьшится на величину Δx_k^0 , а все остальные невязки R_i^0 ($i \neq k$) увеличатся на величину $b_{ik}\Delta x_k^0$. Таким образом, чтобы обратить очередную невязку R_k^1 в нуль, достаточно в величине x_k^0 дать приращение

$$\Delta x_k^0 = R_k^0,$$

и мы будем иметь:

$$R_k^1 = 0$$

и

$$R_i^1 = R_i^0 + b_{ik}\Delta x_k^0 \quad (i \neq k). \quad (1.54)$$

Вообще говоря [11], идея метода релаксации заключается в том, что на каждом шаге обращается в нуль максимальная по модулю невязка путем изменения значения соответствующей компоненты приближения. Процесс заканчивается, когда все невязки последней преобразованной системы будут равны нулю с заданной точностью.

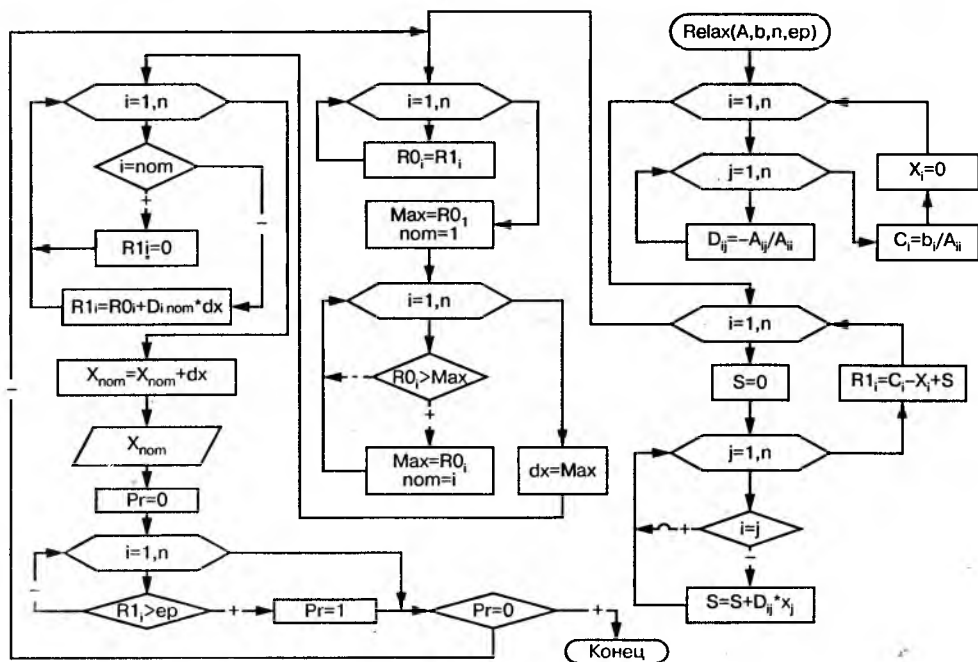


Рис. 1.14 ▼ Алгоритм метода релаксации

$$W = \frac{df}{dx} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = A.$$

Следовательно,

$$x^{p+1} = x^p - \mu_p \cdot A^T \cdot r_p, \quad (1.56)$$

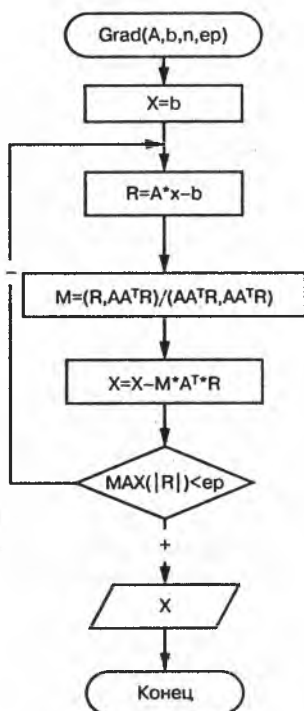
где

$$r_p = A \cdot x^p - b \quad (1.57)$$

– невязка вектора x^p и

$$\mu = \frac{(r_p, AA^T r_p)}{(AA^T r_p, AA^T r_p)}, \quad (p = 0, 1, 2, \dots). \quad (1.58)$$

Запишем алгоритм градиентного метода для решения линейных систем.



1. Выбираем начальное приближение, например $x = b$.
2. Вычисляем невязку r по формуле (1.57).
3. Находим расчетный коэффициент μ^1 по (1.58).
4. Определяем x по формуле (1.56).
5. Если максимальная невязка системы не превышает точность вычислений ε , то решение x найдено, иначе переходим к п.2.

Блок-схема алгоритма приведена на рис. 1.15.

Рис. 1.15 ▼ Алгоритм метода градиента

¹ Здесь под выражениями в скобках понимаются скалярные произведения соответствующих векторов.

1.3. Собственные значения и собственные векторы

Одна из важных задач вычислительной линейной алгебры – это нахождение собственных чисел и собственных векторов матрицы. Прежде чем перейти к изучению методов ее решения, сделаем некоторые вводные замечания.

Пусть A – матрица размерностью $n \times n$. Любой ненулевой вектор x , принадлежащий некоторому векторному пространству, для которого $A \cdot x = \lambda \cdot x$, где λ – некоторое число, называется *собственным вектором* матрицы A , а λ – принадлежащим ему или соответствующим ему *собственным значением* матрицы A .

Уравнение $A \cdot x = \lambda \cdot x$ эквивалентно уравнению $(A - \lambda \cdot E) \cdot x = 0$. Это однородная система линейных уравнений, нетривиальные решения которой являются искомыми собственными векторами. Она имеет нетривиальные решения только тогда, когда $r(A - \lambda \cdot E) < n$, то есть если $\det(A - \lambda \cdot E) = 0$.

Многочлен $\det(A - \lambda \cdot E)$ называется *характеристическим многочленом* матрицы A , а уравнение $\det(A - \lambda \cdot E) = 0$ – *характеристическим уравнением* матрицы A . Если λ_i – собственные значения A , то нетривиальные решения однородной системы линейных уравнений $(A - \lambda \cdot E) \cdot x = 0$ есть собственные векторы A , принадлежащие собственному значению λ_i . Множество решений этой системы уравнений называют *собственным подпространством* матрицы A , принадлежащим собственному значению λ_i , каждый ненулевой вектор собственного подпространства является собственным вектором матрицы A .

Иногда требуется найти собственные векторы y и собственные значения μ , определяемые соотношением $A \cdot y = \mu B y$ ($y \neq 0$), где B – невырожденная матрица. Векторы y и числа μ обязательно являются собственными векторами и собственными значениями матрицы $B^{-1}A$. Пусть $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$, причем матрица B является положительно определенной, тогда собственные значения μ совпадают с корнями уравнения n -й степени $\det(A - \mu B) = \det(a_{ij} - \mu b_{ij}) = 0$. Это уравнение называют *характеристическим уравнением для обобщенной задачи о собственных значениях*. Для каждого корня μ кратности m существует ровно m линейно независимых собственных векторов y .

Кроме того, большое значение при решении задач на собственные значения и собственные векторы имеет понятие *нормы*. Дадим несколько определений

Матричная норма – это некоторая скалярная числовая характеристика, которую ставят в соответствие матрице. В задачах линейной алгебры используют различные матричные нормы:

► *первая норма* $\|A\|_1$ квадратной матрицы $A = \{a_{ij}\}$:

$$\|A\|_1 = \max_j \sum_{i=1}^n |a_{ij}|;$$

- вторая норма $\|A\|_2$ квадратной матрицы $A = \{a_{ij}\}$:

$$\|A\|_2 = \sqrt{\lambda_{\max}(AA^T)},$$

где $\lambda_{\max}(AA^T)$ – максимальное собственное значение матрицы A ;

- евклидова норма $\|A\|_e$ квадратной матрицы $A = \{a_{ij}\}$:

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

- бесконечная норма $\|A\|_i$ квадратной матрицы A :

$$\|A\|_i = \max_i \sum_{j=1}^n |a_{ij}|.$$

Введем также понятие числа обусловленности матрицы, так как оно напрямую связано с нормой. Число обусловленности матрицы A используется для определения меры чувствительности системы линейных уравнений $A \cdot x = b$ к погрешностям задания вектора b . Чем больше число обусловленности, тем более неустойчив процесс нахождения решения системы. Существует несколько вариантов вычисления числа обусловленности, но все они связаны с нормой матрицы и равны произведению нормы исходной матрицы на норму обратной:

- число обусловленности матрицы, вычисленное в норме $\|A\|_1$:

$$N = \|A\|_1 \cdot \|A^{-1}\|_1;$$

- число обусловленности матрицы, вычисленное в норме $\|A\|_2$:

$$M = \|A\|_2 \cdot \|A^{-1}\|_2;$$

- число обусловленности матрицы, вычисленное в норме $\|A\|_i$:

$$P = \|A\|_i \cdot \|A^{-1}\|_i;$$

- число обусловленности матрицы, вычисленное в норме $\|A\|_e$:

$$H = \|A\|_e \cdot \|A^{-1}\|_e.$$

Теперь, когда нам известны все основные определения, перейдем к решению проблем собственных значений и соответствующих им векторов.

1.3.1. Метод скалярных произведений

Одним из распространенных методов решения проблемы *собственных значений* является *метод скалярных произведений*. Рассмотрим его применение на *частичной проблеме* собственных значений, которая заключается в нахождении наибольшего собственного значения матрицы.

Пусть дана квадратная действительная матрица A размерности n и известно, что она имеет ровно n линейно-независимых собственных векторов:

$$x_1 = \begin{pmatrix} x_{11} \\ x_{12} \\ \dots \\ x_{1n} \end{pmatrix}, x_2 = \begin{pmatrix} x_{21} \\ x_{22} \\ \dots \\ x_{2n} \end{pmatrix}, \dots, x_n = \begin{pmatrix} x_{n1} \\ x_{n2} \\ \dots \\ x_{nn} \end{pmatrix},$$

ЕМ

$$|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|.$$

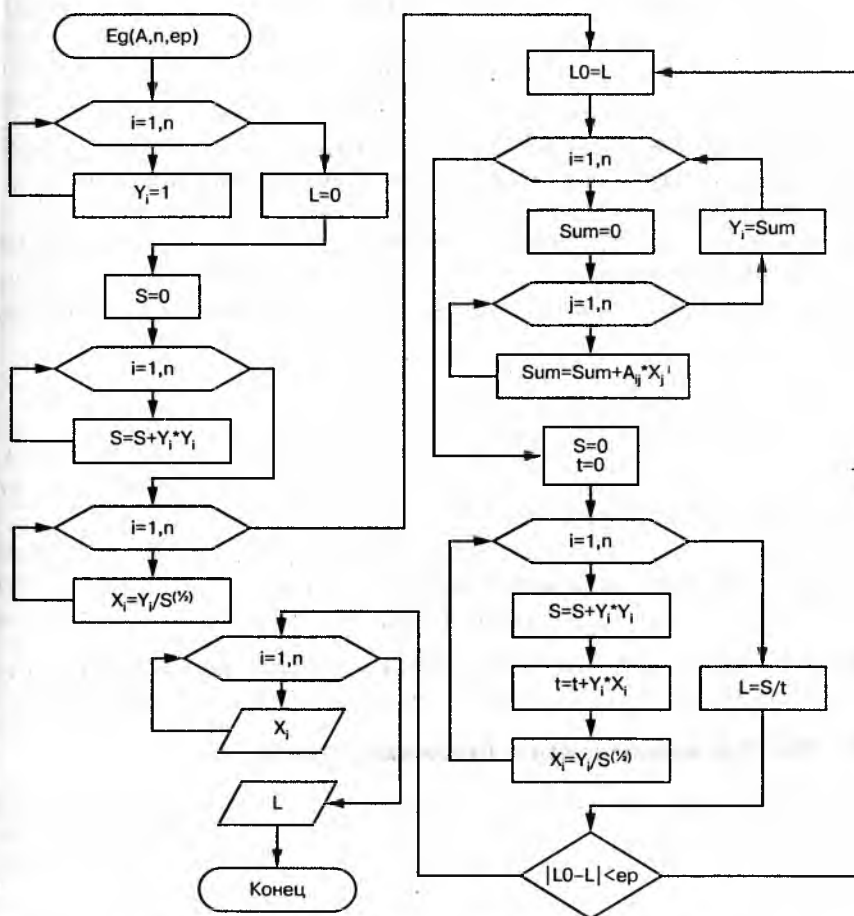


Рис. 1.16 ▽ Алгоритм метода скалярных произведений

$$(y^k, y^k) \text{ и } (y^k, x^{k-1}),$$

Приведем без доказательства алгоритм решения этой задачи [9].

1. Задать начальный вектор $y^0 \neq 0$ и начальное значение λ^0 , например $\lambda^0 = 0$.
2. Вычислить скалярное произведение $s^0 = (y^0, y^0)$, норму $\|y^0\|_2 = (s^0)^{1/2}$ и начальное значение собственного вектора $x^0 = y^0 / \|y^0\|_2$.
3. Уточнить начальное приближение y^0 по формуле $y^k = A \cdot x^{k-1}$.
4. Найти скалярное произведение $s^k = (y^k, y^k)$ и $t^k = (y^k, x^{k-1})$, затем норму $\|y^k\|_2 = (s^k)^{1/2}$ и приближение к собственному вектору $x^k = y^k / \|y^k\|_2$.
5. Уточнить собственное значение по формуле $\lambda^k = s^k / t^k$.
6. Если $|\lambda^k - \lambda^{k-1}|$ меньше допустимой абсолютной погрешности ε , то работа алгоритма завершена и $\lambda_i = \lambda^k$ и $x_i = x^k$, иначе вернуться к п.3.

Блок-схема алгоритма приведена на рис. 1.16 (пример программирования см. в задаче 1.27).

1.3.2. Метод итерации

В разделе 1.2.9 был описан *метод итерации* для решения линейных систем. Рассмотрим, как он применяется для одновременного вычисления собственных значений и собственных векторов

Пусть матрица $A = \{a_{ij}\}$ – симметрическая и положительно определенная, тогда корни ее характеристического уравнения $\lambda_1, \lambda_2, \dots, \lambda_n$ действительны и положительны, а собственные векторы удовлетворяют условию ортогональности

$$\sum_{i=1}^n x_i^j \cdot x_i^k = 0, j \neq k.$$

Значит, для определения собственного вектора x^1 можно записать систему

$$\begin{aligned} & (a_{11} - \lambda_1) \cdot x_1^1 + a_{12} \cdot x_2^1 + \dots + a_{1n} \cdot x_n^1 = 0, \\ & a_{21} \cdot x_1^1 + (a_{22} - \lambda_1) \cdot x_2^1 + \dots + a_{2n} \cdot x_n^1 = 0, \\ & \dots \\ & a_{n1} \cdot x_1^1 + a_{n2} \cdot x_2^1 + \dots + (a_{nn} - \lambda_1) \cdot x_n^1 = 0 \end{aligned}$$

ИЛИ

$$\begin{cases} x_1^1 = \frac{1}{\lambda_1} (a_{11} \cdot x_1^1 + a_{12} \cdot x_2^1 + \dots + a_{1n} \cdot x_n^1), \\ x_2^1 = \frac{1}{\lambda_1} (a_{21} \cdot x_1^1 + a_{22} \cdot x_2^1 + \dots + a_{2n} \cdot x_n^1), \\ \dots \\ x_{n-1}^1 = \frac{1}{\lambda_1} (a_{n-1,1} \cdot x_1^1 + a_{n-1,2} \cdot x_2^1 + \dots + a_{n-1,n} \cdot x_n^1), \\ \lambda_1 = \frac{1}{x_n^1} (a_{n1} \cdot x_1^1 + a_{n2} \cdot x_2^1 + \dots + a_{nn} \cdot x_n^1). \end{cases}$$

Эту систему можно решить методом итерации, положив $x_n^1 = 1$ и выбирая начальные значения $x_i^{(1)(0)}$, λ_i^0 и полагая

$$x_i^{(l)(k+1)} = \frac{1}{\lambda_i^k} \cdot \left(\sum_{j=1}^{n-1} a_{ij} \cdot x_j^{(l)(k+1)} + a_{in} \right), (i = 1, 2, \dots, n-1),$$

$$\lambda_i^{(k+1)} = \sum_{j=1}^{n-1} a_{nj} \cdot x_j^{(l)(k+1)} + a_{nn}, (k = 0, 1, 2, \dots).$$

Для определения второго корня λ_2 и второго собственного вектора x^2 определим $x_{n-1}^2 = 1$ и решим методом итерации следующую систему:

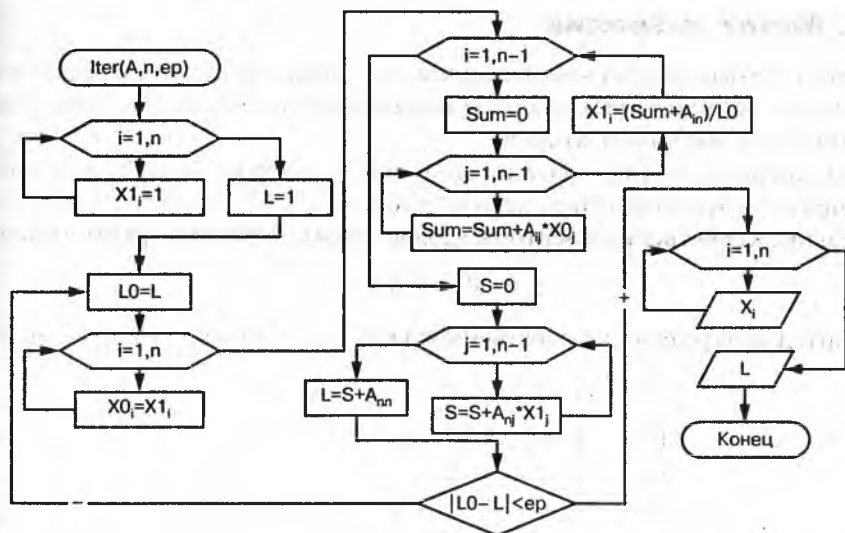


Рис. 1.17 ▼ Алгоритм метода итерации

$$x_i^{(2)} = \frac{1}{\lambda_2} \cdot \left(\sum_{j=1}^{n-1} a_{ij}^2 \cdot x_j^{(2)} \right), (i = 1, 2, \dots, n-2),$$

$$\lambda_2 = \frac{1}{x_{n-1}^2} \sum_{j=1}^{n-1} a_{n-1,j}^2 \cdot x_j^{(2)}.$$

Аналогично можно вычислить остальные корни λ_j ($j = 3, \dots, n$) характеристического уравнения и соответствующие им векторы x_i^j .

Поскольку процесс решения СЛАУ уже был описан, приведем алгоритм (рис. 1.17) решения *частичной проблемы* собственных значений, то есть нахождения наибольшего собственного значения матрицы и соответствующего ему вектора, без дополнительных разъяснений. Заметим, что с таким же успехом можно было бы применить метод Зейделя.

1.4. Решение задач линейной алгебры в Mathcad

Ознакомимся с возможностями, которые предоставляет Mathcad для непосредственного решения задач линейной алгебры.

Работать с матрицами и векторами в пакете можно с помощью панели инструментов, выбирая необходимые пункты основного меню (см. приложение 1) или обращаясь к соответствующим функциям, которые делятся на группы: функции определения матриц и операции с блоками матриц, функции вычисления различных числовых характеристик матриц и функции, реализующие численные алгоритмы решения задач линейной алгебры. Рассмотрим наиболее часто используемые функции.

Функции определения матриц и операции с блоками матриц:

► $\text{matrix}(m, n, f)$ – создает и заполняет матрицу $A = \{a_{ij}\}$ размерности m на n , каждый элемент которой a_{ij} равен значению $f(i, j)$, пример приведен на рис. 1.18;

ORIGIN := 1

$$f(x, y) := |y^2 - x^2|$$

$$\underline{A} := \text{matrix}(4, 3, f) \quad A = \begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 3 \\ 4 & 3 & 0 \\ 9 & 8 & 5 \end{pmatrix}$$

ORIGIN := 1

$$i := 1..4 \quad d_i := 2 \cdot i \quad D := \text{diag}(d) \quad D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 8 \end{pmatrix}$$

Рис. 1.18 ▼ Создание матрицы, каждый элемент которой формируется по заданной функции

Рис. 1.19 ▼ Формирование диагональной матрицы

- $\text{diag}(v)$ – создает диагональную матрицу (рис. 1.19), элементы главной диагонали которой хранятся в векторе v ;
- $\text{identity}(n)$ – создает единичную матрицу (рис. 1.20) порядка n ;
- $\text{augment}(A, B)$ – из матриц A и B формируется третья матрица (рис. 1.21), первые столбцы которой содержат матрицу A , а последние – матрицу B (матрицы A и B должны иметь одинаковое число строк);

ORIGIN := 1

$$E := \text{identity}(3) \quad E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$B := \text{augment}(A, D)$

$$B = \begin{pmatrix} 0 & 1 & 4 & 2 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 4 & 0 & 0 \\ 4 & 3 & 0 & 0 & 0 & 6 & 0 \\ 9 & 8 & 5 & 0 & 0 & 0 & 8 \end{pmatrix}$$

Рис. 1.20 ▼ Создание единичной матрицы

Рис. 1.21 ▼ Формирование матрицы при помощи функции $\text{augment}(A, B)$

- $\text{stack}(A, B)$ – из матриц A и B формируется третья матрица (рис. 1.22), первые строки которой содержат матрицу A , а последние – матрицу B (матрицы A и B должны иметь одинаковое число столбцов);
- $\text{submatrix}(A, l, k, p, r)$ – формирует матрицу (рис. 1.23), которая является блоком матрицы A , расположенным в строках с l по k и в столбцах с p по r ($l < k, p < r$);

C := $\text{stack}(A, E)$

$$C = \begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 3 \\ 4 & 3 & 0 \\ 9 & 8 & 5 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$F := \text{submatrix}(B, 3, 4, 1, 2) \quad F = \begin{pmatrix} 4 & 3 \\ 9 & 8 \end{pmatrix}$$

Рис. 1.22 ▼ Формирование матрицы при помощи функции $\text{stack}(A, B)$

Рис. 1.23 ▼ Формирование матрицы при помощи функции $\text{submatrix}(A, l, k, p, r)$

- $\text{Re}(A)$ – возвращает матрицу (вектор) действительных частей матрицы (вектора) A с комплексными элементами (рис. 1.23);
- $\text{Im}(A)$ – возвращает матрицу (вектор) мнимых частей матрицы (вектора) A с комплексными элементами (рис. 1.24).

$$V := \begin{pmatrix} 1 + 2i \\ 2 + i \\ 3 - 5i \end{pmatrix} \quad \text{Re}(V) = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \text{Im}(V) = \begin{pmatrix} 2 \\ 1 \\ -5 \end{pmatrix}$$

Рис. 1.24 ▼ Применение функций $\text{Re}(A)$ и $\text{Im}(A)$

Функции вычисления различных числовых характеристик матриц:

- $\text{last}(v)$ – вычисление номера последней компоненты вектора v ;
- $\text{length}(v)$ – вычисление количества компонент вектора v ;
- $\text{rows}(A)$ – вычисление числа строк в матрице A ;
- $\text{cols}(A)$ – вычисление числа столбцов в матрице A ;
- $\text{max}(A)$ – вычисление наибольшего элемента в матрице (векторе) A ;
- $\text{min}(A)$ – вычисление наименьшего элемента в матрице (векторе) A ;
- $\text{mean}(A)$ – вычисление среднего значения матрицы (вектора) A ;
- $\text{median}(A)$ – вычисление медианы матрицы (вектора) A ;
- $\text{tr}(A)$ – вычисление следа (суммы диагональных элементов) квадратной матрицы A ;
- $\text{rank}(A)$ – вычисление ранга матрицы A ;
- $\text{norm1}(A)$ – вычисление первой нормы $\|A\|_1$ квадратной матрицы $A = \{a_{ij}\}$;
- $\text{norm2}(A)$ – вычисление второй нормы $\|A\|_2$ квадратной матрицы $A = \{a_{ij}\}$;
- $\text{norme}(A)$ – вычисление евклидовой нормы $\|A\|_e$ квадратной матрицы $A = \{a_{ij}\}$;
- $\text{normi}(A)$ – вычисление бесконечной нормы $\|A\|_i$ квадратной матрицы A ;
- $\text{cond1}(A)$ – определяет число обусловленности матрицы, вычисленное в первой норме $\|A\|_1$;
- $\text{cond2}(A)$ – определяет число обусловленности матрицы, вычисленное во второй норме $\|A\|_2$;
- $\text{condi}(A)$ – определяет число обусловленности матрицы, вычисленное в бесконечной норме $\|A\|_i$;

$$\begin{array}{lcl}
 \text{ORIGIN} := 1 & & \\
 \underline{A} := \begin{pmatrix} 1 & 2 & 0 & 4 \\ 5 & 1 & 7 & 3 \\ 0 & 6 & 2 & 3 \\ 4 & 5 & 6 & 0 \end{pmatrix} & \underline{V} := A^{(2)} & V = \begin{pmatrix} 2 \\ 1 \\ 6 \\ 5 \end{pmatrix} \\
 \text{last}(V) = 4 & & \text{length}(V) = 4 \\
 \text{rows}(A) = 4 & & \text{cols}(A) = 4 \\
 \text{max}(A) = 7 & & \text{min}(A) = 0 \\
 \text{mean}(A) = 3.063 & & \text{median}(A) = 3 \\
 \text{tr}(A) = 4 & & \text{rank}(A) = 4 \\
 \text{norm1}(A) = 15 & & \text{norm2}(A) = 13.221 \\
 \text{normi}(A) = 16 & & \text{norme}(A) = 15.199 \\
 \text{cond1}(A) = 15.652 & & \text{cond2}(A) = 13.004 \\
 \text{condi}(A) = 23.884 & & \text{conde}(A) = 15.625
 \end{array}$$

Рис. 1.25 ▼ Применение функций вычисления числовых характеристик матриц



- $\text{conde}(A)$ – определяет число обусловленности матрицы, вычисленное в евклидовой норме $\|A\|_e$.

Примеры применения этих функций приведены на рис. 1.25.

Функции, реализующие численные алгоритмы решения задач линейной алгебры:

- $\text{rref}(A)$ – приведение матрицы к ступенчатому виду путем элементарных операций со строками матрицы A ;
- $\text{eigenvals}(A)$ – вычисление собственных значений квадратной матрицы A ;
- $\text{eigenvecs}(A)$ – вычисление собственных векторов квадратной матрицы A ;
- $\text{eigenvec}(A, p)$ – вычисление собственного вектора матрицы A , отвечающего собственному значению p ;
- $\text{genvals}(A, B)$ – вычисляет вектор v обобщенных собственных значений, соответствующий решению уравнения $A \cdot x = v \cdot B \cdot x$;
- $\text{genvecs}(A, B)$ – вычисляет матрицу, содержащую собственные векторы, соответствующие обобщенным собственным значениям;
- $\text{lsolve}(A, b)$ – решение системы линейных алгебраических уравнений $A \cdot x = b$;
- $\text{lu}(A)$ – выполняет треугольное разложение матрицы A : $A = C \cdot L \cdot U$, где L и U – соответственно нижняя и верхняя треугольные матрицы, все четыре матрицы квадратные и одного порядка;
- $\text{qr}(A)$ – выполняет разложение матрицы A : $A = Q \cdot R$, где Q – ортогональная матрица, а R – верхняя треугольная матрица;
- $\text{cholesky}(A)$ – выполняет разложение матрицы A по схеме Халецкого: $A = L \cdot L^T$, где A – квадратная, симметричная, положительно определенная матрица, L – треугольная матрица;
- $\text{sort}(v)$ – сортировка элементов вектора v в порядке возрастания их значений;
- $\text{reverse}(v)$ – перестановка элементов вектора v в обратном порядке;
- $\text{csort}(A, n)$ – перестановка строк матрицы A таким образом, чтобы отсортированным оказался n -й столбец;
- $\text{rsort}(A, n)$ – перестановка столбцов матрицы A таким образом, чтобы отсортированной оказалась n -я строка.

Перейдем к решению конкретных задач линейной алгебры в Mathcad.

1.4.1. Задачи алгебры матриц

Примеры, приведенные в этом разделе, дают представление о возможностях пакета при выполнении действий над матрицами, вычислении определителей и решении матричных уравнений.

ЗАДАЧА 1.1. Для матриц A , B и C проверить выполнение следующих деств: $(A \cdot B) \cdot C = A \cdot (B \cdot C)$ и $(A^T + B) \cdot C = A^T \cdot C + B \cdot C$.
Решение приведено на рис. 1.26.

$$\underline{A} = \begin{pmatrix} 2 & 3 & -2 \\ 1 & 2 & 5 \end{pmatrix} \quad \underline{B} = \begin{pmatrix} 2 & -1 \\ 3 & 1 \\ 1 & 0 \end{pmatrix} \quad \underline{C} = \begin{pmatrix} 3 & -2 \\ 1 & 4 \end{pmatrix}$$

1. Проверка выполнения тождества $(AB)C = A(BC)$

$$\left[\begin{pmatrix} 2 & 3 & -2 \\ 1 & 2 & 5 \end{pmatrix} \cdot \begin{pmatrix} 2 & -1 \\ 3 & 1 \\ 1 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 3 & -2 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 34 & -18 \\ 40 & -22 \end{pmatrix} \quad \text{Вычисление правой части}$$

$$\begin{pmatrix} 2 & 3 & -2 \\ 1 & 2 & 5 \end{pmatrix} \cdot \left[\begin{pmatrix} 2 & -1 \\ 3 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 3 & -2 \\ 1 & 4 \end{pmatrix} \right] = \begin{pmatrix} 34 & -18 \\ 40 & -22 \end{pmatrix} \quad \text{Вычисление левой части}$$

$(AB)C = A(BC)$, следовательно, тождество выполнено

2. Проверка выполнения тождества $(A^T + B)C = A^T C + BC$

$$\left[\begin{pmatrix} 2 & 3 & -2 \\ 1 & 2 & 5 \end{pmatrix}^T + \begin{pmatrix} 2 & -1 \\ 3 & 1 \\ 1 & 0 \end{pmatrix} \right] \cdot \begin{pmatrix} 3 & -2 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 12 & -8 \\ 21 & 0 \\ 2 & 22 \end{pmatrix} \quad \text{Вычисление правой части}$$

$$\begin{pmatrix} 2 & 3 & -2 \\ 1 & 2 & 5 \end{pmatrix}^T \cdot \begin{pmatrix} 3 & -2 \\ 1 & 4 \end{pmatrix} + \begin{pmatrix} 2 & -1 \\ 3 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 3 & -2 \\ 1 & 4 \end{pmatrix} = \begin{pmatrix} 12 & -8 \\ 21 & 0 \\ 2 & 22 \end{pmatrix} \quad \text{Вычисление левой части}$$

$(A^T + B)C = A^T C + BC$, следовательно, тождество выполнено

Рис. 1.26 ▼ Решение задачи 1.1

ЗАДАЧА 1.2. Упростить выражение $\Delta_A - 3 \cdot \Delta_B + 2 \cdot x - 10$, где Δ_A и Δ_B делители матриц A и B соответственно.

$$\underline{A} = \begin{pmatrix} x & x \\ 4 & 5 \end{pmatrix} \quad \underline{B} = \begin{pmatrix} 1 & -1 & x \\ 2 & 0 & 1 \\ 3 & 4 & 1 \end{pmatrix}$$

$$\left| \begin{pmatrix} x & x \\ 4 & 5 \end{pmatrix} \right| - 3 \cdot \left| \begin{pmatrix} 1 & -1 & x \\ 2 & 0 & 1 \\ 3 & 4 & 1 \end{pmatrix} \right| + 2 \cdot x - 10 \rightarrow -21 \cdot x + 5$$

Рис. 1.27 ▼ Решение задачи 1.2

На рис. 1.27 показано, как можно решить эту задачу, воспользовавшись символическими операциями Mathcad.

ЗАДАЧА 1.3. Проверить, является ли матрица симметрической.

Решение приведено на рис. 1.28.

$$\underline{A} := \begin{pmatrix} 2 & -1 & 3 \\ -1 & 0 & 5 \\ 3 & 5 & -4 \end{pmatrix} \quad \text{Исходная матрица}$$

$$B := A^T$$

$$B = \begin{pmatrix} 2 & -1 & 3 \\ -1 & 0 & 5 \\ 3 & 5 & -4 \end{pmatrix} \quad \text{Транспонированная матрица}$$

Матрица A симметрическая, так как $A = A^T$

Рис. 1.28 ▼ Решение задачи 1.3

ЗАДАЧА 1.4. Проверить, является ли матрица кососимметрической.

На рис. 1.29 показано, как можно решить эту задачу.

$$\underline{A} := \begin{pmatrix} 0 & -1 & 3 \\ 1 & 0 & -5 \\ -3 & 5 & 0 \end{pmatrix} \quad \text{Исходная матрица}$$

$$A^T = \begin{pmatrix} 0 & 1 & -3 \\ -1 & 0 & 5 \\ 3 & -5 & 0 \end{pmatrix} \quad \text{Транспонированная матрица}$$

$$-1 \cdot A = \begin{pmatrix} 0 & 1 & -3 \\ -1 & 0 & 5 \\ 3 & -5 & 0 \end{pmatrix} \quad \text{Матрица } -A$$

Матрица A кососимметрическая, так как $A^T = -A$

Рис. 1.29 ▼ Решение задачи 1.4

ЗАДАЧА 1.5. Проверить, является ли матрица ортогональной.

Для решения этой задачи необходимо вычислить определитель заданной матрицы и убедиться в том, что он не равен нулю. Затем транспонировать исходную матрицу и найти обратную к ней. Если визуально сложно убедиться в том, что транспонированная матрица равна обратной, можно вычислить их разность. В результате должна получиться нулевая матрица (рис. 1.30).

$$A := \begin{pmatrix} \frac{1}{3} & 0 & \frac{2}{3} & -\frac{2}{3} \\ 0 & 1 & 0 & 0 \\ -\frac{2}{3} & 0 & \frac{1}{3} & -\frac{2}{3} \\ -\frac{2}{3} & 0 & \frac{2}{3} & \frac{1}{3} \end{pmatrix} \quad \begin{array}{l} \text{Исходная матрица} \\ |A| = -1 \quad \text{Определитель} \\ \text{не равен нулю} \end{array}$$

$$A^T = \begin{pmatrix} 0.333 & 0 & -0.667 & -0.667 \\ 0 & 1 & 0 & 0 \\ -0.667 & 0 & 0.333 & -0.667 \\ -0.667 & 0 & -0.667 & 0.333 \end{pmatrix} \quad A^{-1} = \begin{pmatrix} 0.333 & 0 & -0.667 & -0.667 \\ 0 & 1 & 0 & 0 \\ -0.667 & 0 & 0.333 & -0.667 \\ -0.667 & 0 & -0.667 & 0.333 \end{pmatrix}$$

$$A^T - A^{-1} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \begin{array}{l} \text{Транспонированная матрица} \\ \text{равна обратной} \end{array}$$

Матрица A является ортогональной,
так как $A^T = A^{-1}$ и определитель A не равен нулю

Рис. 1.30 ▼ Решение задачи 1.5

ЗАДАЧА 1.6. Проверить, является ли матрица A идемпотентной. Показать, что матрица $B = 2A - E$, где E – единичная матрица, инволютивна.

$$A := \begin{pmatrix} 6 & -15 \\ 2 & -5 \end{pmatrix} \quad \text{Исходная матрица}$$

$$A^2 = \begin{pmatrix} 6 & -15 \\ 2 & -5 \end{pmatrix} \quad \text{Возведение матрицы во вторую степень}$$

Матрица A идемпотентна, так как $A = A^2$

Формирование единичной матрицы

$$E := \text{identity}(2) \quad E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Формирование матрицы B

$$B := 2 \cdot A - E \quad B = \begin{pmatrix} 11 & -30 \\ 4 & -11 \end{pmatrix}$$

$$B^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{Матрица } B \text{ инволютивна, так как } E = B^2$$

Рис. 1.31 ▼ Решение задачи 1.6

Для того чтобы выяснить, является ли матрица A идемпотентной, ее необходимо возвести в квадрат и сравнить с исходной. В формировании матрицы B участвует единичная матрица, которую можно получить при помощи описанной выше функции $\text{identity}(n)$. Определить, является ли матрица B инволютивной, можно, сравнив ее с единичной матрицей. Подробно решение задачи приведено на рис. 1.31.



Операция возведения в целую неотрицательную степень определена только для квадратных матриц, так как при выполнении этой операции матрица умножается сама на себя заданное количество раз: $A^0 = E$, $A^1 = A$, $A^2 = A \cdot A$, $A^3 = A \cdot A \cdot A$, ...

ЗАДАЧА 1.7. Решить матричные уравнения $A \cdot X = B$ и $X \cdot A = B$ и выполнить проверку.

Матричное уравнение – это уравнение вида $A \cdot X = B$ или $X \cdot A = B$, где X – это неизвестная матрица. Если умножить матричное уравнение на матрицу, обратную к A , то оно примет вид: $A^{-1} \cdot A \cdot X = A^{-1} \cdot B$ или $X \cdot A \cdot A^{-1} = B \cdot A^{-1}$. Так как $A^{-1} \cdot A = A \cdot A^{-1} = E$, а $E \cdot X = X \cdot E = X$, то неизвестную матрицу X можно вычислить так: $X = A^{-1} \cdot B$ или $X = B \cdot A^{-1}$. Понятно, что матричное уравнение имеет единственное решение, если A и B – квадратные матрицы n -го порядка и определитель матрицы A не равен нулю. Как решить матричное уравнение в Mathcad, показано на рис. 1.32.

Вычислить неизвестную матрицу X :

$$\begin{pmatrix} 3 & 2 \\ 4 & 3 \end{pmatrix} \cdot X = \begin{pmatrix} -1 & 7 \\ 3 & 5 \end{pmatrix}$$

Решение

$$\begin{pmatrix} 3 & 2 \\ 4 & 3 \end{pmatrix}^{-1} \cdot \begin{pmatrix} -1 & 7 \\ 3 & 5 \end{pmatrix} = \begin{pmatrix} -9 & 11 \\ 13 & -13 \end{pmatrix}$$

Проверка:

$$\begin{pmatrix} 3 & 2 \\ 4 & 3 \end{pmatrix} \cdot \begin{pmatrix} -9 & 11 \\ 13 & -13 \end{pmatrix} = \begin{pmatrix} -1 & 7 \\ 3 & 5 \end{pmatrix}$$

Вычислить неизвестную матрицу X :

$$X \cdot A = B \quad A := \begin{pmatrix} 3 & 2 \\ 4 & 3 \end{pmatrix} \quad B := \begin{pmatrix} -1 & 7 \\ 3 & 5 \end{pmatrix}$$

$$X := B \cdot A^{-1} \quad X = \begin{pmatrix} -31 & 23 \\ -11 & 9 \end{pmatrix}$$

Проверка: $X \cdot A = \begin{pmatrix} -1 & 7 \\ 3 & 5 \end{pmatrix}$

Рис. 1.32 ▼ Решение матричного уравнения

1.4.2. Решение систем линейных уравнений

Задачи, рассматриваемые в этом разделе, посвящены различным приемам решения систем линейных алгебраических уравнений в Mathcad.

ЗАДАЧА 1.8. Решить систему линейных уравнений при помощи правила Крамера:

$$\begin{aligned} 2x_1 + x_2 - 5x_3 + x_4 &= 8, \\ x_1 - 3x_2 - 6x_4 &= 9, \\ 2x_2 - x_3 + 2x_4 &= -5, \\ x_1 + 4x_2 - 7x_3 + 6x_4 &= 0. \end{aligned}$$

Правило Крамера заключается в следующем. Если определитель $\Delta = \det A$ матрицы системы из n уравнений с n неизвестными $A \cdot x = b$ отличен от нуля, то система имеет единственное решение $x_1, x_2, \dots, x_n$, определяемое по формулам Крамера $x_i = \Delta_i / \Delta$, где Δ_i – определитель матрицы, полученной из матрицы

$$\begin{aligned} \underline{A} &:= \begin{pmatrix} 2 & 1 & -5 & 1 \\ 1 & -3 & 0 & -6 \\ 0 & 2 & -1 & 2 \\ 1 & 4 & -7 & 6 \end{pmatrix} & \underline{B} &:= \begin{pmatrix} 8 \\ 9 \\ -5 \\ 0 \end{pmatrix} & \text{Запись системы} \\ & & & & \text{в матричном виде} \\ \Delta &:= |A| & \Delta &= 27 & \text{Главный определитель} \\ & & & & \text{системы отличен от нуля} \\ \underline{A1} &:= \begin{pmatrix} 8 & 1 & -5 & 1 \\ 9 & -3 & 0 & -6 \\ -5 & 2 & -1 & 2 \\ 0 & 4 & -7 & 6 \end{pmatrix} & \underline{A2} &:= \begin{pmatrix} 2 & 8 & -5 & 1 \\ 1 & 9 & 0 & -6 \\ 0 & -5 & -1 & 2 \\ 1 & 0 & -7 & 6 \end{pmatrix} & \text{Формирование} \\ & & & & \text{вспомогатель-} \\ \underline{A3} &:= \begin{pmatrix} 2 & 1 & 8 & 1 \\ 1 & -3 & 9 & -6 \\ 0 & 2 & -5 & 2 \\ 1 & 4 & 0 & 6 \end{pmatrix} & \underline{A4} &:= \begin{pmatrix} 2 & 1 & -5 & 8 \\ 1 & -3 & 0 & 9 \\ 0 & 2 & -1 & -5 \\ 1 & 4 & -7 & 0 \end{pmatrix} & \text{ных матриц} \end{aligned}$$

Вычисление определителей вспомогательных матриц

$$\Delta_1 := |A1| \quad \Delta_2 := |A2| \quad \Delta_3 := |A3| \quad \Delta_4 := |A4|$$

$$\Delta_1 = 81 \quad \Delta_2 = -108 \quad \Delta_3 = -27 \quad \Delta_4 = 27$$

Решение системы по формулам Крамера

$$x_1 := \frac{\Delta_1}{\Delta} \quad x_2 := \frac{\Delta_2}{\Delta} \quad x_3 := \frac{\Delta_3}{\Delta} \quad x_4 := \frac{\Delta_4}{\Delta}$$

$$x_1 = 3 \quad x_2 = -4 \quad x_3 = -1 \quad x_4 = 1$$

Рис. 1.33 ▼ Решение системы линейных уравнений по формулам Крамера

системы A заменой i -го столбца столбцом свободных членов b . Итак, для решения поставленной задачи необходимо выполнить следующие действия:

1. Представить систему в матричном виде, то есть сформировать матрицу системы A и вектор правых частей b .
2. Вычислить главный определитель Δ .
3. Сформировать вспомогательные матрицы (удобно скопировать матрицу A несколько раз и последовательно заменять в ней столбцы на вектор b) для вычисления определителей Δ_i .
4. Вычислить определители Δ_i .
5. Найти решение системы по формуле $x_i = \Delta_i / \Delta$.

На рис. 1.33 приведен фрагмент рабочего документа, содержащий решение поставленной задачи.

ЗАДАЧА 1.9. Решить систему линейных уравнений из задачи 1.8 методом обратной матрицы.

Метод обратной матрицы: для системы из n линейных уравнений с n неизвестными $A \cdot x = b$, при условии что определитель матрицы A не равен нулю, единственное решение можно представить в виде $x = A^{-1} \cdot b$ (вывод формулы см. в задаче 3.7). Итак, для того, чтобы решить систему линейных уравнений методом обратной матрицы, необходимо выполнить следующие действия:

1. Сформировать матрицу коэффициентов и вектор свободных членов заданной системы.
2. Решить систему, представив вектор неизвестных как произведение матрицы, обратной к матрице системы, и вектора свободных членов (рис. 1.34).

$$A := \begin{pmatrix} 2 & 1 & -5 & 1 \\ 1 & -3 & 0 & -6 \\ 0 & 2 & -1 & 2 \\ 1 & 4 & -7 & 6 \end{pmatrix} \quad b := \begin{pmatrix} 8 \\ 9 \\ -5 \\ 0 \end{pmatrix} \quad \text{Запись системы в матричном виде}$$

$$x := A^{-1} \cdot b \quad x = \begin{pmatrix} 3 \\ -4 \\ -1 \\ 1 \end{pmatrix} \quad \text{Решение системы}$$

Проверка

$$A \cdot x - b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Рис. 1.34 ▼ Решение системы линейных уравнений методом обратной матрицы

ЗАДАЧА 1.10. Решить систему линейных уравнений из задачи 1.8 методом Гаусса.

Решение системы линейных уравнений при помощи *метода Гаусса* подробно описано в разделе 1.2.1. Напомним, что оно основывается на том, что от заданной системы переходят к эквивалентной системе, которая решается проще, чем исходная система. В связи с этим порядок решения задачи в Mathcad (рис. 1.35) следующий:

1. Сформировать матрицу коэффициентов и вектор свободных членов заданной системы.
2. Сформировать расширенную матрицу системы при помощи функции `augment(A,b)`.
3. Используя функцию `rref(A)`, привести расширенную матрицу к ступенчатому виду.
4. Получить решение системы, выделив последний столбец матрицы, полученной в предыдущем пункте.
5. Выполнить вычисление $A \cdot x - B$. Если в результате получился нулевой вектор, задача решена верно.

ORIGIN := 1

$$P := \text{augment}(A, b) \quad P = \begin{pmatrix} 2 & 1 & -5 & 1 & 8 \\ 1 & -3 & 0 & -6 & 9 \\ 0 & 2 & -1 & 2 & -5 \\ 1 & 4 & -7 & 6 & 0 \end{pmatrix} \quad \text{Расширенная матрица}$$

$$R := \text{rref}(P) \quad R = \begin{pmatrix} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & -4 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \text{Ступенчатая матрица}$$

$$n := \text{cols}(R) \quad x := R^{(n)} \quad x = \begin{pmatrix} 3 \\ -4 \\ -1 \\ 1 \end{pmatrix} \quad \text{Решение системы}$$

Проверка

$$A \cdot x - b = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Рис. 1.35 ▼ Решение системы линейных уравнений методом Гаусса

ЗАДАЧА 1.11. Решить систему линейных уравнений с помощью LU-разложения:

$$\begin{aligned} 2x_1 - x_2 + x_3 &= 0, \\ 3x_1 + 2x_2 - 5x_3 &= 1, \\ x_1 + 3x_2 - 2x_3 &= 4. \end{aligned}$$

Идея метода LU-разложения сформулирована в разделе 1.2.6 этой главы. В соответствии со средствами, которыми располагает Mathcad, решение поставленной задачи будет выглядеть так:

1. Сформируем матрицу коэффициентов A и вектор свободных членов B заданной системы.
2. Воспользовавшись функцией $lu(A)$, создадим матрицу LU – треугольное разложение матрицы A , состоящую из трех подматриц размерностью $n \times n$ (рис. 1.36).

ORIGIN := 1

$$\underline{A} := \begin{pmatrix} 2 & -1 & 1 \\ 3 & 2 & -5 \\ 1 & 3 & -2 \end{pmatrix} \quad \underline{B} := \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$$

Разложение матрицы A на множители L и U

$$\underline{LU} := lu(\underline{A}) \quad \underline{LU} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 2 & -1 & 1 \\ 0 & 0 & 1 & 0.5 & 1 & 0 & 0 & 3.5 & -2.5 \\ 0 & 1 & 0 & 1.5 & 1 & 1 & 0 & 0 & -4 \end{pmatrix}$$

Рис. 1.36 ▼ Треугольное разложение матрицы A

3. Выделим из матрицы LU первую подматрицу $C = \{c_{kp} = lu_{ij}, k = 1, \dots, n, p = 1, \dots, n, i = 1, \dots, n, j = 1, \dots, n\}$, содержащую информацию о перестановке строк в матрице A при формировании ее множителей, иначе это единичная матрица, в которой строки переставлены соответствующим образом, в нашем случае (рис. 1.37) вторая и третья строки поменялись местами, а поскольку в задаче речь идет о решении системы, то таким же образом необходимо поменять элементы вектора B , то есть вычислить выражение $b = C \cdot B$.
4. Выделим из матрицы LU вторую и третью подматрицы: $L = \{l_{kp} = lu_{ij}, k = 1, \dots, n, p = 1, \dots, n, i = 1, \dots, n, j = (n + 1), \dots, 2n\}$ – нижнюю треугольную матрицу, $U = \{U_{kp} = lu_{ij}, k = 1, \dots, n, p = 1, \dots, n, i = 1, \dots, n, j = (2n + 1), \dots, 3n\}$ – верхнюю треугольную матрицу, перемножим матрицы L и U , чтобы убедиться, что в результате получится матрица A (рис. 3.33).

Выделение подматриц из матрицы LU

$C := \text{submatrix}(LU, 1, 3, 1, 3)$ Матрица перестановок

$L := \text{submatrix}(LU, 1, 3, 4, 6)$ Нижняя треугольная матрица

$U := \text{submatrix}(LU, 1, 3, 7, 9)$ Верхняя треугольная матрица

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad b := C \cdot b \quad b = \begin{pmatrix} 0 \\ 4 \\ 1 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ 1.5 & 1 & 1 \end{pmatrix} \quad U = \begin{pmatrix} 2 & -1 & 1 \\ 0 & 3.5 & -2.5 \\ 0 & 0 & -4 \end{pmatrix} \quad L \cdot U = \begin{pmatrix} 2 & -1 & 1 \\ 1 & 3 & -2 \\ 3 & 2 & -5 \end{pmatrix}$$

Рис. 1.37 ▼ Выделение подматриц из разложения матрицы A

5. Перейдем к решению системы уравнений $L \cdot y = b$, для чего сформируем расширенную матрицу Lb системы при помощи функции $\text{augment}(L, b)$.
6. Используя функцию $\text{rref}(Lb)$, приведем расширенную матрицу к ступенчатому виду.
7. Получим решение системы $L \cdot y = b$, выделив последний столбец матрицы Y (рис. 1.38).

Решение системы $Ly = b$

$$Lb := \text{augment}(L, b) \quad Lb = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 4 \\ 1.5 & 1 & 1 & 1 \end{pmatrix}$$

$$Y := \text{rref}\left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 4 \\ 1.5 & 1 & 1 & 1 \end{pmatrix}\right) \quad Y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & -3 \end{pmatrix} \quad y := Y^{(4)}$$

Рис. 1.38 ▼ Решение системы уравнений $L \cdot y = b$

8. Зная U и y , можно решить систему $U \cdot x = y$ аналогично системе $L \cdot y = b$ (рис. 1.39).

ЗАДАЧА 1.12. Решить систему линейных уравнений из задачи 1.11 с помощью QR-разложения.

Метод решения линейных систем с помощью QR-разложения описан в разделе 1.2.6. Как решить эту задачу средствами Mathcad, показано на рис. 1.40. Обратите внимание на то, что для представления матрицы в виде произведения ортогональной матрицы и верхней треугольной используется функция $\text{qr}(A)$.

Решение системы $Ux=y$

$$Uy := \text{augment}(U, y) \quad Uy = \begin{pmatrix} 2 & -1 & 1 & 0 \\ 0 & 3.5 & -2.5 & 4 \\ 0 & 0 & -4 & -3 \end{pmatrix}$$

$$X := \text{rref}\left(\begin{pmatrix} 2 & -1 & 1 & 0 \\ 0 & 3.5 & -2.5 & 4 \\ 0 & 0 & -4 & -3 \end{pmatrix}\right) \quad X = \begin{pmatrix} 1 & 0 & 0 & 0.464 \\ 0 & 1 & 0 & 1.679 \\ 0 & 0 & 1 & 0.75 \end{pmatrix}$$

$$x := X^{(4)} \quad x = \begin{pmatrix} 0.464 \\ 1.679 \\ 0.75 \end{pmatrix} \quad \text{Решение системы } Ax=b$$

Рис. 1.39 ▽ Решение системы уравнений $U \cdot x = y$

Разложение матрицы A на множители Q и R

$$QR := \text{qr}(A) \quad QR = \begin{pmatrix} 0.535 & -0.617 & 0.577 & 3.742 & 1.871 & -4.009 \\ 0.802 & 0.154 & -0.577 & 0 & 3.24 & -2.932 \\ 0.267 & 0.772 & 0.577 & 0 & 0 & 2.309 \end{pmatrix}$$

Выделение ортогональной подматрицы Q

$$Q := \text{submatrix}(QR, 1, 3, 1, 3) \quad Q = \begin{pmatrix} 0.535 & -0.617 & 0.577 \\ 0.802 & 0.154 & -0.577 \\ 0.267 & 0.772 & 0.577 \end{pmatrix}$$

Выделение треугольной подматрицы R

$$R := \text{submatrix}(QR, 1, 3, 4, 6) \quad R = \begin{pmatrix} 3.742 & 1.871 & -4.009 \\ 0 & 3.24 & -2.932 \\ 0 & 0 & 2.309 \end{pmatrix}$$

$$Q \cdot R = \begin{pmatrix} 2 & -1 & 1 \\ 3 & 2 & -5 \\ 1 & 3 & -2 \end{pmatrix}$$

Решение системы $Qy=b$

$$Qb := \text{augment}(Q, b) \quad Qb = \begin{pmatrix} 0.535 & -0.617 & 0.577 & 0 \\ 0.802 & 0.154 & -0.577 & 1 \\ 0.267 & 0.772 & 0.577 & 4 \end{pmatrix}$$

$$Y := \text{rref}(Qb)$$

$$Y = \begin{pmatrix} 1 & 0 & 0 & 1.871 \\ 0 & 1 & 0 & 3.24 \\ 0 & 0 & 1 & 1.732 \end{pmatrix} \quad y := Y^{(4)} \quad y = \begin{pmatrix} 1.871 \\ 3.24 \\ 1.732 \end{pmatrix}$$

Решение системы $Rx=y$

$$Ry := \text{augment}(R, y) \quad Ry = \begin{pmatrix} 3.742 & 1.871 & -4.009 & 1.871 \\ 0 & 3.24 & -2.932 & 3.24 \\ 0 & 0 & 2.309 & 1.732 \end{pmatrix}$$

$$X := \text{rref}(Ry) \quad x := X^{(4)} \quad x = \begin{pmatrix} 0.464 \\ 1.679 \\ 0.75 \end{pmatrix} \quad \text{Вектор-решение системы } Ax=b$$

Рис. 1.40 ▽ Решение системы уравнений $A \cdot x = b$ методом QR-разложения

ЗАДАЧА 1.13. Исследовать систему на совместность и, если возможно, решить ее:

$$\begin{aligned}x_1 + 2x_2 + 5x_3 &= -9, \\x_1 - x_2 + 3x_3 &= 2, \\3x_1 - 6x_2 - x_3 &= 25.\end{aligned}$$

Поясним ход решения этой задачи, приведенный на рис. 1.41. После ввода исходных данных, то есть матрицы коэффициентов системы и вектора правых частей, формируется расширенная матрица системы. Затем выполняется вычисление рангов матрицы коэффициентов и расширенной матрицы системы. В нашем случае ранги матриц равны и совпадают с количеством неизвестных: $r(A|b) = r(A) = 3$, значит, система совместна и имеет единственное решение. Отыскание решения системы линейных уравнений выполняется при помощи функции `lsolve(A,b)`, аргументами которой являются матрица коэффициентов и вектор правых частей, а результатом, соответственно, вектор-решение.

$$\begin{aligned}A &:= \begin{pmatrix} 1 & 2 & 5 \\ 1 & -1 & 3 \\ 3 & -6 & -1 \end{pmatrix} & b &:= \begin{pmatrix} -9 \\ 2 \\ 25 \end{pmatrix} & \text{Матрица системы и} \\ & & & \text{вектор свободных} \\ & & & \text{коэффициентов} \\ \\ Ab &:= \text{augment}(A,b) & Ab &= \begin{pmatrix} 1 & 2 & 5 & -9 \\ 1 & -1 & 3 & 2 \\ 3 & -6 & -1 & 25 \end{pmatrix} & \text{Расширенная} \\ & & & \text{матрица} \\ & & & \text{системы} \\ \\ \text{rank}(A) &= 3 & \text{Ранг матрицы системы} \\ \text{rank}(Ab) &= 3 & \text{Ранг расширенной матрицы системы} \\ \\ \text{lsolve}(A,b) &= \begin{pmatrix} 2 \\ -3 \\ -1 \end{pmatrix} & \text{Единственное решение системы}\end{aligned}$$

Рис. 1.41 ▼ Решение задачи 1.13

ЗАДАЧА 1.14. Исследовать систему на совместность и, если возможно, решить ее:

$$\begin{aligned}x_1 - 5x_2 - 8x_3 + x_4 &= 3, \\3x_1 + x_2 - 3x_3 - 5x_4 &= 1, \\x_1 - 7x_3 + 2x_4 &= -5, \\11x_2 + 20x_3 - 9x_4 &= 2.\end{aligned}$$

Ход решения задачи приведен на рис. 1.42. Вычисление рангов матрицы системы и расширенной матрицы показывает, что ранг расширенной матрицы больше ранга матрицы системы $r(A|b) > r(A)$, что означает несовместность системы. Следовательно, попытка решить эту систему при помощи функции $\text{lsolve}(A, b)$ завершится неудачей.

$$A := \begin{pmatrix} 1 & -5 & -8 & 1 \\ 3 & 1 & -3 & -5 \\ 1 & 0 & -7 & 2 \\ 0 & 11 & 20 & -9 \end{pmatrix} \quad b := \begin{pmatrix} 3 \\ 1 \\ -5 \\ 2 \end{pmatrix}$$

Исследование системы на совместность

$$\begin{aligned} Ab &:= \text{augment}(A, b) \\ \text{rank}(A) &= 3 \\ \text{rank}(Ab) &= 4 \end{aligned} \quad Ab = \begin{pmatrix} 1 & -5 & -8 & 1 & 3 \\ 3 & 1 & -3 & -5 & 1 \\ 1 & 0 & -7 & 2 & -5 \\ 0 & 11 & 20 & -9 & 2 \end{pmatrix}$$

$$\text{lsolve}(A, b) = \text{singular matrix}$$

$$|A| = 0$$

Рис. 1.42 ▼ Решение задачи 1.14

ЗАДАЧА 1.15. Исследовать систему на совместность и, если возможно, решить ее:

$$\begin{aligned} 4x_1 + x_2 - 3x_3 - x_4 &= 0, \\ 2x_1 + 3x_2 + x_3 - 5x_4 &= 0, \\ x_1 - 2x_2 - 2x_3 + 3x_4 &= 0. \end{aligned}$$

$$A := \begin{pmatrix} 4 & 1 & -3 & -1 \\ 2 & 3 & 1 & -5 \\ 1 & -2 & -2 & 4 \end{pmatrix} \quad b := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\begin{aligned} Ab &:= \text{augment}(A, b) \\ \text{rank}(A) &= 3 \\ \text{rank}(Ab) &= 3 \end{aligned} \quad Ab = \begin{pmatrix} 4 & 1 & -3 & -1 & 0 \\ 2 & 3 & 1 & -5 & 0 \\ 1 & -2 & -2 & 4 & 0 \end{pmatrix}$$

$$\text{lsolve}(A, b) =$$

This matrix must be square.

Рис. 1.43 ▼ Решение задачи 1.15

В процессе вычислений (рис. 1.43) выясняется, что ранг расширенной матрицы равен рангу матрицы системы $r(A|b) = r(A) = 3$, но меньше, чем количество неизвестных системы $r(A|b) = r(A) < 4$. Значит, система совместна, но имеет бесконечное множество решений и использование функции `lsolve(A,b)` для решения заданной системы не обосновано.

Итак, проанализировав все изложенные выше методы решения систем линейных уравнений при помощи Mathcad, можно заключить, что одним из самых простых способов является использование функции `lsolve(A,b)`. Если же при обращении к этой функции возникает сообщение об ошибке, как было показано в задачах 1.14 и 1.15, то следует выяснить, имеет ли система решение вообще и является ли оно единственным.

ЗАДАЧА 1.16. Найти общее и частное решение системы:

$$\begin{aligned}x_1 - 2x_2 + 4x_3 + 5x_4 &= 0, \\ 3x_1 - x_2 - 3x_3 + 5x_4 &= 0, \\ x_1 + 3x_2 - 11x_3 - 5x_4 &= 0.\end{aligned}$$

Приведем пояснения к рис. 1.44, на котором отображено решение данной задачи.

Исследование системы показывает, что она совместна и неопределенна, так как ранг расширенной матрицы равен рангу матрицы системы и меньше числа ее неизвестных. А это значит, что система имеет бесконечное множество решений.

Приведем расширенную матрицу исходной системы к ступенчатому виду. Матрица, полученная в результате, содержит нулевую строчку, которую можно отбросить и составить систему, эквивалентную данной. В нашем случае получилась система из двух уравнений с четырьмя неизвестными. Найдем общее решение этой системы, разрешив ее относительно базисных переменных x_1 и x_2 , а переменные x_3 и x_4 будем считать свободными.

Для отыскания общего решения системы воспользуемся *решающим блоком*. Начинается этот блок с ключевого слова `Given`, которое необходимо ввести с клавиатуры. Затем правее и ниже ключевого слова записываются уравнения системы. Обратите внимание на то, что знак равенства в уравнениях вводится при помощи клавиш `Ctrl+=` или выбирается на панели инструментов **Boolean** (Логические операторы). Правее и ниже последнего уравнения системы вводится функция `Find(x1, x2, ..., xn)`, в скобках перечисляются имена переменных, значения которых нужно найти. После ввода функции с именами искомых переменных необходимо выбрать знак стрелки на панели инструментов **Symbolic** (Символьные операторы) или нажать комбинацию клавиш `Ctrl+.` Вычисленное решение системы будет отображено после щелчка мышью по свободной

области документа. В нашем случае вектор, появившийся справа от стрелки, содержит общее решение системы, в нем базисные переменные x_1 и x_2 выражены через свободные переменные x_3 и x_4 .

Частное решение системы найдем так. Составим функцию, определяющую общее решение, в которой в качестве аргументов будут выступать свободные переменные x_3 и x_4 . Подставляя в эту функцию любые значения, будем получать частные решения неоднородной системы линейных уравнений.

$$A := \begin{pmatrix} 1 & -2 & 4 & 5 \\ 3 & -1 & -3 & 5 \\ 1 & 3 & -11 & -5 \end{pmatrix} \quad b := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\text{rank}(A) = 2$$

Ранг матрицы системы

$$\text{rank}(\text{rref}(\text{augment}(A, b))) = 2$$

Ранг расширенной матрицы

$$\text{rref}(\text{augment}(A, b)) = \begin{pmatrix} 1 & 0 & -2 & 1 & 0 \\ 0 & 1 & -3 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Переход к эквивалентной системе уравнений

Решение системы, эквивалентной данной

Given

$$x_1 - 2x_3 + x_4 = 0$$

$$x_2 - 3x_3 - 2x_4 = 0$$

$$\text{Find}(x_1, x_2) \rightarrow \begin{pmatrix} 2 \cdot x_3 - x_4 \\ 3 \cdot x_3 + 2 \cdot x_4 \end{pmatrix}$$

Свободные переменные x_3, x_4 Базисные переменные x_1, x_2

Общее решение

Частное решение

$$X(x_3, x_4) := \begin{pmatrix} 2 \cdot x_3 - x_4 \\ 3 \cdot x_3 + 2 \cdot x_4 \\ x_3 \\ x_4 \end{pmatrix}$$

$$x_3=0, x_4=1$$

$$x_3=1, x_4=0$$

$$X(0, 1) = \begin{pmatrix} -1 \\ 2 \\ 0 \\ 1 \end{pmatrix}$$

$$X(1, 0) = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 0 \end{pmatrix}$$

Рис. 1.44 ▼ Решение задачи 1.16

1.5.3. Собственные значения, собственные векторы, нормы, числа обусловленности матриц

В этом разделе приведены примеры решения в Mathcad задач на собственные значения и собственные векторы. Кроме того, продемонстрированы возможности пакета при вычислении норм и чисел обусловленности матриц.

ЗАДАЧА 1.17. Найти собственные значения и собственные векторы матрицы A .

Решим задачу, не пользуясь встроенными функциями Mathcad (рис. 1.45). Для этого выполним следующие операции:

1. Введем заданную матрицу A .
2. Сформируем единичную матрицу E того же порядка.
3. Вычислим собственные значения матрицы, решив характеристическое уравнение $\det(A - \lambda \cdot E) = 0$.
4. Составим собственные подпространства матрицы A для соответствующих собственных значений.
5. Найдем базисные решения систем $(A - \lambda \cdot E) \cdot x = 0$ (см. задачу 1.16).
6. Определим собственные векторы y матрицы A , отыскав частные решения систем $(A - \lambda \cdot E) \cdot x = 0$.
7. Выполним проверку, убедившись, что $(A - \lambda \cdot E) \cdot y = 0$.

$$A := \begin{pmatrix} 3 & -2 \\ -4 & 1 \end{pmatrix} \quad E := \text{identity}(2) \quad E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$A - \lambda \cdot E \rightarrow \begin{pmatrix} 3 - \lambda & -2 \\ -4 & 1 - \lambda \end{pmatrix}$$

$$|A - \lambda \cdot E| \rightarrow -5 - 4 \cdot \lambda + \lambda^2 \quad \text{Характеристическое уравнение}$$

$$(\lambda + 1) \cdot (\lambda - 5) \quad \lambda_1 := -1 \quad \lambda_2 := 5 \quad \text{Собственные значения}$$

$$A - \lambda_1 \cdot E = \begin{pmatrix} 4 & -2 \\ -4 & 2 \end{pmatrix} \quad \text{Собственное подпространство } A \text{ для } \lambda_1$$

$$A - \lambda_2 \cdot E = \begin{pmatrix} -2 & -2 \\ -4 & -4 \end{pmatrix} \quad \text{Собственное подпространство } A \text{ для } \lambda_2$$

Общее решение системы $(A - \lambda_1 E)x = 0$

$$\text{rref}\left[\text{augment}\left(\begin{pmatrix} 4 & -2 \\ -4 & 2 \end{pmatrix}\right)\right] = \begin{pmatrix} 1 & -0.5 \\ 0 & 0 \end{pmatrix} \quad x_1 - \frac{1}{2}x_2 = 0 \quad x_1 = \frac{1}{2}x_2$$

Общее решение системы $(A - \lambda_2 E)x = 0$

$$\text{rref}\left[\text{augment}\left(\begin{pmatrix} -2 & -2 \\ -4 & -4 \end{pmatrix}\right)\right] = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \quad x_1 + x_2 = 0 \quad x_1 = -x_2$$

Частные решения систем

$$y_1 := \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad y_2 := \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad \text{Собственные векторы}$$

$$\text{Проверка} \quad (A - \lambda_1 \cdot E) \cdot y_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (A - \lambda_2 \cdot E) \cdot y_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Рис. 1.45 ▼ Решение задачи 1.17

На рис. 1.46 приведен фрагмент рабочего документа, из которого понятно, что решение будет менее громоздким, если применить встроенные функции Mathcad $\text{eigenvals}(A)$ и $\text{eigenvec}(A, p)$.

$$A := \begin{pmatrix} 3 & -2 \\ -4 & 1 \end{pmatrix} \quad E := \text{identity}(2)$$

Собственные значения матрицы A

$$\text{eigenvals}(A) = \begin{pmatrix} 5 \\ -1 \end{pmatrix} \quad \lambda_1 := 5 \quad \lambda_2 := -1$$

Собственный вектор для λ_1

$$y_1 := \text{eigenvecs}(A, \lambda_1) \quad y_1 = \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix}$$

Собственный вектор для λ_2

$$y_2 := \text{eigenvecs}(A, \lambda_2) \quad y_2 = \begin{pmatrix} 0.447 \\ 0.894 \end{pmatrix}$$

$$\text{Проверка} \quad (A - \lambda_1 \cdot E) \cdot y_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (A - \lambda_2 \cdot E) \cdot y_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Рис. 1.46 ▼ Вычисление собственных значений и собственных векторов



Полученные с помощью встроенных функций собственные векторы отличны от тех, которые были рассчитаны вручную. Происходит это потому, что решается однородная система линейных уравнений, которая имеет бесконечное множество частных решений. Заметим, что даже одна и та же функция вычисления собственных векторов может давать различные результаты.

ЗАДАЧА 1.18. Привести заданную матрицу A к диагональному виду.

Задача состоит в том, чтобы для квадратной матрицы A подобрать такую матрицу C, чтобы матрица $B = C^{-1}AC$ имела диагональный вид. Эта задача связана с теорией собственных значений, так как разрешима только в том случае, если матрица C состоит из собственных векторов матрицы A.

$$A := \begin{pmatrix} 2 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 2 & 2 \end{pmatrix} \quad b := \text{eigenvals}(A) \quad b = \begin{pmatrix} 5.4 \\ -1 \\ -2.4 \end{pmatrix} \quad \text{Собственные значения матрицы A}$$

Матрица собственных векторов матрицы A

$$C := \text{eigenvecs}(A) \quad C = \begin{pmatrix} 0.6797649 & 0.7071068 & 0.1326443 \\ 0.1875472 & 0 & 0.8653849 \\ 0.7090457 & -0.7071068 & -0.4832333 \end{pmatrix}$$

Формирование диагональной матрицы

$$C^{-1} \cdot A \cdot C = \begin{pmatrix} 5.4 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2.4 \end{pmatrix} \quad B := \text{diag}(b) \quad B = \begin{pmatrix} 5.4 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -2.4 \end{pmatrix}$$

Рис. 1.47 ▼ Формирование диагональной матрицы

На рис. 1.47 показано, как можно решить поставленную задачу двумя способами. Первый способ сводится к вычислению значения выражения $C^{-1}AC$, для чего предварительно, при помощи функции `eigenvecs(A)` формируется матрица C , каждый столбец которой соответствует собственному вектору матрицы A . Результаты вычислений, полученные подобным образом, показывают, что главная диагональ диагональной матрицы совпадает с собственными значениями матрицы A . Поэтому второй способ решения этой задачи сводится к формированию диагональной матрицы при помощи функции `diag(b)`, где b – вектор, элементы которого соответствуют собственным значениям матрицы A .

ЗАДАЧА 1.19. Преобразовать квадратичную форму $6x_1^2 + 5x_2^2 + 7x_3^2 + 4x_1x_2 + 4x_2x_3$ к каноническому виду.

Под *квадратичной формой* относительно переменных $x_1, x_2, \dots, x_n$ понимают выражение вида $x^T A x$, где $x = (x_1, x_2, \dots, x_n)^T$ и $A = \{a_{ij}\}$ – симметрическая матрица, a_{ij} – действительные числа. Матрица A называется *матрицей квадратичной формы*.

Постановка задачи такова: найти такую ортогональную матрицу C , чтобы после введения новых переменных $y_1, y_2, \dots, y_n$ при помощи уравнения $x = C \cdot y$ данная квадратичная форма содержала только слагаемые с квадратами текущих координат: $x^T \cdot A \cdot x \rightarrow \lambda_1 y_1^2 + \lambda_2 y_2^2 + \dots + \lambda_n y_n^2$. Такой вид квадратичной формы называют *каноническим*.

Замена переменных $x = C \cdot y$ приведет к тому, что форма $x^T \cdot A x$ перейдет в форму $y^T \cdot (C^T \cdot A \cdot C) \cdot y$, которая должна содержать только слагаемые с квадратами переменных. Таким образом, задача равнозначна следующей: найти такую ортогональную матрицу C , чтобы матрица $C^T \cdot A \cdot C$ имела диагональный вид. Это всегда возможно для симметрической матрицы A , если в качестве столбцов искомой матрицы C выбрать систему собственных векторов матрицы A . Собственные векторы матрицы A , стоящие в столбцах C , называют *главными осями* квадратичной формы, а процесс преобразования квадратичной формы к каноническому виду называют *приведением к главным осям*.

$$A := \begin{pmatrix} 6 & -2 & 2 \\ -2 & 5 & 0 \\ 2 & 0 & 7 \end{pmatrix} \quad C := \text{eigenvecs}(A) \rightarrow \begin{pmatrix} \frac{1}{3} & \frac{-2}{3} & \frac{-2}{3} \\ \frac{-2}{3} & \frac{-2}{3} & \frac{1}{3} \\ \frac{-2}{3} & \frac{1}{3} & \frac{-2}{3} \end{pmatrix}$$

$$\left[C \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \right]^T \cdot A \cdot C \cdot \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \rightarrow 6 \cdot y_1^2 + 3 \cdot y_2^2 + 9 \cdot y_3^2$$

Рис. 1.48 ▼ Преобразование квадратичной формы к главным осям

Решение задачи, выполненное при помощи встроенной функции и символьных возможностей пакета, приведено на рис. 1.48.

ЗАДАЧА 1.20. Найти решение обобщенной задачи о собственных значениях для матриц A и B .

Для решения обобщенной задачи о собственных значениях в Mathcad определены две функции. Функция $\text{genvals}(A, B)$ вычисляет собственные значения μ , определяемые соотношением $A \cdot y = \mu \cdot B \cdot y$ ($y \neq 0$), а функция $\text{genvecs}(A, B)$ – собственные векторы y , соответствующие этим собственным значениям. На рис. 1.49 показано, как решается поставленная задача при помощи этих функций.

$$A := \begin{pmatrix} 1 & -3 \\ -3 & 4 \end{pmatrix} \quad B := \begin{pmatrix} 1 & 2 \\ -3 & 1 \end{pmatrix}$$

Вектор обобщенных собственных значений

$$v := \text{genvals}(A, B) \quad v = \begin{pmatrix} 1 \\ -0.714 \end{pmatrix}$$

Матрица обобщенных собственных векторов

$$X := \text{genvecs}(A, B) \quad X = \begin{pmatrix} 1 & -0.676 \\ 0 & -0.737 \end{pmatrix}$$

Проверка

$$(A - v_0 \cdot B) \cdot X^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \quad (A - v_1 \cdot B) \cdot X^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Рис. 1.49 ▼ Решение обобщенной задачи о собственных значениях

Фрагмент рабочего документа, изображенный на рис. 1.50, демонстрирует, как можно решить эту же задачу, учитывая, что векторы y и числа μ обязательно являются собственными векторами и собственными значениями матрицы $B^{-1}A$.

$$C := B^{-1} \cdot A \quad C = \begin{pmatrix} 1 & -1.571 \\ 0 & -0.714 \end{pmatrix}$$

Вектор обобщенных
собственных значений

$$\mu := \text{eigenvals}(C)$$

$$\mu = \begin{pmatrix} 1 \\ -0.714 \end{pmatrix}$$

Проверка

$$(A - \mu_0 \cdot B) \cdot Y^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Матрица обобщенных
собственных векторов

$$Y := \text{eigenvecs}(C)$$

$$Y = \begin{pmatrix} 1 & 0.676 \\ 0 & 0.737 \end{pmatrix}$$

$$(A - \mu_1 \cdot B) \cdot Y^{(1)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Рис. 1.50 ▼ Другое решение обобщенной задачи о собственных значениях

Напомним, что матрицы обобщенных собственных векторов у на рис. 1.45 и 1.46 различны, потому что даже одна и та же функция вычисления собственных векторов может давать различные результаты.

ЗАДАЧА 1.21. Вычислить нормы и числа обусловленности матрицы A .

На рис. 1.51 приведен фрагмент документа, в котором происходит вычисление норм матрицы A с помощью встроенных функций $\text{norm1}(A)$, $\text{norm2}(A)$, $\text{norme}(A)$, $\text{normi}(A)$ и по соответствующим формулам.

Значения чисел обусловленности отображены на рис. 1.52. Их вычисление было проведено при помощи функций Mathcad ($\text{cond1}(A)$, $\text{cond2}(A)$, $\text{conde}(A)$, $\text{condi}(A)$) и по формулам, отражающим зависимость числа обусловленности от соответствующей нормы матрицы.

$$A := \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix}$$

Первая норма

$$\text{norm1}(A) = 33$$

$$j := 0..3$$

$$B_j := \sum_{i=0}^3 |A_{i,j}|$$

$$\max(B) = 33$$

Вторая норма

$$\text{norm2}(A) = 30.289$$

$$\sqrt{\max(\text{eigenvals}(A \cdot A^T))} = 30.289$$

Бесконечная норма

$$\text{normi}(A) = 33$$

$$i := 0..3$$

$$B_i := \sum_{j=0}^3 |A_{i,j}|$$

$$\max(B) = 33$$

Евклидова норма

$$\text{norme}(A) = 30.545$$

$$\sqrt{\sum_{i=0}^3 \sum_{j=0}^3 (|A_{i,j}|)^2} = 30.545$$

Рис. 1.51 ▼ Вычисление норм матрицы

$$A := \begin{pmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{pmatrix}$$

Число обусловленности для $\|A\|_1$

$$\text{cond1}(A) = 4488$$

$$\text{norm1}(A) \cdot \text{norm1}(A^{-1}) = 4488$$

Число обусловленности для $\|A\|_2$

$$\text{cond2}(A) = 2984.093$$

$$\text{norm2}(A) \cdot \text{norm2}(A^{-1}) = 2984.093$$

Число обусловленности для $\|A\|_e$

$$\text{conde}(A) = 3009.579$$

$$\text{norme}(A) \cdot \text{norme}(A^{-1}) = 3009.579$$

Число обусловленности для $\|A\|_i$

$$\text{condi}(A) = 4488$$

$$\text{normi}(A) \cdot \text{normi}(A^{-1}) = 4488$$

Рис. 1.52 ▼ Вычисление чисел обусловленности матрицы

ЗАДАЧА 1.22. Определить обусловленность матриц A и B .

Вычисления, приведенные на рис. 1.53, показывают, что матрица A обусловлена хуже, чем матрица B , так как числа обусловленности матрицы A больше чисел обусловленности матрицы B .

$$A := \begin{pmatrix} 20 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.05 \end{pmatrix}$$

$$B := \begin{pmatrix} 20 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}$$

Числа обусловленности A

$$\text{cond1}(A) = 400$$

$$\text{cond2}(A) = 400$$

$$\text{conde}(A) = 401.002$$

$$\text{condi}(A) = 400$$

Числа обусловленности B

$$\text{cond1}(B) = 100$$

$$\text{cond2}(B) = 100$$

$$\text{conde}(B) = 128.083$$

$$\text{condi}(B) = 100$$

Рис. 1.53 ▼ Определение обусловленности матриц

1.4.3. Реализация численных методов решения систем линейных уравнений

Воспользуемся средствами программирования Mathcad для реализации некоторых численных алгоритмов, описанных в этой главе

ЗАДАЧА 1.23. Составить программу решения системы линейных уравнений методом ортогонализации. Найти решения системы

$$\begin{aligned} 2x_1 - x_2 + x_3 &= 0, \\ 3x_1 + 2x_2 - 5x_3 &= 1, \\ x_1 + 3x_2 - 2x_3 &= 4. \end{aligned}$$

Алгоритм (рис. 1.10) решения линейной системы методом ортогонализации подробно описан в разделе 1.2.7. На рис. 1.54 приведена Mathcad-программа, соответствующая этому алгоритму. Результатом работы программы является ортонормированная система $Q \cdot x = d$, получаемая из исходной системы $A \cdot x = b$. Результат выводится в виде расширенной матрицы $(Q|d)$.

Рис. 1.55 содержит пример применения Mathcad-программы для решения заданной системы. Полученную в результате работы программы матрицу S разбивают на матрицу Q и вектор y , а затем получают решение системы как вектор $x = Q^T \cdot y$.

```

ORT(A,b,n) := for j ∈ 1..n
               S ← √(∑k=1n (A1,k)2)
               q1,j ← A1,j / S
               d1 ← b1 / S
               q1,j ← 0
               for i ∈ 2..n
                 for j ∈ 1..n
                   q1,j ← Ai,j - ∑m=1i-1 [ qi-m,j · ∑k=1n (Ai,k · qi-m,k) ]
                   di ← bi - ∑m=1i-1 [ di-m · ∑k=1n (Ai,k · qi-m,k) ]
                   S ← √(∑k=1n (qi,k)2)
                   for j ∈ 1..n
                     qi,j ← qi,j / S
                     di ← di / S
                   qi,n+1 ← di
               q

```

Рис. 1.54 ▼ Mathcad-программа решения СЛАУ методом ортогонализации

```

ORIGIN := 1
A := ( 2 -1 1 )   b := ( 0 )
     ( 3  2 -5 )       ( 1 )
     ( 1  3 -2 )
C := ORT(A,b,3)    C = ( 0.816 -0.408  0.408  0 )
                       ( 0.542  0.298 -0.786  0.163 )
                       ( 0.199  0.863  0.465  1.889 )

Q := submatrix(C,1,3,1,3) y := submatrix(C,1,3,4,4) x := QT · y
Q = ( 0.816 -0.408  0.408 )   y = ( 0 )   x = ( 0.464 )
     ( 0.542  0.298 -0.786 )       ( 0.163 )       ( 1.679 )
     ( 0.199  0.863  0.465 )       ( 1.889 )       ( 0.75 )

```

Рис. 1.55 ▼ Решение задачи 1.23



ЗАДАЧА 1.24. Составить программу решения системы линейных уравнений методом QR-разложения. Найти решения системы из задачи 1.23.

Идея метода решения системы с помощью QR-разложения описана в разделе 1.2.6. Реализация этой идеи в Mathcad-программе продемонстрирована на рис. 1.56.

$$\begin{array}{l}
 \text{QR}(A, b, n) := \left\{ \begin{array}{l}
 c \leftarrow \text{ORT}(A^T, b, n) \\
 q \leftarrow \text{submatrix}(c^T, 1, 3, 1, 3) \\
 r \leftarrow q^T \cdot A \\
 y \leftarrow q^T \cdot b \\
 x_n \leftarrow \frac{y_n}{r_{n,n}} \\
 \text{for } i \in n-1..1 \\
 \quad y_i - \sum_{k=1}^n (r_{i,k} \cdot x_k) \\
 \quad x_i \leftarrow \frac{\quad}{r_{i,i}} \\
 x
 \end{array} \right.
 \end{array}$$

$\text{ORIGIN} := 1$
 $A := \begin{pmatrix} 2 & -1 & 1 \\ 3 & 2 & -5 \\ 1 & 3 & -2 \end{pmatrix} \quad b := \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$
 $\text{QR}(A, b, 3) = \begin{pmatrix} 0.464 \\ 1.679 \\ 0.75 \end{pmatrix}$

Рис. 1.56 ▾ Mathcad-программа решения СЛАУ методом QR-разложения

ЗАДАЧА 1.25. Найти решения системы методом прогонки

$$\begin{cases}
 2 \cdot x_1 + x_2 = -10, \\
 2 \cdot x_1 + 9 \cdot x_2 + 2 \cdot x_3 = -26, \\
 4 \cdot x_2 + 17 \cdot x_3 - 4 \cdot x_4 = -16, \\
 4 \cdot x_3 + 15 \cdot x_4 - 8 \cdot x_5 = -2, \\
 2 \cdot x_4 + 3 \cdot x_5 = 16.
 \end{cases}$$

Решение задачи представлено на рис. 1.57. Mathcad-программа соответствует алгоритму, описанному в разделе 1.2.8.

ЗАДАЧА 1.26. Решить систему методом Зейделя

$$\begin{cases}
 4x_1 - 0.24x_2 + 0.08x_3 = 8, \\
 0.9x_1 + 3x_2 + 0.15x_3 = 9, \\
 0.4x_1 + 0.8x_2 + 4x_3 = 20.
 \end{cases}$$

Численное решение линейной системы методом Зейделя описано в разделе 1.2.10. Программа в Mathcad представлена на рис. 1.58, а ее применение – на рис. 1.59.

ORIGIN := 1

PROG(b,c,d,r,n) :=

$$\begin{aligned} & \delta_1 \leftarrow \frac{-d_1}{c_1} \\ & \lambda_1 \leftarrow \frac{r_1}{c_1} \\ & \text{for } i \in 2..n \\ & \quad \left| \begin{aligned} & \delta_i \leftarrow \frac{-d_i}{c_i + b_i \cdot \delta_{i-1}} \\ & \lambda_i \leftarrow \frac{r_i - b_i \cdot \lambda_{i-1}}{c_i + b_i \cdot \delta_{i-1}} \end{aligned} \right. \\ & x_n \leftarrow \lambda_n \\ & \text{for } i \in n-1..1 \\ & \quad x_i \leftarrow \delta_i \cdot x_{i+1} + \lambda_i \\ & x \end{aligned}$$

$$b := \begin{pmatrix} 0 \\ 2 \\ 4 \\ 4 \\ 2 \end{pmatrix} \quad c := \begin{pmatrix} 2 \\ 9 \\ 17 \\ 15 \\ 3 \end{pmatrix} \quad d := \begin{pmatrix} 1 \\ 2 \\ -4 \\ -8 \\ 0 \end{pmatrix} \quad r := \begin{pmatrix} -10 \\ -26 \\ -16 \\ -2 \\ 16 \end{pmatrix} \quad \text{PROG}(b, c, d, r, 5) = \begin{pmatrix} -4 \\ -2 \\ 0 \\ 2 \\ 4 \end{pmatrix}$$

Рис. 1.57 ▼ Mathcad-программа решения СЛАУ методом прогонки

Zeydel(A,b,ε) :=

$$\begin{aligned} & n \leftarrow \text{cols}(A) \\ & F \leftarrow A^T \cdot A \\ & H \leftarrow A^T \cdot b \\ & \text{for } i \in 0..n-1 \\ & \quad \left| \begin{aligned} & \text{for } j \in 0..n-1 \\ & \quad \left| \begin{aligned} & A_{i,j} \leftarrow 0 \text{ if } i=j \\ & A_{i,j} \leftarrow \frac{-F_{i,j}}{F_{i,i}} \text{ otherwise} \end{aligned} \right. \\ & b_i \leftarrow \frac{H_i}{F_{i,i}} \end{aligned} \right. \\ & k \leftarrow 0 \\ & x1 \leftarrow b \\ & \text{while } 1 \\ & \quad \left| \begin{aligned} & x \leftarrow x1 \\ & \text{for } i \in 0..n-1 \\ & \quad \left| \begin{aligned} & S \leftarrow 0 \\ & \text{for } j \in 0..n-1 \\ & \quad S \leftarrow S + A_{i,j} \cdot x_j \\ & x_i \leftarrow S + b_i \end{aligned} \right. \end{aligned} \right. \\ & y \leftarrow \|x - x1\| \\ & x1 \leftarrow x \\ & m \leftarrow \max(y) \\ & k \leftarrow k + 1 \\ & \text{break if } (m < \varepsilon) \vee (k > 200) \\ & (x \quad k \quad m) \end{aligned}$$

Рис. 1.58 ▼ Mathcad-программа решения СЛАУ методом Зейделя

Обращение к подпрограмме решения системы методом Зейделя

$(x \quad k \quad \varepsilon) := \text{Zeydel}(A, b, .000001)$

$$x = \begin{pmatrix} 1.778 \\ 2.248 \\ 4.373 \end{pmatrix} \quad k = 9 \quad \varepsilon = 5.08 \times 10^{-8}$$

Решение этой же системы с помощью встроенной функции MathCAD

$$\text{lsolve}(A, b) = \begin{pmatrix} 1.778 \\ 2.248 \\ 4.373 \end{pmatrix}$$

Рис. 1.59 ▼ Решение задачи 1.26

ЗАДАЧА 1.27. Вычислить значение наибольшего по модулю собственного числа и соответствующего ему собственного вектора для матрицы A . Решить задачу методом скалярных произведений.

Описание метода скалярных произведений для вычисления наибольшего по модулю собственного числа и соответствующего ему собственного вектора можно найти в разделе 1.3.1. Решение задачи этим методом приведено на рис. 1.60

ORIGIN := 1

```

eg(A,n,ε) := for i ∈ 1..n
               Yi ← 1
               λ ← 0
               s ← Y·Y
               X ←  $\frac{Y}{\sqrt{s}}$ 
               while 1
                   λ0 ← λ
                   Y ← A·X
                   s ← Y·Y
                   t ← Y·X
                   X ←  $\frac{Y}{\sqrt{s}}$ 
                   λ ←  $\frac{s}{t}$ 
                   break if |λ - λ0| < ε
               Xn+1 ← λ
               X

```

$A := \begin{pmatrix} 3 & -2 \\ -4 & 1 \end{pmatrix}$
 $eg(A, 2, 0.01) = \begin{pmatrix} 0.707 \\ -0.707 \\ 5 \end{pmatrix}$
 $eigenvals(A) = \begin{pmatrix} 5 \\ -1 \end{pmatrix}$
 $eigenvec(A, 5) = \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix}$

Рис. 1.60 ▼ Решение задачи 1.27

1.5. Решение задач линейной алгебры с использованием Maple

Численные методы для решения задач линейной алгебры в Maple реализованы в виде пакета `linalg`, который является достаточно мощным инструментом этой области и содержит более ста функций. Вызов этого пакета осуществляется командой `with(linalg)` (см. Приложение 3).

Опишем наиболее распространенные функции линейной алгебры, разделив их, так же как это было в Mathcad, на три категории.

Функции определения матриц и операции с блоками матриц:

- `matrix(n, m, [<список элементов>])` – создает матрицу размером $n \times m$, элементы которой указаны в списке (рис. 1.61);

$$\begin{aligned} & \left[\begin{array}{l} > A:=\text{matrix}(2,2,[1,2,3,4]); \\ > B:=\text{matrix}(3,2,[5,6,7,8,9,10]); \end{array} \right. \end{aligned}$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{bmatrix}$$

Рис. 1.61 ▼ Пример создания матриц

- `augment(A, B)` или `concat(A, B)` – объединяет две или более матриц горизонтально (рис. 1.62);
- `delcols(A, i..j)` – удаляет столбцы с i по j в матрице A ;

$$\begin{aligned} & \left[\begin{array}{l} > A:=\text{matrix}(2,2,[1,2,3,4]); \\ > B:=\text{matrix}(2,2,[5,6,7,8]); \end{array} \right. \end{aligned}$$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$\left[\begin{array}{l} > AB:=\text{augment}(A,B); \\ > \text{delcols}(AB,1..2); \# \text{удаление первого столбца из } A \\ > \text{delrows}(AB,2..2); \# \text{удаление второй строки из } B \\ > \text{submatrix}(AB,1..2,2..3); \end{array} \right.$$

$$AB = \begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$[1 \ 2 \ 5 \ 6]$$

$$\begin{bmatrix} 2 & 5 \\ 4 & 7 \end{bmatrix}$$

Рис. 1.62 ▼ Пример работы с блоками матрицы

- `delrows(A, i..j)` – удаляет строки с i по j в матрице A ;
- `submatrix(A, i1..j1, i2..j2)` – из матрицы A выделить подматрицу с $i1$ по $j1$ строку и с $i2$ по $j2$ столбец;
- `entermatrix(A)` – интерактивный ввод матрицы A (рис. 1.63);

```

> C:=array(1..2,1..2):
>
Warning, premature end of input

[enter element 1,1 > 1/2;
[enter element 1,2 > 2/3;
[enter element 2,1 > 3/4;
[enter element 2,2 > 4/5;


$$\begin{bmatrix} \frac{1}{2} & \frac{2}{3} \\ \frac{3}{4} & \frac{4}{5} \end{bmatrix}$$


> C[1,2]+A[2,1]; # обращение к элементам матрицы
>

$$\frac{11}{3}$$


> evalm((A+B)*C-3*B^2); #элементарные матричные операции

$$\begin{bmatrix} -192 & -\frac{1118}{5} \\ -259 & \frac{-4526}{15} \end{bmatrix}$$


```

Рис. 1.63 ▽ Ввод матрицы и элементарные операции над матрицами

```

> mulcol(C,1,4); # умножить первый столбец C на 4

$$\begin{bmatrix} 2 & \frac{2}{3} \\ 3 & \frac{4}{5} \end{bmatrix}$$


> mulrow(C,2,5); #умножить вторую строку C на 5

$$\begin{bmatrix} \frac{1}{2} & \frac{2}{3} \\ \frac{15}{4} & 4 \end{bmatrix}$$


> transpose(C); # транспонировать C

$$\begin{bmatrix} \frac{1}{2} & \frac{3}{4} \\ \frac{2}{3} & \frac{4}{5} \end{bmatrix}$$


```

Рис. 1.64 ▽ Пример операций над матрицами

- `mulcol(A, j, k)` – умножение в матрице A столбца с номером j на k ;
- `mulrow(A, i, k)` – умножение в матрице A строки с номером i на k ;
- `transpose(A)` – транспонирование A (рис. 1.64);
- `vector(n, [<список элементов>])` – создает вектор размерностью n , элементы которого указаны в списке (рис. 1.65);
- `scalarmul(V, k)` – поэлементное умножение вектора V на k ;
- `crossprod(X, Z)` – вычисляет векторное произведение векторов X, Z ;

```

> V:=vector(3,[a,b,c]);#создание вектора
                                     V=[a,b,c]
> W:=scalarmul(V,3);#поэлементное умножение вектора на 3
                                     W=[3a,3b,3c]
> Q:=vector(3,[2,1,-1]):crossprod(V,Q);
                                     [-b-c,2c+a,a-2b]

```

Рис. 1.65 ▼ Создание вектора и пример работы с ним

- `diag(n)` – создание диагональной матрицы (рис. 1.66).

```

> diag(a,b,c);#диагональная матрица
                                     [ a  0  0 ]
                                     [ 0  b  0 ]
                                     [ 0  0  c ]
> diag(1,1);#единичная матрица
                                     [ 1  0 ]
                                     [ 0  1 ]
> diag(0,0);#нулевая матрица
                                     [ 0  0 ]
                                     [ 0  0 ]

```

Рис. 1.66 ▼ Создание некоторых специальных матриц

Функции вычисления различных числовых характеристик матриц:

- `coldim(A)` – определяет число столбцов матрицы A ;
- `rowdim(A)` – определяет число строк матрицы A ;
- `row(A, k)` – выделяет k -ю строку матрицы A ;
- `col(A, k)` – выделяет k -й столбец матрицы A ;
- `vectdim(X)` – определяет длину вектора X ;
- `dotprod(X, Z)` – вычисляет скалярное произведение векторов X, Z (рис. 1.67);
- `det(A)` – вычисляет определитель матрицы A (рис. 1.68);
- `minor(A, i, j)` – возвращает минор матрицы A для элемента с индексами i, j ;

- rank(A) – определяет ранг матрицы A;
- trace(A) – вычисляет след матрицы A;
- norm(A) – норма матрицы или вектора A;
- cond(A) – вычисляет число обусловленности матрицы A.

```

> A:=matrix(2,3,[1,2,3,4,5,6]);
                                     A =  $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ 
> coldim(A); #число столбцов матрицы A
                                     3
> rowdim(A); #число строк матрицы A
                                     2
> X:=col(A,3); #третий столбец матрицы A
                                     X =  $[3, 6]$ 
> Y:=row(A,1); #первая строка матрицы A
                                     Y =  $[1, 2, 3]$ 
> vectdim(Y); #длина вектора Y
                                     3
> Z:=row(A,2); #второй столбец матрицы A
                                     Z =  $[4, 5, 6]$ 
> dotprod(Y,Z); #скалярное произведение Y и Z
                                     32

```

Рис. 1.67 ▽ Вычисление скалярного произведения векторов X, Z

```

> M:=matrix(3,3,[1,2,-1,1,-1,3,-2,1,-1]);
                                     M =  $\begin{bmatrix} 1 & 2 & -1 \\ 1 & -1 & 3 \\ -2 & 1 & -1 \end{bmatrix}$ 
> det(M); #определитель
                                     -11
> minor(M,2,3); #минор для элемента с номером 2,3
                                      $\begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$ 
> rank(M); #ранг матрицы
                                     3
> trace(M); #след матрицы
                                     -1
> norm(M); #норма матрицы
                                     5
> cond(M); #число обусловленности матрицы
                                      $\frac{60}{11}$ 

```

Рис. 1.68 ▽ Вычисление определителя матрицы A

Функции, реализующие численные алгоритмы решения задач линейной алгебры:

- `inverse(A)` – формирует матрицу, обратную к A ;
- `linsolve(A,b)` или `linsolve(A,B)` – находит решение линейной системы уравнений $A \cdot x = b$ или решает матричное уравнение $A \cdot X = B$;
- `rref(A)` – приведение матрицы A к ступенчатому виду путем элементарных операций со строками;
- `cholesky(A)` – выполняет разложение по Халецкому для квадратной положительно определенной матрицы A ;
- `LUdecomp(A)` – выполняет LU-разложение матрицы A ;
- `QRdecomp(A)` – выполняет QR-разложение матрицы A ;
- `eigenvals(A)` – находит собственные значения квадратной матрицы A ;
- `eigenvects(A)` – находит собственные векторы квадратной матрицы A ;

Приведем несколько конкретных задач линейной алгебры.

ЗАДАЧА 1.28. Решить матричное уравнение $A \cdot X = B$.

Методика решения матричных уравнений и ее реализация в Mathcad описана в задаче 1.7. На рис.1.69 показано, как Maple справляется с этой задачей.

```

> A:=matrix(2,2,[3,2,4,3]);
> B:=matrix(2,2,[-1,7,3,5]);
> evalm(inverse(A)&*B); #первый способ
> linsolve(A,B); #второй способ

```

$$A := \begin{bmatrix} 3 & 2 \\ 4 & 3 \end{bmatrix}$$

$$B := \begin{bmatrix} -1 & 7 \\ 3 & 5 \end{bmatrix}$$

$$\begin{bmatrix} -9 & 11 \\ 13 & -13 \end{bmatrix}$$

$$\begin{bmatrix} -9 & 11 \\ 13 & -13 \end{bmatrix}$$

Рис. 1.69 ▼ Решение матричных уравнений

ЗАДАЧА 1.29. Решить систему линейных уравнений различными методами:

$$\begin{aligned} 2x_1 - x_2 + x_3 &= 0, \\ 3x_1 + 2x_2 - 5x_3 &= 1, \\ x_1 + 3x_2 - 2x_3 &= 4. \end{aligned}$$

На рис. 1.70 показано, как решить линейную систему при помощи специально для этого предназначенной функции `linsolve`, а также методом обратной матрицы и Гаусса (описание см. в разделе 1.2.1 и задачах 1.9, 1.10).

```

> A:=matrix(3,3,[2,-1,1,3,2,-5,1,3,-2]):b:=vector(3,[0,1,4]);
                                     b:=[0,1,4]
> linsolve(A,b); #решение линейной системы Ax=b
                                     [ 13 47 3 ]
                                     [ 28' 28' 4 ]
> evalm(inverse(A)&*b); #решение Ax=b методом обратной матрицы
                                     [ 13 47 3 ]
                                     [ 28' 28' 4 ]
> col((rref(augment(A,b))),4); #решение Ax=b методом Гаусса
                                     [ 13 47 3 ]
                                     [ 28' 28' 4 ]

```

Рис. 1.70 ▼ Решение СЛАУ

Методы решения заданной системы с помощью LU- и QR-разложений (описание см. в разделе 1.2.6 и задачах 1.11, 1.12) показаны на рис. 1.71.

```

> #Решение Ax=b с использованием LU разложения
> U:=LUdecomp(A,P='p',L='l'):L:=evalm(l):C:=evalm(p):
> y:=col((rref(augment(L,b))),4):x:=col((rref(augment(U,y))),4);
                                     x:=[ 13 47 3 ]
                                     [ 28' 28' 4 ]
> #Решение Ax=b с использованием QR разложения
> R:=QRdecomp(A,Q='q'):Q:=evalm(q):
> y:=evalm(transpose(Q)&*b):x:=col((rref(augment(R,y))),4);
Warning, unable to find a provably non-zero pivot
                                     x:=[ 13 47 3 ]
                                     [ 28' 28' 4 ]

```

Рис. 1.71 ▼ Решение СЛАУ с помощью методов разложения

Листинг ниже содержит текст Maple-процедуры, реализующей алгоритм метода Гаусса. На рис. 1.72 показаны результаты работы программы.

Листинг 1.1

```

> GAUSS:=proc(A,b,n)
> local k,i,j,max,c,M,S,r;global x;
> #Прямой ход
> for k from 1 to n do
> max:=abs(A[k,k]);
> r:=k;
> for i from k+1 to n do
> if abs(A[i,k])>max then
> max:=abs(A[i,k]);
> r:=i;
> fi;

```

```

> od;
> for j from 1 to n do
> c:=A[k,j];
> A[k,j]:=A[r,j];
> A[r,j]:=c;
> od;
> c:=b[k];
> b[k]:=b[r];
> b[r]:=c;
> for i from k+1 to n do
> M:=A[i,k]/A[k,k];
> for j from k to n do
> A[i,j]:=A[i,j]-M*A[k,j];
> od;
> b[i]:=b[i]-M*b[k];
> od;
> od;
> #Обратный ход
> if a[n,n]=0 then
> if b[n]=0 then
> ERROR("Бесконечное множество");
> else ERROR("Нет решений");
> fi;
> else
> for i from n by -1 to 1 do
> S:=0;
> for j from i+1 to n do
> S:=S+A[i,j]*x[j];
> od;
> x[i]:=(b[i]-S)/A[i,i];
> od;
> fi;
> evalm(x);
> end;

```

```

> A:=array(1..3,1..3,[[1,2,5],[1,-1,3],[3,-6,-1]]);

```

$$A = \begin{bmatrix} 1 & 2 & 5 \\ 1 & -1 & 3 \\ 3 & -6 & -1 \end{bmatrix}$$

```

> b:=array(1..3,[-9,2,25]);

```

$$b = [-9, 2, 25]$$

```

> X:=GAUSS(A,b,3);

```

$$X = [2, -3, -1]$$

```

> evalm(A&*X-b); #Проверка Ax-b=0

```

$$[0, 0, 0]$$

```

> evalm(1/A*b); #Решение Ax=b методом обратной матрицы

```

$$[2, -3, -1]$$

Рис. 1.72 ▼ Решение СЛАУ с помощью Maple-процедуры, реализующей метод



ЗАДАЧА 1.30. Найти собственные значения и собственные векторы матрицы A .

Проблема собственных значений была рассмотрена в разделе 1.3 и в задачах 1.17 и 1.27. Рис. 1.73 представляет решение этой задачи в Maple.

```
[> A:=matrix(2,2,[3,-2,-4,1]):
> eigenvals(A);#собственные числа
                                         5, -1
> #собственные числа и собственные векторы
> v:=eigenvectors(A):
> L1:=v[1][1];L2:=v[2][1];#собственные числа
                                         L1:=-1
                                         L2:=5
> v1:=v[1][3];v2:=v[2][3];#собственные векторы
                                         v1:=[1,2]
                                         v2:=[-1,1]
> #проверка
> evalm((A-L2*diag(1,1))*v2[1]);
                                         [0,0]
> evalm((A-L1*diag(1,1))*v1[1]);
                                         [0,0]
```

Рис. 1.73 ▼ Решение задачи 1.30

1.6. Возможности MATLAB для решения задач линейной алгебры

MATLAB обладает достаточно мощными возможностями в области решения задач линейной алгебры. В этом разделе мы опишем простейшие инструменты, которые предоставляет пакет.

Краткая информация об элементарных операциях над массивами имеется в приложении 2. Несколько примеров, содержащихся в листинге ниже, демонстрируют ввод векторов и матриц и способ обращения к их элементам.

Листинг 1.2

```
>> %Ввод вектора-строки
```

```
>> V=[1,2,3,4,5]
```

```
V =
```

```
    1    2    3    4    5
```

```
>> %Ввод вектора-столбца
```

```
>> W=[6;7;8;9;10]
```

```
W =
```

```

7
8
9
10
>> %Обращение к элементу массива по номеру
>> W(3)
ans =
8
>> %Ввод матрицы по строкам
>> M=[1 2 3;4 5 6]
M =
1     2     3
4     5     6
>> %Ввод матрицы по столбцам
>> A=[[1;2] [3;4] [5;6]]
A =
1     3     5
2     4     6
>> %Обращение к элементу матрицы
>> A(1,3)+M(2,2)*A(1,2)-M(1,1)^2
ans =
19

```

Следующий листинг содержит примеры элементарных операций.

Листинг 1.3

```

>> v1=[2 5 -1];
>> v2=[1 -1 3];
>> v1+v2 %Поэлементное сложение векторов
ans =
3     4     2
>>%-----
>> v1+2 %Сложение элементов вектора с числом
ans =
4     7     1
>>%-----
>> 2*v2-v1/4 %Поэлементные арифметические операции
ans =
1.5000   -3.2500    6.2500
>>%-----
>> v1.*v2 %Поэлементное умножение
ans =
2     -5     -3
>>%-----
>> v1.^2 %Поэлементное возведение в степень
ans =

```

```

4      25      1
>>%-----
>> v1.^v2 %Поэлементное возведение в степень
ans =
    2.0000    0.2000   -1.0000
>>%-----
>> v1./3 %Поэлементное деление на число
ans =
    0.6667    1.6667   -0.3333 >> v2./v1
>>%-----
>> v2./v1 %Поэлементное деление вектора на вектор ,
ans =
    0.5000   -0.2000   -3.0000

```

Элементарные действия над матрицами продемонстрированы в листинге ниже.

 *Знак обратного деления в MATLAB предназначен для решения линейных систем $A \cdot x = b$.*

Листинг 1.4

```

>> A=[1 -1 3;-1 2 0;3 -2 1];
>> B=2*A %Поэлементное умножение матрицы на число
B =
     2     -2     6
    -2      4     0
     6     -4     2
>>%-----
>> A/3+2*(B-A) %Поэлементные действия над матрицами
ans =
    2.3333   -2.3333    7.0000
   -2.3333    4.6667     0
    7.0000   -4.6667    2.3333
>>%-----
>> A' %Транспонирование матрицы
ans =
     1     -1      3
    -1      2     -2
     3      0      1
>>%-----
>> A*B %Матричное умножение
ans =
    22    -18    12
    -6     10    -6
    16    -18    20
>>%-----
>> B^2 %Возведение в степень, то же что B*B

```

```

ans =
    44    -36     24
   -12     20    -12
    32    -36     40
>>%
>> A.*B %Поэлементное умножение
ans =
     2     2    18
     2     8     0
    18     8     2
>>%
>> B.^3 %Поэлементное возведение в степень
ans =
     8    -8   216
    -8   64     0
   216  -64     8
>>%
>> A=[1 2 5; 1 -1 3; 3 -6 -1];
>> b=[-9;2;25];
>> x=A\b %Решение линейной системы Ax=b
x =
    2.0000
   -3.0000
   -1.0000
>> A*x %Проверка, действительно Ax=b
ans =
   -9.0000
    2.0000
   25.0000

```

Приведем перечень наиболее распространенных функций

► `length(V)` – определяет длину вектора V ;

Листинг 1.5

```

>> V=[-1 0 3 -2 1 -1 1];%Вектор-строка
>> length(V)%Длина вектора
ans =
     7
>> W=[0,3,1,1,2]; %Вектор-столбец
>> length(W) %Длина вектора
ans =
     5

```

► `prod(V)` или `prod(A,k)` – вычисляет произведение элем или произведения столбцов или строк матрицы в зависи ния k ;

Листинг 1.6

```
>> V=[1,2,3];
>> prod(V) %Произведение элементов вектора
ans =
    6
>> A=[1 2;3 4];
>> prod(A) %Произведения столбцов матрицы
ans =
    3    8
>> prod(A,1) %Произведения столбцов матрицы
ans =
    3    8
>> prod(A,2) %Произведения строк матрицы
ans =
    2
   12
```

- `sum(V)` или `sum(A,k)` – вычисляет сумму элементов массива V или сумму столбцов или строк матрицы в зависимости от значения k ;

Листинг 1.7

```
>> V=[-1 0 3 -2 1 -1 1];
>> sum(V) %Сумма элементов вектора
ans =
    1
>> C=[1 2 3;1 2 3]
C =
    1    2    3
    1    2    3
>> sum(C,1) %Сумма элементов матрицы по столбцам
ans =
    2    4    6
>> sum(C,2) %Сумма элементов матрицы по строкам
ans =
    6
    6
```

- `dot(v1,v2)` – вычисляет скалярное произведение векторов $v1$ и $v2$, то же значение выдаст функция `sum(v1.*v2)`;

Листинг 1.8

```
>> v1=[1.2;0.3;-1.1];
>> v2=[-0.9;2.1;0.5];
>> dot(v1,v2) %Скалярное произведение
ans =
   -1
>> sum(v1.*v2) %Скалярное произведение
ans =
   -1
```

► `cross(v1,v2)` – определяет векторное произведение векторов v_1 и v_2 ;

Листинг 1.9

```
>> v1=[1.2;0.3;-1.1];
>> v2=[-0.9;2.1;0.5];
>> cross(v1,v2) %Векторное произведение
ans =
    2.4600
    0.3900
    2.7900
```

► `min(V)` – находит минимальный элемент массива V , вызов в формате `[k, n]=min(V)` дает возможность определить минимальный элемент k и его номер в массиве n ;

► `max(V)` – находит максимальный элемент массива V или при `[k, n]=max(V)` определяет максимум и его номер;

Листинг 1.10

```
>> V=[-1 0 3 -2 1 -1 1];
>> min(V) %Минимальный элемент
ans =
    -2
>> max(V) %Максимальный элемент
ans =
     3
>> [k,n]=min(V) %Минимальный элемент и его номер
k =
    -2
n =
     4
>> [k,n]=max(V) % Максимальный элемент и его номер
k =
     3
n =
     3
```

► `sort(V)` – выполняет упорядочивание массива V ;

Листинг 1.11

```
>> V=[-1 0 3 -2 1 -1 1];
>> sort(V) %Сортировка по возрастанию
ans =
    -2    -1    -1     0     1     1     3
>> -sort(-V) %Сортировка по убыванию
ans =
     3     1     1     0    -1    -1    -2
```

► `det(M)` – вычисляет определитель квадратной матрицы M ;

► `rank(M)` – определяет ранг матрицы M ;

Листинг 1.12

```
>> M=[3 2;4 3];
>> det(M)%Определитель матрицы
ans =
    1
>> M=[1 -2 4 5;3 -1 -3 5;1 3 -11 -5]
M =
     1     -2      4      5
     3     -1     -3      5
     1      3    -11     -5
>> rank(M)%Ранг матрицы
ans =
     2
```

- `norm(M, p)` – возвращает различные виды норм матрицы M в зависимости от p ($p=1, 2, \text{inf}, \text{fro}$);
- `cond(M, p)` – возвращает число обусловленности матрицы M , основанное на норме p ;

Листинг 1.13

```
>> M=[5 7 6 5; 7 10 8 7;6 8 10 9;5 7 9 10];
>> norm(M)%Норма матрицы
ans =
    30.2887
>> cond(M)%Число обусловленности матрицы
ans =
    2.9841e+003
>> norm(M,2)%Вторая норма матрицы, то же что и norm(M)
ans =
    30.2887
>> cond(M,2)%Число обусловленности для второй нормы, аналогично cond(M)
ans =
    2.9841e+003
```

- `eye(n,m)` или `eye(n)` – возвращает прямоугольную матрицу с единицами по главной диагонали или квадратную единичную матрицу;

Листинг 1.14

```
>> eye(3,3)%единичная матрица
ans =
     1     0     0
     0     1     0
     0     0     1
```

- `ones(n,m)` или `ones(n)` – формирует прямоугольную или квадратную матрицу, состоящую из единиц;

Листинг 1.15

```
>> ones(2,1)%Единичная матрица-столбец
ans =
     1
     1
```

- `zeros(n,m)` или `zeros(n)` – возвращает прямоугольную или квадратную левую матрицу;

Листинг 1.16

```
>> zeros(1,3)%Нулевая матрица-строка
ans =
     0     0     0
```

- `diag(V,n)` или `diag(V)` – возвращает квадратную матрицу с элементами на k -й диагонали или элементами V на главной диагонали;

Листинг 1.17

```
>> V=[1,2,3]
V =
     1     2     3
>> diag(V)%Диагональная матрица, V на главной диагонали
ans =
     1     0     0
     0     2     0
     0     0     3
>> diag(V,1) %Диагональная матрица, V на первой диагонали
ans =
     0     1     0     0
     0     0     2     0
     0     0     0     3
     0     0     0     0
```

- `cat(n, A, B)` или `cat(n, A, B, C, ...)` – объединяет матрицы A и B или входящие матрицы;

Листинг 1.18

```
>> A=[1 2;3 4];
>> B=[5 6 ;7 8];
>> cat(2,A,B)%Объединение матриц
ans =
     1     2     5     6
     3     4     7     8
>> [A,B] %Объединение матриц, аналогично cat(2,A,B)
ans =
     1     2     5     6
     3     4     7     8
>> cat(1,A,B) %Объединение матриц
```

```
ans =
     1     2
     3     4
     5     6
     7     8
```

```
>> [A;B] %Объединение матриц, аналогично cat(1,A,B)
```

```
ans =
     1     2
     3     4
     5     6
     7     8
```

► `inv(M)` – возвращает матрицу, обратную к M ;

Листинг 1.19

```
>> M=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6]
```

```
M =
     2     1    -5     1
     1    -3     0    -6
     0     2    -1     2
     1     4    -7     6
```

```
>> P=inv(M)%Матрица обратная к M
```

```
P =
     1.3333    -0.6667     0.3333    -1.0000
    -0.0741     0.2593     1.1481    -0.1111
     0.3704    -0.2963     0.2593    -0.4444
     0.2593    -0.4074    -0.5185    -0.1111
```

```
>> M*P %Проверка M*P=E
```

```
ans =
     1.0000    -0.0000    -0.0000     0.0000
         0     1.0000     0.0000     0.0000
     0.0000    -0.0000     1.0000    -0.0000
     0.0000    -0.0000    -0.0000     1.0000
```

► `eig(M)` – возвращает вектор собственных значений матрицы M , вызов функции в формате `[V, D]=eig(M)` даст матрицу V , столбцы которой – собственные векторы матрицы M , и диагональную матрицу D , содержащую собственные значения матрицы M .

Листинг 1.20

```
>> M=[3 -2;-4 1]
```

```
>> eig(M)%Собственные значения матрицы M
```

```
ans =
     5
    -1
```

```
>> [V,D]=eig(M)
```

```
V = %Первый столбец – собственный вектор для 5, второй для -1
```

```
     0.7071     0.4472
    -0.7071     0.8944
```

```
D = %Собственные значения матрицы M
```

```
5      0
```

```
0      -1
```

```
>> %Проверка M*V=V*D
```

```
>> V*D
```

```
ans =
```

```
3.5355    -0.4472
```

```
-3.5355    -0.8944
```

```
>> M*V
```

```
ans =
```

```
3.5355    -0.4472
```

```
-3.5355    -0.8944
```

- `linsolve(A, b)` – возвращает решение системы линейных уравнений $A \cdot x = b$, вызов в формате `linsolve(A, b, options)` позволяет задать метод решения уравнения¹. Если задать функцию в виде `[x, r] = linsolve(A, b)`, то она вернет `x` – решение системы и `r` – ранг матрицы `A`;

Листинг 1.21

```
>> A=[1 2 3;-2 -4 -6]
```

```
A =
```

```
1      2      3
```

```
-2     -4     -6
```

```
>> b=[5;6]
```

```
b =
```

```
5
```

```
6
```

```
>> x=linsolve(A,b)%Решение системы Ax=b
```

```
>>%Предупреждение: ранг меньше размерности!
```

```
Warning: Rank deficient, rank = 1,  tol = 4.4686e-015.
```

```
x =
```

```
0
```

```
0
```

```
-0.4667
```

```
>> A*x %Найденное решение не верно, т.к. Ax не равно b
```

```
ans =
```

```
-1.4000
```

```
2.8000
```

```
>> [x,r]=linsolve(A,b)
```

```
x = %Решение системы
```

```
0
```

```
0
```

```
-0.4667
```

¹ В случае когда для решения линейной системы используется знак обратного слэша $x = A \backslash b$, выбор метода остается за MATLAB.

```

r = %Ранг матрицы A
1
>>%
>> A=[2 -1 1;3 2 -5;1 3 -2];
>> b=[0;1;4];
>> x=linsolve(A,b)%Решение линейной системы
x =
    0.4643
    1.6786
    0.7500
>> A*x %Решение верно
ans =
    0.0000
    1.0000
    4.0000

```

- **rref(M)** – осуществляет приведение матрицы M к треугольной форме, используя метод исключения Гаусса;

Листинг 1.22

```

>>%Решение системы методом Гаусса
>> A=[2 -1 1;3 2 -5;1 3 -2];
>> b=[0;1;4];
>> C=rref([A b])%Приведение расширенной матрицы к треугольному виду
C =
    1.0000         0         0    0.4643
         0    1.0000         0    1.6786
         0         0    1.0000    0.7500
>> x=C(1:3,4:4)%Выделение последнего столбца из матрицы
x = %Решение системы
    0.4643
    1.6786
    0.7500
>> A*x %Проверка
ans =
     0
     1
     4

```

- **chol(M)** – возвращает разложение по Халецкому для положительно определенной симметрической матрицы M ;

Листинг 1.23

```

>> A=[10 1 1;2 10 1;2 2 10];
>> chol(A)%Разложение по Халецкому
ans =
    3.1623    0.3162    0.3162
         0    3.1464    0.2860
         0         0    3.1334

```

```
>> A=[1 2;1 1];%Матрица не симметрическая
>> chol(A)
??? Error using ==> chol
Matrix must be positive definite.
>>%Матрица содержит отрицательные элементы
>> Z=[3,1,-1,2;-5,1,3,-4;2,0,1,-1;1,-5,3,-3];
>> chol(Z)
??? Error using ==> chol
Matrix must be positive definite.
```

► `lu(M)` – выполняет LU-разложение, возвращает две матрицы: нижнюю треугольную `L` и верхнюю треугольную `U`;

Листинг 1.24

```
>>%Решение линейной системы с помощью LU-разложения
>> A=[3,1,-1,2;-5,1,3,-4;2,0,1,-1;1,-5,3,-3];
>> b=[6;-12;1;3];
>> [L,U,P]=lu(A)%LU-разложение
L = %Нижняя треугольная матрица
    1.0000         0         0         0
   -0.2000    1.0000         0         0
   -0.4000   -0.0833    1.0000         0
   -0.6000   -0.3333    0.8000    1.0000
U = %Верхняя треугольная матрица
   -5.0000    1.0000    3.0000   -4.0000
         0   -4.8000    3.6000   -3.8000
         0         0    2.5000   -2.9167
         0         0         0    0.6667
P = %Матрица перестановок
     0     1     0     0
     0     0     0     1
     0     0     1     0
     1     0     0     0
>> b=P*b %Модификация вектора b
b =
   -12
     3
     1
     6
>> Y=rref([L b])%Решение уравнения Ly=b
Y =
    1.0000         0         0         0   -12.0000
         0    1.0000         0         0    0.6000
         0         0    1.0000         0   -3.7500
         0         0         0    1.0000    2.0000
>> y=Y(1:4,5:5)
```

```

y =
-12.0000
  0.6000
-3.7500
  2.0000
>> X=rref([U y])%Решение уравнения Ux=y
X =
  1.0000         0         0         0  1.0000
         0  1.0000         0         0 -1.0000
         0         0  1.0000         0  2.0000
         0         0         0  1.0000  3.0000
>> x=X(1:4,5:5)%Решение заданной системы Ax=b
x =
  1.0000
-1.0000
  2.0000
  3.0000
>> A*x %Проверка
ans =
  6.0000
-12.0000
  1.0000
  3.0000

```

► **qr (M)** – выполняет QR-разложение, возвращает ортогональную и верхнюю треугольную R ;

Листинг 1.25

```

>>%Решение линейной системы с помощью QR-разложения
>> A=[3,1,-1,2;-5,1,3,-4;2,0,1,-1;1,-5,3,-3];
>> b=[6;-12;1;3];
>> [Q,R]=qr(A)% QR-разложение
Q = %Ортогональная матрица
-0.4804 -0.3032 -0.3585  0.7408
  0.8006 -0.0202 -0.5455  0.2469
-0.3203 -0.0708 -0.7357 -0.5926
-0.1601  0.9501 -0.1808  0.1975
R = %Верхняя треугольная
-6.2450  1.1209  2.0817 -3.3627
         0 -5.0738  3.0221 -3.3051
         0         0 -2.5561  2.7432
         0         0         0  0.4939
>> y=Q'*b
Y = %Результат умножения транспонированной матрицы Q на b
-13.2906
  1.2028
  3.1172
  1.4816

```

```
>> X=rref([R y])
X = %Решение уравнения Rx=y
    1.0000         0         0         0     1.0000
         0     1.0000         0         0    -1.0000
         0         0     1.0000         0     2.0000
         0         0         0     1.0000     3.0000

>> x=X(1:4,5:5)
x = % Решение заданной системы Ax=b
    1.0000
   -1.0000
    2.0000
    3.0000
```

Далее приведем тексты программ, составленных с помощью встроенного языка программирования пакета MATLAB и реализующих численные методы решения задач линейной алгебры.

Листинг ниже содержит М-функцию, которая позволяет решить систему линейных уравнений методом квадратных корней (см. раздел 1.2.4).

Листинг 1.26

%Решение СЛАУ методом квадратных корней

function [x]=kvadrat(A,b,n)

t(1,1)=A(1,1)^(1/2);

for j=2:n

 t(1,j)=A(1,j)/t(1,1);

end;

for i=2:n

 S=0;

 for k=1:i-1

 S=S+t(k,i)*t(k,i);

 end;

 t(i,i)=(A(i,i)-S)^(1/2);

 for j=i+1:n

 S=0;

 for k=1:i-1

 S=S+t(k,i)*t(k,j);

 end;

 t(i,j)=(A(i,j)-S)/t(i,i);

 end;

end;

y(1)=b(1)/t(1,1);

for i=2:n

 S=0;

 for k=1:i-1

 S=S+t(k,i)*y(k);

 end;


```

    y(i)=(b(i)-S)/t(i,i);
end;
x(n)=y(n)/t(n,n);
i=n-1;
while i>=1
    S=0;
    for k=i+1:n
        S=S+t(i,k)*x(k);
    end;
    x(i)=(y(i)-S)/t(i,i);
    i=i-1;
end;
end
>>%Результат работы функции
>> Z=[3,1,-1,2;-5,1,3,-4;2,0,1,-1;1,-5,3,-3];
>> q=[0.5,5.4,5,7.5,3.3];
>> [y]=kvadrat(Z,q,5)
y = %Корни в комплексном виде
    -6.1000 - 0.0000i    -2.2000 - 0.0000i    -6.8000 - 0.0000i    -0.9000 + 0.0000i
    0.2000 - 0.0000i
>> real(y) '
ans = % Решение заданной системы
    -6.1000
    -2.2000
    -6.8000
    -0.9000
     0.2000
>> Z*real(y) '
ans = %Проверка
     0.5000
     5.4000
     5.0000
     7.5000
     3.3000

```

М-функция, представленная в следующем листинге, решает СЛАУ по схеме Халецкого (см. раздел 1.2.5).

Листинг 1.27

%Решение СЛАУ по схеме Халецкого

```
function [x]=holecki(A,b,n)
```

```
for i=1:n
```

```
    D(i,1)=A(i,1);
```

```
    C(1,i)=A(1,i)/D(1,1);
```

```
    C(i,i)=1;
```

```
end;
```

```
for i=2:n
```

```

for j=2:n
    if i>=j
        S=0;
        for k=1:j-1
            S=S+D(i,k)*C(k,j);
        end;
        D(i,j)=A(i,j)-S;
    end;
    if i<j
        S=0;
        for k=1:i-1
            S=S+D(i,k)*C(k,j);
        end;
        C(i,j)=(A(i,j)-S)/D(i,i);
    end;
end;
end;
y(1)=b(1)/D(1,1);
for i=2:n
    S=0;
    for k=1:i-1
        S=S+D(i,k)*y(k);
    end;
    y(i)=(b(i)-S)/D(i,i);
end;
x(n)=y(n);
i=n-1;
while i>=1
    S=0;
    for k=i+1:n
        S=S+C(i,k)*x(k);
    end;
    x(i)=y(i)-S;
    i=i-1;
end;
end
%Результат работы функции
>> Z=[3,1,-1,2;-5,1,3,-4;2,0,1,-1;1,-5,3,-3];
>> q=[6,-12,1,3];
>> [y]=holeski(Z,q,4)
y = %Решение СЛАУ
    1.0000   -1.0000    2.0000    3.0000
>> Z*y'
ans = %Проверка
     6
    -12
     1
     3

```

Листинг ниже представляет М-функцию для решения СЛАУ методом LU-разложения, подробно описанном в разделе 1.2.6.

Листинг 1.28

%Решение СЛАУ с помощью LU-разложения

```
function [L,U,x]=LU(A,b,n)
for i=1:n
    for j=1:n
        if i<=j
            S=0;
            for k=1:i-1
                S=S+L(i,k)*U(k,j);
            end;
            U(i,j)=A(i,j)-S;
        end;
        if i>j
            S=0;
            for k=1:j-1
                S=S+L(i,k)*U(k,j);
            end;
            L(i,j)=(A(i,j)-S)/U(j,j);
        end;
    end;
end;
y(1)=b(1);
for i=2:n
    S=0;
    for k=1:i-1
        S=S+L(i,k)*y(k);
    end;
    y(i)=(b(i)-S);
end;
x(n)=y(n)/U(n,n);
i=n-1;
while i>=1
    S=0;
    for k=i+1:n
        S=S+U(i,k)*x(k);
    end;
    x(i)=(y(i)-S)/U(i,i);
    i=i-1;
end;
end
%Результат работы функции
>> Z=[3,1,-1,2;-5,1,3,-4;2,0,1,-1;1,-5,3,-3];
>> q=[6,-12,1,3];
> [l,u,X]=LU(Z,q,4)
```

l =

0	0	0
-1.6667	0	0
0.6667	-0.2500	0
0.3333	-2.0000	3.0000

u =

3.0000	1.0000	-1.0000	2.0000
0	2.6667	1.3333	-0.6667
0	0	2.0000	-2.5000
0	0	0	2.5000

X =

1	-1	2	3

Программа для решения линейных систем методом итераций (1.2.9) представлена в листинге ниже.

Листинг 1.29

%Решение СЛАУ методом итерации

function [x1]=iter(A,b,n,ep)

Pr=0

for i=1:n

S=0;

for j=1:n

if i~=j

S=S+abs(A(i,j));

end;

end;

if abs(A(i,i))<=S

Pr=1;

break;

end;

end;

if Pr==0

for i=1:n

betta(i)=b(i)/A(i,i);

x1(i)=betta(i);

for j=1:n

if i==j

alfa(i,j)=0;

else

alfa(i,j)=-A(i,j)/A(i,i);

end;

end;

end;

while 1

for i=1:n

x0(i)=x1(i);

end;

```

for i=1:n
    S=0;
    for j=1:n
        S=S+alfa(i,j)*x0(j);
    end;
    x1(i)=betta(i)+S;
end;
max=abs(x1(1)-x0(1));
for i=2:n
    if abs(x1(i)-x0(i))>max
        max=abs(x1(i)-x0(i));
    end;
end;
if max<ep
    break;
end;
end;
else
    warning("ERROR");
end
%Результат работы функции
>> Z=[4,0.24,-0.08;0.09,3,-0.15;0.04,-0.08,4];
>> q=[8,9,20];
>> [X]=iter(Z,q,3,0.0001)
X'=
    1.9092    3.1950    5.0448
>> Z*X'%Проверка
ans =
    8.0000
    9.0000
   20.0000

```

Реализация метода релаксации (см. раздел 1.2.11) для решения систем линейных уравнений изложена в листинге ниже.

Листинг 1.30

%Решение СЛАУ методом релаксации

function [X]=relax(A,b,n,ep)

```

for i=1:n
    for j=1:n
        D(i,j)=-A(i,j)/A(i,i);
    end;
    C(i)=b(i)/A(i,i);
    X(i)=0;
end;
for i=1:n
    S=0;

```

```

    for j=1:n
        if i~=j
            S=S+D(i,j)*X(j);
        end;
    end;
    R1(i)=C(i)-X(i)+S;
end;
while 1
    for i=1:n
        R0(i)=R1(i);
    end;
    max=R0(1);
    nom=1;
    for i=1:n
        if R0(i)>max
            max=R0(i);
            nom=i;
        end;
    end;
    dx=max;
    for i=1:n
        if i~=nom
            R1(i)=R0(i)+D(i,nom)*dx;
        else
            R1(i)=0;
        end;
    end;
    X(nom)=X(nom)+dx;
    Pr=0;
    for i=1:n
        if R1(i)>ep
            Pr=1;
            break;
        end;
    end;
    if Pr==0
        break;
    end;
end;
end
%Результат работы функции
>> A=[10,-2,-2;-1,10,-2;-1,-1,10];
>> b=[6,7,8];
>> [X]=relax(A,b,3,0.0001)
X =
    1.0000    0.9999    1.0000

```

```
>> A*X' %Проверка
ans =
    5.9997
    6.9992
    8.0000
```

Метод градиента (см. раздел 1.2.12), который также позволяет находить решения линейных систем, представляет М-функцию в листинге ниже.

Листинг 1.31

%Решение СЛАУ методом градиента

```
function [X]=grad(A,b,n,ep)
```

```
for i=1:n
```

```
    Y(i)=b(i)/A(i,i);
```

```
end;
```

```
k=0;
```

```
X=Y';
```

```
while 1
```

```
    R=A*X-b';
```

```
    M=dot(R,A*A'*R)/dot(A*A'*R,A*A'*R);
```

```
    X=X-M*A'*R;
```

```
    k=k+1;
```

```
    if max(abs(R))<ep
```

```
        break;
```

```
    end;
```

```
end;
```

```
end
```

%Результат работы функции

```
>> A=[8,-1,-2,0;0,10,1,2;-1,0,6,2;3,-1,2,12];
```

```
>> b=[2.3,-0.5,-1.2,3.7];
```

```
>> [Q]=grad(A,b,4,0.01)
```

```
Q =
```

```
    0.2125
```

```
   -0.0822
```

```
   -0.2607
```

```
    0.2919
```

```
>> A*Q %Проверка
```

```
ans =
```

```
    2.3038
```

```
   -0.4991
```

```
   -1.1929
```

```
    3.7010
```

Наконец, следующий листинг представляет М-функцию для нахождения наибольшего собственного значения и соответствующего ему вектора. Здесь реализован метод скалярных произведений.

Листинг 1.32

```

%Метод скалярных произведений
%Для определения наибольшего собственного
%значения и соответствующего ему собственного вектора
function [L1,x1]=iter_eg(A,n,ep)
for i=1:n
    x1(i)=1;
end;
L0=1;
while 1
    for i=1:n
        x0(i)=x1(i);
    end;
    for i=1:n
        S=0;
        for j=1:n-1
            S=S+A(i,j)*x0(j);
        end;
        x1(i)=(S+A(i,n))/L0;
    end;
    S=0;
    for j=1:n-1
        S=S+A(n,j)*x1(j);
    end;
    L1=S+A(n,n);
    if abs(L0-L1)<ep
        break;
    end;
    L0=L1;
end;
end
%Результат работы программы
>> A=[4,2,2;2,5,1;2,1,6];
>> [L,X]=iter_eg(A,3,0.001)
L = %Наибольшее собственное значение
    8.3884
X = %Собственный вектор, соответствующий L
    0.8080    0.7725    1.0000
>> (A-L*eye(3))*X'
ans = %Проверка
   -0.0008
   -0.0015
    0

```

1.7. Советы по применению математических пакетов при решении задач линейной алгебры

В заключение можно сказать, что и Mathcad, и Maple, и MATLAB обладают достаточно широкими возможностями для решения задач линейной алгебры. Все рассматриваемые математические пакеты работают с векторами и матрицами и содержат команды их создания и обработки. Кроме того, каждый из пакетов предлагает набор функций для выполнения основных векторных и матричных операций и для решения основных задач линейной алгебры: решение систем линейных уравнений, нахождение собственных векторов и собственных значений матрицы, приведение матриц к специальным формам и так далее.

Хотелось бы отметить особенность MATLAB при решении системы линейных уравнений. Здесь уравнение вида $A \cdot x = b$ можно решить при помощи обратного знака деления $x=A/b$, а можно воспользоваться функцией $x=\text{linsolve}(A, b, \text{options})$ или $[x, r] = \text{linsolve}(A, b)$, которая, в свою очередь, позволяет задать метод решения системы или одновременно получить и решение, и ранг матрицы системы.

По поводу применения Maple для решения задач линейной алгебры, конечно, надо сказать о его возможности выполнять всевозможные операции над векторами и матрицами, элементами которых являются алгебраические выражения, и выдавать результат также в виде общих алгебраических выражений.

1.8. Задания для самостоятельного решения

Выполнить решение приведенных далее задач в Mathcad, Maple и MATLAB.

1. Для матриц A, B и C проверить выполнение следующих тождеств: $(A \cdot B) \cdot C = A \cdot (B \cdot C)$ и $(A^T + B) \cdot C = A^T \cdot C + B \cdot C$, если:

$$A = \begin{pmatrix} 2 & 4 & 1 \\ 1 & -2 & 3 \end{pmatrix}, B = \begin{pmatrix} 1 & 3 \\ -2 & 0 \\ 5 & 1 \end{pmatrix}, C = \begin{pmatrix} 2 & 3 \\ -3 & 1 \end{pmatrix}.$$

2. Решить матричные уравнения $A \cdot X = B$ и $X \cdot A = B$ и выполнить проверку, если:

$$A = \begin{pmatrix} 2 & 1 & 3 \\ 0 & 1 & 2 \\ -1 & 2 & 1 \end{pmatrix}, B = \begin{pmatrix} 2 & 1 & 4 \\ 3 & 0 & 1 \\ -1 & 2 & 3 \end{pmatrix}.$$

Решить систему линейных уравнений:

а) при помощи правила Крамера:

$$\begin{aligned} 4x_1 + x_2 - x_3 &= 6, \\ x_1 - x_2 + x_3 &= 4, \\ 2x_1 - 3x_2 - 3x_3 &= 4. \end{aligned}$$

б) методом обратной матрицы:

$$\begin{aligned} 6x_1 - x_2 - x_3 &= 0, \\ x_1 - 2x_2 + 3x_3 &= 1, \\ 3x_1 + 4x_2 + 4x_3 &= -1. \end{aligned}$$

Решить систему линейных уравнений методом Гаусса:

$$\begin{aligned} 2x_1 - x_2 - 3x_4 &= -9, \\ x_1 - x_3 + 2x_4 &= 8, \\ 3x_1 - 2x_2 + x_3 - x_4 &= -5, \\ -x_1 + 3x_2 - x_3 + x_4 &= 9. \end{aligned}$$

Решить систему линейных уравнений:

а) с помощью LU-разложения:

$$\begin{aligned} 2,34x_1 - 1,42x_2 - 0,54x_3 + 0,21x_4 &= 0,66, \\ 1,44x_1 - 0,53x_2 + 1,43x_3 - 1,27x_4 &= -1,44, \\ 0,63x_1 - 1,32x_2 - 0,65x_3 + 1,43x_4 &= 0,94, \\ 0,56x_1 + 0,88x_2 - 0,67x_3 - 2,38x_4 &= 0,73; \end{aligned}$$

б) с помощью QR-разложения:

$$\begin{aligned} 0,18x_1 + 2,11x_2 + 0,13x_3 - 0,22x_4 &= 0,22, \\ 0,33x_1 - 0,22x_2 - 1,00x_3 + 0,17x_4 &= 0,11, \\ -1,00x_1 + 0,11x_2 + 2,00x_3 - 0,44x_4 &= 1,00, \\ 7,00x_1 - 0,17x_2 - 0,22x_3 + 0,33x_4 &= 0,21; \end{aligned}$$

в) методом ортогонализации:

$$\begin{aligned} 0,63x_1 + 1,00x_2 + 0,71x_3 + 0,34x_4 &= 2,08, \\ 1,17x_1 + 0,18x_2 - 0,65x_3 + 0,71x_4 &= 0,17, \\ 2,71x_1 - 0,75x_2 + 0,17x_3 - 2,35x_4 &= 1,28, \\ 3,58x_1 + 0,21x_2 - 3,45x_3 - 1,18x_4 &= 0,05. \end{aligned}$$

Решить линейную систему методом квадратных корней

$$\begin{aligned} 3,14x_1 - 2,12x_2 + 1,17x_3 &= 1,27, \\ -2,12x_1 + 1,32x_2 - 2,45x_3 &= 2,13, \\ 1,17x_1 - 2,45x_2 + 1,18x_3 &= 3,14. \end{aligned}$$

7. Выполнить решение системы линейных уравнений по схеме Халецкого:

$$9,1x_1 + 5,6x_2 + 7,8x_3 = 9,8,$$

$$3,8x_1 + 5,1x_2 + 2,8x_3 = 6,7,$$

$$4,1x_1 + 5,7x_2 + 1,2x_3 = 5,8.$$

8. Решить линейную систему:

а) методом итерации:

$$-0,76x_1 - 0,04x_2 + 0,21x_3 - 0,18x_4 = -1,24,$$

$$0,45x_1 - 1,23x_2 + 0,66x_3 = 0,88,$$

$$0,26x_1 + 0,34x_2 - 1,11x_3 = -0,63,$$

$$0,05x_1 - 0,26x_2 + 0,34x_3 - 1,12x_4 = 1,17;$$

б) методом Зейделя:

$$-0,76x_1 - 0,04x_2 + 0,21x_3 - 0,18x_4 = -1,24,$$

$$0,45x_1 - 1,23x_2 + 0,66x_3 = 0,88,$$

$$0,26x_1 + 0,34x_2 - 1,11x_3 = -0,63,$$

$$0,05x_1 - 0,26x_2 + 0,34x_3 - 1,12x_4 = 1,17.$$

9. Решить линейную систему:

а) методом релаксации:

$$2,7x_1 + 3,3x_2 + 1,3x_3 = 2,1,$$

$$3,5x_1 - 1,7x_2 + 2,8x_3 = 1,7,$$

$$4,1x_1 + 5,8x_2 - 1,7x_3 = 0,8;$$

б) градиентным методом:

$$1,00x_1 - 2,01x_2 + 2,04x_3 + 0,17x_4 = 0,18,$$

$$0,33x_1 - 0,77x_2 - 0,44x_3 - 0,51x_4 = 0,19,$$

$$0,31x_1 + 0,17x_2 - 0,21x_3 + 0,54x_4 = 0,21,$$

$$0,17x_1 + 1,00x_2 - 0,13x_3 + 0,21x_4 = 0,31.$$

10. Найти собственные значения и собственные векторы матрицы A :

$$A = \begin{pmatrix} 2 & 1 & 4 & 1 \\ 3 & 0 & 1 & 1 \\ -1 & 2 & 3 & 4 \\ 3 & 1 & 1 & 1 \end{pmatrix}.$$

Глава 2

Интерполяция функций

Эта глава посвящена решению часто встречающихся на практике задач по обработке реальных количественных экспериментальных данных, полученных в результате всевозможных научных опытов, технических испытаний методом интерполяции. В первом разделе описаны численные методы интерполирования. В последующих рассмотрено решение задач интерполирования в пакетах Mathcad, MATLAB и Maple.

2.1. Интерполирование функций

Простейшая задача *интерполирования* заключается в следующем. На отрезке $[a, b]$ заданы $n + 1$ точки $x_0, x_1, x_2, \dots, x_n$, которые называют *узлами интерполяции*, и значения некоторой функции $f(x)$ в этих точках

$$f(x_0) = y_0, f(x_1) = y_1, f(x_2) = y_2, \dots, f(x_n) = y_n. \quad (2.1)$$

Требуется построить *интерполирующую функцию* $F(x)$, принадлежащую известному классу и принимающую в узлах интерполяции те же значения, что и $f(x)$:

$$F(x_0) = y_0, F(x_1) = y_1, F(x_2) = y_2, \dots, F(x_n) = y_n. \quad (2.2)$$

Геометрически (рис. 2.1) это означает, что нужно найти кривую $y = F(x)$ определенного типа, проходящую через заданную систему точек $M_i(x_i, y_i)$, где $(i = 0, 1, 2, \dots, n)$.

Решая эту систему относительно переменных $a_0, a_1, \dots, a_n$, найдем коэффициенты интерполяционного полинома (2.3).

Блок-схема нахождения значения функции в точке $t \in [x_0, x_n]$ с помощью интерполяционного канонического полинома приведена на рис. 2.2. Решение системы (2.4) (блок 7 на рис. 2.2) может осуществляться одним из рассмотренных в первой главе методов решения систем линейных алгебраических уравнений (например, методом Гаусса).

На рис. 2.8 приведена реализация канонического полинома в Mathcad, в листинге 2.1 (раздел 2.3) – в MATLAB.

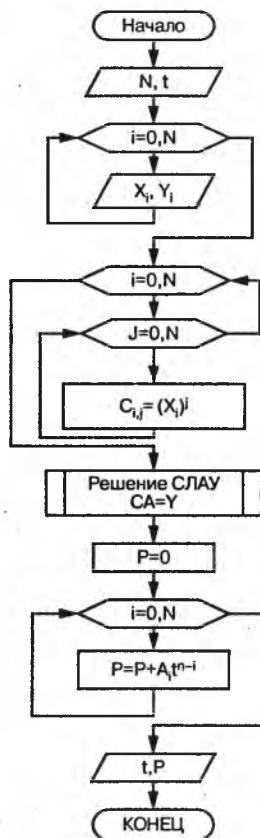


Рис. 2.2 ▼ Блок-схема интерполирования с помощью канонического полинома

2.1.2. Полином Ньютона

Исаак Ньютон предложил записать интерполирующую функцию в виде следующего полинома n -й степени:

$$F(t) = A_0 + A_1(t - x_0) + A_2(t - x_0)(t - x_1) + \dots + A_n(t - x_0)(t - x_1) \dots (t - x_{n-1}). \quad (2.5)$$

Подставим $F(x_0) = y_0$ в (2.5), вычислив тем самым значение коэффициента A_0

$$A_0 = y_0.$$

Подставим $F(x_1) = y_1$ в (2.5), для вычисления A_1 воспользуемся следующим соотношением:

$$F(x_1) = A_0 + A_1(x_1 - x_0) = y_1.$$

Отсюда коэффициент A_1 рассчитывается по формуле:

$$A_1 = \frac{y_0 - y_1}{x_0 - x_1} = y_{01},$$

где y_{01} – *разделенная разность первого порядка*, которая стремится к первой производной функции при $x_1 \rightarrow x_0$. Аналогичным образом вводятся остальные разделенные разности первого порядка:

$$y_{02} = \frac{y_0 - y_2}{x_0 - x_2}, y_{03} = \frac{y_0 - y_3}{x_0 - x_3}, \dots, y_{0n} = \frac{y_0 - y_n}{x_0 - x_n}.$$

Подставим $F(x_2) = y_2$ в (2.5):

$$A_0 + A_1(x_2 - x_0) + A_2(x_2 - x_0)(x_2 - x_1) = y_2;$$

$$y_0 + y_{01}(x_2 - x_0) + A_2(x_2 - x_0)(x_2 - x_1) = y_2;$$

$$A_2(x_2 - x_0)(x_2 - x_1) = y_2 - y_0 - y_{01}(x_2 - x_0);$$

$$A_2 = \frac{y_2 - y_0}{(x_2 - x_0)(x_2 - x_1)} - \frac{y_{01}(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)};$$

$$A_2 = \frac{y_{02}}{x_2 - x_1} - \frac{y_{01}}{x_2 - x_1} = \frac{y_{01} - y_{02}}{x_1 - x_2} = y_{012},$$

где y_{012} – *разделенная разность второго порядка*, эта величина стремится ко второй производной. Аналогично вводятся $y_{013}, y_{014}, \dots, y_{01n}$:

$$y_{013} = \frac{y_{01} - y_{03}}{x_1 - x_3}, y_{014} = \frac{y_{01} - y_{04}}{x_1 - x_4}, \dots, y_{01n} = \frac{y_{01} - y_{0n}}{x_1 - x_n}.$$

Подставим $F(x_3) = y_3$ в (2.5), получим:

$$A_3 = y_{0123} = \frac{y_{012} - y_{013}}{x_2 - x_3}.$$

Аналогично можно ввести коэффициенты:

$$y_{0124} = \frac{y_{012} - y_{014}}{x_2 - x_4}.$$

Этот процесс будем продолжать до тех пор, пока не вычислим:

$$A_n = y_{012\dots n} = \frac{y_{012\dots n-1} - y_{012\dots n-2n}}{x_{n-1} - x_n}.$$

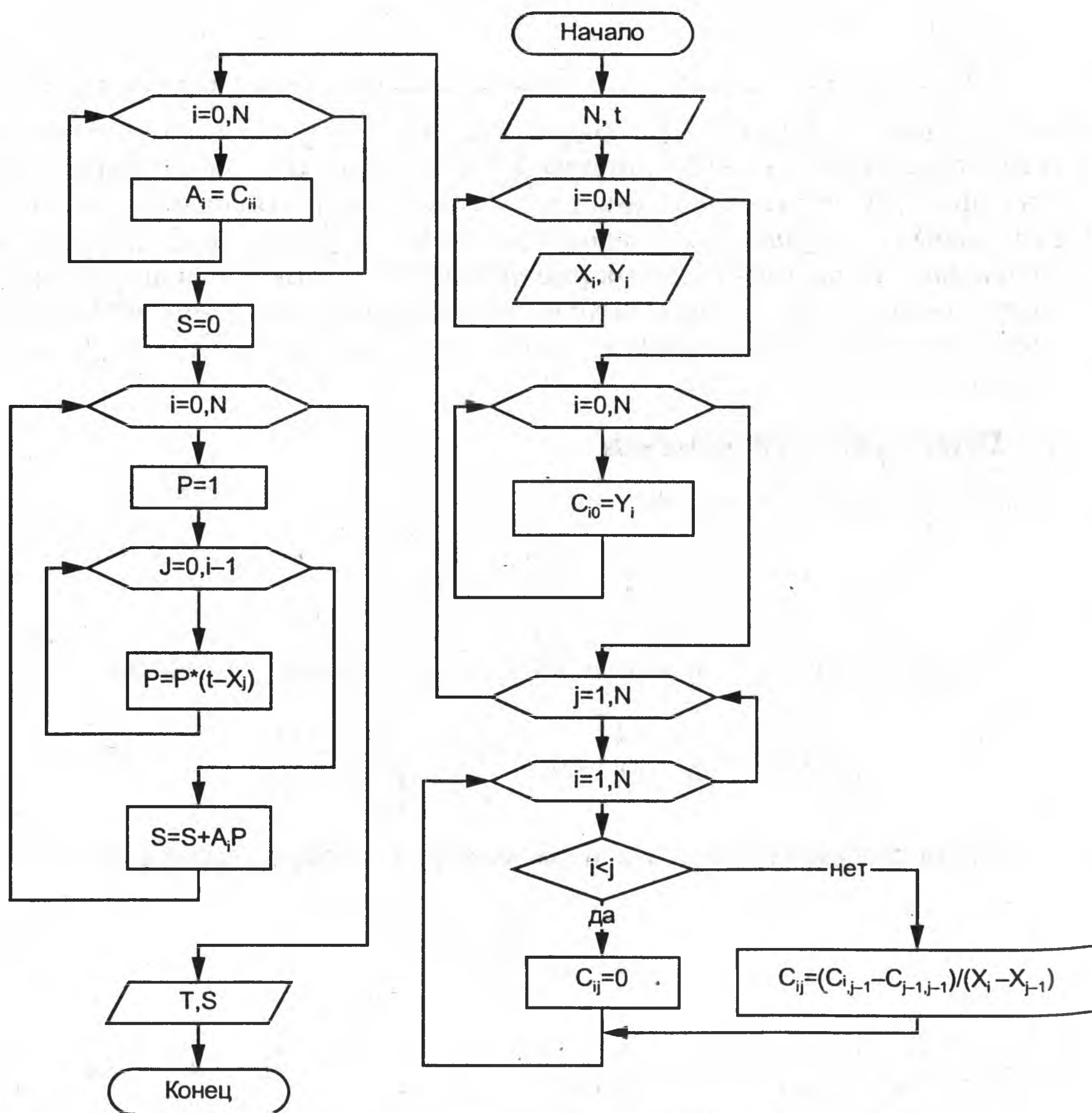


Рис. 2.3 ▼ Алгоритм интерполирования с помощью полинома Ньютона

Полученные результаты запишем в табл. 2.1.

Таблица 2.1 ▼ Таблица разделенных разностей полинома Ньютона

x	$f(x)$	1	2	3	4	...	n
x_0	y_0						
x_1	y_1	y_{01}					
x_2	y_2	y_{02}	y_{012}				
x_3	y_3	y_{03}	y_{013}	y_{0123}			
x_4	y_4	y_{04}	y_{014}	y_{0124}	y_{01234}		
...	...	...	...	...	...	...	...
x_n	y_n	y_{0n}	y_{01n}	y_{012n}	y_{0123n}	...	$y_{012...n}$

В вычислении по формуле (2.5) будут участвовать только диагональные элементы таблицы (то есть коэффициенты A_i), а все остальные элементы таблицы являются промежуточными и нужны для вычисления диагональных элементов. Исходя из вышеизложенного, составим блок-схему алгоритма (рис. 2.3) решения следующей задачи: вычислить значение функции, заданной в виде таблицы, в некоторой точке $t \in [x_0; x_n]$, используя интерполяционный полином Ньютона.

На рис. 2.10 приведена реализация полинома Ньютона в Mathcad, в листинге 2.2 (раздел 2.3) – в MATLAB.

2.1.3. Полином Лагранжа

Полином Лагранжа будем искать в виде:

$$F(t) = \sum_{i=1}^n y_i L_i(t),$$

где $L_i(t)$ – функция, удовлетворяющая в узлах x_k следующему свойству:

$$L_i(x_k) = \begin{cases} 1, & i = k \\ 0, & i \neq k \end{cases} \quad \text{или} \quad L_i(t) = \prod_{\substack{j=0 \\ j \neq i}}^n \frac{t - x_j}{x_i - x_j}.$$

Таким образом, полином Лагранжа выражается следующей формулой:

$$F(t) = \sum_{i=0}^n y_i \prod_{\substack{j=0 \\ j \neq i}}^n \frac{t - x_j}{x_i - x_j}. \quad (2.6)$$

Алгоритм вычисления значения функции в заданной точке (рис. 2.4) достаточно прост и не нуждается в пояснении.

На рис. 2.9 приведена реализация полинома Лагранжа в Mathcad, в листинге 2.3 (раздел 2.3) – в MATLAB.

Все три рассмотренных полинома – это три различных способа построения одной и той же кривой $F(x) = P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1} + a_nx^n$

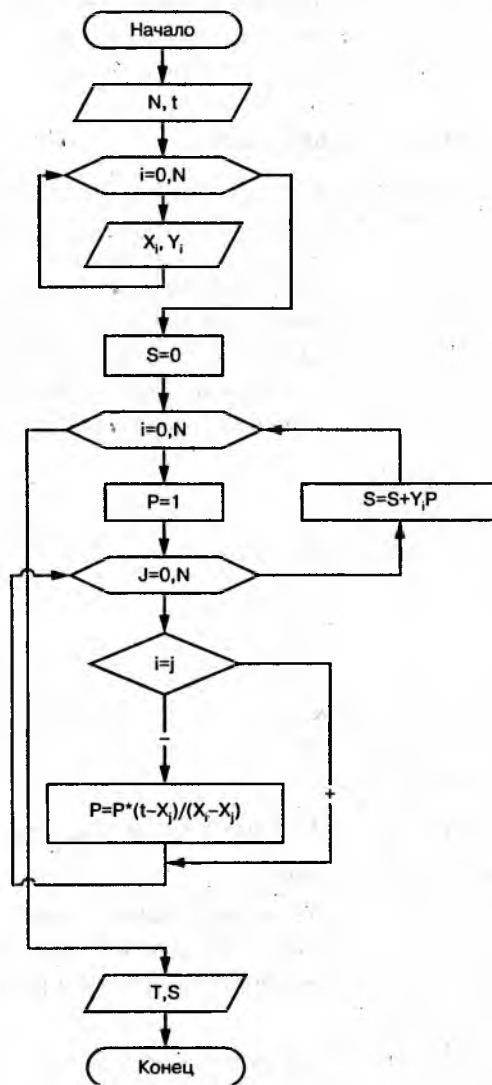


Рис. 2.4 ▼ Алгоритм интерполирования с помощью полинома Лагранжа



Полином Лагранжа лучше использовать, если необходимо вычислить значение в небольшом количестве точек. Для расчета во многих точках рационально использовать полином Ньютона, в котором, вычислив значения коэффициентов A_r , можно затем рассчитать ожидаемое значение в точках по формуле (2.5). При использовании канонического полинома приходится решать систему линейных алгебраических уравнений (2.4), поэтому он используется значительно реже.

2.1.4. Интерполяция сплайнами

Полиномиальная интерполяция не всегда дает удовлетворительные результаты при аппроксимации зависимостей. Так, например, при представлении полиномами резонансных кривых колебательных систем большая погрешность возникает на концах этих кривых. Несмотря на выполнение условий в узлах, интерполяционная функция может иметь значительное отклонение от аппроксимируемой кривой между узлами. При этом повышение степени интерполяционного полинома приводит не к уменьшению, а к увеличению погрешности. Решение этой проблемы предложено теорией сплайн-интерполяции (от английского слова spline – рейка, линейка).

Рассмотрим один из наиболее распространенных вариантов интерполяции кубическими сплайнами [9]. Используя законы упругости, можно установить, что недеформируемая линейка между соседними углами проходит по линии, удовлетворяющей уравнению

$$\varphi^{IV}(x) = 0. \quad (2.7)$$

Функцию $\varphi(x)$ будем использовать для интерполяции зависимости $y(x)$, заданной на интервале $(a; b)$ в узлах $a = x_0, x_1, \dots, b = x_n$ значениями $y_0, y_1, \dots, y_n$.

Кубическим сплайном, интерполирующим на отрезке $[a; b]$ данную функцию $y(x)$, называется функция [9]

$$g_k(s) = a_k + b_k(s - x_k) + c_k(s - x_k)^2 + d_k(s - x_k)^3, \quad s \in [x_{k-1}, x_k], \quad k = 1, \dots, n, \quad (2.8)$$

удовлетворяющая следующим условиям:

- $g_k(x_k) = y_k; g_k(x_{k-1}) = y_{k-1}$ (условие интерполяции в узлах сплайна);
- функция $g(x)$ дважды непрерывно дифференцируема на интервале $[a; b]$;
- на концах интервала функция g должна удовлетворять следующим соотношениям $g_1''(a) = g_n''(b) = 0$.

Для построения интерполяционного сплайна необходимо найти $4n$ коэффициентов a_k, b_k, c_k, d_k ($k = 1, 2, \dots, n$).

Из определения сплайна получаем $n + 1$ соотношение (2.9) [9]:

$$g_1(x_0) = y_0; \quad g_k(x_k) = y_k \quad k = 1, 2, \dots, n. \quad (2.9)$$

Из условий гладкой стыковки звеньев сплайна (во внутренних узловых точках совпадают значения двух соседних звеньев сплайна¹, их первые и вторые производные) получаем еще ряд соотношений (2.10–2.11) [9]:

$$\left. \begin{aligned} g_{k-1}(x_{k-1}) &= g_k(x_{k-1}) \\ g'_{k-1}(x_{k-1}) &= g'_k(x_{k-1}) \\ g''_{k-1}(x_{k-1}) &= g''_k(x_{k-1}) \end{aligned} \right\}, \quad k = 2, 3, \dots, n, \quad (2.10)$$

$$g''_1(x_0) = 0, \quad g''_n(x_n) = 0. \quad (2.11)$$

Соотношения (2.9)–(2.11) образуют $4n$ соотношений для нахождения коэффициентов сплайна. Подставляя выражения функций (2.8) и их производных (2.12)

$$\begin{aligned} g'_k(s) &= b_k + 2c_k(s - x_k) + 3d_k(s - x_k)^2, \\ g''_k(s) &= 2c_k + 6d_k(s - x_k) \end{aligned} \quad (2.12)$$

в соотношения (2.9)–(2.11) и принимая во внимание соотношение

$$h_k = x_k - x_{k-1}, \quad k = 1, 2, \dots, n, \quad (2.13)$$

получим следующую систему уравнений (2.14)–(2.20) [9]:

$$a_1 - b_1 h_1 + c_1 h_1^2 - d_1 h_1^3 = y_0, \quad (2.14)$$

$$a_k = y_k, \quad k = 1, 2, \dots, n, \quad (2.15)$$

$$a_{k-1} = a_k - b_k h_k + c_k h_k^2 - d_k h_k^3; \quad k = 2, 3, \dots, n, \quad (2.16)$$

$$b_{k-1} = b_k - 2c_k h_k + 3d_k h_k^2; \quad k = 2, 3, \dots, n, \quad (2.17)$$

$$c_{k-1} = c_k - 3d_k h_k; \quad k = 2, 3, \dots, n, \quad (2.18)$$

$$c_1 - 3d_1 h_1 = 0, \quad (2.19)$$

$$c_n = 0. \quad (2.20)$$

Задача интерполяции свелась к решению системы (2.14–2.20). Из соотношения (2.15) следует, что все коэффициенты $a_k = y_k, \quad k = 1, 2, \dots, n$. Подставив соотношения

¹ Звеном сплайна называется функция $g_i(x)$ на интервале $[x_{i-1}; x_i]$.

(2.14), (2.15) в (2.16) и используя фиктивный коэффициент $C_0 = 0$, получим соотношение между b_k , c_k и d_k [9]:

$$b_k h_k - c_k h_k^2 + d_k h_k^3 = y_k - y_{k-1}.$$

Отсюда коэффициенты b_k вычисляются при $k = 1, 2, \dots, n$ по формуле [9]:

$$b_k = \frac{y_k - y_{k-1}}{h_k} + c_k h_k - d_k h_k^2. \quad (2.21)$$

Из (2.18) и (2.19) выразим d_k через c_k (с учетом коэффициента $c_0 = 0$) [9]:

$$d_k = \frac{c_k - c_{k-1}}{3h_k}; \quad k = 1, 2, \dots, n. \quad (2.22)$$

Подставим (2.22) в (2.21):

$$b_k = \frac{y_k - y_{k-1}}{h_k} + \frac{2}{3}c_k h_k + \frac{1}{3}h_k c_{k-1}; \quad k = 1, 2, \dots, n. \quad (2.23)$$

Введем обозначение

$$l_k = \frac{y_k - y_{k-1}}{h_k}; \quad k = 1, 2, \dots, n, \quad (2.24)$$

после чего соотношение (2.23) примет вид:

$$b_k = l_k + \frac{2}{3}c_k h_k + \frac{1}{3}h_k c_{k-1}; \quad k = 1, 2, \dots, n. \quad (2.25)$$

Подставим (2.25) и (2.22) в соотношение (2.17), получим систему относительно c_k :

$$h_{k-1} c_{k-2} + 2(h_{k-1} + h_k) c_{k-1} + h_k c_k = 3l_k - 3l_{k-1}; \quad k = 2, 3, \dots, n, \quad (2.26)$$

$$c_0 = 0, c_n = 0. \quad (2.27)$$

Систему (2.26) можно решить, используя метод прогонки [9]. Этот метод сводится к нахождению прогоночных коэффициентов по формулам прямой прогонки [9]:

$$\delta_1 = -\frac{h_2}{2(h_1 + h_2)}, \quad \lambda_1 = \frac{3(l_2 - l_1)}{2(h_1 + h_2)}, \quad (2.28)$$

$$\delta_{k-1} = -\frac{h_k}{2h_{k-1} + 2h_k + h_{k-1}\delta_{k-2}}, \quad \lambda_{k-1} = \frac{3l_k - 3l_{k-1} - h_{k-1}\lambda_{k-2}}{2h_{k-1} + 2h_k + h_{k-1}\delta_{k-2}}; \quad k = 3, 4, \dots, n, \quad (2.29)$$

а затем к нахождению искоемых коэффициентов c_k по формулам обратной прогонки:

$$c_{k-1} = \delta_{k-1} c_k + \lambda_{k-1}; \quad k = n, n-1, \dots, 2. \quad (2.30)$$

После нахождения коэффициентов c по формуле (2.30) находим b и d по формулам (2.22), (2.25).

Таким образом, алгоритм расчета коэффициентов интерполяционного сплайна состоит из следующих этапов:

Этап 1. Ввод значений табличной зависимости $y(x)$, массивов x и y ;

Этап 2. Расчет элементов массивов h и l по формулам (2.13) и (2.24);

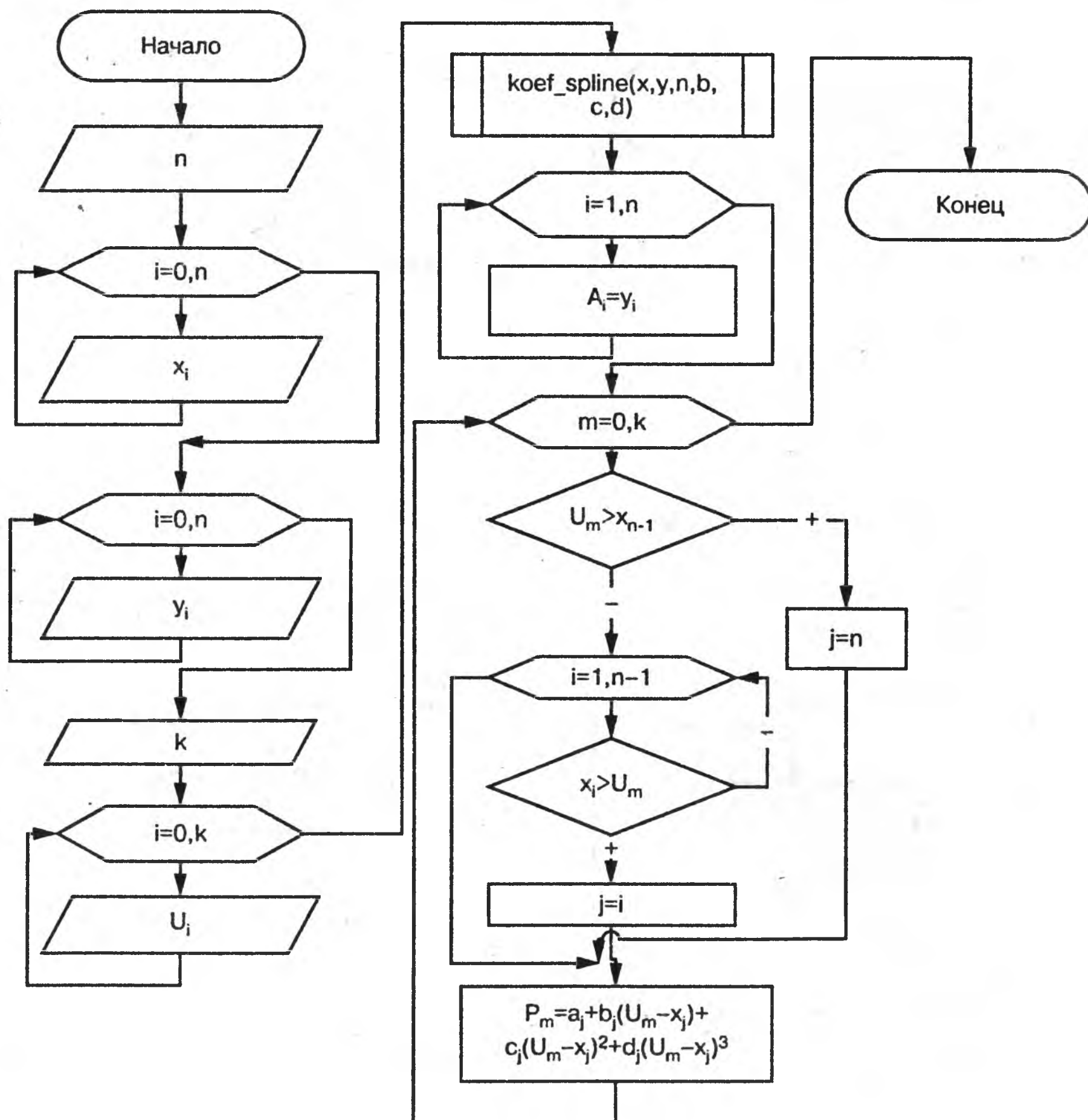


Рис. 2.5 ▽ Блок-схема алгоритма сплайн-интерполяции

Этап 3. Расчет массивов прогоночных коэффициентов δ и λ по формулам (2.28), (2.29);

Этап 4. Расчет массивов коэффициентов c по формуле (2.30);

Этап 5. Расчет массивов коэффициентов b по формуле (2.25);

Этап 6. Расчет массивов коэффициентов d по формуле (2.22).

После этого в формулу (2.8) можно подставлять любую точку s и вычислять ожидаемое значение.

На рис. 2.5 представлена блок-схема решения следующей задачи интерполяции. Задана табличная зависимость $y(x)$ в узлах $x_0, x_1, \dots, x_n$ значениями $y_0, y_1, \dots, y_n$.

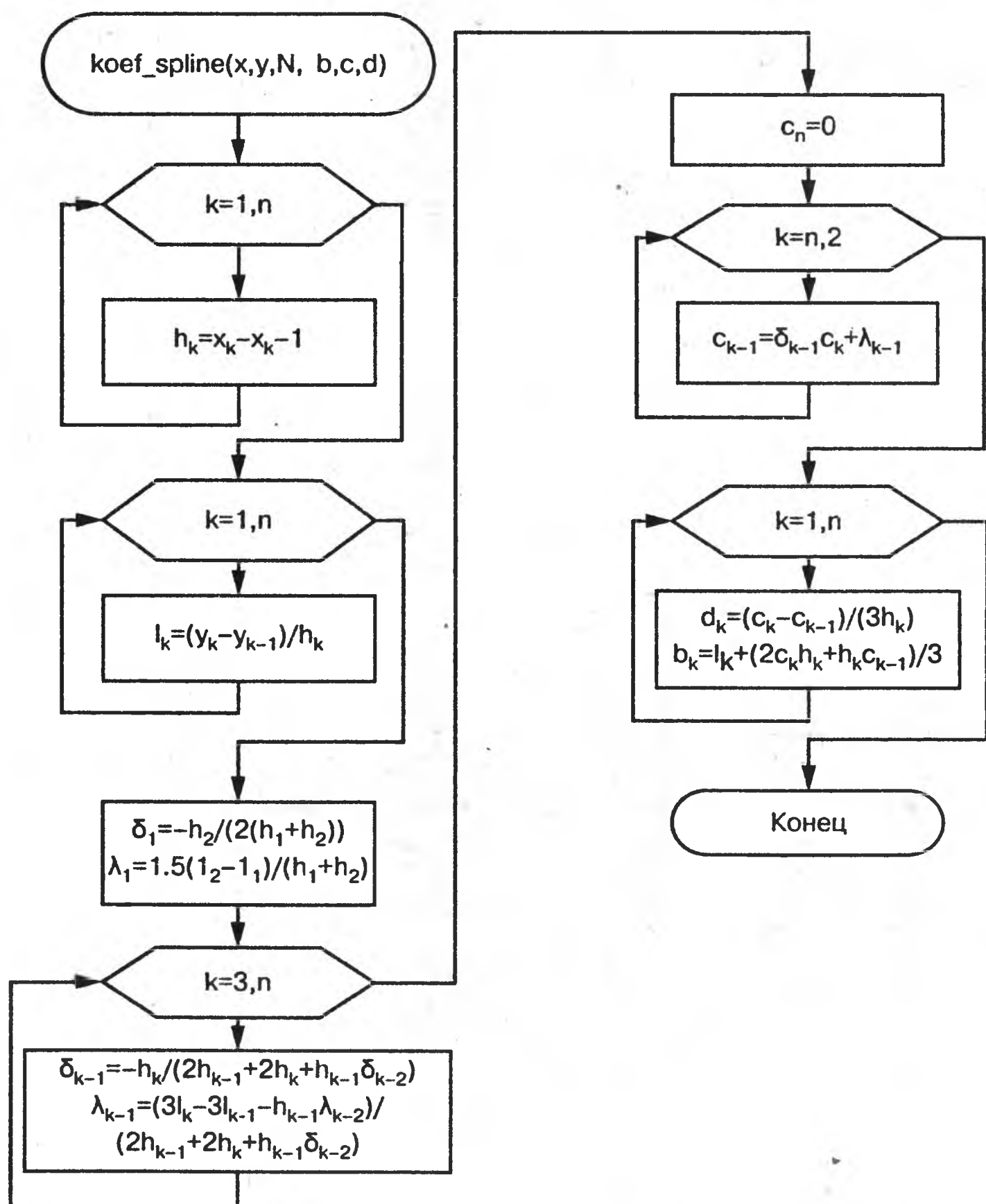


Рис. 2.6 ▼ Блок-схема функции coef_spline расчета коэффициентов сплайна

Построить кубический сплайн и вычислить значения функции в точках U_i ($i = 0, 1, \dots, k$).

Блок 10 блок-схемы реализует обращение к функции `coef_spline` расчета коэффициентов сплайна b , c и d (формулы 2.22, 2.25, 2.26–2.29), блок-схема которой представлена на рис. 2.6.

В листингах 2.5 и 2.6 приведена реализация интерполяции сплайнами в MATLAB.

2.1.5. Линейная интерполяция

Расчет коэффициентов кубического сплайна очень громоздкий и зачастую на практике вместо кубического сплайна используется линейная интерполяция (линейный сплайн)¹.

В случае линейной интерполяции в качестве сплайна выступает линейная функция

$$f_k(s) = a_k + b_k s, s \in [x_{k-1}, x_k], k = 1, \dots, n, \quad (2.32)$$

удовлетворяющая условию интерполяции в узлах сплайна $g_k(x_k) = y_k; g_k(x_{k-1}) = y_{k-1}$. Коэффициенты a и b в этом случае рассчитываются по формулам (2.33), которые получаются из уравнения прямой, проходящей через две точки:

$$\begin{aligned} a_k &= y_{k-1} - \frac{y_k - y_{k-1}}{x_k - x_{k-1}} x_{k-1}, \\ b_k &= \frac{y_k - y_{k-1}}{x_k - x_{k-1}}. \end{aligned} \quad (2.33)$$

Найдя коэффициенты линейного сплайна, можно рассчитать значения в любой точке интервала $[x_0; x_n]$. Блок-схема подбора зависимости с помощью линейного сплайна (рис. 2.7) будет отличаться от программы подбора зависимости кубической интерполяции более простой подпрограммой расчета коэффициентов линейного сплайна a_k и b_k (см. формулу (2.33)).

Линейная интерполяция дает достаточно хорошие результаты при практическом счете внутри интервала $[x_0; x_n]$, когда от получаемой функции не требуют дополнительных свойств (дифференцируемости и т.д.).

Далее мы рассмотрим реализацию интерполяции в математических пакетах. Обратите внимание, что в Mathcad, MATLAB и Maple отсутствуют встроенные

¹ Использование линейного сплайна оправдано в случае, если необходимо просто вычислить значение в определенных точках и нет требования непрерывности производных интерполяционной функции.

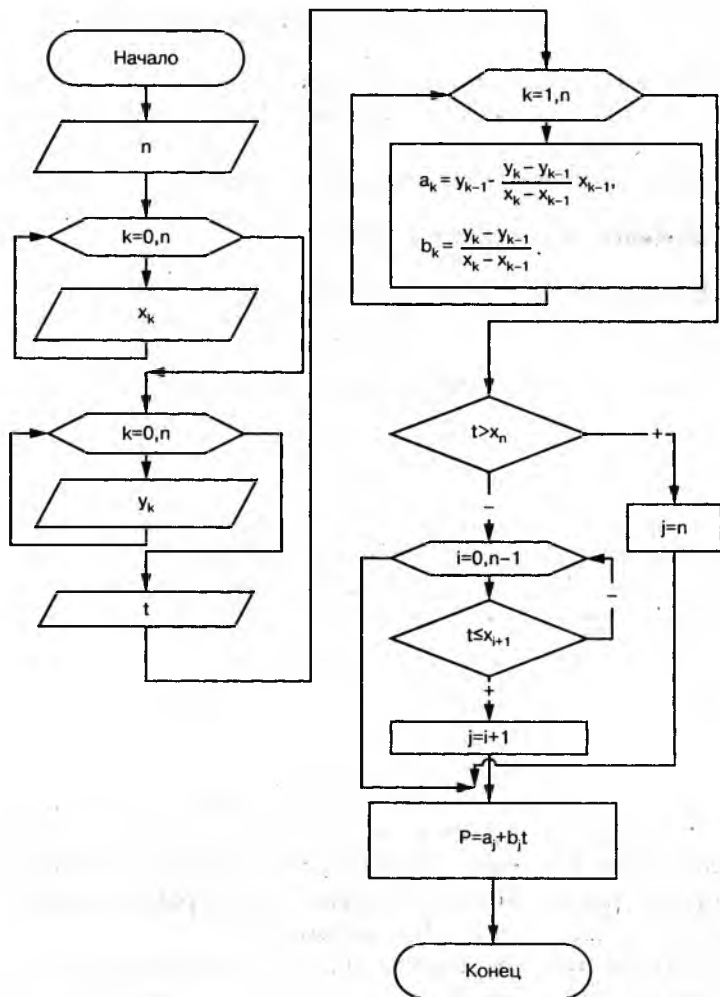


Рис. 2.7 ▼ Блок-схема алгоритма линейного сплайна

функции, реализующие интерполяционный полином n -й степени. Однако эти функции без проблем реализуются с помощью других функций. Для построения сплайнов во всех пакетах есть встроенные функции.

2.2. Решение задач интерполяции в Mathcad

Реализацию интерполяции в Mathcad рассмотрим на примере следующей задачи.

ЗАДАЧА 2.1. В результате опыта холостого хода определена зависимость потребляемой из сети мощности (P_0 , Вт) от входного напряжения (U_1 , В) для асинхронного двигателя МТН111-6 (табл. 2.2).

Таблица 2.2 ▼ Зависимость $P_0(U_1)$

Напряжение (U_1 , В)	132	140	150	162	170	180	190	200	211	220	232	240	251
Мощность (P), Вт	330	350	385	425	450	485	540	600	660	730	920	1020	1350

Построить график интерполяционной зависимости $P_0(U_1)$. На рис. 2.8 приведено решение задачи 2.1 с помощью канонического полинома.

Экспериментальные точки

$$U_1 := (132 \ 140 \ 150 \ 162 \ 170 \ 180 \ 190 \ 200 \ 211 \ 220 \ 232 \ 240 \ 251)^T$$

$$P_0 := (330 \ 350 \ 385 \ 425 \ 450 \ 485 \ 540 \ 600 \ 660 \ 730 \ 920 \ 1020 \ 1350)^T$$

Формирование матрицы коэффициентов **A** системы (2.4)

$$i := 0..12 \quad j := 0..12$$

$$A_{i,j} := (U_1)^j$$

Решение системы (2.4) $a := \text{lsolve}(A, P_0)$

$$\text{Формирование канонического полинома} \quad P(t) := \sum_{i=0}^{12} (a_i \cdot t^i)$$

$$t := 132, 133..251$$

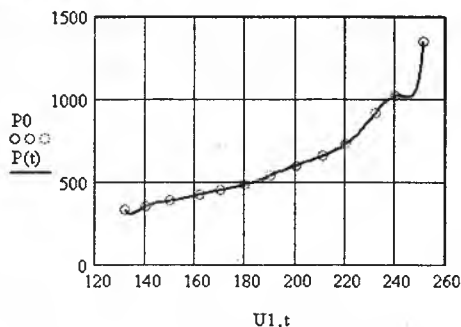


Рис. 2.8 ▼ Реализация канонического полинома в Mathcad

На рис. 2.9 приведено решение задачи 2.1 с помощью полинома Лагранжа.

$n := 12$

$$\text{Lagrang}(t) := \sum_{i=0}^n \left(P0_i \cdot \prod_{j=0}^n \text{if} \left(i = j, 1, \frac{t - U1_j}{U1_i - U1_j} \right) \right)$$

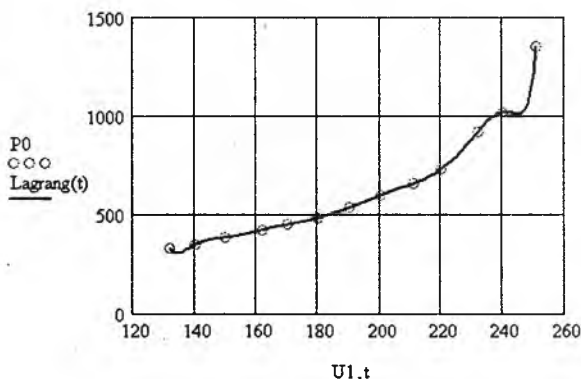


Рис. 2.9 ▼ Реализация полинома Лагранжа в Mathcad

На рис. 2.10 приведено решение задачи 2.1 с помощью полинома Ньютона.

Как видно из рис. 2.8–2.10, интерполяционные полиномы Ньютона, Лагранжа и канонический полином – это разные способы построения одной и той же функции.

В Mathcad реализованы функции построения кубического, квадратичного и линейного сплайна:

- ▶ $\text{cspline}(x, y)$ – возвращает массив коэффициентов кубического сплайна;
- ▶ $\text{pspline}(x, y)$ – возвращает массив коэффициентов квадратичного сплайна;
- ▶ $\text{lspline}(x, y)$ – возвращает массив коэффициентов линейного сплайна.

Во всех этих функциях x – массив абсцисс экспериментальных точек, y – массив ординат экспериментальных точек. После вычисления коэффициентов сплайна можно, обратившись к функции $\text{interp}(k, x, y, t)$, вычислить значение интерполяционного полинома в конкретной точке. Для этого в функцию interp первым параметром (k) необходимо передать массив коэффициентов сплайна; второй (x) и третий (y) параметры – абсциссы и ординаты экспериментальных точек; t – точка, в которой необходимо рассчитать значение.

Экспериментальные точки

$$U1 := (132 \ 140 \ 150 \ 162 \ 170 \ 180 \ 190 \ 200 \ 211 \ 220 \ 232 \ 240 \ 251)^T$$

$$P0 := (330 \ 350 \ 385 \ 425 \ 450 \ 485 \ 540 \ 600 \ 660 \ 730 \ 920 \ 1020 \ 1350)^T$$

Формирование матрицы C $i := 0..n$ $C_{i,0} := P0_i$ $n := \text{last}(P0)$

$$j := 1..n \quad i := 1..n$$

$$t := 131, 132..251$$

$$C_{i,j} := \text{if} \left(i \geq j, \frac{C_{i,j-1} - C_{j-1,j-1}}{U1_i - U1_{j-1}}, 0 \right)$$

Формирование полинома Ньютона

$$F(t) := \sum_{i=0}^n \left(C_{i,i} \prod_{j=0}^{i-1} \text{if}(j \geq i, 1, t - U1_j) \right)$$

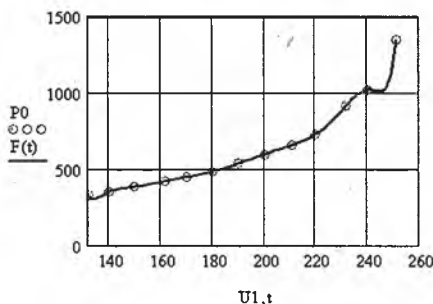


Рис. 2.10 ▼ Реализация полинома Ньютона в Mathcad

На рис. 2.11 представлено решение задачи 2.1 с помощью сплайн-интерполяции.

В Mathcad реализован еще один вид интерполяции – линейная интерполяция; в этом случае все экспериментальные точки просто соединяются прямыми линиями. Для построения линейной интерполяции в Mathcad предусмотрено использование функции $\text{linerp}(x, y, t)$ ¹. Функция $\text{linerp}(x, y, t)$ возвращает значение в точке t , вычисленное с помощью линейной интерполяции. На рис. 2.12 представлено решение задачи 2.1 методом линейной интерполяции.

¹ Здесь, как и ранее, x – массив абсцисс экспериментальных точек, y – массив ординат экспериментальных точек.

Вычисление коэффициентов кубического сплайна $c := \text{cspline}(U1, P0)$

Вычисление коэффициентов квадратичного сплайна $p := \text{pspline}(U1, P0)$

Вычисление коэффициентов линейного сплайна $l := \text{lspline}(U1, P0)$

Функция, вычисляющая значение в точке с помощью кубического сплайна

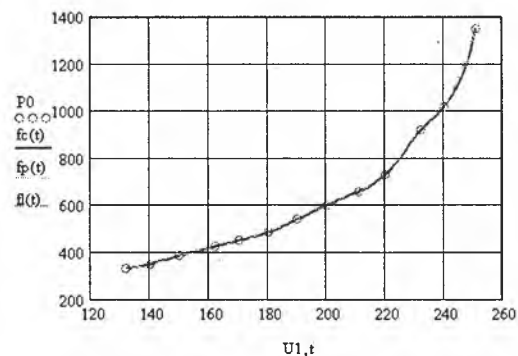
$fc(t) := \text{interp}(c, U1, P0, t)$

Функция, вычисляющая значение в точке с помощью квадратичного сплайна

$fp(t) := \text{interp}(p, U1, P0, t)$

Функция, вычисляющая значение в точке с помощью линейного сплайна

$fl(t) := \text{interp}(l, U1, P0, t)$



ис. 2.11 ▼ Решение задачи 2.1 с помощью сплайн-интерполяции

$U1 := (132 \ 140 \ 150 \ 162 \ 170 \ 180 \ 190 \ 200 \ 211 \ 220 \ 232 \ 240 \ 251)^T$

$P0 := (330 \ 350 \ 385 \ 425 \ 450 \ 485 \ 540 \ 600 \ 660 \ 730 \ 920 \ 1020 \ 1350)^T$

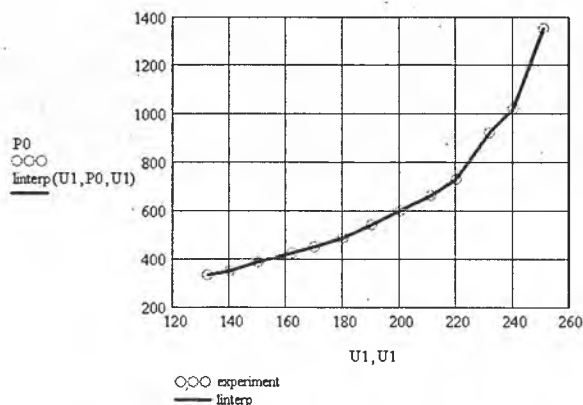


Рис. 2.12 ▼ Решение задачи 2.1 с помощью функции linterp

2.3. Реализация интерполяции в MATLAB

Рассмотрим построение полинома n -й степени в MATLAB на примере следующей задачи.

ЗАДАЧА 2.2. Найти приближенное значение функции при заданном значении аргумента с помощью интерполяционных полиномов n -й степени в точках $x_1 = 0,702$, $x_2 = 0,512$, $x_3 = 0,608$. Функция задана таблично (табл. 2.3).

Таблица 2.3 ▼ Исходные данные к задаче 2.2

x	0.43	0.48	0.55	0.62	0.7	0.75
y	1.63597	1.73234	1.87686	2.03345	2.22846	2.35973

В MATLAB отсутствуют встроенные функции для построения интерполяционного полинома n -й степени, однако средств пакета достаточно для написания функций, реализующих полином. В листингах 2.1–2.3 представлена функция на М-языке для вычисления значения канонического полинома, полиномов Ньютона и Лагранжа в точке t по экспериментальным значениям $(x_i, y_i, i = 1, 2, \dots, n)^1$.

Листинг 2.1²

```
function s=kanon(x,y,t)
% Вычисление количества точек в массивах x и y
n=length(x);
% Формирование коэффициентов системы уравнений (2.4)
for i=1:n
    for j=1:n
        A(i,j)=x(i).^(j-1);
    end
end
% Решение системы уравнений (2.4)
a=A^(-1)*y';
% Вычисление значения полинома по формуле (2.5)
s=0;
for i=1:n
    s=s+a(i)*t^(i-1);
end
end
```

¹ В функциях учтено, что элементы массивов и матриц нумеруются с нуля.

² Функция `kanon` предназначена для вычисления значения канонического полинома в точке t по экспериментальным значениям $x(i)$, $y(i)$.

Листинг 2.2¹

```
function s=newton(x,y,t)
% Вычисление количества точек в массивах x и y
n=length(x);
% Запись в первый столбец матрицы разделенных разностей вектора y
for i=1:n
    C(i,1)=y(i);
end
% Формирование матрицы разделенных разностей
for i=2:n
    for j=2:n
        if (i<j)
            C(i,j)=0;
        else
            C(i,j)=(C(i,j-1)-C(j-1,j-1))/(x(i)-x(j-1))
        end
    end
end
% Формирование массива коэффициентов полинома Ньютона
for i=1:n
    A(i)=C(i,i);
end
% Расчет значения полинома в точке t по формуле (2.5)
s=0;
for i=1:n
    p=1;
    for j=1:i-1
        p=p*(t-x(j));
    end
    s=s+A(i)*p;
end
end
```

Листинг 2.3²

```
function s=lagrang(x,y,t)
% Вычисление количества точек в массивах x и y
n=length(x);
s=0;
% Расчет суммы произведений по формуле (2.6)
% для вычисления значения полинома Лагранжа в точке t
for i=1:n
```

¹ Функция newton предназначена для вычисления значения полинома по экспериментальным значениям $x(i)$, $y(i)$.

² Функция lagrang предназначена для вычисления значения полинома по экспериментальным значениям $x(i)$, $y(i)$.


```

p=1;
for j=1:n
    if (j~=i)
        p=p*(t-x(j))/(x(i)-x(j));
    end
end
s=s+y(i)*p;
end
end

```

Для вычисления значения функции в точках $x_1 = 0,702$, $x_2 = 0,512$, $x_3 = 0,608$ необходимо сформировать массив $r = (0,702 \cdot 0,512 \cdot 0,608)$ и в цикле обратиться к одной из приведенных в листингах 2.1–2.3 функций. В листинге 2.4 показано обращение к функции `lagrang`¹ для вычисления значения полинома в точках $x_1 = 0,702$, $x_2 = 0,512$, $x_3 = 0,608$.

Листинг 2.4

```

r=[0.702 0.512 0.608]
r =
0.7020    0.5120    0.6080
for i=1:3
    rs(i)=lagrang(x,y,r(i));
end
rs
rs =
2.2336    1.7970    2.0057

```

На рис. 2.13 приведен график подобранного полинома.

Задачу 2.2 можно решить и с помощью сплайн-интерполяции. В листинге 2.5 приведена функция `coef_spline(x,y)`, предназначенная для расчетов кубического сплайна, в листинге 2.6 представлена функция, `my_spline(x,y,t)`, которая, используя `coef_spline`, вычисляет значение кубического сплайна в заданной точке t .

Листинг 2.5

```

function [b, c, d]=coef_spline(x,y)
% Функция предназначена для вычисления коэффициентов сплайна,
% здесь x, y - массивы абсцисс и ординат экспериментальных точек
% b, c, d - коэффициенты сплайна, рассчитываемые по формулам (2.21),
% (2.23), (2.27), (2.30)
*
n=length(x);
for k=2:n
    h(k)=x(k)-x(k-1);

```

¹ Так же как к функции `lagrang`, можно обратиться к функциям `kanon` и `newton`.

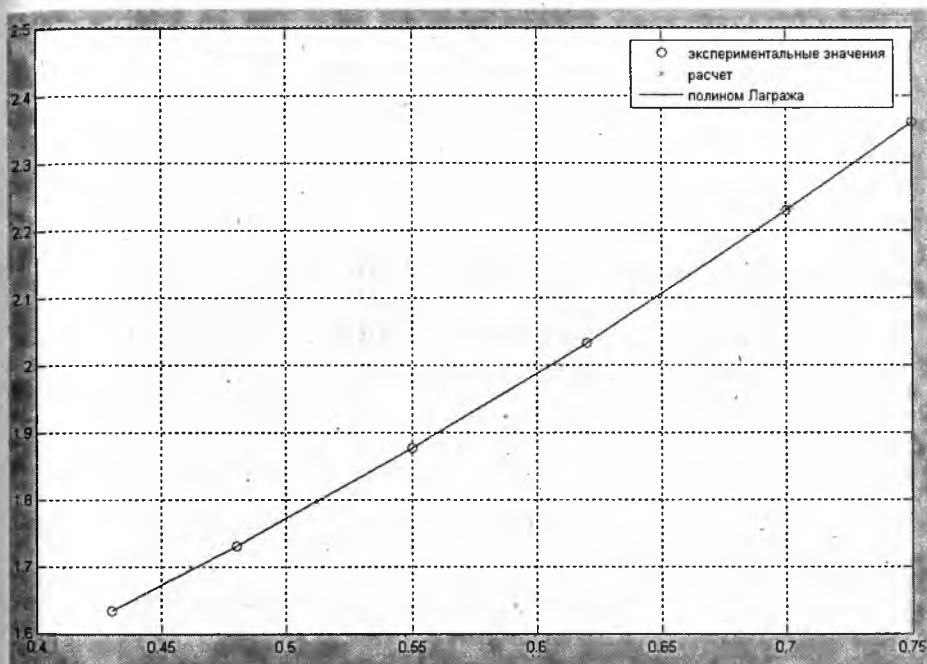


Рис. 2.13 ▽ График решения задачи 2.2

```

end
for k=2:n
    l(k)=(y(k)-y(k-1))/h(k);
end
delt(2)=-h(3)/(2*(h(3)+h(2)));
lyam(2)=1.5*(l(3)-l(2))/(h(3)+h(2));
for k=4:n
    delt(k-1)=-h(k)/(2*(h(k-1)+h(k))+h(k-1)*delt(k-2));
    lyam(k-1)=(3*(l(k)-l(k-1))-h(k-1)*delt(k-2))/
        (2*(h(k-1)+h(k))+h(k-1)*delt(k-2));
end
c(n)=0;
for k=n:-1:3
    c(k-1)=delt(k-1)*c(k)+delt(k-1);
end
for k=2:n
    d(k)=(c(k)-c(k-1))/3/h(k);
    b(k)=l(k)+(2*c(k)*h(k)+h(k)*c(k-1))/3;
end
end

```

Листинг 2.6

```

function z=my_spline(x,y,t)
% Вычисление значения кубического сплайна в точке t,
% здесь x,y - массивы абсцисс и ординат экспериментальных точек
[b,c,d]=koef_spline(x,y);
n=length(x);
a=y;
if t>x(n-1)
    j=n;
else
    for i=2:n-1
        if t<=x(i)
            j=i;
            break
        end
    end
end
z=a(j)+b(j)*(t-x(j))+c(j)*(t-x(j))^2+d(j)*(t-x(j))^3;
end

```

Предлагаем читателю самостоятельно написать функцию на М-языке, реализующую линейную интерполяцию.

Для реализации сплайнов в MATLAB есть встроенная функция `interp1` следующего вида:

```
yi = interp1(x,y,xi,method) ,
```

где x – массив абсцисс экспериментальных точек, y – массив ординат экспериментальных точек, xi – точки, в которых необходимо вычислить значение с помощью сплайна, `method` – определяет метод построения сплайна. Параметр `method` может принимать одно из следующих значений:

- 'linear' – линейная интерполяция;
- 'spline' – кубический сплайн;
- 'nearest' – интерполяция по соседним точкам – это метод построения кусочной функции, при котором значение в любой точке равно значению в ближайшей узловой точке.

Функция `interp1(x,y,xi,method)` возвращает значения интерполирующей функции в точках xi .

В листинге 2.7 представлено решение задачи 2.2 с помощью функции `interp1`.

Листинг 2.7

```

% Определяем экспериментальные значения
x=[0.43 0.48 0.55 0.62 0.7 0.75];

```

```

y=[1.63597 1.73234 1.87686 2.03345 2.22846 2.35973];
% Вычисляем количество экспериментальных точек
n=length(x);
% Определяем точки, в которых необходимо вычислить значения
t=[0.702 0.512 0.608];
% Уплотняем массив x для построения графика
xi=[x(1):0.02:x(length(x))];
% Вычисляем ожидаемые значения функции тремя способами в точках xi
yin=interp1(x,y,xi,'nearest');
yil=interp1(x,y,xi,'linear');
yis=interp1(x,y,xi,'spline');
% Вычисляем ожидаемые значения функции тремя способами в точках,
% заданных в условии
% точках x1 = 0,702, x2 = 0,512, x3 = 0,608
ytn=interp1(x,y,t,'nearest');
ytl=interp1(x,y,t,'linear');
yts=interp1(x,y,t,'spline');
% Построение графика
plot(x,y,'bo', xi,yin,'-r',xi,yil,'-r',xi,yis,'-r',

```

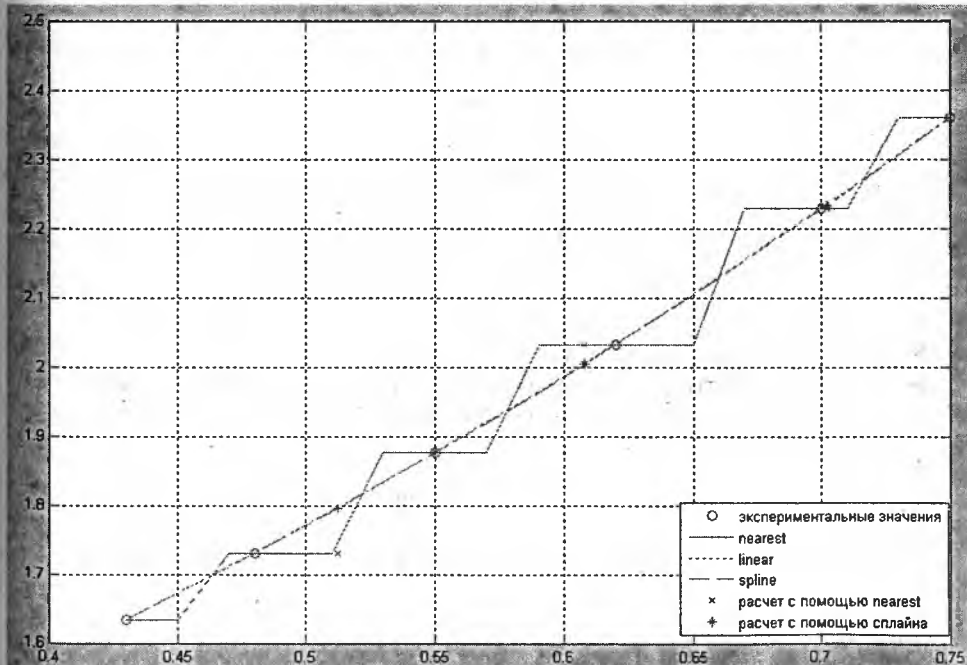


Рис. 2.14 ▼ График решения задачи 2.2 с помощью функции interp1

```
t,ytn,'x',t,yts,'*');
grid on
legend("экспериментальные значения",'nearest','linear','spline',
"расчет с помощью nearest", "расчет с помощью сплайна" );
```

На рис. 2.14 представлен график решения задачи 2.2 с помощью функции `interp1`.

2.4. Решение задачи интерполяции средствами Maple

В Maple 9 есть встроенные функции как для построения полинома n -й степени, так и для сплайн-интерполяции.

Для построения интерполяционного полинома в Maple предусмотрена функция следующего вида:

```
interp(X,Y,x) , .
```

Здесь X – массив абсцисс экспериментальных точек, Y – массив ординат экспериментальных точек, x – имя переменной интерполирования или координата точки, в которой вычисляется значение функции.

Функция `interp` может возвращать полином (если в качестве третьего параметра указано имя переменной) или вычислять значение в точке (если третьим параметром является координата точки). Использование функции `interp` рассмотрим на примере следующей задачи.

ЗАДАЧА 2.3. Найти приближенное значение функции при заданном значении аргумента с помощью интерполяции в точках $x_1 = 1$, $x_2 = 1,5$, $x_3 = 2$. Функция задана таблично (табл. 2.4). Изобразить график построенной интерполяционной зависимости, экспериментальные и рассчитанные точки на экране дисплея.

Таблица 2.4 ▼ Экспериментальная зависимость $z(t)$

t	0.66	0.9	1.17	1.47	1.7	1.74	2.08	2.63	3.12
z	38.9	68.8	64.4	66.5	64.95	59.36	82.6	90.63	113.5

Решение задачи 2.3 с использованием функции `interp` и необходимые комментарии представлены на рис. 2.15.

```

> t:=[0.66,0.9,1.17,1.47,1.7,1.74,2.08,2.63,3.12];#Массив абсцисс экспериментальных точек
    t:=[0.66,0.9,1.17,1.47,1.7,1.74,2.08,2.63,3.12]
> z:=[38.9,68.8,64.4,66.5,64.95,59.36,82.6,90.63,113.5];#Массив ординат экспериментальных точек
    z:=[38.9,68.8,64.4,66.5,64.95,59.36,82.6,90.63,113.5]
> x:=[1.0,2.0,3.0];#Массив точек, в которых необходимо вычислить ожидаемое значение
    x:=[1.0,2.0,3.0]
> interp(t,z,y);#Выводим на экран интерполяционный полином 8-й степени
2396.985478y8 - 32720.40994y7 + 66355.05379 + 1.892608631 105y6 - 3.980310774 105y5 - 6.052905418 105y4 + 1.005112979 106y3
    + 1.169561907 106y2 - 1.396562534 106y
> z:=y->interp(t,z,y);# Формируем функцию Z(y) интерполяционного полинома
    Z:=y->interp(t,z,y)
> for i from 1 to 3 do X[i]:=Z(x[i]); end do;#С помощью функции Z(y) вычисляем значения в точках 1, 2, 3
    X1:=83.22474236
    X2:=48.76122053
    X3:= -1742.604136
    X
> #Обратите внимание, что для вычисления полинома в точке x=3 интерполяционный полином 8 степени использовать
    нерационально, для этого расчета лучше воспользоваться сплайн-интерполяцией.

```

Рис. 2.15 ▼ Решение задачи 2.3 в Maple с помощью функции interp

На рис. 2.16 представлен график построенного интерполяционного полинома.

```

> l := [[ t[n], z[n]] $n=1..9];
plot([l,Z(y)],y=0.66..3.12, style=[point,line],symbol=circle);
l:=[[0.66, 38.9], [0.9, 68.8], [1.17, 64.4], [1.47, 66.5], [1.7, 64.95], [1.74, 59.36], [2.08, 82.6], [2.63, 90.63], [3.12, 113.5]]

```

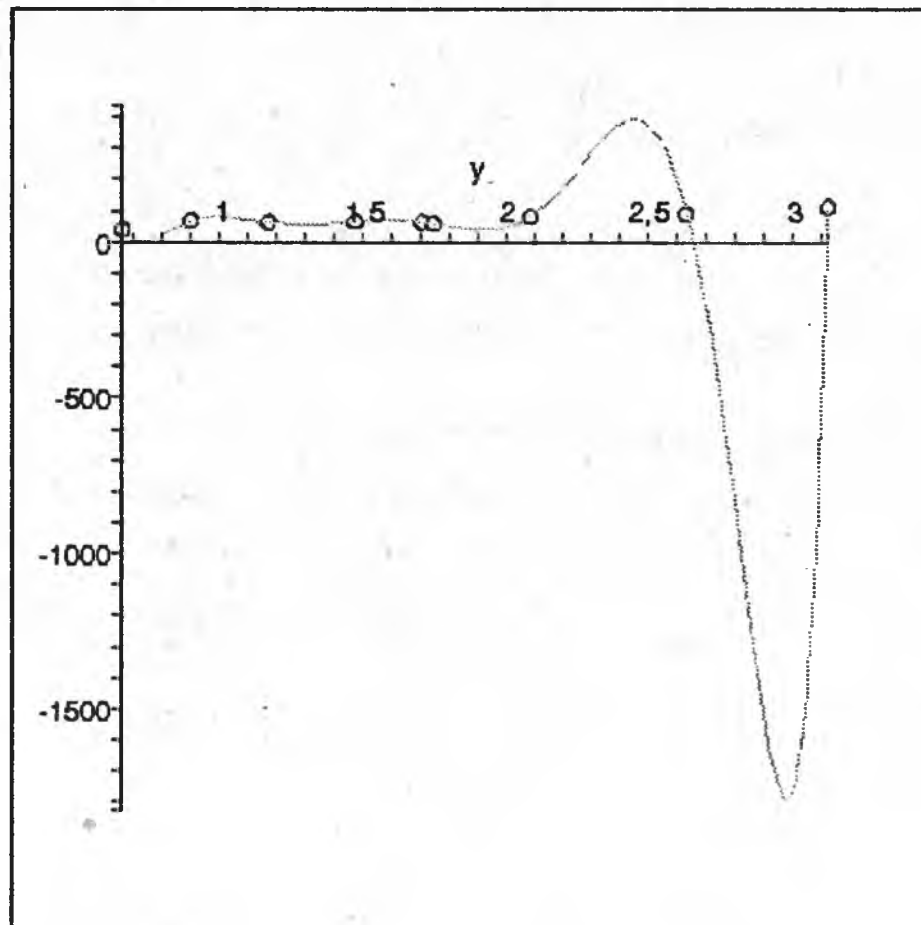


Рис. 2.16 ▼ График интерполяционного полинома задачи 2.3



Еще раз обращаем внимание читателя на то, что полиномиальная интерполяция не всегда дает удовлетворительные результаты, большая погрешность может возникать на концах интерполяционных кривых (см. рис. 2.16).

В связи с большими погрешностями интерполирования зависимости $z(t)$ (табл. 2.5) на интервале $[2; 3]$ рассмотрим решение задачи 2.3 с помощью сплайн-интерполяции.

В Maple для решения задачи сплайн-интерполяции используется функция `spline(X, Y, x, method)`. Параметры X , Y , x имеют тот же смысл, что и в функции `interp`, параметр `method` определяет метод сплайн-интерполяции. В качестве этого параметра могут использоваться ключевые слова `linear`, `quadratic`, `cubic`, `quartic` (или числа от 1 до 4), определяющие вид сплайна: линейный, параболический, кубический или полином 4-й степени. Если параметр `method` пропущен, то интерполяция будет осуществляться с помощью кубического сплайна.

На рис. 2.17 представлено решение задачи 2.3 с помощью кубического и линейного сплайнов. А на рис. 2.18 показаны графики полученных зависимостей.

```
> spline(t,z,y); # Выводим на экран кубический сплайн
      -67.15581550 + 160.6906295000000000 y - 626.8627813000000052 (y - 0.66)3           y < 0.9
      21.66813322 + 52.36874087000000035 y - 451.341202623959191 (y - 0.9)2 + 729.7268523000000018 (y - 0.9)3   y < 1.17
      101.5641677 - 31.76424593000000014 y + 139.737547690512342 (y - 1.17)2 - 35.07798177000000010 (y - 1.17)3   y < 1.47
      3.86737543 + 42.60722760000000018 y + 108.167364137616346 (y - 1.47)2 - 1403.116291000000005 (y - 1.47)3   y < 1.7
      286.4775783 - 130.31034020000000013 y - 859.982876491421962 (y - 1.7)2 + 15599.78456999999998 (y - 1.7)3   y < 1.74
      275.5202075 - 124.23000430000000004 y + 1011.99127134290143 (y - 1.74)2 - 1310.502480000000011 (y - 1.74)3   y < 2.08
      -145.0389442 + 109.4418000999999994 y - 324.721257826318378 (y - 2.08)2 + 276.8756748999999978 (y - 2.08)3   y < 2.63
      81.39056938 + 3.513091489999999987 y + 132.123605771382273 (y - 2.63)2 - 89.88000397000000042 (y - 2.63)3   otherwise

> ZS:=y->spline(t,z,y); # формируем функцию ZS(y) кубического сплайна
      ZS:=y -> spline(t,z,y)

> for i from 1 to 3 do XS[i]:=ZS(x[i]); end do; # С помощью функции ZS(y) вычисляем значения в точках 1, 2, 3
      XS1 = 70.25318891
      XS2 = 72.43741723
      XS3 = 105.4648737

> ZSL:=y->spline(t,z,y,1); # формируем функцию ZSL(y) линейной интерполяцией
      ZSL:=y -> spline(t,z,y,1)
```

Рис. 2.17 ▼ Решение задачи 2.3 в Maple с помощью функции `spline`

```

> l := [[ t[n], z[n]] $n=1..9];
plot([l, ZS(y), ZSL(y)], y=0.66..3.12, style=[point, line, line], symbol=circle, color=[black, red, blue]);#
строим графики сплайнов
l := [[0.66, 38.9], [0.9, 68.8], [1.17, 64.4], [1.47, 66.5], [1.7, 64.95], [1.74, 59.36], [2.08, 82.6], [2.63, 90.63], [3.12, 113.5]]

```

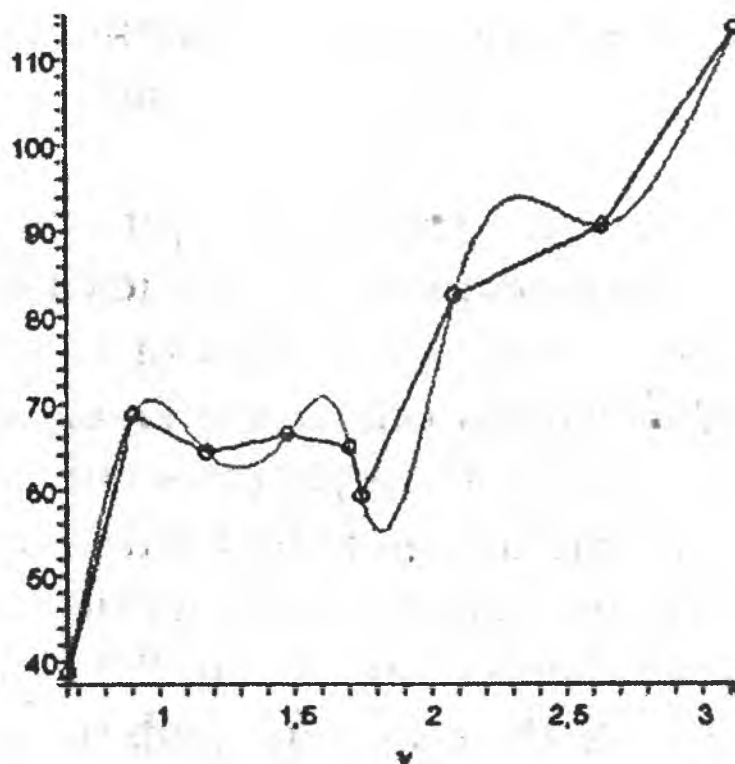


Рис. 2.18 ▼ Графики сплайнов для задачи 2.3

2.5. Советы по использованию математических пакетов при решении задач интерполяции

Все три пакета обладают хорошими возможностями для решения задач интерполяции. Для решения практических инженерных и экономических задач можно использовать любой из пакетов.

В Mathcad и MATLAB отсутствуют встроенные функции для построения интерполяционного полинома n -й степени, однако средств пакетов достаточно для реализации полиномов Ньютона, Лагранжа и канонического (см. рис. 2.8–2.10, листинги 2.1–2.3). Все пакеты обладают встроенными функциями для построения сплайн-полиномов.

На взгляд авторов, самые широкие возможности предоставляет пакет Maple: четыре вида сплайнов, возможность полинома n -й степени. Кроме того, в Maple есть возможность увидеть построенные полиномы в алгебраическом виде (см. рис. 2.15, 2.17).

Глава 3

Метод наименьших квадратов

В этой главе рассмотрен еще один способ обработки экспериментальных данных – *метод наименьших квадратов*. Первый раздел посвящен описанию решения задач обработки экспериментальных данных методом наименьших квадратов. В последующих разделах описаны возможности пакетов Mathcad, MATLAB и Maple для решения задач аппроксимации.

3.1. Подбор параметров экспериментальной зависимости методом наименьших квадратов

Пусть в результате эксперимента были получены некоторые данные, представленные в виде таблицы (табл. 3.1).

Таблица 3.1 ▼ Экспериментальные данные

x_1	x_1	x_2	x_3	x_4	x_5	x_6	x_7	...	x_n
y_1	y_1	y_2	y_3	y_4	y_5	y_6	y_7	...	y_n

Если при решении задачи интерполяции подбираемая кривая проходит через экспериментальные точки, то в методе наименьших квадратов необходимо построить аналитическую зависимость $f(x, a_0, a_1, \dots, a_k)$, наиболее близко проходящую к экспериментальным точкам¹.

¹ Подобранную методом наименьших квадратов зависимость принято называть аппроксимирующей. Синонимом термина *метод наименьших квадратов* является аппроксимация данных.

Задача подбора экспериментальной зависимости методом наименьших квадратов состоит из двух этапов: на первом этапе по экспериментальным данным выбирается вид зависимости (прямая, парабола, экспонента и т.д.), на втором – подбираются параметры выбранной зависимости.

Идея метода наименьших квадратов [9] заключается в том, что функцию

$$Y = f(x, a_0, a_1, \dots, a_k)$$

необходимо подобрать таким образом, чтобы сумма квадратов отклонений экспериментальных значений y_i от расчетных $Y_i = f(x_i, a_0, a_1, \dots, a_k)$ была наименьшей (рис. 3.1):

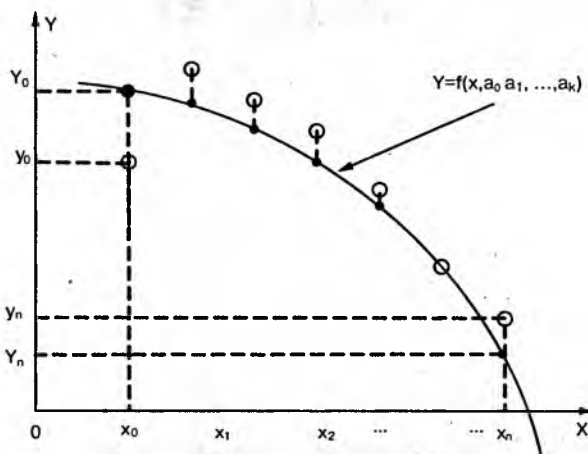


Рис. 3.1 ▼ Геометрическая интерпретация метода наименьших квадратов

$$S(a_0, a_1, \dots, a_k) = \sum_{i=1}^n (y_i - f(x_i, a_0, a_1, \dots, a_k))^2 \rightarrow \min. \quad (3.1)$$

Задача сводится к определению коэффициентов a_i из условия (3.1). Необходимо найти значения $a_0, a_1, \dots, a_k$, при которых функция $S(a_0, a_1, a_2, \dots, a_k)$ достигает минимального значения. Из условия экстремума функции многих переменных следует, что задача поиска минимума функции $S(a_0, a_1, a_2, \dots, a_k)$ эквивалентна решению системы уравнений:

$$\left. \begin{aligned} \frac{\partial S}{\partial a_0} &= 0, \\ \frac{\partial S}{\partial a_1} &= 0, \\ \dots\dots\dots \\ \frac{\partial S}{\partial a_k} &= 0. \end{aligned} \right\} \quad (3.2)$$

Если параметры a_i входят в зависимость $Y = f(x, a_0, a_1, \dots, a_k)$ линейно, то система уравнений является системой из $k+1$ линейного уравнения с $k+1$ неизвестными:

$$\left. \begin{aligned} \sum_{i=1}^n -2(y_i - f(x_i, a_0, a_1, \dots, a_k)) \frac{\partial f}{\partial a_0} &= 0, \\ \sum_{i=1}^n -2(y_i - f(x_i, a_0, a_1, \dots, a_k)) \frac{\partial f}{\partial a_1} &= 0, \\ \dots\dots\dots \\ \sum_{i=1}^n -2(y_i - f(x_i, a_0, a_1, \dots, a_k)) \frac{\partial f}{\partial a_k} &= 0. \end{aligned} \right\} \quad (3.3)$$

Решив систему уравнений (3.3), найдем коэффициенты a_i подбираемой зависимости $Y = f(x, a_0, a_1, \dots, a_k)$. В методе наименьших квадратов стараются преобразовать исходную функцию $f(x, a_0, a_1, \dots, a_k)$ таким образом, чтобы получаемая система (3.3) была системой линейных алгебраических уравнений.

Рассмотрим подробнее, как подбирать параметры наиболее часто используемых функций.

3.1.1. Подбор коэффициентов линейной зависимости

Для подбора параметров линейной функции

$$Y = a_0 + a_1 x$$

составим функцию (3.1) для линейной зависимости:

$$S(a_0, a_1) = \sum_{i=1}^n (y_i - a_0 - a_1 x_i)^2.$$

Продифференцировав функцию S по a_0 и a_1 , получим систему уравнений:

$$\left. \begin{aligned} 2 \cdot \sum_{i=1}^n (y_i - a_0 - a_1 x_i) \cdot (-1) &= 0, \\ 2 \cdot \sum_{i=1}^n (y_i - a_0 - a_1 x_i) \cdot (-x_i) &= 0 \end{aligned} \right\} \Rightarrow \left. \begin{aligned} a_0 \cdot n + a_1 \sum_{i=1}^n x_i &= \sum_{i=1}^n y_i, \\ a_0 \sum_{i=1}^n x_i + a_1 \sum_{i=1}^n x_i^2 &= \sum_{i=1}^n y_i x_i \end{aligned} \right\},$$

решив которую определим коэффициенты функции $Y = a_0 + a_1 x$:

$$\begin{cases} a_0 = \frac{\sum_{i=1}^n y_i}{n} - a_1 \frac{\sum_{i=1}^n x_i}{n}, \\ a_1 = \frac{n \sum_{i=1}^n y_i x_i - \sum_{i=1}^n y_i \sum_{i=1}^n x_i}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2}. \end{cases} \quad (3.4)$$

3.1.2. Подбор коэффициентов полинома k-й степени

Для определения параметров зависимости $Y = a_0 + a_1 \cdot x + a_2 \cdot x^2$ составим функцию для нее (3.1):

$$S(a_0, a_1, a_2) = \sum_{i=1}^n \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 \right)^2.$$

После дифференцирования S по a_0 , a_1 и a_2 получим систему уравнений:

$$\begin{cases} n a_0 + a_1 \sum_{i=1}^n x_i + a_2 \sum_{i=1}^n x_i^2 = \sum_{i=1}^n y_i, \\ a_0 \sum_{i=1}^n x_i + a_1 \sum_{i=1}^n x_i^2 + a_2 \sum_{i=1}^n x_i^3 = \sum_{i=1}^n y_i x_i, \\ a_0 \sum_{i=1}^n x_i^2 + a_1 \sum_{i=1}^n x_i^3 + a_2 \sum_{i=1}^n x_i^4 = \sum_{i=1}^n y_i x_i^2. \end{cases} \quad (3.5)$$

Решив систему (3.5) одним из методов, описанных в первой главе, найдем значение параметров a_0 , a_1 , a_2 .

Аналогично определим параметры многочлена третьей степени:

$$Y = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

Составим функцию $S(a_0, a_1, a_2, a_3)$:

$$S(a_0, a_1, a_2, a_3) = \sum_{i=1}^n \left(y_i - a_0 - a_1 x_i - a_2 x_i^2 - a_3 x_i^3 \right)^2.$$

После дифференцирования S по a_0 , a_1 , a_2 и a_3 система уравнений для вычисления параметров a_0 , a_1 , a_2 , a_3 примет вид:

Составим функцию $S(A, b, c)$ по формуле (3.1):

$$S(A, b, c) = \sum_{i=1}^n (Y_i - A - b \ln(x_i) - cx_i)^2$$

После дифференцирования получим систему трех линейных алгебраических уравнений для определения коэффициентов A , b и c .

$$\left. \begin{aligned} nA + b \sum_{i=1}^n \ln(x_i) + c \sum_{i=1}^n x_i &= \sum_{i=1}^n y_i, \\ A \sum_{i=1}^n \ln(x_i) + b \sum_{i=1}^n (\ln(x_i))^2 + c \sum_{i=1}^n x_i \ln(x_i) &= \sum_{i=1}^n y_i \ln(x_i), \\ A \sum_{i=1}^n x_i + b \sum_{i=1}^n x_i \ln(x_i) + c \sum_{i=1}^n x_i^2 &= \sum_{i=1}^n y_i x_i. \end{aligned} \right\} \quad (3.8)$$

После решения системы (3.8) необходимо вычислить значение коэффициента a по формуле $a = e^A$.

3.1.4. Функции, приводимые к линейной

Для вычисления параметров функции $Y = ax^b$ необходимо предварительно ее прологарифмировать:

$$\ln(y) = \ln(ax^b) = \ln(a) + b \ln(x),$$

после чего сделаем замену:

$$Y = \ln(y), X = \ln(x), A = \ln(a),$$

которая приведет заданную функцию к линейному виду $Y = bX + A$, где коэффициенты A и b вычисляются по формулам (3.4) и, соответственно, $a = e^A$.

Аналогично можно подобрать параметры функции вида $Y = ae^{bx}$. Прологарифмируем заданную функцию:

$$\ln(y) = \ln(a) + bx \cdot \ln(e) \Rightarrow \ln(y) = \ln(a) + bx.$$

Проведем замену $Y = \ln(y)$, $A = \ln(a)$ и получим линейную зависимость $Y = bx + A$. По формулам (3.4) найдем A и b , а затем вычислим $a = e^A$.

Рассмотрим еще ряд зависимостей, которые сводятся к линейной.

Для подбора параметров функции $y = \frac{1}{ax + b}$ сделаем замену $z = \frac{1}{y}$. В результате получим линейную зависимость $z = ax + b$. Функция $y = \frac{x}{ax + b}$ заменим

$Z = \frac{1}{y}, X = \frac{1}{x}$ сводится к линейной $Z = a + bX$. Для определения коэффициентов

функциональной зависимости $Y = \frac{1}{a e^{-x} + b}$ необходимо сделать следующие замечания: $Z = \frac{1}{Y}$, $X = e^{-x}$. В результате также получим линейную функцию $Z = aX + b$.

Аналогичными приемами (логарифмированием, заменами и т.п.) можно многие подбираемые зависимости преобразовать к такому виду, чтобы получаемая при решении задачи оптимизации система (3.3) была системой линейных алгебраических уравнений.

3.1.5. Уравнение регрессии и коэффициент корреляции

Линия, описываемая уравнением вида $y = a_0 + a_1 x$, называется *линией регрессии* y на x , параметры a_0 и a_1 называются *коэффициентами регрессии* и определяются формулами (3.4).

Чем меньше величина

$$S = \sum_{i=1}^n (y_i - a_1 x_i - a_0)^2,$$

тем более обоснованно предположение, что экспериментальные данные описываются линейной функцией. Существует показатель, характеризующий тесноту линейной связи между x и y , который называется *коэффициентом корреляции* и рассчитывается по формуле:

$$r = \frac{\sum_{i=1}^n (x_i - M_x) (y_i - M_y)}{\sqrt{\sum_{i=1}^n (x_i - M_x)^2 \sum_{i=1}^n (y_i - M_y)^2}}, \quad M_x = \frac{\sum_{i=1}^n x_i}{n}, \quad M_y = \frac{\sum_{i=1}^n y_i}{n}. \quad (3.9)$$

Значение коэффициента корреляции удовлетворяет соотношению:

$$-1 \leq r \leq 1.$$

Чем меньше отличается абсолютная величина r от единицы, тем ближе к линии регрессии располагаются экспериментальные точки. Если $|r| = 1$, то все экспериментальные точки находятся на линии регрессии. Если коэффициент корреляции близок к нулю, то это означает, что между x и y не существует линейной связи, но между ними может существовать зависимость, отличная от линейной.

Для того чтобы проверить, значимо ли отличается от нуля коэффициент корреляции, можно использовать *критерий Стьюдента*. Вычисленное значение критерия определяется по формуле:

$$t = r \sqrt{\frac{n-2}{1-r^2}}. \quad (3.10)$$

Рассчитанное по формуле (3.10) значение t сравнивается со значением, взятым из таблицы распределения Стьюдента (табл. 3.2) в соответствии с уровнем значимости p (стандартное значение $p = 0.95$) и числом степеней свободы $k = n - 1$. Если полученная по формуле (3.10) величина t больше табличного значения, то коэффициент корреляции значимо отличен от нуля.

Таблица 3.2 ▼ Распределение Стьюдента

$k \backslash p$	0,99	0,98	0,95	0,90	0,80	0,70	0,60
1	63,657	31,821	12,706	6,314	3,078	1,963	1,376
2	9,925	6,965	4,303	2,920	1,886	1,386	1,061
3	5,841	4,541	3,182	2,353	1,638	1,250	0,978
4	4,604	3,747	2,776	2,132	1,533	1,190	0,941
5	4,032	3,365	2,571	2,05	1,476	1,156	0,920
6	3,707	3,141	2,447	1,943	1,440	1,134	0,906
7	3,499	2,998	2,365	1,895	1,415	1,119	0,896
8	3,355	2,896	2,306	1,860	1,387	1,108	0,889
9	3,250	2,821	2,261	1,833	1,383	1,100	0,883
10	3,169	2,764	2,228	1,812	1,372	1,093	0,879
11	3,106	2,718	2,201	1,796	1,363	1,088	0,876
12	3,055	2,681	2,179	1,782	1,356	1,083	0,873
13	3,012	2,650	2,160	1,771	1,350	1,079	0,870
14	2,977	2,624	2,145	1,761	1,345	1,076	0,868
15	2,947	2,602	2,131	1,753	1,341	1,074	0,866
16	2,921	2,583	2,120	1,746	1,337	1,071	0,865
17	2,898	2,567	2,110	1,740	1,333	1,069	0,863
18	2,878	2,552	2,101	1,734	1,330	1,067	0,862
19	2,861	2,539	2,093	1,729	1,328	1,066	0,861
20	2,845	2,528	2,086	1,725	1,325	1,064	0,860
21	2,831	2,518	2,080	1,721	1,323	1,063	0,859
22	2,819	2,508	2,074	1,717	1,321	1,061	0,858
23	2,807	2,500	2,069	1,714	1,319	1,060	0,858
24	2,797	2,492	2,064	1,711	1,318	1,059	0,857
25	2,779	2,485	2,060	1,708	1,316	1,058	0,856
26	2,771	2,479	2,056	1,706	1,315	1,058	0,856
27	2,763	2,473	2,052	1,703	1,314	1,057	0,855
28	2,756	2,467	2,048	1,701	1,313	1,056	0,855
29	2,750	2,462	2,045	1,699	1,311	1,055	0,854
30	2,704	2,457	2,042	1,697	1,310	1,055	0,854
40	2,660	2,423	2,021	1,684	1,303	1,050	0,851
60	2,612	2,390	2,000	1,671	1,296	1,046	0,848
120	2,617	2,358	1,980	1,658	1,289	1,041	0,845
∞	2,576	2,326	1,960	1,645	1,282	1,036	0,842

Кроме того, таблицу распределений Стьюдента можно использовать для построения доверительного интервала, накрывающего изучаемую величину. Оценка в этом случае выполняется по формуле:

$$\delta = \sqrt{\sum_{i=1}^n (y_i - M_y)^2 \cdot \frac{1 - r^2}{n - 2} \left(1 + \frac{(\tau - M_x)^2}{\sum_{i=1}^n (x_i - M_x)^2} \right)}, \quad (3.11)$$

где τ – значение, для которого необходимо найти доверительный интервал, используя метод наименьших квадратов. Доверительным для τ будет интервал:

$$[\tau - \gamma \cdot \delta; \tau + \gamma \cdot \delta]$$

где γ – значение, определенное по таблице Стьюдента при уровне значимости p и с числом степеней свободы $k = n - 2$.

3.1.6. Криволинейная корреляция

Обычно коэффициент корреляции r применяется только в тех случаях, когда между данными существует прямолинейная связь. Если же связь криволинейная, то пользуются *индексом корреляции*, который рассчитывается по формуле:

$$R = \sqrt{1 - \frac{\sum_{i=1}^n (y_i - Y_i)^2}{\sum_{i=1}^n (y_i - M_y)^2}}, \quad (3.12)$$

где y – экспериментальные значения, Y – теоретические значения (рассчитанные по подобранной методом наименьших квадратов формуле), M_y – среднее значение y .

Индекс корреляции по своему абсолютному значению колеблется в пределах от 0 до 1. При наличии функциональной зависимости индекс корреляции близок к 1. При отсутствии связи R практически равен нулю. Если коэффициент корреляции r является мерой тесноты связи только для линейной формы связи, то индекс корреляции R – и для линейной, и для криволинейной. При прямолинейной связи коэффициент корреляции по своей абсолютной величине равен индексу корреляции: $|r| = R$.

3.2. Решение задач аппроксимации в Mathcad

Изучение возможностей Mathcad для обработки экспериментальных данных начнем с простейшей задачи подбора линейной зависимости методом наименьших квадратов.

ЗАДАЧА 3.1. В «Основах химии» Д. И. Менделеева приводятся данные о растворимости азотнокислого натрия NaNO_3 в зависимости от температуры воды. В 100 частях воды (табл. 3.3) растворяется следующее число условных частей NaNO_3 при соответствующих температурах:

Таблица 3.3 ▽ Данные о растворимости NaNO_3 в зависимости от температуры воды

0°	4°	10°	15°	21°	29°	36°	51°	68°
66,7	71,0	76,3	80,6	85,7	92,9	99,4	113,6	125,1

Необходимо определить, какова будет растворимость азотнокислого натрия при температуре $t = 32^\circ$ в случае линейной зависимости, и вычислить коэффициент корреляции.

Для определения коэффициентов регрессии a_0, a_1 в Mathcad существуют следующие функции:

- $\text{line}(x, y)$ возвращает массив $a = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix}$ коэффициентов регрессии;
- $\text{intercept}(x, y)$ возвращает коэффициент регрессии a_0 ;
- $\text{slope}(x, y)$ возвращает коэффициент регрессии a_1 .

Для вычисления коэффициента корреляции в Mathcad предназначена функция $\text{corr}(x, y)$.

В этих функциях x – массив абсцисс экспериментальных точек, y – массив ординат экспериментальных точек.

На рис. 3.2 приведено решение задачи 3.1 в Mathcad с комментариями.

Построение нелинейной зависимости рассмотрим на примере решения следующей задачи.

ЗАДАЧА 3.2. Известна табличная зависимость y от x (табл. 3.4).

Таблица 3.4 ▽ Табличная зависимость y от x

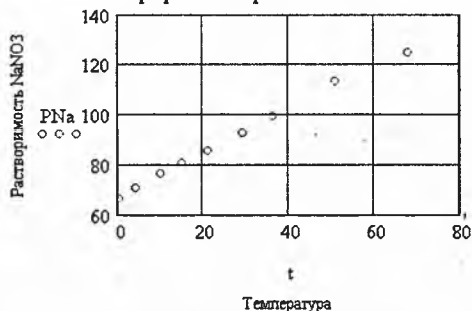
x	10.1	10.2	10.3	10.8	10.9	11	11.1	11.4	12.2	13.3	13.8	14	14.4	14.5	15	15.6	15.8	17	18.1	19
y	24	36	26	45	34	37	55	51	75	84	74	91	85	87	94	92	96	97	98	99

АВА 3 ▽ Метод наименьших квадратов

$$t := (0 \ 4 \ 10 \ 15 \ 21 \ 29 \ 36 \ 51 \ 68)^T$$

$$PNa := (66.7 \ 71 \ 76.3 \ 80.6 \ 85.7 \ 92.9 \ 99.4 \ 113.6 \ 125.1)^T$$

График экспериментальных точек



Коэффициент корреляции

$$\text{corr}(t, PNa) = 0.999$$

Коэффициенты регрессии

$$a := \text{line}(t, PNa)$$

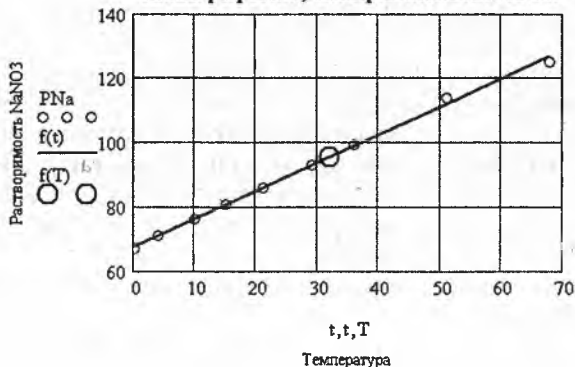
$$a = \begin{pmatrix} 67.508 \\ 0.871 \end{pmatrix}$$

Линия регрессии

$$f(y) := a_0 + a_1 \cdot y$$

$$f(T) = 95.368$$

Линия регрессии, экспериментальные точки



Построение линии регрессии и вычисление коэффициента

Аппроксимировать эту зависимость методом наименьших квадратов с помощью функций $y = a_0 + a_1x + a_2x^2$, $y = b_0 + b_1x + b_2x^2 + b_3x^3$ и $y = c_0 + c_1x^2 + c_2x^3$ и сравнить результаты полученных вычислений.

На рис. 3.3. изображен график табличной зависимости y от x .

$x := (10.1 \ 10.2 \ 10.3 \ 10.8 \ 10.9 \ 11 \ 11.1 \ 11.4 \ 12.2 \ 13.3 \ 13.8 \ 14 \ 14.4 \ 14.5 \ 15 \ 15.6 \ 15.8 \ 17 \ 18.1 \ 19)^T$
 $y := (24 \ 36 \ 26 \ 45 \ 34 \ 37 \ 55 \ 51 \ 75 \ 84 \ 74 \ 91 \ 85 \ 87 \ 94 \ 92 \ 96 \ 97 \ 98 \ 99)^T$



Рис. 3.3 ▽ График экспериментальной зависимости

Для построения квадратичной и кубической аппроксимирующей зависимости в Mathcad можно воспользоваться функциями `regress` и `interp`.

Функция `regress(x,y,k)` возвращает вектор коэффициентов полинома k -й степени, подобранного методом наименьших квадратов по экспериментальным точкам (x – массив абсцисс, y – массив ординат экспериментальных точек). Элементы массива x должны быть упорядочены по возрастанию. После определения коэффициентов полинома для вычисления значения полинома в конкретных точках можно воспользоваться функцией `interp`. Функция `interp(s,x,y,t)` вычисляет значения полинома в точке t , x – массив абсцисс, y – массив ординат экспериментальных точек, s – массив коэффициентов полинома, найденный с помощью функции `regress`.

С помощью функций `regress` и `interp` можно подобрать коэффициенты полного полинома любой степени.



Функция `regress` возвращает специальным образом сформированный массив, предназначенный для использования в функции `interp`, первые три элемента которого являются специальными значениями, используемыми функцией `interp`, а последующие элементы массива – коэффициентами подобранного полинома.

Для аппроксимации табличной зависимости $y(x)$ полиномами второй и третьей степени необходимо сформировать массивы коэффициентов полиномов a и b , обратившись к функции `regress`. Далее можно, используя `interp`, построить функции $A(t)$ и $B(t)$, с помощью которых можно вычислить значения аппроксимирующих полиномов второй и третьей степени в любой точке t (рис. 3.4).

Подобрать зависимость $y = c_0 + c_1x^2 + c_2x^3$ с помощью функций `regress` и `interp` не получится. Поэтому вспомним, что аппроксимирующую зависимость необходимо подбирать таким образом, чтобы сумма квадратов отклонений измеренных

В массив a запишем коэффициенты аппроксимирующего полинома второй степени

$$a := \text{regress}(x, y, 2) \quad a^T = (3 \quad 3 \quad 2 \quad -319.265 \quad 48.552 \quad -1.407)$$

В массив b запишем коэффициенты аппроксимирующего полинома третьей степени

$$b := \text{regress}(x, y, 3) \quad b^T = (3 \quad 3 \quad 3 \quad -701.76 \quad 131.838 \quad -7.303 \quad 0.136)$$

Функции $A(t)$ и $B(t)$ предназначены для вычисления аппроксимирующих полиномов второй и третьей степени в любой точке t

$$A(t) := \text{interp}(a, x, y, t) \quad B(t) := \text{interp}(b, x, y, t) \quad t := 10, 10.1 \dots 19$$

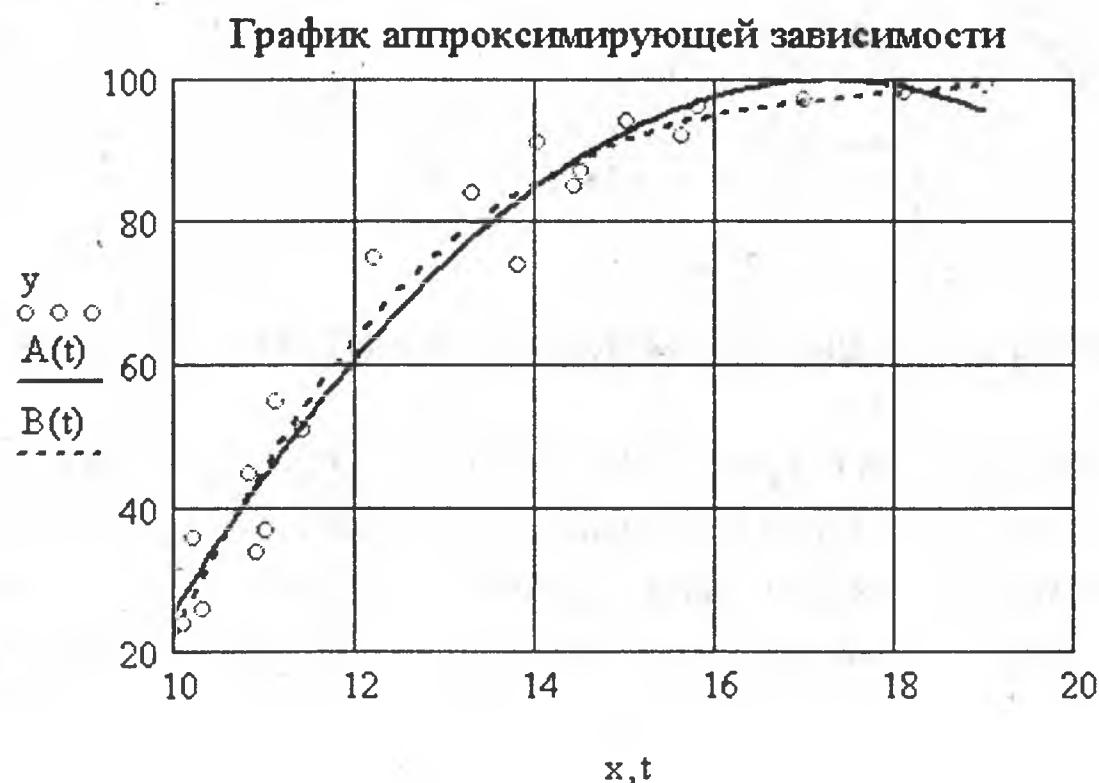


Рис. 3.4 ▼ Построение аппроксимирующих зависимостей $y = a_0 + a_1x + a_2x^2$,
 $y = b_0 + b_1x + b_2x^2 + b_3x^3$

значений от расчетных была наименьшей. В связи с этим проблема подбора зависимости $y = c_0 + c_1 x + c_2 x^3$ эквивалентна следующей задаче оптимизации. Найти значения c_0 , c_1 и c_2 , при которых функция $S(c_0, c_1, c_2) = \sum_{i=1}^n (y_i - c_0 - c_1 x_i^2 - c_2 x_i^3)^2$ достигает своего минимального значения¹. Решение этой задачи представлено на рис. 3.5.

$$n := \text{last}(x)$$

Формируем функцию $s(a, b, c) := \sum_{i=1}^n [y_i - a \cdot (x_i)^3 - b \cdot (x_i)^2 - c]^2$

Задача подбора коэффициентов зависимости $y = c_0 + c_1 x^2 + c_2 x^3$ эквивалентна поиску минимума функции s

$$\begin{matrix} a \\ b \\ c \end{matrix} := \text{Minimize}(s, a, b, c)$$

$$a = -0.077 \quad b = 1.99 \quad c = -93.729$$

$$S(t) := a \cdot t^3 + b \cdot t^2 + c$$

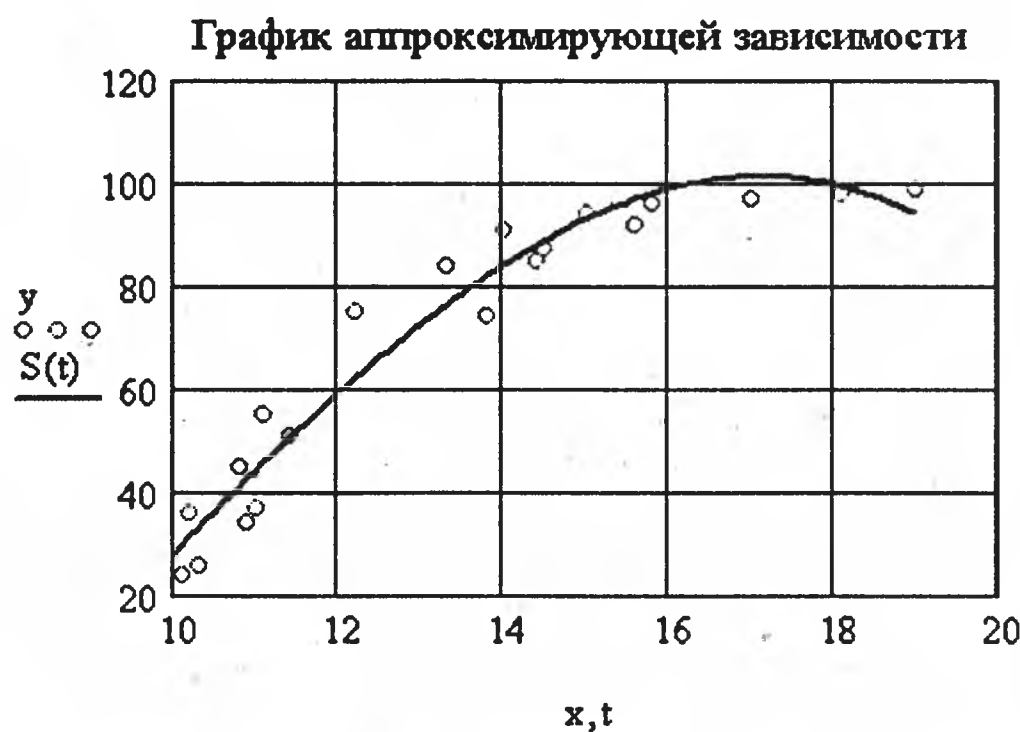


Рис. 3.5 ▼ Построение аппроксимирующей зависимости $y = c_0 + c_1 x^2 + c_2 x^3$

Таким образом, построены все три зависимости. Осталось определить, какая из них лучше описывает экспериментальные значения, что можно сделать, сравнив индексы корреляции или величины суммы квадратов отклонений.

На рис. 3.6–3.8 представлены расчеты индексов корреляции для трех зависимостей.

¹ Для решения оптимизационных задач в Mathcad используется решающий блок Given - Minimize.

Расчет индекса регрессии для квадратичной зависимости

Расчет коэффициентов квадратичной зависимости

$\underline{a} := \text{regress}(x, y, 2)$

Вычисление расчетных значений

$i := 0..n$

$Y2_i := \text{interp}(a, x, y, x_i)$

Вычисление индекса корреляции

$$\gamma_2 := \sqrt{1 - \frac{\sum_{i=0}^{\text{last}(y)} (y_i - Y2_i)^2}{\sum_{i=0}^{\text{last}(y)} (y_i - \text{mean}(y))^2}} \quad \gamma_2 = 0.977$$

Рис. 3.6 ▼ Вычисление индекса корреляции для квадратичной зависимости

Расчет индекса регрессии для кубической зависимости

Расчет коэффициентов кубической зависимости

$\underline{b} := \text{regress}(x, y, 3)$

Вычисление расчетных значений

$i := 0..n$

$Y3_i := \text{interp}(b, x, y, x_i)$

$$\gamma_3 := \sqrt{1 - \frac{\sum_{i=0}^{\text{last}(y)} (y_i - Y3_i)^2}{\sum_{i=0}^{\text{last}(y)} (y_i - \text{mean}(y))^2}} \quad \gamma_3 = 0.979$$

Рис. 3.7 ▼ Вычисление индекса корреляции для кубической зависимости

Расчет индекса регрессии для зависимости $y = c_0 + c_1x^2 + c_2x^3$

Вычисление расчетных значений

$i := 0..n$

$Y4_i := S(x_i)$

Вычисление индекса корреляции

$$\gamma_4 := \sqrt{1 - \frac{\sum_{i=0}^{\text{last}(y)} (y_i - Y4_i)^2}{\sum_{i=0}^{\text{last}(y)} (y_i - \text{mean}(y))^2}} \quad \gamma_4 = 0.973$$

Рис. 3.8 ▼ Вычисление индекса корреляции для зависимости $y = c_0 + c_1x^2 + c_2x^3$

На рис. 3.9 представлены величины сумм квадратов отклонений для всех трех зависимостей.

Сумма квадратов отклонений для квадратичной зависимости

$$S2 := \sqrt{\sum_{i=0}^n (y_i - Y2_i)^2} \quad S2 = 25.469$$

Сумма квадратов отклонений для кубической зависимости

$$S3 := \sqrt{\sum_{i=0}^n (y_i - Y3_i)^2} \quad S3 = 24.031$$

Сумма квадратов отклонений для зависимости $y = c_0 + c_1x^2 + c_2x^3$

$$S4 := \sqrt{\sum_{i=0}^n (y_i - Y4_i)^2} \quad S4 = 27.463$$

Рис. 3.9 ▽ Вычисление сумм квадратов отклонений подобранных зависимостей

Как видно из результатов, приведенных на рис. 3.6–3.9, табличная зависимость, представленная в табл. 3.2, лучше всего аппроксимируется полиномом третьей степени.

 Представленный на рис. 3.5 метод подбора зависимости является универсальным и подходит для аппроксимации экспериментальных значений любой функцией.

Кроме рассмотренных выше методов аппроксимации в Mathcad существует возможность подбора параметров приближающей функции следующего вида: $k_0F_0(x) + k_1F_1(x) + \dots + k_nF_n(x)$, где $F_i(x)$ – любые известные функции. Обращение к функции `linfit` имеет вид

`linfit (x, y, F).`

Здесь x – массив абсцисс экспериментальных точек, y – массив ординат экспериментальных точек, F – вектор, содержащий функции $F_i(x)$ в символьном виде. Функция `linfit` возвращает вектор коэффициентов K .

Решение задачи подбора табличной зависимости y от x (см. табл. 3.2) методом наименьших квадратов $y = k_0 + k_1x^2 + k_2x^3$ с помощью функции `linfit` представлено на рис. 3.10.

$$\underline{x} := (10.1 \ 10.2 \ 10.3 \ 10.8 \ 10.9 \ 11 \ 11.1 \ 11.4 \ 12.2 \ 13.3 \ 13.8 \ 14 \ 14.4 \ 14.5 \ 15 \ 15.6 \ 15.8 \ 17 \ 18.1 \ 19)^T$$

$$\underline{y} := (24 \ 36 \ 26 \ 45 \ 34 \ 37 \ 55 \ 51 \ 75 \ 84 \ 74 \ 91 \ 85 \ 87 \ 94 \ 92 \ 96 \ 97 \ 98 \ 99)^T$$

$$F(x) := \begin{pmatrix} 1 \\ x^2 \\ x^3 \end{pmatrix}$$

$$\underline{K} := \text{linfit}(\underline{x}, \underline{y}, F) \quad f(x) := \underline{K} \cdot F(x)$$

График аппроксимирующей зависимости

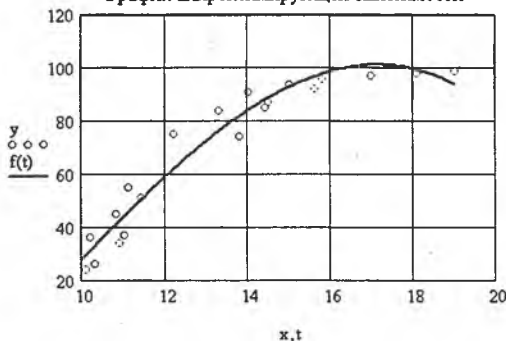


Рис. 3.10 ▼ Подбор зависимости с использованием функции linfit

Кроме описанных выше универсальных возможностей подбора аппроксимирующих зависимостей в Mathcad существуют следующие функции для подбора параметров специального вида:

- $\text{expfit}(\underline{x}, \underline{y}, g)$ – возвращает коэффициенты зависимости $a e^{bx} + c$;
- $\text{lgsfit}(\underline{x}, \underline{y}, g)$ – возвращает коэффициенты зависимости $\frac{a}{1 + b e^{-cx}}$;
- $\text{sinfit}(\underline{x}, \underline{y}, g)$ – возвращает коэффициенты зависимости $a \sin(x + b) + c$;
- $\text{pwrfit}(\underline{x}, \underline{y}, g)$ – возвращает коэффициенты зависимости $a x^b + c$;
- $\text{logfit}(\underline{x}, \underline{y}, g)$ – возвращает коэффициенты зависимости $a \lg(x + b) + c$;
- $\text{lnfit}(\underline{x}, \underline{y})$ – возвращает коэффициенты зависимости $a \lg(x) + b$.

Во всех этих функциях x – массив абсцисс экспериментальных точек, y – массив ординат экспериментальных точек, g – вектор, задающий начальное приближение параметров a , b и c .

Пример использования функции $\text{pwrfit}(\underline{x}, \underline{y}, g)$ приведен на рис. 3.11.

$$x := (10.1 \ 10.2 \ 10.3 \ 10.8 \ 10.9 \ 11 \ 11.1 \ 11.4 \ 12.2 \ 13.3 \ 13.8 \ 14 \ 14.4 \ 14.5 \ 15 \ 15.6 \ 15.8 \ 17 \ 18.1 \ 19)^T$$

$$y := (24 \ 36 \ 26 \ 45 \ 34 \ 37 \ 55 \ 51 \ 75 \ 84 \ 74 \ 91 \ 85 \ 87 \ 94 \ 92 \ 96 \ 97 \ 98 \ 99)^T$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} := \text{pwrfit} \left[x, y, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right]$$



Рис. 3.11 ▼ Пример использования функции `pwrfit(x,y,g)`

Зависимости, для которых в Mathcad не предусмотрены стандартные функции, можно решать так, как на рис. 3.5, или сформировать систему (3.3) и решить ее средствами пакета. На взгляд авторов, метод, рассмотренный на рис. 3.5, является более универсальным и может быть использован при построении любой зависимости.

3.3. Реализация метода наименьших квадратов в MATLAB

Рассмотрим решение задачи 3.2 в MATLAB.

Для подбора коэффициентов полинома k -й степени методом наименьших квадратов в MATLAB есть функция `polyfit(x,y,k)` (x – массив абсцисс экспериментальных точек, y – массив ординат экспериментальных точек, k – степень полинома). Функция возвращает массив коэффициентов полинома. Затем можно вычислить значение полинома в любой точке с помощью функции `polyval(k,t)`. В массиве k хранятся коэффициенты полинома, t – точка, в которой

необходимо вычислить значение полинома. Функция `polyval(k,t)` вычисляет значение полинома в точке t по формуле $k_1 t^n + k_2 t^{n-1} + \dots + k_n t + k_{n+1}$. С помощью функций `polyfit`, `polyval` можно подобрать зависимости $y = a_0 + a_1 x + a_2 x^2$ и $y = b_0 + b_1 x + b_2 x^2 + b_3 x^3$, что показано в листинге 3.1. На рис. 3.12 представлены графики подобранных зависимостей.

Листинг 3.1

```
x=[10.1,10.2,10.3,10.8,10.9,11,11.1,11.4,12.2,13.3,13.8,14,14.4,14.5,15,
15.6,15.8,17,18.1,19];
y=[24,36,26,45,34,37,55,51,75,84,74,91,85,87,94,92,96,97,98,99];
%Подбор коэффициентов полинома 2-й степени с помощью МНК
a=polyfit(x,y,2);
%Подбор коэффициентов полинома 3-й степени с помощью МНК
b=polyfit(x,y,3);
%Определение массива для построения графика функции
t=x(1):0.1:x(length(x));
%Вычисление значений подобранных полиномов в точках t(i).
p2t=polyval(a,t);
p3t=polyval(b,t);
%Построение графика функции
plot(x,y,'ko',t,p2t,'r-',t,p3t,'b-.');
title("Подбор зависимости методом наименьших квадратов");
legend("экспериментальные значения", "полином 2-й степени",
"полином 3-й степени");
grid on;
```

Несколько сложнее в MATLAB подобрать параметры зависимости $y = c_0 + c_1 x^2 + c_2 x^3$ методом наименьших квадратов. Эта задача эквивалентна поиску минимума функции $S(c_0, c_1, c_2) = \sum_{i=1}^n (y_i - c_0 - c_1 x_i^2 - c_2 x_i^3)^2$, которую можно решать двумя способами:

- с помощью функции `fminsearch` найти минимум функции S ;
- можно найти частные производные функции $S(c_0, c_1, c_2) = \sum_{i=1}^n (y_i - c_0 - c_1 x_i^2 - c_2 x_i^3)^2$ по c_0, c_1, c_2 , приравнять их к нулю, решить полученную систему линейных алгебраических уравнений.

Рассмотрим оба способа последовательно.

Первый способ решения. Для поиска минимума функции многих переменных в MATLAB можно использовать функцию `fminsearch(fun,x0)`. `fun` – функция, минимум которой необходимо найти, `x0` – вектор начального приближения точки

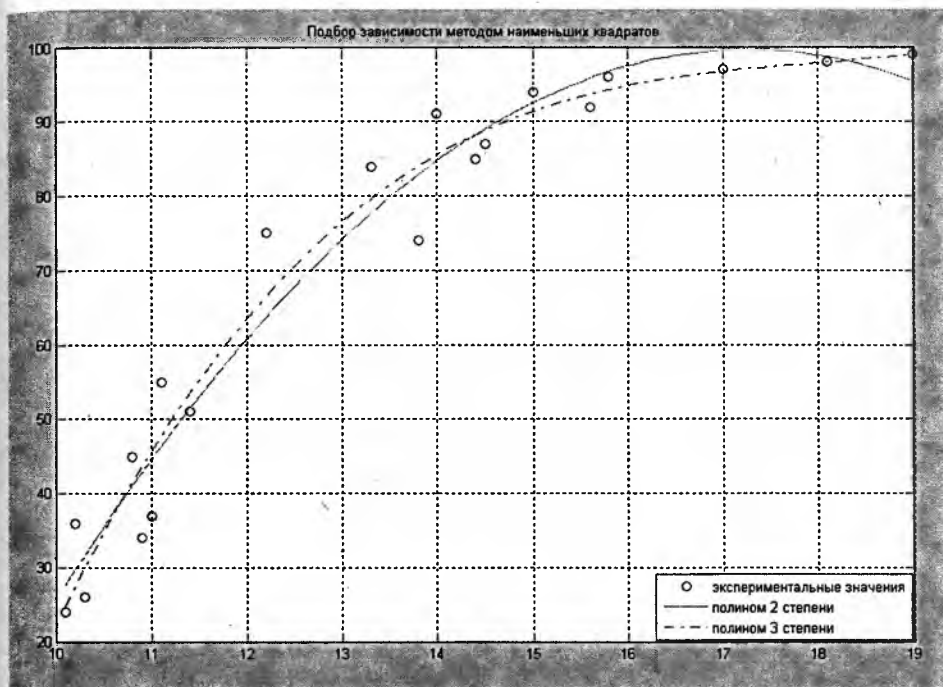


Рис. 3.12 ▽ Графики подобранных аппроксимирующих зависимостей $y = a_0 + a_1x + a_2x^2$, $y = b_0 + b_1x + b_2x^2 + b_3x^3$

минимума¹. Функция `fminsearch` возвращает массив, в котором хранятся координаты точки минимума и само значение минимума.

В нашей задаче необходимо создать М-функцию, вычисляющую значение $S(c_0, c_1, c_2)$ по формуле $\sum_{i=1}^n (y_i - c_0 - c_1 x_i^2 - c_2 x_i^3)^2$. Текст этой функции приведен в листинге 3.2.

Листинг 3.2

```
function s=f_mnk(c)
%Переменные x,y являются глобальными, используются в нескольких функциях
global x;
global y;
s=0;
```

¹ Неправильный выбор точки начального приближения может привести к некорректным результатам.

```

for i=1:length(x)
    s=s+(y(i)-c(1)-c(2)*x(i)^2-c(3)*x(i)^3)^2;
end
end

```

Затем необходимо обратиться к функции `fminsearch(fun,x0)` для поиска минимума $S(c_0, c_1, c_2)$ (листинг 3.3), после чего можно будет изобразить график подобранной функции (рис. 3.13).

Листинг 3.3

```

global x;
global y;
%Задание начального значения вектора c, при неправильном его определении
%минимум может быть найден неправильно
c=[2,1,0];
%Определение координат экспериментальных точек
x=[10.1, 10.2, 10.3,10.8, 10.9, 11, 11.1,11.4, 12.2, 13.3, 13.8, 14,
14.4,14.5, 15,15.6,15.8,17, 18.1,19];
y=[24, 36,26, 45,34,37,55, 51, 75,84,74,91,85,87,94,92,96,97,98,99];
% Поиск минимума и тем самым определение коэффициентов зависимостей
[c,err]=fminsearch("f_mnk",c)
%Определение массива для построения графика функции
t=x(1):0.1:x(length(x));
%Вычисление значений подобранных полиномов в точках t(i).
p2t=c(3)*t.^3+c(2)*t.^2+c(1);
%Построение графика функции
plot(x,y,'ko',t,p2t,'r-');
legend("экспериментальные значения","Подобранная кривая");
grid on

```

Второй способ решения начинается с формирования системы линейных алгебраических уравнений. Для этого продифференцируем функцию $S(c_0, c_1, c_2) = \sum_{i=1}^n (y_i - c_0 - c_1 x_i^2 - c_2 x_i^3)^2$ по c_0, c_1, c_2 и приравняем все частные производные к нулю.

$$\left. \begin{aligned} n c_0 + c_1 \sum_{i=1}^n x_i^2 + c_2 \sum_{i=1}^n x_i^3 &= \sum_{i=1}^n y_i, \\ c_0 \sum_{i=1}^n x_i^2 + c_1 \sum_{i=1}^n x_i^4 + c_2 \sum_{i=1}^n x_i^5 &= \sum_{i=1}^n y_i x_i^2, \\ c_0 \sum_{i=1}^n x_i^3 + c_1 \sum_{i=1}^n x_i^5 + c_2 \sum_{i=1}^n x_i^6 &= \sum_{i=1}^n y_i x_i^3. \end{aligned} \right\} \quad (3.13)$$

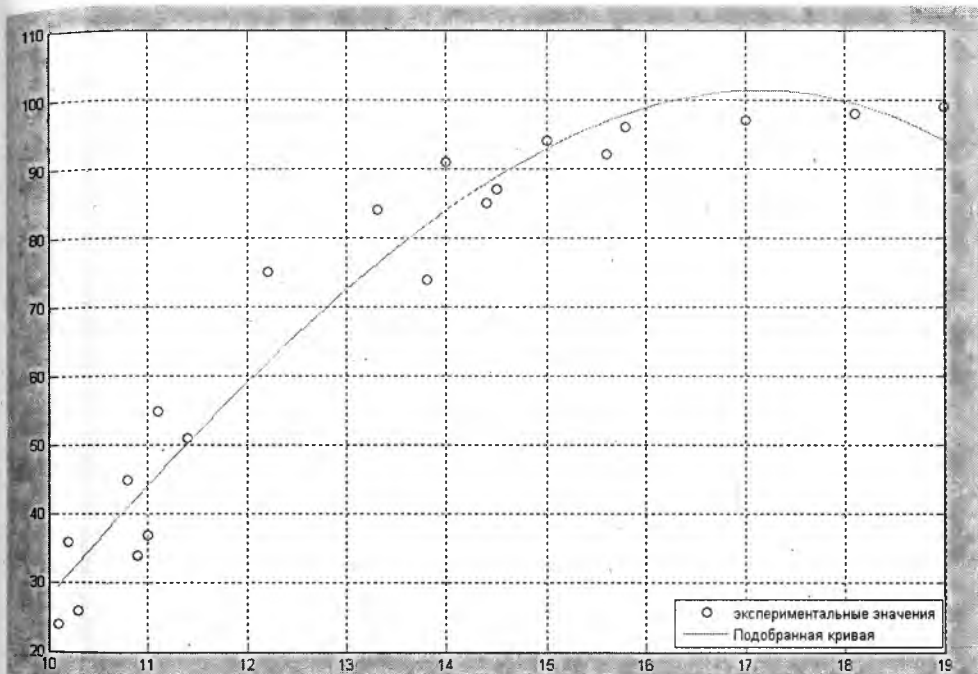


Рис. 3.13 ▾ График подобранной зависимости $y = c_0 + c_1x^2 + c_2x^3$

В листинге 3.4 приведена функция `my_mnk(x,y)`, реализующая подбор зависимости вторым способом.

Листинг 3.4

```
function c=my_mnk(x,y)
%Определение количества элементов в массивах экспериментальных точек
n=length(x);
%Накапливание сумм
sx=0; sx2=0; sx3=0; sx4=0; sx5=0; sx6=0; sy=0; syx2=0; syx3=0;
for i=1:length(x)
    sx=sx+x(i); sx2=sx2+x(i)^2; sx3=sx3+x(i)^3;
    sx4=sx4+x(i)^4; sx5=sx5+x(i)^5; sx6=sx6+x(i)^6;
    sy=sy+y(i); syx2=syx2+y(i)*x(i)^2; syx3=syx3+y(i)*x(i)^3;
end;
%Формирование матрицы коэффициентов системы (3.13)
A(1,1)=n; A(1,2)=sx2; A(1,3)=sx3;
A(2,1)=sx2; A(2,2)=sx4; A(2,3)=sx5;
A(3,1)=sx3; A(3,2)=sx5; A(3,3)=sx6;
```

```
%Формирование вектора правой части системы (3.13)
B(1)=sy;B(2)=syx2;B(3)=syx3;
% Решение системы (3.13),
% нахождение параметров подбираемой зависимости
c=A^(-1)*B'
end
```

Параметры зависимости $y = c_0 + c_1x^2 + c_2x^3$, найденные первым и вторым способом, полностью совпадают.

Параметры любой другой зависимости в MATLAB можно также реализовать аналогично: решив задачу оптимизации функции $S(a_0, a_1, \dots, a_k)$ (см. формулу (3.1)) или сформировав систему уравнений (3.3) и решив ее в MATLAB. Желательно приводить подбираемую функцию к такому виду, чтобы система (3.3) была системой линейных алгебраических уравнений¹.

3.4. Решение задач аппроксимации средствами Maple

Для решения задачи аппроксимации в Maple есть функция `LeastSquares` из пакета `CurveFitting` следующей структуры:

```
LeastSquares(x, y, v, curve=function)
```

или

```
LeastSquares(L, v, curve=function)
```

Здесь x, y – массив (список) абсцисс и ординат экспериментальных точек соответственно, v – имя переменной в подбираемой кривой, параметр `function` определяет уравнение подбираемой кривой, кроме переменных в уравнение должны входить подбираемые параметры (например, a, b, c и т.д.).

Вместо массивов x и y можно использовать список L , каждый элемент которого является списком из двух элементов: первый – абсцисса соответствующей экспериментальной точки, второй – ордината.

Последний параметр (`curve=function`) в функции `LeastSquares` может быть опущен, в этом случае функция будет подбирать линию регрессии.

Решение задачи 3.2 в Maple с помощью функции `LeastSquares` представлено на рис. 3.14–3.16.

¹ Хотя в MATLAB есть возможность решать системы нелинейных уравнений, но, на взгляд авторов, легче преобразовать (прологарифмировать, сделать замены и т.д.) исходную функцию, чем решать систему нелинейных уравнений.

Решение задач аппроксимации средствами Maple

```

x:=[10.1,10.2,10.3,10.8,10.9,11,11.1,11.4,12.2,13.3,13.8,14,14.4,14.5,15,15.6,15.8,
y:=[24,36,26,45,34,37,55,51,75,84,74,91,85,87,94,92,96,97,98,99]:
with(CurveFitting):#Подключаем пакет CurveFitting
L:=[x[n],y[n]] $n=1..20):# Из массивов x и y формируем единый список L
F:=v->LeastSquares(x,y,v,curve=a*v^2+b*v+c);#Формируем аппроксимирующую функцию
F:=v->CurveFitting-LeastSquares(x,y,v,curve=a*v^2+b*v+c)
LeastSquares(x,y,v,curve=a*v^2+b*v+c);#Выводим на экран подобранные функции
-319.2647893 + 48.5515464829097568 v - 1.40661421221142180 v^2
plot([L,F(h)],h=10..20,style=[point,line]);#График подобранной зависимости

```

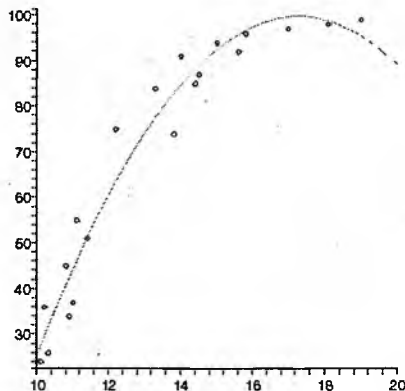


Рис. 3.14 ▾ Построение аппроксимирующей зависимости $y = a_0 + a_1x + a_2x^2$ с помощью функции `LeastSquares`


```

x:=[10.1,10.2,10.3,10.8,10.9,11,11.1,11.4,12.2,13.3,13.8,14,14.4,14.5,15,15.6,15.8
y:=[24,36,26,45,34,37,55,51,75,84,74,91,85,87,94,92,96,97,98,99]:
with(CurveFitting):#Подключаем пакет CurveFitting
L:=[[x[n],y[n]] $n=1..20]:# Из массивов x и y формируем единый список L
F:=v->LeastSquares(x,y,v,curve=a*v^3+b*v^2+c);#Формируем аппроксимирующую функцию
F:=v->CurveFitting-LeastSquares(x,y,v,curve=a*v^3+b*v^2+c)
LeastSquares(x,y,v,curve=a*v^3+b*v^2+c);#Выводим на экран подобранный функцию
-93.72944686 + 1.98990795413157962 v^2 - 0.0773661646055274905 v^3
plot([L,F(h)],h=10..20,style=[point,line]);#График подобранной зависимости

```

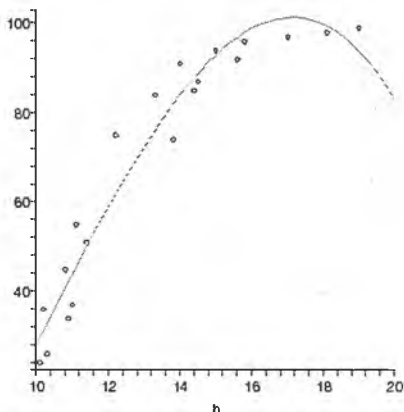


Рис. 3.15 ▼ Построение аппроксимирующей зависимости $y = b_0 + b_1x + b_2x^2$ в Maple с помощью функции LeastSquares

```

> x:=[10.1,10.2,10.3,10.8,10.9,11,11.1,11.4,12.2,13.3,13.8,14,14.4,14.5,15,15.6,15.8,17,18.1,19]:
> y:=[24,36,26,45,34,37,55,51,75,84,74,91,85,87,94,92,96,97,98,99]:
> with(CurveFitting):#Подключаем пакет CurveFitting
> L:=[[x[n],y[n]] $n=1..20]:# Из массивов x и y формируем единый список L
> F:=v->LeastSquares(x,y,v,curve=a*v^3+b*v^2+c*v+d);#Формируем аппроксимирующую функцию
      F:=v -> CurveFitting-LeastSquares(x,y,v,curve=a*v^3+b*v^2+c*v+d)
> LeastSquares(x,y,v,curve=a*v^3+b*v^2+c*v+d);#Выводим на экран подобранную функцию
      -701.7599213 + 131.838379578986860 v - 7.30339207770544796 v^2 + 0.135946532818949000 v^3
> plot([L,F(h)],h=10..20,style=[point,line]);#График подобранной зависимости

```

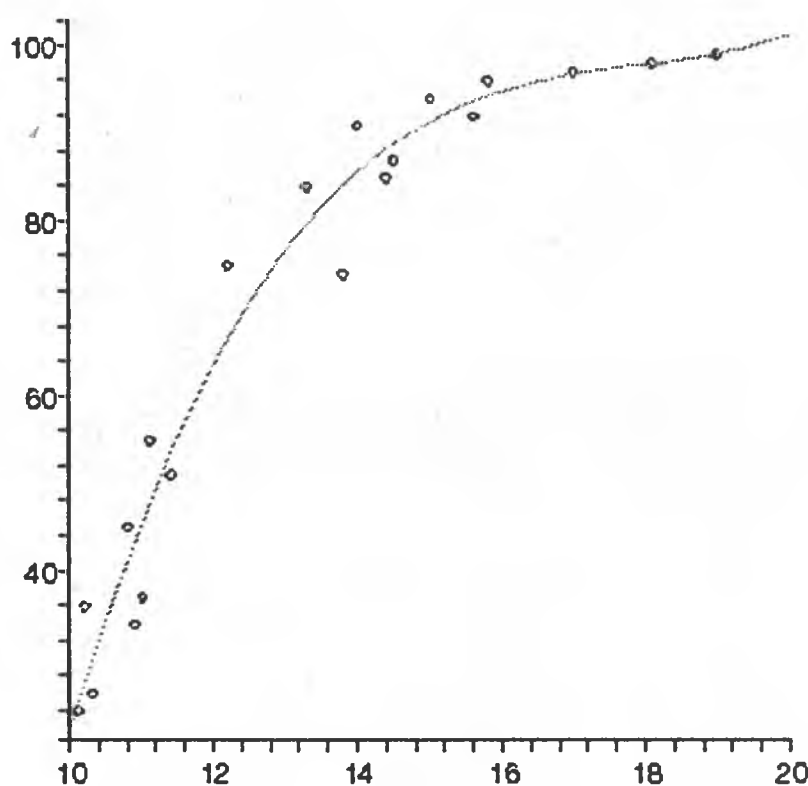


Рис. 3.16 ▽ Построение аппроксимирующей зависимости $y = c_0 + c_1x^2 + c_2x^3$ в Maple с помощью функции LeastSquares

Функция Maple LeastSquares позволяет подобрать любую зависимость методом наименьших квадратов.

3.5. Советы по применению математических пакетов при подборе зависимостей методом наименьших квадратов

Все три пакета можно использовать для решения задач обработки экспериментальных данных методом наименьших квадратов. Однако, на взгляд авторов, лучше всего для решения задач аппроксимации использовать пакет Mathcad, в котором есть большое количество встроенных функций для расчета параметров зависимостей, а также с помощью решающего блока Given – Minimize можно

подобрать параметры практически любой зависимости. Функция Maple Least-Squares также позволяет подобрать параметры практически любой зависимости. Однако отсутствие функций, аналогичных Mathcad corr, intercept, slope, line, для расчета параметров линейной зависимости и неудобство обработки в Maple числовых данных не позволяют авторам рекомендовать читателю Maple в качестве идеального средства для решения задач аппроксимации при инженерных расчетах. Но при подборе очень сложных аппроксимирующих зависимостей Maple может оказаться единственным пакетом, в котором можно решить задачу, не прибегая к программированию. В MATLAB есть встроенная функция для подбора коэффициентов полинома, однако наличие функции fminsearch и развитые средства программирования позволяют подобрать параметры большого количества зависимостей. Для решения более сложных задач аппроксимации можно использовать Curve Fitting Toolbox, входящий в состав последних версий MATLAB.

3.6. Задания для самостоятельного решения

Подобрать функциональную зависимость заданного вида с помощью метода наименьших квадратов (задания 1–5). Определить суммарную ошибку.

1. В результате эксперимента была определена некоторая табличная зависимость (табл. 3.5). Подобрать функциональную зависимость вида $P(s) = As^3 + Bs^2 + D$.

Таблица 3.5 ▼ Экспериментальные данные к заданию 1

s	0	1	1.5	2	2.5	3	3.5	4	4.5	5
P	12	10.1	11.58	17.4	30.68	53.6	87.78	136.9	202.5	287

2. В результате эксперимента были получены некоторые данные, представленные в виде таблицы (табл. 3.6). Необходимо построить аналитическую зависимость вида $G(s) = As^b$.

Таблица 3.6 ▼ Экспериментальные данные к заданию 2

s	0.5	1.5	2	2.5	3	3.5	4	4.5	5
G	3.99	5.65	6.41	6.71	7.215	7.611	7.83	8.19	8.3

3. Данные, представленные в виде таблицы (табл. 3.7), были получены в результате эксперимента. Определить коэффициенты функции $K(s) = Ae^{bs}$.

Таблица 3.7 ▼ Экспериментальные данные к заданию 3

s	0	0.5	1	1.5	2	2.5	3.5	3.5	4
K	2.31	2.899	3.534	4.412	5.578	6.92	8.699	10.69	13.39

4. Определить параметры функции $V(s) = As^b e^{Cs}$ для экспериментальных данных, представленных в виде таблицы (табл. 3.8).

Таблица 3.8 ▼ Экспериментальные данные к заданию 4

s	0.2	0.7	1.2	1.7	2.2	2.7	3.2
V	2.3198	2.8569	3.5999	4.4357	5.5781	6.9459	8.6621

5. Подобрать аналитическую зависимость вида $Y = x/(Ax - B)$, используя данные, полученные в результате эксперимента и сведенные в таблицу (табл. 3.9).

Таблица 3.9 ▼ Экспериментальные данные к заданию 5

x	3	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9
Y	0.61	0.6	0.592	0.58	0.585	0.583	0.582	0.57	0.572	0.571

6. Найти приближенное значение функции при заданном значении аргумента с помощью сплайн-интерполяции в точках $X_1 = 3,75$, $X_2 = 4,75$, $X_3 = 5,25$. Функция задана таблично (табл. 3.10). Изобразить график построенной интерполяционной зависимости, экспериментальные и рассчитанные точки на экране дисплея.

Таблица 3.10 ▼ Исходные данные к заданию 6

x	2	2.5	3	3.5	4	4.5	5	5.5	6
y	5.197	7.78	11.14	15.09	19.245	23.11	26.25	28.6	30.3

7. Найти приближенное значение функции при заданном значении аргумента с помощью линейной интерполяции в точках $X_1 = 1$, $X_2 = 1,5$, $X_3 = 2$. Функция задана таблично (табл. 3.11). Изобразить график построенной интерполяционной зависимости, экспериментальные и рассчитанные точки на экране дисплея.

Таблица 3.11 ▼ Исходные данные к заданию 7

x	0.66	0.9	1.17	1.47	1.7	1.74	2.08	2.63	3.12
y	38.9	68.8	64.4	66.5	64.95	59.36	82.6	90.63	113.5

8. Найти приближенное значение функции при заданном значении аргумента с помощью интерполяционного полинома Лагранжа в точках $X_1 = 0,702$, $X_2 = 0,512$, $X_3 = 0,608$. Функция задана таблично (табл. 3.12).

Таблица 3.12 ▼ Исходные данные к заданию 8

x	0.43	0.48	0.55	0.62	0.7	0.75
y	1.63597	1.73234	1.87686	2.03345	2.22846	2.35973

9. Найти приближенное значение функции при заданном значении аргумента с помощью интерполяционного полинома Ньютона в точках $X_1 = 0,308$, $X_2 = 0,325$, $X_3 = 0,312$. Функция задана таблично (табл. 3.13).

Таблица 3.13 ▼ Исходные данные к заданию 9

x	0.298	0.303	0.310	0.317	0.323	0.330
y	3.25578	3.17639	3.12180	3.04819	2.98755	2.91950

10. Найти приближенное значение функции при заданном значении аргумента с помощью канонического полинома в точках $X_1 = 0,608$, $X_2 = 0,594$, $X_3 = 0,631$. Функция задана таблично (табл. 3.14).

Таблица 3.14 ▼ Исходные данные к заданию 10

x	0.593	0.598	0.605	0.613	0.619	0.627
y	0.53205	0.53562	0.54059	0.54623	0.55043	0.55598

Глава 4

Методы решения нелинейных уравнений

Настоящая глава посвящена методам решения нелинейных уравнений

$$f(x) = 0, \quad (4.1)$$

где $f(x)$ – заданная функция действительного или комплексного аргумента. Причем если $f(x)$ – многочлен, отличный от нулевого, то уравнение (4.1) называют *алгебраическим*, иначе – *трансцендентным*¹. Большинство этих методов предполагает, что известны достаточно малые окрестности, в каждой из которых имеется только один корень уравнения [5]. Принимая за начальное приближение корня одну из точек этой окрестности, можно вычислить искомый корень с заданной точностью. Таким образом, задача приближенного вычисления корней уравнения (4.1) распадается на две:

- *отделение корней уравнения*, то есть отыскание достаточно тесных промежутков, в каждом из которых содержится только один корень уравнения;
- *вычисление корня с заданной точностью*, если известно некоторое начальное его приближение в области, не содержащей других корней.

Однако существуют специальные методы решения алгебраических уравнений, которые не требуют знания начального приближения корня.

¹ Трансцендентные функции – аналитические функции, не являющиеся алгебраическими, например показательные, логарифмические, тригонометрические и т.п.

В следующих разделах кратко описаны некоторые элементарные способы отделения корней, рассмотрены методы уточнения корней по известным начальным приближениям и особое внимание уделено методам, предназначенным для решения алгебраических уравнений.

4.1. Локализация (отделение) корней

Графические методы решения нелинейных уравнений – одни из самых простых. Однако применять их можно лишь для грубой оценки корней. С другой стороны, графические методы можно использовать для решения задачи *отделения корней*¹.

Суть графического метода решения нелинейных уравнений состоит в следующем. Необходимо построить график функции $y = f(x)$, а затем найти абсциссы точек пересечения этого графика с осью x , которые и будут приближенными значениями действительных корней уравнения. На рис. 4.1 видно, что функция четырежды пересекает ось абсцисс, следовательно, уравнение имеет соответствующее число корней. Расположены корни на интервалах $[-0.5; 0]$, $[0; 0.5]$, $[1.5; 2]$ и $[2; 2.5]$.

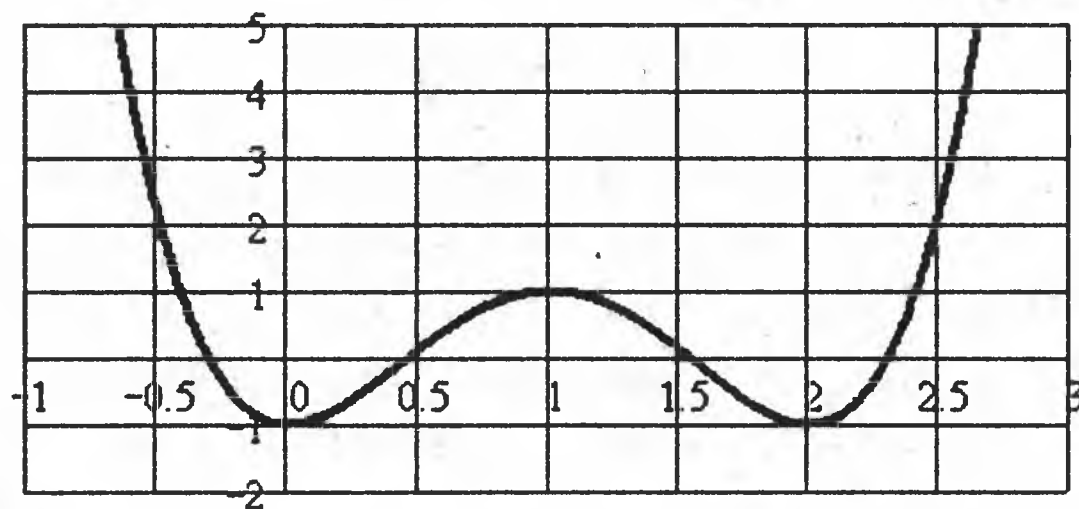


Рис. 4.1 ▼ Графическое решение уравнения $y = f(x)$

Иногда бывает удобнее представить уравнение (4.1) в виде

$$\phi(x) = \psi(x),$$

где функции $\phi(x)$ и $\psi(x)$ более простые, чем $f(x)$. Тогда найти приближенные значения корней уравнения можно, определив абсциссы точек пересечения

¹ Не существует достаточно эффективных методов решения задачи отделения корней в общем случае. Особенно большие трудности возникают при переходе от одного уравнения к системе $f_k(x_1, x_2, \dots, x_n) = 0$.

графиков $y = \phi(x)$ и $y = \psi(x)$. На рис. 4.2 приведен пример графического решения уравнения

$$\left(\sqrt[3]{(x-1)^2} - \sqrt[3]{x^2} = 0\right) \Rightarrow \left(\sqrt[3]{(x-1)^2} = \sqrt[3]{x^2}\right).$$

Несложно заметить, что корень уравнения находится на интервале от нуля до единицы.

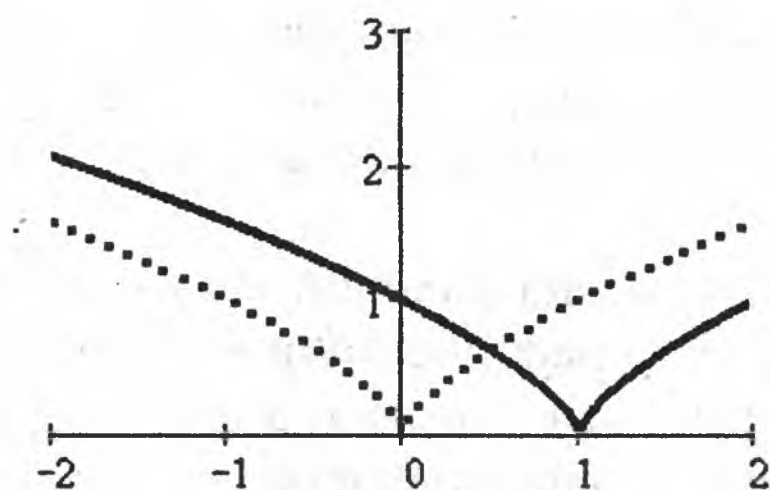


Рис. 4.2 ▽ Графическое решение уравнения $f(x) = y(x)$

Для отыскания комплексных корней уравнения $f(z) = 0$ можно, положив $z = x + iy$, где $i^2 = -1$, представить уравнение в виде

$$\phi(x, y) + i \cdot \psi(x, y) = 0, \quad (4.2)$$

где $\phi(x, y)$ и $\psi(x, y)$ – действительные функции действительных аргументов x и y . Построив кривые $\phi(x, y) = 0$ и $i \cdot \psi(x, y) = 0$ и отыскав абсциссы и ординаты их точек пересечения, получим действительные и мнимые части корней уравнения $f(z) = 0$. Например, уравнение $z^2 + 2 \cdot z + 5 = 0$ действительных корней не имеет. Попробуем отыскать комплексные корни. Приведем уравнение к виду (4.2):

$$(x + i \cdot y)^2 + 2(x + i \cdot y) + 5 = 0 \Rightarrow (x^2 - y^2 + 2 \cdot x + 5) + 2 \cdot i \cdot (x \cdot y + y) = 0.$$

Таким образом, задача свелась к построению графиков функций

$$\phi(x, y) = x^2 - y^2 + 2 \cdot x + 5 = 0 \text{ и } \psi(x, y) = x \cdot y + y = 0.$$

Для удобства построения проведем элементарные преобразования и получим уравнения линий

$$y = \pm (x^2 + 2 \cdot x + 5)^{1/2} \text{ и } x = -1.$$

На рис. 4.3 изображено графическое решение уравнения $z^2 + 2 \cdot z + 5 = 0$. Несложно заметить, что линии пересекаются в двух точках: $(-1; 2)$ и $(-1; -2)$, следовательно, уравнение имеет два комплексных корня: $x = -1 + 2 \cdot i$ и $x = -1 - 2 \cdot i$.

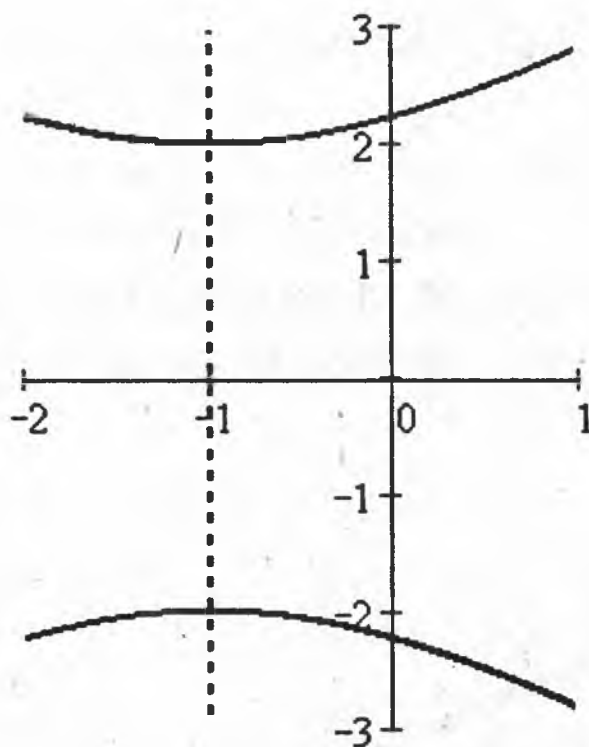


Рис. 4.3 ▼ Графическое решение уравнения $f(z) = 0$

Для выделения интервалов, в которых находятся действительные корни уравнения $f(x) = 0$, если $f(x)$ – непрерывная функция¹, можно воспользоваться следующими предположениями [5]:

- если на концах некоторого отрезка $[a; b]$ непрерывная функция $f(x)$ принимает значения разных знаков $f(a) \cdot f(b) < 0$, то на этом отрезке уравнение $f(x) = 0$ имеет хотя бы один корень, при этом отрезок $[a; b]$ называют *интервалом изоляции корня*, а значения a и b называют *пределами интервала изоляции*;
- если при этом функция $f(x)$ имеет первую производную, не меняющую знака, то корень единственный.

Теперь, когда нам известно, как определить интервалы изоляции корней уравнения (отделить корни), рассмотрим методы вычисления корней с заданной точностью (уточнение корней).

4.2. Метод половинного деления (дихотомии)

Метод половинного деления (дихотомии) заключается в следующем. Пусть дано уравнение $f(x) = 0$, где функция $f(x)$ непрерывна на $[a; b]$ и $f(a) \cdot f(b) < 0$. Для отыскания корня уравнения $f(x) = 0$, принадлежащего отрезку $[a; b]$, этот отрезок

¹ В некоторых случаях может понадобиться существование первой и второй производных, что будет оговорено.

необходимо поделить пополам и принять за первое приближение корня точку c , которая является серединой отрезка $[a; b]$. Далее из двух отрезков – $[a; c]$ и $[c; b]$ – выбирается тот, на концах которого функция $f(x)$ имеет противоположные знаки. Это будет означать сужение интервала изоляции корня уравнения $f(x) = 0$. Новый суженный отрезок снова делится пополам, и проводятся аналогичные вычисления. Графическая интерпретация метода половинного деления приведена на рис. 4.4.

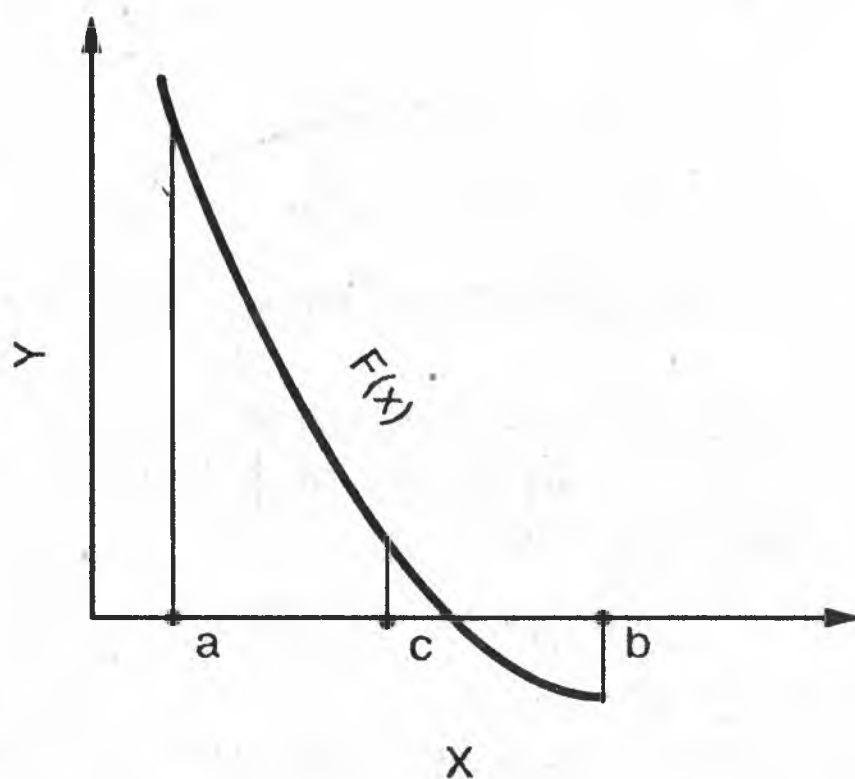


Рис. 4.4 ▼ Графическая интерпретация метода половинного деления

Метод половинного деления легко реализуется в виде следующего алгоритма (блок-схема приведена на рис. 4.5):

1. Находим точку $c = (a + b) / 2$.
2. Если $f(a) \cdot f(c) < 0$, то корень лежит на интервале $[a; c]$, если иначе – корень лежит на интервале $[c; b]$.
3. Если величина интервала не превышает некоторое достаточно малое число ϵ , то найден корень с точностью ϵ , иначе возвращаемся к п.1.

Однако, несмотря на простоту, такое последовательное сужение интервала проводится редко, так как требует слишком большого количества вычислений. Кроме того, этот способ не всегда позволяет найти решение с заданной точностью. С другой стороны, его можно использовать как один из методов отделения корней (пример программирования см. в задаче 4.8).

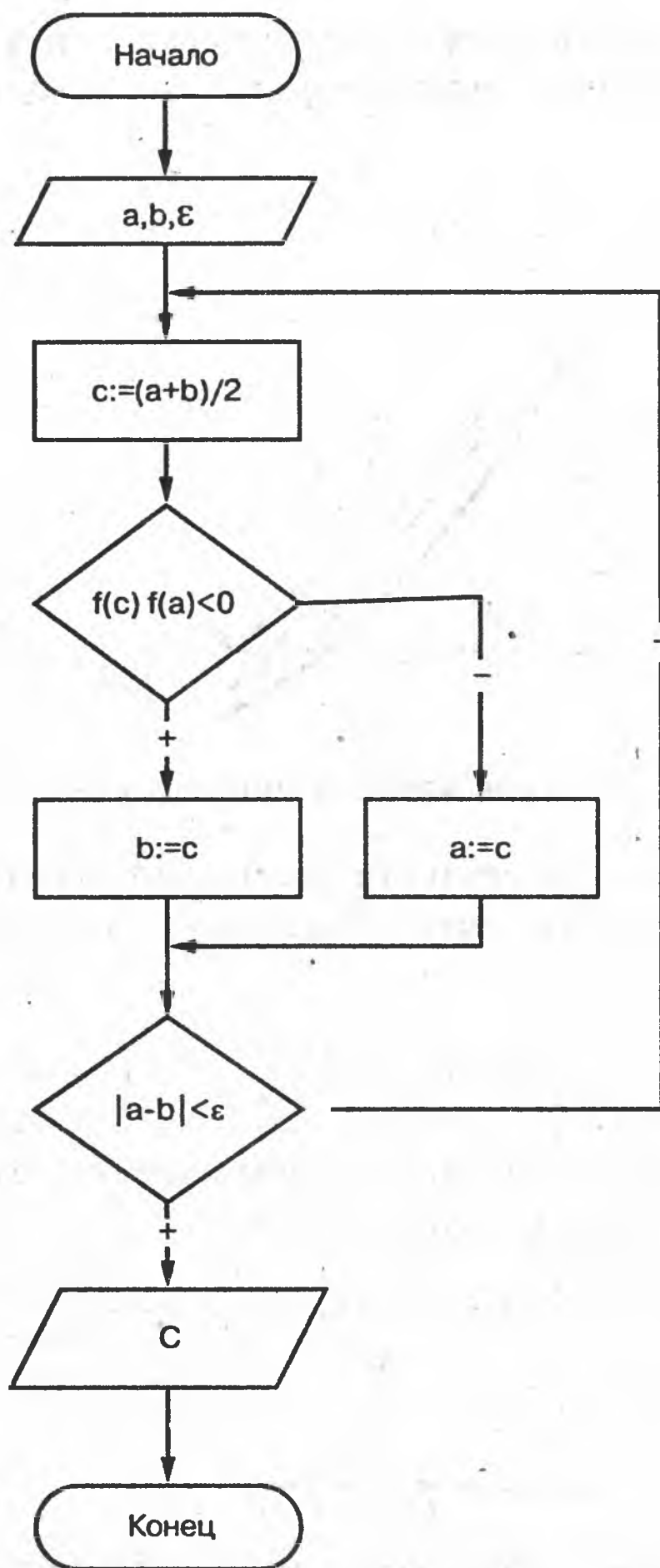


Рис. 4.5 ▼ Алгоритм решения уравнения методом дихотомии

4.3. Метод хорд (пропорциональных частей)

Рассмотрим более быстрый способ (в отличие от метода дихотомии) нахождения корня уравнения $f(x) = 0$, лежащего на интервале $[a; b]$, таком, что $f(a) \cdot f(b) < 0$. Идея метода хорд состоит в следующем. Вместо того чтобы делить отрезок $[a; b]$ пополам (см. метод дихотомии), разделим его в отношении $-f(a) : f(b)$. Таким образом, первое приближение корня $x = c$ находится в точке пересечения отрезка $[a; b]$ хордой, проходящей через точки $(a, f(a))$ и $(b, f(b))$.

Далее, применяя этот прием к тому из отрезков $[a; c]$ или $[c; b]$, на концах которого функция имеет противоположные знаки, получим следующее приближение корня (рис. 4.6).

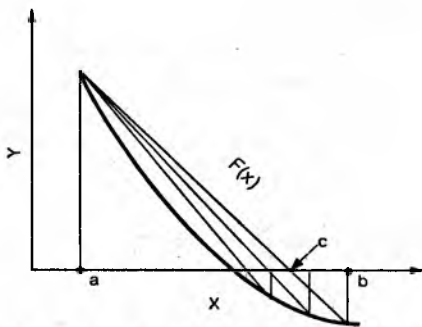


Рис. 4.6 ▽ Графическая интерпретация метода хорд

Выведем формулу для вычисления приближенного значения c корня уравнения $f(x) = 0$. Запишем уравнение прямой, проходящей через точки с координатами $(a; f(a))$ и $(b; f(b))$:

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a} \Rightarrow y = \frac{f(b) - f(a)}{b - a} \cdot (x - a) + f(a). \quad (4.3)$$

Прямая, заданная уравнением (4.3), пересекает ось X при условии $y = 0$. Найдём точку пересечения хорды с осью X :

$$y = \frac{f(b) - f(a)}{b - a} \cdot (x - a) + f(a), \quad x = a - \frac{f(a) \cdot (b - a)}{f(b) - f(a)}.$$

Итак,

$$c = a - \frac{f(a)}{f(b) - f(a)}(b - a). \quad (4.4)$$

Алгоритм, реализующий метод хорд, можно представить так (блок-схема приведена на рис. 4.7):

1. Находим точку c по формуле (4.4).
2. Если $f(a) \cdot f(c) < 0$, то корень лежит на интервале $[a; c]$, иначе – корень лежит на интервале $[c; b]$.
3. Если абсолютное значение $f(c)$ не превышает некоторое достаточно малое число ϵ , то найден корень с точностью ϵ , иначе возвращаемся к п. 1.

Заметим, что описанный способ позволяет достаточно быстро (за меньшее количество шагов, чем в методе дихотомии) вычислить значение корня уравнения (4.1) с заданной точностью (пример по определению см. в задаче 4.8).

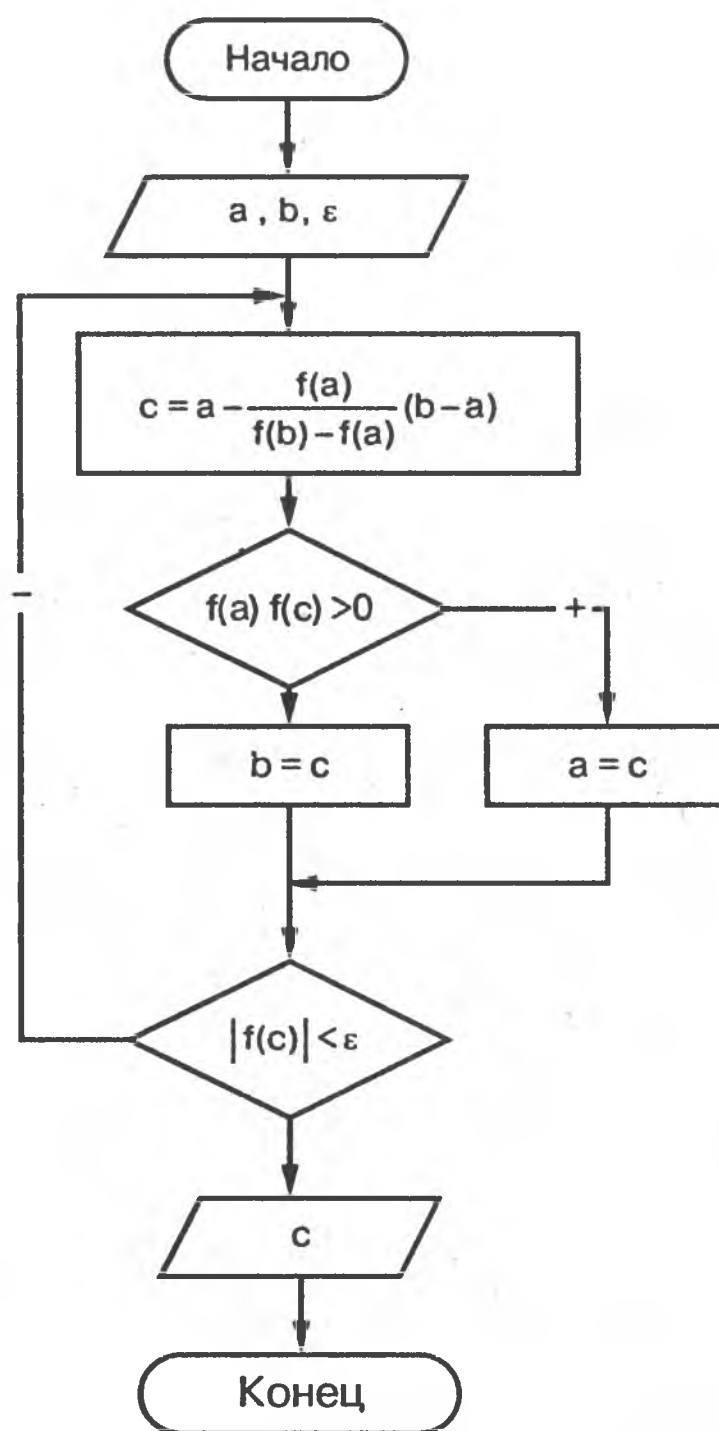


Рис. 4.7 ▼ Алгоритм метода хорд

4.4. Метод касательных (метод Ньютона)

Пусть корень уравнения $f(x) = 0$ отделен на отрезке $[a; b]$, причем $f'(x)$ и $f''(x)$ непрерывны и сохраняют определенные знаки при $a \leq x \leq b$. Вычислив какое-либо приближенное значение корня $x = c$, можно уточнить его при помощи *метода касательных*. Геометрически метод касательных эквивалентен [11] замене небольшой дуги кривой $y = f(x)$ касательной, проведенной в некоторой точке кривой так, как показано на рис. 4.8.

Пусть $x_0 = b$, причем $f(x_0) \cdot f''(x_0) > 0$. Проведем касательную к кривой $y = f(x)$ в точке $(x_0; f(x_0))$. В качестве начального приближения корня $x_1 = c$ выберем абсциссу точки пересечения этой касательной с осью Ox . Далее проведем касательную через точку $(x_1; f(x_1))$, абсцисса точки пересечения которой с осью Ox даст второе приближение корня x_2 , и так далее до тех пор, пока не будет достигнута точность ε .

Уравнение касательной в точке $(c; f(c))$ имеет вид:

$$y - f(c) = f'(c) \cdot (x - c).$$

2. Уточняем значение корня по формуле (4.5).
3. Если абсолютное значение функции в найденной точке не превышает некоторое достаточно малое значение ε , то корень найден, иначе переходим к п.2.

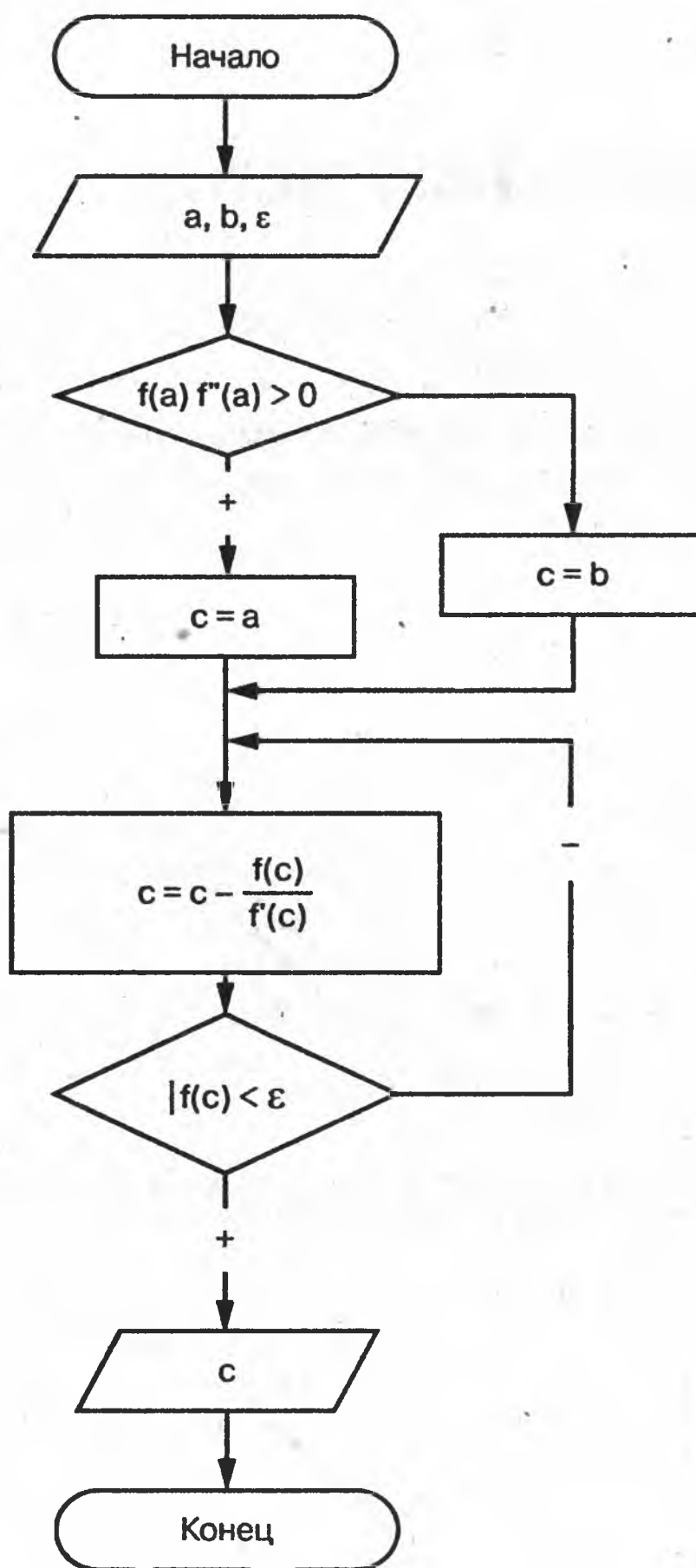


Рис. 4.10 ▼ Алгоритм метода Ньютона

Из формулы (4.5) видно, что чем больше численное значение производной $f'(x)$ в окрестности данного корня, тем меньшего уточнения требует n -е приближение. Поэтому метод касательных (Ньютона) удобен, когда в окрестности данного корня график функции имеет большую кривизну. Но если численное значение производной $f'(x)$ в окрестности корня мало, то его уточнение требует

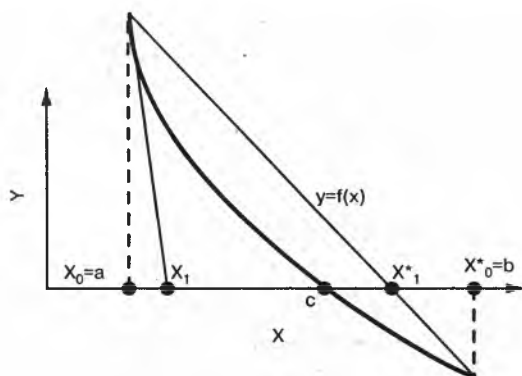


Рис. 4.13 ▼ Геометрическая интерпретация комбинированного метода для $f'(x) < 0$ и $f''(x)(x) > 0$

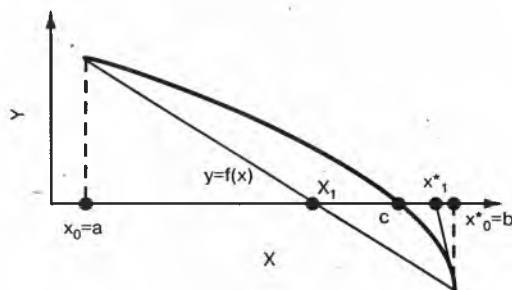


Рис. 4.14 ▼ Геометрическая интерпретация комбинированного метода для $f'(x) < 0$ и $f''(x)(x) < 0$

Рассмотрим первый случай (рис. 4.11), так как характер вычислений в остальных случаях легко понять из соответствующих чертежей. Кроме того, эти случаи можно свести к первому, если заменить рассматриваемое уравнение $f(x) = 0$ равносильными ему: $-f(x) = 0$ или $\pm f(z) = 0$, где $z = -x$.

Итак, пусть $f'(x) > 0$ и $f''(x) > 0$ при $a \leq x \leq b$. Тогда полагаем $x_0 = a$ и $x_0^* = b$. Следовательно, из формул (4.4) и (4.5) имеем:

$$x = x - \frac{f(x)}{f(x^*) - f(x)}(x^* - x),$$

$$x^* = x^* - \frac{f(x^*)}{f'(x^*)}.$$

Процесс вычисления корня прекращается тогда, когда будет обнаружено, что разность между значением корня по недостатку и значением по избытку не

превышает некоторого, заранее заданного, достаточно малого числа ε . По окончании процесса значение корня вычисляется как среднее арифметическое последних полученных значений x и x^* . Блок-схема алгоритма решения нелинейного уравнения комбинированным методом приведена на рис. 4.15 (пример программирования см. в задаче 4.9).

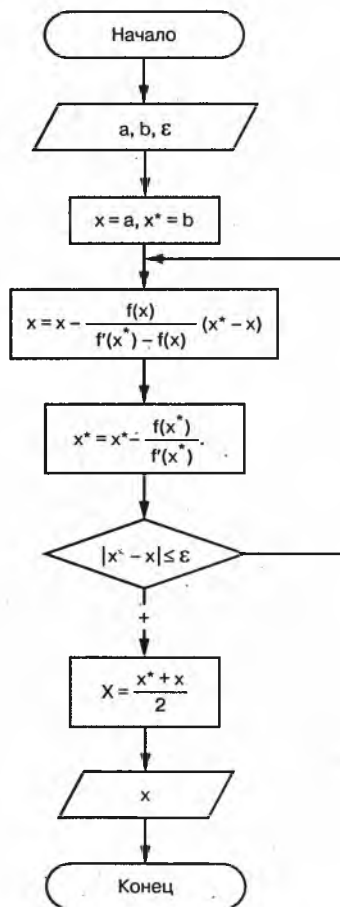


Рис. 4.15 ▼ Алгоритм комбинированного метода

4.6. Метод простой итерации (последовательных приближений)

Идея *метода простой итерации* заключается в следующем [11]. Пусть требуется найти корни уравнение $f(x) = 0$, где $f(x)$ – непрерывная функция. Заменим это уравнение равносильным уравнением

$$x = \varphi(x). \quad (4.6)$$

Предположим, что нам известно грубое приближенное значение корня x_0 . Подставим его в правую часть уравнения (4.6) и получим некоторое значение

$$x_1 = \varphi(x_0). \quad (4.7)$$

Теперь подставим вычисленное значение x_1 в правую часть равенства (4.7) и получим новое число $x_2 = \varphi(x_1)$. Повторяя этот процесс, будем иметь последовательность чисел

$$x_n = \varphi(x_{n-1}), \quad n = 1, 2, \dots \quad (4.8)$$

Если эта последовательность сходящаяся, то есть существует предел $c = \lim_{n \rightarrow \infty} x_n$, то, переходя к пределу в равенстве (4.8) и предполагая функцию $\varphi(x)$ непрерывной, найдем:

$$\lim_{n \rightarrow \infty} x_n = \varphi\left(\lim_{n \rightarrow \infty} x_{n-1}\right) \Rightarrow c = \varphi(c).$$

Из чего следует, что c – корень уравнения (4.6) и может быть вычислен по формуле (4.8) с любой степенью точности. Вообще, процесс решения нелинейного уравнения методом простой итерации считается завершенным, если два последовательных приближения x_n и x_{n-1} совпадают между собой с заданной точностью ε .

Геометрически метод простой итерации можно объяснить следующим образом. Построим графики функций $y = x$ и $y = \varphi(x)$ так, как показано на рис. 4.16. Абсцисса c точки пересечения M кривой $y = \varphi(x)$ и прямой $y = x$ – действительный корень уравнения (4.6). Для дальнейших расчетов изобразим ломаную линию $A_0B_1A_1B_2A_2\dots$ начиная с некоторой точки $A_0(x_0, \varphi(x_0))$. Вершины ломаной $A_0, A_1, A_2, \dots$ лежат на кривой $y = \varphi(x)$, а $B_1, B_2, \dots$ – на прямой $y = x$. Общие абсциссы пар точек A_1 и B_1, A_2 и $B_2, \dots$ представляют собой последовательные приближения $x_1, x_2, \dots$ корня c .

Обратите внимание, что кривая $y = \varphi(x)$ в окрестности корня c – пологая, то есть $|\varphi'(x)| < 1$ и процесс итерации сходится. На рис. 4.17 представлен случай расходящегося процесса, где $|\varphi'(x)| > 1$.

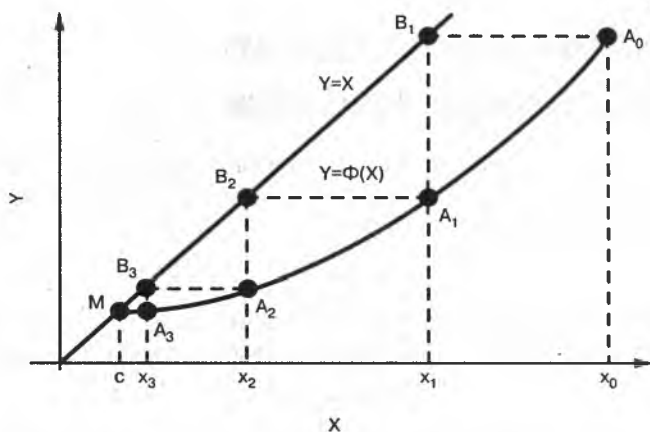


Рис. 4.16 ▼ Геометрическая интерпретация метода простой итерации для $|\varphi'(x)| < 1$.

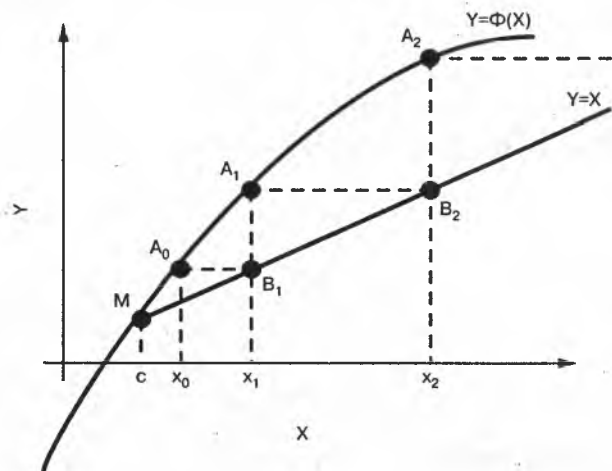


Рис. 4.17 ▼ Геометрическая интерпретация метода простой итерации для $|\varphi'(x)| > 1$.

Для практического применения метода простой итерации следует помнить, что функцию (4.6) нужно представлять таким образом, чтобы было истинным неравенство:

$$|\varphi'(x)| \leq q < 1, \quad (4.9)$$

причем чем меньше число q , тем быстрее последовательные приближения сходятся к корню c [11].

Рассмотрим общий прием приведения уравнения (4.1) к виду (4.6), для которого выполнялось бы неравенство (4.9). Пусть искомый корень лежит на отрезке $[a; b]$, причем

$$0 < m \leq f'(x) \leq M \quad (4.10)$$

для $a \leq x \leq b$, где m и M – соответственно наименьшее и наибольшее положительные¹ значения производной $f'(x)$ на отрезке $[a; b]$. Заменим уравнение (4.6) эквивалентным ему уравнением

$$x = x - \lambda f(x) \Rightarrow \varphi(x) = x - \lambda f(x), \text{ для } \lambda > 0.$$

Подберем λ так, чтобы выполнялось неравенство

$$0 \leq \varphi'(x) = 1 - \lambda f'(x) \leq q < 1. \quad (4.11)$$

На основании неравенства (4.10) получим:

$$0 \leq 1 - \lambda \cdot M \leq 1 - \lambda \cdot m \leq q.$$

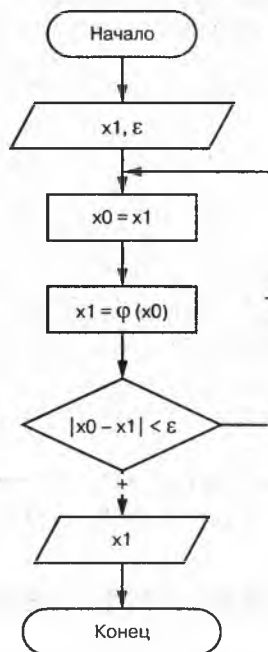


Рис. 4.18 ▼ Алгоритм метода простой итерации

¹ Если производная $f'(x)$ отрицательна, то вместо уравнения $f(x) = 0$ рассматривается уравнение $-f(x) = 0$.

Откуда следует, что $\lambda = 1 / M$ и $q = 1 - m / M < 1$. Таким образом, неравенство (4.11) выполнено.

Вообще, при выполнении условия (4.9) метод простой итерации сходится при любом выборе начального значения x_0 из $[a; b]$. Это свойство делает метод одним из самых надежных вычислительных методов.

Блок-схема метода простой итерации приведена на рис. 4.18 (пример программирования см. в задаче 4.10).

4.7. Вычисление комплексных корней

Пусть необходимо вычислить *комплексный корень* уравнения

$$f(z) = 0, \quad (4.12)$$

где $z = x + iy$ и $i^2 = -1$. Для достижения цели воспользуемся методом, аналогичным методу Ньютона (касательных). Предположим, что z_n – приближенное значение корня уравнения (4.12), принадлежащее некоторой окрестности U , в которой функция $f(z)$ определена, а $z_{n+1} = z_n + \Delta z_n$ – уточненное значение корня. Применяя разложение в ряд Тейлора в точке z_n и считая, что $f(z_{n+1}) \approx 0$, получим:

$$f(z_{n+1}) \approx f(z_n) + \Delta z_n \cdot f'(z_n) = 0.$$

Следовательно:

$$\Delta z_n = -\frac{f(z_n)}{f'(z_n)}.$$

Таким образом, если известно некоторое начальное значение z_0 , то дальнейшие приближения корня можно получить, воспользовавшись формулой:

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}, \quad (n = 0, 1, 2, \dots). \quad (4.13)$$

Алгоритм решения нелинейных уравнений методом Ньютона рассмотрен в разделе 4.5. Реализацию этого метода на примере можно увидеть в задаче 4.11.

4.8. Методы решения алгебраических уравнений

Сделаем несколько общих замечаний. Любое уравнение вида $P(x) = 0$, где $P(x)$ – это ненулевой многочлен, называют *алгебраическим уравнением*, или *полиномом*. Всякое алгебраическое уравнение n -й степени относительно x можно записать в виде

$$a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n = 0, \quad (4.14)$$

где $n \geq 1$ и коэффициенты a_i – действительные числа, причем $a_0 \neq 0$. Кроме того, алгебраическое уравнение n -й степени (4.14) имеет ровно n корней, действительных или комплексных.

При решении алгебраических уравнений нам придется многократно вычислять значения многочленов, значения его производных и коэффициенты частного при делении многочлена на многочлен. В связи с этим остановимся на некоторых приемах подобных вычислений.

4.8.1. Вычисление значений многочлена

Пусть требуется найти *значение многочлена*

$$P_n(x) = a_0 x^n + a_1 x^{n-1} + \dots + a_{n-1} x + a_n \quad (4.15)$$

в точке x_0 . Решение этой задачи сводится к вычислению значения выражения:

$$P_n(x_0) = \sum_{i=0}^n a_i x_0^{n-i}.$$

Алгоритм решения задачи приведен на рис. 4.19. Он достаточно простой и не требует дополнительных пояснений.

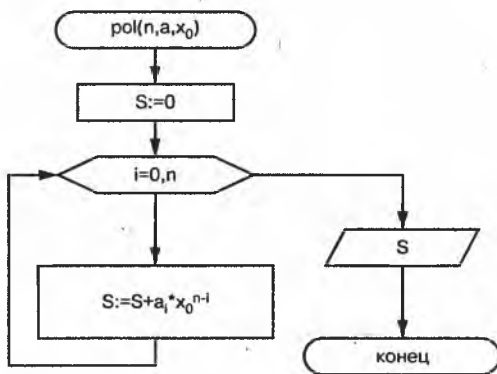


Рис. 4.19 ▼ Алгоритм вычисления значения многочлена

4.8.2. Деление многочлена на линейный множитель

Пусть требуется найти частное и остаток от *деления многочлена* (4.15) на линейный множитель $x - x_0$.

Пусть частным будет многочлен

$$Q_{n-1}(x) = b_0 x^{n-1} + b_1 x^{n-2} + \dots + b_{n-1},$$

а остаток от деления обозначим через b_n . Тогда

$$P_n(x) = Q_{n-1}(x)(x - x_0) + b_n. \quad (4.16)$$

Сравнение коэффициентов при одинаковых степенях x в правой и левой частях тождества (4.16) дает:

$$a_0 = b_0 \text{ и } a_i = b_i - b_{i-1}x_0, \quad (i = 1, 2, \dots, n).$$

Следовательно, имеем рекуррентные соотношения

$$b_0 = a_0 \text{ и } b_i = a_i + b_{i-1}x_0, \quad (i = 1, 2, \dots, n). \quad (4.17)$$

На рис. 4.20 формулы (4.17) представлены в виде схемы, называемой *схемой Горнера*, где $b_0 = a_0$, а каждое число нижней строки равно сумме двух чисел, стоящих над ним.

Алгоритм решения задачи деления многочлена на линейный множитель приведен на рис. 4.21.

Кроме того, схему Горнера удобно использовать для вычислений значений многочлена в точке x_0 , так как из тождества (4.16) следует, что $P_n(x_0) = b_n$.

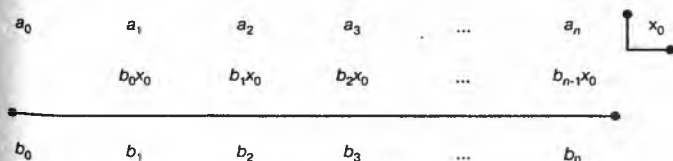


Рис. 4.20 ▼ Схема Горнера

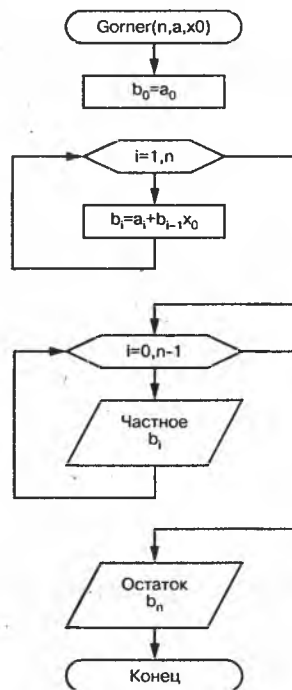


Рис. 4.21 ▼ Алгоритм
схемы Горнера

4.8.3. Вычисление производных многочлена

Рассмотрим *вычисление производных многочлена* (4.15) в точке $x = x_0$. Пусть частное от деления $P_n(x)$ на $x - x_0$:

$$Q_{n-1}(x) = b_0^0 x^{n-1} + b_1^0 x^{n-2} + \dots + b_{n-2}^0 x + b_{n-1}^0,$$

тогда

$$P_n(x) = (x - x_0)Q_{n-1}(x) + b_n^0.$$

Частное от деления $Q_{n-1}(x)$ на $x - x_0$ есть

$$Q_{n-2}(x) = b_0^1 x^{n-2} + b_1^1 x^{n-3} + \dots + b_{n-3}^1 x + b_{n-2}^1$$

и

$$Q_{n-1}(x) = (x - x_0)Q_{n-2}(x) + b_{n-1}^1.$$

Следовательно, частное от деления многочлена $Q_{n-k}(x)$ на $x - x_0$ запи

$$Q_{n-k-1}(x) = b_0^k x^{n-k-1} + b_1^k x^{n-k-2} + \dots + b_{n-k-1}^k$$

и

$$Q_{n-k}(x) = (x - x_0)Q_{n-k-1}(x) + b_{n-k}^k.$$

Таким образом, получим последовательность многочленов

$$P_n(x), Q_{n-1}(x), Q_{n-2}(x), \dots, Q_1(x), Q_0(x)$$

и матрицу коэффициентов этих многочленов

$$\begin{array}{ccccccc} a_0 & a_1 & a_2 & \dots & a_{n-2} & a_{n-1} & a_n, \\ b_0^0 & b_1^0 & b_2^0 & \dots & b_{n-2}^0 & b_{n-1}^0 & b_n^0, \\ b_0^1 & b_1^1 & b_2^1 & \dots & b_{n-2}^1 & b_{n-1}^1 & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ b_0^{n-1} & b_1^{n-1}, & & & & & \\ b_0^n. & & & & & & \end{array}$$

Аналогично (4.17) получим соотношения:

$$b_0^k = b_0^{k-1}, \quad b_i^k = b_i^{k-1} + b_{i-1}^{k-1} x_0, \quad (i = 1, 2, \dots, n-k; k = 0, 1, 2, \dots, n),$$

где для симметрии положим

$$b_0^{-1} = a_i, \quad (i = 1, 2, \dots, n).$$

Используя тождества (4.19) и (4.21) при $k = 1, 2, \dots, n-1$, получим

$$P_n(x) = b_n^0 + b_{n-1}^1 (x - x_0) + b_{n-2}^2 (x - x_0)^2 + \dots + b_0^n (x - x_0)^n.$$

Запишем разложение $P_n(x)$ в ряд Тейлора в окрестности точки $x = x_0$:

$$P_n(x) = P_n(x_0) + \frac{P'_n(x_0)}{1!}(x - x_0) + \frac{P''_n(x_0)}{2!}(x - x_0)^2 + \dots + \frac{P^{(n)}_n(x_0)}{n!}(x - x_0)^n. \quad (4.24)$$

Сравнивая (4.23) и (4.24) получим соотношения

$$P_n(x_0) = b^n_0, \quad P'_n(x_0) = b_{n-1}, \quad P''_n(x_0) = 2!b_{n-2}, \dots, \quad P^{(n)}_n(x_0) = n!b_0^n. \quad (4.25)$$

Таким образом, производные многочлена $P_n(x)$ в точке $x = x_0$ можно вычислять, используя формулы (4.22) и (4.25). Алгоритм вычисления i -й производной полинома n -й степени с коэффициентами a_i в точке x_0 приведен на рис. 4.22.

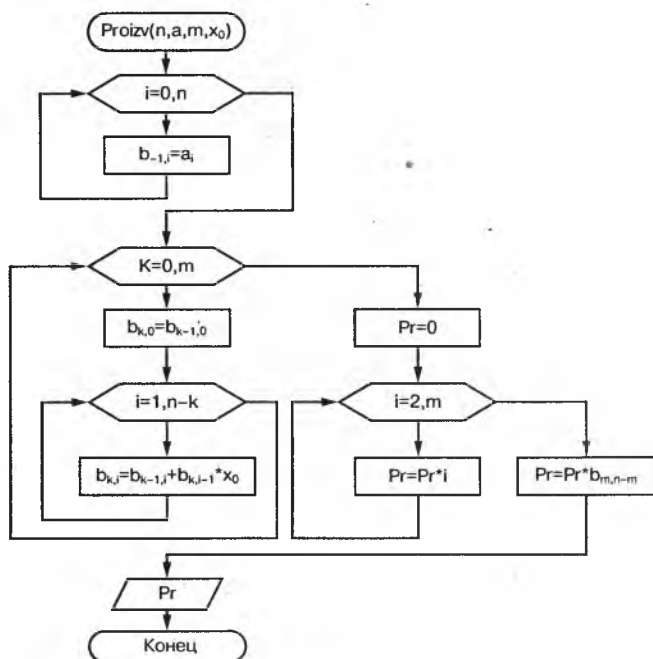


Рис. 4.22 ▼ Алгоритм вычисления производных многочлена

Перейдем к описанию методов решения алгебраических уравнений. В принципе методы решения нелинейных уравнений вида (4.1) могут быть использованы и для решения уравнений (4.14). Но все описанные методы имеют локальный характер, то есть позволяют выполнить процесс уточнения корня при достаточно хорошо заданном начальном приближении. Далее будут рассмотрены

методы, позволяющие находить все корни полиномов, в том числе и комплексные, без знания их начальных приближений.

4.8.4. Метод спуска

Рассмотрим алгебраическое уравнение

$$P_n(z) = a_0 z^n + a_1 z^{n-1} + \dots + a_{n-1} z + a_n, \quad (a_0 \neq 0). \quad (4.26)$$

В общем случае переменную z будем считать комплексной и запишем многочлен в виде

$$P_n(z) = u(x, y) + i \cdot v(x, y), \text{ где } z = x + i \cdot y.$$

Кроме того, функция $P_n(z)$ удовлетворяет условиям дифференцируемости Коши–Римана, то есть определена в области G , дифференцируема в точке $z \in G$, так как функции $u(x, y)$ и $v(x, y)$ обладают в этой точке непрерывными частными производными:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}; \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

Рассмотрим квадрат модуля функции $P_n(z)$:

$$F(x, y) = P_n(z) \bar{P}_n(z) = u^2(x, y) + v^2(x, y).$$

Известно, что функция $F(x, y)$ имеет только нулевые минимумы. Следовательно, задача отыскания корней уравнения (4.26) эквивалентна задаче отыскания всех локальных минимумов функции $F(x, y)$.

В [5] приведено доказательство того, что если задана точка $z_0 = x_0 + i \cdot y_0$, находящаяся в окрестности минимума функции $F(x, y)$, то для отыскания следующего приближения $z_1 = x_1 + i \cdot y_1$ необходимо сделать шаг $h = z_0 - z_1$, где

$$h_t = -t \frac{P_n'(z_0)}{P_n(z_0)},$$

причем t – положительный параметр, для которого в точке $z_t = x_t + i \cdot y_t = z_0 + h_t$ имеет место неравенство $F(x_t, y_t) < F(x_0, y_0)$. При $t = 1$ получим метод Ньютона, который, как известно, достаточно быстро сходится в окрестности корня. Итак, каждое новое приближение вычисляется по формуле

$$z_{k+1} = z_k + h_k, \quad (4.27)$$

где

$$h_k = -\frac{P_n'(z_k)}{P_n(z_k)}. \quad (4.28)$$

При заданной точности ϵ счет прекращается при выполнении условия $|h_k| < \epsilon$ и z_k принимается за приближенное значение корня уравнения (4.26).

Если нет информации о распределении корней многочлена $P_n(z)$, то за начальное приближение принимают

$$z_0 = \gamma + i \cdot \sqrt[n]{|P_n(\gamma)| \frac{1}{|a_0|}}, \quad (4.29)$$

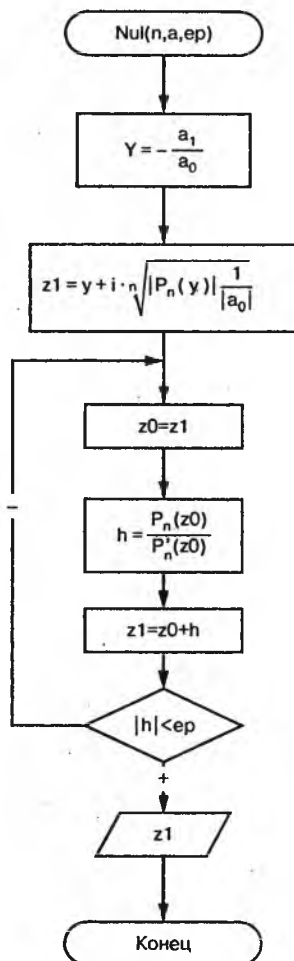


Рис. 4.23 ▼ Алгоритм вычисления корня полинома методом спуска

где

$$\gamma = -\frac{a_1}{a_0}, \quad (4.30)$$

так как γ – среднее арифметическое всех корней многочлена¹. Наличие мнимой части в (4.29) позволяет найти комплексные корни при действительных коэффициентах уравнения (4.26).

На рис. 4.23 приведен алгоритм вычисления одного корня уравнения (4.26). Он заключается в следующих действиях:

1. Вычисляем γ по формуле (4.30).
2. Рассчитываем приближенное значение корня z_1 по формуле (4.29).
2. Принимаем вычисленное значение корня за начальное $z_0 = z_1$;
3. Определяем шаг изменения корня по (4.28), причем $z_k = z_0$.
4. Вычисляем следующее приближение корня по рекуррентному соотношению (4.27).
5. Проверяем точность вычислений, если условие $|h_k| < \varepsilon$ не выполняется, то переходим к п.2, иначе z_1 – корень уравнения.

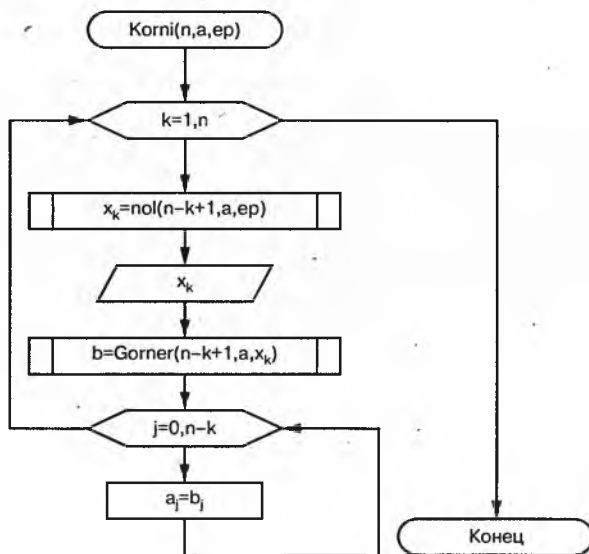


Рис. 4.24 ▼ Алгоритм вычисления всех корней полинома

¹ По определению (4.26) многочлена n -й степени $a_0 \neq 0$.

Обратите внимание, что при практическом применении этого алгоритма нужно будет воспользоваться методами вычисления значения полинома (раздел 4.7.1) и значения его производной в точке (раздел 4.7.3).

Получив один из корней уравнения, переходят к вычислению следующего. Для этого выделяют из многочлена множитель, соответствующий этому корню и ищут следующий корень уравнения, применяя изложенный выше метод. Алгоритм вычисления всех корней полинома приведен на рис. 4.24. Он базируется на том, что число корней многочлена равно его степени. Это утверждение позволяет организовать цикл из n итераций и последовательно вычислить соответствующее количество корней, всякий раз уменьшая степень полинома (4.26) по схеме Горнера, как описано в разделе 4.7.2.

Итак, *метод спуска* – это итерационный метод, который позволяет найти все корни уравнения (4.26) без знания их начальных приближений. Естественно, что если начальные приближения корней известны, то метод можно использовать для их уточнения. Практическое применение рассмотренного метода можно увидеть в задаче 4.19.

4.8.5. Метод парабол

Метод парабол – это также итерационный метод отыскания всех корней алгебраического уравнения (4.26) без знания их начальных приближений. Заключается он в следующем. Исходя из некоторых трех точек z_0, z_1, z_2 строится последовательность z_m . Причем, если уже известны z_{m-2}, z_{m-1}, z_m ($m \geq 2$), то для построения z_{m+1} по значениям многочлена $P_n(z)$ в точках z_{m-2}, z_{m-1}, z_m строится интерполяционный многочлен Лагранжа второй степени

$$L_{2,m}(z) = P_n(z_{m-2}) \frac{(z - z_{m-1})(z - z_m)}{(z_{m-2} - z_{m-1})(z_{m-2} - z_m)} + \\ + P_n(z_{m-1}) \frac{(z - z_{m-2})(z - z_m)}{(z_{m-1} - z_{m-2})(z_{m-1} - z_m)} + P_n(z_m) \frac{(z - z_{m-2})(z - z_{m-1})}{(z_m - z_{m-2})(z_m - z_{m-1})} \quad (4.31)$$

и за z_{m+1} принимается ближайший к z_m корень уравнения $L_{2,m}(z) = 0$.
Введем обозначения

$$\left. \begin{aligned} h_m &= z_m - z_{m-1}, \\ \lambda_m &= \frac{h_m}{h_{m-1}}, \\ \delta_m &= 1 + \lambda_m, \\ m &= 2, 3, \dots \end{aligned} \right\} \quad (4.32)$$

и вместо переменной z введем в (4.31) новую переменную $-\lambda$:

$$L_{2,m}(z) = \bar{L}_m(\lambda) = \lambda^2 \frac{P_n(z_{m-2})\lambda_m^2 - P_n(z_{m-1})\lambda_m \delta_m + P_n(z_m)\lambda_m}{\delta_m} + \\ + \lambda \frac{P_n(z_{m-2})\lambda_m^2 - P_n(z_{m-1})\delta_m^2 + P_n(z_m)(\lambda_m + \delta_m)}{\delta_m} + P_n(z_m). \quad (4.33)$$

Корни этого квадратного трехчлена относительно λ будут:

$$g_m \pm \sqrt{\frac{-2P_n(z_m)\delta_m}{g_m^2 - 4P_n(z_m)\delta_m\lambda_m(P_n(z_{m-2})\lambda_m - P_n(z_{m-1})\delta_m + P_n(z_m))}}, \quad (4.34)$$

где

$$g_m = P_n(z_{m-2})\lambda_m^2 - P_n(z_{m-1})\delta_m^2 + P_n(z_m)(\lambda_m + \delta_m). \quad (4.35)$$

Следовательно,

$$\lambda_{m+1} = \frac{-2P_n(z_m)\delta_m}{g_m \pm \sqrt{g_m^2 - 4P_n(z_m)\delta_m\lambda_m(P_n(z_{m-2})\lambda_m - P_n(z_{m-1})\delta_m + P_n(z_m))}}, \quad (4.36)$$

причем $\pm$ означает, что необходимо выбрать такой знак, при котором знаменатель имеет максимальный модуль.

Итак, получили формулу для вычисления приближения корня по методу парабол:

$$z_{m+1} = z_m + \lambda_{m+1} \cdot h_m. \quad (4.37)$$

В [5] доказано, что последовательность z_m всегда сходится к некоторому корню уравнения (4.26) независимо от выбора z_0, z_1, z_2 . При этом если $z_0 = -1, z_1 = 1, z_2 = 0$, то последовательность сходится к наименьшему по модулю корню. При практическом применении расчеты прекращаются, если

$$\left| \frac{z_m - z_{m-1}}{z_m} \right| < \varepsilon, \quad (4.38)$$

где ε – заданная положительная величина, характеризующая точность приближенного значения корня.

Когда корень найден, поступают так же, как и в методе спуска, то есть выделяют корень и применяют метод к уравнению меньшей степени.

Блок-схема вычисления корней полинома представлена двумя алгоритмами на рис. 4.25. Первый алгоритм определяет один корень уравнения (4.26) и представляет собой последовательное выполнение формул (4.32), (4.35)–(4.36)¹, (4.37) и проверки условия окончания процесса вычислений (4.38). Во втором алгоритме

¹ Здесь для удобства вычислений выражение с корнем в знаменателе обозначим через b_m и вычислим отдельно, затем найдем $\max(|g_m + \beta_m|, |g_m - \beta_m|)$.

заложен процесс вычисления всех корней уравнения (4.26), который практически не отличается от алгоритма, используемого при решении алгебраического уравнения методом спуска. Разница лишь в том, что схема Горнера (п. 4.7.2) используется как для выделения корня, так и для вычисления значения многочлена в точке.

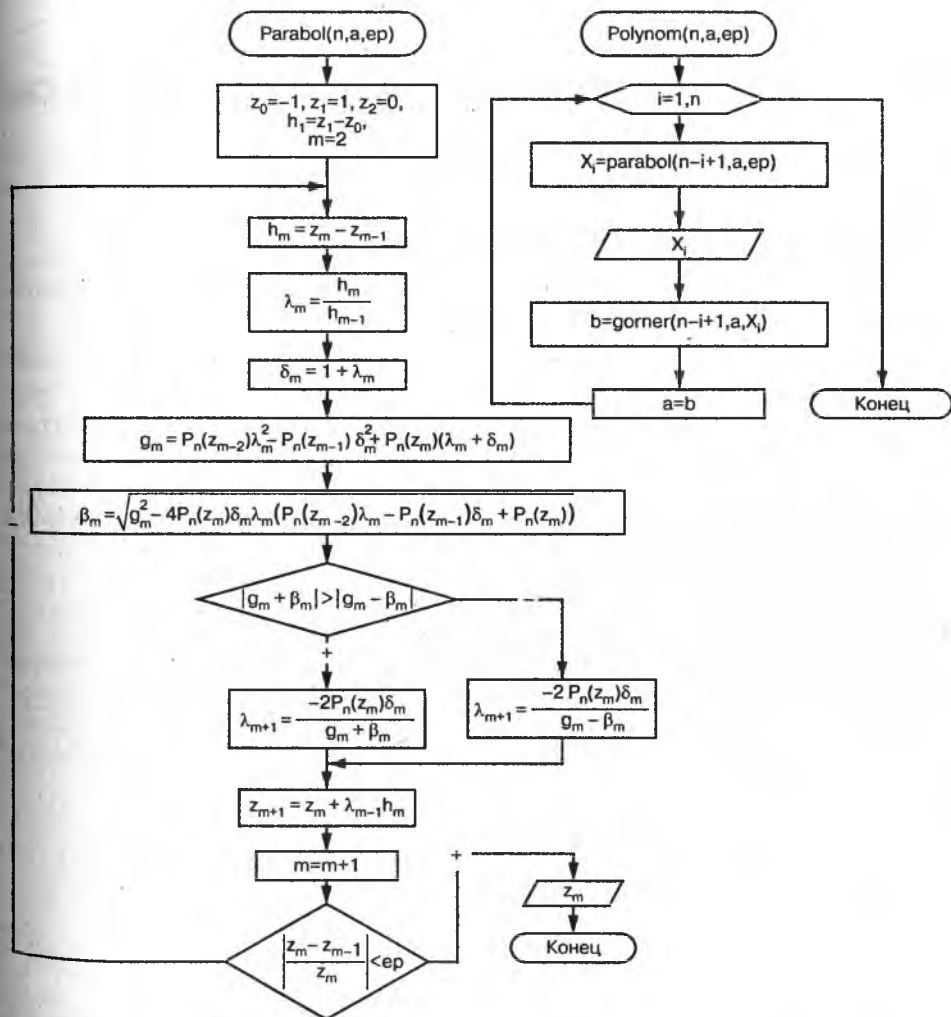


Рис. 4.25 ▽ Алгоритм вычисления корней полинома методом парабол

Преимущество метода парабол по отношению к методу спуска, описанному в разделе 4.7.4, состоит в том, что в первом методе на каждом шаге приходится вычислять только значение многочлена, а в методе спуска вычисляется еще и значение производной. С другой стороны, метод спуска сходится быстрее, если известны начальные приближения корней. Программа, реализующая этот метод, представлена в задаче 4.10.

4.9. Решение нелинейных уравнений в Mathcad

Рассмотрим возможности численного и символьного решения нелинейных уравнений средствами Mathcad.

4.9.1. Алгебраические уравнения

Решить *алгебраическое уравнение в Mathcad* можно при помощи двух встроенных функций:

- $\text{root}(F(x), x)$ – возвращает с заданной точностью значение переменной x , при котором выражение $F(x)$ равно нулю, функция реализует вычисление итерационным методом, и перед ее применением необходимо задать *начальное значение* переменной x , принадлежащее интервалу изоляции корня;
- $\text{polyroots}(v)$ – возвращает вектор всех корней (как вещественных так и комплексных) полинома n -й степени, коэффициенты которого хранятся в массиве v , длиной $n + 1$.

Решим несколько алгебраических уравнений.

ЗАДАЧА 4.1. Найти корни полинома $2x^4 - 8x^3 + 8x^2 - 1 = 0$.

Решим это уравнение при помощи функции $\text{polyroots}(v)$. Массив коэффициентов, используемый в этой функции, определим как вектор-столбец из пяти элементов. Обратите внимание, что в уравнении отсутствует переменная x в первой степени. Это означает, что соответствующий коэффициент равен нулю. Полностью решение задачи приведено на рис. 4.26.

Вектор коэффициентов

$$v := \begin{pmatrix} -1 \\ 0 \\ 8 \\ -8 \\ 2 \end{pmatrix}$$

Корни полинома

$$\text{polyroots}(v) = \begin{pmatrix} -0.307 \\ 0.459 \\ 1.541 \\ 2.307 \end{pmatrix}$$

Рис. 4.26 ▼ Вычисление корней полинома

Найдем графическое решение заданного уравнения. Для чего создадим функцию $F(x)$, определив полином как сумму произведений коэффициентов на x в соответствующей степени, и построим ее график. Точки пересечения графика с осью абсцисс и будут корнями уравнения. На рис. 4.27 видно, что графическое решение совпадает с аналитическим.

$$F(x) := \sum_{i=0}^4 (V_i \cdot x^i) \quad \text{Полином: } V_0 + V_1 \cdot x + V_2 \cdot x^2 + V_3 \cdot x^3 + V_4 \cdot x^4$$

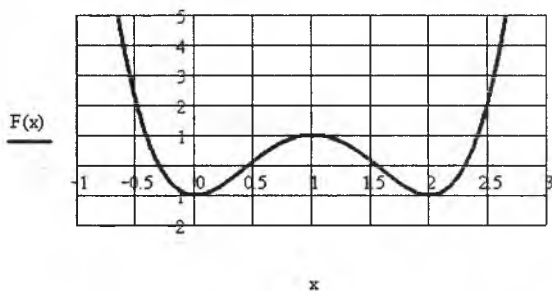


Рис. 4.27 ▼ Графическое решение уравнения

ЗАДАЧА 4.2. Найти решение уравнения $x^3 + 0,4x^2 + 0,6x - 1 = 0$.

Решение этой задачи практически ничем не отличается от решения предыдущей. Разница заключается в способе задания массива коэффициентов для дальнейшего использования в функции `polyroots(v)`. Остановимся на этом вопросе подробнее.

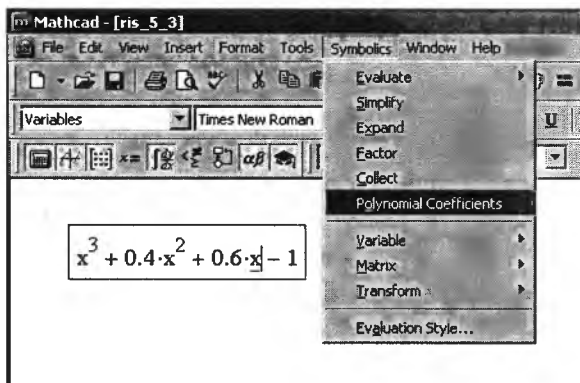


Рис. 4.28 ▼ Вызов команды, определяющей коэффициенты полинома

Итак, определить вектор коэффициентов многочлена можно при помощи символьной операции **Polynomial Coefficient** (Коэффициенты полинома), вызываемой из меню **Symbolics** (Символика). При этом многочлен должен быть введен в рабочий лист и одна из переменных выделена (рис. 4. 28).

Далее полученный вектор необходимо присвоить какой-либо переменной и найти решение, вызвав функцию вычисления корней полинома. Результаты аналитического и графического решений приведены на рис. 4. 29.

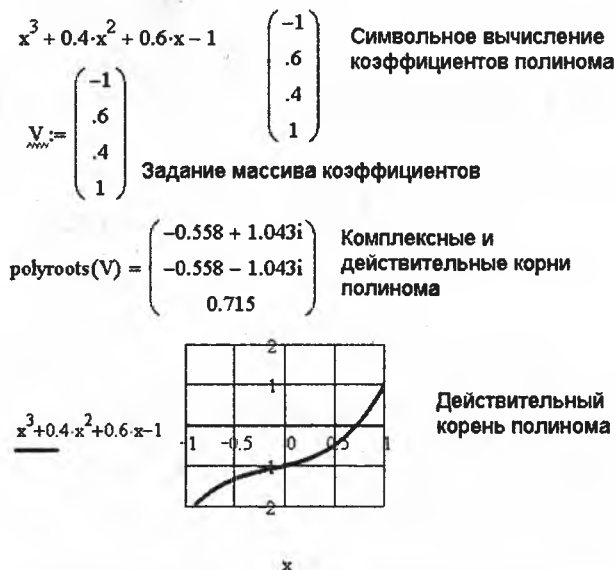


Рис. 4.29 ▼ Численное и графическое решение полинома

ЗАДАЧА 4.3. Найти решение уравнения $y(x) = 0$, если $y(x) = x^4 - 18x^2 + 6$.

В этой задаче, как и в задаче 4.2, акцент делается на метод определения коэффициентов полинома. На этот раз воспользуемся операцией **Coeffs** (Коэффициенты), которая находится на панели инструментов **Symbolics**. На рис. 4.30 видно, что эта операция содержит два поля ввода. Слева вводят многочлен, справа – неизвестную переменную, от которой он зависит.

На рис. 4.31 приведено численное и графическое решение задачи.

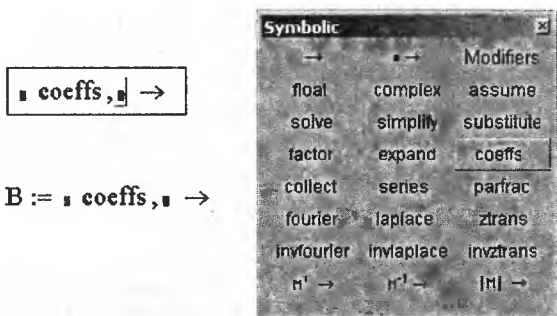


Рис. 4.30 ▼ Вызов операции определения коэффициентов полинома

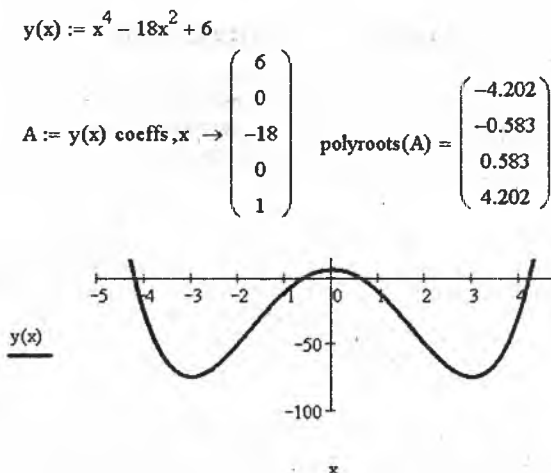


Рис. 4.31 ▼ Решение алгебраического уравнения

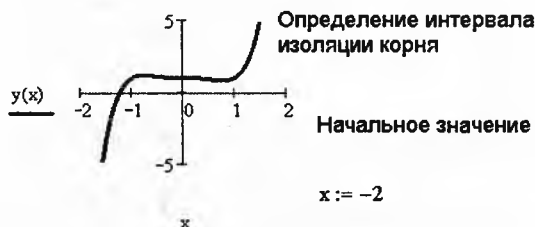
ЗАДАЧА 4.4. Найти решение уравнения $y(x) = 0$, если $y(x) = x^5 - x^3 + 1$.

Решим эту задачу при помощи функции $\text{root}(F(x), x)$. Для этого выполним следующую последовательность действий:

1. Введем функцию $y(x)$ в рабочий лист.
2. Определим интервал изоляции корня, для чего выполним построение графика функции $y(x)$.
3. Присвоим переменной x значение одного из пределов интервала изоляции, задав тем самым начальное значение.
4. Запишем функцию $\text{root}(y(x), x)$ и вычислим корень уравнения.

На рис. 4.32 приведен ход решения задачи.

$$y(x) := x^5 - x^3 + 1$$



$\text{root}(y(x), x) = -1.237$ Единственный корень полинома

Рис. 4.32 ▼ Вычисление действительного корня полинома

Заметим, что заданное уравнение кроме действительного корня имеет и мнимые. На рис. 4.33 показано их вычисление при помощи функции `polyroots(v)`.

Коэффициенты
уравнения $x^5 - x^3 + 1 = 0$

Действительные и комплексные
корни полинома $x^5 - x^3 + 1 = 0$

$$\begin{array}{lll} \underline{V_0} := 1 & \underline{V_1} := 0 & \underline{V_2} := 0 \\ \underline{V_3} := -1 & \underline{V_4} := 0 & \underline{V_5} := 1 \end{array}$$

$$\text{polyroots}(V) = \begin{pmatrix} -1.237 \\ -0.341 - 0.785i \\ -0.341 + 0.785i \\ 0.959 - 0.428i \\ 0.959 + 0.428i \end{pmatrix}$$

Рис. 4.33 ▼ Вычисление корней полинома

Как найти комплексные корни полинома, используя функцию `root(y(x), x)`, показано на рис. 4.34. При вычислении второго корня x_2 первый исключается делением функции $y(x)$ на $(x - x_1)$. Для определения третьего корня функция $y(x)$ делится на $(x - x_1)(x - x_2)$ и так далее. Заметим, что авторам подобная процедура вычисления корней кажется громоздкой.

Далее будет рассмотрено применение функции `root(F(x), x)` для решения неалгебраических уравнений.

$x1 := -2$ Действительный корень
 $i := \sqrt{-1}$ $x := 1 + 1 \cdot i$ Комплексное начальное значение
 Комплексные корни
 $x2 := \text{root}\left(\frac{y(x)}{x - x1}, x\right)$ $x2 = 0.959 + 0.428i$
 $x3 := \text{root}\left[\frac{y(x)}{(x - x1) \cdot (x - x2)}, x\right]$ $x3 = -0.341 - 0.785i$
 $x4 := \text{root}\left[\frac{y(x)}{(x - x1) \cdot (x - x2) \cdot (x - x3)}, x\right]$ $x4 = -0.341 + 0.785i$
 $x5 := \text{root}\left[\frac{y(x)}{(x - x1) \cdot (x - x2) \cdot (x - x3) \cdot (x - x4)}, x\right]$ $x5 = 0.959 - 0.428i$

Рис. 4.34 ▼ Вычисление мнимых корней полинома

4.9.2. Трансцендентные уравнения

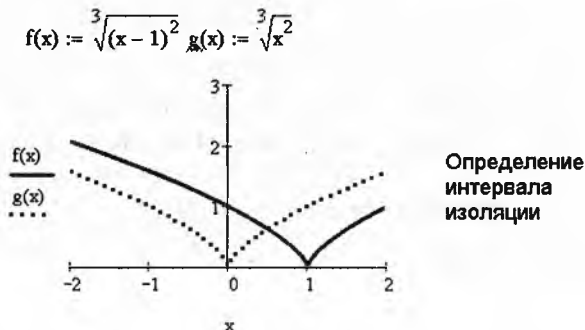
Рассмотрим применение функции $\text{root}(F(x), x)$ для решения трансцендентных уравнений.

ЗАДАЧА 4.5. Найти решение уравнения:

$$\sqrt[3]{(x-1)^2} - \sqrt[3]{x^2} = 0,$$

Выражение, стоящее в правой части уравнения, можно представить разности двух функций $f(x) - g(x) = 0$, где

$$f(x) = \sqrt[3]{(x-1)^2}, \quad g(x) = \sqrt[3]{x^2}.$$



$x := 0$ Начальное значение

$\text{root}(f(x) - g(x), x) = 0.5$ Решение уравнения

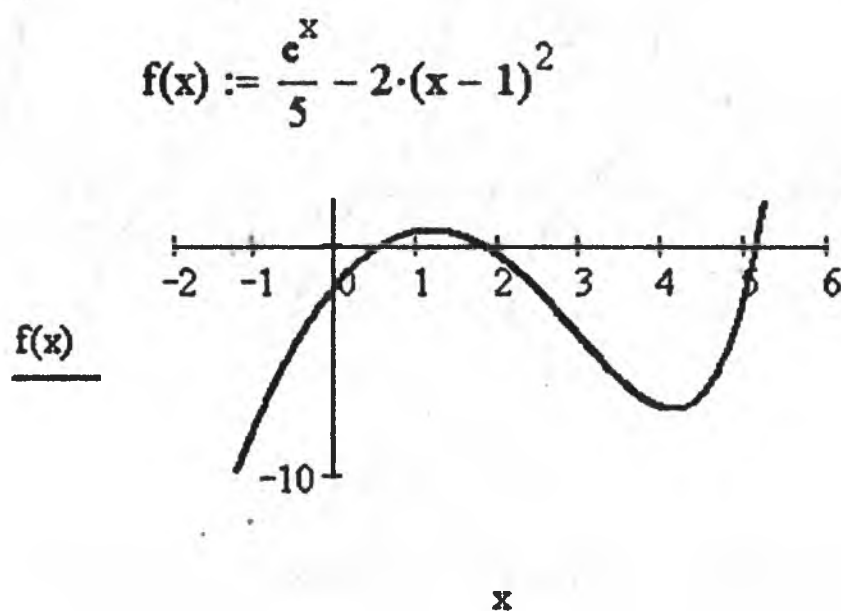
Рис. 4.35 ▼ Решение трансцендентного уравнения

Следовательно, $f(x) = g(x)$, а точка, в которой графики этих функций пересекаются, является графическим решением уравнения. Итак, для решения заданного уравнения (рис. 4.35) выполним следующую последовательность действий:

1. Введем функции $f(x)$ и $g(x)$.
2. Определим интервал изоляции корня.
3. Зададим начальное значение x .
4. Найдем решение при помощи функции `root(f(x)-g(x), x)`.

ЗАДАЧА 4.6. Найти корни уравнения $f(x) = 0$.

На рис. 4.36 видно, что график функции $f(x)$ трижды пересекает ось абсцисс, то есть уравнение имеет три корня. Для решения этой задачи воспользуемся тем, что функцию `root(f(x), x)` можно записать в виде `root(F(x), x, a, b)`, где a и b – пределы интервала изоляции корня. Понятно, что при такой форме записи функции нет необходимости задавать начальное значение x , так как оно определено в интервале $[a; b]$.



$$\text{root}(f(x), x, 0, 1) = 0.578$$

$$\text{root}(f(x), x, 1, 2) = 1.764$$

$$\text{root}(f(x), x, 5, 6) = 5.148$$

$$\text{root}(f(x), x, 0, 2) = \text{■}$$

The function values on the end points must be of opposite signs.

Рис. 4.36 ▼ Вычисление корней трансцендентного уравнения

Обратите внимание на последнее обращение к функции `root` на рис. 4.36. Mathcad выдал сообщение об ошибке: **Значения на обоих концах интервала должны иметь противоположные знаки.** Произошло это потому, что интервал изоляции задан неверно. На графике видно, что на концах этого интервала функция знак не меняет.

ЗАДАЧА 4.7. Вычислить десять положительных корней уравнения $1/\operatorname{tg}(x) - x/2 = 0$.

На рис. 4.37 изображено графическое решение уравнения. Первая точка пересечения графиков функций $1/\operatorname{tg}(x)$ и $x/2$ находится в интервале $(0; \pi/2)$, вторая – $(\pi/2; 3\pi/2)$, третья – $(3\pi/2; 5\pi/2)$ и т.д. Следовательно, интервалы изоляции корней заданного уравнения в общем виде можно записать так: $((k-1)\pi; k\pi + \pi/2)$, где k – целое число.

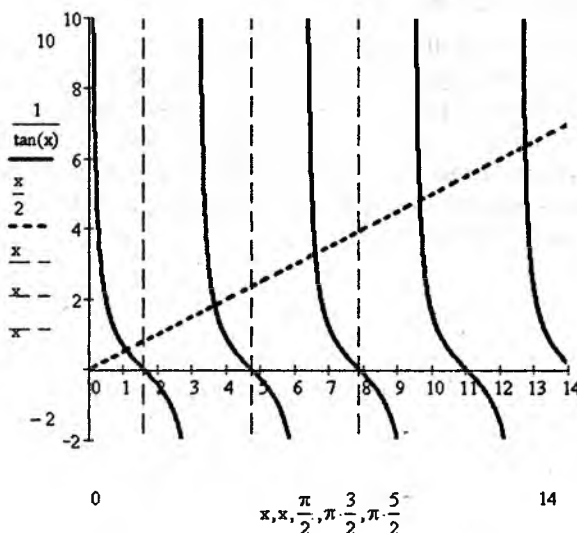


Рис. 4.37 ▽ Графическое решение уравнения $1/\operatorname{tg}(x) - x/2 = 0$

Аналитическое решение уравнения $1/\operatorname{tg}(x) - x/2 = 0$ приведено на рис. 4.38. Обратите внимание на то, как заданы интервалы изоляции корней. Так как функция $1/\operatorname{tg}(x)$ не существует в точках $(k-1) \cdot \pi$, к ним будем прибавлять заведомо очень маленькое число.

$k := 1..10$

$\text{root}\left[\frac{1}{\tan(x)} - \frac{x}{2}, x, (k-1) \cdot \pi + 0.0001, k \cdot \pi + \frac{\pi}{2}\right]$

1.077
3.644
6.578
9.63
12.722
15.834
18.955
22.081
25.212
28.345

Рис. 4.38 ▽ Численное решение уравнения $1/\operatorname{tg}(x) - x/2 = 0$

4.9.3. Реализация численных методов решения нелинейных уравнений

Воспользуемся возможностями программирования в пакете Mathcad и решим несколько задач некоторыми численными методами, описанными в этой главе.

ЗАДАЧА 4.8. Решить уравнение из задачи 4.4 методом половинного деления и методом хорд.

Из рис. 4.32 видно, что действительный корень уравнения находится в интервале $[-2; -1]$. Кроме того, нам известно более точное решение $x = -1,237$, найденное при помощи встроенной функции Mathcad. Составим Mathcad-программу (рис. 4.39), воспользовавшись алгоритмом метода дихотомии (рис. 4.5), и убедимся, что поиск корня даст тот же результат за 10 шагов.

$$f(x) := x^5 - x^3 + 1$$

$$\text{dihot}(a, b, \varepsilon) := \left(\begin{array}{l} k \leftarrow 0 \\ \text{while } 1 \\ \quad c \leftarrow \frac{a+b}{2} \\ \quad b \leftarrow c \text{ if } f(c) \cdot f(a) < 0 \\ \quad a \leftarrow c \text{ otherwise} \\ \quad k \leftarrow k + 1 \\ \quad \text{break if } |a - b| < \varepsilon \\ c \\ k \end{array} \right)$$

$$\text{dihot}(-2, -1, 0.001) = \left(\begin{array}{l} -1.237 \\ 10 \end{array} \right)$$

Рис. 4.39 ▼ Mathcad-программа решения нелинейного уравнения методом дихотомии

Реализация метода хорд, представленного в виде алгоритма на рис. 4.7, приведена на рис. 4.40.

ЗАДАЧА 4.9. Найти корни уравнения $e^x/5 - 2 \cdot (x - 1)^2 = 0$ методом касательных (блок-схема на рис. 4.10) и комбинированным методом (блок-схема на рис. 4.15).

Заданное уравнение уже рассматривалось в задаче 4.6, и нам известны из графического решения (рис. 4.36) интервалы изоляции его трех корней. На рис. 4.41 приведена Mathcad-программа для метода касательных (Ньютона), а на рис. 4.42 – для комбинированного метода. Нетрудно заметить, что оба метода достаточно быстро приводят к искомому решению.

$$f(x) := x^5 - x^3 + 1$$

```

horda(a,b,ε) :=
    k ← 0
    while 1
        c ← a -  $\frac{f(a)}{f(b) - f(a)} \cdot (b - a)$ 
        b ← c if f(c)·f(a) < 0
        a ← c otherwise
        k ← k + 1
        break if |f(c)| < ε
     $\begin{pmatrix} c \\ k \end{pmatrix}$ 

horda(-2,-1,0.001) =  $\begin{pmatrix} -1.2364 \\ 30 \end{pmatrix}$ 

```

Рис. 4.40 ▼ Mathcad-программа решения нелинейного уравнения методом хорд

$$f(x) := \frac{e^x}{5} - 2 \cdot (x-1)^2 \quad f_1(x) := \frac{d}{dx} f(x) \quad f_2(x) := \frac{d}{dx} f_1(x)$$

```

newton(a,b,ε) :=
    k ← 0
    c ← a if f(a)·f2(a) > 0
    c ← b otherwise
    while 1
        c ← c -  $\frac{f(c)}{f_1(c)}$ 
        k ← k + 1
        break if |f(c)| < ε
     $\begin{pmatrix} c \\ k \end{pmatrix}$ 

```

$$\text{newton}(0,1,0.001) = \begin{pmatrix} 0.578 \\ 3 \end{pmatrix}$$

$$\text{newton}(1,2,0.001) = \begin{pmatrix} 1.764 \\ 3 \end{pmatrix}$$

$$\text{newton}(5,6,0.001) = \begin{pmatrix} 5.148 \\ 4 \end{pmatrix}$$

Рис. 4.41 ▼ Mathcad-программа для метода Ньютона

```

combo(a,b,ε) :=
  k ← 0
  x ← a
  y ← b
  while 1
    x ← x -  $\frac{f(x)}{f(y) - f(x)} \cdot (y - x)$ 
    y ← y -  $\frac{f(y)}{f'(y)}$ 
    k ← k + 1
    break if |y - x| < ε
  x ←  $\frac{x + y}{2}$ 
  (
    x
    k
  )

```

$\text{combo}(0,1,0.001) = \begin{pmatrix} 0.578 \\ 4 \end{pmatrix}$
 $\text{combo}(1,2,0.001) = \begin{pmatrix} 1.764 \\ 3 \end{pmatrix}$
 $\text{combo}(5,6,0.001) = \begin{pmatrix} 5.148 \\ 4 \end{pmatrix}$

Рис. 4.42 ▼ Mathcad-программа для комбинированного метода

ЗАДАЧА 4.10. Решить трансцендентное уравнение $x - \sin(x) = 0,25$ методом простой итерации.

Будем решать поставленную задачу следующим образом (рис. 4.43):

1. Определим функцию $f(x)$ и построим ее график.
2. Найдем начальное приближение корня $x_0 = 1,2$, решив задачу графически.
3. Вычислим первую производную $f'(x)$ функции $f(x)$.
4. Перейдем к уравнению $x = \varphi(x) = x - \lambda f(x)$, для чего вычислим $\lambda > 0$ из неравенства $0 \leq 1 - \lambda f'(x) < 1$.
5. Так как $\lambda \in (0; 1,57)$, положим $\lambda = 1,5$ и определим функцию $\varphi(x) = x - \lambda f(x)$.
6. Составим программу в соответствии с алгоритмом, изображенным на рис. 4.18.
7. Уточним корень и убедимся, что итерационный процесс сошелся достаточно быстро и найденное решение совпадает с решением встроенной функции.

ЗАДАЧА 4.11. Найти корни уравнения $e^z - 0,2 \cdot z + 1 = 0$.

На рис. 4.44 изображен график функции $f(z) = e^z - 0,2 \cdot z + 1$. Нетрудно определить, что заданное уравнение действительных корней не имеет, так как график функции $f(z)$ не пересекает ось абсцисс.

$$f(x) := x - \sin(x) - 0.25 \quad x_0 := 1.2$$

$$f_1(x) := \frac{d}{dx} f(x) \quad f_1(x_0) = 0.638$$

$$1 - \lambda \cdot 0.638 < 1 \text{ solve, } \lambda \rightarrow 0 < \lambda$$

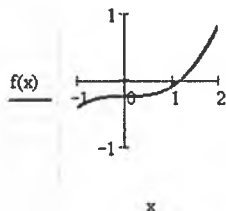
$$1 - \lambda \cdot 0.638 \geq 0 \text{ solve, } \lambda \rightarrow \lambda \leq 1.5673981191222570533$$

$$\lambda := 1.5 \quad \phi(x) := x - \lambda \cdot f(x)$$

```

iter(x1, ε) := | k ← 0
                while 1
                    x0 ← x1
                    x1 ← φ(x0)
                    k ← k + 1
                    break if |x0 - x1| < ε
                ( x1
                  k )

```



$$\text{iter}(x_0, 0.001) = \begin{pmatrix} 1.171 \\ 3 \end{pmatrix}$$

$$x := x_0 \quad \text{root}(f(x), x) = 1.171$$

Рис. 4.43 ▼ Mathcad-программа для метода простой итерации

$$f(z) := e^z - 0.2 \cdot z + 1$$

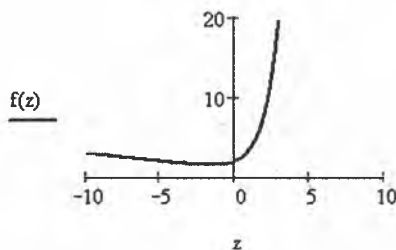


Рис. 4.44 ▼ Графическое решение задачи 4.44, поиск действительных корней

Попытаемся определить комплексные корни заданного уравнения. Для локализации комплексного корня приведем уравнение $e^z - 0.2 \cdot z + 1 = 0$ к виду (4.2):

$$(e^{x+iy} - 0.2 \cdot (x + iy) + 1 = 0) \Rightarrow (e^x(\cos(y) + i \cdot \sin(y)) - 0.2 \cdot (x + iy) + 1 = 0) \Rightarrow$$

$$\Rightarrow e^x \cdot \cos(y) - x/5 + 1 + i \cdot (e^x \cdot \sin(y) - y/5).$$

Графики функций $\phi(x, y) = e^x \cdot \cos(y) - x/5 + 1 = 0$ и $\psi(x, y) = e^x \cdot \sin(y) - y/5$ представлены на рис. 4.45. Так как они пересекаются, то комплексный корень существует и приблизительно равен $z_0 = 0,1 + 2,5i$.

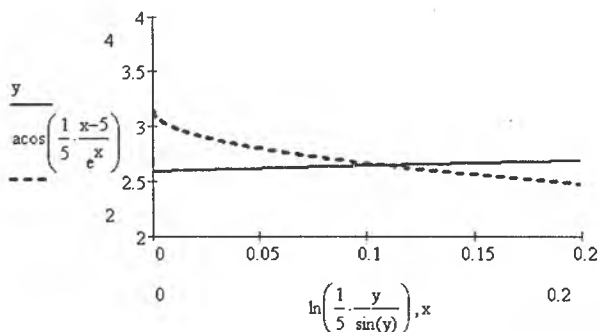


Рис. 4.45 ▼ Графическое решение задачи 4.44, поиск комплексных корней

Составим программу уточнения комплексного корня по методу Ньютона так, как было описано в разделе 4.6. Решение поставленной задачи для случая комплексных корней приведено на рис. 4.46.

```

komplex(z, ep) :=  $\left\{ \begin{array}{l} k \leftarrow 0 \\ \text{while } 1 \\ \quad \left\{ \begin{array}{l} z \leftarrow z - \frac{f(z)}{f'(z)} \\ k \leftarrow k + 1 \\ \text{break if } |f(z)| < ep \end{array} \right. \\ \left( \begin{array}{l} z \\ k \end{array} \right) \end{array} \right.$ 
 $\begin{array}{l} i := \sqrt{-1} \\ z := 0.1 + 2.5 \cdot i \end{array}$ 
komplex(z, 0.01) =  $\left( \begin{array}{l} 0.107 + 2.646i \\ 2 \end{array} \right)$ 

```

Рис. 4.46 ▼ Mathcad-программа метода Ньютона для вычисления комплексных корней

4.10. Использование Maple для решения нелинейных уравнений

Достаточно мощными средствами для решения нелинейных уравнений обладает и пакет Maple. Рассмотрим основные функции.

Для решения уравнений в аналитическом виде используется функция `solve(<уравнение>, <переменная>)`, где <уравнение> – это уравнение, содержащее функцию, <переменная> – переменная, по которой необходимо найти реше-

ние. Возможны различные варианты записи этой функции, например запись $\text{solve}(F(x), x)$ означает, что решается уравнение $f(x) = 0$, а в случае $\text{solve}(F(x), G(x), x)$ решению относительно x подлежит выражение $f(x) = g(x)$.

Функцию $\text{fsolve}(\langle \text{уравнение} \rangle, \langle \text{переменная} \rangle, \langle \text{опции} \rangle)$ применяют для получения численного решения нелинейного уравнения в форме вещественных чисел. Параметры $\langle \text{уравнение} \rangle$ и $\langle \text{переменная} \rangle$ имеют те же значения, что в функции solve , а параметр $\langle \text{опции} \rangle$ может принимать следующие значения

- **complex** – находит один или все корни полинома в комплексной форме;
- **folldigits** – задает вычисления для полного числа цифр, заданного функцией Digits^1 ;
- **maxsols** – задает вычисление только n корней полинома;
- **interval** – обеспечивает поиск корней в указанном интервале и принимает значения $a..b$ или $x = a..b$.

Кроме того, в Maple встроены достаточно специфические функции решения нелинейных уравнений, такие как rsolve – решение рекуррентных уравнений, isolve – решение уравнений в целочисленном виде, msolve – решение уравнений по модулю m и специальная функция вычисления корней полинома $\text{root}(\langle \text{полином} \rangle)$, результатом работы которой является список $[[r_1, m_1], \dots, [r_n, m_n]]$, где r_i – корень полинома, а m_i – его кратность. Мы остановимся на основной функции solve и fsolve и рассмотрим их применение на конкретных примерах.

4.10.1. Алгебраические уравнения

Рассмотрим, как работают описанные функции при решении алгебраических уравнений.

ЗАДАЧА 4.12. Найти корни полинома $2x^4 - 8x^3 + 8x^2 - 1 = 0$.

Это уравнение уже было решено в задаче 4.1 средствами Mathcad, и мы знаем, что оно имеет четыре действительных корня. На рис. 4.47 показано, как

$$\begin{aligned} & \left[\begin{array}{l} > \text{solve}(2*x^4-8*x^3+8*x^2-1, x); \\ > \text{fsolve}(2*x^4-8*x^3+8*x^2-1, x); \end{array} \right. \\ & \qquad \qquad \qquad 1 - \frac{\sqrt{4+2\sqrt{2}}}{2}, 1 + \frac{\sqrt{4+2\sqrt{2}}}{2}, 1 - \frac{\sqrt{4-2\sqrt{2}}}{2}, 1 + \frac{\sqrt{4-2\sqrt{2}}}{2} \\ & \qquad \qquad \qquad -0.3065629649, 0.4588038999, 1.541196100, 2.306562965 \end{aligned}$$

Рис. 4.47 ▼ Решение задачи 4.12

¹ Управляет количеством выводимых после десятичной точки цифр.

Maple справляется с этой задачей. Понятно, что функция `solve` нашла все корни полинома в аналитическом виде, а `fsolve` – в более привычном численном.

ЗАДАЧА 4.13. Найти решение уравнения $x^3 + 0,4x^2 + 0,6x - 1 = 0$.

Этот полином имеет один действительный и два комплексных сопряженных корня (см. задачу 4.2). Из рис. 4.48 видно, что функция `solve`, в отличие от `fsolve`, выводит все корни полинома, в том числе и комплексные.

```
> solve(x^3+0.4*x^2+0.6*x-1,x);
      -0.5576818009 - 1.042536106 I, -0.5576818009 + 1.042536106 I, 0.7153636019
> fsolve(x^3+0.4*x^2+0.6*x-1,x);
      0.7153636019
```

Рис. 4.48 ▼ Решение задачи 4.13

ЗАДАЧА 4.14. Вычислить корни полинома $p(x)$.

Решение этой задачи выполнено при помощи функции `roots` и приведено на рис. 4.49. Здесь в качестве результата получены не только корни уравнения, но и их кратность.

```
> p:=(x-1)*(x-1)*(x+3)*(x+3)*(x+3)*(x-4)*(2*x+1);
      p = (x - 1)^2 (x + 3)^3 (x - 4) (2 x + 1)
> roots(p);
      [[1, 2], [4, 1], [-1/2, 1], [-3, 3]]
```

Рис. 4.49 ▼ Решение задачи 4.14

4.10.2. Трансцендентные уравнения

Приведем несколько примеров решения трансцендентных уравнений с помощью функций `solve` и `fsolve`.

ЗАДАЧА 4.15. Решить трансцендентное уравнение $e^x/5 - 1/2 = 0$.

На рис. 4.50 приведено графическое, символьное и численное решение этой задачи.

ЗАДАЧА 4.16. Найти корни уравнения $e^z - 0,2 \cdot z + 1 = 0$.

В задаче 4.11 было доказано, что это уравнение не имеет действительных корней. На рис. 4.51 можно увидеть, что функция `solve` пытается вычислить аналитический корень, а `fsolve` выдает сообщение об отсутствии корней заданного уравнения.

```
> solve(exp(x)/5-1/2,x);
```

$$\ln\left(\frac{5}{2}\right)$$

```
> fsolve(exp(x)/5-1/2,x);
```

0.9162907319

```
> plot(exp(x)/5-1/2,x=-1..2,y=-1..1,color=black);
```

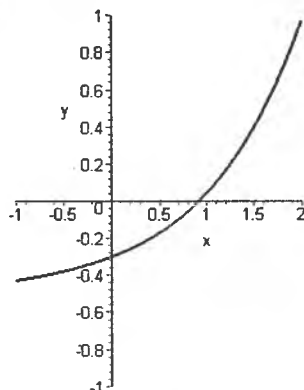


Рис. 4.50 ▼ Решение задачи 4.15

```
> solve(e^z-0.2*z+1,z);
```

$$\frac{-1 \cdot \text{LambertW}(-5 \cdot \ln(e) e^5) + 5 \cdot \ln(e)}{\ln(e)}$$

```
> fsolve(e^z-0.2*z+1,z);
```

Error, (in fsolve) e is in the equation, and is not solved for

Рис. 4.51 ▼ Решение задачи 4.16

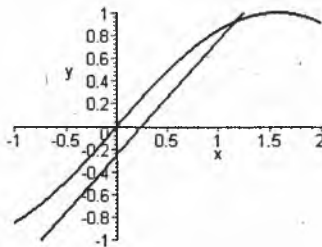
ЗАДАЧА 4.17. Найти решение уравнения $x - \sin(x) = 0,25$.

Для решения задачи уравнение удобно представить в виде $\sin(x) = x - 0,25$. На рис. 4.52 представлено графическое, символьное и численное решение, причем в этот раз функции solve и fsolve выдали одинаковый результат. Однако если уравнение заменить тождественным ему $\sin(x) = x - 1/4$, то численное решение функции fsolve останется прежним, а аналитическое решение функции solve будет записано через функцию RootOf, которая означает, что корни нельзя выразить в радикалах.


```

> f(x):=sin(x):g(x):=x-0.25:
> solve(f(x)=g(x),x);
1.171229652
> fsolve(f(x)=g(x),x);
1.171229653
> plot([f(x),g(x)],x=-1..2,y=-1..1,color=black);

```



```

> g(x):=x-1/4:solve(f(x)=g(x),x);fsolve(f(x)=g(x),x);
RootOf(4*_Z-4*sin(_Z)-1)
1.171229653

```

Рис. 4.52 ▼ Решение задачи 4.17

ЗАДАЧА 4.18. Вычислить десять положительных корней уравнения $1/x - x/2 = 0$.

Идея поиска десяти корней уравнения была изложена в задаче 4.7. Приведем только ее реализацию в пакете Maple (рис. 4.53).

```

> i:=0:for k from 1 to 10 do
> a:=(k-1)*3.14;b:=k*3.14+3.14/2;i:=i+1;
> y[i]:=fsolve(1/tan(x)-x/2,x,x=a..b);
> print(eval(y[i]));
> od:

```

3.643597167
6.578333733
9.629560343
12.72229877
15.83361141
18.95468177
22.08147577
25.21190306
28.34477687
31.47937492

Рис. 4.53. Решение задачи 4.18

4.10.3. Реализация численных методов решения алгебраических уравнений

Как известно, Maple обладает собственным достаточно мощным языком программирования. Опишем при помощи Maple-языка численные методы решения алгебраических уравнений.

ЗАДАЧА 4.19. Решить алгебраические уравнения из 4.4 методом спуска.

Как было сказано в разделе 4.7.4, реализация метода спуска требует определения значения многочлена в точке (см. раздел 4.7.1), вычисления частного от деления многочлена на линейный множитель по схеме Горнера (см. раздел 4.7.2) и расчета значения первой производной многочлена в точке (см. раздел 4.7.3). Приведем тексты соответствующих процедур на Maple-языке.

Рис. 4.54 содержит процедуру вычисления значения многочлена в точке (алгоритм приведен на рис. 4.19). В качестве тестового примера показано вычисление значения полинома $y(x) = x^5 - x^3 + 1$ в точке $x_0 = 1$.

```
> pol:=proc(n,a,x0) #Значение полинома степени n в точке x0
> local i::integer;
> s:=0;
> for i from 0 to n do
> s:=s+a[i]*x0^(n-i)
> od;
> evalf(s);
> end;
Warning, 's' is implicitly declared local to procedure 'pol'

> w:=5;q:=array(0..w,[1,0,-1,0,0,1]):pol(w,q,1);
>
1.
```

Рис. 4.54 ▼ Процедура вычисления значения полинома в точке

```
> gorner:=proc(n,a,x0) #Деление многочлена на (x-x0) по схеме Горнера
> local i::integer;
> b[0]:=a[0];
> for i from 1 to n-1 do
> b[i]:=a[i]+b[i-1]*x0 od;
> eval(b);
> end;
Warning, 'b' is implicitly declared local to procedure 'gorner'

> gorner(w,q,-1.237);
table([0 = 1, 1 = -1.237, 2 = 0.530169, 3 = -0.655819053, 4 = 0.8112481686])
```

Рис. 4.55 ▼ Процедура деления многочлена на линейный множитель

Процедура вычисления частного от деления многочлена на линейный множитель по схеме Горнера изображена на рис. 4.55 (блок-схема на рис. 4.21). Здесь многочлен $y(x) = x^5 - x^3 + 1$ делится на $(x + 1,237)$, результатом являются коэффициенты нового многочлена меньшей степени.

Текст процедуры, содержащийся на рис. 4.56, представляет собой программную реализацию алгоритма вычисления значения производной в точке (рис. 4.22). В данном случае вычисляется значение первой производной многочлена $y(x) = x^5 - x^3 + 1$ в точке.

```
> proizv:=proc(n,a,m,x0)
> local i,k::integer; global b;
> for i from 0 to n do
> b[-1,i]:=a[i] od;
> for k from 0 to m do
> b[k,0]:=b[k-1,0];
> for i from 1 to n-k do
> b[k,i]:=b[k-1,i]+b[k,i-1]*x0 od;
> od;
> Pr:=1;
> for i from 2 to m do
> Pr:=Pr*i od;
> Pr:=(Pr*b[m,n-m]);
> Pr;
> end:
Warning, 'Pr' is implicitly declared local to procedure 'proizv'

> proizv(w,q,1,1);
```

2

Рис. 4.56 ▼ Процедура вычисления производной в точке $x_0 = 1$

В следующей процедуре отражено вычисление корня алгебраического уравнения по методу спуска (рис. 4.57).

И в итоге на рис. 4.58 представлена основная процедура, которая позволяет найти все корни уравнения. Как видим, вызов процедуры для заданного уравнения дает и комплексные, и действительные корни, причем они полностью совпадают с расчетами, проведенными средствами Mathcad в задаче 4.4.

```

> pol_nol:=proc(n,a,ep)
> local i,gamma,alfa,betta,h,z0,t;
> gamma:=-a[1]/a[0];
> t:=pol(n,a,gamma);
> z1:=evalf(gamma+I*(abs(t/a[0]))^(1/n));
> while true do
> z0:=z1;
> alfa:=pol(n,a,z0);
> betta:=evalf(proizv(n,a,1,z0));
> h:=-alfa/betta;
> z1:=z0+h;
> if abs(h)<ep then break;fi;
> od;
> evalf(z1);
> end:

```

Рис. 4.57 ▽ Процедура вычисления корня многочлена по методу спуска

```

> korni:=proc(n,a,ep)
> local j,y; global m,c,d,x,k,r;
> c:=a;m:=n;x:=n;
> for k from 1 to r do
> y:=evalf(pol_nol(m,c,ep));
> x[k]:=y;
> d:=eval(gornex(m,c,y));
> m:=m-1;
> for j from 0 to m do
> c[j]:=aval(f(d[j])); od;
> od;
> for i from 1 to r do
> if abs(Im(x[i]))<ep then x[i]:=Re(x[i]) fi;od;
> RETURN(eval(x));
> end:
>

```

Warning, 'i' is implicitly declared local to procedure 'korni'

```

> korni(w,q,0.01);
table([1 = -0.3407957372 + 0.7854222561 I, 2 = 0.9590620591 - 0.4284231883 I, 3 = -1.236490817, 4 = 0.9590683283 + 0.4284
5 = -0.3408438332 - 0.7854333389 I

```

Рис. 4.58 ▽ Процедура вычисления корней многочлена (основная процедура)

ЗАДАЧА 4.20. Решить алгебраические уравнения из 4.13 методом парабол (см. раздел 4.7.5).

Решение этой задачи основано на алгоритме, изображенном на рис. 4.25, и состоит из трех процедур. Первая процедура отражает схему Горнера. Она представлена на рис. 4.59.

```
> w:=3;q:=array(0..w,[10,4,6,-10]):
[
> gorner:=proc(n,a,x0) #Схеме Горнера
> local i::integer;
> b[0]:=a[0];
> for i from 1 to n do
> b[i]:=a[i]+b[i-1]*x0 od;
> eval(b);
> end:
Warning, 'b' is implicitly declared local to procedure 'gorner'
> gorner(w,q,4.202);

table([0 = 10, 1 = 46.020, 2 = 199.376040, 3 = 827.7781201])
```

Рис. 4.59 ▼ Процедура, реализующая схему Горнера

```
> parabol:=proc(n,a,ep)
> local m,h,Lambda,sigma,alfa,betta,g,p1,p2,p3;
> z[0]:=-1.;z[1]:=1.;z[2]:=0.;h[1]:=z[1]-z[0];m:=2;
> while true do
> h[m]:=z[m]-z[m-1];
> Lambda[m]:=h[m]/h[m-1];
> sigma[m]:=1+Lambda[m];
> p1:=gorner(n,a,z[m-2]);
> p2:=gorner(n,a,z[m-1]);
> p3:=gorner(n,a,z[m]);
> g[m]:=p1[n]*Lambda[m]^2-p2[n]*sigma[m]^2+p3[n]*(Lambda[m]+sigma[m]);
> alfa[m]:=p1[n]*Lambda[m]-p2[n]*sigma[m]+p3[n];
> betta[m]:=(g[m]^2-4*p3[n]*sigma[m]*Lambda[m]*alfa[m])^(1/2);
> if abs(g[m]+betta[m])>abs(g[m]-betta[m]) then
> Lambda[m+1]:=(-2*p3[n]*sigma[m])/(g[m]+betta[m])
> else
> Lambda[m+1]:=(-2*p3[n]*sigma[m])/(g[m]-betta[m]) fi;
> z[m+1]:=z[m]+Lambda[m+1]*h[m];
> m:=m+1;
> if abs((z[m]-z[m-1])/z[m])<ep then break fi;
> od;
> z[m];
> end:
```

Рис. 4.60 ▼ Процедура вычисления корня многочлена методом парабол

Вторая процедура реализует поиск корня алгебраического уравнения методом парабол (рис. 4.60).

Обе эти процедуры используются в основной, которая и вычисляет все корни многочлена (рис. 4.61).

```
> poly:=proc(n,a,ep)
> local k,c,b,i;
> c:=a;
> k:=n;
> for i from 1 to k do
> x[i]:=parabol(k-i+1,c,ep);
> b:=gorner(k-i+1,c,x[i]);
> c:=b;
> od;
> eval(x);
> end;
Warning, 'x' is implicitly declared local to procedure 'poly'
```

```
> poly(w,q,0.001);
```

```
printlevel:=1
```

```
table([1 = 0.7153631965, 2 = -0.5576815980 + 1.042535859 I, 3 = -0.5576815980 - 1.042535859 I])
```

Рис. 4.61 ▼ Процедура вычисления всех корней многочлена

4.11. Возможности MATLAB для решения нелинейных уравнений

Вычисление корней нелинейных уравнений в MATLAB осуществляет встроенная функция `fzero`. Для решения алгебраических уравнений применяют функцию `roots`. Рассмотрим несколько примеров решения нелинейных уравнений с использованием этих функций.

ЗАДАЧА 4. 21. Вычислить корни полинома $x^3 + 0,4x^2 + 0,6x - 1 = 0$.

Полином в MATLAB определяется вектором его коэффициентов. Следует помнить, что коэффициенты в векторе должны располагаться от большей степени к меньшей и наличие нулевых коэффициентов обязательно. Решение задачи в MATLAB приведено в листинге 4.1.

Листинг 4.1

```
>> p=[1 0.4 0.6 -1];
>> x=roots(p);
>> x
x =
-0.5577 + 1.0425i
```

```
-0.5577 - 1.0425i
0.7154
```

ЗАДАЧА 4. 22. Найти решение уравнения $x - \sin(x) = 0,25$.

Для решения этой задачи воспользуемся функцией `fzero('name', x0)`, где `name` – имя файл-функции, вычисляющей левую часть уравнения, `x0` – начальное приближение к корню. Определим файл-функцию так, как показано в листинге 4.2:

Листинг 4.2

```
function y=f(x)
y=x-sin(x)-0.25;
```

А затем решим уравнение (листинг 4.3).

Листинг 4.3

```
>> fzero("f",1)

ans =

    1.1712
```

Сравнивая полученные результаты с решением этого же уравнения в задаче 4.17, убеждаемся в правильности решения.

ЗАДАЧА 4. 23. Найти корни уравнения $e^x/5 - 2 \cdot (x-1)^2 = 0$.

Графическое решение этого уравнения было найдено в задаче 4.6. Поэтому нам известно, что уравнение имеет три корня. Интервалы изоляции корней также определены. Поэтому для решения поставленной задачи воспользуемся функцией `fzero('name', [a b])`, где `name` – имя файл-функции, вычисляющей левую часть уравнения, `[a b]` – интервал изоляции корня. Листинг 4.4 содержит текст файл-функции, а решение представлено в листинге 4.5.

Листинг 4.4

```
function y=h(x)
y=exp(x)/5-2*(x-1)^2;
```

Листинг 4.5

```
>> z(1)=fzero("h", [0 1]);
>> z(2)=fzero("h", [1 2]);
>> z(3)=fzero("h", [5 6]);
>> z
```

```
z =
```

```
0.5778    1.7639    5.1477
```

В листинге 4.6 приведен еще один пример работы функции `fzero`. В данном случае интервал изоляции корня задан неверно. На границах указываемого

интервала функция h не меняет знак, а смена знака – обязательное условие локализации корня.

Листинг 4.6

```
>> z(4)=fzero("h", [0 3]);
```

```
??? Error using ==> fzero
```

The function values at the interval endpoints must differ in sign.

Важной особенностью функции `fzero` является возможность выдавать не только значение корня, но и значение функции в этой точке, например, так, как показано в листинге 4.7.

Листинг 4.7

```
>> [r,ep]=fzero("h", [0 1])
```

```
r =
```

```
0.5778
```

```
ep =
```

```
-1.1102e-016
```

4.12. Советы по использованию математических пакетов при решении нелинейных уравнений

Оценивая возможности математических пакетов Mathcad, Maple и MATLAB по решению нелинейных уравнений, можно сделать следующие выводы.

Идея решения алгебраических уравнений в этих пакетах одна и та же. Вводится либо полином, либо его коэффициенты, и соответствующая функция выдает все корни, в том числе и комплексные. Стоит отметить лишь специфические особенности пакета Maple. Так, например, функция `solve` вычисляет корни полинома, в том числе комплексные, аналитически (в радикалах), а `fsolve` выдает только действительные корни в численном виде, но, с другой стороны, вывод результатов в этой функции можно регулировать различными опциями.

Трансцендентные уравнения в Mathcad, Maple и MATLAB также решить не сложно, достаточно знать либо начальное приближение корня, либо интервал его изоляции. Кроме того, стоит отметить, что в Maple и Mathcad возможно получить символьное решение уравнения, а функция MATLAB `fzero` обладает возможностью не только вычислять корень, но и рассчитывать значение функции в этой точке.

4.12. Задания для самостоятельного решения

В табл. 4.1 приведены уравнения. Для каждого уравнения нужно составить программы вычисления корней с погрешностью $\varepsilon = 0,0001$ в соответствии с указанным номером метода решения (1 – метод половинного деления, 2 – метод хорд, 3 – метод касательных, 4 – комбинированный метод, 5 – метод простой итерации, 6 – метод спуска, 7 – метод парабол). Вычислить количество итераций. Проверить работу программы, выполнив вычисления средствами математических пакетов.

Таблица 4.1 ▼ Задания для самостоятельного решения

№ Уравнение	Метод	№ Уравнение	Метод
$x - 0,2\sin(x + 0,5) = 0$	1	$x - \cos x = 0$	2
1 $x^2 - \lg(x + 2) = 0$	3	6 $e^{-x} + x^2 = 0$	4
$x^3 + 25x + 19 = 0$	6	$x^3 + 35x - 12 = 0$	7
$x^2 - 20\sin x = 0$	5	$x^2 - \cos x^2 = 6$	1
2 $\ln x + (x + 1)^3 = 0$	2	7 $(x - 1)^2 - 0,5e^x = 0$	3
$x^4 - 2x^3 - 5x^2 - 10x - 1 = 0$	7	$x^4 - 8x^3 - 10x^2 - 40x + 7 = 0$	6
$x^2 - \sin 5x = 0$	4	$\sqrt{x} - 2\cos x = 0$	5
3 $e^x + x^2 = 2$	1	8 $2 - x = \ln x$	2
$x^3 - 18x + 33 = 0$	6	$x^3 + 33x + 21 = 0$	7
$1,8x^2 - \sin 10x = 0$	3	$3x - \cos x = 1$	4
4 $2\ln x - \frac{1}{x} = 0$	5	9 $2\lg x - \frac{x}{2} = -1$	1
$x^4 - 4x^3 - 19x^2 - 26x + 1 = 0$	7	$x^4 + 4x^3 - 19x^2 - 10x + 1 = 0$	6
$\sqrt{x+1} - \frac{1}{x} = 0$	2	$\lg x - \frac{1}{x^2} = 0$	3
5 $x\ln x - 100 = 0$	4	10 $2 - xe^x = 0$	5
$x^3 - 21x - 47 = 0$	6	$x^4 + 6x^3 - 5x^2 + 24x + 2 = 0$	7

Приближенное решение систем нелинейных уравнений

В дальнейшем будем рассматривать систему

(5.1)

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix},$$
$$F = \begin{pmatrix} F_1 \\ F_2 \\ \dots \\ F_n \end{pmatrix}.$$

Тогда систему (5.1) можно представить в виде

$$F(x) = 0. \quad (5.2)$$

Воспользуемся *методом Ньютона* (последовательных приближений) для решения системы (5.2). Предположим, что для одного из изолированных корней¹ x этой системы найдено k -е приближение x^k . Тогда точный корень для (5.2) можно представить так:

$$x = x^k + \varepsilon^k, \quad (5.3)$$

где ε^k – погрешность корня.

Подставим (5.3) в векторное уравнение (5.2). Получим

$$F(x^k + \varepsilon^k) = 0. \quad (5.4)$$

Если функция $F(x)$ непрерывно дифференцируема в некоторой области, которой принадлежат x и x^k , то

$$F(x^k + \varepsilon^k) = F(x^k) + F'(x^k) \cdot \varepsilon^k = 0, \quad (5.5)$$

где

$$F'(x) = W(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \dots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \dots & \frac{\partial F_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \dots & \frac{\partial F_n}{\partial x_n} \end{pmatrix} = \left(\frac{\partial F_i}{\partial x_j} \right) -$$

матрица Якоби, причем $i, j = 1, 2, \dots, n$.

Уравнение (5.5) представляет собой линейную систему², где в роли неизвестных выступают поправки ε_i^k , а главная матрица – $W(x)$:

$$F(x^k) + W(x^k) \cdot \varepsilon^k = 0, \quad (5.5)$$

Предполагая, что определитель матрицы $W(x^k)$ не равен нулю, получим

$$\varepsilon^k = -W^{-1}(x^k) \cdot F(x^k). \quad (5.6)$$

Из чего будем иметь основную формулу метода Ньютона:

$$x^{k+1} = x^k - W^{-1}(x^k) \cdot F(x^k), \quad (k = 1, 2, \dots), \quad (5.7)$$

¹ Подробно изоляция корня описана в первом параграфе четвертой главы.

² См. главу 1.

где за x^0 можно взять грубое значение искомого корня. При практическом применении метода для решения нелинейных систем (5.2) вычисления по формуле (5.7) прекращают, когда

$$|x^{k+1} - x^k| < \varepsilon. \quad (5.8)$$

Итак, исходя из вышеизложенного, запишем алгоритм метода Ньютона:

1. Определяем начальное приближение x^0 .
2. Уточняем значение корня по формуле (5.7).
3. Если условие (5.8) выполняется, то задача решена и x^{k+1} – корни векторного уравнения (5.2), иначе переходим к п.2.

Будем считать, что функции $F(x)$ нелинейной системы (5.2) и матрица их производных $W(x)$ уже определены, тогда блок-схема алгоритма решения этой системы будет выглядеть так, как показано на рис. 5.1. Обратите внимание, что в блок-схеме не показан процесс решения линейной системы (5.6). При практическом программировании здесь можно использовать любой из методов, описанных в первой главе. Пример создания программы метода Ньютона приведен в задаче 5.5.

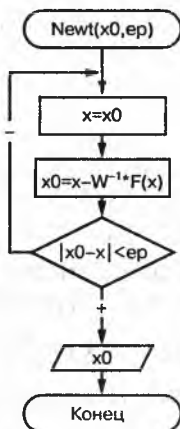


Рис. 5.1 ▼ Алгоритм метода Ньютона для решения нелинейных систем

5.2. Метод итерации

Преобразуем систему (5.1) следующим образом:

$$\begin{cases} x_1 = \varphi_1(x_1, x_2, \dots, x_n), \\ x_2 = \varphi_2(x_1, x_2, \dots, x_n), \\ \dots \\ x_n = \varphi_n(x_1, x_2, \dots, x_n), \end{cases} \quad (5.9)$$

где функции $\varphi_1, \varphi_2, \dots, \varphi_n$ – действительные, определенные и непрерывные в некоторой изолированной области, которой принадлежат решения $(x_1, x_2, \dots, x_n)$ этой системы.

Введем в рассмотрение векторы

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \quad \text{и} \quad \varphi = \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \dots \\ \varphi_n \end{pmatrix}$$

и запишем систему (5.9) в векторной форме

$$x = \varphi(x). \quad (5.10)$$

Для решения уравнения (5.10) удобно использовать *метод итерации*. Основная формула этого метода имеет вид:

$$x^{k+1} = \varphi(x^k), \quad (5.11)$$

где за $x^0 = x$ можно взять грубое значение искомого корня. Заметим, что если процесс итерации сходится [11], то предельное значение

$$x = \lim_{k \rightarrow \infty} x^k$$

обязательно является корнем уравнения (5.10), а если все приближения x^k принадлежат области изоляции корня, то x – единственный корень.

Условием окончания процесса итерации может служить неравенство (5.8).

Алгоритм метода итерации показан на рис. 5.2. Он настолько прост, что не нуждается в дополнительных пояснениях.

Если систему (5.1) записать в векторной форме

$$F(x) = 0, \quad (5.11)$$

где $F(x)$ – векторная функция, определенная и непрерывная в области изоляции корня x , то затем ее можно преобразовать и записать в виде

$$x = x + \Lambda \cdot F(x), \quad (5.12),$$

где Λ – матрица с ненулевым определителем. Теперь, введя обозначение

$$x + \Lambda \cdot F(x) = \varphi(x), \quad (5.13),$$

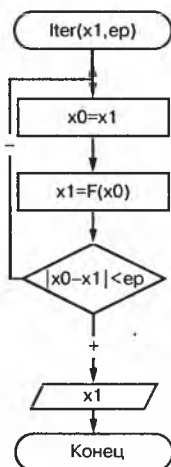


Рис. 5.2 ▽ Алгоритм метода итерации для решения нелинейных систем (5.10)

получим

$$x = \varphi(x). \quad (5.14)$$

Это означает, что метод итерации применим и к нелинейным системам, представленным в виде (5.1). Приведем без вывода формулу для вычисления матрицы Λ :

$$\Lambda = -\left(F'(x^0)\right)^{-1} = -\begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \dots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \dots & \frac{\partial F_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \dots & \frac{\partial F_n}{\partial x_n} \end{pmatrix}^{-1} = -\left(\frac{\partial F_i}{\partial x_j}\right)^{-1}. \quad (5.15)$$

Если при вычислении Λ окажется, что определитель матрицы $F'(x^0)$ равен нулю, то необходимо подобрать другое начальное приближение.

Заметим, что изложенный метод решения системы (5.11) называют *модифицированным методом Ньютона*. Опишем его алгоритм (5.3), учитывая, что условие окончания процесса вычисляется по формуле (5.8), а также то, что уравнения системы и их производные определены ранее в виде функций.

1. Выбрать начальное приближение x^0 .
2. Определить значения матрицы $\Lambda(x^0)$ по формуле (5.14).

3. Уточнить значение корня по формуле (5.12), то есть $x^{k+1} = x^k + \Lambda \cdot F(x^k)$.
4. Если условие (5.8) выполняется, то задача решена и x^{k+1} – корни векторного уравнения (5.11), иначе переходим к п.3.

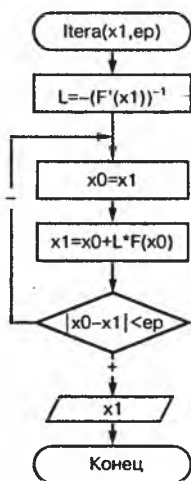


Рис. 5.3 ▾ Алгоритм модифицированного метода Ньютона

5.3. Метод градиента

Довольно типичной проблемой при решении нелинейных систем является сложность выбора хорошего начального приближения. В этом случае применение изложенных выше методов становится затруднительным, так как все они имеют достаточно локальный характер сходимости. В подобной ситуации обосновано применение *градиентного метода* решения системы нелинейных уравнений.

Пусть дана система (5.1), тогда в векторной форме она имеет вид

$$F(x) = 0, \quad (5.16)$$

где x – вектор, а $F(x)$ – вектор-функция.

Предположим, что F_i – действительные, непрерывно дифференцируемые функции в их общей области определения. Рассмотрим функцию

$$U(x) = \sum_{i=1}^n (F_i(x))^2 = (F_1(x), F_1(x)). \quad (5.17)$$

Предположим, что система (5.17) имеет изолированное решение, которое является точкой минимума функции $U(x)$. Поставим задачу вычисления минимума функции $U(x)$.

Дадим геометрическую интерпретацию решения данной задачи (рис. 5.4). Пусть x – вектор решений системы, а x^0 – его начальное приближение. Через точку x^0 проведем поверхность уровня функции $U(x)$. Далее из точки x^0 двигаемся по нормали к поверхности $U(x) = U(x^0)$ до тех пор, пока эта нормаль не коснется в некоторой точке x^1 какой-то другой поверхности уровня $U(x) = U(x^1)$. Затем, отправляясь от точки x^1 , снова двигаемся по нормали к поверхности уровня $U(x) = U(x^1)$ до тех пор, пока эта нормаль не коснется в некоторой точке x^2 новой поверхности уровня $U(x) = U(x^2)$, и т.д. Так как $U(x^0) > U(x^1) > U(x^2) > \dots$, то, двигаясь таким же образом, мы быстро приблизимся к точке с наименьшим значением U , которая соответствует искомому корню x системы (5.1).

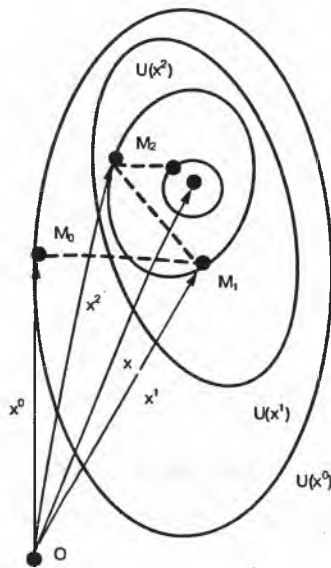


Рис. 5.4 ▼ Геометрическая интерпретация градиентного метода

Дадим определение градиента [6]. Градиент функции $U(x)$ ¹ есть вектор, приложенный в точке x , имеющий направление нормали n к поверхности уровня функции в данной точке в сторону возрастания U и длину, равную $\partial U / \partial n$. Справедлива формула

$$\nabla U = \frac{\partial U}{\partial x_1} e_1 + \frac{\partial U}{\partial x_2} e_2 + \dots + \frac{\partial U}{\partial x_n} e_n, \quad (i = 1, 2, \dots, n),$$

где e_i – орты пространства E_n .

¹ Приняты обозначения $\text{grad} U$ или ∇U , где символ ∇ называют наблой.

Теперь обозначим через

$$\nabla U(x) = \begin{pmatrix} \frac{\partial U}{\partial x_1} \\ \frac{\partial U}{\partial x_2} \\ \dots \\ \frac{\partial U}{\partial x_n} \end{pmatrix}$$

градиент функции $U(x)$. Из векторных треугольников OM_0M_1 , OM_1M_2 , ... заключаем, что

$$x^{k+1} = x^k - \lambda_k \cdot \nabla U(x^k), \quad (k = 0, 1, 2, \dots),$$

где пока неизвестен множитель λ_k . Рассмотрим функцию

$$\Phi(\lambda) = U(x^k - \lambda \cdot \nabla U(x^k)).$$

Функция $\Phi(\lambda)$ дает изменение уровня функции U вдоль соответствующей нормали к поверхности уровня в точке x^k . Множитель $\lambda = \lambda_k$ необходимо подобрать таким образом, чтобы функция $\Phi(\lambda)$ имела минимум. Для этого продифференцируем $\Phi(\lambda)$ по λ и полученное выражение приравняем к нулю:

$$\Phi'(\lambda) = \frac{d}{d\lambda} U(x^k - \lambda \nabla U(x^k)) = 0.$$

Наименьший положительный корень этого уравнения и даст нам значение λ_k (подробный вывод см. [11]):

$$2\lambda_k = \frac{(F(x^k), W(x^k) \cdot W^T(x^k) \cdot F(x^k))}{(W(x^k) \cdot W^T(x^k) \cdot F(x^k), W(x^k) \cdot W^T(x^k) \cdot F(x^k))} = \mu_k, \quad (5.18)$$

где

$$W(x) = \frac{dF}{dx} = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \dots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \dots & \frac{\partial F_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \dots & \frac{\partial F_n}{\partial x_n} \end{pmatrix}$$

– матрица Якоби вектор-функции F .

Имеем основную формулу градиентного метода:

$$x^{k+1} = x^k - \mu_k \cdot W_k^T(x^k) \cdot F(x^k), \quad (k = 0, 1, 2, \dots). \quad (5.19)$$

Несмотря на громоздкие вычисления, составить алгоритм градиентного метода будет несложно.

1. Задать начальное приближение x^0 .
2. Рассчитать μ по формуле 5.18.
3. Вычислить следующее значение x по формуле (5.19).
4. Если условие (5.8) истинно, то решение найдено и равно x , иначе перейти к п.2.

Блок-схема этого алгоритма изображена на рис. 5.5, а пример программы дан в листинге 5.1.

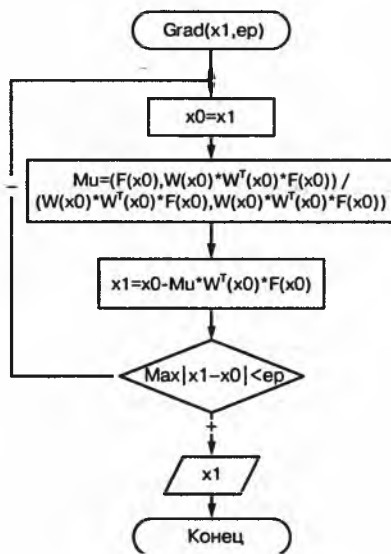


Рис. 5.5 ▼ Алгоритм градиентного метода

5.4. Решение нелинейных уравнений в Mathcad

Для решения систем уравнений в Mathcad существует так называемый *решающий блок*. Открывается он ключевым словом *Given* и имеет следующую структуру:

Начальные условия

Given

Уравнения

Ограничительные условия

Выражения, содержащие функцию для решения системы

Рассмотрим назначение каждого из элементов структуры:

- начальные условия определяют начальные значения искомых переменных и задаются обычным присваиванием;
- уравнения задаются с применением жирного знака равенства (панель инструментов **Boolean** (Логические операции)) между левой и правой частями уравнения;
- ограничительные условия задаются в виде неравенств или равенств (панель инструментов **Boolean**), которые должны выполняться при решении системы уравнений;
- выражение, содержащее функцию для решения системы, может быть представлено одной из функций: Find, Minerr, Maximize, Minimize, Ode-solve, Pdesolve.

В этом разделе мы будем иметь дело только с функциями Find и Minerr, поэтому дадим их описание:

- Find ($x_1, x_2, \dots, x_n$) возвращает значение одной или ряда переменных $x_1, x_2, \dots, x_n$ для точного решения системы;
- Minerr ($x_1, x_2, \dots, x_n$) возвращает значение одной или ряда переменных $x_1, x_2, \dots, x_n$ для приближенного решения системы с минимальной ошибкой.

Разница между этими функциями заключается в том, что первая использует, когда решение $F(x_1, x_2, \dots, x_n) = 0$ реально существует, вторая пытается найти такие значения переменных $x_1, x_2, \dots, x_n$, при которых функция $F(x_1, x_2, \dots, x_n)$ будет максимально приближена к нулю. Например, если функция $F(x_1, x_2, \dots, x_n)$ такова, что при заданной точности вычислений Find выдаст сообщение о том, что решения не существует, то Minerr выдаст значения переменных $x_1, x_2, \dots, x_n$, но с меньшей точностью. Применение функций рассмотрим на конкретных примерах.

ЗАДАЧА 5.1. Найти уравнение окружности, проходящей через три точки: $A(26; 4)$, $B(9; 21)$ и $C(17; 17)$.

Уравнение окружности имеет вид: $(x - x_0)^2 + (y - y_0)^2 = R^2$, где $M(x; y)$ – координаты некоторой точки, лежащей на окружности, $O(x_0; y_0)$ – координаты центра, а R – радиус окружности. Таким образом, задача сводится к отысканию значений x_0, y_0, R . Составим систему из трех уравнений и разрешим ее относительно неизвестных x_0, y_0, R :

$$\begin{aligned} (26 - x_0)^2 + (4 - y_0)^2 &= R^2, \\ (9 - x_0)^2 + (21 - y_0)^2 &= R^2, \\ (17 - x_0)^2 + (17 - y_0)^2 &= R^2. \end{aligned}$$

Файл взят с сайта
www.kodges.ru,
на котором есть еще
много интересной
литературы

На рис. 5.6 показано решение этой системы при помощи функции Find.

$x0 := 0$ $y0 := 0$ $R := 1$ Начальные значения

Given

$$(26 - x0)^2 + (4 - y0)^2 = R^2$$

$$(9 - x0)^2 + (21 - y0)^2 = R^2$$

$$(17 - x0)^2 + (17 - y0)^2 = R^2$$

Решение системы из трех уравнений с тремя неизвестными

$$\begin{pmatrix} x0 \\ y0 \\ R \end{pmatrix} := \text{Find}(x0, y0, R)$$

$$\begin{pmatrix} x0 \\ y0 \\ R \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ 25 \end{pmatrix}$$

Рис. 5.6 ▼ Решение системы уравнений использованием функции Find

Поскольку необходимые параметры найдены, составим уравнение и отобразим окружность и точки, через которые она проходит, на графике (рис. 5.7).

$$\begin{pmatrix} x0 \\ y0 \\ R \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ 25 \end{pmatrix}$$

Параметры окружности

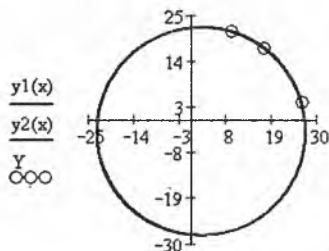
Уравнение окружности

$$(x - x0)^2 + (y - y0)^2 = R^2$$

Функции, задающие линии окружности

$$\begin{aligned} y0 + \left(-x^2 + 2 \cdot x \cdot x0 - x0^2 + R^2 \right)^{\frac{1}{2}} & y1(x) := y0 + \left(-x^2 + 2 \cdot x \cdot x0 - x0^2 + R^2 \right)^{\frac{1}{2}} \\ y0 - \left(-x^2 + 2 \cdot x \cdot x0 - x0^2 + R^2 \right)^{\frac{1}{2}} & y2(x) := y0 - \left(-x^2 + 2 \cdot x \cdot x0 - x0^2 + R^2 \right)^{\frac{1}{2}} \end{aligned}$$

$X := (26 \ 9 \ 17)$ $Y := (4 \ 21 \ 17)$ Координаты точек А, В, С



Окружность и точки А, В, С

x, x, X

Рис. 5.7 ▼ Графическое решение задачи 5.1

На рис. 5.8 приведено решение рассматриваемой системы уравнений с помощью функции Minerr. Как видим, оно полностью совпадает с решением с помощью функции Find (рис. 5.6).

На рис. 5.9 приведен пример некорректного решения поставленной задачи. В начальных условиях переменной R присвоено нулевое значение, то есть предполагается, что радиус окружности равен нулю. При попытке решения с помощью функции Find Mathcad выдает сообщение об отсутствии решений. Minerr решает систему, но предложенное решение не представляет интереса с точки зрения поставленной задачи. Следовательно, решая системы нелинейных уравнений, функцию Minerr использовать нужно осторожно, а начальные значения переменных подбирать как можно точнее.

$x0 := 0$ $y0 := 0$ $R := 1$ Начальные значения

Given

$$(26 - x0)^2 + (4 - y0)^2 = R^2$$

$$(9 - x0)^2 + (21 - y0)^2 = R^2$$

$$(17 - x0)^2 + (17 - y0)^2 = R^2$$

Решение системы из трех уравнений с тремя неизвестными

$$\begin{pmatrix} x0 \\ y0 \\ R \end{pmatrix} := \text{Minerr}(x0, y0, R)$$

$$\begin{pmatrix} x0 \\ y0 \\ R \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \\ 25 \end{pmatrix}$$

Рис. 5.8 ▼ Решение системы уравнений с использованием функции Minerr

$x0 := 0$ $y0 := 0$ $R := 0$

Given

$$(26 - x0)^2 + (4 - y0)^2 = R^2$$

$$(9 - x0)^2 + (21 - y0)^2 = R^2$$

$$(17 - x0)^2 + (17 - y0)^2 = R^2$$

$\text{Find}(x0, y0, R) = \text{■}$

$x0 := 0$ $y0 := 0$ $R := 0$

Given

$$(26 - x0)^2 + (4 - y0)^2 = R^2$$

$$(9 - x0)^2 + (21 - y0)^2 = R^2$$

$$(17 - x0)^2 + (17 - y0)^2 = R^2$$

$\text{Minerr}(x0, y0, R) = \begin{pmatrix} 16.544 \\ 14.815 \\ 0 \end{pmatrix}$ Проверка

$$(26 - x0)^2 + (4 - y0)^2 = 692$$

$$(9 - x0)^2 + (21 - y0)^2 = 522$$

$$(17 - x0)^2 + (17 - y0)^2 = 578$$

No solution was found. Try changing the guess value or the value of TOL or CTOL.

Рис. 5.9 ▼ Пример использования функций Find и Minerr

ЗАДАЧА 5.2. Решить систему нелинейных уравнений:

$$x^2 + y^2 + 3x - 2y = 17,$$

$$x^2 + y^2 + x - y = 12.$$

Графическое решение системы приведено на рис. 5.10.

$$\begin{aligned} x^2 + y^2 + 3x - 2y &= 17 & x^2 + y^2 + x - y &= 12 \\ \left[\begin{array}{c} 1 + (18 - x^2 - 3 \cdot x)^{\frac{1}{2}} \\ 1 - (18 - x^2 - 3 \cdot x)^{\frac{1}{2}} \end{array} \right] & \left[\begin{array}{c} -\frac{1}{2} + \frac{1}{2} \cdot (49 - 4 \cdot x^2 - 4 \cdot x)^{\frac{1}{2}} \\ -\frac{1}{2} - \frac{1}{2} \cdot (49 - 4 \cdot x^2 - 4 \cdot x)^{\frac{1}{2}} \end{array} \right] \\ y1(x) := 1 + (18 - x^2 - 3 \cdot x)^{\frac{1}{2}} & y3(x) := -\frac{1}{2} + \frac{1}{2} \cdot (49 - 4 \cdot x^2 - 4 \cdot x)^{\frac{1}{2}} \\ y2(x) := 1 - (18 - x^2 - 3 \cdot x)^{\frac{1}{2}} & y4(x) := -\frac{1}{2} - \frac{1}{2} \cdot (49 - 4 \cdot x^2 - 4 \cdot x)^{\frac{1}{2}} \end{aligned}$$

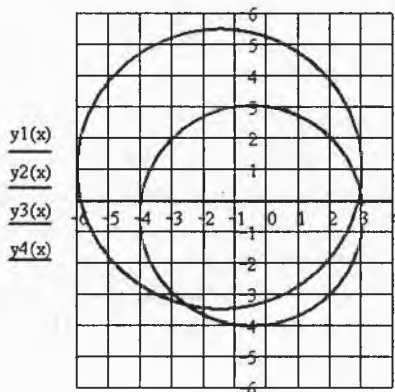


Рис. 5.10 ▼ Графическое решение задачи 5.2

Очевидно, что система уравнений имеет два решения, так как окружности пересекаются в двух точках. Для определения каждого из них необходимо дважды воспользоваться решающим блоком и соответствующим образом подобрать начальные значения переменных. На рис. 5.11 показано, как это сделать.

$$x := 2 \quad y := 0$$

Given

Решение в первой четверти
системы координат

$$x^2 + y^2 + 3 \cdot x - 2 \cdot y = 17$$

$$x^2 + y^2 + x + y = 12$$

$$\begin{pmatrix} x1 \\ y1 \end{pmatrix} := \text{Find}(x, y) \quad \begin{pmatrix} x1 \\ y1 \end{pmatrix} = \begin{pmatrix} 2.945 \\ 0.296 \end{pmatrix}$$

$$x := -2 \quad y := 0$$

Given

Решение в третьей четверти
системы координат

$$x^2 + y^2 + 3 \cdot x - 2 \cdot y = 17$$

$$x^2 + y^2 + x + y = 12$$

$$\begin{pmatrix} x2 \\ y2 \end{pmatrix} := \text{Find}(x, y) \quad \begin{pmatrix} x2 \\ y2 \end{pmatrix} = \begin{pmatrix} -2.56 \\ -3.373 \end{pmatrix}$$

Проверка

$$x1^2 + y1^2 + 3 \cdot x1 - 2 \cdot y1 = 17 \quad x2^2 + y2^2 + 3 \cdot x2 - 2 \cdot y2 = 17$$

$$x1^2 + y1^2 + x1 + y1 = 12 \quad x2^2 + y2^2 + x2 + y2 = 12$$

Рис. 5.11 ▼ Численное решение задачи 5.2

ЗАДАЧА 5.3. Решить систему уравнений:

$$\begin{aligned} x^2 + y^2 + 3x - 2y &= 4, \\ x + 2y &= 5. \end{aligned}$$

Данная система легко сводится к одному уравнению при помощи элементарных преобразований (рис. 5.12). Линейное уравнение решается относительно одного из двух неизвестных: например, можно выразить x через y , выполнив команду **Symbolic** ► **Variable** ► **Solve** (Символика ► Переменная ► Решить) при выделенном x . Полученное выражение необходимо подставить в квадратное уравнение и упростить: **Symbolic** ► **Variable** ► **Collect** (Символика ► Переменная ► Привести подобные).

В уравнении $x + 2 \cdot y = 5$ выразим x через y : $-2 \cdot y + 5$ Подставим в уравнение $x^2 + y^2 + 3 \cdot x - 2 \cdot y - 4 = 0$ Получим: $(-2 \cdot y + 5)^2 + y^2 + 3 \cdot (-2 \cdot y + 5) - 2 \cdot y - 4 = 0$ Упростим, полученное выражение: $5 \cdot y^2 - 28 \cdot y + 36 = 0$

Рис. 5.12 ▼ Приведение системы из задачи 5.3 к одному уравнению



Решение квадратного уравнения с одним неизвестным, полученного в результате преобразований заданной системы, приведено на рис. 5.13. Графическое решение уравнения показало, что имеется два действительных корня. Поэтому решающий блок используется дважды с соответствующими начальными значениями.

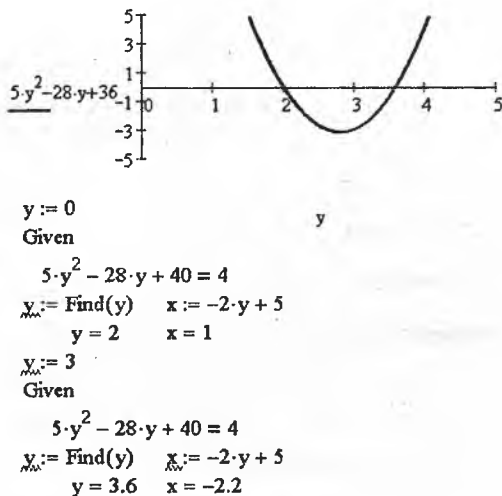


Рис. 5.13 ▾ Решение одного уравнения с помощью решающего блока

Эта задача была приведена исключительно для демонстрации использования решающего блока при определении корней одного уравнения. Советуем читателю проверить полученные здесь решения способом, описанным в предыдущих задачах.

ЗАДАЧА 5.4. Решить систему линейных уравнений.

Задача приводится как пример использования решающего блока для вычисления корней системы линейных уравнений. Ход решения приведен на рис. 5.14.

Обращаем внимание читателя на то, что в Mathcad существует возможность выбрать численный алгоритм, которым будет решена система. На рис. 5.15 изображено меню, которое появляется, если щелкнуть правой кнопкой мыши по функции Find. По умолчанию установлен автоматический выбор численного алгоритма (**AutoSelect**). Для решения нелинейных систем предусмотрены алгоритмы сопряженных градиентов, Левенберга и квазиньютоновский. Кроме того, можно воспользоваться пунктом **Advanced Options** (Дополнительные параметры) для установки параметров численного алгоритма.

$$x_1 := 0 \quad x_2 := 0 \quad x_3 := 0 \quad x_4 := 0$$

Given

$$7.5 \cdot x_1 + 1.8 \cdot x_2 - 2.1 \cdot x_3 - 7.7 \cdot x_4 = 1.1$$

$$-10 \cdot x_1 + 1.3 \cdot x_2 - 20 \cdot x_3 + 1.4 \cdot x_4 = 1.5$$

$$2.8 \cdot x_1 - 1.7 \cdot x_2 + 3.9 \cdot x_3 + 8.8 \cdot x_4 = 1.2$$

$$10 \cdot x_1 + 31.4 \cdot x_2 - 2.1 \cdot x_3 - 10 \cdot x_4 = -1.1$$

$$\text{Find}(x_1, x_2, x_3, x_4) = \begin{pmatrix} 0.244 \\ -0.085 \\ -0.194 \\ 0.128 \end{pmatrix}$$

Рис. 5.14 ▼ Решение системы линейных уравнений с помощью решающего

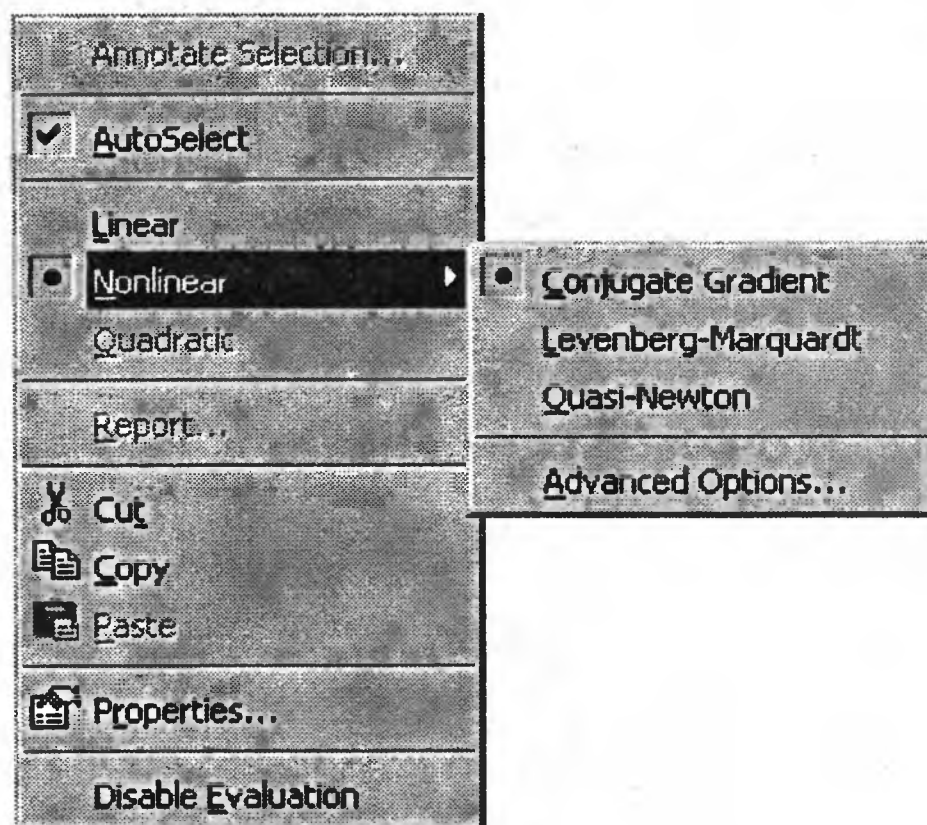


Рис. 5.15 ▼ Меню выбора численного алгоритма решения системы

ЗАДАЧА 5.5. Решить нелинейную систему методом Ньютона:

$$\begin{aligned} x_1 + 3 \cdot \log(x_1) - x_2^2 &= 0, \\ 2x_1 - x_1 x_2 - 5x_1 + 1 &= 0. \end{aligned}$$

Идея метода Ньютона изложена в разделе 5.1. Там же приведен алгоритм. Рис. 5.16 содержит графическое решение заданной системы. Нетрудно заметить, что система имеет два решения.

Заданная нелинейная система

$$2 \cdot x_1^2 - x_1 \cdot x_2 - 5 \cdot x_1 + 1 = 0$$

$$x_1 + 3 \cdot \log(x_1) - x_2^2 = 0$$

Символьные преобразования (Symbolics\Variable\Solve).

Для первого уравнения

$$\frac{2 \cdot x_1^2 - 5 \cdot x_1 + 1}{x_1}$$

Для второго уравнения

$$\left[\begin{array}{l} \frac{1}{\ln(10)} \cdot [\ln(10) \cdot (x_1 \cdot \ln(10) + 3 \cdot \ln(x_1))]^{\frac{1}{2}} \\ \frac{-1}{\ln(10)} \cdot [\ln(10) \cdot (x_1 \cdot \ln(10) + 3 \cdot \ln(x_1))]^{\frac{1}{2}} \end{array} \right]$$

Графическое решение системы

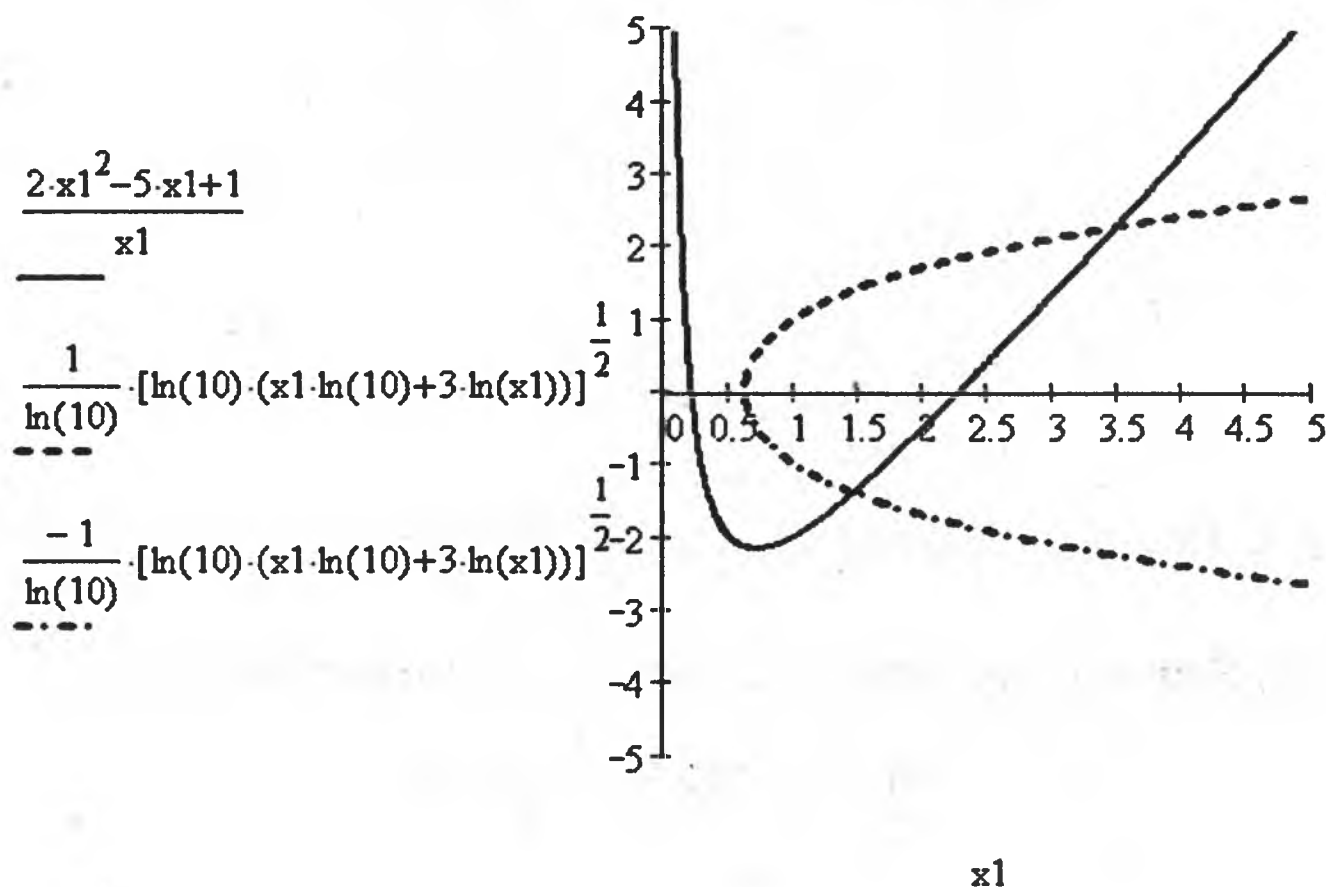


Рис. 5.16 ▾ Графическое решение системы из задачи 5.5

Решение задачи показано на рис. 5.17. Здесь процесс отыскания корней системы при помощи метода Ньютона организован в виде функции, входные параметры которой – вектор начальных приближений и точность вычислений.

ORIGIN := 1

$$F1(x1, x2) := x1 + 3 \cdot \log(x1) - x2^2$$

$$F2(x1, x2) := 2 \cdot x1^2 - x1 \cdot x2 - 5 \cdot x1 + 1$$

$$W11(x1, x2) := \frac{d}{dx1} F1(x1, x2) \quad W12(x1, x2) := \frac{d}{dx2} F1(x1, x2)$$

$$W21(x1, x2) := \frac{d}{dx1} F2(x1, x2) \quad W22(x1, x2) := \frac{d}{dx2} F2(x1, x2)$$

```
newtl(x0, ε) := | while 1
                  |   x ← x0
                  |   W ←  $\begin{pmatrix} W11(x1, x2) & W12(x1, x2) \\ W21(x1, x2) & W22(x1, x2) \end{pmatrix}$ 
                  |   b ←  $\begin{pmatrix} F1(x1, x2) \\ F2(x1, x2) \end{pmatrix}$ 
                  |   x0 ← x - lsolve(W, b)
                  |   break if min(|x0 - x|) < ε
                  x0
```

Решение системы

Проверка

$$X := \text{newtl} \left[\begin{pmatrix} 3.5 \\ 2 \end{pmatrix}, 0.0001 \right] \quad X = \begin{pmatrix} 3.48744 \\ 2.26163 \end{pmatrix} \quad F1(X_1, X_2) = 0.00000$$

$$Y := \text{newtl} \left[\begin{pmatrix} 1.5 \\ -2 \end{pmatrix}, 0.0001 \right] \quad Y = \begin{pmatrix} 1.45889 \\ -1.39677 \end{pmatrix} \quad F2(Y_1, Y_2) = -0.00000$$

Рис. 5.17 ▼ Решение нелинейной системы методом Ньютона

ЗАДАЧА 5.6. Найти положительные корни системы из задачи 5.5 методом итерации и модифицированным методом Ньютона (описание см. в разделе 5.2).

Рис. 5.18 содержит решение поставленной задачи методом итерации.

Приведение системы к виду $x = \phi(x)$

$$\phi_1(x, y) := \sqrt{\frac{x \cdot (y + 5) - 1}{2}} \quad \phi_2(x, y) := \sqrt{x + 3 \log(x)}$$

```
iter(x1, y1, ε) :=
    k ← 0
    while 1
        x0 ← x1
        y0 ← y1
        x1 ← φ1(x0, y0)
        y1 ← φ2(x0, y0)
        x ← x1 - x0
        y ← y1 - y0
        k ← k + 1
        break if max(|x|, |y|) < ε
    ( x1
      y1 )
```

Положительное решение системы

$$\text{iter}(3.5, 2.2, 0.001) = \begin{pmatrix} 3.4858 \\ 2.26084 \end{pmatrix}$$

Рис. 5.18 ▼ Решение нелинейной системы методом итерации

Отыскание решений системы нелинейных уравнений модифицированным методом Ньютона показано на рис. 5.19.

$$\begin{aligned}
 &\underline{\text{ORIGIN}} := 1 \\
 &\underline{F}(x, y) := \begin{pmatrix} 2 \cdot x^2 - x \cdot y - 5 \cdot x + 1 \\ x + \log(x) \cdot 3 - y^2 \end{pmatrix} \quad \text{Вектор-функция заданной системы} \\
 &\underline{D}(x, y) := - \begin{pmatrix} \frac{d}{dx} F(x, y)_1 & \frac{d}{dy} F(x, y)_1 \\ \frac{d}{dx} F(x, y)_2 & \frac{d}{dy} F(x, y)_2 \end{pmatrix}^{-1} \quad \text{Матрица } \Lambda \\
 &\text{itera}(X, \varepsilon) := \begin{array}{l} \Lambda \leftarrow D(X_1, X_2) \\ \text{while } 1 \\ \quad \begin{array}{l} x_1 \leftarrow X_1 \\ x_2 \leftarrow X_2 \\ Y \leftarrow \Lambda \cdot F(x_1, x_2) \\ X \leftarrow x + Y \\ \text{break if } \max(|x - X|) < \varepsilon \end{array} \\ X \end{array} \\
 &\text{Положительное решение системы} \\
 &X0 := \begin{pmatrix} 3.5 \\ 2.1 \end{pmatrix} \quad \text{itera}(X0, 0.0001) = \begin{pmatrix} 3.487 \\ 2.262 \end{pmatrix}
 \end{aligned}$$

Рис. 5.19 ▼ Решение нелинейной системы модифицированным методом Ньютона

5.5. Использование Maple для решения нелинейных уравнений

Для решения уравнений и систем в Maple предназначены функции `solve` и `fsolve`. Подробно они были описаны в разделе 4.9. Напомним, что `solve` ищет решение в аналитической форме, а `fsolve` – в численной. Синтаксис этих команд при решении систем уравнений следующий:

- `solve({<уравнения>}, {<переменные>});`
- `fsolve({<уравнения>}, {<переменные>}, <опции>).`

То есть обе команды решают систему уравнений, заданную первым параметром, относительно переменных, указанных во втором¹. Важной особенностью синтаксиса этих функций является то, что система уравнений и ее неизвестные задаются в виде множества. Кроме того, если не задано множество переменных, относительно которых следует разрешить уравнение, Maple выдаст все решения относительно всех неопределенных переменных. Если же уравнения системы заданы не в виде равенства, а в виде выражения, то это выражение воспринимается пакетом как левая часть уравнения с нулевой правой частью.

Рассмотрим несколько примеров решения систем нелинейных уравнений в Maple.

ЗАДАЧА 5.7. Решить задачу 5.1 средствами Maple.

На рис. 5.20 показано решение системы, к которой сводится задача 5.1 с помощью функции `solve`. Как видим, она предлагает два аналитических решения. Но одно из них не удовлетворяет условию задачи, так как радиус окружности не может быть отрицательным. Чтобы выделить из множества верное решение, можно применить функцию `eval`.

```
[> u1:=(26-x0)^2+(4-y0)^2-R^2:
[> u2:=(9-x0)^2+(21-y0)^2-R^2:
[> u3:=(17-x0)^2+(17-y0)^2-R^2:
[> solve({u1,u2,u3},{x0,y0,R});
      {R=25,y0=-3,x0=2},{R=-25,y0=-3,x0=2}
[> S:=solve({u1,u2,u3},{x0,y0,R}):
[> eval(S[1]);
      {R=25,y0=-3,x0=2}
```

Рис. 5.20 ▼ Аналитическое решение задачи 5.7

Рис. 5.21 содержит решение задачи с применением функции `fsolve`. В первом случае пользователю предлагается лишь отрицательное решение, не корректное с точки зрения поставленной задачи. Во втором случае получаем верное решение, установив в опциях положительный диапазон изменения радиуса.

```
[> fsolve({u1,u2,u3},{x0,y0,R});
      {R=-25.00000000,x0=2.000000000,y0=-3.000000000}
[> fsolve({u1,u2,u3},{x0,y0,R},{R=0..100});
      {x0=2.000000000,y0=-3.000000000,R=25.00000000}
```

Рис. 5.21 ▼ Численное решение задачи 5.7

¹ Параметр <опции> описан в разделе 4.9.

ЗАДАЧА 5.8. Найти численное и аналитическое решение задачи 5.5 средствами Maple.

Графическое решение системы из задачи 5.5, из которого видно, что система имеет два корня, показано на рис. 5.16.

Если система не имеет решения, которое можно было бы выразить через радикалы, то для представления результатов работы функции `solve` Maple использует функцию `RootOf()`, при этом заменяя переменные уравнения системой переменной `_Z`. Решение данной задачи (рис. 5.22) приводит именно к таким результатам, которые мы не приводим в силу их громоздкости. Однако приближенное численное значение функции `RootOf()` можно получить с помощью функции `evalf`.

```
[> solve({x1+3*log10(x1)-x2^2, 2*x1^2-x1*x2-5*x1+1}, {x1, x2}):
> evalf(%);
({x2 = -1.396767010, x1 = 1.458890230})
```

Рис. 5.22 ▼ Аналитическое решение задачи 5.8

Заметим, что решение нелинейных уравнений и систем в аналитическом (символьном) виде достаточно сложная задача и иногда довольно простые на вид системы могут не решаться в Maple. В этом случае он вообще не выдает никаких результатов или выдает, но не все. Так, в нашей задаче с помощью функции `solve` найдено только одно решение системы, тогда как графически было найдено два корня.

Функция `fsolve` по умолчанию находит только одно решение. На рис. 5.23 показано, как можно найти все корни уравнения, задавая интервал изменения неизвестной величины.

```
> fsolve({x1+3*log10(x1)-x2^2, 2*x1^2-x1*x2-5*x1+1},
{x1, x2});
{x1 = 1.458890230, x2 = -1.396767009}
> fsolve({x1+3*log10(x1)-x2^2, 2*x1^2-x1*x2-5*x1+1},
{x1, x2}, {x1=3..4, x2=2..3});
{x1 = 3.487442788, x2 = 2.261628631}
```

Рис. 5.23 ▼ Численное решение задачи 5.8

ЗАДАЧА 5.9. Решить систему уравнений

$$\begin{aligned} x^2 + y^2 + 3x - 2y &= 0, \\ x + 2y &= 5. \end{aligned}$$

Система не имеет действительных решений, так как линии на графике (рис. 5.24) не пересекаются.


```
> implicitplot({x^2+y^2+3*x-2*y=0,x+2*y=5},
  x=-4..2,y=-2..4,color=black);
```

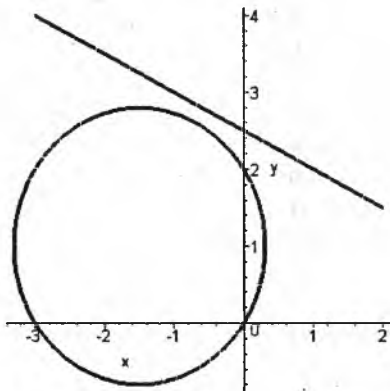


Рис. 5.24 ▽ Графическое решение задачи 5.9

Попытка решить эту задачу численно приводит к тому, что Maple просто повторяет введенную команду `fsolve` так, как показано на рис. 5.25. Происходит это потому, что по умолчанию эта функция ищет действительные корни системы. Указав в опциях `complex`, получим комплексное решение. Аналитическая функция `solve` выдает комплексный ответ сразу.

```
> fsolve({x^2+y^2+3*x-2*y=0,x+2*y=5},{x,y});
      fsolve({x^2+y^2+3*x-2*y=0,x+2*y=5},{x,y})
> evalf(solve({x^2+y^2+3*x-2*y=0,x+2*y=5},{x,y}));
      {y = 2.800000000 + 0.4000000000 I, x = -0.6000000000 - 0.8000000000 I}
> fsolve({x^2+y^2+3*x-2*y=0,x+2*y=5},{x,y},complex);
      {y = 2.800000000 - 0.4000000000 I, x = -0.6000000000 + 0.8000000000 I}
```

Рис. 5.25 ▽ Решение задачи 5.9

5.6. Возможности MATLAB для решения нелинейных уравнений

Решение нелинейных уравнений и систем выполняется в MATLAB с помощью функции `fsolve`. Синтаксис этой функции выглядит так:

```
fsolve(<имя функции>, <начальное значение>),
```

где <имя функции> – имя М-функции, которая определяет левую часть уравнения $f(x) = 0$ или системы уравнений $F(x) = 0$: она должна принимать на входе вектор аргументов и возвращать вектор значений; <начальное значение> – вектор приближений, относительно которого будет осуществляться поиск решения.

Рассмотрим несколько конкретных задач.

ЗАДАЧА 5.10. Найти решение системы нелинейных уравнений

$$\begin{aligned}x_1^2 + x_2^2 &= 1, \\ x_1^3 - x_2 &= 0.\end{aligned}$$

Решим систему графически. Для этого выполним перечень команд, указанных в листинге 5.1. Результат работы этих команд показан на рис. 5.26. Понятно, что система имеет два корня – положительный и отрицательный.

Листинг 5.1

```
>> x=-1:0.01:1; %Аргумент
>> y1=(1-x.^2).^(1/2); %Первое уравнение, положительные значения
>> y2=-(1-x.^2).^(1/2); %Первое уравнение, отрицательные значения
>> y3=x.^3; %Второе уравнение
>> y=[y1;y2;y3]; %
>> plot(x,y,'k.')
>> grid on
```

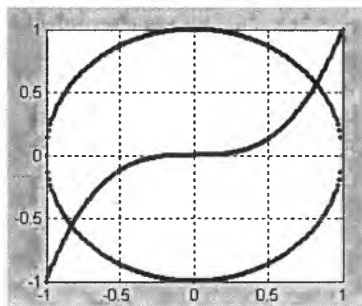


Рис. 5.26 ▼ Графическое решение задачи 5.10

Составим М-функцию, соответствующую левой части системы (листинг 5.2). Здесь важно помнить, что все уравнения должны иметь вид $F(x) = 0$.

Листинг 5.2

```
function f=fun(x)
f(1)=x(1)^2+x(2)^2-1;
f(2)=x(1)^3-x(2);
end
```

Теперь решим систему (листинг 5. 3), указав в качестве начального приближения функции `fsolve` вначале вектор (1, 1), затем (-1, -1).

Листинг 5.3

```
>> fsolve('fun',[1 1])
Optimization terminated: first-order optimality is less than options.TolFun.
ans =
    0.8260    0.5636
>> fsolve('fun',[-1 -1])
Optimization terminated: first-order optimality is less than options.TolFun.
ans =
   -0.8260   -0.5636
```

ЗАДАЧА 5.11. Решить систему нелинейных уравнений

$$\begin{aligned}x_1^2 + x_2^2 + x_3^2 &= 1, \\ 2x_1^2 + x_2^2 - 4x_3 &= 0, \\ 3x_1^2 - 4x_2 + x_3^2 &= 0.\end{aligned}$$

Листинг 5.4 содержит М-функцию заданной системы, а ее решение показано в листинге 5.5. Обратите внимание на выходные параметры функции `fsolve`. Здесь `x` – это решение системы, `f` – значение вектор-функции для найденного решения, а `ex` – признак завершения алгоритма решения нелинейной системы. Отрицательное значение параметра `ex` означает, что решение не найдено, ноль – досрочное прерывание вычислительного процесса при достижении максимально допустимого числа итераций, положительное значение подтверждает, что решение найдено с заданной точностью. В нашем случае значения функции для найденного решения близки к нулю и признак завершения положительный, значит, найдено верное решение.

Листинг 5.4

```
function f=Y(x)
f(1)=x(1)^2+x(2)^2+x(3)^2-1;
f(2)=2*x(1)^2+x(2)^2-4*x(3);
f(3)=3*x(1)^2-4*x(2)+x(3)^2;
end
```

Листинг 5.5

```
>> [x,f,ex]=fsolve('Y',[0.5 0.5 0.5])
Optimization terminated: first-order optimality is less than options.TolFun.
x =
    0.7852    0.4966    0.3699
f =
    1.0e-009 *
    0.1828    0.3652    0.5480
ex =
```

ЗАДАЧА 5.12. Решить систему из задачи 5.9 средствами MATLAB.

Графическое решение системы (рис. 5.24) показало, что она корней не имеет. Однако применение к системе функции `fsolve` дает положительный ответ, что видно из листинга 5.6. Происходит это потому, что алгоритм, реализованный в этой функции, основан на минимизации суммы квадратов компонент вектор-функции. Следовательно, наличие точки минимума не гарантирует существования корней системы в ее окрестности.

ЛИСТИНГ 5.6

```
function y=Fu(x)
y(1)=x(1)^2+x(2)^2-3*x(1)-2*x(2);
y(2)=x(1)+2*x(2)-5;
end
%-----
>> [x,f,ex]=fsolve("fu",[1 1])
Optimization terminated: first-order optimality is less than
options.TolFun.
x =
    0.3033    2.3483
f =
    1.0e-009 *
    0.2012    0.0001
ex =
    1
```

ЗАДАЧА 5.13. Используя градиентный метод, решить нелинейную систему

$$\begin{aligned}x + x^2 - 2yz &= 0,1, \\ y - y^2 + 3xz &= -0,2, \\ z + z^2 + 2xy &= 0,3.\end{aligned}$$

Алгоритм метода градиента описан в разделе 5.3. Для того чтобы реализовать его программно, необходимо составить следующие М-функции:

► вычисление значений системы нелинейных уравнений (листинг 5.7);

ЛИСТИНГ 5.7

```
function f=F(x,y,z)
f(1)=x+x^2-2*y*z-0.1;
f(2)=y-y^2+3*x*z+0.2;
f(3)=z+z^2+2*x*y-0.3;
end
```

► определение матрицы Якоби (листинг 5.8);

ЛИСТИНГ 5.8

```
function W=Yak(x0,y0,z0)
syms x y z
```

```

w1=diff(F(x,y,z),x);
w2=diff(F(x,y,z),y);
w3=diff(F(x,y,z),z);
w=[w1;w2;w3];
W1=subs(w1,{'x','y','z'},{x0,y0,z0});
W2=subs(w2,{'x','y','z'},{x0,y0,z0});
W3=subs(w3,{'x','y','z'},{x0,y0,z0});
W=[W1;W2;W3];
End

```

► реализация метода градиента (листинг 5.9).

Листинг 5.9

```

function x1=grad(x1,ep)
while 1
    x0=x1;
    a=F(x0(1),x0(2),x0(3));
    w=Yak(x0(1),x0(2),x0(3))';
    b=w*w'*a';
    Mu=dot(a,b)/dot(b,b);
    x1=x0-Mu*w'*a';
    R=x0-x1;
    if max(abs(R))<ep
        break;
    end;
end;
end
%_____
%Результат работы М-функции
>> [X1]=grad(X0,0.001)
X1 = %Решение системы
    0.0129
   -0.1779
    0.2447
>> F(X1(1),X1(2),X1(3))
ans = %Проверка
    1.0e-003 *
    0.1048   -0.0521    0.0152

```

Примите к сведению, что функция решения уравнений и их систем может выглядеть так:

`fsolve(<имя функции>, <начальное значение>, <опции>),`

где <опции> – это дополнительные параметры управления вычислительным процессом. Определить эти параметры можно при помощи функции `optimset(<вид контроля>, <значение>)`.

Например, в листингах 5.3, 5.5 и 5.6 функция `fsolve` используется без дополнительных параметров и всегда сопровождается сообщением о модификации алгоритма. Это сообщение можно подавить, если использовать команду `fsolve('fun', x0, optimset('fsolve'))`, тогда будут установлены все принятые по умолчанию настройки.

Если в качестве параметра использовать функцию `optimset('Display', 'iter')`, то будет выводиться информация о каждом шаге вычислительного процесса, а функция `optimset('Display', 'final')` выведет информацию только о завершении процесса. Отключить вывод информации вообще можно с помощью команды `optimset('Display', 'off')`. Параметр `optimset('Display', 'notify')` установлен по умолчанию.

Функция `optimset('TolX', '1.0e-8')` поможет установить точность вычислительного процесса.

Допустима и такая конструкция `optimset('Display', 'iter', 'TolX', '1.0e-8')`, в которой одновременно задается и вывод информации о вычислительном процессе, и точность вычислений.

5.7. Советы по использованию математических пакетов при решении нелинейных уравнений

И Mathcad, и Maple, и MATLAB отлично справляются с решением нелинейных систем. Mathcad и его решающий блок Given привлекают высокой степенью наглядности. Maple поражает обилием разнообразных функций, предназначенных для решения уравнений и систем. Кроме универсальных команд `solve` (аналитическое решение) и `fsolve` (численное решение) он содержит специализированные команды решения задач определенного класса, например можно искать только целочисленные корни (`isolve`), корни по заданному модулю (`msolve`) или решать рекуррентные уравнения (`ksolve`). MATLAB, несмотря на небогатый выбор функций (`fsolve`), обладает уникальной возможностью управления вычислительным процессом. Так, например, при изучении численных методов весьма полезными могут оказаться сведения о ходе вычислений, представленные в листинге 5.10 (решение задачи 5.11). Здесь кроме результата выведена и таблица, каждая строка которой соответствует одной итерации. Из таблицы можно узнать, сколько раз вызывалась исследуемая функция, текущее приближение и значение функции от него, а также численный метод, которым проводилось решение.

Листинг 5.10

```
>> x=fsolve("Y",[0.5 0.5 0.5],optimset("Display",'iter'))
```

Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	4	2.625		5.75	1
1	8	0.294922	0.395285	3.55	1
2	12	0.000739126	0.0854008	0.161	1
3	16	6.30212e-009	0.00460618	0.000466	1
4	20	4.67008e-019	1.35103e-005	4.02e-009	1

Optimization terminated: first-order optimality is less than options.TolFun.
x =
0.7852 0.4966 0.3699

5.8. Задания для самостоятельного решения

1. Найти численное и графическое решения системы нелинейных уравнений средствами Mathcad:

$$\begin{aligned}\cos x + 2y &= 2, \\ x^2/3 - y^2/3 &= 1.\end{aligned}$$

2. Решить систему нелинейных уравнений в MATLAB численно и графически:

$$\begin{aligned}y^2 - x^2/2 &= 1, \\ 2,1\sin(x-1) + y &= 0,1.\end{aligned}$$

3. Определить корни системы нелинейных уравнений в символьном виде, численно и графически используя возможности Maple:

$$\begin{aligned}x^2 + 2y^2 &= 1, \\ x^2 - y^2 &= 1.\end{aligned}$$

4. Решить нелинейную систему методом Ньютона в MATLAB:

$$\begin{aligned}\sin(x-0,6) - y &= 1,6, \\ 3x - \cos y &= 0,9.\end{aligned}$$

5. Применить метод итерации для решения системы нелинейных уравнений в среде пакета Maple

$$\begin{aligned}\sin(2x-y) - 1,2x &= 0,4, \\ 0,8x^2 + 1,5y^2 &= 1.\end{aligned}$$

6. Найти корни нелинейной системы с помощью модифицированного метода Ньютона, используя возможности MATLAB:

$$\begin{aligned}\sin y + 2x &= 2, \\ \cos(x-1) + y &= 0,7.\end{aligned}$$

7. Используя градиентный метод, решить систему нелинейных уравнений в Mathcad:

$$\begin{aligned}\cos x + y &= 1,5, \\ 2x - \sin(y - 0,5) &= 1.\end{aligned}$$

8. Определить корни уравнения в Mathcad, MATLAB и Maple, используя только функции, описанные в этой главе:

$$\begin{aligned}\text{a) } 5x - 8\ln(x) &= 8; \\ \text{b) } x^{1/2} - \cos(0,387x) &= 0; \\ \text{c) } 1,8x^2 - \sin(10x) &= 0.\end{aligned}$$

9. Свести нелинейную систему к одному уравнению и решить его с помощью функций, описанных в этой главе, в Mathcad, MATLAB и Maple:

$$\begin{aligned}\operatorname{tg}(xy) &= x^2, \\ 0,8x^2 + 3y^2 &= 1.\end{aligned}$$

10. Решить систему линейных уравнений в Mathcad, MATLAB и Maple, используя только функции, описанные в этой главе:

$$\begin{aligned}0,64x_1 + 0,72x_2 - 0,83x_3 + 4,2x_4 &= 2,32, \\ 0,58x_1 - 0,83x_2 + 1,43x_3 - 0,62x_4 &= 1,71, \\ 0,86x_1 + 0,77x_2 - 1,83x_3 + 0,88x_4 &= -0,54, \\ 1,32x_1 - 0,52x_2 - 0,65x_3 + 1,22x_4 &= 0,65.\end{aligned}$$



Численное интегрирование и дифференцирование функций

Эта глава посвящена проблемам численного интегрирования и дифференцирования. Рассмотрены методы приближенного вычисления определенных интегралов, уделено внимание задаче вычисления несобственного интеграла, а также приведены формулы для приближенного вычисления производных.

6.1. Методы численного интегрирования

Пусть дана функция $f(x)$, известно, что она непрерывна на интервале $[a; b]$, и уже определена ее первообразная $F(x)$, тогда определенный интеграл от этой функции можно вычислить в пределах от a до b по формуле Ньютона-Лейбница [6]:

$$\int_a^b f(x)dx = F(b) - F(a), \quad (6.1)$$

где $F'(x) = f(x)$.

Однако на практике часто встречаются интегралы с первообразной, которая не может быть выражена через элементарные функции или является слишком сложной, что затрудняет или делает невозможным вычисления по формуле (6.1). Кроме того, нередко подынтегральная функция задается таблицей или графиком, и тогда понятие первообразной вообще теряет смысл. В этом случае большое значение имеют численные методы интегрирования, основная задача которых заключается в вычислении значения определенного интеграла на основании значений подынтегральной функции.

Численное вычисление определенного интеграла называют *механической квадратурой*. Формулы, соответствующие тому или иному численному методу приближенного интегрирования, называют *квадратурными*. Подобное название связано с *геометрическим смыслом* определенного интеграла: значение определенного интеграла

$$y = \int_b^a f(x) dx, \quad f(x) \geq 0 \quad (6.2)$$

равно площади квадрата, которая, в свою очередь, совпадает с площадью криволинейной трапеции с основаниями $[a; b]$ и $f(x)$.

6.1.1. Интегрирование по методу прямоугольников

Интегрирование по *методу прямоугольников* заключается в том, что интервал интегрирования $[a; b]$ делится точками $x_0, x_1, \dots, x_n$ на n равных частей (рис. 6.1), причем

$$x_0 = a, \quad x_n = b,$$

длина каждой части составляет

$$h = (b - a) / n,$$

и тогда

$$x_i = x_0 + ih, \quad i = 1, \dots, n - 1.$$

Из каждой точки x_i проведем перпендикуляр до пересечения с кривой $f(x)$, а затем заменим кривую подынтегральной функции ломаной линией, отрезки которой параллельны оси абсцисс.

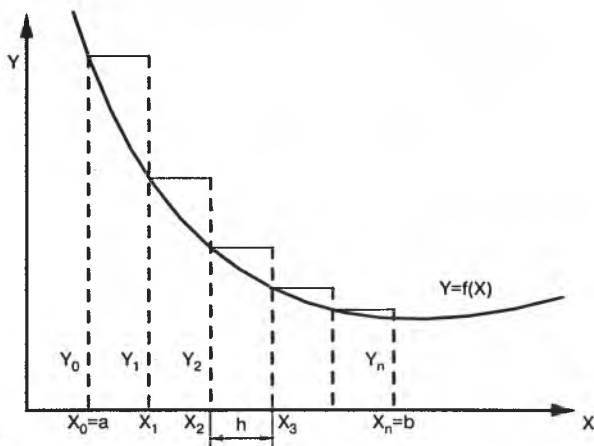


Рис. 6.1 ▼ Графическая интерпретация метода левых прямоугольников

Площадь полученной ступенчатой фигуры можно найти как сумму площадей прямоугольников, стороны которых равны h и y_i . Следовательно, площадь отдельного прямоугольника составит

$$S_i = y_i \cdot h,$$

тогда

$$S = \int_a^b f(x)dx \approx y_0h + y_1h + y_2h + \dots + y_{n-1}h = f(x_0)h + f(x_1)h + f(x_2)h + \dots + f(x_{n-1})h.$$

Следовательно, формула вычисления определенного интеграла по методу прямоугольников имеет вид:

$$I = \int_a^b f(x)dx = h \sum_{i=1}^n f(x_{i-1}). \quad (6.3)$$

Формулу (6.3) называют еще *квадратурной формулой левых прямоугольников*. Поясним почему. Рассмотрим другую геометрическую интерпретацию метода прямоугольников, изображенную на рис. 6.2. В этом случае площадь ступенчатой фигуры будем вычислять так:

$$S = \int_a^b f(x)dx \approx y_1h + y_2h + \dots + y_nh = f(x_1)h + f(x_2)h + \dots + f(x_n)h.$$

Из чего вытекает *квадратурная формула правых прямоугольников*:

$$I = \int_a^b f(x)dx = h \sum_{i=1}^n f(x_i). \quad (6.4)$$

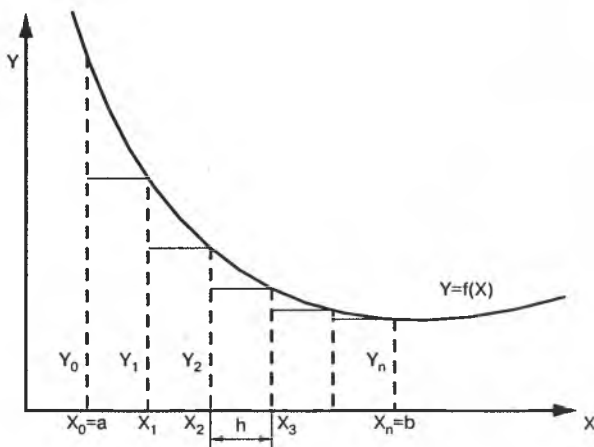


Рис. 6.2 ▾ Графическая интерпретация метода правых прямоугольников

Рис. 6.3 содержит еще один пример геометрической интерпретации интегрирования по методу прямоугольников. Идея этого метода заключается в том, что значение функции вычисляется в точке, расположенной в середине отрезка $[x_{i-1}; x_i]$, то есть

$$x_{i-1} + \frac{h}{2} = x_i - \frac{h}{2}.$$

Из чего вытекает *квадратурная формула средних прямоугольников*:

$$I = \int_a^b f(x) dx = h \sum_{i=1}^n f\left(x_{i-1} + \frac{h}{2}\right) = h \sum_{i=1}^n f\left(x_i - \frac{h}{2}\right). \quad (6.5)$$

Формула (6.5) имеет большую точность [9] по сравнению с (6.3) и (6.4), поэтому именно ее применяют чаще всего и называют *формулой прямоугольников*.

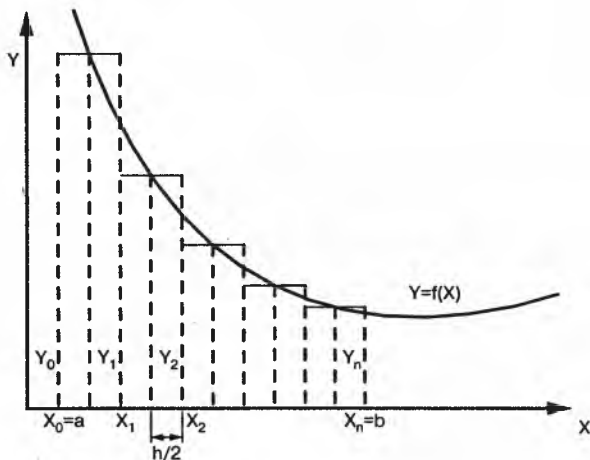


Рис. 6.3 ▽ Графическая интерпретация метода средних прямоугольников

С точки зрения алгоритмизации метод прямоугольников достаточно прост. Необходимо выполнить следующие действия:

1. Определить шаг разбиения интервала $[a; b]$.
2. Рассчитать сумму значений функции в соответствующих точках интервала.
3. Получить результат, умножив полученную сумму на значение шага.

Блок-схемы вычисления определенного интеграла по всем трем квадратурным формулам прямоугольников показаны на рис. 6.4 (примеры программирования в задаче 6.3).

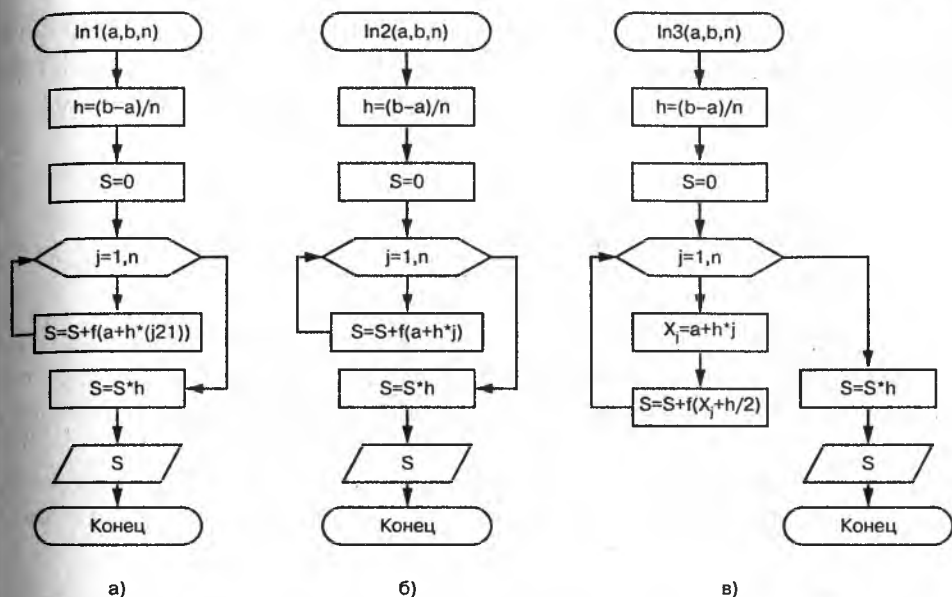


Рис. 6.4 ▼ Алгоритмы методов а) левых прямоугольников; б) правых прямоугольников; в) средних прямоугольников.

6.1.2. Интегрирование по методу трапеций

Для интегрирования *методом трапеций* участок интегрирования $[a; b]$ также разбивают на n равных частей. Если провести ординаты во всех точках деления

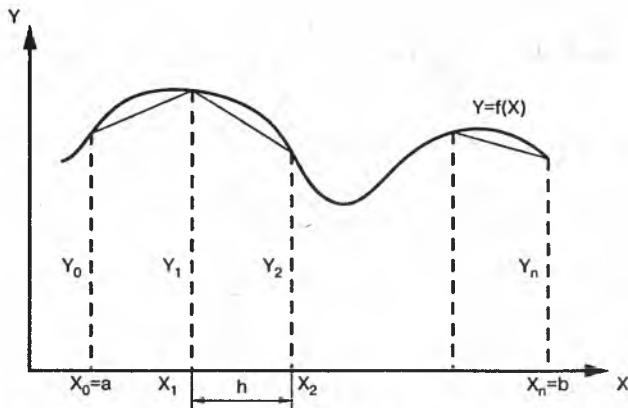


Рис. 6.5 ▼ Геометрическая интерпретация метода трапеций

и заменить каждую из полученных криволинейных трапеций прямолинейной (рис. 6.5), то приближенное значение интеграла будет равно сумме площадей прямолинейных трапеций.

Площадь отдельной трапеции составляет:

$$S = \frac{y_{i-1} + y_i}{2} \cdot h,$$

следовательно, площадь искомой фигуры вычисляют по формуле:

$$S = \int_a^b f(x) dx \approx \sum_{i=1}^n S_i = \frac{h}{2} \sum_{i=1}^n (y_{i-1} + y_i) = h \cdot \left(\frac{y_0 + y_n}{2} + \sum_{i=1}^{n-1} y_i \right).$$

Таким образом, получена *квадратурная формула трапеций* для численного интегрирования:

$$I = \int_a^b f(x) dx = h \cdot \left(\frac{f(a) + f(b)}{2} + \sum_{i=1}^{n-1} f(x_i) \right). \quad (6.6)$$

Рис. 6.6 содержит алгоритм, реализующий численное интегрирование по квадратурным формулам трапеций. Задача 6.6 демонстрирует пример использования этого алгоритма.

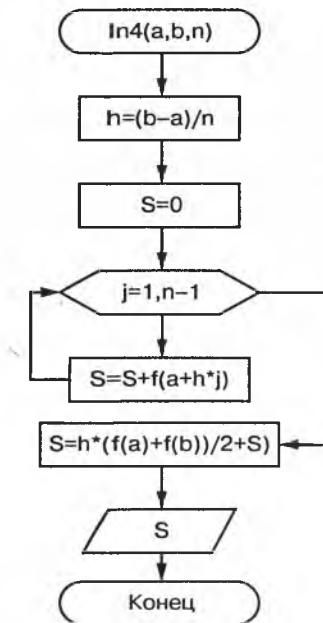


Рис. 6.6 ▼ Алгоритмы метода трапеций

6.1.3. Интегрирование по методу Симпсона

Пусть $n = 2m$ – четное число, а $y_i = f(x_i)$ ($i = 0, n$) – значения функции $y = f(x)$ для равноотстоящих точек $a = x_0, x_1, x_2, \dots, x_n = b$ с шагом $h = (b - a) / n = (b - a) / 2m$. На паре соседних участков (рис. 6.7) кривая $y = f(x)$ заменяется параболой $y = L(x)$, коэффициенты которой подобраны так, что она проходит через точки y_0, y_1, y_2 .

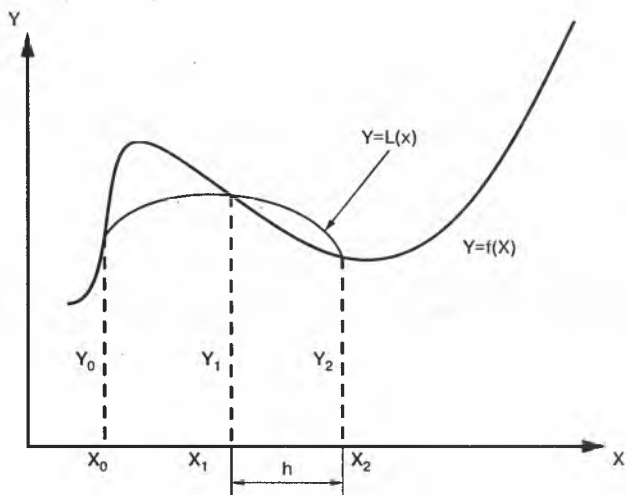


Рис. 6.7 ▽ Геометрическая интерпретация интегрирования по методу Симпсона

Площадь криволинейной трапеции [11], ограниченной сверху параболой, составит:

$$S_i = \frac{h}{3}(y_{i-1} + 4y_i + y_{i+1}).$$

Суммируя площади всех криволинейных трапеций, получим:

$$S = \int_a^b f(x) dx \approx \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + \dots + 2y_{2m-2} + 4y_{2m-1} + y_{2m}) = \frac{h}{3} \left(y_0 + y_{2m} + \sum_{i=1}^{2m-1} p \cdot y_i \right),$$

где $p = 6 - p, p = 4$. Следовательно, формула Симпсона для численного интегрирования имеет вид:

$$I = \int_a^b f(x) dx = \frac{h}{3} \left(f(a) + f(b) + \sum_{i=1}^{2m-1} p \cdot y_i \right). \quad (6.7)$$

Вычислительный алгоритм метода Симпсона выглядит так, как показано на рис. 6.8. Практическое применение этого алгоритма реализовано в задаче 6.6.

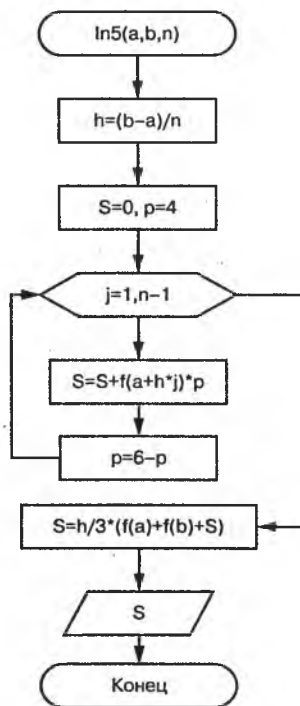


Рис. 6.8 ▼ Алгоритм метода Симпсона

6.1.4. Квадратурные формулы Ньютона–Котеса

Пусть необходимо вычислить интеграл для функции $y = f(x)$:

$$\int_a^b y \, dx.$$

Выберем шаг $h = (b - a) / n$ и разобьем отрезок $[a, b]$ на n равных частей что

$$x_0 = a, x_i = x_0 + ih, i = 1, \dots, n-1, x_n = b,$$

и пусть

$$y_i = f(x_i) \text{ для } i = 0, 1, \dots, n.$$

Заменим функцию y соответствующим полиномом Лагранжа и получим близкую к квадратурную формулу

$$\int_a^b y \, dx = \sum_{i=0}^n A_i \cdot y_i,$$

где A_i – некоторые постоянные коэффициенты.



Как известно (см. раздел 2.1.3), в общем виде полином Лагранжа записывают так:

$$L_n(x) = \sum_{i=0}^n p_i(x) \cdot y_i, \quad (6.9)$$

где

$$p_i(x) = \frac{(x - x_0) \cdot (x - x_1) \dots (x - x_{i-1}) \cdot (x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \cdot (x_i - x_1) \dots (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \dots (x_i - x_n)}. \quad (6.10)$$

Введем замену $x = x_0 + q \cdot h$ и перепишем полином Лагранжа в соответствии с ней:

$$L_n(x_0 + q \cdot h) = \sum_{i=0}^n \frac{(-1)^{n-i} \cdot y_i}{i! \cdot (n-i)!} \cdot \frac{q \cdot (q-1) \dots (q-n)}{q-i}. \quad (6.11)$$

Для того чтобы использовать выражение (6.11) вместо y , нужно изменить границы интегрирования. Значению $x = a$ будет соответствовать $q = 0$, а значению $x = b - q = n$. Также необходимо учесть, что $dx = h \cdot dq$. Таким образом, имеем

$$A_i = h \cdot \frac{(-1)^{n-i}}{i! \cdot (n-i)!} \int_0^n \frac{q \cdot (q-1) \dots (q-n)}{q-i} dq, \quad i = 0, 1, 2, \dots, n. \quad (6.12)$$

Перепишем (6.12) в виде

$$A_i = (b - a) \cdot H_i, \quad (6.13)$$

где

$$H_i = \frac{1}{n} \cdot \frac{(-1)^{n-i}}{i! \cdot (n-i)!} \cdot \int_0^n \frac{q \cdot (q-1) \dots (q-n)}{q-i} dq - \quad (6.14)$$

постоянные, которые называют коэффициентами Котеса (пример формирования матрицы коэффициентов Котеса см. в задаче 6.4).

Квадратурная формула Ньютона-Котеса при этом имеет вид:

$$\int_a^b y dx = (b - a) \cdot \sum_{i=0}^n H_i \cdot y_i. \quad (6.15)$$

Применим формулы (6.14) и (6.15) для $n = 1$. Имеем:

$$H_0 = - \cdot \int_0^1 \frac{q \cdot (q-1)}{q} dq = \frac{1}{2},$$

$$H_1 = \int_0^1 q dq = \frac{1}{2}.$$

Следовательно,

$$\int_{x_0}^{x_1} y dx = \frac{h}{2} \cdot (y_0 + y_1).$$

Таким образом, мы получили уже известную (см. раздел 6.1.2) *формулу трапеций* (рис. 6.9) для вычисления определенного интеграла.

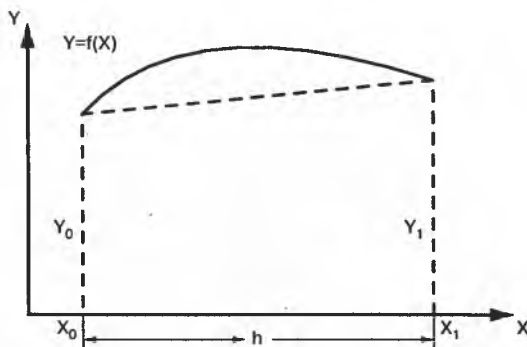


Рис. 6.9 ▽ Геометрическая интерпретация формулы Ньютона-Котеса для $n = 1$

Положим $n = 2$ и подставим в (6.14) и (6.15):

$$H_0 = \frac{1}{2} \cdot \frac{1}{2} \cdot \int_0^2 (q-1)(q-2) dq = \frac{1}{6},$$

$$H_1 = -\frac{1}{2} \cdot \frac{1}{1} \cdot \int_0^2 q(q-2) dq = \frac{2}{3},$$

$$H_2 = \frac{1}{2} \cdot \frac{1}{2} \cdot \int_0^2 q(q-1) dq = \frac{1}{6}.$$

Следовательно, так как $x_2 - x_0 = 2 \cdot h$, имеем *формулу Симпсона* (см. раздел 6.1.3):

$$\int_{x_0}^{x_2} y dx = \frac{h}{3} (y_0 + 4 \cdot y_1 + y_2).$$

Подобные рассуждения приводят к выводу о том, что формулы Ньютона-Котеса определяют *семейство квадратурных формул*, параметром которого является число n – степень интерполяционного многочлена, которым заменяется подынтегральная функция.

Для $n > 2$ получают *квадратурные формулы Ньютона-Котеса высших порядков*. Проведем вычисления, например, для $n = 3$. Из формул (6.14) и (6.15) получим:

$$\int_{x_0}^{x_4} y dx = \frac{3h}{8} (y_0 + 3 \cdot y_1 + 3 \cdot y_2 + y_3).$$

Эта формула носит название *квадратурной формулы Ньютона*, или *правила трех восьмых*.

Вообще говоря, коэффициенты Котеса при большом числе ординат вычислить не так просто. Поэтому на практике для приближенного вычисления определенных интегралов поступают так. Разбивают промежуток интегрирования на достаточно большое число маленьких отрезков и к каждому из них применяют квадратурную формулу Ньютона–Котеса с небольшим числом ординат. В результате получаются не сложные формулы и расчеты по ним дают достаточно высокую точность.

Будем считать, что задан интервал интегрирования $[a; b]$, определена подынтегральная функция $f(x)$, сформирована по формуле (6.14) матрица H коэффициентов Котеса, известно число разбиений интервала n и порядок N квадратурной формулы Ньютона–Котеса. Приведем алгоритм вычисления значения определенного интеграла по формулам Ньютона–Котеса.

1. Определить шаг $h = (b - a) / n$ разбиения отрезка $[a; b]$.
2. Сформировать массив X , соответствующий абсциссам точек разбиения отрезка.
3. Начать процесс вычисления значения определенного интеграла, положив $\text{Sum} = 0$.
4. Организовать цикл, параметр которого i будет изменяться от 0 до $n - 1$ и обозначать i -е разбиение интервала $[a; b]$ на более мелкие отрезки. В цикле необходимо выполнить следующее.
5. Определить шаг dh разбиения отрезка $[X_{i+1}; X_i]$.
6. Сформировать массив значений x точек разбиения отрезка $[X_{i+1}; X_i]$.
7. Вычислить значения подынтегральной функции в этих точках, сформировав массив y .
8. Выполнить расчет коэффициентов A по формуле (6.13).
9. Определить значение определенного интеграла S на интервале $[X_{i+1}; X_i]$ по формуле (6.15).
10. Добавить найденное значение S к общей сумме Sum и перейти к п.5.

Значение интеграла Sum будет найдено, когда будут определены и просуммированы значения всех определенных интегралов

$$\int_{X_i}^{X_{i+1}} y \, dx.$$

Блок-схема алгоритма вычисления значения определенного интеграла по квадратурным формулам Ньютона–Котеса изображена на рис. 6.10 (пример программирования см. в задаче 6.4).

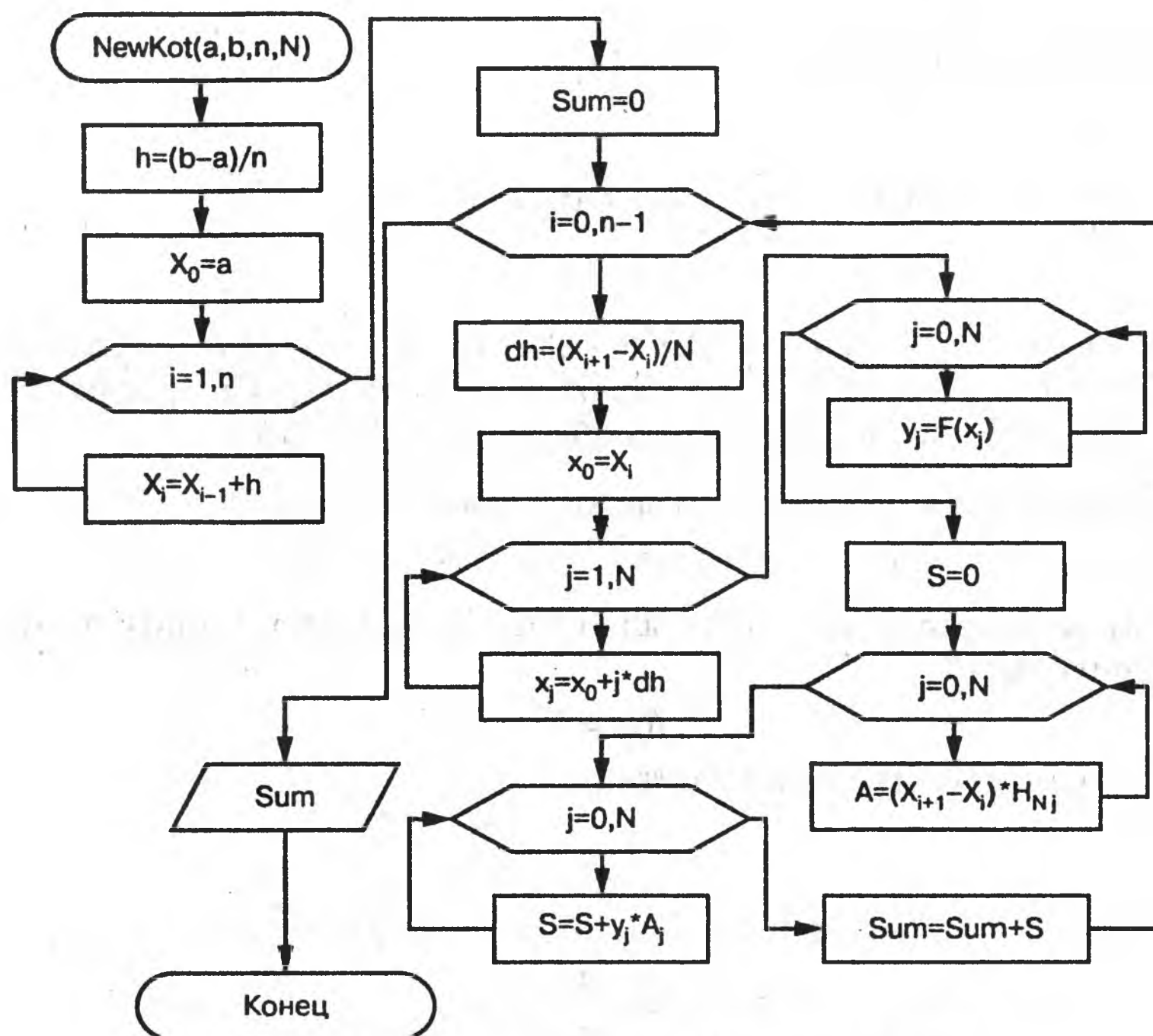


Рис. 6.10 ▼ Алгоритм метода Ньютона–Котеса

6.1.5. Квадратурные формулы Чебышева

Напомним общую квадратурную формулу, выведенную в предыдущем разделе:

$$\int_a^b y \, dx = \sum_{i=0}^n A_i \cdot f(x_i).$$

Во всех рассмотренных нами квадратурных формулах x_i располагались на равном расстоянии h . В формулах, предложенных Чебышевым, квадратурные коэффициенты рассчитываются специальным образом.

Сделаем линейную замену

$$x = \frac{a+b}{2} + \frac{b-a}{2} \cdot t \quad (6.16)$$

и преобразуем исходный интеграл к интегралу с промежутком интегрирования $[-1;1]$:

$$\int_a^b y \, dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{a+b}{2} + \frac{b-a}{2} \cdot t\right) dt. \quad (6.17)$$

Далее будем рассматривать интеграл

$$\int_{-1}^1 f(t) dt$$

и соответствующую ему квадратурную форму вида

$$\sum_{i=0}^n B_i \cdot f(t_i). \quad (6.18)$$

Обратите внимание, что в (6.18) n – это число узлов интегрирования t_i , а не число разбиений отрезка $[a; b]$, как это было в квадратурных формулах Ньютона–Котеса. Причем t_i выбирают таким образом [11], чтобы

► коэффициенты B_i были равны между собой:

$$B_1 = B_2 = \dots = B_n = B; \quad (6.19)$$

► квадратурная формула (6.18) была точной для всех полиномов до степени n включительно

$$f(t) \equiv 1. \quad (6.20)$$

Учитывая эти условия, будем иметь

$$2 = \sum_{i=1}^n B_i \Rightarrow B = \frac{2}{n}.$$

Следовательно, квадратурная формула Чебышева имеет вид:

$$\int_{-1}^1 f(t) dt = \frac{2}{n} \cdot \sum_{i=1}^n f(t_i). \quad (6.21)$$

или исходя из (6.16) и (6.17):

$$\int_a^b y dx = \frac{b-a}{2} \cdot \sum_{i=1}^n f\left(\frac{a+b}{2} + \frac{b-a}{2} \cdot t_i\right). \quad (6.22)$$

Заметим, что формула (6.21), согласно условию (6.20), должна быть точной для функций вида

$$f(t) = t, t^2, \dots, t^n.$$

Подставляя эти функции в (6.20), получим систему уравнений, из которых могут быть определены неизвестные t_i ($i = 1, 2, \dots, n$):

$$\begin{cases} t_1 + t_2 + \dots + t_n = 0, \\ t_1^2 + t_2^2 + \dots + t_n^2 = \frac{n}{3}, \\ t_1^3 + t_2^3 + \dots + t_n^3 = 0, \\ t_1^4 + t_2^4 + \dots + t_n^4 = \frac{n}{5}, \\ \dots \\ t_1^n + t_2^n + \dots + t_n^n = \frac{n \cdot (1 - (-1)^{n+1})}{2 \cdot (n+1)}. \end{cases} \quad (6.21)$$

Доказано, что система (6.21) имеет единственный набор решений t_i при $i = 1, 2, 3, 4, 5, 6, 7, 9$ и не имеет действительных решений при $i = 8$ и $i \geq 10$. В табл. 6.1 приведены значения корней t_i системы (6.21) для $n = 2 \dots 7$.

Таблица 6.1 ▼ Значения t_i в формуле Чебышева

n	i	t_i
2	1; 2	$\pm 0,577350$
3	1; 3	$\pm 0,707107$
	2	0
4	1; 4	$\pm 0,794654$
	2; 3	$\pm 0,187592$
5	1; 5	$\pm 0,832498$
	2; 4	$\pm 0,374541$
	3	0
6	1; 6	$\pm 0,866247$
	2; 5	$\pm 0,422519$
	3; 4	$\pm 0,266635$
7	1; 7	$\pm 0,883862$
	2; 6	$\pm 0,529657$
	3; 5	$\pm 0,323912$
	4	0

Итак, если заданы пределы интегрирования a и b , известно число n и определены коэффициенты Чебышева t_i , то алгоритм решения определенного

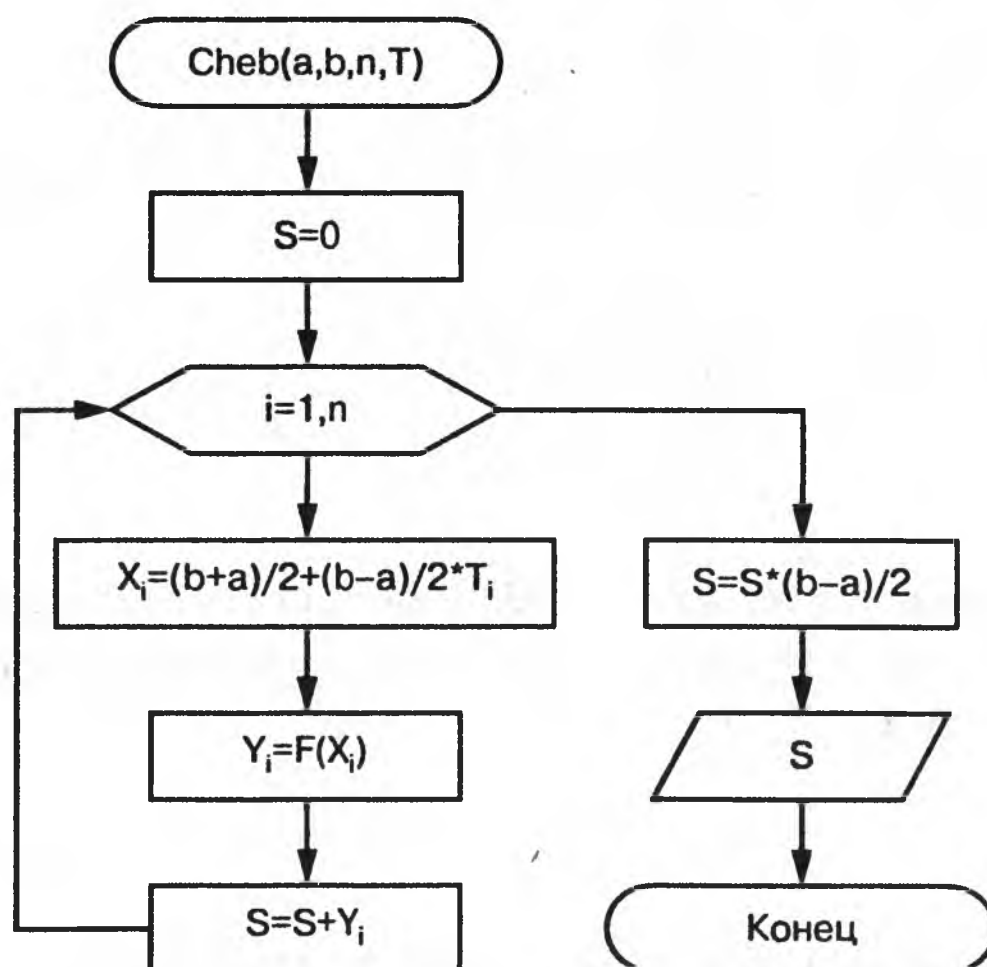


Рис. 6.11 ▼ Алгоритм интегрирования по Чебышеву

интеграла с помощью квадратурных формул Чебышева будет выглядеть так (рис. 6.11):

1. Формирование массива абсцисс

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cdot t_i.$$

2. Расчет значений функции $y_i = F(x_i)$.
3. Вычисление значения определенного интеграла как суммы y_i .

Практическое применение этого алгоритма можно увидеть в задаче 6.4.

6.1.6. Квадратурные формулы Гаусса

Вернемся к общему виду квадратурной формулы для функции, заданной на промежутке $[-1; 1]$

$$\int_{-1}^1 f(t) dt = \sum_{i=1}^n A_i \cdot f(t_i). \quad (6.22)$$

Попытаемся найти коэффициенты A_i и узловые точки t_i таким образом, чтобы квадратурная формула (6.22) была точной для всех полиномов $f(t)$ степени $2n-1$:

$$f(t) = 1, t, t^2, \dots, t^{n-1}.$$

В этом случае построение квадратурной формулы вида (6.22) приводит к определению A_i и t_i из нелинейной системы $2n$ уравнений:

$$\begin{cases} \sum_{i=1}^n A_i = 2, \\ \sum_{i=1}^n A_i \cdot t_i = 0, \\ \dots \\ \sum_{i=1}^n A_i \cdot t_i^{2n-2} = \frac{2}{2n-1}, \\ \sum_{i=1}^n A_i \cdot t_i^{2n-1} = 0. \end{cases} \quad (6.23)$$

Решение нелинейной системы (6.23) задача нетривиальная, но ее можно обойти, если знать, что значениями t_i квадратурной формулы (6.22) служат корни многочлена Лежандра $P_n(t)$:

$$P_n(t) = \frac{1}{2^n \cdot n!} \cdot \frac{d^n}{dt^n} \left((t^2 - 1)^n \right), \quad (n = 0, 1, 2, \dots). \quad (6.24)$$

Как известно, корни полинома (6.24) существуют при любом n , различны и принадлежат интервалу $[-1; 1]$.

Значения A_i вычисляются интегрированием базисных многочленов Лежандра $P_i(t)$ степени $n-1$:

$$A_i = \int_{-1}^1 \frac{(t-t_1) \dots (t-t_{i-1}) \cdot (t-t_{i+1}) \dots (t-t_n)}{(t_1-t_1) \dots (t_1-t_{i-1}) \cdot (t_1-t_{i+1}) \dots (t_1-t_n)} dt. \quad (6.25)$$

Итак, выражение (6.22) или

$$\int_a^b f(x) dx = \frac{b-a}{2} \cdot \sum_{i=1}^n A_i \cdot f\left(\frac{a+b}{2} + \frac{b-a}{2} \cdot t_i\right), \quad (6.26)$$

где t_i – корни полинома (6.24), а A_i определяется из (6.25), называют *квадратурной формулой Гаусса*. Для удобства практического применения квадратурной формулы Гаусса значения t_i и A_i для $n = 2, 3, 4, 5$ приведены в табл. 6.2.

Таблица 6.2 ▼ Значения t_i и A_i в формуле Гаусса

n	i	t_i	A_i
2	1; 2	$\pm 0,577350$	1
3	1; 3	$\pm 0,707107$	$5/9$
	2	0	$8/9$
4	1; 4	$\pm 0,794654$	0,347855

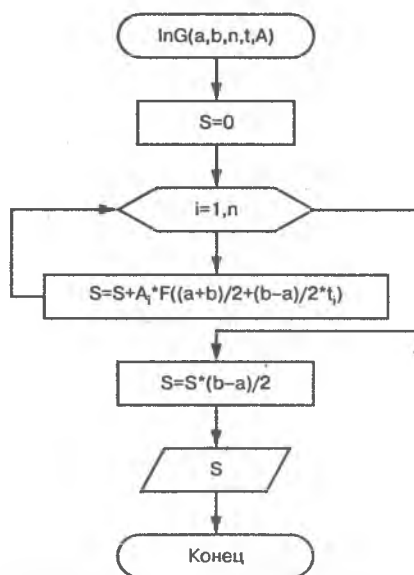


Рис. 6.12 ▼ Алгоритм интегрирования по Гауссу

Таблица 6.2 ▼ Значения t_i и A_i в формуле Гаусса (окончание)

n	i	t_i	A_i
5	2; 3	$\pm 0,187592$	0,652145
	1; 5	$\pm 0,832498$	0,236927
	2; 4	$\pm 0,374541$	0,478629
3	0	0	0,568889

Алгоритм (рис. 6.12) вычисления значения определенного интеграла по квадратурным формулам Гаусса представлен блок-схемой на рис. 6.12. Пример программного использования этого алгоритма приведен в задаче 6.5.

6.1.7. Вычисление интеграла с заданной точностью

При одном и том же числе точек формула Симпсона в большинстве случаев намного точнее, чем формулы прямоугольников и трапеций [11].

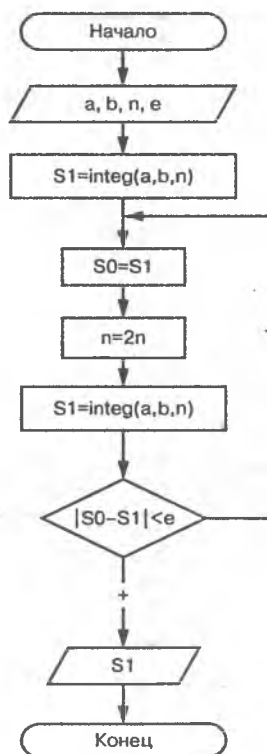


Рис. 6.13 ▼ Численное интегрирование с автоматическим выбором шага

Точность интегрирования ε по методу прямоугольников с постоянным шагом h связана с величиной шага следующим образом:

$$\varepsilon \approx h.$$

При интегрировании методом трапеций это соотношение имеет вид:

$$\varepsilon \approx h^2.$$

Метод Симпсона обеспечивает точность вычислений

$$\varepsilon \approx h^2 + h^4.$$

Ошибка в выборе величины шага интегрирования либо не обеспечит нужной точности, либо приведет к необоснованным затратам машинного времени. Заданную точность при минимальных затратах времени обеспечивают алгоритмы интегрирования с автоматическим выбором величины шага. Блок-схема вычисления интеграла с заданной точностью приведена на рис. 6.13 (пример программирования в задаче 6.6). Здесь $\text{Integ}(a, b, n)$ – функция, вычисляющая значение интеграла на участке от a до b любым из приведенных выше методов с заданной точностью ε , n – число разбиений подынтегральной функции на заданном участке интегрирования.

6.2. Численное дифференцирование

Выполнить аналитическое вычисление значения производной заданной функции $f(x)$ в некоторой точке x не столь актуальная задача по сравнению с численным интегрированием. Тем не менее в практических задачах довольно часто требуется вычислить производные разных порядков от функций, заданных таблично или имеющих сложное аналитическое выражение. В этих случаях используют *приближенное дифференцирование*.

Практически все формулы приближенного дифференцирования основаны на том, что данную функцию $f(x)$ на отрезке $[a, b]$ заменяют интерполирующим полиномом $P(x)$, а затем полагают

$$f'(x) = P'(x), \quad a \leq x \leq b.$$

Запишем интерполяционный полином Ньютона (см. раздел 2.1.2) для данной функции $y = f(x)$, если известно, что $x_i = x_0 + ih$ ($i = 0, 1, \dots, n$) и $y_i = f(x_i)$, а также определены конечные разности $\Delta^k y_i$:

$$f(x) \approx y_0 + q\Delta y_0 + \frac{q(q-1)}{2!}\Delta^2 y_0 + \frac{q(q-1)(q-2)}{3!}\Delta^3 y_0 + \dots + \frac{q(q-1)\dots(q-n+1)}{n!}\Delta^n y_0,$$

где

$$q = \frac{x - x_0}{h}, \quad h = x_{i+1} - x_i \quad (i = 0, 1, \dots).$$

Так как

$$\frac{dy}{dx} = \frac{dy}{dq} \cdot \frac{dq}{dx} = \frac{1}{h} \cdot \frac{dy}{dq},$$

то получаем конечноразностную формулу численного дифференцирования [9]:

$$f'(x) \approx \frac{1}{h} \cdot \left(\Delta y_0 + \frac{2q-1}{2} \Delta^2 y_0 + \frac{3q^2-6q+2}{6} \Delta^3 y_0 + \frac{4q^3-18q^2+22q-6}{24} \Delta^4 y_0 + \dots \right). \quad (6.27)$$

Если требуется найти производную функции в основных табличных точках x_p , то формула (6.27) упрощается, так как каждое табличное значение можно принять за начальное, то есть $x = x_0, q = 0$:

$$f'(x) = \frac{1}{h} \cdot \left(\Delta y_0 - \frac{\Delta^2 y_0}{2} + \frac{\Delta^3 y_0}{3} - \frac{\Delta^4 y_0}{4} + \frac{\Delta^5 y_0}{5} - \dots \right).$$

Построим интерполяционный полином Лагранжа (см. раздел 2.1.3) для равноотстоящих точек x_p , таких что $x_{i+1} - x_i = h \quad (i = 0, 1, 2, \dots, n-1)$:

$$L_n(x) = \sum_{i=0}^n p_i(x) \cdot y_i,$$

где

$$p_i(x) = \frac{(x - x_0) \cdot (x - x_1) \dots (x - x_{i-1}) \cdot (x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0) \cdot (x_i - x_1) \dots (x_i - x_{i-1}) \cdot (x_i - x_{i+1}) \dots (x_i - x_n)}.$$

Введем замену $x = x_0 + qh$ и перепишем полином Лагранжа в соответствии с ней:

$$L_n(x_0 + qh) = \sum_{i=0}^n \frac{(-1)^{n-i} \cdot y_i}{i! \cdot (n-i)!} \cdot \frac{q \cdot (q-1) \dots (q-n)}{q-i}.$$

Отсюда, учитывая, что

$$\frac{dx}{dq} = h,$$

получаем формулу численного дифференцирования:

$$y'(x) \approx L'_n(x) = \frac{1}{h} \cdot \sum_{i=0}^n \frac{(-1)^{n-i} \cdot y_i}{i! \cdot (n-i)!} \cdot \frac{d}{dq} \frac{q \cdot (q-1) \dots (q-n)}{q-i}. \quad (6.28)$$

Проведем расчет по формуле (6.28) для трех точек ($n = 2$):

$$y'(x) \approx L'_2(x) = \frac{1}{h} \cdot \left(\frac{1}{2} y_0 (2q - 3) - y_1 (2q - 2) + \frac{1}{2} y_2 (2q - 1) \right).$$

Таким образом, для производных $y(x_i) = y_i (i = 0, 1, 2)$ получим

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

$$y'_1 = \frac{1}{2h} \cdot (-y_0 + y_2),$$

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

При вычислении производных высших порядков поступают аналогично.

6.3. Интегрирование и дифференцирование в Mathcad

Перейдем к изучению возможностей Mathcad в вопросах интегрирования и дифференцирования.

Выполнить вычисление *неопределенного интеграла* в Mathcad не составит труда. Достаточно воспользоваться оператором неопределенного интегрирования, указав подынтегральную функцию и переменную интегрирования. Применение символьной операции приведет к выводу аналитического выражения интеграла. Фрагмент рабочего документа, приведенный на рис. 6.14, демонстрирует интегрирование в Mathcad.

$$\begin{aligned} \int \frac{x}{x^2 + 1} dx &\rightarrow \frac{1}{2} \cdot \ln(x^2 + 1) \\ \int \ln(x)^3 dx &\rightarrow \ln(x)^3 \cdot x - 3 \cdot x \cdot \ln(x)^2 + 6 \cdot x \cdot \ln(x) - 6 \cdot x \\ \int \sqrt{9 - 4 \cdot x^2} dx &\rightarrow \frac{1}{2} \cdot x \cdot (9 - 4 \cdot x^2)^{\frac{1}{2}} + \frac{9}{4} \cdot \operatorname{asin}\left(\frac{2}{3} \cdot x\right) \\ \int e^{3 \cdot x} \cdot \cos(5 \cdot x) dx &\rightarrow \frac{3}{34} \cdot e^{3 \cdot x} \cdot \cos(5 \cdot x) + \frac{5}{34} \cdot e^{3 \cdot x} \cdot \sin(5 \cdot x) \\ \int \frac{x - 1}{x^2 + 2x + 3} dx &\rightarrow \frac{1}{2} \cdot \ln(x^2 + 2 \cdot x + 3) - 2^{\frac{1}{2}} \cdot \operatorname{atan}\left[\frac{1}{4} \cdot (2 \cdot x + 2) \cdot 2^{\frac{1}{2}}\right] \end{aligned}$$

Рис. 6.14 ▼ Вычисление неопределенных интегралов

ЗАДАЧА 6.1. Построить семейство интегральных линий функции $f(x)$.
Решение задачи приведено на рис. 6.15.

$$f(x) := \frac{1}{\sqrt{1+2 \cdot x-3 \cdot x^2}} \quad F(x) := \int f(x) dx \rightarrow \frac{1}{3} \cdot 3^{\frac{1}{2}} \cdot \arcsin\left(\frac{3}{2} \cdot x - \frac{1}{2}\right)$$

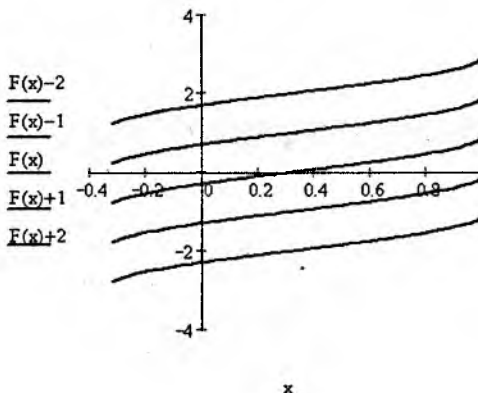


Рис. 6.15 ▾ Семейство интегральных линий

Для того чтобы вычислить *определенный интеграл* в Mathcad, можно воспользоваться оператором определенного интегрирования, который отличается от неопределенного тем, что имеет поля ввода для значений концов интервала интегрирования. Кроме того, значение определенного интеграла может быть получено как в численном виде, так и в символьном. Примеры приведены на рис. 6.16.

Численное решение

$$\int_5^{13} \sqrt{2 \cdot x - 1} dx = 32.667$$

$$\int_{-1}^{\frac{1}{2}} \sqrt{1-x^2} dx = 0.307$$

$$\int_0^{\frac{\pi}{2}} \sin(x)^4 dx = 0.589$$

Символьное решение

$$\int_5^{13} \sqrt{2 \cdot x - 1} dx \rightarrow \frac{98}{3}$$

$$\int_{-1}^{\frac{1}{2}} \sqrt{1-x^2} dx \rightarrow \frac{-1}{8} \cdot 3^{\frac{1}{2}} + \frac{1}{6} \cdot \pi$$

$$\int_0^{\frac{\pi}{2}} \sin(x)^4 dx \rightarrow \frac{3}{16} \cdot \pi$$

Рис. 6.16 ▾ Вычисление определенных интегралов

ЗАДАЧА 6.2. Вычислить площадь фигуры, ограниченной линиями $y^2 = 9x$, $y = 3x$.

Итак, фигура ограничена прямой и параболой, развернутой относительно оси абсцисс. Разрешив уравнение параболы относительно x (рис. 6.17), получим функции, описывающие ее ветви. Построим графики линий и найдем точки их пересечения. Площадь образованной фигуры будем искать как разность площадей фигуры, ограниченной положительной ветвью параболы и фигуры, ограниченной прямой. Каждая из рассматриваемых фигур ограничена слева осью ординат, а справа – прямой $x = 1$. Следовательно, интегрирование проводится на интервале $[0; 1]$.



Рис. 6.17 ▽ Решение задачи 6.2

Рассмотрим *интегралы с бесконечными пределами*. Примеры сходящихся и не сходящихся интегралов показаны на рис. 6.18.

Интегралы сходятся

$$\int_0^{\infty} \frac{1}{x^2 + 1} dx \rightarrow \frac{1}{2} \cdot \pi$$

$$\int_1^{\infty} \frac{1}{x^2 \cdot (1 + e^x)} dx = 0.119$$

$$\int_{-\infty}^{\infty} \frac{1}{x^2 + 1} dx \rightarrow \pi$$

Интегралы расходятся

$$\int_1^{\infty} \frac{x+1}{\sqrt{x^3}} dx \rightarrow \infty$$

$$\int_2^{\infty} \frac{\ln(x)}{x} dx \rightarrow \infty$$

$$\int_e^{\infty} \frac{1}{x \cdot \ln(x)} dx \rightarrow \infty$$

Рис. 6.18 ▽ Несобственные интегралы

Примеры интегралов от разрывных функций приведены на рис. 6.19.

$$\int_0^1 \frac{1}{\sqrt{1-x^2}} dx \rightarrow \frac{1}{2} \cdot \pi$$

Функция имеет разрыв справа, интеграл сходится

$$\int_1^2 \frac{x}{\sqrt{x-1}} dx \rightarrow \frac{8}{3}$$

Функция имеет разрыв слева, интеграл сходится

$$\int_3^5 \frac{x^2}{\sqrt{(x-3) \cdot (5-x)}} dx \rightarrow \frac{33}{2} \cdot \pi$$

Функция имеет разрыв справа и слева, интеграл сходится

$$\int_{-1}^1 \frac{3 \cdot x^2 + 2}{\sqrt[3]{x^2}} dx \rightarrow \frac{102}{7}$$

Функция имеет разрыв на интервале, интеграл сходится

$$\int_1^2 \frac{1}{x \cdot \ln(x)} dx \rightarrow \infty$$

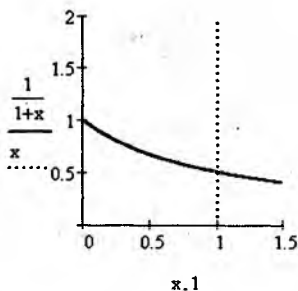
Функция имеет разрыв на интервале, интеграл расходится

Рис. 6.19 ▽ Интегралы от разрывных функций

ЗАДАЧА 6.3. Вычислить определенный интеграл по квадратурным формулам прямоугольников:

$$\int_0^1 \frac{1}{x+1} dx.$$

На рис. 6.20 можно увидеть геометрическую интерпретацию задачи и ее решение при помощи оператора определенного интегрирования Mathcad.



$$\int_0^1 \frac{1}{1+x} dx = 0.69315$$

Рис. 6.20 ▽ Решение задачи 6.3

В разделе 6.1.1 приводились квадратурные формулы левых, правых и средних прямоугольников, а также были описаны алгоритмы. Приведем их реализацию в виде Mathcad-функций.

Рис. 6.21 представляет Mathcad-функцию интегрирования по формулам левых прямоугольников.

$$\text{In1}(a, b, n, f) := \left| \begin{array}{l} h \leftarrow \frac{b - a}{n} \\ S \leftarrow 0 \\ \text{for } j \in 1..n \\ \quad \left| \begin{array}{l} x_{j-1} \leftarrow a + h \cdot (j - 1) \\ S \leftarrow S + f(x_{j-1}) \end{array} \right. \\ S \leftarrow S \cdot h \\ S \end{array} \right.$$

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

$$\text{In1}(0, 1, 100, f) = 0.69565$$

Рис. 6.21 ▼ Решение задачи 6.3 по формулам левых прямоугольников

Интегрирование заданной функции по формулам правых прямоугольников представлено на рис. 6.22.

$$\text{In2}(a, b, n, f) := \left| \begin{array}{l} h \leftarrow \frac{b - a}{n} \\ S \leftarrow 0 \\ \text{for } j \in 1..n \\ \quad \left| \begin{array}{l} x_j \leftarrow a + h \cdot (j) \\ S \leftarrow S + f(x_j) \end{array} \right. \\ S \leftarrow S \cdot h \\ S \end{array} \right.$$

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

$$\text{In2}(0, 1, 100, f) = 0.69065$$

Рис. 6.22 ▼ Решение задачи 6.3 по формулам правых прямоугольников



И наконец, Mathcad-функция интегрирования по формулам средних прямоугольников приведена на рис. 6.23.

$$\text{In3}(a, b, n, f) := \left| \begin{array}{l} h \leftarrow \frac{b-a}{n} \\ S \leftarrow 0 \\ \text{for } j \in 1..n \\ \quad \left| \begin{array}{l} x_j \leftarrow a + h \cdot (j) \\ S \leftarrow S + f\left(x_j - \frac{h}{2}\right) \end{array} \right. \\ S \leftarrow S \cdot h \\ S \end{array} \right.$$

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

$$\text{In3}(0, 1, 100, f) = 0.69314$$

Рис. 6.23 ▼ Решение задачи 6.3 по формулам средних прямоугольников

Нетрудно заметить, что значение, вычисленное по формулам средних прямоугольников, меньше всего отличается от значения, которое выдал оператор интегрирования Mathcad.

Формирование матрицы коэффициентов Котеса

$k := 20$ Порядок матрицы коэффициентов

$M := 1..L_{\text{max}} := 0..k$

$$H_{M,L} := \text{if} \left[M > L, \frac{(-1)^{M-L}}{M \cdot L! \cdot (M-L)!} \int_0^M \prod_{j=0}^M \text{if}(j = L, 1, q - j) dq, 0 \right]$$

$$H_{M,M} := 1 - \sum_{L=0}^M H_{M,L}$$

$H =$

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0.5	0.5	0	0	0	0
2	0.167	0.667	0.167	0	0	0
3	0.125	0.375	0.375	0.125	0	0
4	0.078	0.356	0.133	0.356	0.078	0
5	0.066	0.26	0.174	0.174	0.26	0.066
6	0.049	0.257	0.032	0.324	0.032	0.257

Рис. 6.24 ▼ Формирование матрицы коэффициентов Котеса

ЗАДАЧА 6.4. Вычислить определенный интеграл из задачи 6.3 по квадратурным формулам Ньютона–Котеса третьего порядка.

Алгоритм вычисления интеграла по квадратурным формулам Ньютона–Котеса описан в разделе 6.1.4. Из описания понятно, что, прежде чем создавать Mathcad-функцию, реализующую этот алгоритм, необходимо сформировать матрицу коэффициентов Котеса. Как это сделать средствами Mathcad, показано на рис. 6.24.

Рис. 6.25 содержит текст Mathcad-функции, вычисляющей интеграл по формулам Ньютона–Котеса, и результат ее работы. Как видим, они практически не отличаются от значения оператора интегрирования Mathcad (рис. 6.20).

```

NewKotGL(a,b,n,F) :=
    Sum ←  $\frac{b-a}{n}$ 
    X0 ← a
    for i ∈ 1..n
        Xi ← Xi-1 + Sum
    Sum ← 0
    for i ∈ 0..n-1
        h ←  $\frac{Xi+1 - Xi}{N}$ 
        x0 ← Xi
        for j ∈ 1..N
            xj ← x0 + j·h
        for j ∈ 0..N
            yj ← F(xj)
        S ← 0
        for j ∈ 0..N
            Aaj ← (Xi+1 - Xi)·HN,j
        for j ∈ 0..N
            S ← S + yj·Aaj
        Sum ← Sum + S
    Sum

```

$$F(x) := \frac{1}{x+1} \quad \text{NewKotGL}(0,1,100,3,F) = 0.69315$$

Рис. 6.25 ▼ Решение задачи 6.4

Вычисление производных в Mathcad, даже от достаточно сложных функций (рис. 6.26), также процесс не обременительный, необходимо только правильно заполнить конструкцию оператора дифференцирования.

$$\frac{d}{dx} \left(\frac{5 \cdot \sin(2 \cdot x)}{\sqrt{\cos(x \cdot 2)}} \right) \rightarrow -10 \cdot \cos(2 \cdot x) \frac{1}{2} - 5 \cdot \frac{\sin(2 \cdot x)^2}{\cos(2 \cdot x)^{\frac{3}{2}}}$$

$$\frac{d}{dx} \tan \left(\sqrt[3]{\ln(x)} \right) \rightarrow \frac{1}{3} \cdot \frac{1 + \tan \left(\ln(x)^{\frac{1}{3}} \right)^2}{\ln(x)^{\frac{2}{3}} \cdot x}$$

Рис. 6.26 ▼ Вычисление производной

На рис. 6.27 приведен пример вычисления *производной от функции, заданной параметрическом виде*. Напомним, что если функция $y(x)$ задана параметрическими уравнениями $x = \varphi(t)$ и $y = \rho(t)$, то $y'(x) = \rho'(t)/\varphi'(t) = y_i/x_i$.

$$x(t) := 3 \cdot \cos(t)^3 \quad y(t) := 3 \cdot \sin(t)^3$$

$$\frac{\frac{d}{dt} y(t)}{\frac{d}{dt} x(t)} \rightarrow \frac{-\sin(t)}{\cos(t)} \quad \begin{array}{l} \text{Производная от функции.} \\ \text{заданной в параметрическом виде} \end{array}$$

Рис. 6.27 ▼ Вычисление производной от параметрической функции

Фрагмент документа, приведенного на рис. 6.28, демонстрирует процесс вычисления *производной неявной функции*. Здесь важно помнить: y есть функция от x . Кроме того, в результате обычно получается выражение, в которое кроме функции $y(x)$ и ее производная $y'(x)$. Поэтому при необходимости результат вычислений можно преобразовать и выразить $y'(x)$ через x и $y(x)$.

$$\frac{d}{dx} \left(\frac{y(x)^2}{3} - \frac{x^2}{5} - 1 \right) \rightarrow \frac{2}{3} \cdot y(x) \cdot \frac{d}{dx} y(x) - \frac{2}{5} \cdot x$$

$$\frac{2}{3} \cdot y(x) \cdot \frac{d}{dx} y(x) - \frac{2}{5} \cdot x \text{ solve, } \frac{d}{dx} y(x) \rightarrow \frac{3}{5} \cdot \frac{x}{y(x)}$$

$$\frac{d}{dx} (\sin(y(x)) - \ln(x)) \rightarrow \cos(y(x)) \cdot \frac{d}{dx} y(x) - \frac{1}{x}$$

$$\cos(y(x)) \cdot \frac{d}{dx} y(x) - \frac{1}{x} \text{ solve, } \frac{d}{dx} y(x) \rightarrow \frac{1}{\cos(y(x)) \cdot x}$$

Рис. 6.28 ▼ Вычисление производной неявной функции

Решена в Mathcad и достаточно сложная задача вычисления производной функции вида $[f(x)]^{g(x)}$. Пример приведен на рис. 6.29.

$$\frac{d}{dx} x^x \rightarrow x^x \cdot (\ln(x) + 1)$$

$$\frac{d}{dx} \sin(x)^{\tan(x)} \rightarrow \sin(x)^{\tan(x)} \cdot \left[(1 + \tan(x)^2) \cdot \ln(\sin(x)) + \tan(x) \cdot \frac{\cos(x)}{\sin(x)} \right]$$

Рис. 6.29 ▼ Вычисление производной функции вида $[f(x)]^{g(x)}$

Примеры, приведенные на рис. 6.30, демонстрируют вычисление производных различных порядков. Заметим, что выражение для вычисления второй производной параметрической функции легко выводится из формулы $y'(x) = \rho'(t)/\varphi'(t)$.

Производная второго порядка

$$\frac{d^2}{dx^2} \ln(\cos(x)) \rightarrow -1 - \frac{\sin(x)^2}{\cos(x)^2}$$

Производная четвертого порядка

$$\frac{d^4}{dx^4} \tan(x) \rightarrow 16 \cdot (1 + \tan(x)^2)^2 \cdot \tan(x) + 8 \cdot \tan(x)^3 \cdot (1 + \tan(x)^2)$$

Вторая производная параметрической функции

$$x(t) := t - \sin(t) \quad y(t) := 1 - \cos(t)$$

$$\frac{\frac{d}{dt} x(t) \cdot \frac{d^2}{dt^2} y(t) - \frac{d}{dt} y(t) \cdot \frac{d^2}{dt^2} x(t)}{\left(\frac{d}{dt} x(t) \right)^2} \rightarrow \frac{(1 - \cos(t)) \cdot \cos(t) - \sin(t)^2}{(1 - \cos(t))^2}$$

Третья производная от неявной функции

$$\frac{d^3}{dx^3} (x^2 + z(x)^2 - 3) \rightarrow 6 \cdot \frac{d}{dx} z(x) \cdot \frac{d}{dx} \frac{d}{dx} z(x) + 2 \cdot z(x) \cdot \frac{d}{dx} \frac{d}{dx} \frac{d}{dx} z(x)$$

Рис. 6.30 ▼ Вычисление производных различных порядков

6.4. Вычисление интегралов и производных в MATLAB

Теперь перейдем к изучению возможностей MATLAB при интегрировании и дифференцировании.

Символьное вычисление *неопределенных интегралов* в MATLAB осуществляется при помощи функции

`int (<функция>, <переменная интегрирования>)`

Особенностью этой функции является то, что входными аргументами должны быть символьная функция, или ее имя, и символьная переменная. Листинг 6.1 содержит пример вычисления неопределенного интеграла.

Листинг 6.1

```
>> syms x %Определение символьной переменной
>> f=sym("x/(x^2+1)"); %Определение символьной функции
>> int(f,x)%Вычисление неопределенного интеграла
ans =
  1/2*log(x^2+1)
>> g=sym("ln(x)^3"); %Определение символьной функции
>> int(g,x) %Вычисление неопределенного интеграла
ans =
  log(x)^3*x-3*x*log(x)^2+6*x*log(x)-6*x
```

Определенные интегралы также можно вычислять символьно, используя функцию `int (<функция>, <переменная интегрирования>, <нижний предел>, <верхний предел>)`

Здесь символьными должны быть не только функция и переменная интегрирования, но и пределы интегрирования. Листинг 6.2 содержит примеры аналитического вычисления определенных и *несобственных* интегралов.

Листинг 6.2

```
>> syms x a b %Определение символьных переменных
>> f=sym("(2*x-1)^(1/2)"); %Определение символьной функции
>> int(f,x,a,b) %Вычисление определенного интеграла
ans =
  1/3*(2*b-1)^(3/2)-1/3*(2*a-1)^(3/2)
>> %-----
>> f=sym("1/(x^2+1)"); %Определение символьной функции
>> %Вычисление несобственного интеграла
>> %в диапазоне от 1 до бесконечности
>> int(f,x,0,inf)
ans = % Интеграл сходится
  1/2*pi
>> %-----
>> f=sym("1/(x*ln(ln(x)))"); %Определение символьной функции
>> %Вычисление несобственного интеграла
>> %в диапазоне от e^2 до бесконечности
>> int(f,x,exp(2),inf)
ans = %Интеграл расходится
  Inf
```

```
>>%-----
>>%Вычисление интеграла от функции,
>>%которая терпит разрыв на интервале
>> F=sym("1/(1-x^2)^(1/2)");
>> int(F,x,0,1)
ans = %Интеграл сходится
1/2*pi
```

Для вычисления численного значения *определенного интеграла* предназначена функция

```
quad(<имя функции>, <нижний предел>,<верхний предел>)
```

Первым параметром этой команды должно быть имя файл-функции, вычисляющей подынтегральное выражение. Второй и третий параметры – это нижний и верхний пределы интегрирования. Листинг 6.3 содержит примеры численного и для сравнения символьного вычисления определенного интеграла.

ЛИСТИНГ 6.3

```
%Функция для вычисления подынтегрального выражения
function y=Fi(t)
y=sin(t).^4;
end
%-----
%Вычисление численного значения интеграла
>> quad("Fi",0,pi/2)
ans =
0.5890
%-----
>> g=sym("sin(x)^4"); %Определение символьной функции
>> syms x %Определение символьной переменной
%Вычисление значения интеграла в символьном виде
>> int(g,x,0,pi/2)
ans =
3/16*pi
```

ЗАДАЧА 6.4. Найти интеграл

$$\int_0^1 \frac{x}{1+x} dx$$

по формулам Чебышева пятого порядка.

Листинг 6.4 содержит две М-функции. Первая вычисляет значение подынтегрального выражения, а вторая реализует вычисление интеграла по формулам Чебышева (см. раздел 6.1.5).

ЛИСТИНГ 6.4

```
%Вычисление подынтегральной функции
function y=I(x)
```

```

y=x./(1+x);
end
%-----
%Расчет интеграла по формулам Чебышева
function S=Cheb(a,b,n,T)
S=0;
for i=1:n
    x(i)=(b+a)/2+(b-a)/2*T(i);
    y(i)=I(x(i));
    S=S+y(i);
end;
S=(b-a)/n*S;
End
%-----
%Коэффициенты Чебышева для n=5(табл. 6.1)
>> t=[-0.832498 -0.374541 0 0.374541 0.832498];
>> s=Cheb(0,1,5,t)
s = %Значение интеграла
    0.3069
%-----
%Вычисление интеграла с помощью встроенной функции
>> quad("I",0,1)
ans = % Значение интеграла
    0.3069

```

ЗАДАЧА 6.5. Вычислить интеграл

$$\int_0^1 \sqrt{1+2 \cdot x} \, dx$$

по квадратурным формулам Гаусса.

Квадратурные формулы Гаусса описаны в разделе 6.1.6. Листинг 6.5 содержит функции, предназначенные для решения задачи, и результаты их работы.

Листинг 6.5

```

%Вычисление подынтегральной функции
function y=G(x)
y=(1+2*x).^(1/2);
end
%-----
%Расчет интеграла по формулам Гаусса
function S=InG(a,b,n,T,A)
S=0;
for i=1:n
    S=S+A(i)*G((a+b)/2+(b-a)/2*T(i));
end;
S=S*(b-a)/2;
End
%-----

```

```
%Коэффициенты Гаусса для n=3(табл. 6.2)
>> a=[0.555555 0.555555 0.888888];
>> t=[-0.774596 0.774596 0];
>> InG(0,1,3,t,a)
ans =%значение интеграла
1.3987

%
%Вычисление интеграла с помощью встроенной функции
>> quad("G",0,1)
ans =
1.3987
```

Символьное дифференцирование в MATLAB выполняет команда

diff(<функция>, <переменная>, <порядок>)

где <функция> – символьная запись функции или ее имя, <переменная> – переменная дифференцирования, <порядок> – порядок (номер) производной, которую необходимо найти. Листинг 6.6 содержит примеры вычисления производных аналитически.

Листинг 6.6

```
>> diff("5*sin(2*x)/(cos(2*x)^(1/2))",x,1)
ans =
10*cos(2*x)^(1/2)+5*sin(2*x)^2/cos(2*x)^(3/2)

%
>> syms x
>> F=sym("tan(ln(x)^(1/3))");
>> diff(F,x,1)
ans =
1/3*(1+tan(log(x)^(1/3))^2)/log(x)^(2/3)/x
```

Для вычисления численного значения *производной* в некоторой точке воспользуемся функцией, которая позволяет произвести подстановку одного выражения в другое:

subs(<функция>, <переменная>, <выражение>)

В общем виде функция subs вызывается с тремя параметрами: первый – символьная функция или ее имя, второй – переменная, подлежащая замене, третий – выражение, которое нужно подставить вместо переменной.

Примеры применения функции subs для вычисления значения производной в точке можно найти в листинге 6.7.

Листинг 6.7

```
>> syms x
>> f=sym("(3*x^2-7)/(2*x+1)");
>> F=diff(f,x,1)
```



```

F := %Первая производная от f
6*x/(2*x+1)-2*(3*x^2-7)/(2*x+1)^2
>> subs(F,x,2)
ans := %Значение первой производной в точке 2
2
>> subs(F,x,0)
ans := %Значение первой производной в точке 0
14
>> subs(F,x,-2)
ans := %Значение первой производной в точке -2
2.8889

```

6.5. Возможности Maple для интегрирования и дифференцирования

И в заключение ознакомимся с ресурсами, которыми обладает Maple, для интегрирования и дифференцирования функций.

Неопределенные интегралы в Maple вычисляются с помощью функций

`Int(<функция>, <переменная>)` или `int(<функция>, <переменная>)`

Примеры применения этих функций приведены на рис. 6.31. Обратите внимание, что `Int` выводит сам интеграл, а `int` – первообразную интегрирования

$$\begin{array}{l}
 > \text{Int}((9-4x)^{(1/2)}, x); \\
 \int \sqrt{9-4x} \, dx \\
 > \text{int}((9-4x)^{(1/2)}, x); \\
 \frac{(9-4x)^{(3/2)}}{6} \\
 > \text{Int}((9-4x)^{(1/2)}, x) = \text{int}((9-4x)^{(1/2)}, x); \\
 \int \sqrt{9-4x} \, dx = -\frac{(9-4x)^{(3/2)}}{6}
 \end{array}$$

Рис. 6.31 ▼ Неопределенный интеграл в Maple

Для вычисления *определенных интегралов* используют команды

`Int(<функция>, <переменная>=a..b)` или `int(<функция>, <переменная>=a..b)`

где *a* и *b* – пределы интегрирования

Пример вычисления определенного интеграла приведен на рис. 6.32. Обратите внимание, что и для определенного интеграла Maple пытается найти аналитическое выражение. Для получения численного значения определенного интеграла нужно использовать функцию evalf.

```

> Int((1-x^2)^(1/2), x=-1..-1/2);

$$\int_{-1}^{-1/2} \sqrt{1-x^2} dx$$

> int((1-x^2)^(1/2), x=-1..-1/2);

$$\frac{\pi}{6} - \frac{\sqrt{3}}{8}$$

> evalf(%);
>
0.3070924248

```

Рис. 6.32 ▼ Определенный интеграл в Maple

Вычисление несобственных интегралов также не вызывает затруднений. Примеры сходящегося и расходящегося интегралов приведены на рис. 6.33. Здесь стоит напомнить, что запись infinity обозначает бесконечность.

```

> Int(1/(1+x^2), x=0..infinity)=int(1/(1+x^2), x=0..infinity);

$$\int_0^{\infty} \frac{1}{1+x^2} dx = \frac{\pi}{2}$$

> Int(ln(x)/x, x=2..infinity)=int(ln(x)/x, x=2..infinity);

$$\int_2^{\infty} \frac{\ln(x)}{x} dx = \infty$$


```

Рис. 6.33 ▼ Несобственный интеграл в Maple

ЗАДАЧА 6.6. Вычислить определенный интеграл

$$\int_0^1 \frac{1}{x+1} dx$$

по квадратурным формулам трапеций и Симпсона.

Зададим подынтегральную функцию и для проверки вычислим интеграл с помощью функции int так, как показано на рис. 6.34.

```

> #Вычисление подынтегральной
  функции
> f:=proc(x)
> global F;
> F:=1/(x+1);
> end:
> #Вычисление интеграла с помощью
  встроенной функции
> int(f(x), x=0..1);
> #аналитическое
                                ln(2)
> #численное
> evalf(%);
0.6931471806
>

```

Рис. 6.34 ▼ Вычисление интеграла из задачи 6.6

Квадратурная формула трапеций и вычисление интеграла с ее помощью описаны в разделе 6.1.2. Рис. 6.35 содержит текст функции, созданной средствами программирования Maple, для вычисления значения определенного интеграла методом трапеций.

```

> #Метод трапеций
> in4:=proc(a,b,n)
> local j,h; global S;
> h:=(b-a)/n;
> S:=0;
> for j from 1 to n-1 do
> S:=S+f(a+h*j);
> od;
> S:=h*((f(a)+f(b))/2+S);
> evalf(S);
> end:
> #Значение интеграла
> in4(0,1,100);
                                0.6931534305

```

Рис. 6.35 ▼ Вычисление интеграла по методу трапеций

Вычисление интеграла методом Симпсона (см. раздел 6.1.3) показано на рис. 6.36.

Проанализировав числовые значения определенного интеграла, полученные в результате применения формул трапеций и Симпсона с одинаковым

```

> #Метод Симпсона
> in5:=proc(a,b,n)
> local j,h,p;global S;
> h:=(b-a)/n;
> S:=0;
> p:=4;
> for j from 1 to n-1 do
> S:=S+f(a+h*j)*p;
> p:=6-p;
> od;
> S:=h/3*(f(a)+f(b)+S);
> evalf(S);
> end:
> #Значение интеграла
> in5(0,1,100);
0.6931471809

```

Рис. 6.36 ▽ Вычисление интеграла по методу Симпсона

количеством шагов, нетрудно заметить, что квадратурные формулы Симпсона точнее формул трапеций. Воспользуемся алгоритмом, описанным в разделе 6.1.7, и уточним значение интеграла, вычисленного по квадратурным формулам трапеций.

```

> In6:=proc(a,b,N,ep)
> local S0,Pr,n,s;global S1;
> n:=N;
> S1:=in4(a,b,n);
> Pr:=0;
> while Pr=0 do
> S0:=S1;
> n:=n*2;
> S1:=in4(a,b,n);
> s:=abs(S0-S1);
> if s<ep then
> Pr:=1;fi;
> od;
> evalf(S1);
> end:
>
> In6(0,1,100,0.000001);
0.6931472782

```

Рис. 6.37 ▽ Вычисление интеграла с заданной точностью

Для вычисления производной в Maple имеются следующие функции:

`Diff(<выражение>, <переменная1>, [<переменная2>, ..., <переменнаяN>])`

или

`diff(<выражение>, <переменная1>, [<переменная2>, ..., <переменнаяN>])`

Здесь <выражение> – дифференцируемое алгебраическое выражение или функция ряда переменных, <переменная1>... – список переменных, по которым проводится дифференцирование. Различие между функциями Diff и diff видно на рис. 6.38. Первая представляет выражение в естественной математической форме, а вторая непосредственно вычисляет производную.

```
> Diff((3*x^2+7)/(2*x+1), x);
      d
      dx
      (3x^2+7)
      (2x+1)

> diff((3*x^2+7)/(2*x+1), x);
      6x      2(3x^2+7)
      2x+1 - (2x+1)^2

> Diff((3*x^2+7)/(2*x+1), x)=diff((3*x^2+7)/(2*x+1), x);
      d
      dx
      (3x^2+7)
      (2x+1) = 6x      2(3x^2+7)
                2x+1 - (2x+1)^2

> diff(x^2/7-y^2/5-1, x);
      2x
      7

> diff(x^x, x);
      x^x (ln(x)+1)

> diff(sin(y(x))-ln(x), x);
      cos(y(x)) (d
                  dx
                  y(x)) - 1/x
```

Рис. 6.38 ▽ Дифференцирование в Maple

Вычислить значение производной в точке можно, применив функцию подстановки

`subs(<выражение1>=<выражение2>, <выражение3>)`

которая в математическом выражении <выражение3> заменяет <выражение1> на <выражение2>. В частности, <выражение2> может быть числом (рис. 6.39).

Примеры вычисления производных высших порядков приведены на рис. 6.40.

```

> pr:=diff((3*x^2+7)/(2*x+1),x)
> subs(x=2,pr);
                                22
                                25
> evalf(%);
>
                                0.8800000000

```

Рис. 6.39 ▼ Вычисление значения производной в точке

```

> #Производная второго порядка
> diff(ln(cos(x)),x$2);
                                -1 -  $\frac{\sin(x)^2}{\cos(x)^2}$ 
> #Производная четвертого порядка
> diff(tan(x),x$4);
                                2
                                16 (1+tan(x)2) tan(x) + 8 tan(x)3 (1+tan(x)2)

```

Рис. 6.40 ▼ Производные высших порядков

6.6. Советы по использованию математических пакетов при интегрировании и дифференцировании

В заключение отметим, что все пакеты достаточно успешно справляются с задачами интегрирования и дифференцирования. В Mathcad и Maple результаты вычисления интегралов и производных максимально приближены к стандартному математическому виду, но это не умаляет возможностей MATLAB. Например, в нем можно не только найти значение определенного интеграла, но и отрегулировать точность вычислений (листинг 6.8).

Листинг 6.8

%Подынтегральная функция

```
function y=Fa(x)
```

```
y=(1+2*x).^(1/2);
```

```
end
```

```
>> format long %Формат вывода значений
```

```
>> quad("Fa",0,1)
```

```
ans =%Значение с точностью 1.0e-06 (по умолчанию)
```

```
1.39871741104235
```

```
>> quad("Fa",0,1,1.0e-07)
ans =% Значение с точностью 1.0e-07 (устанавливается пользователем)
1.39871747285309
>> quad("Fa",0,1,1.0e-09)
ans = Значение с точностью 1.0e-09
1.39871747422048
```

Кроме того, MATLAB может предоставить информацию о ходе вычислений (функция `quad` основана на квадратурных формулах Симпсона с автоматическим выбором шага) в виде таблицы. В каждой строке таблицы содержится значение количества вычислений, начальная точка текущего промежутка интегрирования, его длина и значение интеграла (листинг 6.9).

Листинг 6.9

```
%Вызов функций с разной степенью точности
>> quad("Fa",0,1,1.0e-05,5)
    9    0.0000000000    2.71580000e-001    0.3056582994
   11    0.2715800000    4.56840000e-001    0.6446507127
   13    0.7284200000    2.71580000e-001    0.4484083989
ans =
1.39871741104235
>> quad("Fa",0,1,1.0e-07,5)
    9    0.0000000000    2.71580000e-001    0.3056582994
   11    0.0000000000    1.35790000e-001    0.1446296661
   13    0.1357900000    1.35790000e-001    0.1610286480
   15    0.2715800000    4.56840000e-001    0.6446507127
   17    0.2715800000    2.28420000e-001    0.3038173931
   19    0.5000000000    2.28420000e-001    0.3408333666
   21    0.7284200000    2.71580000e-001    0.4484083989
ans =
1.39871747285309
```

6.7. Задания для самостоятельного решения

Найти приведенные далее интегралы аналитически и численно в Mathcad, Maple и MATLAB, используя стандартные средства пакетов. Выполнить проверку, убедившись, что производная от первообразной равна подынтегральной функции.

1. Создать М-функцию вычисления значения определенного интеграла методом левых прямоугольников, количество разбиений отрезка интегрирования $n = 10$:

$$\int_{1,2}^{1,6} \frac{1}{\sqrt{1 + 2 \cdot x^3}} dx.$$

2. Вычислить определенный интеграл, создав М-функцию, реализующую метод правых прямоугольников, количество разбиений отрезка интегрирования $n = 10$:

$$\int_{0,4}^{0,5} \frac{1}{1 + \sin(x) + x} dx.$$

3. Найти значение определенного интеграла методом средних прямоугольников для $n = 20$, создав М-функцию:

$$\int_{0,4}^{0,6} \sqrt{x} \cdot \cos(x^2) dx.$$

4. Найти заданный интеграл в Mathcad, применив метод трапеций для количества разбиений отрезка интегрирования, равного 20:

$$\int_{0,3}^{1,9} \sqrt{7 + 2 \cdot x^3} dx.$$

5. С помощью метода Симпсона вычислить интеграл в Mathcad для $n = 20$:

$$\int_{0,5}^{1,5} \frac{\sin(2 \cdot x)}{x^2 + 1} dx.$$

6. Вычислить интеграл средствами Maple, применив метод Ньютона–Котеса четвертого порядка и выбрав количество разбиений отрезка, равное 50:

$$\int_{0,2}^{1,8} \sqrt{0,2 \cdot x^2 + 1} dx.$$

7. Вычислить определенный интеграл в MATLAB, используя метод прямоугольников с автоматическим выбором шага, точность вычислений $\epsilon = 0,001$:

$$\int_{0,1}^{0,9} \frac{\sqrt{x + 1}}{\cos^2(x)} dx.$$

8. Найти средствами Maple заданный интеграл, используя квадратурные формулы Чебышева четвертого порядка:

$$\int_{1,1}^{2,1} \frac{\sqrt{x}}{e^{x^2}} dx.$$

9. Вычислить определенный интеграл средствами Maple, используя квадратурные формулы Гаусса пятого порядка:

$$\int_{0,35}^{0,75} \frac{\operatorname{tg}(x)}{\sqrt{x + 3}} dx.$$

10. Провести в Mathcad сравнительный анализ вычислений определенного интеграла методами прямоугольников, трапеций и Симпсона с автоматическим выбором шага ($\epsilon = 0,0001$), вычислить количество итераций в каждом из методов:

$$\int_{0,1}^{0,5} \frac{e^x}{1 + \sin(3 \cdot x)} dx.$$

Глава 7

Решение обыкновенных дифференциальных уравнений и систем

Дифференциальные уравнения и системы описывают очень многие динамические процессы и используются при решении различных задач физики, электротехники, химии и других наук. Данная глава посвящена численному решению дифференциальных уравнений и систем. Первый раздел напомним читателю общие сведения о дифференциальных уравнениях и системах. Во втором разделе описаны численные методы решения дифференциальных уравнений и систем и их реализация в MATLAB. Третий раздел посвящен описанию встроенных функций MATLAB для решения дифференциальных уравнений и систем. В последующих двух разделах рассматриваются возможности Mathcad и Maple для решения дифференциальных уравнений и систем.

7.1. Общие сведения о дифференциальных уравнениях

Дифференциальным уравнением n -го порядка называется соотношение вида:

$$H(t, x, x', x'', \dots, x^{(n)}) = 0. \quad (7.1)$$

Решением дифференциального уравнения называется функция $x(t)$, которая обращает уравнение в тождество.

известным аналитическим решением очень часто необходимо вычислить числовое значение при определенных исходных данных. Поэтому широкое распространение получили численные методы решения обыкновенных дифференциальных уравнений.

7.2. Численные методы решения дифференциальных уравнений и их реализация в MATLAB

Численные методы решения дифференциального уравнения первого порядка будем рассматривать для следующей задачи Коши. Найти решение дифференциального уравнения

$$x' = f(x, t), \quad (7.4)$$

удовлетворяющее начальному условию

$$x(t_0) = x_0. \quad (7.5)$$

Иными словами, требуется найти интегральную кривую $x = x(t)$, проходящую через заданную точку $M_0(t_0, x_0)$ (рис. 7.1).

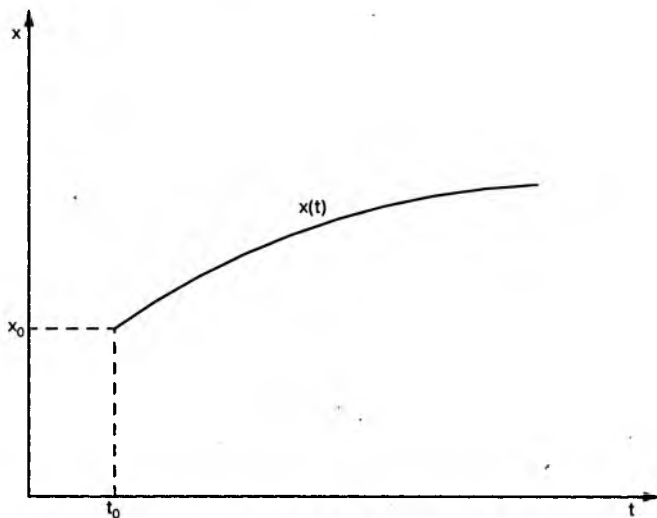


Рис. 7.1 ▽ Интегральная кривая $x(t)$, проходящая через точку (t_0, x_0)

Для дифференциального уравнения n -го порядка

$$x^{(n)} = f(t, x, x', x'', \dots, x^{(n-1)}) \quad (7.6)$$

задача Коши состоит в нахождении решения $x = x(t)$, удовлетворяющего уравнению (7.6) и начальным условиям

$$x(t_0) = x_0, x'(t_0) = x'_0, \dots, x^{(n-1)}(t_0) = x_0^{n-1} \quad (7.7)$$

Рассмотрим основные численные методы решения задачи Коши.

7.2.1. Решение дифференциальных уравнений методом Эйлера

При решении задачи Коши (7.4), (7.5) на интервале $[t_0, t_n]$, выбрав достаточно малый шаг h , построим систему равностоящих точек:

$$h = \frac{t_n - t_0}{n}, t_i = t_0 + i h, i = 0, 1, 2, \dots, n. \quad (7.8)$$

Для вычисления значения функции в точке t_1 разложим функцию $x = x(t)$ в окрестности точки t_0 в ряд Тейлора:

$$x(t_1) = x(t_0 + h) = x(t_0) + h x'(t_0) + x''(t_0) \frac{h^2}{2} + \dots \quad (7.9)$$

При достаточно малом значении h членами выше второго порядка можно пренебречь, и с учетом $x'(t_0) = f(x_0, t_0)$ получим следующую формулу для вычисления приближенного значения функции $x(t)$ в точке t_1 :

$$x(t_1) \approx x_1 = x_0 + h f(x_0, t_0). \quad (7.10)$$

Рассматривая найденную точку (x_1, t_1) как начальное условие задачи Коши, запишем аналогичную формулу для нахождения значения функции $x(t)$ в точке t_2 :

$$x_2 = x_1 + h f(x_1, t_1).$$

Повторяя этот процесс, сформируем последовательность значений x_i в точках t_i по формуле:

$$x_{i+1} = x_i + h f(x_i, t_i), i = 1, 2, \dots, n. \quad (7.11)$$

Процесс нахождения значений функции x_i в узловых точках t_i по формуле (7.11) называется *методом Эйлера*. Геометрическая интерпретация метода Эйлера состоит в замене интегральной кривой $x(t)$ ломаной $M_0, M_1, M_2, \dots, M_n$ с вершинами $M_i(x_i, y_i)$. Звенья ломаной Эйлера $M_i M_{i+1}$ в каждой вершине M_i имеют

направление $y_i = f(t_i; x_i)$, совпадающее с направлением интегральной кривой $x(t)$ уравнения (7.4), проходящей через точку M_i (рис. 7.2). Последовательность ломаных Эйлера при $h \rightarrow 0$ на достаточно малом отрезке $[x_i; x_i + h]$ стремится к искомой интегральной кривой.

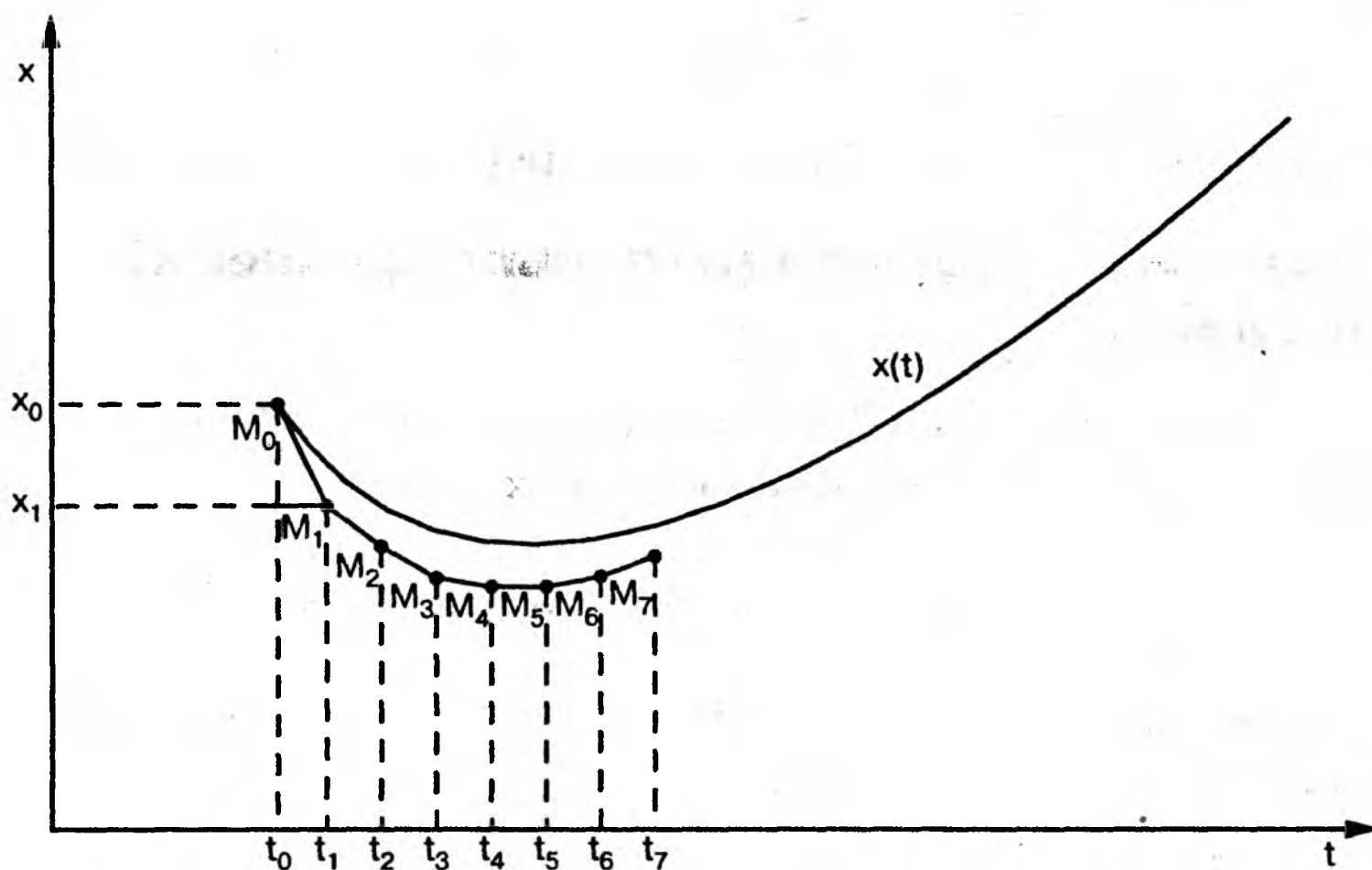


Рис. 7.2 ▼ Геометрическая интерпретация метода Эйлера

На каждом шаге решение $x(t)$ определяется с ошибкой за счет отбрасывания членов ряда Тейлора выше первой степени, что в случае быстроменяющейся функции $f(t, x)$ может привести к быстрому накоплению ошибки. В методе Эйлера следует выбирать достаточно малый шаг h . На рис. 7.3 представлена блок-схема решения задачи Коши (7.4)–(7.5) методом Эйлера на интервале $[a; b]$.

Рассмотрим применение метода Эйлера на примере решения следующей задачи.

ЗАДАЧА 7.1. Решить задачу Коши

$$x' = \frac{2}{t^2} - x^2, \quad x(1) = 0 \quad (7.12)$$

на интервале $[1; 3]$.

В листинге 7.1 представлена функция на М-языке для решения дифференциального уравнения методом Эйлера¹.

¹ В этой и последующих функциях этой главы учтено, что массивы и матрицы в MATLAB индексируются с 1.

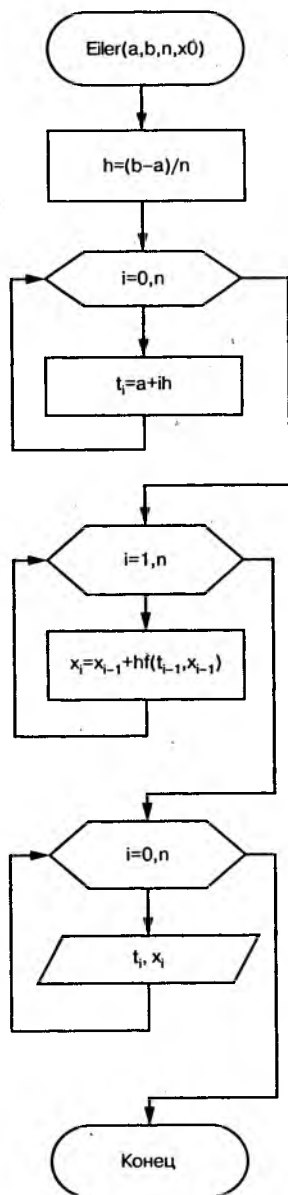


Рис. 7.3 ▼ Блок-схема метода Эйлера

ЛИСТИНГ 7.1

```

function [x,t]=euler(a,b,n,x0)
% Функция решения задачи Коши  $x'(t)=g(t,x)$   $x(a)=x_0$  методом Эйлера на
% интервале интегрирования  $[a,b]$ ,
%  $n$  - количество отрезков, на которые разбивается интервал  $[a,b]$ .
% Вычисление шага  $h$ 
h=(b-a)/n;
x(1)=x0;
% Формирование системы равноотстоящих узлов  $t_i$ 
for i=1:n+1
    t(i)=a+(i-1)*h;
end
% Вычисление значения функции в узловых точках методом Эйлера по формуле
% (7.11)
for i=2:n+1
    x(i)=x(i-1)+h*g(t(i-1),x(i-1));
end
end

```

На рис. 7.4–7.5 представлены графики решения задачи 7.1 методом Эйлера (при $n = 15$ и $n = 40$) и точного решения.

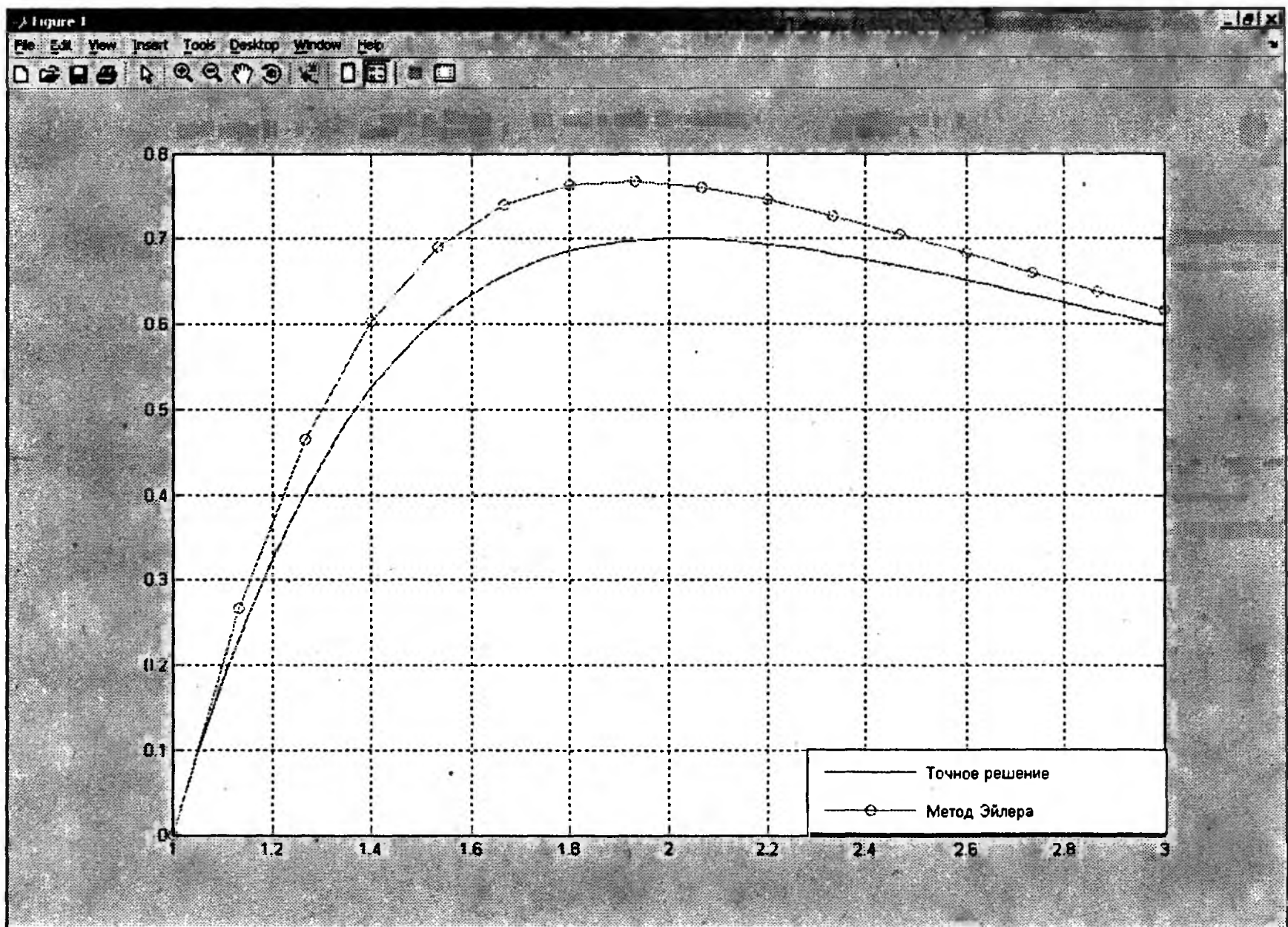


Рис. 7.4 ▼ Графики решения задачи 7.1 методом Эйлера ($n = 15$) и точного решения

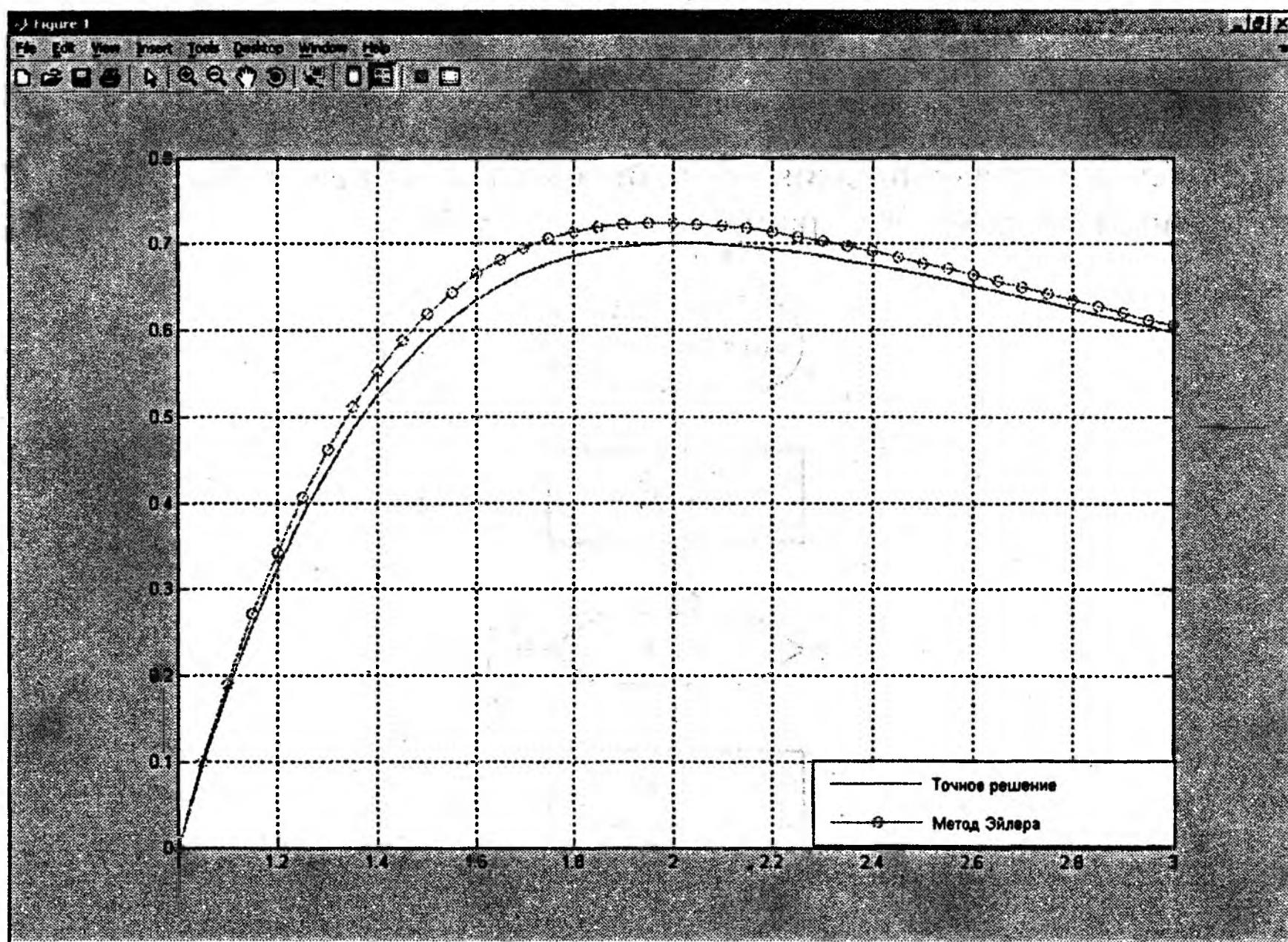


Рис. 7.5 ▼ Графики решения задачи 7.1 методом Эйлера ($n = 40$) и точного решения

Как видно из рис. 7.4–7.5, уменьшение шага интегрирования h приводит к повышению точности решения уравнения методом Эйлера.

Метод Эйлера является очень простым методом решения задачи Коши, но недостаточно точным: для его использования нужно выбирать достаточно маленький шаг интегрирования h . Так, для того, чтобы решение уравнения (7.12) методом Эйлера практически совпадало с точным решением на интервале $[1; 3]$, необходимо уменьшить шаг до 0,004.

В связи с недостаточной точностью метода Эйлера для повышения точности были разработаны его модификации.

7.2.2. Решение дифференциальных уравнений при помощи модифицированного метода Эйлера

Более точным методом решения задачи (7.4)–(7.5) является *модифицированный метод Эйлера*, при котором сначала вычисляют промежуточные значения:

$$t_p = t_i + \frac{h}{2}, x_p = x_i + \frac{h}{2} f(t_i, x_i), \quad (7.13)$$

после чего находят значение x_{i+1} по формуле:

$$x_{i+1} = x_i + hf(x_p, t_p), \quad i = 1, 2, \dots, n. \quad (7.14)$$

На рис. 7.6 представлена блок-схема решения задачи Коши (7.4)–(7.5) модифицированным методом Эйлера на интервале $[a, b]$.

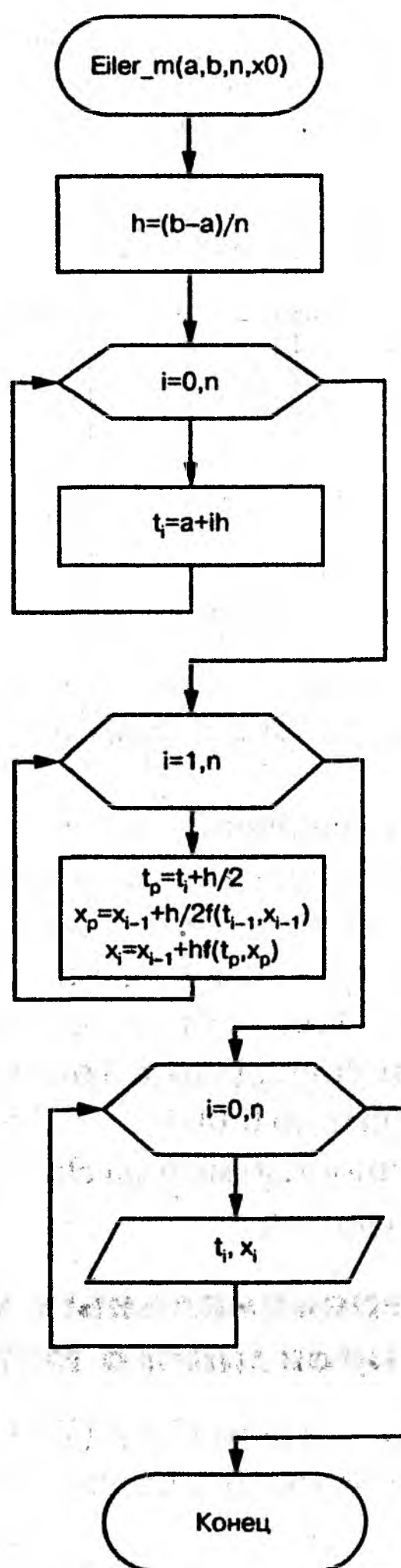


Рис. 7.6 ▼ Блок-схема модифицированного метода Эйлера

В листинге 7.2 приведена функция решения задачи Коши модифицированным методом Эйлера.

Листинг 7.2

```
function [x,t]=euler_m(a,b,n,x0)
% Функция решения задачи Коши  $x'(t)=g(t,x)$   $x(a)=x_0$  модифицированным
% методом Эйлера на интервале интегрирования  $[a,b]$ ,
%  $n$  - количество отрезков, на которые разбивается интервал  $[a,b]$ .
% Вычисление шага  $h$ 
h=(b-a)/n;
x(1)=x0;
% Формирование системы системы равноотстоящих узлов  $t_i$ 
for i=1:n+1
    t(i)=a+(i-1)*h;
end
% Вычисление значение функции в узловых точках модифицированным
% методом Эйлера по формулам (7.13) - (7.14)
for i=2:n+1
    tp=t(i-1)+h/2;
```

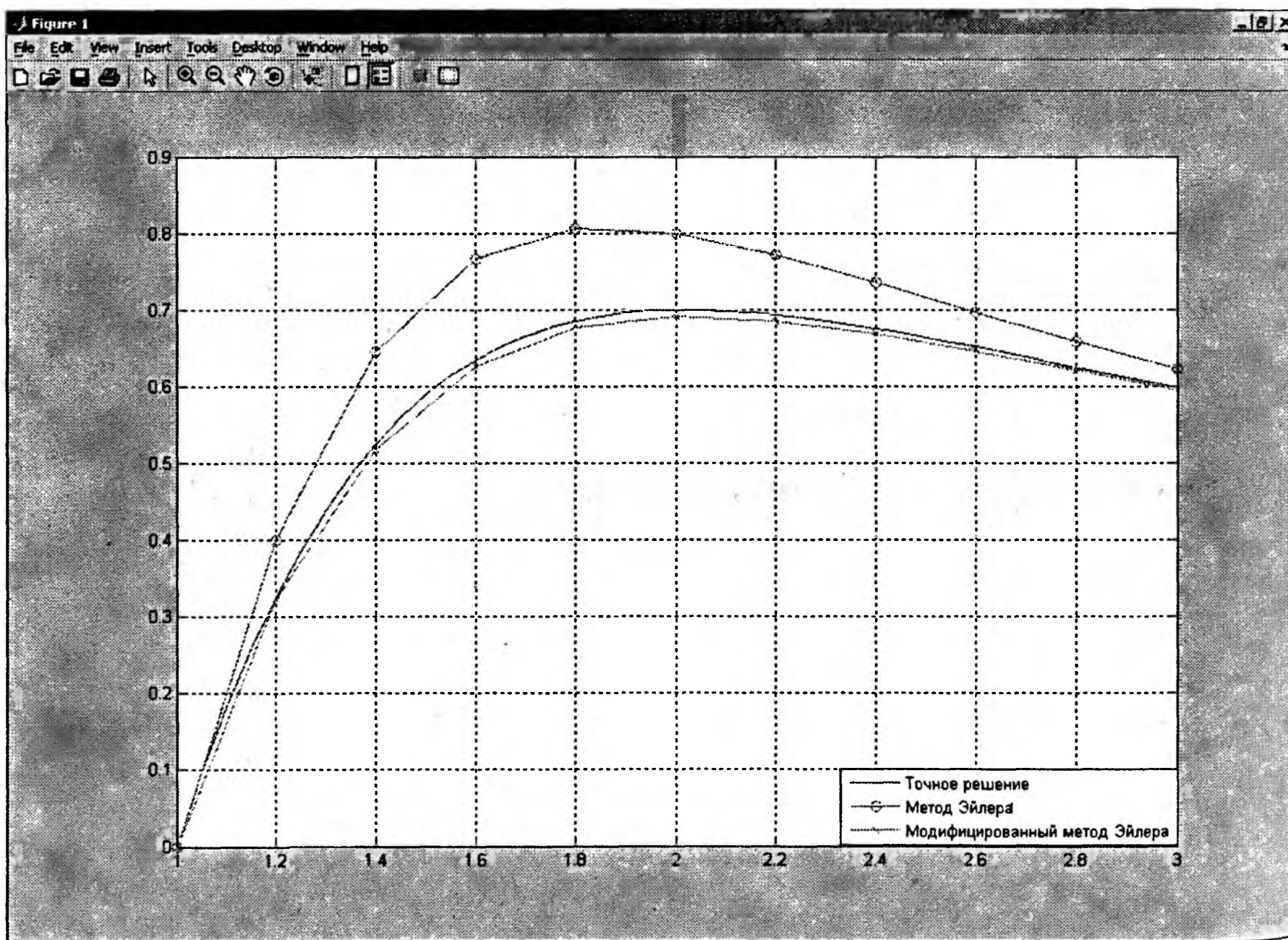


Рис. 7.7 ▼ Графики решения задачи 7.1 методом Эйлера, модифицированным методом Эйлера ($n = 10$) и точного решения

```

xp=x(i-1)+h/2*g(t(i-1),x(i-1));
x(i)=x(i-1)+h*g(tp,xp);
end
end

```

На рис. 7.7 приведены графики решения задачи 7.1 методом Эйлера, модифицированным методом Эйлера ($n = 10$) и точного решения.

Как видно из рис. 7.7, с помощью модифицированного метода Эйлера можно найти намного более точное решение, чем обычным методом Эйлера. Так, при решении задачи 7.1 практическое совпадение с точным решением в модифицированном методе Эйлера достигается при $h = 0,1$, в то время как для достижения такой же точности методом Эйлера приходилось уменьшать шаг до 0,004.

7.2.3. Решение дифференциальных уравнений методами Рунге-Кутта

Рассмотренные выше методы Эйлера (как обычный, так и модифицированный) являются частными случаями явного метода Рунге-Кутта k -го порядка. В общем случае формула вычисления очередного приближения методом Рунге-Кутта имеет вид:

$$x_{i+1} = x_i + h\varphi(x_i, t_i, h), \quad i = 1, 2, \dots, n. \quad (7.15)$$

Функция $\varphi(x, t, h)$ приближает отрезок ряда Тейлора до k -го порядка и не содержит частных производных $f(x, t)$ [9].

Метод Рунге-Кутта первого порядка

Метод Эйлера является методом Рунге-Кутта первого порядка ($k = 1$) и получается при $\varphi(x, t, h) = f(x, t)$.

Методы Рунге-Кутта второго порядка

Семейство методов Рунге-Кутта второго порядка имеет вид [9]:

$$x_{i+1} = x_i + h \left((1 - \alpha) f(t_i, x_i) + \alpha f\left(t_i + \frac{h}{2\alpha}, x_i + \frac{h}{2\alpha} f(t_i, x_i)\right) \right), \quad i = 1, 2, \dots, n. \quad (7.16)$$

Два наиболее известных среди методов Рунге-Кутта второго порядка – это метод Хорна ($\alpha = \frac{1}{2}$) и модифицированный метод Эйлера ($\alpha = 1$).

Подставив $\alpha = \frac{1}{2}$ в формулу (7.16), получаем расчетную формулу метода Хорна:

$$x_{i+1} = x_i + \frac{h}{2} \left(f(t_i, x_i) + f\left(t_i + h, x_i + h f(t_i, x_i)\right) \right), \quad i = 1, 2, \dots, n. \quad (7.17)$$

Блок-схема метода Хорна будет отличаться от рассмотренных ранее блок-схем методов Эйлера (см. рис. 7.3, 7.6) расчетной формулой для вычисления x_{i+1} (см. формулу (7.17)).

Подставив $\alpha = 1$ в формулу (7.16), получаем расчетную формулу уже рассмотренного выше *модифицированного метода Эйлера*:

$$x_{i+1} = x_i + h \left(f \left(t_i + \frac{h}{2}, x_i + \frac{h}{2} f(t_i, x_i) \right) \right), i = 1, 2, \dots, n. \quad (7.18)$$

Метод Рунге-Кутты четвертого порядка

Наиболее известным является *метод Рунге-Кутты четвертого порядка*, расчетные формулы которого можно записать в виде:

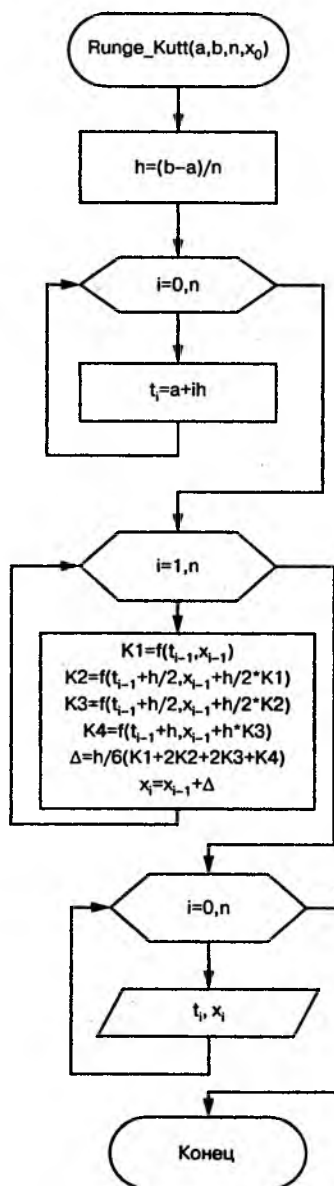
$$\begin{cases} x_{i+1} = x_i + \Delta x_i, i = 1, 2, \dots, n, \\ \Delta x_i = \frac{h}{6} (K_1^i + 2K_2^i + 2K_3^i + K_4^i), \\ K_1^i = f(t_i, x_i), \\ K_2^i = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2} K_1^i\right), \\ K_3^i = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2} K_2^i\right), \\ K_4^i = f(t_i + h, x_i + h K_3^i). \end{cases} \quad (7.19)$$

На рис. 7.8 представлена блок-схема решения задачи Коши методом Рунге-Кутты.

Рассмотрим решение задачи 7.1 методом Рунге-Кутты. В листинге 7.3 представлена функция на М-языке решения дифференциального уравнения методом Рунге-Кутты.

Листинг 7.3

```
function [x,t]=runge_kut(a,b,n,x0)
% Функция решения задачи Коши  $x'(t)=g(t,x)$   $x(a)=x_0$  методом Рунге-Кутты на
% интервал интегрирования  $[a,b]$ ,
% n - количество отрезков, на которые разбивается интервал  $[a,b]$ .
% Вычисление шага h
h=(b-a)/n;
x(1)=x0;
% Формирование системы равноотстоящих узлов  $t_i$ 
for i=1:n+1
    t(i)=a+(i-1)*h;
end
% Вычисление значения функции в узловых точках методом Рунге-Кутта по
```

**Рис. 7.8** ▼ Блок-схема решения задачи Коши методом Рунге–Кутты


```

% формуле (7.19)
for i=2:n+1
% Расчет коэффициентов K1, K2, K3, K4
    K1=g(t(i-1),x(i-1));
    K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
    K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
    K4=g(t(i-1)+h,x(i-1)+h*K3);
%Расчет приращения delt
    delt=h/6*(K1+2*K2+2*K3+K4);
    x(i)=x(i-1)+delt;
end
end

```

На рис. 7.9 приведены графики решения задачи 7.1 методом Эйлера, методом Рунге–Кутты ($n = 10$) и точного решения.

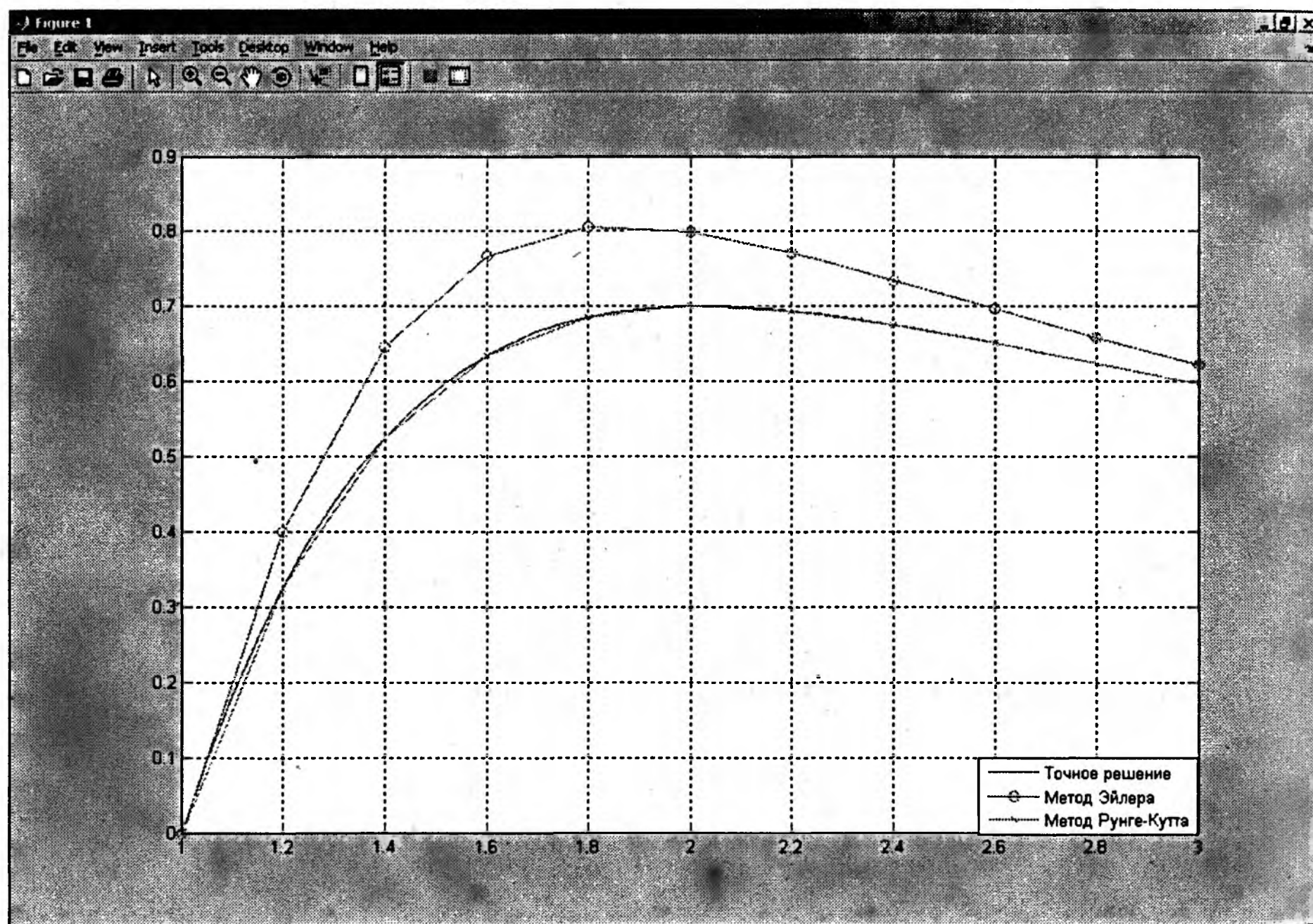


Рис. 7.9 ▼ Графики решения задачи 7.1 методом Эйлера, методом Рунге–Кутты ($n = 10$) и точного решения

Как видно из рис. 7.9, решение задачи 7.1, полученное методом Рунге–Кутты, совпадает в узловых точках с точным решением.

 Одной из главных проблем при использовании рассмотренных ранее численных методов решения дифференциальных уравнений является правильный выбор шага h . Для решения этой проблемы на каждом этапе применяют двойной пересчет. А именно: исходя из текущего верного значения $y(x_i)$, вычисляют величину $y(x_i + 2h)$ двумя способами: один раз с шагом h , а второй раз с двойным шагом $H = 2h$. Если расхождение полученных значений не превышает допустимой погрешности ϵ , то шаг h для данного этапа выбран правильно и полученное с его помощью значение можно принять за $y(x_i + 2h)$. В противном случае шаг уменьшают в два раза.

В рассматриваемом ниже методе Кутта–Мерсона проблема выбора шага интегрирования решена.

Метод Кутта–Мерсона четвертого порядка

Одной из модификаций метода Рунге–Кутта является метод Кутта–Мерсона (или пятиэтапный метод Рунге–Кутта четвертого порядка), который состоит в следующем [9]:

1. На i -м шаге рассчитываются коэффициенты:

$$\begin{aligned} K_1^i &= f(t_i, x_i), \\ K_2^i &= f\left(t_i + \frac{h}{3}, x_i + \frac{h}{3} K_1^i\right), \\ K_3^i &= f\left(t_i + \frac{h}{3}, x_i + \frac{h}{6} K_1^i + \frac{h}{6} K_2^i\right), \\ K_4^i &= f\left(t_i + \frac{h}{2}, x_i + \frac{h}{8} K_1^i + \frac{3h}{12} K_2^i\right), \\ K_5^i &= f\left(t_i + h, x_i + \frac{h}{2} K_1^i - \frac{3h}{2} K_3^i + 2h K_4^i\right). \end{aligned} \quad (7.20)$$

2. Вычисляем приближенное значение $x(t_{i+1})$ по формуле:

$$\tilde{x}_{i+1} = x_i + \frac{h}{2} (K_1^i - 3K_3^i + 4K_4^i). \quad (7.21)$$

3. Вычисляем приближенное значение $x(t_{i+1})$ по формуле:

$$x_{i+1} = x_i + \frac{h}{6} (K_1^i + 4K_3^i + K_5^i). \quad (7.22)$$

4. Вычисляем оценочный коэффициент по формуле:

$$R = 0.2 |x_{i+1} - \tilde{x}_{i+1}| \quad (7.23)$$

5. Сравниваем R с точностью вычислений ε . Если $R \geq \varepsilon$, то уменьшаем шаг вдвое и возвращаемся к п.1. Если $R < \varepsilon$, то значение, вычисленное по формуле (7.22), и будет вычисленным значением $x(t_{i+1})$ (с точностью ε).
6. Перед переходом к вычислению следующего значения x сравниваем R с $\frac{\varepsilon}{64}$. Если $R \leq \frac{\varepsilon}{64}$, то дальнейшие вычисления можно проводить с удвоенным шагом $h = 2h$.

На рис. 7.10 представлена блок-схема решения задачи (7.4)–(7.5) методом Кутта–Мерсона.

В листинге 7.4 представлена функция на М-языке решения дифференциального уравнения методом Кутта–Мерсона с заданной точностью.

Листинг 7.4

```
function [x,t,j]=kut_merson(a,b,n,eps,x0)
% Функция решения задачи Коши  $x'(t)=g(t,x)$   $x(a)=x_0$  методом Кутта–Мерсона
% на интервале интегрирования  $[a,b]$  с точностью eps,
% n – количество отрезков, на которые вначале разбивается интервал  $[a,b]$ .
% Вычисление шага h
h=(b-a)/n;
x(1)=x0;t(1)=a;i=2;
while (t(i-1)+h)<=b
R=3*eps;
while R>eps
% Расчет коэффициентов K1, K2, K3, K4, K5
K1=g(t(i-1),x(i-1));
K2=g(t(i-1)+h/3,x(i-1)+h/3*K1);
K3=g(t(i-1)+h/3,x(i-1)+h/6*K1+h/6*K2);
K4=g(t(i-1)+h/2,x(i-1)+h/8*K1+3*h/8*K2);
K5=g(t(i-1)+h,x(i-1)+h/2*K1-3*h/2*K3+2*h*K4);
% Вычисление сравниваемых значений x(i+1)
X1=x(i-1)+h/2*(K1-3*K3+4*K4);
X2=x(i-1)+h/6*(K1+4*K4+K5);
% Вычисление оценочного коэффициента R
R=0.2*abs(X1-X2);
% Сравнение оценочного коэффициента R с точностью eps
if R>eps
h=h/2;
else
% Если оценочный коэффициент R меньше точности eps,
% то происходит формирование очередной найденной точки и переход
% к следующему этапу по i
t(i)=t(i-1)+h;
x(i)=X2;
i=i+1;
```

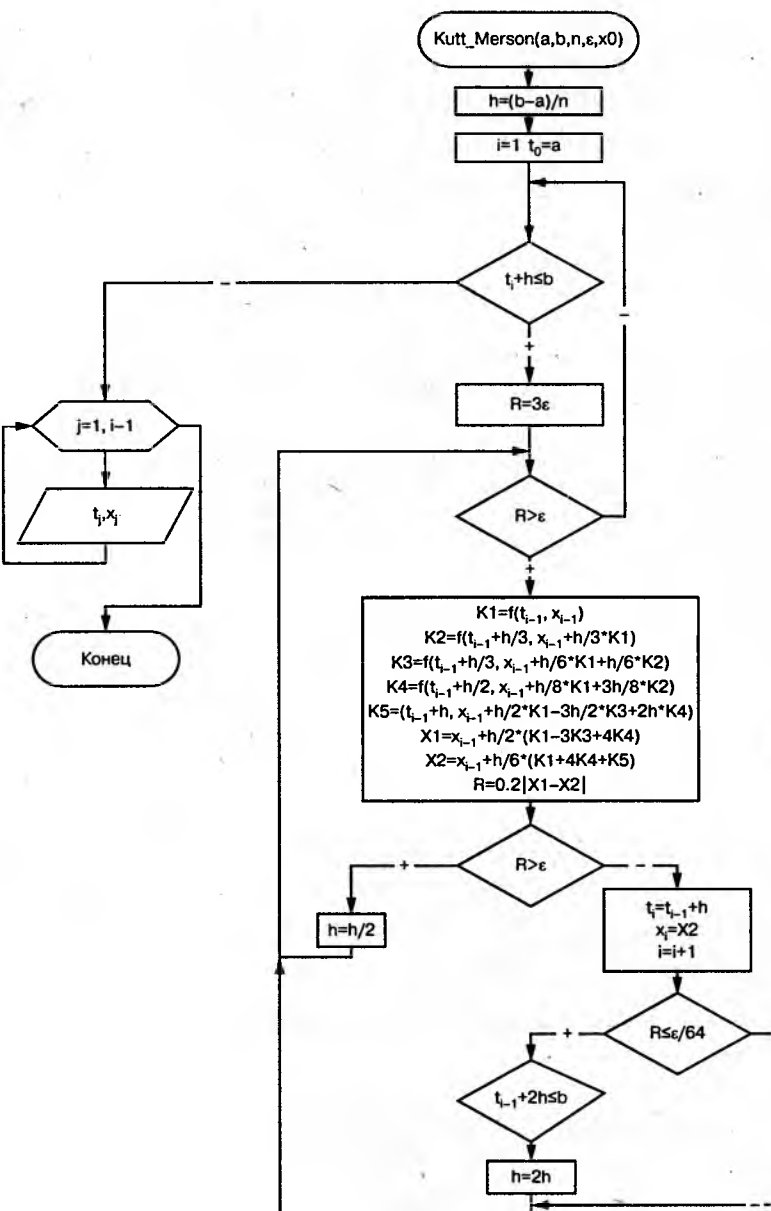



Рис. 7.10 ▼ Блок-схема решения задачи Коши методом Кутты–Мерсона

```

% Если оценочный коэффициент R меньше eps/64,
% то можно попробовать увеличить шаг.
    if R<=eps/64
        if (t(i-1)+2*h)<=b
            h=2*h;
        end
    end
end
end
end
% В переменной j возвращается количество элементов в массивах x и t
j=i-1
end

```

На рис. 7.11 приведены графики решения задачи 7.1 методом Кутта–Мерсона с точностью $\varepsilon = 10^{-6}$ и точного решения. При обращении к функции `kut_merson` в качестве n передавалось число 5. Функция выбирала оптимальный шаг на каждом из отрезков. Как видно из рисунка, шаг то увеличивается, то уменьшается. При решении уравнения методом Кутта–Мерсона невозможно гарантировать вычисление значения в заданных точках интервала $[t_0, t_n]$. Однако после получения решения методом Кутта–Мерсона значение в любой точке интервала $[t_0, t_n]$ можно вычислить, интерполируя полученную зависимость $[t_i, x_i]$.

Рассмотренные методы Рунге–Кутта относятся к классу одношаговых методов, в которых для вычисления значения в очередной точке x_{k+1} нужно знать значение в предыдущей точке x_k .

Еще один класс методов решения задачи Коши – многошаговые методы, в которых используются точки $x_{k-3}, x_{k-2}, x_{k-1}, x_k$ для вычисления x_{k+1} . В многошаговых методах первые четыре начальные точки $(t_0, x_0), (t_1, x_1), (t_2, x_2), (t_3, x_3)$ должны быть получены заранее любым из одношаговых методов (метод Эйлера, Рунге–Кутта и т.д.). Наиболее известными многошаговыми методами являются методы *прогноза-коррекции* Адамса и Милна.

7.2.4. Решение дифференциальных уравнений методом прогноза-коррекции Адамса

Рассмотрим решение уравнения (7.1)–(7.2) на интервале $[t_i; t_{i+1}]$. Будем считать, что решение в точках $t_0, t_1, t_2, \dots, t_i$ уже найдено, и значения в этих точках будем использовать для нахождения значения $x(t_{i+1})$.

Проинтегрируем уравнение (7.1) на интервале $[t_i; t_{i+1}]$ и получим соотношение [9]:

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} f(t, x(t)) dt. \quad (7.24)$$

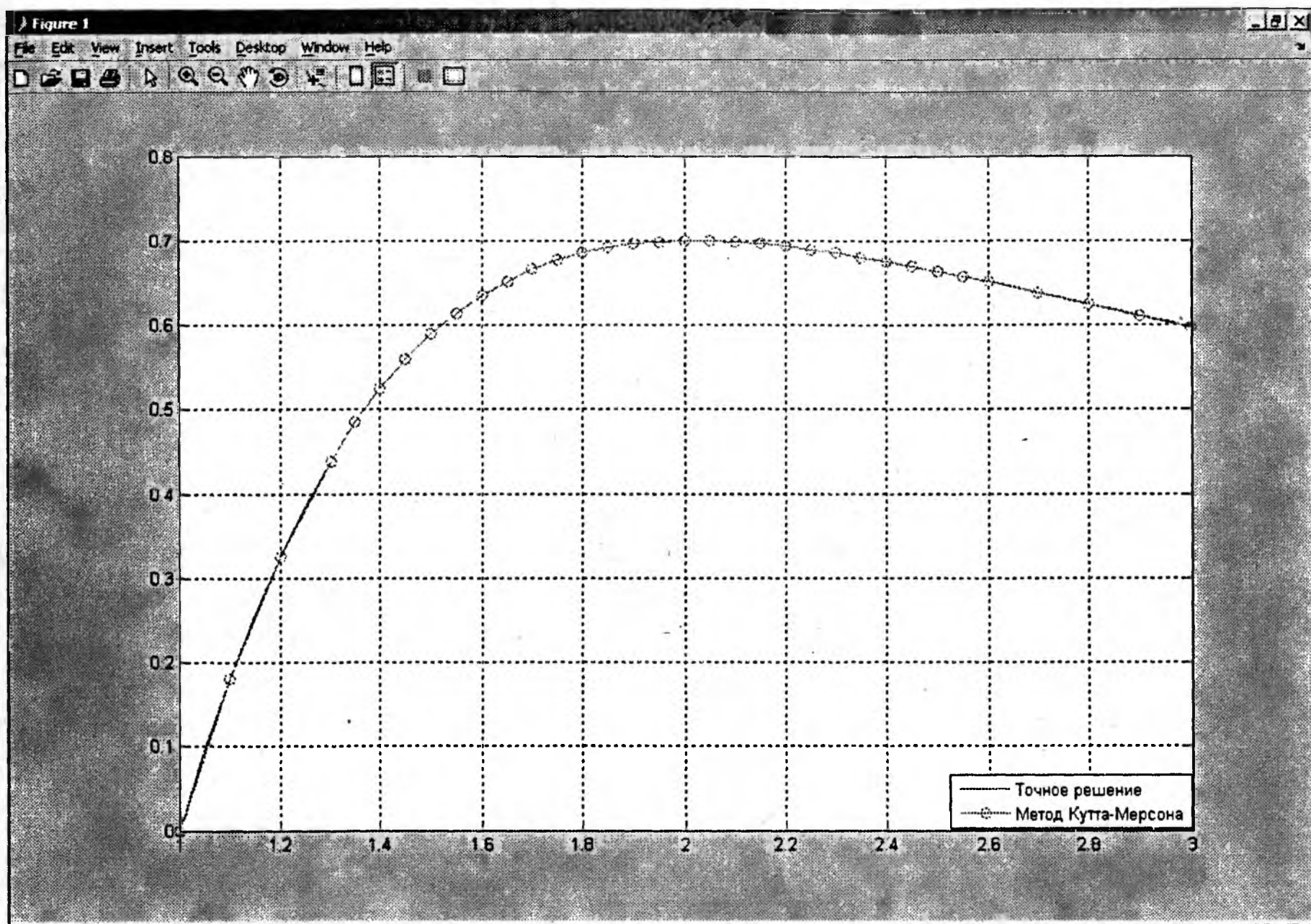


Рис. 7.11 ▽ Графики решения задачи 7.1 методом Кутта-Мерсона с точностью $\varepsilon = 10^{-6}$ и точного решения

При вычислении интеграла, входящего в (7.24), вместо функции $f(t, x(t))$ будем использовать интерполяционный полином Лагранжа, построенный по точкам (t_{i-3}, x_{i-3}) , (t_{i-2}, x_{i-2}) , (t_{i-1}, x_{i-1}) , (t_i, x_i) . Подставив полином Лагранжа в (7.24), получаем первое приближение (прогноз) $\tilde{x}_{i+1}$ для значения функции в точке t_{i+1} :

$$\tilde{x}_{i+1} = x_i + \frac{h}{24} \left(-9f(t_{i-3}, x_{i-3}) + 37f(t_{i-2}, x_{i-2}) - 59f(t_{i-1}, x_{i-1}) + 55f(t_i, x_i) \right). \quad (7.25)$$

Как только $\tilde{x}_{i+1}$ вычислено, его можно использовать. Следующий полином Лагранжа для функции $f(t, x(t))$ построим по точкам (t_{i-2}, x_{i-2}) , (t_{i-1}, x_{i-1}) , (t_i, x_i) и новой точке $(t_{i+1}, \tilde{x}_{i+1})$, после чего подставляем его в (7.24) и получаем второе приближение (корректор):

$$x_{i+1} = x_i + \frac{h}{24} \left(f(t_{i-2}, x_{i-2}) - 5f(t_{i-1}, x_{i-1}) + 19f(t_i, x_i) + 9f(t_{i+1}, \tilde{x}_{i+1}) \right). \quad (7.26)$$

Таким образом, для вычисления значения $x(t_{i+1})$ методом Адамса необходимо последовательно применять формулы (7.25), (7.26), а первые четыре точки можно получить методом Рунге-Кутты.

На рис. 7.12 представлена блок-схема решения задачи (7.4)–(7.5) методом Адамса, а в листинге 7.5 приведена функция на М-языке решения задачи Коши методом Адамса.

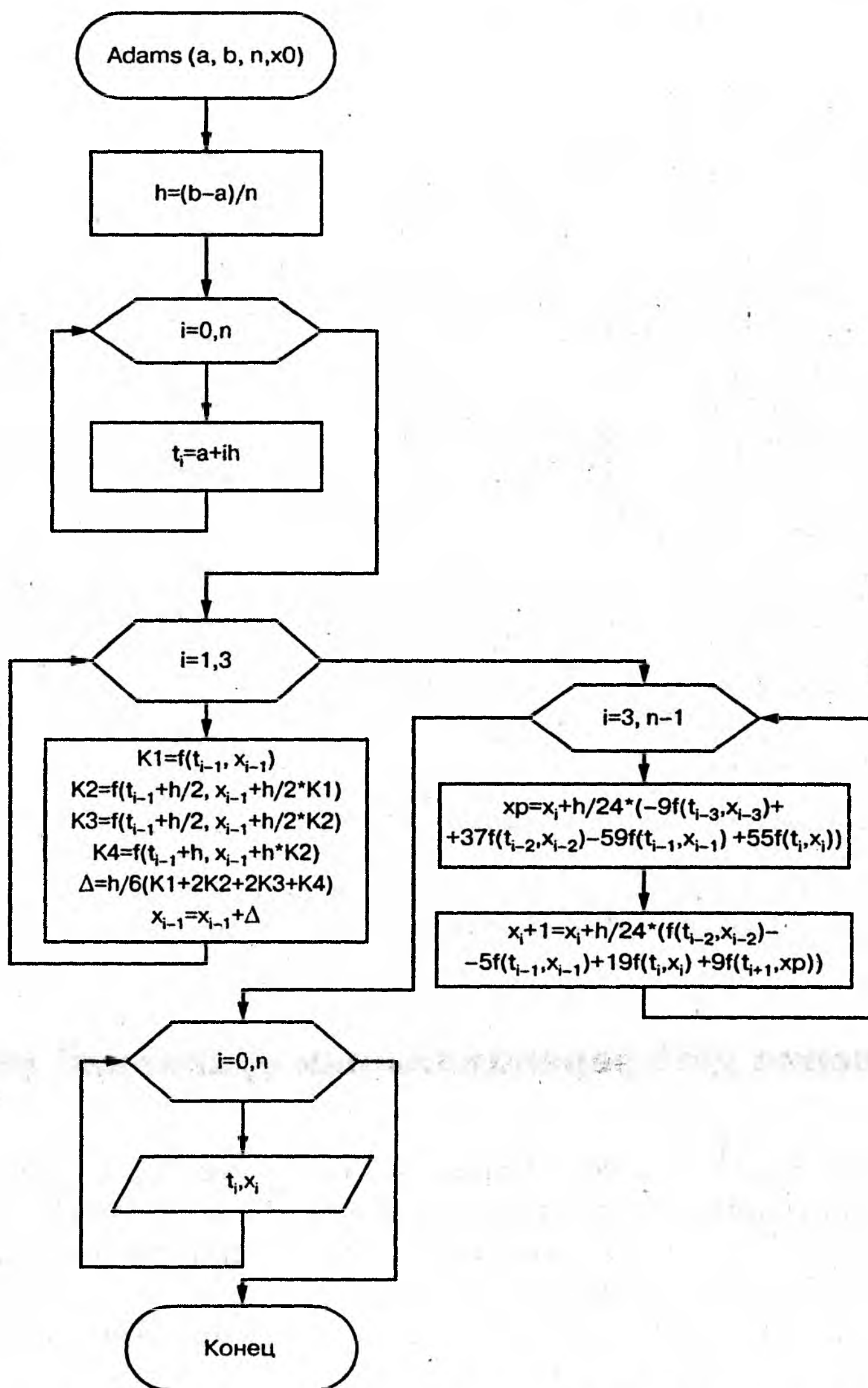


Рис. 7.12 ▼ Блок-схема решения задачи Коши методом Адамса

Листинг 7.5

```

function [x,t]=adams(a,b,n,x0)
% Функция решения задачи Коши  $x'(t)=g(t,x)$   $x(a)=x_0$  методом Адамса на
% интервал интегрирования  $[a,b]$ ,
%  $n$  - количество отрезков, на которые разбивается интервал  $[a,b]$ .
% Вычисление шага  $h$ 
h=(b-a)/n;
x(1)=x0;
% Формирование системы равноотстоящих узлов  $t_i$ 
for i=1:n+1
    t(i)=a+(i-1)*h;
end
% Вычисление значение функции в трех узловых точках методом
% Рунге-Кутты по формуле (7.19)
for i=2:4
    K1=g(t(i-1),x(i-1));
    K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
    K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
    K4=g(t(i-1)+h,x(i-1)+h*K3);
% Расчет приращения  $\Delta t$ 
    delt=h/6*(K1+2*K2+2*K3+K4);
    x(i)=x(i-1)+delt;
end
% Вычисление значение в остальных точках методом Адамса
for i=4:n
    % Вычисление прогноза
    xp=x(i)+h/24*(-9*g(t(i-3),x(i-3))+37*g(t(i-2),x(i-2))-
    -59*g(t(i-1),x(i-1))+55*g(t(i),x(i)));
    % Вычисление корректора
    x(i+1)=x(i)+h/24*(g(t(i-2),x(i-2))-5*g(t(i-1),
    x(i-1))+19*g(t(i),x(i))+9*g(t(i+1),xp));
end
end

```

7.2.5. Решение дифференциальных уравнений методом Милна

Отличие *метода Милна* от метода Адамса состоит в том, что в первом случае в качестве интерполяционного полинома используется полином Ньютона.

Подставив в (7.24) вместо функции $f(t, x(t))$ интерполяционный полином Ньютона, построенный по точкам (t_{k-3}, x_{k-3}) , (t_{k-2}, x_{k-2}) , (t_{k-1}, x_{k-1}) , (t_k, x_k) получаем первое приближение – прогноз Милна $\tilde{x}_{k+1}$ для значения функции в точке t_{k+1} :

$$\tilde{x}_{k+1} = x_{k-3} + \frac{4h}{3} \left(2f(t_{k-2}, x_{k-2}) - f(t_{k-1}, x_{k-1}) + 2f(t_k, x_k) \right). \quad (7.27)$$

Следующий полином Ньютона для функции $f(t, x(t))$ построим по точкам (t_{k-2}, x_{k-2}) , (t_{k-1}, x_{k-1}) , (t_k, x_k) и новой точке $(t_{k+1}, \tilde{x}_{k+1})$, после чего подставляем его в (7.24) и получаем второе приближение – корректор Милна:

$$x_{k+1} = x_{k-1} + \frac{h}{3} \left(f(t_{k-1}, x_{k-1}) + 4f(t_k, x_k) + f(t_{k+1}, \tilde{x}_{k+1}) \right). \quad (7.28)$$

В методе Милна для вычисления значения $x(t_{k+1})$ необходимо последовательно применять формулы (7.27)–(7.28), а первые четыре точки можно получить методом Рунге–Кутты.

Существует *модифицированный метод Милна*. В нем сначала вычисляется первое приближение по формуле (7.27), затем вычисляется управляющий параметр:

$$m_{k+1} = \tilde{x}_{k+1} + \frac{28}{29} (x_k - \tilde{x}_k), \quad (7.29)$$

после чего вычисляется значение второго приближения – корректор Милна – по формуле:

$$x_{k+1} = x_{k-1} + \frac{h}{3} \left(f(t_{k-1}, x_{k-1}) + 4f(t_k, x_k) + f(t_{k+1}, m_{k+1}) \right). \quad (7.30)$$

В модифицированном методе Милна первые четыре точки можно получить методом Рунге–Кутты, а для вычисления значения $x(t_{k+1})$ необходимо последовательно применять формулы (7.27), (7.29), (7.30).

На рис. 7.13 представлена блок-схема решения задачи (7.4)–(7.5) модифицированным методом Милна, а в листинге 7.6 приведена функция на М-языке решения задачи Коши модифицированным методом Милна.

ЛИСТИНГ 7.6

```
function [x,t]=miln(a,b,n,x0)
% Функция решения задачи Коши  $x'(t)=g(t,x)$   $x(a)=x_0$  методом Милна на
% интервал интегрирования  $[a,b]$ ,
% n – количество отрезков, на которые разбивается интервал  $[a,b]$ .
% Вычисление шага h
h=(b-a)/n;
x(1)=x0;xp(1)=x(1);
% Формирование системы системы равноотстоящих узлов ti
for i=1:n+1
    t(i)=a+(i-1)*h;
end
% Вычисление значения функции в трех узловых точках методом
% Рунге-Кутты по формуле(7.19)
for i=2:4
```

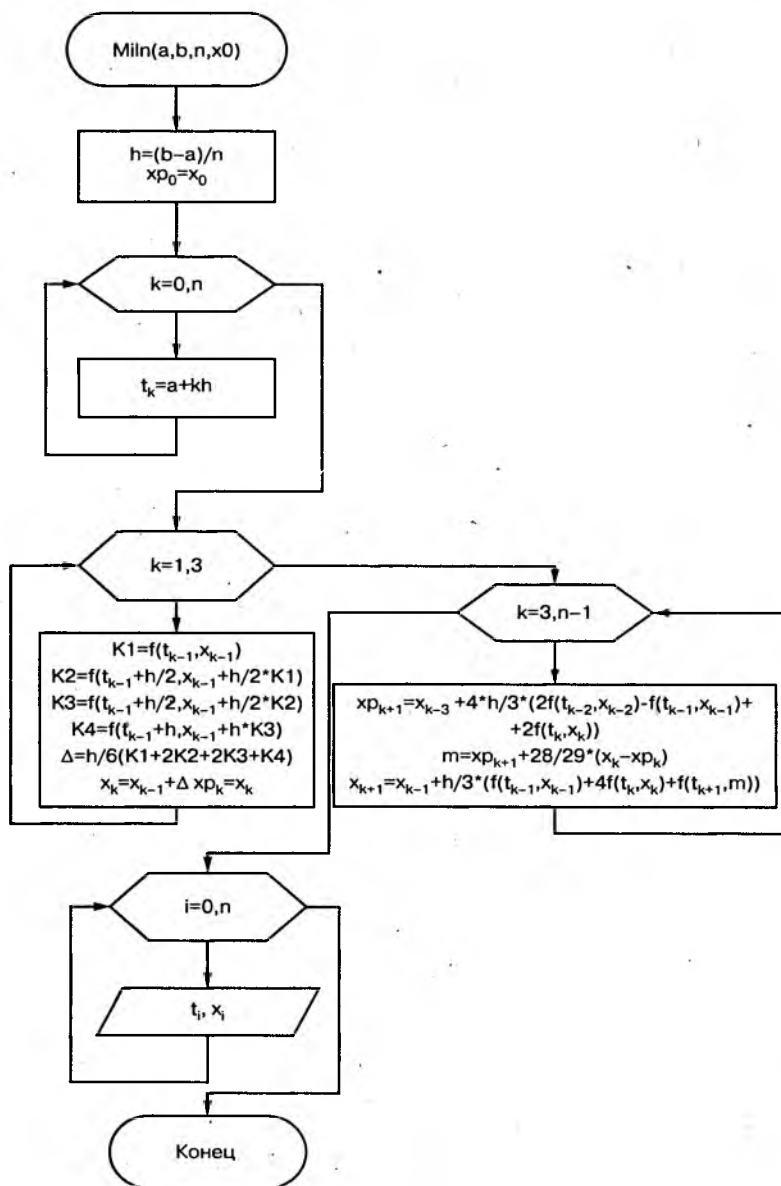


Рис. 7.13 ▼ Блок-схема решения задачи Коши модифицированным методом Милна

```

    K1=g(t(i-1),x(i-1));
    K2=g(t(i-1)+h/2,x(i-1)+h/2*K1);
    K3=g(t(i-1)+h/2,x(i-1)+h/2*K2);
    K4=g(t(i-1)+h,x(i-1)+h*K3);
%Расчет приращения delt
    delt=h/6*(K1+2*K2+2*K3+K4);
    x(i)=x(i-1)+delt;
    xp(i)=x(i);
end
% Вычисление значения в остальных точках методом Адамса
for i=4:n
% Вычисление прогноза
    xp(i+1)=x(i-3)+4*h/3*(2*g(t(i-2),x(i-2))-g(t(i-1),
    x(i-1))+g(t(i),x(i)));
%Вычисление управляющего параметра
    m=xp(i+1)+28/29*(x(i)-xp(i));
% Вычисление корректора
    x(i+1)=x(i-1)+h/3*(g(t(i-1),x(i-1))+4*g(t(i),x(i))+g(t(i+1),m));
end
end

```

В листинге 7.7 представлено обращение к функциям решения задачи Коши (7.12) на интервале $[1; 6]$ методами Кутта–Мерсона (с точностью 0,001), Адамса, модифицированным методом Милна. Графики найденных решений изображены на рис. 7.14.

Листинг 7.7

```

a=1;
b=6;
x0=0;
N=10;
[XA,TA]=adams(a,b,N,x0);
[XM,TM]=miln(a,b,N,x0);
[XRK,TRK,iu]=kut_merson(a,b,N,0.001,x0);
tt=1:0.05:3;
XT=tocn_resn(tt);
%plot(tt,XT,TA,XA,'o-',TM,XM,'x-',TRK,XRK,'*-');
plot(tt,XT,TA,XA,'o-',TM,XM,'x-',TRK,XRK,'*-');
legend('Точное решение','Метод Адамса','Модифицированный метод Милна','Метод
Кутта-Мерсона');
grid on;

```

Из вышеизложенного можно сделать вывод, что наиболее точными методами являются метод Кутта–Мерсона, метод Адамса и модифицированный метод Милна.

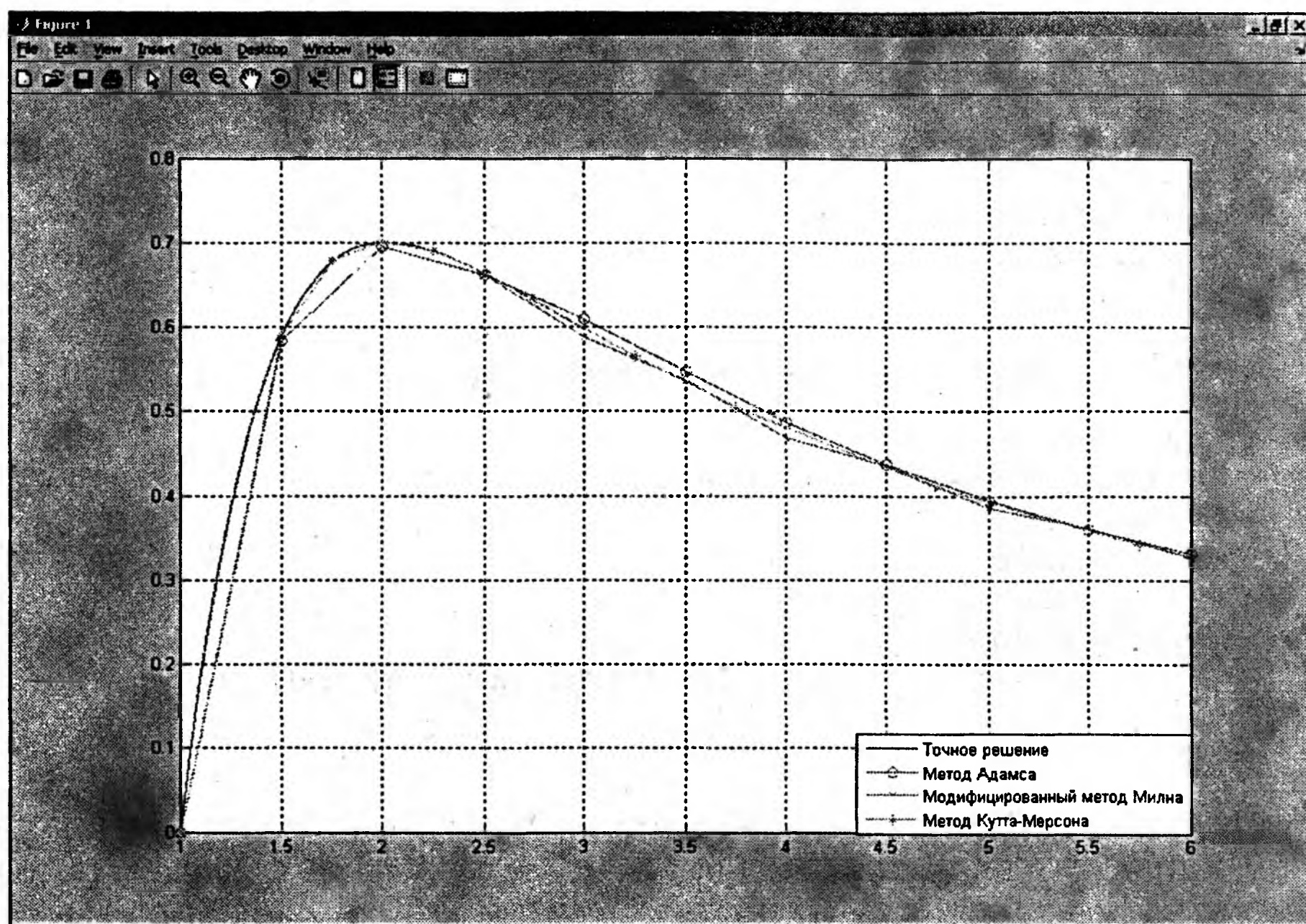


Рис. 7.14 ▽ Графики решения задачи Коши (7.12) на интервале [1; 6] методами Кутты-Мерсона (с точностью 0,001), Адамса, модифицированным методом Милна

7.2.6. Решение систем дифференциальных уравнений

Все описанные методы решения дифференциальных уравнений применимы и для систем дифференциальных уравнений. Рассмотрим на примере метода Рунге-Кутты, как эти методы можно обобщить для систем.

Пусть дана система дифференциальных уравнений в матричном виде:

$$\frac{dx}{dt} = f(t, x) \quad (7.31)$$

с начальным условием $x = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \dots \\ x_n(t) \end{pmatrix}$, $f(t, x) = \begin{pmatrix} f_1(t, x_1, x_2, \dots, x_n) \\ f_2(t, x_1, x_2, \dots, x_n) \\ \dots \\ f_n(t, x_1, x_2, \dots, x_n) \end{pmatrix}$, $x^0 = \begin{pmatrix} x_1^0 \\ x_2^0 \\ \dots \\ x_n^0 \end{pmatrix}$. Задав-

шись некоторым шагом h и введя стандартные обозначения $t_i = t_0 + th$, $x_i = x(t_i)$,

$\Delta x_i = x_{i+1} - x_i$, $i = 1, 2, \dots$, получим формулы для метода Рунге–Кутта для системы:

$$\begin{cases} x_{i+1} = x_i + \Delta x_i, \\ \Delta x_i = \frac{h}{6} \left(K_1^{(i)} + 2 K_2^{(i)} + 2 K_3^{(i)} + K_4^{(i)} \right), \\ K_1^{(i)} = f(t_i, x_i), \\ K_2^{(i)} = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2} K_1^{(i)}\right), \\ K_3^{(i)} = f\left(t_i + \frac{h}{2}, x_i + \frac{h}{2} K_3^{(i)}\right), \\ K_4^{(i)} = f(t_i + h, x_i + h K_3^{(i)}). \end{cases} \quad (7.32)$$

7.3. Встроенные функции MATLAB для решения дифференциальных уравнений

Для решения дифференциальных уравнений и систем в MATLAB предусмотрены следующие функции: `ode45(f, interval, X0, options)`, `ode23(f, interval, X0, options)`, `ode113(f, interval, X0, options)`, `ode15s(f, interval, X0, options)`, `ode23s(f, interval, X0, options)`, `ode23t(f, interval, X0, options)` и `ode23tb(f, interval, X0, options)`.

Входными параметрами этих функций являются:

- f – вектор-функция для вычисления правой части уравнения (7.4) или системы уравнений (7.31);
- `interval` – массив из двух чисел, определяющий интервал интегрирования дифференциального уравнения или системы;
- X_0 – вектор начальных условий системы дифференциальных систем;
- `options` – параметры управления ходом решения дифференциального уравнения или системы.

Все функции возвращают:

- массив T – координат узлов сетки, в которых ищется решение;
- матрицу X , i -й столбец которой является значением вектор-функции решения в узле T_i .

В функции `ode45` реализован метод Рунге–Кутта четвертого-пятого порядка точности, в функции `ode23` также реализован метод Рунге–Кутта, но второго-третьего порядка, функция `ode113` реализует метод Адамса.

Для решения жестких систем предназначены функция `ode15s`, в которой реализован метод Гира, и функция `ode23s`, реализующая метод Розенброка. Для получения более точного решения жесткой системы лучше использовать функцию `ode15s`.

Напомним читателю определение жесткой системы дифференциальных уравнений. Система дифференциальных уравнений n -го порядка

$$\frac{dx}{dt} = Bx \quad (7.33)$$

называется *жесткой* [9], если выполнены следующие условия:

➤ действительные части всех собственных чисел матрицы B отрицательны:

$$\left(\operatorname{Re}(\lambda_k) < 0, k = 0, 1, \dots, n-1 \right);$$

➤ величина $s = \frac{\max_{0 \leq k \leq n-1} |\operatorname{Re}(\lambda_k)|}{\min_{0 \leq k \leq n-1} |\operatorname{Re}(\lambda_k)|}$, называемая *числом жесткости системы*, велика.

При исследовании на жесткость нелинейной системы дифференциальных уравнений (7.31) в роли матрицы B будет выступать матрица частных производных $\frac{\partial F_i}{\partial x_j}$.

Для решения систем с небольшим числом жесткости можно использовать функцию `ode23t`, а для грубой оценки подобных систем служит функция `ode23tb`. Рассмотрим использование функций на примере конкретной задачи.

ЗАДАЧА 7.2. Решить следующую краевую задачу:

$$\begin{aligned} \frac{d^2 x}{dt^2} + 4 \frac{dx}{dt} + 13x &= e^{\sin(t)}, \\ x(0,25) &= -1, \quad x'(0,25) = 1 \end{aligned} \quad (7.34)$$

на интервале $[0,25; 2]$.

Для того чтобы можно было воспользоваться функциями MATLAB, преобразуем уравнение в систему. Сделав замену $y = \frac{dx}{dt}$, получим систему уравнений:

$$\begin{cases} \frac{dy}{dt} = -4y - 13x + e^{\sin(t)}, \\ \frac{dx}{dt} = y. \end{cases} \quad (7.35)$$

с начальными условиями

$$\begin{cases} y(0,25) = 1, \\ x(0,25) = -1. \end{cases} \quad (7.36)$$

Составим функцию вычисления системы (7.35) (см. листинг 7.8). В листинге 7.9 представлено решение уравнения (7.35) с помощью функции `ode45`. График решения приведен на рис. 7.15.

Листинг 7.8

```
function F=FF(t,x)
F=[-4*x(1)-13*x(2)+exp(t);x(1)];
end
```

Листинг 7.9¹

```
% Формируем вектор начальных условий
X0=[1;-1];
% Формируем массив из двух чисел, определяющий интервал интегрирования
interval=[0.25 2];
%Обращение к функции ode45 для решения системы дифференциальных уравнений
[T,X]=ode45(@FF,interval,X0);
% Вывод графика решения
plot(T,X(:,1),'-',T,X(:,2),'-');
legend('y','x-Решение');
grid on;
```

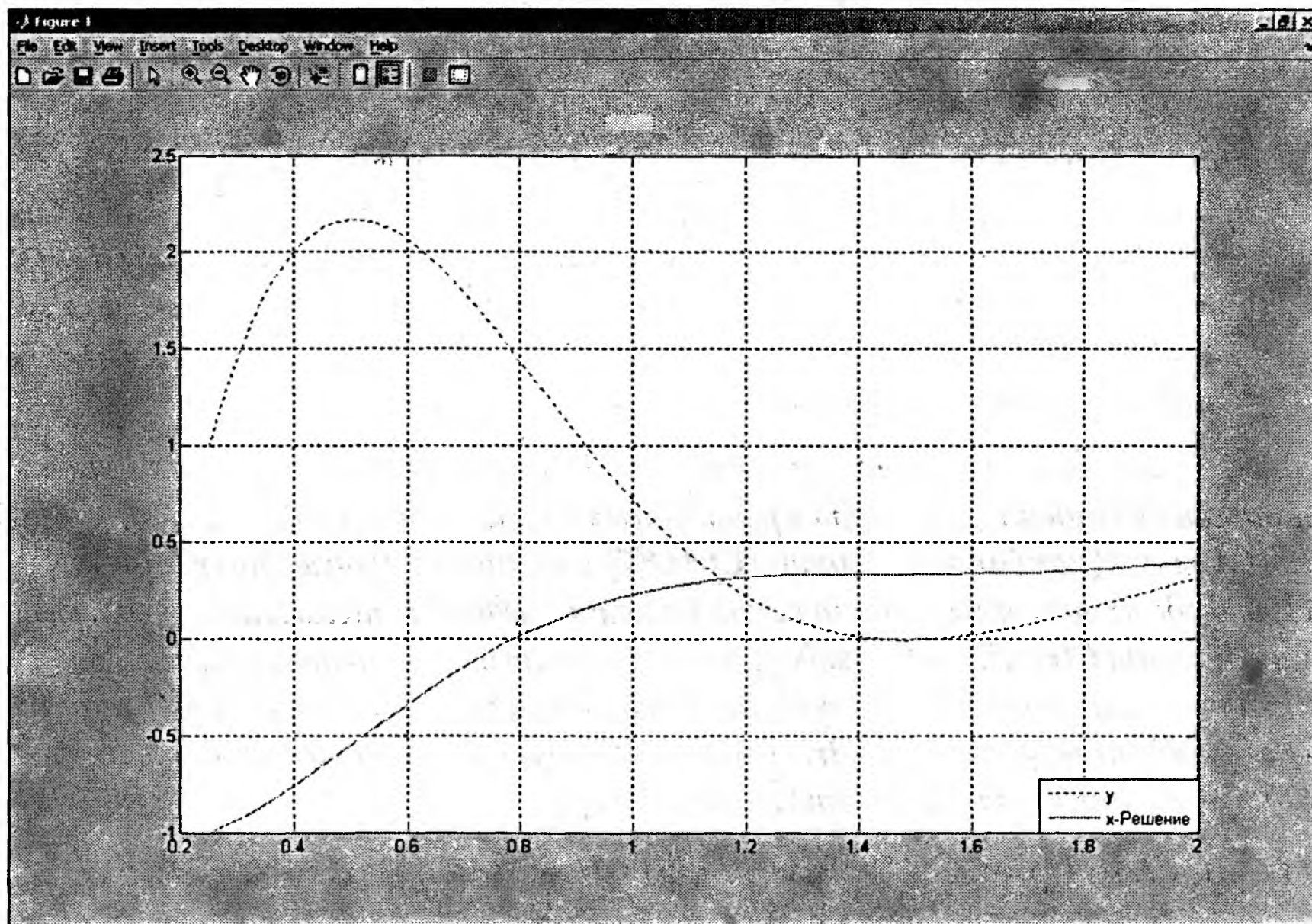


Рис. 7.15 ▼ График решения системы (7.35)

¹ Решение системы уравнений (7.35) на интервале [0.25; 2].

Аналогичным образом можно обращаться и к другим функциям, предназначенным для решения дифференциальных уравнений и систем. Для более детального изучения функций MATLAB, применяемых для решения дифференциальных уравнений, авторы рекомендуют обратиться к справочной системе пакета или к литературе [4].

7.4. Решение дифференциальных уравнений в Mathcad

Рассмотрим особенности решения дифференциальных уравнений с помощью встроенных функций Mathcad.

7.4.1. Встроенные функции Mathcad для решения обыкновенных уравнений и систем

Для решения обыкновенных дифференциальных уравнений и систем в Mathcad предназначены следующие функции:

- `odesolve ([vector], t, b, [n])` возвращает функцию (скалярную для уравнения и вектор-функцию для системы уравнений), являющуюся решением дифференциального уравнения, `vector` – необязательный параметр (используется только при решении систем), содержащий имена функций, `t` – имя независимой переменной, `b` – конечная точка интервала интегрирования, `n` – количество шагов, на которые разбивается интервал интегрирования дифференциального уравнения или системы; эта функция завершает решающий блок `Given`.

 Между служебным словом `Given` и функцией `odesolve` следует вводить уравнения и начальные условия. Для ввода производных в уравнениях следует использовать символ кавычки (комбинация клавиш `Ctrl+F7`) или кнопку производных панели *Calculus*. При вводе производных в начальных условиях следует использовать только комбинацию клавиш `Ctrl+F7`. При вводе уравнений после имени функции обязательно в скобках должен быть указан аргумент. Ввод знака равенства осуществляется с помощью комбинации клавиш `Ctrl+=` или с помощью оператора `=`, расположенного на панели инструментов *Boolean* (Логический).

На рис. 7.16 приведен пример использования оператора `odesolve` для решения следующей задачи Коши:

$$\begin{aligned} x''' + x' + x &= e^{-t} + \cos(t), \\ x(0) &= 1, \quad x'(0) = 0, \quad x''(0) = 0,4; \end{aligned}$$

Пример решения дифференциального уравнения третьего порядка

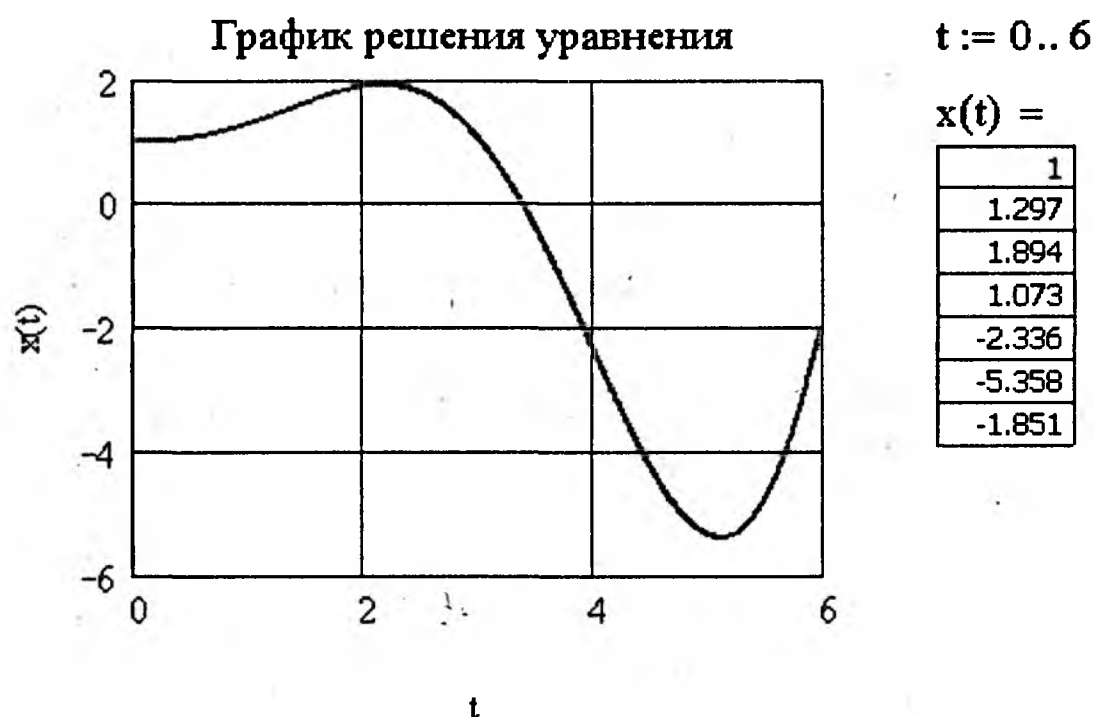
Given Дифференциальное уравнение

$$\frac{d^3}{dt^3}x(t) + x'(t) + x(t) = e^{-t} + \cos(t)$$

Начальные условия

$$x(0) = 1 \quad x''(0) = 0.4 \quad x'(0) = 0$$

Решение уравнения с помощью функции `odesolve`

$$x := \text{Odesolve}(t, 6)$$
Рис. 7.16 ▼ Пример использования функции `odesolve`

► `rkfixed`(init, t1, t2, npoints, D) решает дифференциальное уравнение первого порядка или систему n таких уравнений¹ методом Рунге–Кутты четвертого порядка с фиксированным шагом; `init` – вектор начальных условий, `[t1, t2]` – интервал интегрирования, `npoints` – количество промежутков, на которые разбивается интервал `[t1, t2]`, `D` – вектор правых частей системы дифференциальных уравнений; функция `rkfixed` возвращает матрицу размером $(\text{npoints} + 1) \cdot (n + 1)$, нулевой столбец которой содержит все точки интервала интегрирования `[t1, t2]`, а все остальные содержат значения найденных функций в узловых точках. На рис. 7.17 приведен пример использования функции `rkfixed` при решении задачи Коши:

$$\begin{aligned} x' + x + x^3 &= \cos(t), \\ x(0) &= 2; \end{aligned}$$

$$D(t,x) := -x - x^3 + \cos(t)$$

$$X := \text{rkfixed}(2, 0, 5, 50, D)$$

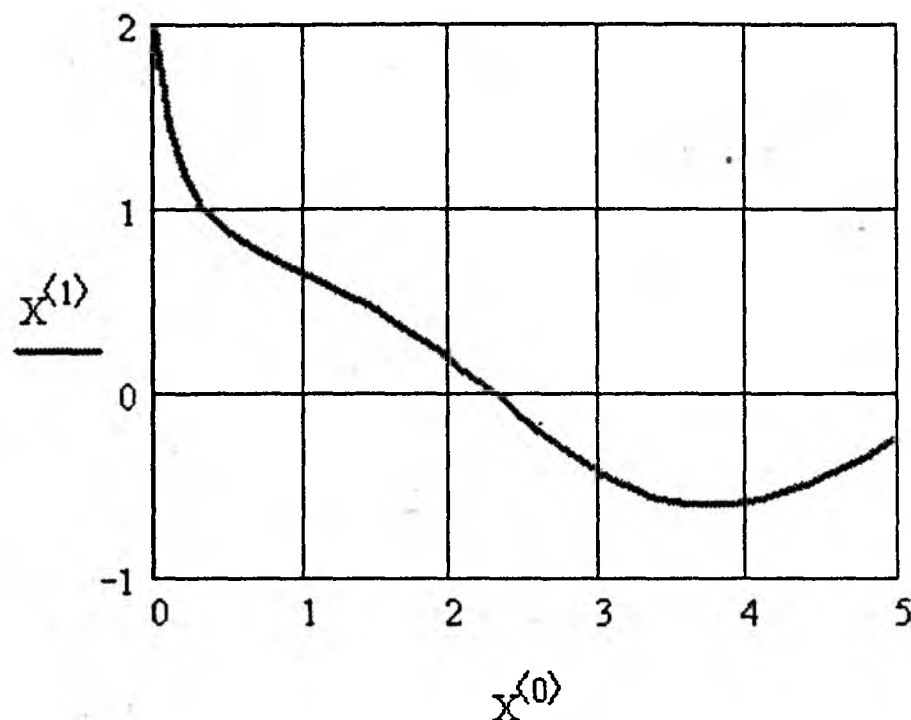


Рис. 7.17 ▼ Пример использования функции rkfixed

- Rkadapt(init, t1, t2, npoints, D) решает дифференциальное уравнение первого порядка или систему n таких уравнений методом Рунге–Кутты

Правая часть дифференциального уравнения

$$D(t,x) := -x^3 - x^2 + \cos(5t)$$

Обращение к функции Rkadapt

$$X := \text{Rkadapt}(3, 0, 5, 50, D)$$

Матрица
решений

X =

	0	1
0	0	3
1	0.1	1.612
2	0.2	1.217
3	0.3	0.993
4	0.4	0.819
5	0.5	0.661
6	0.6	0.515
7	0.7	0.387
8	0.8	0.29
9	0.9	0.238
10	1	0.235
11	1.1	0.277
12	1.2	0.35
13	1.3	0.427
14	1.4	0.485
15	1.5	0.504

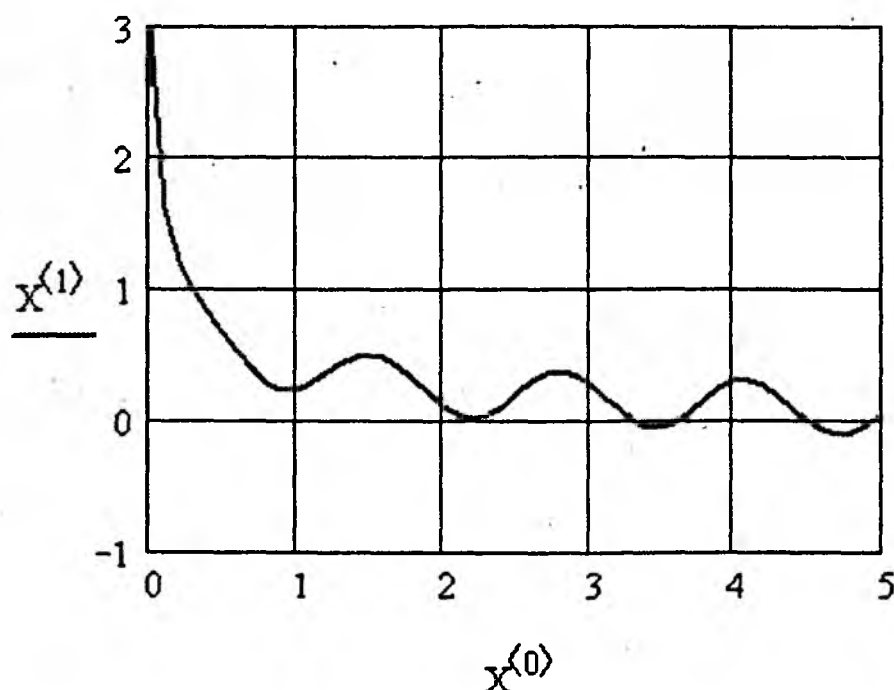


Рис. 7.18 ▼ Пример использования функции Rkadapt

четвертого порядка с автоматическим выбором шага; init – вектор начальных условий, $[t1, t2]$ – интервал интегрирования; $\text{npoints} + 1$ – количество точек, в которых будет вычислено решение уравнения, D – вектор правых частей системы дифференциальных уравнений; функция Rkadapt возвращает матрицу размером $(\text{npoints} + 1) \cdot (n + 1)$, нулевой столбец которой содержит все точки интервала интегрирования $[t1, t2]$, а все остальные содержат значения найденных функций в узловых точках. На рис. 7.18 приведен пример использования функции Rkadapt при решении задачи Коши:

$$\begin{aligned} x' + x^2 + x^3 &= \cos(5t), \\ x(0) &= 3; \end{aligned}$$

- $\text{rkadapt}(\text{init}, t1, t2, \text{eps}, D, k_{\max}, ht)$ решает дифференциальное уравнение или систему методом Рунге–Кутты с автоматическим выбором шага в одной точке; здесь eps – точность вычисления решения при автоматическом выборе шага, $k_{\max}$ – число строк в результирующей матрице (для вычисления в одной точке $k_{\max}$ можно принять равным 2), ht – минимальное значение шага интегрирования; результатом этой функции является матрица размером $k_{\max} \cdot (n+1)$ элементов. На рис. 7.19 приведен пример вычисления значения функции в конечной точке задачи Коши:

$$\begin{aligned} x' + x^2 + x^3 &= \cos(5t), \\ x(0) &= 3; \end{aligned}$$

$$\text{rkadapt}(3, 0, 5, 0.001, D, 2, 0.0001) = \begin{pmatrix} 0 & 3 \\ 5 & 0.057 \end{pmatrix}.$$

Рис. 7.19 ▼ Пример использования функции rkadapt

- $\text{Bulstoer}(\text{init}, t1, t2, \text{npoints}, D)$ решает дифференциальное уравнение или систему методом Булирша–Штера¹, пример использования приведен на рис. 7.20;
- $\text{bulstoer}(\text{init}, t1, t2, \text{eps}, D, k_{\max}, ht)$ решает дифференциальное уравнение или систему методом Булирша–Штера в одной точке;
- функции $\text{Radau}(\text{init}, t1, t2, \text{npoints}, D)$, $\text{Stiffb}(\text{init}, x1, x2, \text{npoints}, D, J)$, $\text{Stiffc}(\text{init}, x1, x2, \text{npoints}, D, J)$ предназначены для решения жестких систем обыкновенных дифференциальных уравнений (работе с этими функциями будет посвящен отдельный раздел).

¹Метод Булирша–Штера имеет смысл использовать в случае гладких правых частей уравнения.

Правая часть дифференциального уравнения

$$D(t, x) := \sqrt{x} + x - e^{\sin(t)}$$

Обращение к функции Bulstoer

$$X := \text{Bulstoer}(3, 0, 1, 20, D)$$

Матрица
решений

$X =$

	0	1
0	0	3
1	0.05	3.191
2	0.1	3.393
3	0.15	3.605
4	0.2	3.827
5	0.25	4.061
6	0.3	4.307
7	0.35	4.565
8	0.4	4.836
9	0.45	5.121
10	0.5	5.421
11	0.55	5.735
12	0.6	6.065
13	0.65	6.412
14	0.7	6.777
15	0.75	7.16

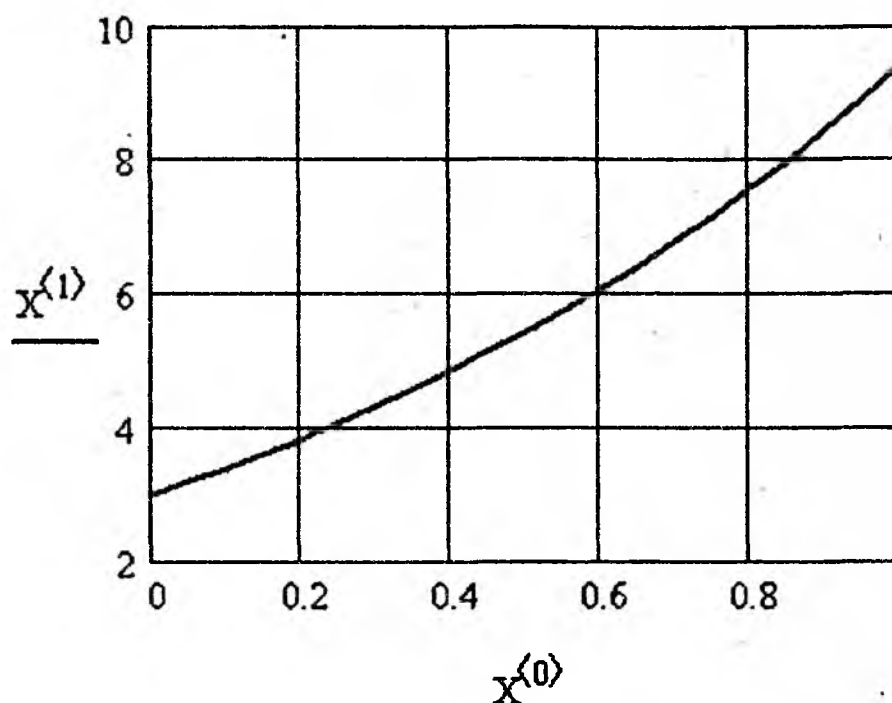


Рис. 7.20 ▼ Пример использования функции Bulstoer

7.4.2. Решение дифференциальных уравнений первого порядка

Рассмотрим использование основных функций Mathcad на примере следующей задачи.

ЗАДАЧА 7.3. Решить задачу Коши:

$$\begin{aligned} \frac{dx}{dt} + x &= \sin(x \cdot t), \\ x(0) &= 1,5. \end{aligned}$$

Решение этой задачи с помощью решающего блока и функции Odesolve представлено на рис. 7.21. Еще раз обращаем внимание читателя, что в решающем блоке обязательно после имени функции указывать аргументы, в нашем случае это $x(t)$.

Для того чтобы дифференциальное уравнение вида $\varphi(t, x, x') = 0$ можно было решить с помощью других функций Mathcad, его необходимо преобразовать к виду $x' = F(t, x)$. Поэтому задачу 7.3 перепишем следующим образом:

$$\begin{aligned} \frac{dx}{dt} &= -x + \sin(x \cdot t), \\ x(0) &= 1,5. \end{aligned} \tag{7.37}$$

Решение дифференциального уравнения первого порядка с помощью функции `Odesolve`

Given

Дифференциальное уравнение и начальное условие

$$x'(t) + x(t) = \sin(x(t) \cdot t) \quad x(0) = 1.5$$

`x := Odesolve(t,35)`

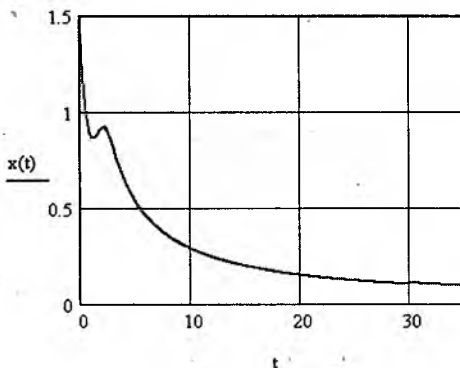


Рис. 7.21 ▼ Решение задачи 7.3 с помощью функции `Odesolve`

На рис. 7.22 представлено решение уравнения (7.37) с помощью функции `rkfixed`.

Как видно из рис. 7.22, при достаточно большом шаге в методе Рунге–Кутты с постоянным шагом накапливается большая ошибка. Если уменьшить шаг, то решение будет найдено точнее (рис. 7.23). Ошибка будет появляться только на последнем этапе интегрирования. При дальнейшем уменьшении шага решения, найденные с помощью функций `rkfixed` и `Odesolve`, совпадут.



Как следует из рис. 7.21–7.23, использование функции `rkfixed` даже при решении уравнения первого порядка может привести к значительным ошибкам вычислений. При использовании этой функции следует обращать внимание на величину шага интегрирования.

Чтобы избавиться от проблемы выбора шага интегрирования, на наш взгляд, рациональнее пользоваться функцией `Rkadapt`. На рис. 7.24 представлено решение уравнения (7.37) с помощью функции `Rkadapt`, которое полностью совпадает с решением, найденным с помощью функции `Odesolve` (см. рис. 7.21).

Правая часть дифференциального уравнения

$$D(t, x) := -x + \sin(x \cdot t)$$

Решение дифференциального уравнения на интервале [0;35]

с шагом интегрирования $\frac{(35 - 0)}{35} = 1$

$X_{\text{rkfixed}} := \text{rkfixed}(1.5, 0, 35, 35, D)$

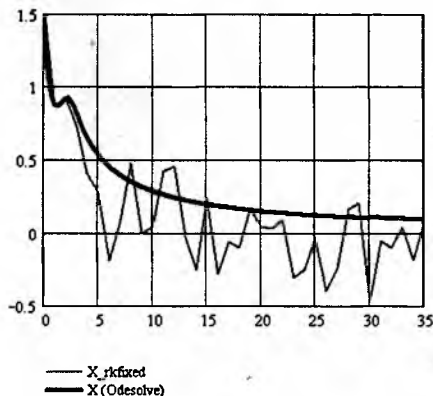


Рис. 7.22 ▽ Решение уравнения (7.37) с помощью функции rkfixed с шагом $ht = 1$

Правая часть дифференциального уравнения

$$D(t, x) := -x + \sin(x \cdot t)$$

Решение дифференциального уравнения на интервале [0;35]

с шагом интегрирования $\frac{(35 - 0)}{200} = 0.175$

$X_{\text{rkfixed}} := \text{rkfixed}(1.5, 0, 35, 200, D)$

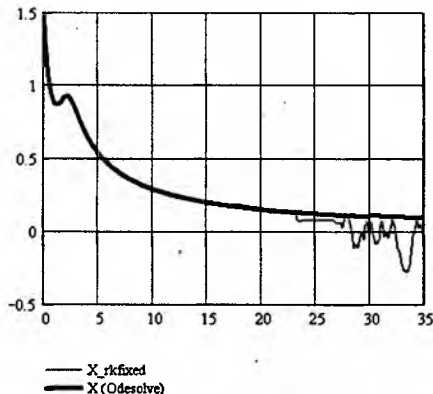


Рис. 7.23 ▽ Решение уравнения (7.37) с помощью функции rkfixed с шагом $ht = 0,175$

Правая часть дифференциального уравнения

$$D(t, x) := -x + \sin(x \cdot t)$$

Решение уравнения с помощью функции Rkadapt

$$X\_rkadapt := Rkadapt(1.5, 0, 35, 35, D)$$

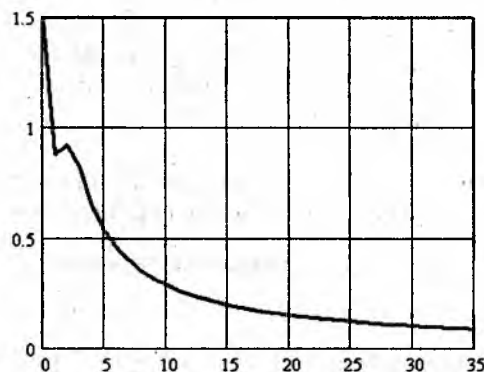


Рис. 7.24 ▼ Решение уравнения (7.37) с помощью функции rkadapt

Правая часть дифференциального уравнения

$$D(t, x) := -x + \sin(x \cdot t)$$

Решение уравнения с помощью функции Bulstoer

$$X\_bul := Bulstoer(1.5, 0, 35, 35, D)$$

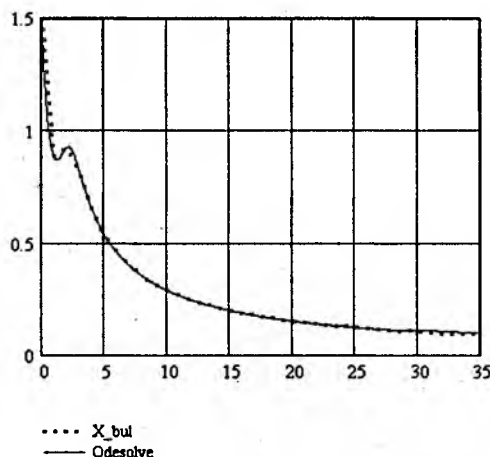


Рис. 7.25 ▼ Решение уравнения (7.37) с помощью функции Bulstoer

Еще одним способом решения обыкновенных дифференциальных уравнений первого порядка является использование функции Bulstoer. Найденное с помощью этой функции решение уравнения (7.37) представлено на рис. 7.25.

Решение, найденное с помощью функции Bulstoer, значительно точнее, чем с помощью функции Rkfixed при одном и том же шаге ($ht = 1$). Однако лучшим способом получения достаточно точного решения является использование rkadapt. Интересен и способ решения дифференциального уравнения с заданной точностью, представленный на рис. 7.26, в котором применена функция rkadapt.

С помощью рассмотренных в этом разделе функций можно решить практически любое дифференциальное уравнение первого порядка.

Количество промежутков на интервале интегрирования $N_{\text{int}} := 35$

Шаг интегрирования $H := \frac{35}{N}$ $t := 0..35$

Массив s - точки, в которых необходимо вычислить решение дифференциального уравнения, $y(i)$ - решение уравнения в точке $s(i)$

Начальное условие $s_0 := 0$ $y_0 := 1.5$

Формируем массив s $i := 1..N$
 $s_i := i \cdot H$

Ищем решение с точностью $\varepsilon := 0.0001$

$y_i := \text{rkadapt}(1.5, 0, s_i, \varepsilon, D, 2, 0.01)_{1,1}$

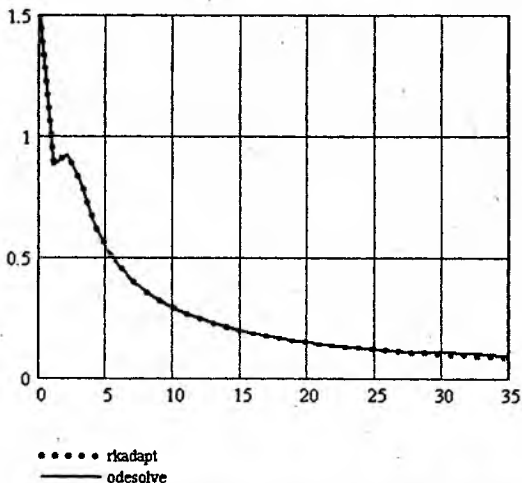


Рис. 7.26 ▽ Решение уравнения (7.37) с помощью функции rkadapt

7.4.3. Решение систем дифференциальных уравнений

Рассмотренные в предыдущем разделе функции позволяют решать не только дифференциальные уравнения, но и системы уравнений. Использование функций `Odesolve`, `rkfixed`, `Rkadapt`, `Bulstoer` при решении систем рассмотрим на примерах конкретных задач.

ЗАДАЧА 7.4. Решить задачу Коши:
$$\begin{cases} x' = \cos(x \cdot y) \\ y' = \sin(x + t \cdot y) \end{cases}, x(0) = 0, y(0) = 0$$

на интервале $[0; 10]$.

Решение задачи с помощью функции `Odesolve`

При решении систем с помощью функции `Odesolve` необходимо сделать следующее:

1. Внутри решающего блока записать уравнения и начальные условия.
2. Обратиться к функции `Odesolve` с тремя обязательными параметрами:

первый – массив имен функций (в нашем случае $\begin{pmatrix} x \\ y \end{pmatrix}$), второй – имя

`a := 0    b := 10`

Given

Уравнения и начальные условия

$$x'(t) = \cos(x(t) \cdot y(t)) \quad y'(t) = \sin(x(t) + t \cdot y(t))$$

$$x(a) = 0$$

$$y(a) = 0$$

Обращение к функции `Odesolve`

$$\begin{pmatrix} x \\ y \end{pmatrix} := \text{Odesolve} \left[\begin{pmatrix} x \\ y \end{pmatrix}, t, b \right]$$

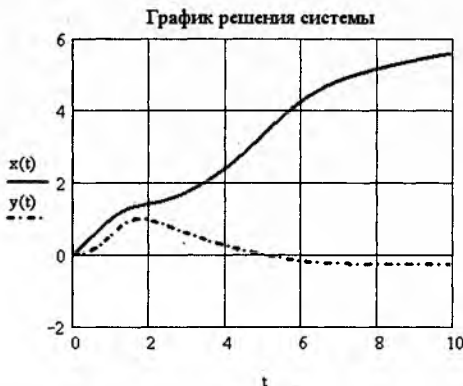


Рис. 7.27 ▼ Решение задачи 7.4 с помощью функции `Odesolve`

независимой переменной, третий – конечная точка интегрирования системы.

Функция `Odesolve` вернет вектор-функцию – решение системы обыкновенных дифференциальных уравнений. На рис. 7.27 представлено решение задачи 7.4 с помощью функции `Odesolve`.

Решение задачи с помощью функций `rkfixed`, `Rkadapt`, `Bulstoer`

Особенностью решения систем с помощью функций `rkfixed`, `Rkadapt`, `Bulstoer` является формирование параметра $D(t, x)$ как векторной функции правых частей системы. При записи этой функции следует учитывать, что t – имя независимой переменной, x – массив неизвестных функций (x_0 – первая функция, x_1 – вторая функция и т.д.). Векторная функция правой части $D(t, x)$ задачи 7.4 представлена на рис. 7.28.

$$D(t, x) := \begin{pmatrix} \cos(x_0 \cdot x_1) \\ \sin(x_0 + t \cdot x_1) \end{pmatrix}$$

Рис. 7.28 ▼ Функция правой части задачи 7.4

Обращение к функциям `rkfixed`, `Rkadapt`, `Bulstoer` при решении задачи 7.4 представлено на рис. 7.29. На рис. 7.30 представлен график решения, полученный с помощью функции `Rkadapt`¹.

$$\begin{aligned} H1 &:= rkfixed \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, a, b, 50, D \right] \\ H2 &:= Rkadapt \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, a, b, 50, D \right] \\ H3 &:= Bulstoer \left[\begin{pmatrix} 0 \\ 0 \end{pmatrix}, a, b, 50, D \right] \end{aligned}$$

Рис. 7.29 ▼ Решение задачи 7.4 с помощью функций `rkfixed`, `Rkadapt`, `Bulstoer`

Системы большей размерности можно решать аналогично задаче 7.4. Особый класс систем обыкновенных дифференциальных уравнений образуют жесткие задачи.

¹ Решения, полученные с помощью функций `rkfixed`, `Bulstoer`, совпадают с представленным на рис. 7.30.

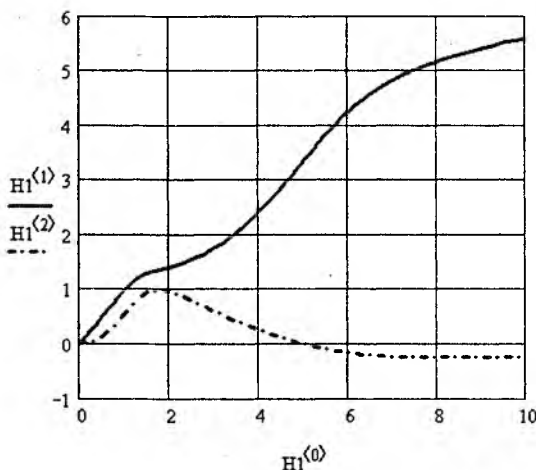


Рис. 7.30 ▼ График решения задачи 7.4, полученного с помощью функции Rkadapt

7.4.4. Решение жестких задач

Решение жестких систем содержит как быстро, так и медленно меняющиеся участки. Использование функции rkfixed при решении жестких систем зачастую нецелесообразно, так как Mathcad часто выдает сообщение о переполнении.

Для решения жестких систем в Mathcad предусмотрены следующие функции: Radau(init, t1, t2, npoints, D), Stiffb(init, t1, t2, eps, D, J) (используется алгоритм Булирша–Шера), Stiffc(init, t1, t2, eps, D, J) (используется алгоритм Розенброка). Параметрами этих функций являются:

- init – вектор начальных условий;
- [t1; t2] – интервал интегрирования;
- npoints – количество узлов на интервале [t1; t2];
- D – вектор правых частей системы дифференциальных уравнений;
- J – матрица размером $n \cdot (n + 1)$, нулевой столбец которой содержит производные правой части, а остальные столбцы представляют якобиан правой части системы дифференциальных уравнений.

Функции возвращают матрицу размером $(\text{npoints} + 1) \cdot (n + 1)$, нулевой столбец которой содержит все точки интервала интегрирования $[t_1; t_2]$, а все остальные содержат значения найденных функций в этих точках.

При обращении к функции Radau не нужно вычислять матрицу J, но методы, заложенные в функциях Stiffb(init, t1, t2, npoints, D, J), Stiffc(init, t1, t2, npoints, D, J), являются более точными при решении нелинейных жестких систем.

ЗАДАЧА 7.5. Найти решение задачи Коши для следующей жесткой системы:

$$\begin{cases} \frac{dX}{dt} = \begin{pmatrix} 119.46 & 185.38 & 126.88 & 121.03 \\ -10.395 & -10.136 & -3.636 & 8.577 \\ -53.302 & -85.932 & -63.182 & 54.211 \\ -115.58 & -181.75 & -112.8 & -199 \end{pmatrix} X; \quad X(0) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}. \end{cases} \quad (7.38)$$

Чтобы узнать, является ли система жесткой, необходимо проверить, удовлетворяют ли собственные числа матрицы B определению жесткой системы (рис. 7.31).

Матрица правых частей B

$$B := \begin{pmatrix} 119.46 & 185.38 & 126.88 & 121.03 \\ -10.395 & -10.136 & -3.636 & 8.577 \\ -53.302 & -85.932 & -63.182 & 54.211 \\ -115.58 & -181.75 & -112.8 & -199 \end{pmatrix}$$

Вычисляем собственные числа матрицы B $\lambda := \text{eigenvals}(B)$

$$\lambda = \begin{pmatrix} -151.442 \\ -0.202 + 17.167i \\ -0.202 - 17.167i \\ -1.012 \end{pmatrix}$$

$$\text{Число жесткости системы } s := \frac{\max(\text{Re}(\overrightarrow{|\lambda|}))}{\min(\text{Re}(\overrightarrow{|\lambda|}))} \quad s = 149.573$$

Рис. 7.31 ▼ Проверка жесткости системы (7.38)

Попробуем решить эту систему, используя стандартные функции. На рис. 7.32 представлено решение уравнения (7.38) с помощью функций `rkfixed` и `Rkadapt`.

Как видно из рис. 7.32, при решении жестких систем методом Рунге–Кутты с постоянным шагом ошибка очень быстро накапливается, что приводит к некорректным результатам, поэтому для его применения необходимо значительно уменьшать шаг. Но если решать подобные системы на достаточно большом интервале времени, то даже значительное уменьшение шага при использовании функции `rkfixed` может не привести к правильным значениям. Поэтому для решения подобных систем предусмотрены специальные функции. На рис. 7.33 представлено решение жесткой системы (7.38) с помощью функций `Radau`, `Stiffb`, `Stiffc`.

$$D(t, x) := \begin{pmatrix} 119.46 \cdot x_0 + 185.38 \cdot x_1 + 126.88 \cdot x_2 + 121.03 \cdot x_3 \\ -10.395 \cdot x_0 - 10.136 \cdot x_1 - 3.636 \cdot x_2 + 8.577 \cdot x_3 \\ -53.302 \cdot x_0 - 85.932 \cdot x_1 - 63.182 \cdot x_2 - 54.211 \cdot x_3 \\ -115.58 \cdot x_0 - 181.75 \cdot x_1 - 112.8 \cdot x_2 - 199 \cdot x_3 \end{pmatrix}$$

$$X := \text{rkfixed} \left[\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, 0, 5, 200, D \right] \quad XA := \text{Rkadapt} \left[\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, 0, 5, 100, D \right]$$

$$X =$$

	0	1	2	3	4
0	0	1	1	1	1
1	0.025	2.042	-2.241	0.844	11.775
2	0.05	-13.857	-9.126	9.121	55.474
3	0.075	-93.602	-27.593	47.466	225.431
4	0.1	-416.758	-90.179	199.176	885.485
5	0.125	-1.681·10 ³	-325.457	788.516	3.454·10 ³
6	0.15	-6.604·10 ³	-1.236·10 ³	3.08·10 ³	1.346·10 ⁴
7	0.175	-2.578·10 ⁴	-4.782·10 ³	1.2·10 ⁴	5.245·10 ⁴
8	0.2	-1.005·10 ⁵	-1.86·10 ⁴	4.676·10 ⁴	2.044·10 ⁵
9	0.225	-3.916·10 ⁵	-7.248·10 ⁴	1.822·10 ⁵	7.966·10 ⁵
10	0.25	-1.526·10 ⁶	-2.824·10 ⁵	7.101·10 ⁵	3.104·10 ⁶

$$XA =$$

	0	1	2	3	4
0	0	1	1	1	1
1	0.05	14.825	-3.82	-4.223	-2.859
2	0.1	18.843	-9.573	-3.482	-0.53
3	0.15	12.222	-11.79	1.63	2.464
4	0.2	-0.403	-8.866	7.423	4.086
5	0.25	-10.364	-2.761	9.815	3.188

Рис. 7.32 ▼ Решение жесткой системы (7.38) с помощью функций rkfixed и Rkadapt

$$X1 := \text{Radau} \left[\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, 0, 5, 100, D \right]$$

Матрица J

$$J(t, x) := \begin{pmatrix} 0 & 119.46 & 185.38 & 126.88 & 121.03 \\ 0 & -10.395 & -10.136 & -3.636 & 8.577 \\ 0 & -53.302 & -85.932 & -63.182 & -54.211 \\ 0 & -115.58 & -181.75 & -112.8 & -199 \end{pmatrix}$$

$$X2 := \text{Stiffb} \left[\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, 0, 5, 250, D, J \right] \quad X3 := \text{Stiff} \left[\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, 0, 5, 100, D, J \right]$$

Рис. 7.33 ▼ Решение жесткой системы (7.38) с помощью функций Radau, Stiffb, Stiff

На рис. 7.34 представлены числовые результаты решения задачи 7.5.

	0	1	2	3	4
85	4.25	-3.973	-0.984	2.976	1.557
86	4.3	-6.556	1.68	3.008	0.693
87	4.35	-4.558	3.093	1.005	-0.624
88	4.4	0.558	2.314	-1.62	-1.481
89	4.45	5.22	-0.076	-3.056	-1.302
90	4.5	6.237	-2.405	-2.343	-0.228
91	4.55	2.985	-3.075	-0.014	0.985
92	4.6	-2.223	-1.657	2.3	1.502
93	4.65	-5.779	0.836	3.012	0.983
94	4.7	-5.275	2.674	1.663	-0.196
95	4.75	-1.142	2.612	-0.779	-1.213
96	4.8	3.713	0.732	-2.621	-1.375
97	4.85	5.944	-1.64	-2.611	-0.587
98	4.9	4.075	-2.866	-0.795	0.591
99	4.95	-0.534	-2.125	1.546	1.342
100	5	-4.667	0.034	2.794	1.162

X1 =

	0	1	2	3	4
234	4.68	-6.005	2.121	2.416	0.306
235	4.7	-5.275	2.674	1.663	-0.196
236	4.72	-3.934	2.906	0.727	-0.671
237	4.74	-2.142	2.794	-0.283	-1.063
238	4.76	-0.112	2.353	-1.249	-1.328
239	4.78	1.918	1.637	-2.059	-1.435
240	4.8	3.712	0.732	-2.621	-1.375
241	4.82	5.064	-0.256	-2.869	-1.154
242	4.84	5.819	-1.21	-2.78	-0.801
243	4.86	5.895	-2.021	-2.365	-0.357
244	4.88	5.288	-2.594	-1.676	0.126
245	4.9	4.074	-2.865	-0.794	0.591
246	4.92	2.4	-2.805	0.175	0.983
247	4.94	0.463	-2.423	1.118	1.259
248	4.96	-1.51	-1.767	1.927	1.386
249	4.98	-3.288	-0.914	2.507	1.352

X2 =

Рис. 7.34 ▾ Результаты решения жесткой системы (7.38) с помощью функций Radau, Stiffb, Stiffc

Описание жестких систем завершим решением нелинейной жесткой системы.

ЗАДАЧА 7.6. Решить нелинейную жесткую систему дифференциальных уравнений:

$$\begin{cases} \frac{dx_0}{dt} = -7 \cdot x_0 + 7 \cdot x_1, \\ \frac{dx_1}{dt} = 157 \cdot x_0 + x_1 - 1.15 \cdot x_0 \cdot x_2, \\ \frac{dx_2}{dt} = 0.96 \cdot x_0 \cdot x_1 - 8.36 \cdot x_2. \end{cases} \quad (7.39)$$

На рис. 7.35 представлено решение системы на интервале $[0; 2]$ с помощью функции `Radau`, на рис. 7.36 – решение этой системы с помощью функции `Stiff`.

Матрица правых частей жесткой нелинейной системы

$$D(t, x) := \begin{pmatrix} -7 \cdot x_0 + 7 \cdot x_1 \\ 157 \cdot x_0 + x_1 - 1.5 \cdot x_0 \cdot x_2 \\ 0.96 \cdot x_0 \cdot x_1 - 8.36 \cdot x_2 \end{pmatrix}$$

$$X := \text{Radau} \left(\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, 0, 2, 100, D \right)$$

$X =$

	0	-1	2	3
0	0	-1	0	1
1	0.02	-1.077	-3.151	0.875
2	0.04	-1.606	-7.3	0.861
3	0.06	-2.765	-14.103	1.136
4	0.08	-4.973	-26.099	2.319
5	0.1	-9.022	-47.307	6.452
6	0.12	-16.209	-82.535	19.875
7	0.14	-27.922	-128.05	58.51
8	0.16	-42.541	-137.894	137.471
9	0.18	-49.584	-40.404	195.932
10	0.2	-40.391	65.452	150.141
11	0.22	-24.89	78.888	82.468
12	0.24	-12.677	58.014	47.299
13	0.26	-4.586	42.811	32.655
14	0.28	1.291	40.196	26.558
15	0.3	6.927	50.693	25.802

Рис. 7.35 ▼ Решение системы (7.39) с помощью функции `Radau`



При исследовании нелинейных жестких систем не следует решать их сразу на всем большом интервале. Эти задачи очень сложные, и даже при использовании функций `Radau`, `Stiffb`, `Stiff` решение на большом интервале может приводить к накоплению больших ошибок интегрирования. Лучшее – большой интервал делить на небольшие участки и интегрировать жесткую систему на каждом из них.

Матрица правых частей жесткой нелинейной системы

$$D(t, x) := \begin{pmatrix} -7 \cdot x_0 + 7 \cdot x_1 \\ 157 \cdot x_0 + x_1 - 1.15 \cdot x_0 \cdot x_2 \\ 0.96 \cdot x_0 \cdot x_1 - 8.36 \cdot x_2 \end{pmatrix}$$

$$J(t, x) := \begin{pmatrix} 0 & -7 & 7 & 0 \\ 0 & 157 - 1.15 \cdot x_2 & 1 & 1.15 \cdot x_0 \\ 0 & 0.96 \cdot x_1 & 0.96 \cdot x_0 & -8.36 \end{pmatrix}$$

$$X1 := \text{Stiff} \left[\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, 0, 2, 100, D, J \right]$$

X1 =

	0	1	2	3
85	1.7	-0.851	4.287	25.464
86	1.72	-0.294	2.944	21.526
87	1.74	0.116	2.794	18.208
88	1.76	0.515	3.71	15.423
89	1.78	1.066	5.973	13.115
90	1.8	1.97	10.36	11.312
91	1.82	3.553	18.416	10.265
92	1.84	6.374	32.923	10.912
93	1.86	11.394	58.351	16.335
94	1.88	20.093	99.36	35.73
95	1.9	33.789	147.967	90.295
96	1.92	49.682	146.098	192.226
97	1.94	55.432	29.033	251.014
98	1.96	43.837	-76.266	184.182
99	1.98	26.881	-84.045	102.785
100	2	13.946	-60.281	61.268

Рис. 7.36 ▽ Решение системы (7.39) с помощью функции Stiff

7.4.5. Решение дифференциальных уравнений высших порядков

Решение дифференциальных уравнений высших порядков в Mathcad можно осуществлять непосредственно с помощью функции Odesolve или свести к системе уравнений первого порядка и воспользоваться функциями rkfixed, Rkadapt.

Рассмотрим обе возможности на примере задачи 7.2.

На рис. 7.37 представлено решение этой задачи с помощью функции Odesolve в Mathcad.

Для того чтобы можно было воспользоваться функциями rkfixed, Rkadapt, преобразуем уравнение в систему (7.35) с начальными условиями (7.36). На рис. 7.38 представлено решение задачи 7.2 в виде системы с помощью функции Rkadapt.

Границы интегрирования

$a := 0.25$ $b := 2 \cdot \pi$

Given

Дифференциальное уравнение

$$\frac{d^2}{dt^2}x(t) + 4 \cdot \left(\frac{d}{dt}x(t) \right) + 13 \cdot x(t) = e^{\sin(t)}$$

Начальные условия $x(a) = -1$ $x'(a) = 1$

$x := \text{Odesolve}(t, b)$

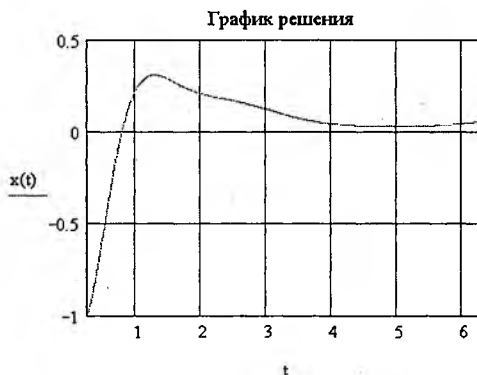


Рис. 7.37 ▼ Решение дифференциального уравнения второго порядка с помощью функции `Odesolve`

$$D(t, x) := \begin{pmatrix} x_1 \\ -4 \cdot x_1 - 13 \cdot x_0 + e^{\sin(t)} \end{pmatrix} \quad \underline{R} := \text{Rkadapt} \left[\begin{pmatrix} -1 \\ 1 \end{pmatrix}, a, b, 25, D \right]$$

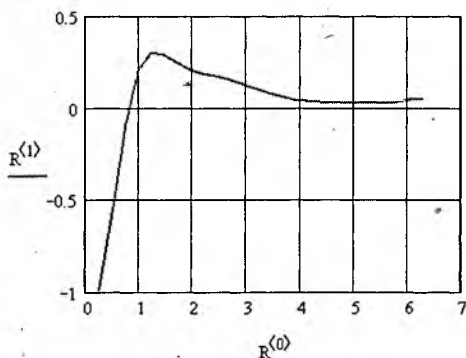


Рис. 7.38 ▼ Решение задачи 7.2 с помощью функции `Rkadapt`

Как видно из рис. 7.37–7.38, решения задачи 7.2, найденные двумя методами, практически совпадают.

7.4.6. Решение двухточечных краевых задач в Mathcad

Особый класс дифференциальных уравнений образуют задачи с дополнительными условиями, заданными на двух концах интервала интегрирования. В этом случае вместо задачи Коши с начальными условиями получается краевая задача. Двухточечная краевая задача решается в два этапа. На первом этапе методом стрельбы определяются недостающие начальные условия. На втором этапе решается обычная задача Коши.

Для нахождения недостающих начальных условий методом стрельбы в Mathcad предназначена функция `sbval` ($v, t1, t2, D, l, s$). Здесь v – вектор недостающих начальных условий, $[t1, t2]$; – интервал интегрирования, D – вектор правых частей системы дифференциальных уравнений, l – векторная функция начальных значений, недостающие элементы определяются как компоненты вектора v , s – вектор, в котором хранится разность между точным и расчетным значениями функций на концах отрезка.

Решение двухточечной задачи рассмотрим на следующем примере.

ЗАДАЧА 7.7. Решить краевую задачу:

$$\begin{aligned} (t^3 + 1)x'' - tx' &= 3t^4, \\ x(2) &= 3; \quad x(4) = 11. \end{aligned}$$

Нам неизвестно значение производной в точке $t=2$. Поэтому вектор v должен содержать единственное произвольное значение, пусть это будет число 3 ($v_0=3$). Определяем значения $t_1=2$ и $t_2=4$. После преобразования исходного уравнения к системе сформируем вектор правых частей:

$$D(t, x) = \begin{pmatrix} x_1 \\ \frac{3 \cdot t^4 + x_1}{t^3 + 1} \end{pmatrix}.$$

Далее определяем вектор l , в котором хранятся два значения: начальное значение x , равное 2, и неизвестное значение производной, определенное нами как v_0 . В связи с тем, что на конце отрезка задано единственное значение $x(4)=11$, скалярная функция s может быть определена как $x_0 - 11$. Затем необходимо обратиться к функции `sbval`, которая вернет вектор недостающих значений (в нашем случае это значение производной). Описанный процесс определения недостающего значения производной в точке $t=2$ для задачи 7.7 методом стрельбы представлен на рис. 7.39.

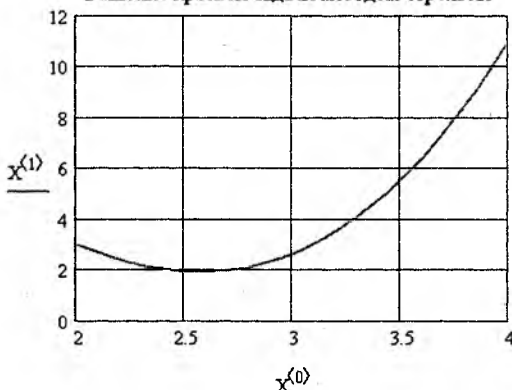
Далее остается решить задачу Коши с помощью одной из функций Mathcad. На рис. 7.40 представлено решение задачи 7.7 с помощью функции `Rkadapt`.

$$\begin{aligned}
 v_0 &:= 3 & t_1 &:= 2 & t_2 &:= 4 \\
 l(t_1, v) &:= \begin{pmatrix} 3 \\ v_0 \end{pmatrix} & s(t_2, x) &:= x_0 - 11 & D(t, x) &:= \begin{pmatrix} x_1 \\ 3 \cdot t^4 + x_1 \\ t^3 + 1 \end{pmatrix} \\
 H &:= \text{sbval}(v, t_1, t_2, D, l, s) & H &= (-3.542)
 \end{aligned}$$

Рис. 7.39 ▼ Реализация метода стрельбы для задачи 7.7

$$X := \text{Rkadapt}\left(\begin{pmatrix} 3 \\ H_0 \end{pmatrix}, t_1, t_2, 25, D\right)$$

Решение краевой задачи методом стрельбы



	0	1	2
0	2	3	-3.542
1	2.08	2.733	-3.132
2	2.16	2.5	-2.694
3	2.24	2.302	-2.228
4	2.32	2.144	-1.736
5	2.4	2.025	-1.218
6	2.48	1.949	-0.675
7	2.56	1.918	-0.108
8	2.64	1.933	0.483
9	2.72	1.996	1.098
10	2.8	2.109	1.735
11	2.88	2.274	2.395
12	2.96	2.493	3.077
13	3.04	2.767	3.781
14	3.12	3.099	4.507
15	3.2	3.489	5.254
16	3.28	3.94	6.022
17	3.36	4.453	6.812
18	3.44	5.03	7.622
19	3.52	5.673	8.453
20	3.6	6.383	9.304
21	3.68	7.162	10.175
22	3.76	8.012	11.067
23	3.84	8.934	11.98
24	3.92	9.929	12.912
25	4	11	13.864

X =

Рис. 7.40 ▼ Решение задачи 7.7 с помощью функции Rkadapt

Кроме функции `sbval` для реализации метода стрельбы в Mathcad предусмотрена функция `bvalfit`, которая вычисляет недостающие начальные условия, используя промежуточную точку интервала интегрирования.

После изучения большого количества функций, предназначенных для решения дифференциальных уравнений и систем, пользователь столкнется с проблемой выбора метода решения обыкновенных дифференциальных уравнений в Mathcad. Наиболее точные результаты в не жестких задачах можно получить, используя функции `Odesolve` и `Rkadapt`. При необходимости получить решение в одной или нескольких точках можно воспользоваться функцией `rkadapt`. При решении жестких линейных систем хорошие результаты могут быть получены при корректном использовании функций `Radau`, `Stiffb`, `Stiffc`. Самый сложный

класс задач представляют жесткие нелинейные систем, для решения которых авторы советуют разбивать интервал на небольшие участки и интегрировать жесткую систему с помощью функций `Radau`, `Stiffb`, `Stiffn` на каждом из них. При решении сложных дифференциальных уравнений и систем следует не только хорошо знать пакет `Mathcad`, но и методы вычислительной математики.

7.5. Решение дифференциальных уравнений в Maple

Многие численные методы, описанные в разделе 7.2, могут быть запрограммированы в Maple. Однако Maple располагает встроенной функцией `dsolve` для численного и аналитического решения дифференциальных уравнений и систем следующей структуры:

```
dsolve({ODE, IC}, { funcs},extra_args).
```

Здесь `ODE` – дифференциальное уравнение или система уравнений, `IC` – начальные или граничные условия, `funcs` – неизвестная функция (или функции), `extra_args` – параметры, определяющие метод решения задачи, для численного решения необходимо указать `type=numeric`.

7.5.1. Решение дифференциальных уравнений

Решение дифференциального уравнения рассмотрим на примере уравнения 7.34. Попробуем найти его аналитическое решение (рис. 7.41).

```
> Eq1:=-diff(x(t),t^2)+4*diff(x(t),t)+13*x(t)=exp(t);# Определяем уравнение
```

$$Eq1 := \left(\frac{d^2}{dt^2} x(t) \right) + 4 \left(\frac{d}{dt} x(t) \right) + 13 x(t) = e^t$$

```
> Init_cond:=x(0.25)=-1,D(x)(0.25)=1;# Определяем начальные условия
Init_cond = x(0.25) = -1, D(x)(0.25) = 1
```

```
> dsolve({Eq1,Init_cond},x(t));# Решаем уравнение аналитически с помощью функции dsolve
```

$$x(t) = -\frac{1}{18} e^{(-2t)} \sin(3t) e^{\left(\frac{1}{2}\right)} \left(\cos\left(\frac{3}{4}\right) e^{\left(\frac{1}{4}\right)} + 6 \cos\left(\frac{3}{4}\right) + e^{\left(\frac{1}{4}\right)} \sin\left(\frac{3}{4}\right) + 18 \sin\left(\frac{3}{4}\right) \right) \\ - \frac{1}{18} e^{(-2t)} \cos(3t) e^{\left(\frac{1}{2}\right)} \left(-e^{\left(\frac{1}{4}\right)} \sin\left(\frac{3}{4}\right) - 6 \sin\left(\frac{3}{4}\right) + \cos\left(\frac{3}{4}\right) e^{\left(\frac{1}{4}\right)} + 18 \cos\left(\frac{3}{4}\right) \right) + \frac{1}{18} e^t$$

Рис. 7.41 ▼ Аналитическое решение уравнения 7.34

Теперь получим численное решение, для чего в функции `dsolve` в качестве третьего параметра допишем `type=numeric` (рис. 7.42).

```
> x(0.25);x(4);# Вычисляем численное значение в точках
[ t = 0.250000000000000000, x(t) = -1., d/dt x(t) = 1. ]
[ t = 4., x(t) = 3.03329879037338657, d/dt x(t) = 3.03120716192098216 ]
```

Рис. 7.42 ▼ Численное решение уравнения 7.34

7.5.2. Решение систем обыкновенных дифференциальных уравнений

Теперь на примере этой же задачи рассмотрим решение систем. Уравнение 7.34 может быть записано как система обыкновенных дифференциальных уравнений (7.35) с начальными условиями (7.36). Решение систем в Maple мало чем отличается от решения уравнений, просто вместо одного уравнения через запятую записывается несколько.

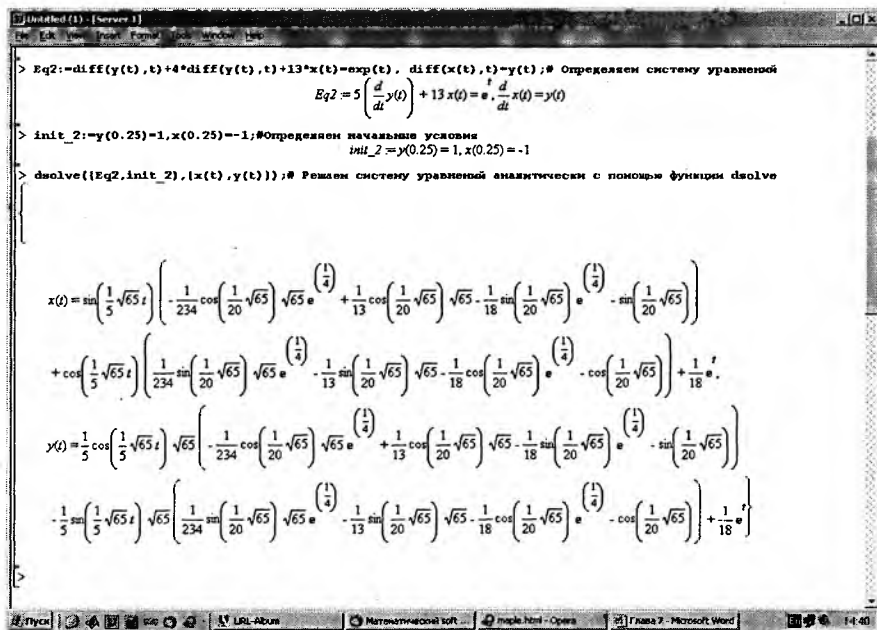


Рис. 7.43 ▼ Аналитическое решение системы (7.35), (7.36).

Аналитическое решение системы (7.35)–(7.36) представлено на рис. 7.43, численное – на рис. 7.44.

```
> Eq2:=diff(y(t),t)+4*diff(y(t),t)+13*x(t)-exp(t), diff(x(t),t)-y(t);# Определяем систему уравнений
Eq2 = 5  $\left(\frac{d}{dt}y(t)\right) + 13 x(t) = e^t, \frac{d}{dt}x(t) = y(t)$ 

> init_2:=y(0.25)=1,x(0.25)=-1;#Определяем начальные условия
init_2 := y(0.25) = 1, x(0.25) = -1

> XY:=dsolve({Eq2,init_2},{x(t),y(t)},type=numeric);# Решаем систему уравнений численно с помощью функции
dsolve
XY := proc(x_kf/4.5) ... end proc

> XY(0.25); XY(1); XY(2);# Вычисляем значение x и y в нужных точках
[t = 0.2500000000000000000, x(t) = -1., y(t) = 1.]
[t = 1., x(t) = 0.310869135779636985, y(t) = 2.09527731732336030]
[t = 2., x(t) = 1.60858055941627653, y(t) = 0.0720054469459847774]
```

Рис. 7.44 ▼ Численное решение системы (7.35), (7.36)

Аналогичным образом можно решать и более сложные системы. В качестве еще одного примера рассмотрим жесткую систему (7.39). Ее решение в Maple показано на рис. 7.45. Однако, как и в Mathcad, интервал интегрирования системы следует разбивать на меньшие участки и решать жесткую систему последовательно на каждом из них.

```
> eq3:=diff(x(t),t)=-7*x(t)+7*y(t), diff(y(t),t)=157*x(t)+y(t)-1.15*x(t)*z(t), diff(z(t),t)=0.96*x(t)*y(t)-8.36
*x(t);#Система дифференциальных уравнений
eq3 =  $\frac{d}{dt}x(t) = -7 x(t) + 7 y(t), \frac{d}{dt}y(t) = 157 x(t) + y(t) - 1.15 x(t) z(t), \frac{d}{dt}z(t) = 0.96 x(t) y(t) - 8.36 z(t)$ 

> init_3:=-x(0)=-1,y(0)=0,z(0)=1;#Начальные условия системы
init_3 := x(0) = -1, y(0) = 0, z(0) = 1

> XYZ:=dsolve({eq3,init_3},{x(t),y(t),z(t)},type=numeric);# Формируем численное решение системы
XYZ := proc(x_kf/4.5) ... end proc

> XYZ(0);XYZ(0.2);XYZ(0.3);# Вычисляем значение x, y и z в нужных точках
[t = 0., x(t) = -1., y(t) = 0., z(t) = 1.]
[t = 0.2, x(t) = -49.5647195037659928, y(t) = 58.2453260574648155, z(t) = 216.304021328251850]
[t = 0.3, x(t) = 6.57020150667878066, y(t) = 54.5444048741985910, z(t) = 33.2890311341916245]
```

Рис. 7.45 ▼ Численное решение системы (7.39)

Для более подробного знакомства с аналитическими методами решения и исследования обыкновенных дифференциальных уравнений авторы рекомендуют читателю прочитать книгу [8].

7.6. Советы по применению математических пакетов при решении дифференциальных уравнений

Рассматриваемые математические пакеты позволяют решать обыкновенные дифференциальные уравнения и системы. Однако, на взгляд авторов, для численного решения уравнений лучше использовать пакет Mathcad: большое количество встроенных функций пакета позволяют решать задачи любой сложности. Для аналитического решения обыкновенных дифференциальных уравнений можно пользоваться функцией `dsolve` пакета Maple, эта функция неплохо справляется и с численным решением уравнений и систем.

Особый класс задач представляют жесткие системы обыкновенных дифференциальных уравнений. Последние версии Mathcad (11-я и 12-я) предоставляют пользователю возможности для исследования жестких систем, однако следует помнить, что при решении этих очень сложных задач в большинстве случаев пользователю придется разбираться с численными методами решения конкретной жесткой системы. Эти задачи очень хорошо решает свободно распространяемый пакет Scilab. По своему интерфейсу и синтаксису он близок к MATLAB, однако жесткие системы, на взгляд авторов, он решает лучше. В листинге 7.10 представлено решение жесткой системы в Scilab, а на рис. 7.46 – полученный график решения.

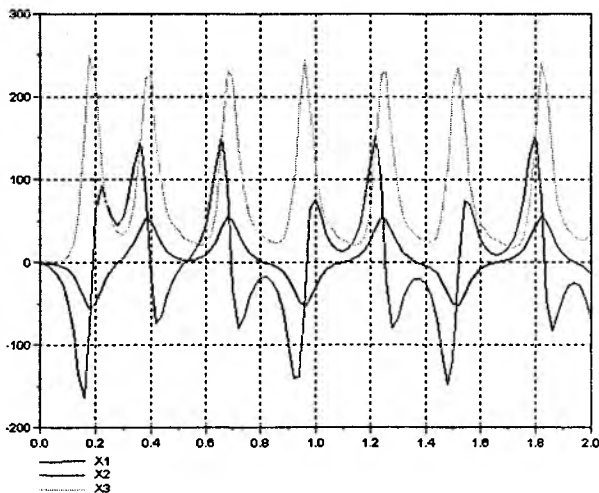


Рис. 7.46 ▽ График полученного в Scilab решения жесткой системы (7.39)

Листинг 7.10

```

function [y]=hh(t,x)
y(1)=-7*x(1)+7*x(2)
y(2)=157*x(1)+x(2)-1.15*x(1)*x(3)
y(3)=0.96*x(1)*x(2)-8.36*x(3)
endfunction
t0=0;
tk=2;
t=t0:0.02:tk;
X=[-1;0;1];
z=ode(X,t0,t,hh);
plot2d(t,z',leg="X1@X2@X3");
xgrid;

```

7.7. Задания для самостоятельного решения

1. Решить задачу Коши $-4x'' + 4x' - x = 0$, $x(2) = -1$, $x'(2) = 0.5$ на интервале $[2; 10]$ с помощью функций Odesolve, Rkadapt и rkfixed.
2. Сравнить решения задачи Коши $-4x'' + 4x' - x = \sin(t) + \cos(3t)$, $x(1) = 5$, $x'(1) = 0.25$ на интервале $[1; 9]$, полученные с помощью Mathcad, MATLAB и Maple.
3. Решить систему дифференциальных уравнений

$$\begin{cases} x' = \cos(x + 2y) \\ y' = \sin(3x - 4y) \end{cases}, x(0) = 2, y(0) = 1 \text{ в Mathcad и Maple.}$$

4. Решить систему дифференциальных уравнений

$$\begin{cases} x' = \cos(x + 2y) \\ y' = \sin(3x - 4y) \end{cases}, x(0) = 2, y(0) = 1 \text{ в MATLAB.}$$

5. Решить граничную задачу методом стрельбы:

$$-4x'' + 4x' - x = \sin(t) + \cos(3t), \quad x(1) = 5, \quad x(3) = 0.25.$$

6. Решить граничную задачу методом стрельбы:

$$-5x'' + 17x' - 3x = \frac{t}{t^2 + 3}, \quad x(0) = 5, \quad x(3) = 9.$$

7. Решить систему дифференциальных уравнений $\begin{cases} x' = \frac{x^2}{y} \\ y' = x^2 - y \end{cases}$, $x(1) = 2$, $y(1) = 4$ на интервале $[1; 4]$.

8. Решить систему дифференциальных уравнений $\begin{cases} x' = y \ln t \\ y' = x + y^2 \end{cases}$, $x(1) = -1$, $y(1) = -3$ на интервале $[1; 5]$, используя решающий блок Mathcad.

9. Решить краевую задачу методом стрельбы:

$$x'' + t^2 x' - tx = \frac{6}{x^4} - \frac{3}{x}, \quad x(1) = 0, \quad x(2) = 0.25.$$

10. Найти решение задачи Коши в Mathcad и Maple для следующей системы:

$$\left\{ \frac{dX}{dt} = \begin{pmatrix} 100 & 156 & 107 & 102 \\ -10.5 & -10 & -4 & -8 \\ -40 & -65 & -47.5 & -41 \\ -95 & -152 & -102 & -99 \end{pmatrix} X; \quad X(0) = \begin{pmatrix} 1 \\ 2 \\ 1.5 \\ -1 \end{pmatrix} \right.$$

Перед решением проверить, не является ли система жесткой.



Решение дифференциальных уравнений в частных производных

Математические модели физических и иных процессов описываются с помощью дифференциальных уравнений в частных производных¹. Аргументами функций этих уравнений являются пространственные координаты x , y , z и время t . Общие сведения о дифференциальных уравнениях в частных производных приведены в первом разделе главы. Один из основных аналитических методов решения уравнений математической физики – метод разделения переменных – рассматривается во втором разделе. В третьем разделе описан метод сеток решения дифференциальных уравнений в частных производных, представлена реализация этого метода в Mathcad, в четвертом разделе описаны встроенные функции Mathcad для решения уравнений в частных производных. В пятом разделе рассматривается использование PDE Toolbox MATLAB 7 при решении уравнений в частных производных. Шестой раздел познакомит читателя с возможностями Maple для аналитического решения уравнений математической физики.

8.1. Общие сведения о дифференциальных уравнениях в частных производных

Линейным уравнением в частных производных второго порядка называется соотношение между функцией $u(x, y)$ (или $u(x, t)$) и ее частными производными вида [7, 9, 11, 21]:

$$\begin{aligned}
 L(u) = & \\
 = & A(x, y) \frac{\partial^2 u}{\partial x^2} + 2B(x, y) \frac{\partial^2 u}{\partial x \partial y} + C(x, y) \frac{\partial^2 u}{\partial y^2} + D(x, y) \frac{\partial u}{\partial x} + \\
 & + E(x, y) \frac{\partial u}{\partial y} + G(x, y)u = F(x, y).
 \end{aligned} \tag{8.1}$$

Если переменная функция u зависит от x и t , то уравнение (8.1) может быть записано следующим образом:

$$\begin{aligned}
 L(u) = & \\
 = & A(x, t) \frac{\partial^2 u}{\partial x^2} + 2B(x, t) \frac{\partial^2 u}{\partial x \partial t} + C(x, t) \frac{\partial^2 u}{\partial t^2} + D(x, t) \frac{\partial u}{\partial x} + \\
 & + E(x, t) \frac{\partial u}{\partial t} + G(x, t)u = F(x, t).
 \end{aligned} \tag{8.2}$$

В случае если $F = 0$, то уравнения (8.1)–(8.2) называются однородными, иначе – неоднородными [7, 9, 11, 21].

Если $B^2 - AC < 0$, то уравнение (8.2) относится к классу *эллиптических уравнений*; если $B^2 - AC > 0$, то (8.2) – это *гиперболическое уравнение*; если $B^2 - AC = 0$ – *параболическое уравнение*. Когда $B^2 - AC$ не имеет постоянного знака, получается уравнение смешанного типа.

С помощью преобразования переменных x, y (или x, t) уравнение можно привести к виду, когда $B = 0$ [7, 21]. В этом случае очень просто определяется тип уравнения. Если A и C имеют один и тот же знак, то (8.2) – *эллиптическое уравнение*, если разные, то *гиперболическое*, а если A или C равно 0, то уравнение относится к *параболическим* [7, 9, 21].

К классическим эллиптическим уравнениям относятся [7, 9, 21]:

- уравнение Лапласа $\Delta u = 0$, которое используется для описания магнитных и стационарных тепловых полей¹;
- уравнение Пуассона $\Delta u = f$, которое применяется в электростатике, теории упругости и других науках;
- уравнение Гельмгольца $\Delta u + cu = f$, описывающее установившиеся колебательные процессы.

¹ В двумерном случае оператор Лапласа имеет вид $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$, в трехмерном –

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}.$$

Среди гиперболических уравнений можно выделить [7, 9, 21]:

➤ волновые уравнения: одномерное $\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} + f(x, t)$, которое описывает вынужденные колебания струны; двумерное $\frac{\partial^2 u}{\partial t^2} = a^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + f(x, y, t)$, которое описывает колебания мембраны¹;

➤ телеграфное уравнение $\frac{\partial^2 u}{\partial t^2} + \frac{RC + LG}{LC} \frac{\partial u}{\partial t} + \frac{RG}{LC} u - \frac{1}{LC} \frac{\partial^2 u}{\partial x^2} = 0$, которое описывает изменение потенциала u в линиях электропередачи. Здесь L , C , R , G – коэффициент самоиндукции, емкость, сопротивление, характеристика потерь на единицу длины линии.

К классическим параболическим уравнениям относится уравнение теплопроводности $\frac{\partial u}{\partial t} = a^2 \Delta u + f$.

Для нахождения единственного решения дифференциального уравнения в частных производных необходимо задать начальные и граничные условия. Начальными условиями принято называть условия, заданные в начальный момент времени t . Граничные условия задаются при различных значениях пространственных переменных. Для эллиптических уравнений задаются только граничные условия, которые можно разделить на три класса:

➤ условие Дирихле $u|_{(x, y, z) \in \Gamma} = \varphi(t)$ – в этом случае на границе области Γ , в которой ищется решение, задана некая непрерывная функция φ . В одномерном случае это условие принимает вид: $u(0, t) = \varphi_1(t)$; $u(L, t) = \varphi_2(t)$, где $(0, L)$ – интервал, на котором ищется решение одномерной задачи;

➤ условие Неймана $\left. \frac{\partial u}{\partial n} \right|_{(x, y, z) \in \Gamma} = \varphi(t)$ – в этом случае на границе области задана производная по направлению n внешней нормали;

➤ смешанное условие $\left(\alpha u + \beta \frac{\partial u}{\partial n} \right) \Big|_{(x, y, z) \in \Gamma} = \varphi(t)$.

¹ При $f = 0$ уравнение описывает свободные колебания струны или мембраны.

Для *параболических* уравнений, кроме граничных условий, необходимо определить одно начальное, которое может быть таким: $u(x, t_0) = \psi(x)$.

В случае гиперболических уравнений начальные условия могут быть следующими: $u(x, t_0) = \psi_1(x)$ и $\frac{\partial u(x, t_0)}{\partial t} = \psi_2(x)$.

Решение ряда дифференциальных уравнений в частных производных может быть получено аналитически. Одним из наиболее часто используемых методов является метод разделения переменных (метод Фурье). Рассмотрим этот метод подробнее.

8.2. Решение уравнений в частных производных методом разделения переменных

В основе метода разделения переменных (метода Фурье) лежит *принцип суперпозиции*, который состоит в следующем: если функции $u_1, u_2, \dots, u_k, \dots$ являются решением линейного однородного дифференциального уравнения в частных производных $L(u) = 0$ (см. формулы (8.1)–(8.2)), то и бесконечный ряд

$u = \sum_{i=1}^{\infty} c_i u_i = c_1 u_1 + c_2 u_2 + \dots + c_k u_k + \dots$ также является решением уравнения

$L(u) = 0$, если он сходится к некоторой функции u и допускает почленное дифференцирование.

Рассмотрим метод разделения переменных (метод Фурье) на примере задачи колебания струны [21]:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} \quad (8.3)$$

с однородными граничными условиями:

$$u(0, t) = 0, u(L, t) = 0 \quad (8.4)$$

и начальными условиями вида:

$$u(x, 0) = \mu(x), \frac{\partial u(x, 0)}{\partial t} = v(x). \quad (8.5)$$

Решение будем искать в виде произведения [21]:

$$u(x, t) = X(x) T(t). \quad (8.6)$$

Подставив (8.6) в уравнение колебаний струны (8.3), получим следующее соотношение:

$$\frac{T''(t)}{T(t)} = a^2 \frac{X''(x)}{X(x)}. \quad (8.7)$$

Выражение слева от знака равенства является функцией только от переменной t , а выражение справа – функцией только от переменной x . Это возможно только в том случае, если выражение слева и справа от знака равенства является константой, обозначим эту константу буквой λ . В этом случае выражение (8.7) можно записать в виде:

$$\frac{T''(t)}{T(t)} = a^2 \frac{X''(x)}{X(x)} = \lambda. \quad (8.8)$$

Из (8.8) получаются два обыкновенных дифференциальных уравнения:

$$X''(x) - \lambda X(x) = 0, \quad (8.9)$$

$$T''(t) - \lambda a^2 T(t) = 0. \quad (8.10)$$

Нас интересуют только нетривиальные решения уравнений (8.9) и (8.10) (не эквивалентные нулю). Для решения обыкновенного дифференциального уравнения не хватает граничных условий. Для их получения подставим соотношение (8.6) в граничные условия (8.5). В результате получим:

$$X(0)T(t) = 0, \quad (8.11)$$

$$X(L)T(t) = 0. \quad (8.12)$$

В связи с тем, что $T(t) \neq 0$, из (8.11)–(8.12) следует, что:

$$X(0) = X(L) = 0. \quad (8.13)$$

Таким образом, получили уравнение (8.9) с граничными условиями (8.13). Задача (8.9), (8.13) называется задачей Штурма–Лиувилля и состоит в том, чтобы определить значения λ , при которых существуют нетривиальные решения задачи (8.9), (8.13), и найти эти решения. Найденные значения λ называются *собственными значениями*, а соответствующие им функции – *собственными функциями*.

Рассмотрим решение задачи (8.9), (8.13) при различных значениях λ .

1. Если $\lambda > 0$, решение уравнения (8.9) можно записать в виде:

$$X(x) = C_1 e^{\sqrt{\lambda}x} + C_2 e^{-\sqrt{\lambda}x}.$$

Подставим граничные условия (8.13) в это соотношение, в результате чего получим:

$$X(0) = C_1 + C_2 = 0,$$

$$X(l) = C_1 e^{\sqrt{\lambda} L} + C_2 e^{-\sqrt{\lambda} L} = 0.$$

Откуда следует, что в этом случае $C_1 = C_2 = 0$. Следовательно, при $\lambda > 0$ задача (8.9), (8.13) имеет только тривиальное решение $X(x) \equiv 0$.

2. Если $\lambda = 0$, решение уравнения (8.9) $X(x) = C_1 x + C_2$ удовлетворяет граничным условиям (8.13) тоже только в случае $C_1 = C_2 = 0$. Опять получаем тривиальное решение.

3. Если $\lambda < 0$, решение уравнения (8.9) можно записать в виде:

$$X(x) = B_1 \sin(\sqrt{-\lambda} x) + B_2 \cos(\sqrt{-\lambda} x).$$

Подставив в это соотношения граничные условия (8.13), получим:

$$X(0) = B_2 = 0,$$

$$B_1 \sin(\sqrt{-\lambda} L) = 0.$$

Из последнего уравнения следует, что существует нетривиальное решение $X(x)$ ($B_2 \neq 0$) в случае, если

$$\sin(\sqrt{-\lambda} L) = 0.$$

Отсюда $\sqrt{-\lambda} = \frac{\pi n}{L}$. Следовательно, только при

$$\lambda = \lambda_n = -\left(\frac{\pi n}{L}\right)^2 \quad (8.14)$$

существует нетривиальное решение

$$X_n(x) = B_n \sin\left(\frac{\pi n}{L} x\right). \quad (8.15)$$

Значение λ_n , которое рассчитывается по формуле (8.14), является собственным значением, а соответствующая ему функция $X_n(x)$ (см. формулу (8.15)) — собственной функцией задачи Штурма–Лиувилля. Коэффициенты B_n без потери общности в функции $X_n(x)$ примем равными 1.

Таким образом, дальнейшее решение задачи (8.3)–(8.5) будем рассматривать только при $\lambda_n = -\left(\frac{\pi n}{L}\right)^2$. Поставим (8.14) в уравнение (8.10), которое примет вид:

$$T''(t) + \frac{\pi n}{L} a^2 T(t) = 0, \quad (8.16)$$

решение которого можно записать следующим образом:

$$T_n(t) = C_n \cos\left(\frac{\pi n}{L} a t\right) + D_n \sin\left(\frac{\pi n}{L} a t\right). \quad (8.17)$$

Подставим найденные $X_n(x)$ и $T_n(t)$ в (8.6) и найдем семейство частных решений задачи (8.3)–(8.5):

$$u_n(x, t) = \left(C_n \cos\left(\frac{\pi n}{L} a t\right) + D_n \sin\left(\frac{\pi n}{L} a t\right) \right) \sin\left(\frac{\pi n}{L} x\right). \quad (8.18)$$

Воспользовавшись принципом суперпозиции, выпишем решение, удовлетворяющее уравнению (8.3) и граничным условиям:

$$u(x, t) = \sum_{n=1}^{\infty} u_n(x, t) = \sum_{n=1}^{\infty} \left(C_n \cos\left(\frac{\pi n}{L} a t\right) + D_n \sin\left(\frac{\pi n}{L} a t\right) \right) \sin\left(\frac{\pi n}{L} x\right). \quad (8.19)$$

С помощью коэффициентов C_n и D_n необходимо удовлетворить начальным условиям. Для нахождения C_n и D_n подставим (8.19) в (8.5) и получим:

$$\begin{aligned} u(x, 0) &= \sum_{n=1}^{\infty} C_n \sin\left(\frac{\pi n}{L} x\right) = \mu(x), \\ \frac{\partial u(x, 0)}{\partial t} &= \sum_{n=1}^{\infty} a \frac{\pi n}{L} D_n \sin\left(\frac{\pi n}{L} x\right) = v(x). \end{aligned} \quad (8.20)$$

Из теории рядов Фурье [6, 21] найдем коэффициенты C_n и D_n :

$$\begin{aligned} C_n &= \frac{2}{L} \int_0^L \mu(z) \sin\left(\frac{\pi n}{L} z\right) dz, \\ D_n &= \frac{2}{\pi n a} \int_0^L v(z) \sin\left(\frac{\pi n}{L} z\right) dz. \end{aligned} \quad (8.21)$$

Таким образом, решением задачи (8.3)–(8.5) является функция $u(x, t)$, представляемая в виде бесконечного ряда (8.19), коэффициенты которого рассчитываются с помощью формул (8.21).

Рассмотрим решение неоднородного уравнения вида:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \frac{\partial^2 u}{\partial x^2} + g(x, t) \quad (8.22)$$

с однородными граничными условиями:

$$u(0, t) = 0, \quad u(L, t) = 0 \quad (8.23)$$

¹ Как видно из (8.6), (8.15), (8.17), если $B_n \neq 1$ в формуле (8.16), то формула (8.18) для вычисления $u_n(x, t)$ будет такой же.

и начальными условиями вида:

$$u(x, 0) = \mu(x), \quad \frac{\partial u(x, 0)}{\partial t} = v(x). \quad (8.24)$$

В связи с тем, что решение однородного уравнения (8.3) представимо в виде ряда $u(x, t) = \sum_{n=1}^{\infty} u_n(t) \sin\left(\frac{\pi n}{L} x\right)$ (см. формулу (8.19)), то и решение соответствующего неоднородного уравнения будем искать в виде аналогичного ряда:

$$u(x, t) = \sum_{n=1}^{\infty} u_n(t) \sin\left(\frac{\pi n}{L} x\right). \quad (8.25)$$

Отличие в том, что функции $u_n(t)$ для однородного и неоднородного уравнения разные.

Рассмотрим процесс нахождения функций $u_n(t)$ для неоднородного уравнения. Для этого запишем разложение функции $g(x, t)$ в ряд Фурье:

$$g(x, t) = \sum_{n=1}^{\infty} g_n(t) \sin\left(\frac{\pi n}{L} x\right); \quad g_n(t) = \frac{2}{L} \int_0^L g(z, t) \sin\left(\frac{\pi n}{L} z\right) dz. \quad (8.26)$$

Здесь $\sin\left(\frac{\pi n}{L} x\right)$ — собственная функция однородного уравнения (8.3). Функция $g(x, t)$ удовлетворяет граничным условиям. Начальные условия будут удовлетворены при нахождении $u_n(t)$. Подставив (8.25) в (8.22), получим соотношение:

$$\sum_{n=1}^{\infty} \sin\left(\frac{\pi n}{L} x\right) \left(-a^2 \left(\frac{\pi n}{L}\right)^2 u_n(t) - u_n''(t) + g_n(t) \right) = 0, \quad (8.27)$$

из которого следуют обыкновенные дифференциальные уравнения для нахождения функций $u_n(t)$

$$u_n''(t) + a^2 \left(\frac{\pi n}{L}\right)^2 u_n(t) - g_n(t) = 0. \quad (8.28)$$

Для записи начальных условий разложим функции $\mu(x)$ и $v(x)$ в ряд Фурье:

$$\begin{aligned} u(x, 0) &= \sum_{n=1}^{\infty} u_n(0) \sin\left(\frac{\pi n}{L} x\right) = \sum_{n=1}^{\infty} \mu_n \sin\left(\frac{\pi n}{L} x\right) = \mu(x), \\ \mu_n &= \frac{2}{L} \int_0^L \mu(z) \sin\left(\frac{\pi n}{L} z\right) dz, \\ \frac{\partial u(x, 0)}{\partial t} &= \sum_{n=1}^{\infty} u_n'(0) \sin\left(\frac{\pi n}{L} x\right) = \sum_{n=1}^{\infty} v_n \sin\left(\frac{\pi n}{L} x\right) = v(x), \\ v_n &= \frac{2}{L} \int_0^L v(z) \sin\left(\frac{\pi n}{L} z\right) dz. \end{aligned} \quad (8.29)$$

Отсюда следует, что

$$u_n(0) = \mu_n, u'_n(0) = \nu_n. \quad (8.30)$$

Решение задачи (8.22)–(8.24) записывается в виде (8.25), коэффициенты $u_n(t)$ являются решением обыкновенного дифференциального уравнения (8.28), (8.30).

Полученное в нашем случае решение, как и при решении задачи (8.3)–(8.5), является суммой бесконечного ряда довольно сложной структуры, и нахождение численного значения функции в конкретной точке представляет собой отдельную математическую задачу. Поэтому широкое распространение получили численные методы решения уравнений в частных производных.

8.3. Использование метода сеток для решения уравнения в частных производных и его реализация в Mathcad

Одним из наиболее распространенных численных методов решения уравнений является *метод сеток*¹. В методе сеток область Ω , в которой необходимо найти решение уравнения, прямыми, параллельными осям $t = t_j$ и $x = x_i$, разобьем на прямоугольные области (см. рис. 8.1), где $x_i = x_0 + ih$, $h = \frac{x_n - x_0}{n}$, $i = 0, 1, 2, \dots, n$,

$t_j = t_0 + j\Delta$, $\Delta = \frac{t_k - t_0}{k}$, $j = 0, 1, \dots, k$. Точки, которые лежат на границе Γ области Ω , называются *внешними*, остальные точки – *внутренними*. Совокупность всех точек называется сеткой Ω_h^A , величины h и Δ – шагами сетки по x и t соответственно.

Идея метода сеток состоит в том, что вместо любой непрерывной функции $w(x, t)$ будем рассматривать дискретную функцию $w_i^j = w(x_i, t_j)$, которая определена в узлах сетки Ω_h^A . Вместо производных функции будем рассматривать их простейшие разностные аппроксимации в узлах сетки. Таким образом, вместо системы дифференциальных уравнений в частных производных получим систему алгебраических уравнений. Чем меньше величины h и Δ , тем точнее получаемые алгебраические уравнения моделируют исходное дифференциальное уравнение в частных производных. В последующих разделах этой главы будет

¹ Метод сеток в литературе также называют методом конечных разностей.

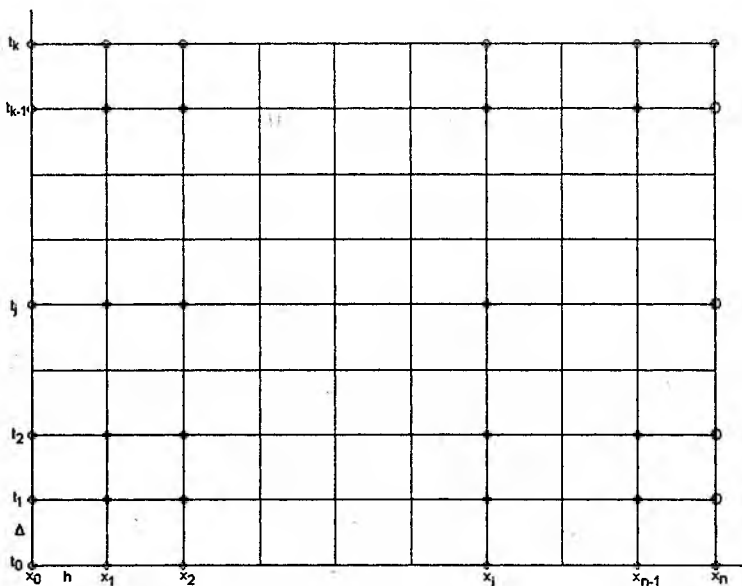


Рис. 8.1 ▼ Сетка Ω_h^Δ для области Ω с границей Γ

рассмотрен метод сеток для каждого из трех типов уравнений и его реализация в Mathcad.

8.3.1. Использование метода сеток для решения параболических уравнений в Mathcad

Знакомство с численными методами решения дифференциальных уравнений в частных производных начнем с разностных схем решения параболических уравнений [9]. Разностные схемы решения параболических уравнений будем рассматривать на примере следующего одномерного уравнения:

$$\begin{aligned} \frac{\partial u}{\partial t} &= a^2 \frac{\partial^2 u}{\partial x^2} + f(x, t), \quad 0 \leq x \leq L, \quad 0 \leq t \leq T, \\ u(0, t) &= \mu(t), \quad u(L, t) = \eta(t), \quad 0 \leq t \leq T, \\ u(x, 0) &= \varphi(x), \quad 0 \leq x \leq L. \end{aligned} \quad (8.31)$$

Построим сетку Ω_h^Δ (см. рис. 8.1). Для получения сеточного уравнения заменим производную $\frac{\partial^2 u}{\partial x^2}$ приближенной разностной формулой [9]:

$$\frac{\partial^2 u(x_i, t_j)}{\partial x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}. \quad (8.32)$$

Для замены $\frac{\partial u}{\partial t}$ можно воспользоваться одной из приближенных разностных формул:

$$\frac{\partial u(x_i, t_j)}{\partial t} = \frac{u_{i,j+1} - u_{i,j}}{\Delta}, \quad (8.33)$$

$$\frac{\partial u(x_i, t_j)}{\partial t} = \frac{u_{i,j} - u_{i,j-1}}{\Delta}. \quad (8.34)$$

Кроме того, заменим начальные и граничные условия их разностной аппроксимацией:

$$u_{i,0} = \varphi(x_i) = \varphi_i, \quad i = 0, 1, \dots, n, \quad (8.35)$$

$$u_{0,j} = \mu(t_j) = \mu_j, \quad u_{n,j} = \eta(t_j) = \eta_j, \quad j = 0, 1, \dots, k. \quad (8.36)$$

Заменяя частные производные в задаче (8.31) соотношениями (8.32) и (8.33) и учитывая условия (8.35)–(8.36), получим следующую вычислительную схему для расчета значений функции u в узлах сетки Ω_h^A :

$$u_{i,j+1} = \gamma u_{i,j-1} + (1 - 2\gamma)u_{i,j} + \gamma u_{i,j+1} + \Delta \cdot f_{i,j}, \quad (8.37)$$

$$u_{0,j} = \mu_j, \quad u_{n,j} = \eta_j, \quad u_{i,0} = \varphi_i, \quad \gamma = \frac{a^2 \Delta}{h^2}. \quad (8.38)$$

Это явная двухслойная разностная схема (рис. 8.2). Учитывая, что на нулевом слое (при $i = 0$) все значения $u_{i,0}$, (как, впрочем, и $u_{0,j}$, $u_{n,j}$) известны, по формуле (8.37) можно сначала явно рассчитать значения $u_{i,1}$, затем $u_{i,2}$ и так до $u_{i,k}$. Для устойчивости разностной схемы (8.37) значения шагов по t и x должны удовлетворять следующему условию:

$$\Delta \leq \frac{h^2}{2a^2}. \quad (8.39)$$

Блок-схема решения уравнения (8.31) с помощью явной разностной схемы (8.37)–(8.38) представлена на рис. 8.3.

¹ Здесь и далее $u_{i,j}$ — это значение функции u в точке (x_i, t_j) , $u_{i+1,j}$ — в точке (x_{i+1}, t_j) , $u_{i-1,j}$ — в точке (x_{i-1}, t_j) , $u_{i,j+1}$ — в точке (x_i, t_{j+1}) и $u_{i,j-1}$ — в точке (x_i, t_{j-1}) .

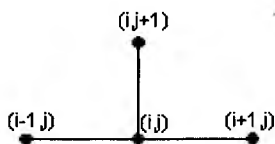


Рис. 8.2 ▼ Шаблон явной двухслойной разностной схемы

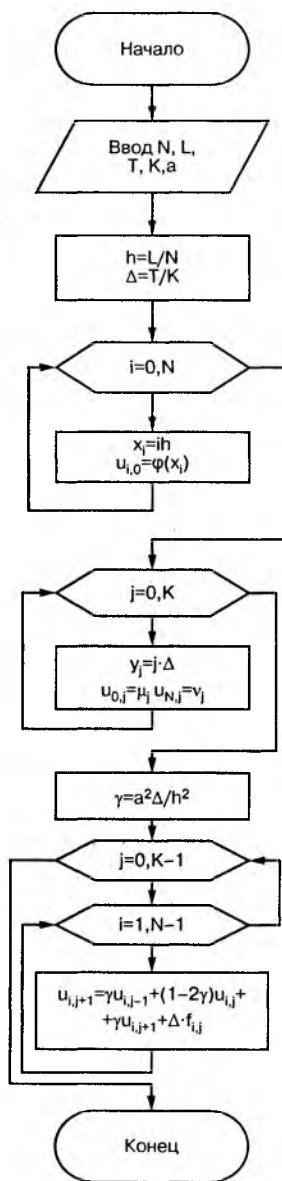


Рис. 8.3 ▼ Блок-схема решения уравнения (8.31) с помощью явной разностной схемы

Рассмотрим решение параболического уравнения в Mathcad на примере следующей задачи теплопроводности [7, 9].

ЗАДАЧА 8.1. Найти распределение температуры в стержне длиной L ($0 \leq x \leq L$), начальная температура стержня задается произвольной функцией $\phi(x)$. Температуры концов стержня равны $u(0, t) = U_1 = \text{const}$, $u(L, t) = U_2 = \text{const}$.

Начально-граничная задача, описывающая распределение температуры стержня, имеет вид:

$$\begin{aligned} \frac{\partial u}{\partial t} &= a^2 \frac{\partial^2 u}{\partial x^2}, \quad a^2 = \frac{\lambda}{c\rho}, \quad 0 < x \leq L, \quad 0 < t < \infty, \\ u(0, t) &= U_1, \quad u(L, t) = U_2, \quad 0 < t < \infty, \\ u(x, 0) &= \phi(x), \quad 0 < x < L. \end{aligned} \quad (8.40)$$

$$f(x, t) := 0 \quad \phi(x) := e^{0.15x} \quad \mu(t) := 1 \quad v(t) := 2.117$$

$$\begin{aligned} \text{parabol}(N, K, L, T, a) := & \left(\begin{array}{l} h \leftarrow \frac{L}{N} \\ \Delta \leftarrow \frac{T}{K} \\ \text{for } i \in 0..N \\ \quad x_i \leftarrow i \cdot h \\ \text{for } j \in 0..K \\ \quad t_j \leftarrow j \cdot \Delta \\ \gamma \leftarrow a^2 \cdot \frac{\Delta}{h^2} \\ \text{for } i \in 0..N \\ \quad u_{i,0} \leftarrow \phi(x_i) \\ \text{for } j \in 0..K \\ \quad \left(\begin{array}{l} u_{0,j} \leftarrow \mu(t_j) \\ u_{N,j} \leftarrow v(t_j) \end{array} \right) \\ \text{for } j \in 0..K-1 \\ \quad \text{for } i \in 1..N-1 \\ \quad \quad u_{i,j+1} \leftarrow \gamma \cdot u_{i-1,j} + (1-2\gamma) \cdot u_{i,j} + \gamma \cdot u_{i+1,j} + \Delta \cdot f(x_i, t_j) \end{array} \right) \\ & \left(\begin{array}{c} u \\ x \\ t \end{array} \right) \end{aligned}$$

Рис. 8.4 ▾ Подпрограмма на Mathcad решения задачи 8.1 с помощью явной разностной схемы

Здесь a^2 – коэффициент температуропроводности, λ – коэффициент теплопроводности материала стержня, c – удельная теплоемкость, ρ – плотность массы.

Подпрограмма решения задачи 8.1 с помощью явной разностной схемы в Mathcad представлена на рис. 8.4.

Входными данными подпрограммы `parabol` решения задачи 8.1 являются: N – количество интервалов, на которые разбивается отрезок $(0, L)$; K – количество интервалов, на которые разбивается отрезок $(0, T)$; L – длина стержня, T – интервал времени, a – параметр дифференциального уравнения. Функция возвращает три параметра: решение – сеточная функция u , определенная на сетке Ω_h^A , массивы x и t .

Результаты решения задачи 8.1 представлены на рис. 8.5. На рис. 8.6 показаны графики сеточной функции u и график точного решения.

$\underline{N} := \text{parabol}(50, 200, 5, 3, 0.4) \quad v := H_0 \quad x := H_1 \quad t := H_2$

	0	1	2	3	4	5	6
0	1	1	1	1	1	1	1
1	1.01511	1.01517	1.01521	1.01524	1.01528	1.0153	1.01533
2	1.03045	1.03051	1.03057	1.03062	1.03067	1.03071	1.03076
3	1.04603	1.04608	1.04614	1.0462	1.04625	1.04631	1.04636
4	1.06184	1.06189	1.06195	1.06201	1.06207	1.06212	1.06218
5	1.07788	1.07794	1.078	1.07806	1.07812	1.07818	1.07823
6	1.09417	1.09423	1.09429	1.09435	1.09441	1.09447	1.09453
7	1.11071	1.11077	1.11083	1.11089	1.11095	1.11101	1.11107
8	1.1275	1.12756	1.12762	1.12768	1.12774	1.1278	1.12786
9	1.14454	1.1446	1.14466	1.14472	1.14478	1.14485	1.14491
10	1.16183	1.1619	1.16196	1.16202	1.16209	1.16215	1.16221
11	1.17939	1.17946	1.17952	1.17958	1.17965	1.17971	1.17978
12	1.19722	1.19728	1.19735	1.19741	1.19748	1.19754	1.19761
13	1.21531	1.21538	1.21544	1.21551	1.21557	1.21564	1.2157
14	1.23368	1.23374	1.23381	1.23388	1.23394	1.23401	1.23408
15	1.25232	1.25239	1.25246	1.25253	1.25259	1.25266	1.25273

Рис. 8.5 ▼ Решение задачи 8.1

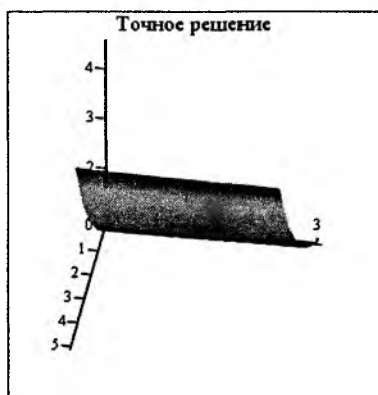
Приведенная на рис. 8.4 программа позволяет решать уравнения вида (8.40) с помощью явной разностной схемы. Рекомендуем читателю поэкспериментировать с этой программой, попробовав решить параболическое уравнение

$\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + \sin(x \cdot t)$. График решения этого уравнений приведен на рис. 8.7.

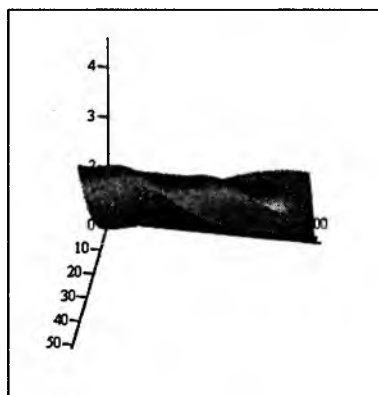
При решении параболических уравнений с помощью явной разностной схемы основной проблемой является устойчивость решения и правильный выбор шага по t , удовлетворяющего соотношению (8.39). Для решения этой проблемы



v



u

Рис. 8.6 ▼ Графики найденной точечной функции u и точного решения задачи 8.1

v

Рис. 8.7 ▼ График решения уравнения $\frac{\partial u}{\partial t} = a^2 \frac{\partial^2 u}{\partial x^2} + \sin(x \cdot t)$

были предложены неявные разностные схемы. Эти схемы абсолютно устойчивы, но алгоритм решения получаемого сеточного уравнения несколько сложнее, чем простой пересчет по формуле (8.37). Для построения неявной разностной схемы заменим частные производные в задаче (8.40) соотношениями (8.32), (8.34) (вместо (8.33) для явной схемы) и с учетом условий (8.35)–(8.36) получим следующую вычислительную схему для расчета значений функции u в узлах сетки Ω_h^A :

$$\gamma u_{i-1,j} + (1 + 2\gamma) u_{i,j} + \gamma u_{i+1,j} = -u_{i,j-1} - \Delta \cdot f_{i,j}, \quad (8.41)$$

$$i = 1, 2, \dots, n-1, j = 1, 2, \dots, k.$$

Соотношения (8.41) вместе с равенствами (8.37) – неявная двухслойная разностная схема (рис. 8.8).

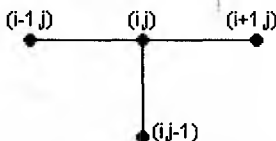


Рис. 8.8 ▼ Шаблон неявной двухслойной разностной схемы

Схема (8.38), (8.41) не позволяет явно выписать решение, и для нахождения u_{ij} при каждом значении j необходимо решить трехдиагональную систему линейных алгебраических уравнений, для чего можно воспользоваться одним из итерационных методов или методом прогонки (см. главу 1). Преобразуем систему (8.41) к следующему виду:

$$u_{i,j} = \frac{\gamma}{1 + 2\gamma} (u_{i-1,j} + u_{i+1,j}) + \frac{u_{i,j-1}}{1 + 2\gamma} + \frac{\Delta}{1 + 2\gamma} f(x_i, t_j). \quad (8.42)$$

Формула (8.42) позволит запрограммировать решение системы, получаемой с помощью неявной разностной схемы методом Зейделя.

На рис. 8.9 представлена блок-схема решения задачи 8.1 с помощью неявной разностной схемы, в которой система уравнений (8.42) решается методом Зейделя (см. блоки 10–14 блок-схемы)¹.

На рис. 8.10–8.11 представлены реализация блок-схемы в виде подпрограммы Mathcad и результаты решения задачи 8.1. Входными данными подпрограммы Ne_yavn являются: коэффициент γ ; величина шага Δ ; матрица решений u , в которой хранятся начальные и граничные условия; точность решения системы уравнений (8.42) ϵ ; массивы x и t . Подпрограмма Ne_yavn возвращает матрицу решений u задачи 8.1, матрицу ошибок R , количество итераций k .

Обратите внимание, что подпрограммы, приведенные на рис. 8.4 и 8.10, позволяют решать задачи вида 8.3 с различными функциями $f(x, t)$, $\mu(t)$, $\eta(t)$, $\varphi(x)$.

Использование неявной разностной схемы в случае, когда $f(x, t) \neq 0$, рассмотрим на примере еще одной задачи теплопроводности [7].

¹ Здесь R – матрица невязки, в переменной k хранится количество итераций при решении системы (8.42) методом Зейделя.

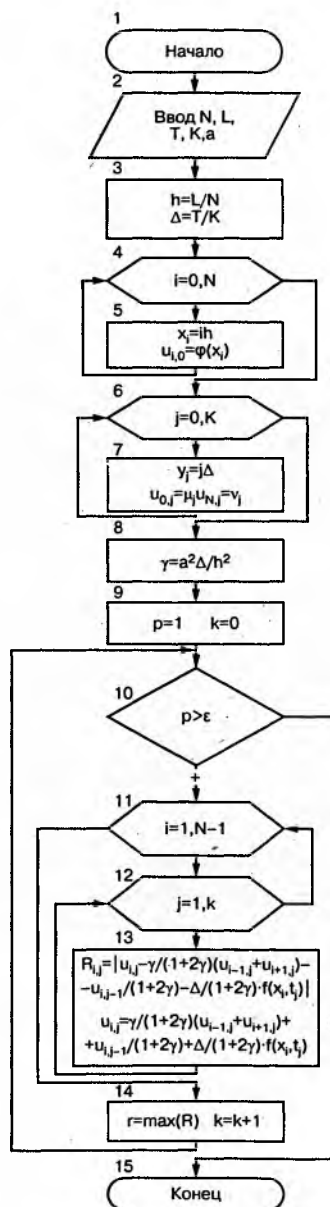


Рис. 8.9 ▼ Блок-схема решения задачи 8.1 с помощью неявной разностной схемы

```

L := 5      T := 3      N := 50      K := 200
f(x,t) := 0      φ(x) := e0.15 x      μ(t) := 1      v(t) := 2.117
h := L/N      Δ := T/K      γ := a2·Δ/h2      i := 0..N      j := 0..K
ij := 0..N
Ui,0 := φ(xi)
U0,j := μ(tj)      UN,j := v(tj)
Nc_yavn(U,γ,Δ,ε,x,t) :=
  p ← 1
  k ← 0
  while p > ε
    for i ∈ 1..N-1
      for j ∈ 1..K
        V ←  $\frac{\gamma}{1+2\gamma} \cdot (U_{i-1,j} + U_{i+1,j}) + \frac{U_{i,j-1}}{1+2\gamma} + \frac{\Delta}{1+2\gamma} \cdot f(x_i, t_j)$ 
        Ri,j ← |V - Ui,j|
        Ui,j ← V
      p ← max(R)
    k ← k + 1
  (
    U
    R
    k
  )

```

Рис. 8.10 ▼ Подпрограмма на Mathcad решения задачи 8.1 с помощью неявной разностной схемы

ЗАДАЧА 8.2. Найти распределение температуры в стержне длиной L ($0 \leq x \leq L$), если его начальная температура задается произвольной функцией $f(x)$. Температуры концов стержня равны $u(0, t) = U_1 = \text{const}$, $u(L, t) = U_2 = \text{const}$. На боковой поверхности стержня происходит теплообмен по закону Ньютона со средой, температура которой равна u_0 .

Начально-граничная задача, описывающая распределение температуры стержня, имеет вид:

$$\begin{aligned}
 \frac{\partial u}{\partial t} &= a^2 \frac{\partial^2 u}{\partial x^2} - h(u - u_0), \quad a^2 = \frac{\lambda}{c\rho}, \quad h = \frac{\alpha p}{c\rho\sigma}, \quad 0 \leq x \leq L, \quad 0 < t < \infty, \\
 u(0, t) &= U_1, \quad u(L, t) = U_2, \quad 0 < t < \infty, \\
 u(x, 0) &= \varphi(x), \quad 0 < x < L.
 \end{aligned} \tag{8.43}$$

$$\underline{H} := \text{Ne\_yavn}(U, \gamma, \Delta, 0.001, x, t) \quad \underline{U} := H_0 \quad \underline{R} := H_1 \quad k := H_2 \quad k = 74$$

$U =$

	0	1	2	3	4	5
0	1	1	1	1	1	1
1	1.015113	1.015159	1.015198	1.015233	1.015264	1.015292
2	1.030455	1.030509	1.030561	1.03061	1.030658	1.030703
3	1.046028	1.046084	1.04614	1.046195	1.046249	1.046302
4	1.061837	1.061894	1.061951	1.062008	1.062065	1.062121
5	1.077884	1.077942	1.078001	1.078059	1.078117	1.078175
6	1.094174	1.094233	1.094292	1.094352	1.094411	1.09447
7	1.110711	1.110771	1.110831	1.110891	1.110951	1.111011
8	1.127497	1.127558	1.127619	1.12768	1.12774	1.127801
9	1.144537	1.144599	1.14466	1.144722	1.144784	1.144846
10	1.161834	1.161897	1.16196	1.162022	1.162085	1.162148
11	1.179393	1.179457	1.179521	1.179584	1.179648	1.179712
12	1.197217	1.197282	1.197347	1.197411	1.197476	1.197541
13	1.215311	1.215377	1.215442	1.215508	1.215574	1.215639
14	1.233678	1.233745	1.233811	1.233878	1.233945	1.234011

Рис. 8.11 ▼ Результаты решения задачи 8.1 с помощью неявной разностной схемы¹

Здесь α – коэффициент теплообмена, σ – площадь поперечного сечения стержня, p – периметр поперечного сечения стержня.

Построим сетку Ω_{hx}^{Δ} ($x_i = i \cdot hx$, $hx = \frac{L}{n}$, $i = 0, 1, 2, \dots, n$, $t_j = j \cdot \Delta$, $\Delta = \frac{T}{k}$, $j = 0, 1, \dots, k$). Для получения сеточного уравнения заменим производные $\frac{\partial^2 u}{\partial x^2}$ и $\frac{\partial u}{\partial t}$ приближенными разностными формулами (8.32), (8.34). Получится следующая неявная разностная схема (8.44)–(8.46):

$$\begin{aligned} u_{i,0} &= \varphi(x_i), \quad i = 0, 1, \dots, N, \\ u_{0,j} &= U_1, \quad u_{N,j} = U_2, \quad j = 0, 1, \dots, K, \end{aligned} \quad (8.44)$$

$$\begin{aligned} u_{i,j} &= \\ &= \frac{1}{1 + 2\gamma + \Delta \cdot h} u_{i,j-1} + \frac{\gamma}{1 + 2\gamma + \Delta \cdot h} (u_{i-1,j} + u_{i+1,j}) + \frac{\Delta \cdot h}{1 + 2\gamma + \Delta \cdot h} u_0, \quad (8.45) \\ i &= 1, 2, \dots, N-1; j = 1, 2, \dots, k, \end{aligned}$$

¹ Читатель может самостоятельно сравнить результаты, полученные с помощью явной и

$$\gamma = a^2 \frac{\Delta}{hx^2}. \quad (8.46)$$

Блок-схема решения задачи 8.2 будет отличаться от блок-схемы, изображенной на рис. 8.9, только формулой неявной разностной схемы (8.45) для расчета $u_{i,j}$.

Применение неявной разностной схемы для решения задачи 8.2 методом Зейделя демонстрируется на рис. 8.12.

```

 $\phi(x) := 0.25 + \sin(0.15 \cdot x)$     $L_x := 8$     $T_x := 3$     $N_x := 50$     $K_x := 200$ 
 $hx := \frac{L}{N}$     $\Delta := \frac{T}{K}$     $\gamma := a^2 \cdot \frac{\Delta}{hx^2}$     $a = 5$ 
 $i := 0..N$     $j := 0..K$ 
 $x_i := i \cdot hx$     $t_j := j \cdot \Delta$ 

 $u_{0,0} := \phi(x_i)$     $u_{0,j} := U_1$     $u_{N,j} := U_2$ 

Ne_iav(u,K,N, $\gamma$ , $\Delta$ ,h, $\epsilon$ ) :=
  p  $\leftarrow$  1
  k  $\leftarrow$  0
  while p >  $\epsilon$ 
    for j  $\in$  1..K
      for i  $\in$  1..N-1
         $V \leftarrow \left[ \frac{\gamma}{1 + 2\gamma + \Delta \cdot h} \cdot (u_{i-1,j} + u_{i+1,j}) + \frac{u_{i,j-1}}{1 + 2\gamma + \Delta \cdot h} \right] + \frac{\Delta \cdot h}{1 + 2\gamma + \Delta \cdot h} u_0$ 
         $R_{i,j} \leftarrow |V - u_{i,j}|$ 
         $u_{i,j} \leftarrow V$ 
      p  $\leftarrow$  max(R)
    k  $\leftarrow$  k + 1
  (
    u
    R
    k
  )

```

Рис. 8.12 ▼ Решение задачи 8.2 с помощью неявной разностной схемы

Входные и выходные данные подпрограммы Ne_iav решения задачи 8.2 аналогичны подпрограмме Ne_yavn, рассмотренной выше (см. рис. 8.10). На рис. 8.13 представлены результаты решения задачи.

Как видно из рис. 8.13, для решения получаемого алгебраического уравнения методом Зейделя потребовалось 1079 итераций. Поэтому для уменьшения количества итераций имеет смысл попробовать ускорить итерационный процесс с помощью методов релаксации или градиентных методов решения систем алгебраических уравнений (см. главу 1). Методика, изложенная в этом разделе,

TT := time(0) H := Ne_iav(u, K, N, γ, Δ, h, 0.0001) TTT := time(1)

Время решения задачи

TTT - TT = 83.75

H₂ = 898

H₀ =

	0	1	2	3	4	5	6	7
0	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
1	0.274	0.283	0.288	0.291	0.294	0.296	0.298	0.299
2	0.298	0.313	0.323	0.329	0.335	0.339	0.343	0.346
3	0.322	0.342	0.356	0.365	0.373	0.38	0.386	0.391
4	0.346	0.37	0.387	0.399	0.41	0.418	0.426	0.433
5	0.37	0.397	0.416	0.431	0.444	0.455	0.464	0.472
6	0.394	0.422	0.444	0.462	0.477	0.489	0.5	0.51
7	0.417	0.448	0.472	0.491	0.508	0.522	0.535	0.546
8	0.441	0.472	0.498	0.519	0.538	0.554	0.568	0.58
9	0.464	0.496	0.523	0.546	0.566	0.584	0.599	0.613
10	0.488	0.52	0.548	0.572	0.593	0.612	0.629	0.644

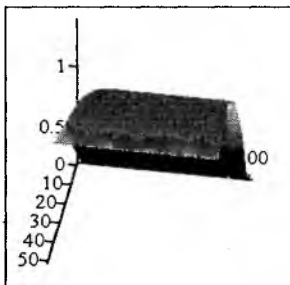


Рис. 8.13 ▼ Результаты решения задачи 8.2

может быть использована и при решении в Mathcad других параболических уравнений.

8.3.2. Использование метода сеток для решения гиперболических уравнений в Mathcad

Решение гиперболических уравнений также можно осуществить с помощью разностных схем. Разностные схемы решения одномерного гиперболического уравнения рассмотрим на примере следующего уравнения:

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= a^2 \frac{\partial^2 u}{\partial x^2} + f(x, t), \\ u(x, 0) &= \varphi(x), \quad \frac{\partial u(x, 0)}{\partial t} = \psi(x), \quad 0 \leq x \leq L, \\ u(0, t) &= \mu(t), \quad u(L, t) = v(t), \quad 0 \leq t \leq T. \end{aligned} \quad (8.47)$$

Построим сетку Ω_h^Δ (см. рис. 8.1), в которой будем искать решение уравнения (8.47). Частную производную $\frac{\partial^2 u}{\partial x^2}$ заменим разностным соотношением (8.32), а производную $\frac{\partial^2 u}{\partial t^2}$ – соотношением (8.48):

$$\frac{\partial^2 u(x_i, t_j)}{\partial t^2} = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta^2}. \quad (8.48)$$

Подставляя (8.48), (8.32), (8.33) в гранично-начальную задачу (8.47), получим следующую явную разностную схему решения уравнения:

$$\begin{aligned} u_{i,j+1} &= -u_{i,j-1} + \gamma(u_{i-1,j} + u_{i+1,j}) + (2 - 2\gamma)u_{i,j} + \Delta^2 f_{i,j}, \\ i &= 1, 2, \dots, N-1, j = 1, 2, \dots, K-1, \\ u_{i,0} &= \varphi(x_i), \quad \frac{u_{i,1} - u_{i,0}}{\Delta} = \psi_i, \quad u_{i,1} = u_{i,0} + \Delta\psi_i, \quad i = 0, 1, \dots, N, \\ u_{0,j} &= \mu_j, \quad u_{N,j} = \nu_j, \quad j = 0, 1, \dots, K, \\ \gamma &= \frac{a^2 \Delta^2}{h^2}, \end{aligned} \quad (8.49)$$

которая устойчива при $\gamma \leq 1$ и по аналогии с разностной схемой (8.37)–(8.38) может быть легко запрограммирована в Mathcad. Блок-схема этой разностной схемы аналогична изображенной на рис. 8.3 и отличается только формулой расчета $u_{i,j+1}$ (см. формулу (8.49)).

В качестве примера рассмотрим следующую начально-граничную задачу [7].

ЗАДАЧА 8.3.

$$\begin{aligned} \frac{\partial^2 \omega}{\partial t^2} &= a^2 \frac{\partial^2 \omega}{\partial x^2} + \sin(x \cdot t), \quad 0 < x < L, \quad t > 0, \\ \omega(x, 0) &= \varphi(x), \quad \omega(L, t) = \varphi(L), \\ \omega_t(x, 0) &= \psi(x). \end{aligned} \quad (8.50)$$

На рис. 8.14 представлена подпрограмма `ggg` решения уравнения (8.50), а на рис. 8.15 – результаты обращения к подпрограмме и график полученного решения. Параметры функции `ggg` аналогичны рассмотренным ранее подпрограммам решения параболических уравнений.

$\phi(x) := \sin(x)$ $\psi(x) := \cos(x)$ $v(t) := 0$ $f(x,t) := \sin(x \cdot t)$
 $\underline{a} := 4$ $\underline{T} := 2$ $\underline{A} := 3$ $\underline{\gamma} := 5$ $\underline{L} := 10$ $\underline{N} := 50$ $\underline{K} := 200$

```

ggg(N,K,L,T,a) :=
    h ← L / N
    Δ ← T / K
    for i ∈ 0..N
        | xi ← i·h
        | ui,0 ← φ(xi)
        | ui,1 ← ui,0 + Δ·ψ(xi)
    for j ∈ 0..K
        tj ← Δ·j
    for j ∈ 1..K
        | u0,j ← 0
        | uN,j ← φ(L)
        γ ← a · 2 · Δ2 / h2
        for j ∈ 1..K-1
            for i ∈ 1..N-1
                ui,j+1 ← -ui,j-1 + γ·ui-1,j + (2-2·γ)ui,j + γ·ui+1,j + Δ2·f(xi,tj)
    u

```

Рис. 8.14 ▼ Подпрограмма решения задачи 8.3

Для решения гиперболических уравнений можно построить и неявные схемы, однако в связи с тем, что у них нет таких преимуществ перед явными [9], как в параболических уравнениях, имеет смысл использовать их, только когда нельзя построить явную (например, при смешанных условиях на границе области).

$$V := \text{ggg}(N, 400, L, T, a)$$

V =

	0	1	2
0	0	0	0
1	0.19867	0.20357	0.20834
2	0.38942	0.39402	0.39847
3	0.56464	0.56877	0.57267
4	0.71736	0.72084	0.72404
5	0.84147	0.84417	0.84654
6	0.93204	0.93385	0.93529
7	0.98545	0.9863	0.98676
8	0.99957	0.99943	0.99888

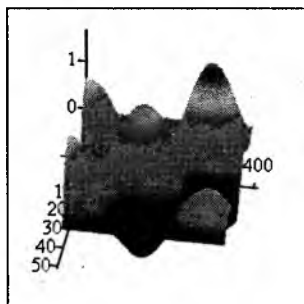


Рис. 8.15 ▼ Решение задачи 8.3

8.3.3. Использование метода сеток для решения эллиптических уравнений в Mathcad

Рассмотрим разностную схему для эллиптического уравнения в прямоугольной области $\Omega (R - a \leq x \leq R + a; -a \leq y \leq a)$ с граничными условиями Дирихле на границе Γ :

$$\Delta u = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} - \frac{5}{x} \frac{\partial \psi}{\partial x} = -2, \quad \psi(x, y)|_{(x, y) \in \Gamma} = 0. \quad (8.51)$$

Построим сетку Ω_{hx}^{hy} , для чего проведем в области Ω прямые, параллельные осям $y = y_j$ и $x = x_i$, где $x_i = R - b + i \cdot hx$, $hx = \frac{2b}{n}$, $i = 0, 1, 2, \dots, n$, $y_j = -a + j \cdot hy$, $hy = \frac{2a}{k}$, $j = 0, 1, \dots, k$. Для построения разностного уравнения заменим частные производные и граничные условия следующими соотношениями:

$$\frac{\partial^2 \Psi(x_i, y_j)}{\partial x^2} = \frac{\Psi_{i+1,j} - 2\Psi_{i,j} + \Psi_{i-1,j}}{hx^2},$$

$$\frac{\partial^2 \Psi(x_i, y_j)}{\partial y^2} = \frac{\Psi_{i,j+1} - 2\Psi_{i,j} + \Psi_{i,j-1}}{hy^2}. \quad (8.52)$$

$$\Psi_{i,0} = \Psi_{i,Ny} = 0, \quad i = 0, 1, \dots, Nx,$$

$$\Psi_{0,j} = \Psi_{Nx,j} = 0, \quad j = 0, 1, \dots, Ny. \quad (8.53)$$

С помощью соотношений (8.52)–(8.53) преобразуем эллиптическую краевую задачу к следующей системе разностных уравнений:

$$\Psi_{i,j} = \frac{1}{A} \left(B_i \Psi_{i+1,j} + C_i \Psi_{i-1,j} + D(\Psi_{i,j-1} + \Psi_{i,j+1}) + 2 \right),$$

$$A = \left(\frac{2}{hx^2} + \frac{2}{hy^2} \right), \quad B_i = \frac{1}{hx^2} + \frac{5}{2 \cdot hx \cdot x_i}, \quad C_i = \frac{1}{hx^2} - \frac{5}{2 \cdot hx \cdot x_i}, \quad D = \frac{1}{hy^2},$$

$$i = 1, 2, \dots, Nx - 1; \quad j = 1, 2, \dots, Ny - 1,$$

$$\Psi_{i,0} = \Psi_{i,Ny} = 0, \quad i = 0, 1, \dots, Nx,$$

$$\Psi_{0,j} = \Psi_{Nx,j} = 0, \quad j = 0, 1, \dots, Ny. \quad (8.54)$$

Эту систему можно решить итерационным методом Зейделя¹. На рис. 8.16 представлено решение уравнения (8.51) сеточным методом, а на рис. 8.17 – график найденного решения.

Авторы надеются, что читатель, разобравшийся с решением уравнения (8.51), без проблем построит разностную схему для решений уравнения Лапласа и Пуассона.

¹ В случае медленной сходимости итерационных процессов при решении сеточных уравнений, получаемых путем аппроксимации гиперболических и эллиптических задач, имеет смысл попробовать заменить метод Зейделя градиентными методами (или методами релаксации) [9].

```

R:=18  a:=3  b:=6  Nx:=16  Ny:=8
i:=0..Nx      j:=0..Ny      hy:=2a/Ny  hx:=2b/Nx
xi:=R-b+i·hx  yj:=-a+j·hy
psi0,0:=0      psi0,Ny:=0      psi0,j:=0      psiNx,j:=0

A:=2/hy^2+2/hx^2      D:=1/hy^2

i:=1..Nx
Bi:=1/hx^2+5/(2·hx·xi)  Ci:=1/hx^2-5/(2·hx·xi)

ellip(ψ,Nx,Ny,ε):=
  p←100
  k←0
  while p>ε
    for i←1..Nx-1
      for j←1..Ny-1
        V←1/A·[Bi·psii-1,j+Ci·psii+1,j+D·(psii,j-1+psii,j+1)+2]
        Ri,j←|V-psii,j|
        psii,j←V
      p←max(R)
      k←k+1
  (ψ
   R
   k)
  
```

Рис. 8.16 ▽ Решение уравнения (8.51) сеточным методом

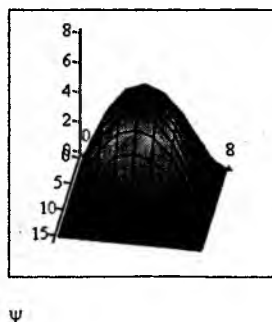


Рис. 8.17 ▽ График решения уравнения (8.51)

8.4. Использование встроенных функций Mathcad для решения уравнений в частных производных

В этом разделе рассмотрены встроенные функции Mathcad, предназначенные для решения уравнений в частных производных.

8.4.1. Решение параболических уравнений

В Mathcad 11 появились встроенные функции для решения одномерных дифференциальных уравнений в частных производных `pdsolve` и `num01`, применение которых рассмотрим на примере задачи 8.2.

Для решения параболического уравнения можно воспользоваться решающим блоком. Для решения параболического уравнения в Mathcad необходимо:

1. Ввести служебное слово **Given**.
2. Ввести уравнения следующим образом: производные вводятся с помощью символа «.», как нижние индексы (u_i или u_{xx}), вместо знака равенства следует использовать оператор = панели **Boolean** математической палитры (или комбинацию клавиш **Ctrl+=**).



В левой части уравнения должна присутствовать только первая производная по t , а все остальное уравнение вводится в правой части. При вводе уравнения в функции и производных должен присутствовать список аргументов, например $u(x, t)$, $u_{xx}(x, t)$, $u_i(x, t)$.

3. Ввести начальные и граничные условия, в которых производные вводятся как нижние индексы, а знак равенства вставлять, используя оператор = панели **Boolean**.
4. Завершить решающий блок функцией `pdsolve(u, x, xrange, t, trange, xpts, tpts)`, которая возвращает функцию двух переменных, являющуюся решением дифференциального уравнения. Функция предназначена для решения одномерного параболического дифференциального уравнения в частных производных. Аргументами функции `pdsolve` являются: u – имя функции (без аргументов); x – имя пространственной переменной; $xrange$ – массив из двух элементов, определяющий границы изменения пространственной переменной; t – имя переменной по времени; $trange$ – массив из двух элементов, определяющий границы изменения переменной t ; $xpts$ и $tpts$ – количество точек, на которые разбиваются интервалы изменения переменных x и t соответственно (эти параметры могут

отсутствовать в функции, в этом случае Mathcad подберет эти значения автоматически)¹.

Решение задачи 8.2 в Mathcad с помощью решающего блока и функции `pde_solve` представлено на рис. 8.18.

На рис. 8.19 показаны графики решения задачи 8.2

$$\begin{aligned}
 f(x) &:= 0.25 + \sin(0.15x) & u_0 &:= 2 \\
 L &:= 8 & U_1 &:= 0.25 & U_2 &:= 1.182 & T &:= 3 \\
 h &:= 1 & a &:= 5 \\
 \text{Given} \\
 u_t(x,t) &= a^2 \cdot u_{xx}(x,t) - h \cdot (u(x,t) - u_0) \\
 u(0,t) &= U_1 & u(L,t) &= U_2 \\
 u(x,0) &= f(x) \\
 u &:= \text{Pdsolve}\left[u, x, \begin{pmatrix} 0 \\ L \end{pmatrix}, t, \begin{pmatrix} 0 \\ T \end{pmatrix}, 30, 30\right]
 \end{aligned}$$

Рис. 8.18 ▼ Решение задачи теплопроводности с помощью функции `pdsolve`

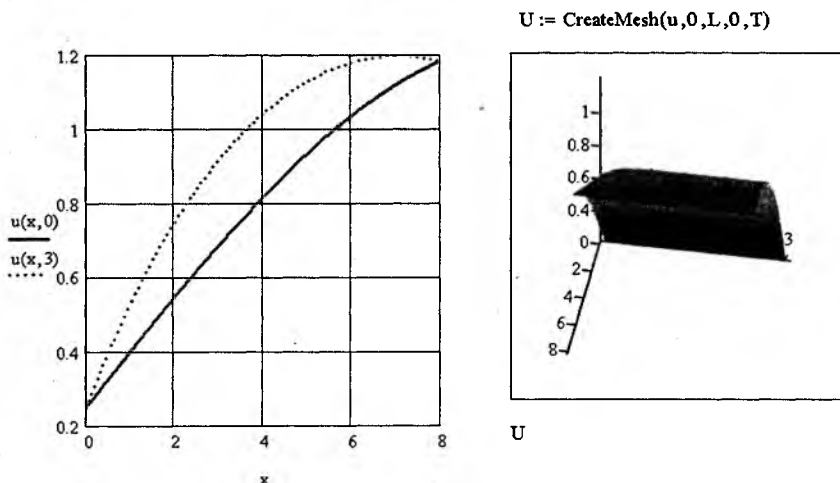


Рис. 8.19 ▼ График решения задачи теплопроводности 8.2

¹ Функция `pdsolve` может решать уравнения и системы, в которых нет дифференциальных уравнений, содержащих производную по t выше первой степени. В случае системы в качестве параметра u используется массив имен функций, функция `pdsolve` возвращает вектор-функцию.

Результатом решения дифференциального уравнения в частных производных является функция. Чтобы вычислить ее значение в любой точке, достаточно подставить в качестве аргументов конкретные величины.

Для решения одномерного параболического уравнения с условиями Дирихле на границе можно воспользоваться функцией `numol`. Функция `numol(xrange, xpts, trange, tpts, Npde, Nae, pde_f, pde_init, pde_bc)` возвращает матрицу значений в узлах сетки.

Здесь:

- `xrange` – массив из двух элементов, определяющий границы изменения пространственной переменной;
- `xpts` – количество точек, на которые разбивается интервал изменения переменной x ;
- `trange` – массив из двух элементов, определяющий границы изменения времени;
- `tpts` – количество точек, на которые разбивается интервал времени;
- `Npde` – количество дифференциальных уравнений в частных производных¹;
- `Nae` – количество дополнительных алгебраических уравнений, которые входят в систему дифференциальных уравнений в частных производных;
- `pde_f` – функция, определяющая правую часть параболического уравнения, которая зависит от переменных x ; t , u , u_x , u_{xx} ². Пример ее определения для параболического уравнения при решении задачи (8.2) приведен на рис. 8.20³;
- `pde_init` – функция, описывающая начальные условия;
- `pde_bc` – векторная функция, описывающая граничные условия. Для параболического уравнения с условиями Дирихле на границе она представляет собой матрицу-строку из трех компонентов, первый элемент которой – значение на левой границе интервала, второй – значение на правой границе, третий – D^4 .

На рис. 8.20 приведено решение задачи 8.2 с помощью функции `numol`.

¹ Как и функция `pdesolve`, функция `numol` позволяет решать и системы дифференциальных уравнений в частных производных.

² Здесь u – имя неизвестной функции, которая и является решением параболического уравнения.

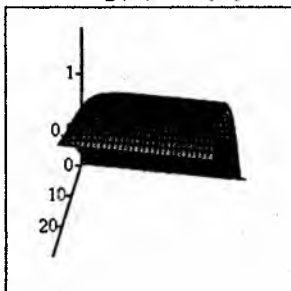
³ Обратите внимание, что функция должна зависеть от всех пяти переменных: x , t , u , u_x , u_{xx} .

⁴ Символ «D» указывает, что на границе заданы условия Дирихле.

```

f(x) := 0.25 + sin(0.15·x)  u0 := 3  L := 8  T := 3
h := 1  U1 := 0.25  U2 := 1.182  a := 5  Npde := 1  Nae := 0
pde_f(t,x,u,ux,uxx) := a^2·uxx - h·(u - u0)
pde_bc(t) := (U1 U2 "D")
V := numol[ [ (0), 30, (0), 30, Npde, Nae, pde_f, f, pde_bc ]

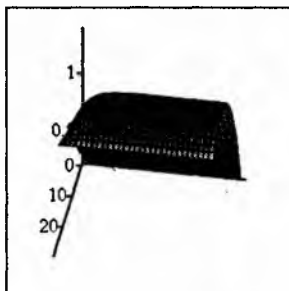
```



V

Рис. 8.20 ▼ Решение задачи теплопроводности с помощью функции numol

Как видно из рис. 8.13, 8.19, 8.20, решение задачи 8.2, полученное с помощью метода сеток, а также функций numol и pdesolve, совпадает с точным решением (рис. 8.21).



v1

Рис. 8.21 ▼ Точное решение задачи теплопроводности 8.2

8.4.2. Решение гиперболических уравнений

Как отмечалось ранее, функция `pdsolve` позволяет решать дифференциальные уравнения и системы, в которые не входят уравнения, содержащие производную по t выше первой степени. В любых гиперболических уравнениях присутствует вторая производная по t . Поэтому для того, чтобы решить гиперболическое уравнение, необходимо преобразовать его в систему дифференциальных уравнений в частных производных, введя дополнительную неизвестную функцию

$$v = \frac{\partial \omega}{\partial t}.$$

Рассмотрим решение гиперболических уравнений на примере задачи 8.3.

Вводя дополнительную переменную $v = \frac{\partial \omega}{\partial t}$, сведем гиперболическое уравнение (8.50) к следующей системе дифференциальных уравнений в частных производных:

$$\begin{aligned} v &= \frac{\partial \omega}{\partial t}, \\ \frac{\partial v}{\partial t} &= a^2 \frac{\partial^2 \omega}{\partial x^2} + \sin(x \cdot t), \\ \omega(x, 0) &= \varphi(0), \quad \omega(L, t) = \varphi(t), \\ \omega(x, 0) &= \varphi(x), \quad v(x, 0) = \psi(x), \end{aligned} \quad (8.55)$$

которую и будем решать с помощью решающего блока `Given - Pdsolve`. При его использовании для решения системы дифференциальных уравнений в частных производных следует учесть следующее:

- первым параметром в функции `pdsolve` будет массив имен функций, в нашем случае $\begin{pmatrix} \omega \\ v \end{pmatrix}$;
- функция `pdsolve` вернет вектор-функцию решения системы.

На рис. 8.22 представлено решение системы уравнений (8.55) с помощью функции `pdsolve`, на рис 8.23 – полученный график решения.

Решения задачи 8.3, полученные с помощью явной разностной схемы и функции `pdsolve`, практически совпадают.

Рассмотрим пример решения еще одной начально-граничной задачи [7].

ЗАДАЧА 8.4. В конце $x = L$ трубопровода $0 \leq x \leq L$ расход жидкости изменяется в момент времени $t = 0$ скачком на величину $A = \text{const}$; конец соединен с большим резервуаром, где давление жидкости остается неизменным. Считая, что до изменения расхода в конце $x = L$ давление и расход в трубопроводе были

$$\phi(x) := \sin(x) \quad \psi(x) := \cos(x) \quad f(x,t) := \sin(x \cdot t)$$

$$a := 4 \quad T := 2 \quad A := 3 \quad \gamma := 5 \quad L := 10$$

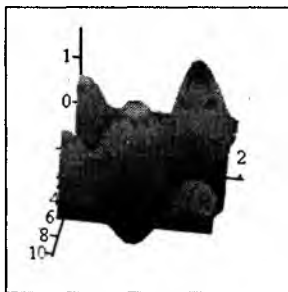
Given

$$v_t(x,t) = a^2 \omega_{xx}(x,t) + \sin(x \cdot t) \quad \omega_t(x,t) = v(x,t)$$

$$v(x,0) = \psi(x) \quad \omega(x,0) = \phi(x) \quad \omega(L,t) = \phi(L) \quad \omega(0,t) = 0$$

$$\begin{pmatrix} \omega \\ v \end{pmatrix} := \text{Pdsolve} \left[\begin{pmatrix} \omega \\ v \end{pmatrix}, x, \begin{pmatrix} 0 \\ L \end{pmatrix}, t, \begin{pmatrix} 0 \\ T \end{pmatrix}, 100, 100 \right]$$

Рис. 8.22 ▼ Решение уравнения (8.55) с помощью функции pdsolve



$$\text{CreateMesh}(\omega, 0, L, 0, T)$$

Рис. 8.23 ▼ График решения уравнения (8.55)

постоянными, найти изменение расхода в трубе при $t > 0$ и изменение давления в сечении $x = L$ при $t > 0$.

Начально-граничная задача, описывающая расход жидкости в трубе, имеет вид:

$$\begin{aligned} \frac{\partial^2 \omega}{\partial t^2} &= a^2 \frac{\partial^2 \omega}{\partial x^2} - 2\theta \frac{\partial \omega}{\partial t}, \quad 0 < x < L, \quad t > 0, \\ \omega_x(x, 0) &= 0, \quad \omega(L, t) = A, \\ \omega(x, 0) &= \omega_t(x, 0) = 0. \end{aligned} \quad (8.56)$$

Здесь $\omega(x, t)$ – расход жидкости в трубе. Для решения задачи (8.56) в Mathcad введем дополнительную переменную $v = \frac{\partial \omega}{\partial t}$ и сведем гиперболическое уравнение к системе дифференциальных уравнений в частных производных:

$$\begin{cases} v = \frac{\partial \omega}{\partial t}, \\ \frac{\partial v}{\partial t} = a^2 \frac{\partial^2 \omega}{\partial x^2} - 2\gamma v, \\ \omega_x(x, 0) = 0, \\ \omega(L, t) = A, \\ \omega(x, 0) = 0, \\ v(x, 0) = 0. \end{cases} \quad (8.57)$$

На рис. 8.24 представлено решение системы уравнений 8.57, на рис 8.25 – график решения, полученный с помощью функции `pdesolve`, и график точного решения.

$$a := 4 \quad T_{\omega} := 20 \quad A := 3 \quad \gamma := 5 \quad L_{\omega} := 10$$

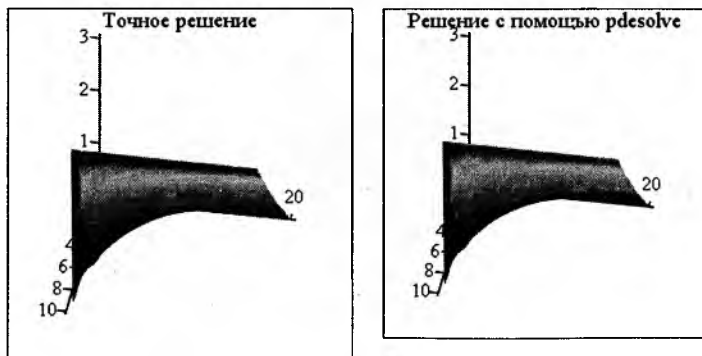
Given

$$v_t(x, t) = a^2 \omega_{xx}(x, t) - 2\gamma \cdot v(x, t) \quad \omega_t(x, t) = v(x, t)$$

$$v(x, 0) = 0 \quad \omega(x, 0) = 0 \quad \omega(L, t) = A \quad \omega(0, t) = 0$$

$$\begin{pmatrix} \omega \\ v \end{pmatrix} := \text{Pdesolve} \left[\begin{pmatrix} \omega \\ v \end{pmatrix}, x, \begin{pmatrix} 0 \\ L \end{pmatrix}, t, \begin{pmatrix} 0 \\ T \end{pmatrix} \right]$$

Рис. 8.24 ▼ Решение уравнения 8.56 с помощью функции `pdesolve`



W

Ω

Рис. 8.25 ▼ График решения уравнения 8.56

8.4.3. Решение эллиптических уравнений

Для решения уравнений Лапласа и Пуассона в Mathcad предусмотрены встроенные функции `relax` и `multigrid`, описанием которых и завершим знакомство с решением дифференциальных уравнений в частных производных в Mathcad.

Функция `multigrid` предназначена для решения уравнения Пуассона с нулевыми граничными условиями, обращение к функции имеет вид:

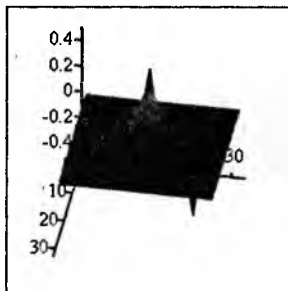
`multigrid(F, n)`

Здесь F – матрица правых частей уравнения Пуассона, размерность матрицы должна быть степенью числа 2 (8, 32, 64 и т.д.); n – число циклов в каждой итерации, число 2 дает хорошее приближение к решению. На рис. 8.26 представлено решение уравнения $\Delta u = -xy$ в области $\Omega(0 \leq x \leq 10; 0 \leq y \leq 5)$.

```

N:=32
i:=1..N
j:=1..N
Ei,j:=0
E10,15:=800
E15,25:= -900
u:=multigrid(-F,2)

```



u

Рис. 8.26 ▼ Решение уравнения Пуассона с помощью функции `multigrid`

Для решения более сложных эллиптических уравнений предназначена функция `relax`. Обращение к ней имеет вид:

`relax(A, B, C, D, E, F, U, rjac)`

Здесь:

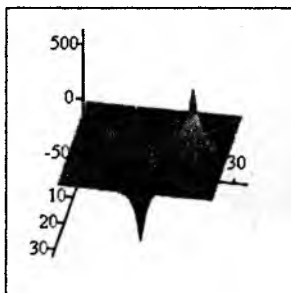
- A, B, C, D, E – квадратные матрицы коэффициентов разностной схемы $Au_{i+1,j} + Bu_{i-1,j} + Cu_{i,j+1} + Du_{i,j-1} + Eu_{i,j}$, аппроксимирующей левую часть уравнения Лапласа (или Пуассона);
- F – матрица правых частей уравнения Пуассона;
- U – матрица начального приближения дифференциального уравнения, в которой обязательно должны быть корректно заданы граничные условия;
- $rjac$ – параметр численного алгоритма (спектральный радиус итераций Якоби), в качестве $rjac$ можно принять величину $1 - \frac{2\pi}{n}$, где n – размерность матрицы решений.

Пример использования функции приведен на рис. 8.27.

```

N:=32
i:=0..N      j:=0..N
ai,j:=1      b:=a      c:=a      d:=a      e:=-4·a
Fi,j:=0      F25,16:=103      F16,25:=−103      rjac:=1−2· $\frac{\pi}{N}$ 
ui,j:=0
U:=relax(a,b,c,d,e,F,u,rjac)

```



U

Рис. 8.27 ▼ Использование функции relax для решения уравнения Пуассона

К сожалению, встроенные функции Mathcad не позволяют решать гиперболические и параболические уравнения с двумя пространственными переменными. Кроме того, далеко не все даже одномерные параболические и гиперболические уравнения могут быть решены с помощью функций pdsolve и num01.

Например, проблемы возникают при решении уравнений с δ -функцией, со смешанными условиями или условиями Неймана на границе. Метод конечных разностей и встроенные функции Mathcad не всегда позволяют решать эллиптические уравнения при сложной геометрии области. Во всех этих случаях можно использовать метод конечных элементов, который реализован в MATLAB в виде специализированного приложения Partial Differential Equations Toolbox (PDE Toolbox).

8.5. Использование MATLAB для решения уравнений в частных производных

Двумерные уравнения математической физики могут быть решены с помощью PDE Toolbox методом конечных элементов. Для вызова графической среды PDE Toolbox в командном окне MATLAB достаточно набрать команду `pdetool` (см. рис. 8.28).

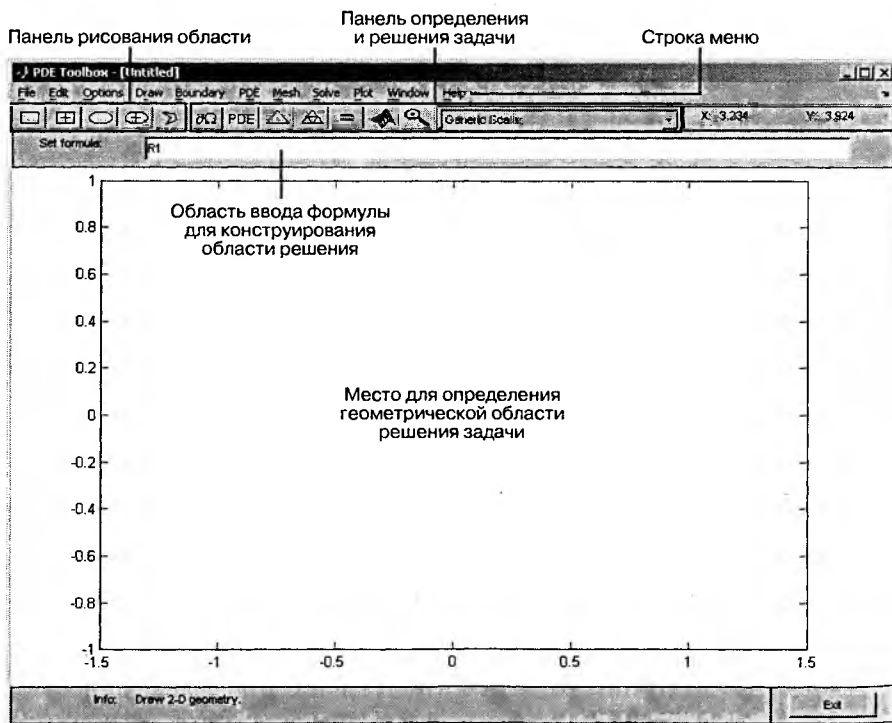


Рис. 8.28 ▼ Графическая среда `pdetool`

Можно выделить следующие этапы решения уравнений в частных производных:

1. Конструирование (построение) области, в которой решается уравнение.
2. Ввод уравнения в частных производных.
3. Определение начальных и граничных условий.
4. Триангуляция области.
5. Решение уравнения.
6. Визуализация результата.

Знакомство с `pdetool` начнем с решения эллиптических уравнений.

8.5.1. Решение эллиптических уравнений

Рассмотрим задачу о распределении тепла в пластине.

ЗАДАЧА 8.5. Найти распределение тепла в области G , изображенной на рис. 8.29.

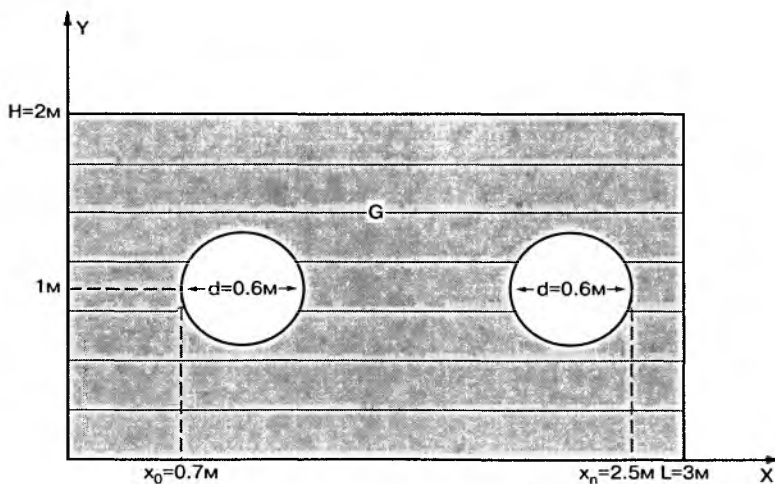


Рис. 8.29 ▽ Область G

Круговые отверстия имеют одинаковый размер диаметра – 0.6 м. Правая и левая границы теплоизолированы. Коэффициент температуропроводности $a^2 = 225$. На верхней и нижней границе области G температура равна 580, на окружностях – 450.

Дифференциальное уравнение, описывающее распределение температуры, можно записать следующим образом:

$$a^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 0. \quad (8.58)$$

Граничные условия на левой и правой границе имеют вид:

$$n a^2 \frac{\partial u}{\partial n} = 0. \quad (8.59)$$

Здесь n – вектор нормали к границе.

Таким образом, задача о распределении тепла в области G полностью описывается уравнением (8.58) и граничными условиями.

Рассмотрим подробно процесс решения этой задачи в `pdetool`.

Первый этап – конструирование области. Запустив среду `pdetool` (см. рис. 8.28), вы увидите, что по умолчанию область решения задачи должна принадлежать прямоугольнику $[-1.5 \ 1.5; -1 \ 1]$. Рассмотрим, как можно установить другие границы изменения x и y (в нашем случае можно установить границы изменения x от -1 до 4 , y – от -1 до 3). Для изменения границ отображаемой области по x и y необходимо выполнить команду меню **Options** ► **Axes Limits** (Настройки ► Пределы осей), после чего появится окно, в котором указаны границы изменения x и y (рис. 8.30). Изменим границы x (от -1 до 4) и y (от -1 до 3) (рис. 8.31).

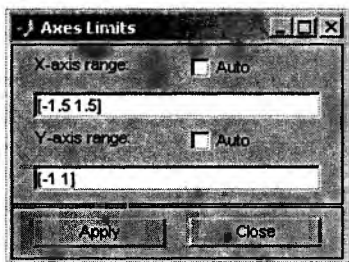


Рис. 8.30 ▼ Окно **Axes Limits**

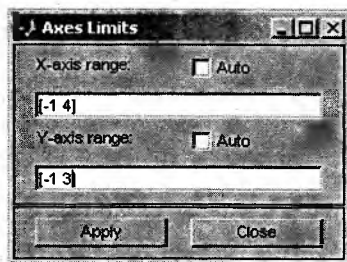


Рис. 8.31 ▼ Окно **Axes Limits**
с измененными значениями границ

Для удобства можно изобразить линии сетки на координатной сетке. Для этого необходимо выполнить команду меню **Options** ► **Grid** (Настройки ► Сетка), после чего среда `pdetool` будет иметь вид, изображенный на рис. 8.32.

Для конструирования области предназначен пункт меню **Draw** (Рисование)¹. Для перехода в режим конструирования можно выполнить команду меню **Draw** ► **Draw Mode** (Рисование ► Режим рисования) или просто выбрать один

¹ Речь идет о рисовании области, в которой решается уравнение.

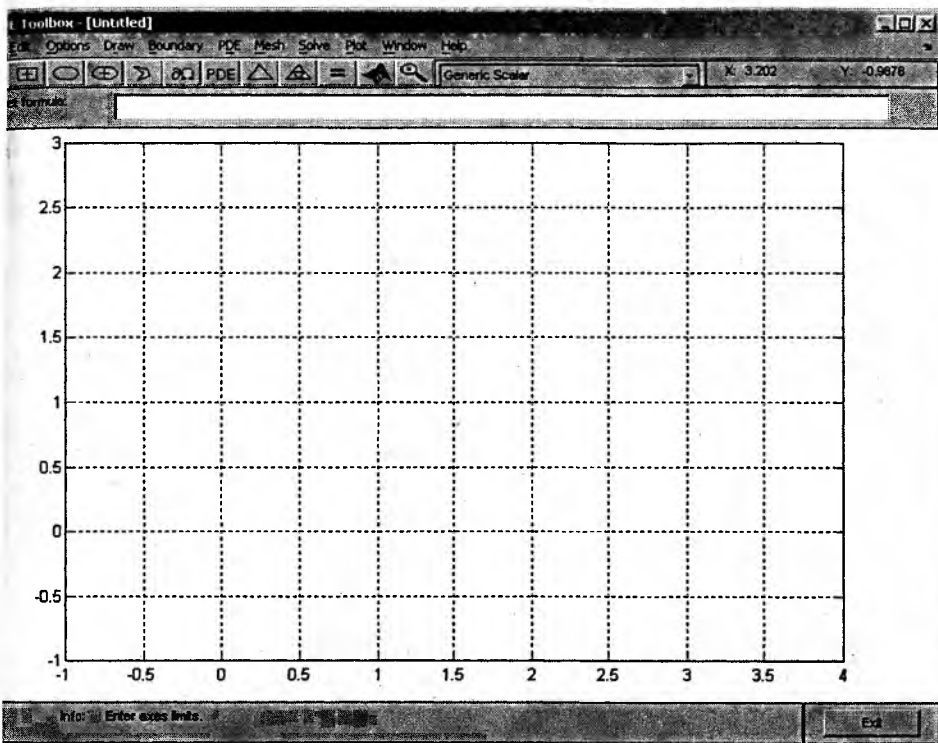


Рис. 8.32 ▾ Графическая среда pdeplot, настроенная для решения задачи 8.5

геометрических примитивов и начать рисовать. Геометрические примитивы можно выбирать в пункте меню **Draw** или воспользоваться панелью рисования области. В pdeplot есть следующие примитивы для рисования области:

- **прямоугольник.** Для выбора прямоугольника можно выполнить команду меню **Draw** ➤ **Rectangle** ➤ **Square** (Рисование ➤ Прямоугольник ➤ Квадрат) или щелкнуть по кнопке  на панели рисования области; если при прорисовке фигуры удерживать клавишу **Ctrl**, то вместо прямоугольника получится квадрат;
- **центрированный прямоугольник.** Для выбора этой фигуры можно выполнить команду меню **Draw** ➤ **Rectangle** ➤ **Square (Centered)** (Рисование ➤ Прямоугольник ➤ Квадрат (Центрированный)) или щелкнуть по кнопке  на панели рисования области; если удерживать клавишу **Ctrl**, то вместо прямоугольника получится квадрат, отличие этого прямоугольника от

обычного состоит в том, что при прорисовке прямоугольник растягивается не только в сторону правого нижнего угла, но и левого верхнего;

- эллипс. Для его выбора можно выполнить команду **Draw ► Ellipse ► Circle** (Рисование ► Эллипс ► Круг) или выбрать инструмент  на панели рисования области), при нажатой клавише **Ctrl** вместо эллипса получится круг; кроме обычного эллипса есть центрированный (команда меню **Draw ► Ellipse ► Circle (Centered)** (Рисование ► Эллипс ► Круг (Центрированный)) или кнопка 
- многоугольник (команда меню **Draw ► Polygon** (Рисование ► Многоугольник) или кнопка 

Построенные объекты можно поворачивать с помощью команды **Draw ► Rotate** (Рисование ► Вращение).

Для решения рассматриваемой задачи необходимо нарисовать прямоугольник, а в нем – два круга. Начнем с рисования прямоугольника, попробуем нарисовать его как можно точнее. Для уточнения координат прямоугольника необходимо дважды щелкнуть по прямоугольнику, что приведет к появлению диалогового окна **Object Dialog** (Вызов диалогового окна для изменения свойств объекта) – рис. 8.33.

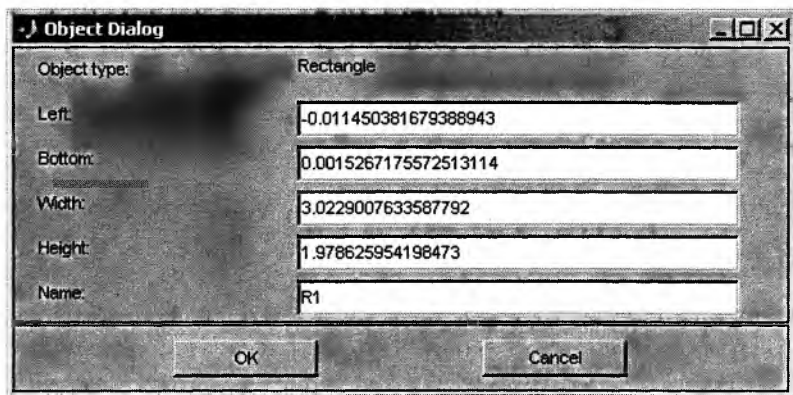


Рис. 8.33 ▼ Диалоговое окно **Object Dialog**

Параметры **Left** (Левый) и **Bottom** (Нижний) определяют координаты левого нижнего угла прямоугольника, введем там нули (левый нижний угол в нашем случае совпадает с началом координат). Параметры **Width** (Ширина) и **Height** (Высота) определяют ширину и высоту прямоугольника (для задачи 8.5 эти величины равны 3 и 2 соответственно). Параметр **Name** (Имя) определяет

имя объекта, все прямоугольники по умолчанию начинаются с «R» (от английского слова Rectangle)¹.

Следующим этапом будет рисование двух кругов, первый – с центром в точке (0, 0) и радиусом 0.3, второй – с центром в точке (2.2, 1) и радиусом 0.3. Для построения круга используем кнопку  при нажатой клавише **Ctrl** и, нарисовав круг, уточним его координаты: **X-Center**, **Y-Center** – координаты центра круга, **Radius** – радиус круга (рис. 8.34).

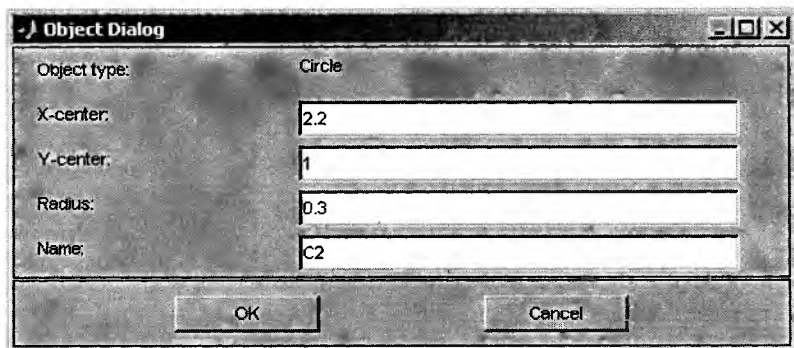


Рис. 8.34 ▼ Диалоговое окно **Object Dialog** для круга

Далее надо определить взаимосвязи между объектами, которые в `pdetool` определяются с помощью операций:

- – – вычитания объектов, имеет наивысший приоритет;
- + – объединения объектов;
- * – пересечения объектов;

Операции пересечения и объединения имеют одинаковый приоритет, для изменения приоритетов, как и в алгебре, используются круглые скобки.

По умолчанию несколько объектов соединяются с помощью знака «плюс». В нашем случае необходимо из прямоугольника вычесть два круга, поэтому в области формулы для конструирования области решения необходимо ввести формулу $R1 - C1 - C2$ (рис. 8.35).

На втором этапе решения задачи необходимо ввести уравнение в частных производных. Для ввода уравнений служит пункт меню **PDE**². Первый подпункт меню **PDE Mode** служит для перехода в режим ввода уравнений. Второй подпункт

¹ Имя эллипса начинается с буквы «E», круга – с буквы «C», многоугольника – с буквы «P».

² PDE – сокращение от Partitional Differential Equation.

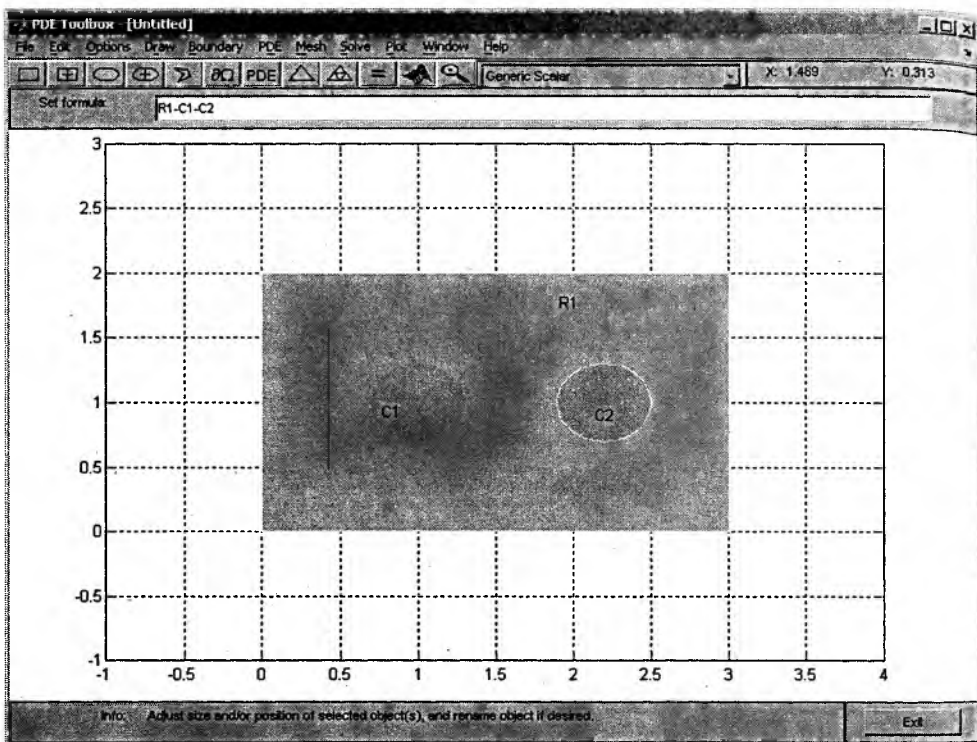


Рис. 8.35 ▼ Графическая среда pdeplot после ввода области решения задачи 8.5

служит для показа подобластей, входящих в область решения. В задаче 8.5 есть всего одна подобласть. Третий пункт **PDE Specification** или кнопка **PDE** позволяет выбрать тип уравнений и ввести его коэффициенты (рис. 8.36).

Кроме того, в pdeplot есть возможность задавать тип решаемой задачи. Для этого можно воспользоваться командой **Options ► Application** (Настройки ► Приложение) – рис. 8.37 – или кнопкой **Generic Scalar**.

При решении задачи 8.5 можно выбрать в качестве типа решаемой задачи **Heat Transfer** (Задача о распределении тепла) или просто ввести коэффициенты эллиптического уравнения в окне **PDE Specification**. Установим тип решаемой задачи **Heat Transfer**, после чего в окне **PDE Specification** (рис. 8.38) определим параметры уравнения в частных производных.

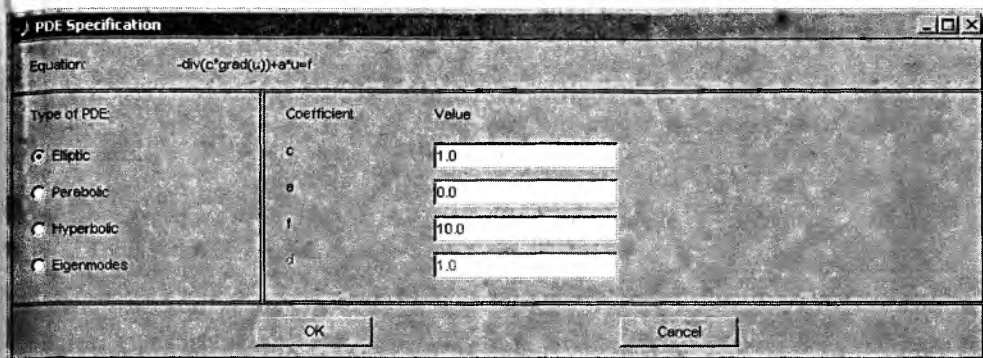


Рис. 8.36 ▼ Окно **PDE Specification** – определение типа и коэффициентов дифференциального уравнения в частных производных

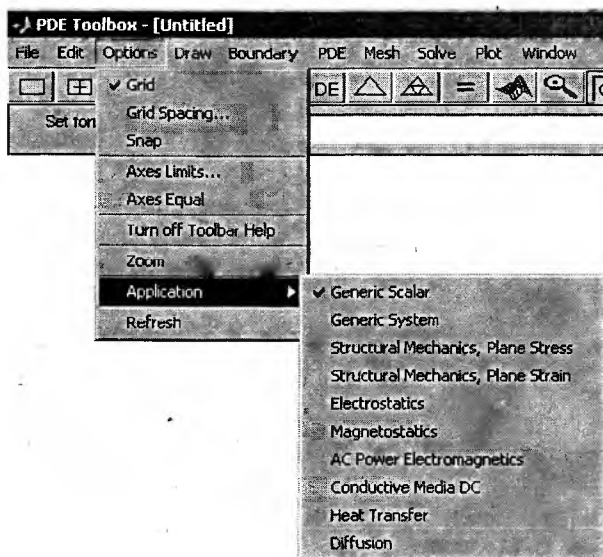


Рис. 8.37 ▼ Выбор типа решаемой задачи

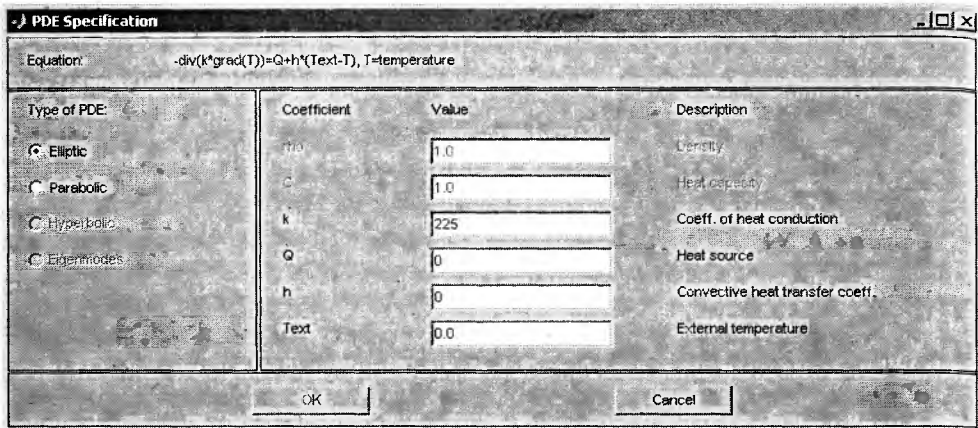


Рис. 8.38 ▾ Окно **PDE Specification** – определение коэффициентов задачи о распределении тепла¹

Третий этап решения задачи посвящен определению граничных условий. Для ввода граничных условий предназначен пункт меню **Boundary** (Граничные условия). Для перехода в режим ввода граничных условий можно выполнить команду меню **Boundary** ▸ **Boundary Mode** (Граничные условия ▸ Режим граничных условий), воспользоваться комбинацией клавиш **Ctrl+B** или кнопкой .

После этого, чтобы определить условия, необходимо щелчком мыши² выделить границу и выполнить команду меню **Boundary** ▸ **Specify Boundary Condition** (Граничные условия ▸ Установить граничные условия) или осуществить двойной щелчок мышью по выделенной границе. Введем граничные условия для задачи 8.5.

Выделяем верхнюю и нижнюю границы области и переходим к вводу граничных условий. На этом участке границы определены условия Дирихле вида $h * T = \tau$, для задачи 8.5 в окне **Boundary Condition** вводим $h = 1$, $\tau = 580$ (рис. 8.39).

¹ Напомним читателю, что для скалярной функции $T(x, y, z)$ $\text{grad}(T)$ – градиентом скалярной функции T является вектор $\left(\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}, \frac{\partial T}{\partial z} \right)$, а дивергенция от градиента $\text{div}(\text{grad}(T))$

вычисляется по формуле: $\text{div}(\text{grad}(T)) = \Delta T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}$.

² Если необходимо объединить несколько участков границы, то в этом случае щелчок по каждому из участков осуществляется при нажатой клавише **Shift**.

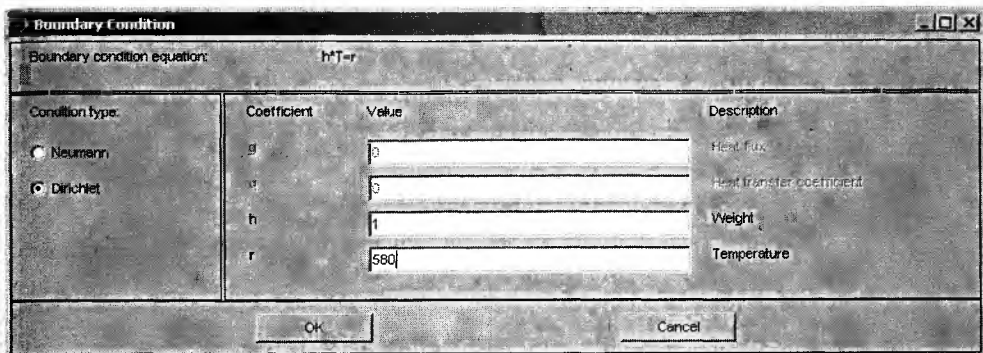


Рис. 8.39 ▼ Ввод условий для верхней и нижней границы области G

Сгруппировав границы отверстий, установим температуру на их границах 450 (рис. 8.40).

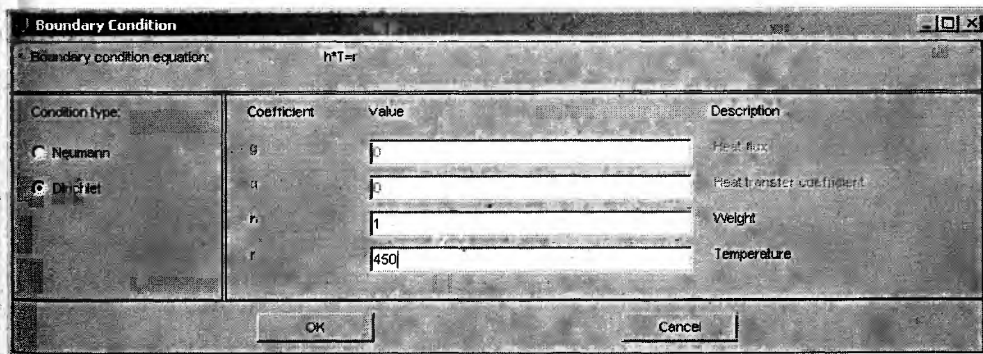


Рис. 8.40 ▼ Ввод условий на границах отверстий области G

Осталось ввести условия Неймана вида (8.60) (или, что то же самое, $n \cdot k \cdot \text{grad}(T) = 0$) на левой и правой границе области G (рис. 8.41).

Ввод условий задачи завершен.

На следующем, четвертом, этапе проведем триангуляцию области – покроем область сеткой, состоящей из треугольников. Это и есть *конечные элементы*, на которые разбивается область. Режимом триангуляции управляет пункт меню **Mesh** (Сетка). Для перехода в режим триангуляции служит команда **Mesh ► Mesh Mode** (Сетка ► Режим сетки). Переход в режим триангуляции сразу разбивает на крупные треугольники (рис. 8.42). Этого же можно добиться с помощью команды начала триангуляции – пункт меню **Mesh ► Initialize Mesh** (Сетка ► Определить (Инициализировать) сетку), комбинация клавиш **Ctrl+I** или кнопка .

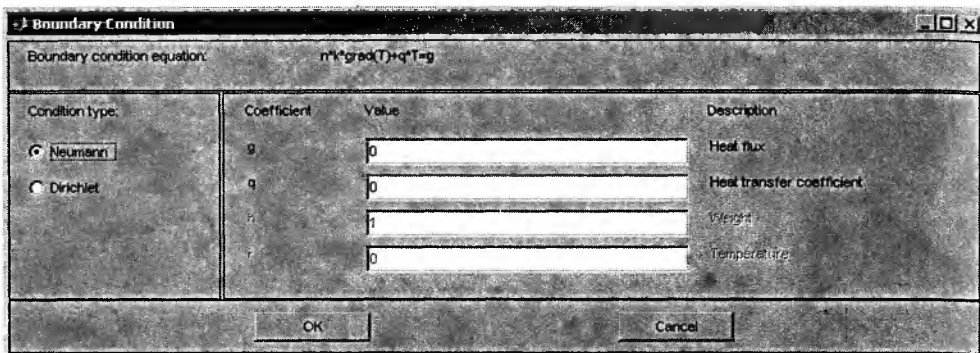


Рис. 8.41 ▾ Ввод условий Неймана для левой и правой границы области G

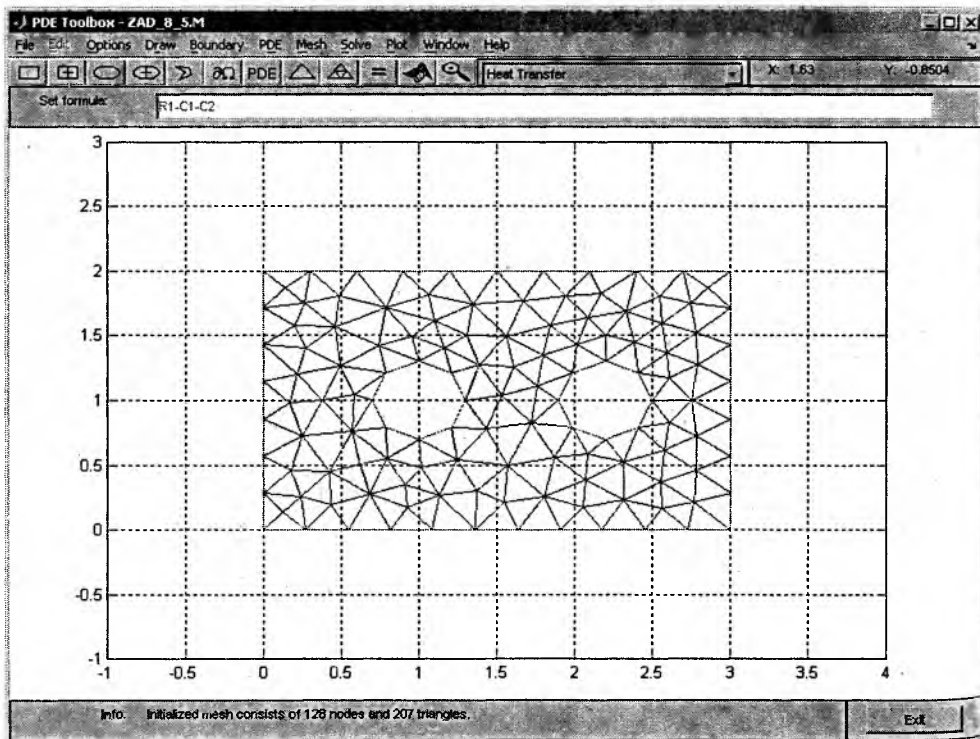


Рис. 8.42 ▾ Начальная триангуляция области G

Обычно начальной триангуляции недостаточно для получения решения с остаточной точностью. Для разбиения области на более мелкие треугольники лужит команда меню **Mesh ► Refine Mesh** (Сетка ► Уплотнить сетку) или кнопка . Для нашей задачи критерием достаточной величины треугольников может служить вид отверстий. Дважды выполнив команду **Mesh ► Refine Mesh**, получим сетку в области решения (рис. 8.43), необходимую для получения решения задачи 8.5 с достаточной точностью. Обращаем внимание читателя, что из-за слишком мелкой сетки время решения задачи значительно увеличивается. Возврат к начальной триангуляции осуществляется с помощью команды меню **Mesh ► Initialize Mesh** (Сетка ► Определить (Инициализировать) сетку) или кнопки .

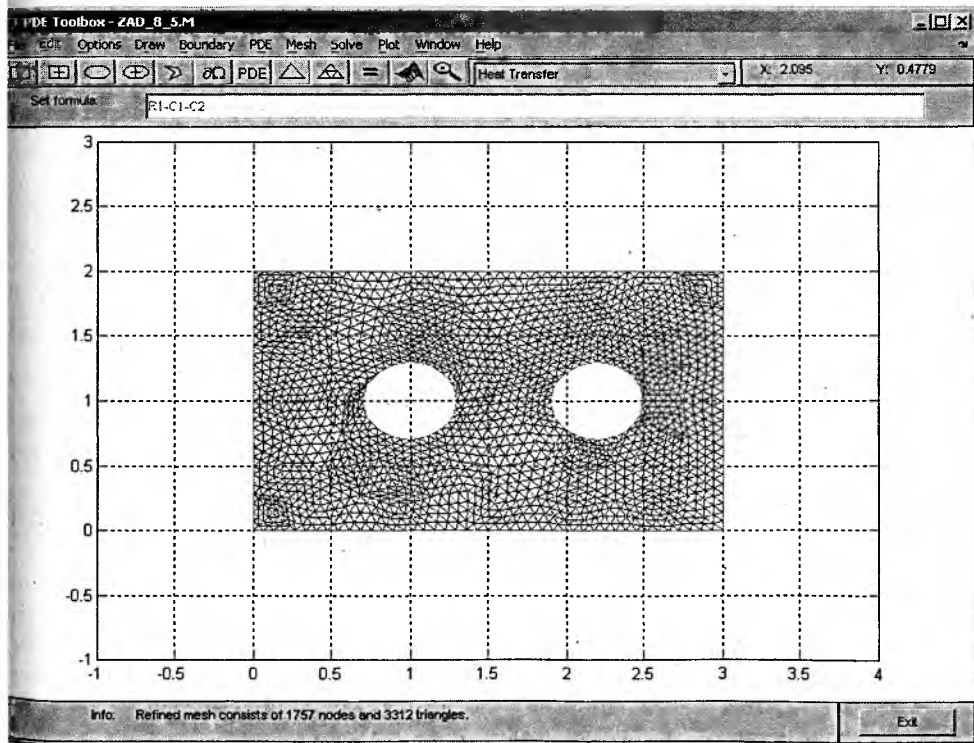


Рис. 8.43 ▼ Уменьшенная сетка в области решения G

На предпоследнем, пятом, этапе для решения поставленной задачи необходимо выполнить команду меню **Solve ▸ PDE Solve** (Решение ▸ Решить PDE) или щелкнуть по кнопке .

После этого найденное решение автоматически изображается на экране в виде цветного контурного графика (шестой этап решения задачи). Контурный график решения задачи 8.5 изображен на рис. 8.44.

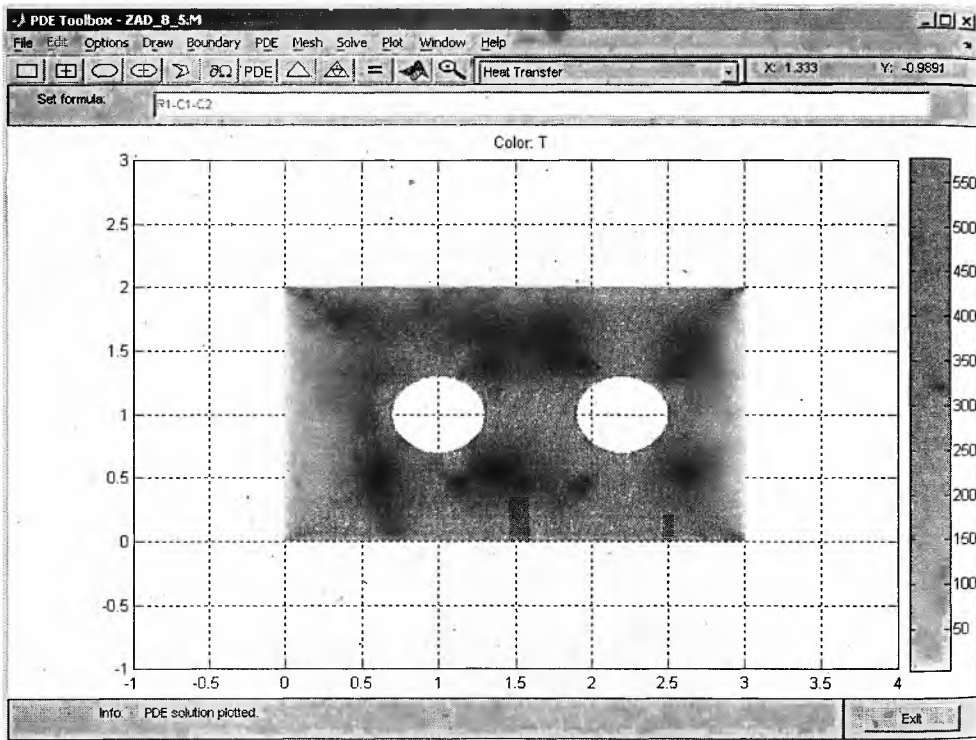


Рис. 8.44 ▾ Контурный график решения задачи 8.5

Контурный график не всегда достаточно хорошо позволяет проанализировать полученное решение. Пользователь может изменить тип построенного графика, для чего можно выполнить команду меню **Plot ▸ Parameters** (График ▸ Параметры) или щелкнуть по кнопке . В появившемся диалоговом окне (рис. 8.45) можно установить необходимые параметры графика решения, для построения трехмерного графика достаточно установить флажок **Height (3-D plot)**. На рис. 8.46 изображен трехмерный график найденного решения задачи 8.5.

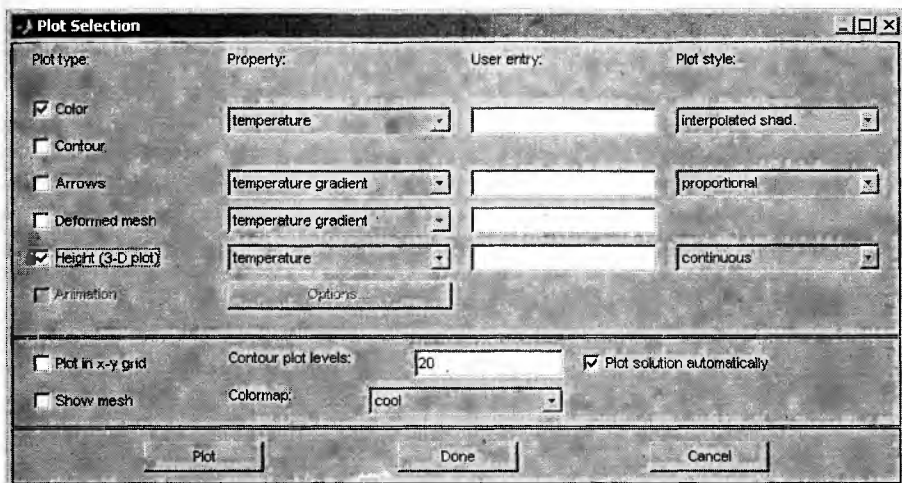


Рис. 8.45 ▼ Контурный график решения задачи 8.5

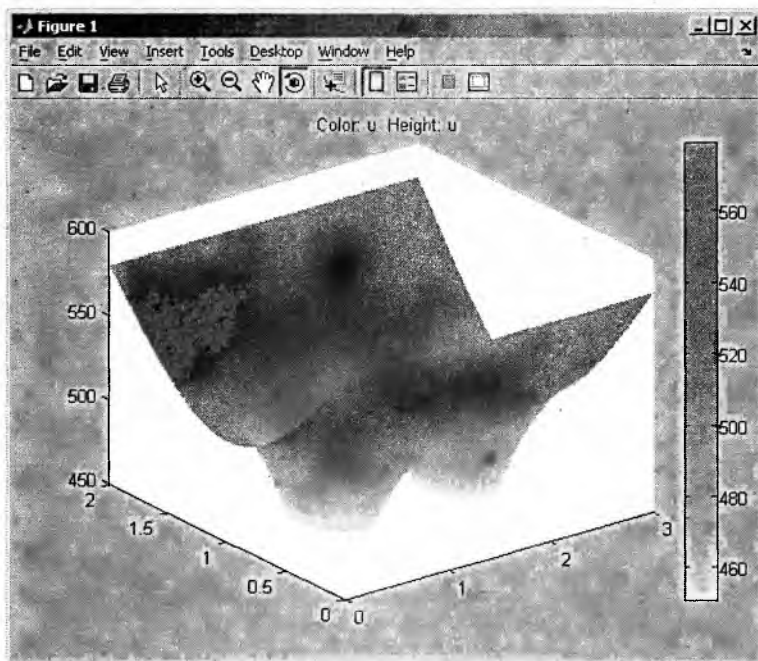


Рис. 8.46 ▼ Трехмерный график решения задачи 8.5

При решении задачи 8.5 были рассмотрены основные этапы решения дифференциального уравнения в частных производных в графической среде pdetool.

Теперь рассмотрим, как в pdetool хранится область решения, сетка и полученное решение. Для экспорта области решения в меню **Draw** есть команда **Export Geometry Description, Set Formula, Labels** (Экспорт описания геометрии области, формулы, меток), с помощью которой (рис. 8.47) можно экспортировать геометрию области и формулу связи между объектами в MATLAB.

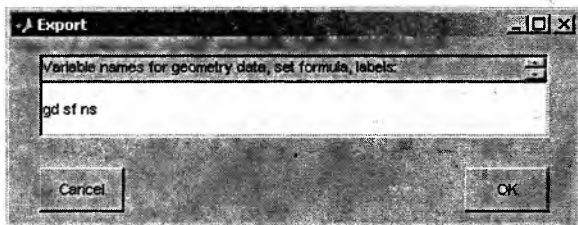


Рис. 8.47 ▾ Диалоговое окно **Export**, функция **Export Geometry Description, Set Formula, Labels...**

Как видно из рис. 8.47, можно экспортировать три объекта *gd*, *sf* и *ns*. Матрица геометрии области хранится в *gd*, формула связи хранится между объектами в переменной *sf*. Рассмотрим структуру матрицы *gd* на примере области *G* (рис. 8.29).

```
gd=
3.0000    1.0000    1.0000
4.0000    1.0000    2.2000
0         1.0000    1.0000
3.0000    0.3000    0.3000
3.0000     0         0
0         0         0
2.0000     0         0
2.0000     0         0
0         0         0
0         0         0
```

Число столбцов матрицы совпадает с количеством объектов, входящих в область решения, каждый столбец отвечает за отдельный объект. Рассмотрим структуру каждого столбца. Первый элемент столбца содержит номер объекта: 1 – круг, 2 – многоугольник, 3 – прямоугольник, 4 – эллипс.

Для прямоугольника и многоугольника второй элемент столбца содержит количество сторон фигуры, далее хранятся координаты вершин.

Для круга второй и третий элементы содержат координаты центра круга, четвертый – радиус.

Эллипс отличается от круга тем, что четвертый и пятый элемент столбца содержат величины полуосей эллипса, а шестой – угол поворота в радианах.

В текстовой переменной `sf` хранится формула.

В связи с тем, что решение задачи ищется в узлах сетки после проведения триангуляции, рассмотрим, как хранятся координаты узлов сетки. Сетка, построенная для задачи 8.5, очень сложна для первого знакомства с матрицами, описывающими структуру сетки. Поэтому построим простую прямоугольную область и инициализируем для нее треугольную сетку (рис. 8.48).

Для вывода на экран номеров узлов сетки служит команда **Mesh ► Show Node Labels** (Сетка ► Показать узлы сетки), а для вывода номеров треугольных конечных элементов – команда **Mesh ► Show Triangle Labels** (Сетка ►

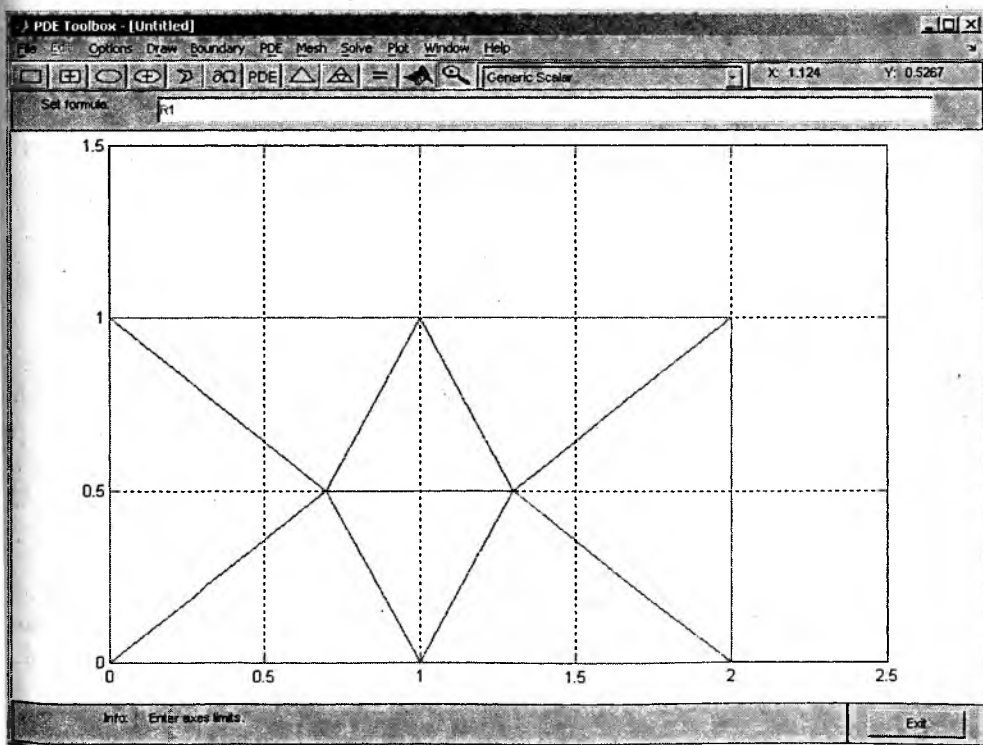


Рис. 8.48 ▼ Сетка для простой области

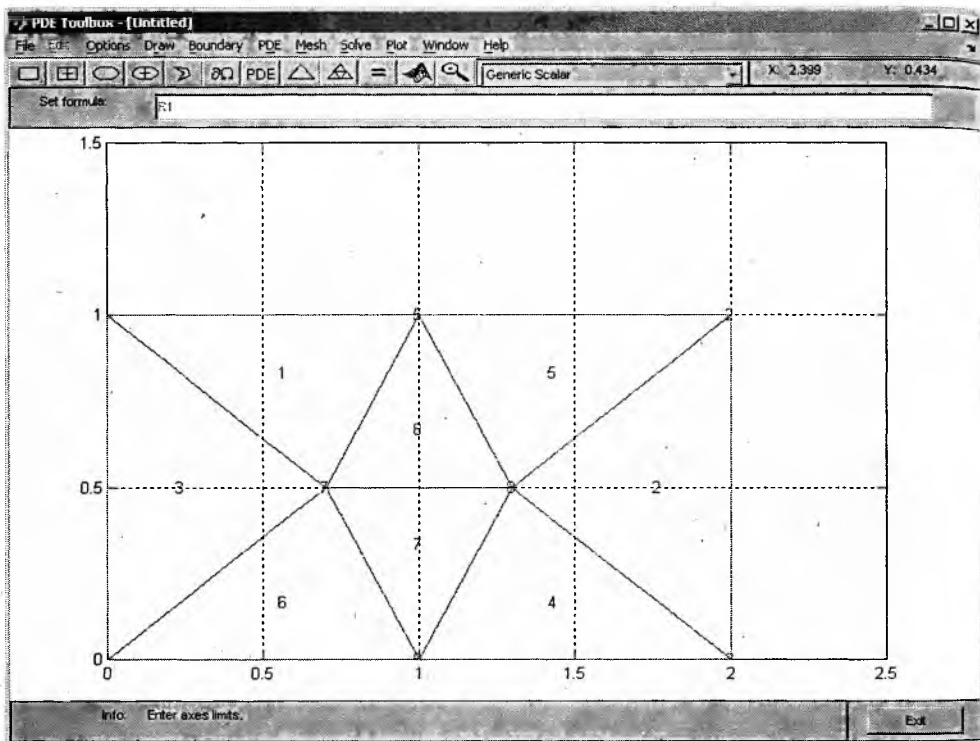


Рис. 8.49 ▾ Сетка для простой области с пронумерованными узлами и конечными элементами

Показать метки треугольных элементов). После нумерации узлов и конечных элементов сетка будет иметь вид, такой как на рис. 8.49.

Теперь рассмотрим, как хранится сетка. Для этого экспортируем ее в MATLAB с помощью пункта меню **Mesh ▸ Export Mesh** (Сетка ▸ Экспорт сетки) – рис. 8.50.

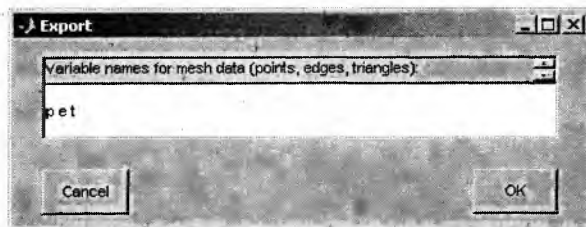


Рис. 8.50 ▾ Окно **Export**, функция **Export Mesh**

Матрица p содержит координаты всех точек, каждый столбец содержит координаты одной точки (x и y координаты); для области, изображенной на рис. 8.49, она имеет вид:

```
p=
0      2.0000    2.0000    0      1.0000    1.0000    0.7000    1.3000
1.0000    1.0000    0      0      1.0000    0      0.5000    0.5000
```

Каждый столбец матрицы t состоит из 4-х элементов: первые три содержат номера узлов, образующих треугольник (конечный элемент), четвертый – номер подобласти, которой он принадлежит. Число столбцов матрицы t равно количеству конечных элементов, на которые разделена область. Для области, изображенной на рис. 8.49, матрица t будет такой¹:

```
t=
5  3  1  6  2  4  7  5
1  2  4  3  5  6  6  7
7  8  7  8  8  7  8  8
1  1  1  1  1  1  1  1
```

В матрице e находится информация об узлах, формирующих границу подобласти. В нашем примере матрица e будет такой:

```
e=
1      5      2      3      6      4
5      2      3      6      4      1
0      0,5    0      0      0,5    0
0,5    1      1      0,5    1      1
1      1      2      3      3      4
0      0      0      0      0      0
1      1      1      1      1      1
```

Решение представляет собой массив значений u : его элементы соответствуют координатам узлов, определенных матрицей p . Количество элементов массива решений u равно количеству столбцов матрицы p . Для экспорта решения в MATLAB служит команда меню **Solve > Export Solution** (Решение > Экспорт решения). Сопоставляя значения матриц p , t и e , можно анализировать полученное решение в различных точках. Чем мельче сетка, тем более точное решение будет получено. Но, с другой стороны, мелкая сетка требует больше времени для решения задачи.

Рассмотрев основные возможности среды `pdetool` на примере задачи 8.5, решим еще несколько уравнений.

¹ 4-я строка постоянно равна 1, так как область решения состоит из одной подобласти – исходного прямоугольника.

ЗАДАЧА 8.6. Решить уравнение:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -x - y \quad (8.60)$$

в области $\Omega(-1 \leq x \leq 1; -1 \leq y \leq 1)$ при $u(x, y) = 0$ на границе области.

Первый этап решения – строим область, в которой будем искать решение (рис. 8.51).

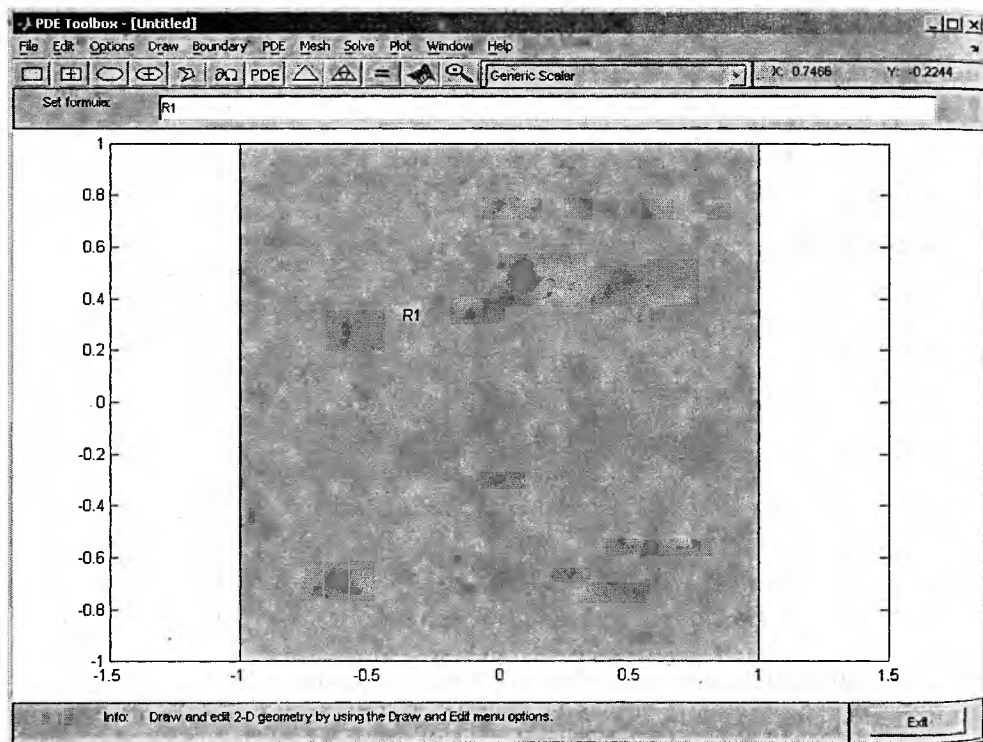


Рис. 8.51 ▽ Область решения задачи 8.6

На втором этапе вводим уравнения (8.59), воспользовавшись пунктом меню **PDE ► PDE Specification** (рис. 8.52). Обращаем внимание читателя на то, что параметры уравнения могут быть не только постоянными, но и функциями от x и y .

PDE Specification

Equation: $-\text{div}(c \cdot \text{grad}(u)) + a \cdot u = f$

Type of PDE:	Coefficient	Value
☒ Elliptic	c	1.0
☐ Parabolic	a	0.0
☐ Hyperbolic	f	$x+y$
☐ Eigenmodes	d	1.0

OK Cancel

Рис. 8.52 ▼ Ввод уравнения для задачи 8.6

На третьем этапе выделяем стороны квадрата (границы области) и определяем нулевые граничные условия (рис. 8.53).

Boundary Condition

Boundary condition equation: $h \cdot u = g$

Condition type	Coefficient	Value	Description
☐ Neumann	a	0	
☒ Dirichlet	d	0	
	h	1	
	r	0	

OK Cancel

Рис. 8.53 ▼ Ввод граничных условий для задачи 8.6

Далее проводим триангуляцию области (рис. 8.54).

С помощью пункта меню **Solve ► PDE Solve** или кнопки **=** запускаем процесс решения задачи. На рис. 8.55 представлен полученный график решения.

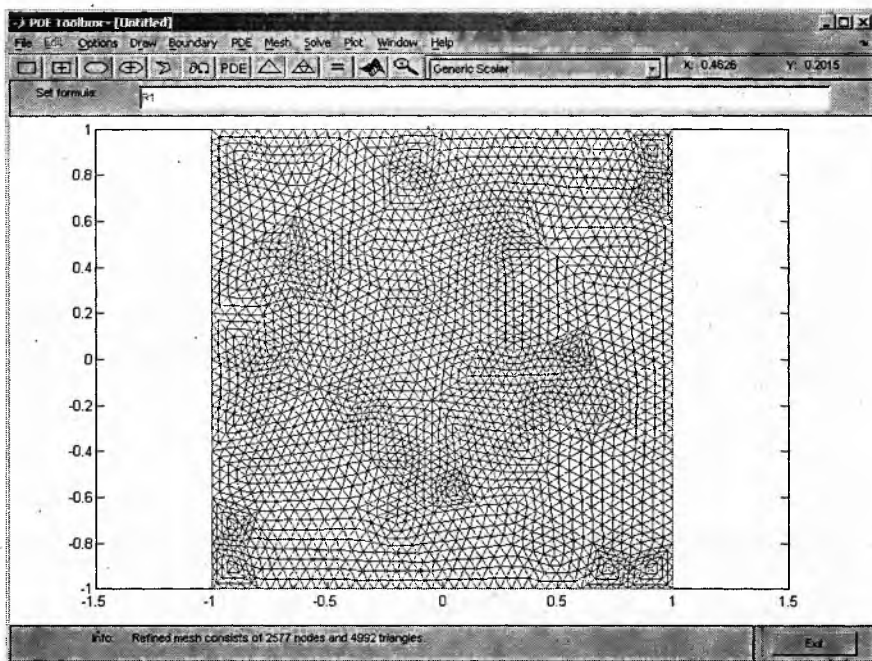


Рис. 8.54 ▽ Триангуляция области решения задачи 8.6

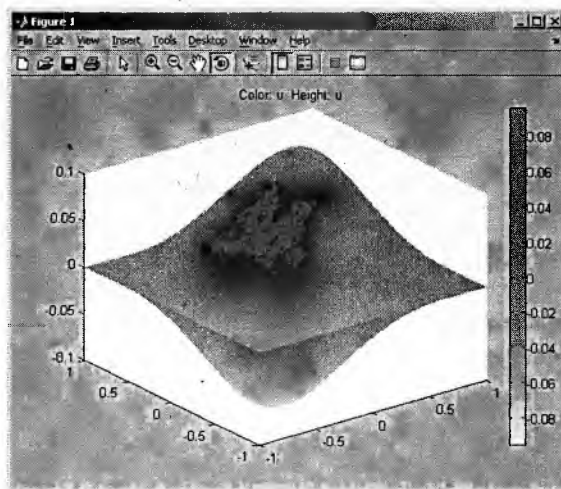


Рис. 8.55 ▽ График решения задачи 8.6

Рекомендуем читателю сравнить полученное решение с приведенным в [9] решением задачи, полученным методом Рунге: $u(x, y) = \frac{35}{208} (x^2 - 1) (y^2 - 1) (x + y)$. Для этого можно построить график разности двух решений. Как это можно сделать, показано на рис. 8.56. Обратите внимание, что с помощью окна **Plot Selection** можно строить график функции ее градиента и пользовательской функции (рис. 8.57).

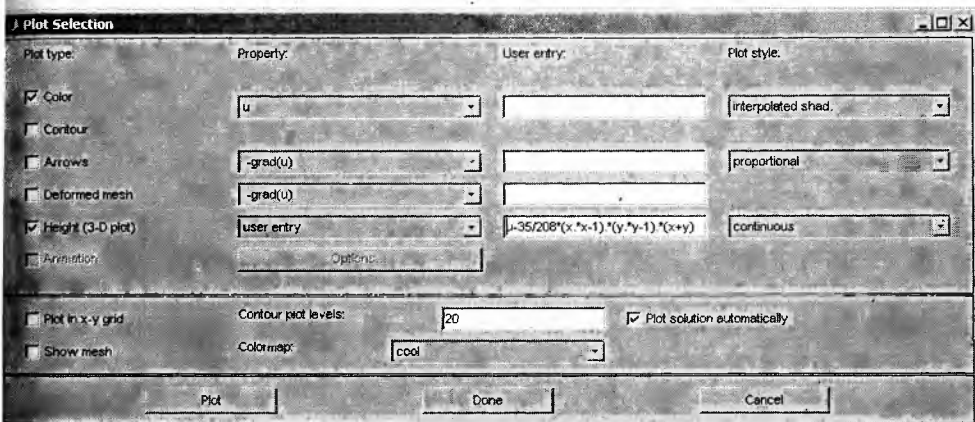


Рис. 8.56 ▼ Построение графика разности двух функций

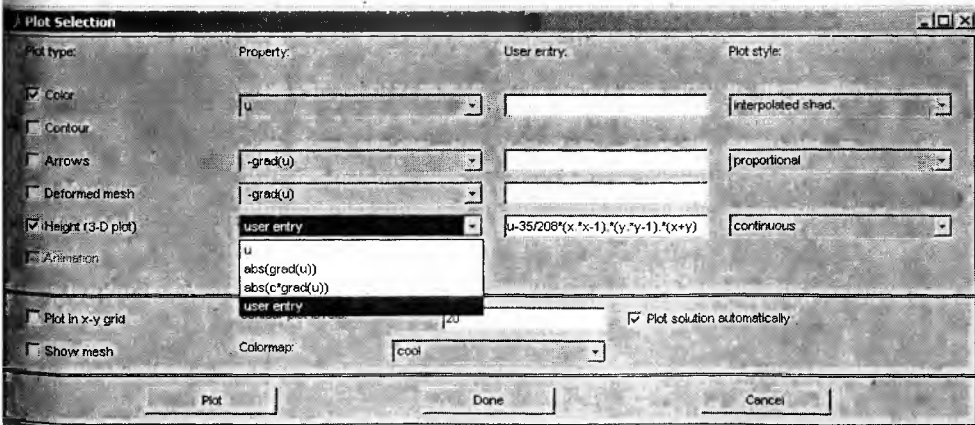


Рис. 8.57 ▼ Выбор параметра **Property** в окне **Plot Selection**

8.5.2. Решение гиперболических и параболических уравнений

Решение нестационарных задач (гиперболических и эллиптических уравнений) рассмотрим на примере решения о колебаниях прямоугольной мембраны.

ЗАДАЧА 8.7. Уравнение колебаний мембраны имеет вид:

$$\frac{\partial^2 u}{\partial t^2} = a^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right),$$

$$u(0, y, t) = u(l_1, y, t) = u(x, 0, t) = u(x, l_2, t) = 0,$$

$$u(x, y, 0) = Axy(l_1 - x)(l_2 - y),$$

$$u_t(x, y, 0) = 0.$$

Пусть $l_1 = 2$, $l_2 = 3$, $A = 1$, $a = 2$.

На первом этапе задаем область решения (рис. 8.58).

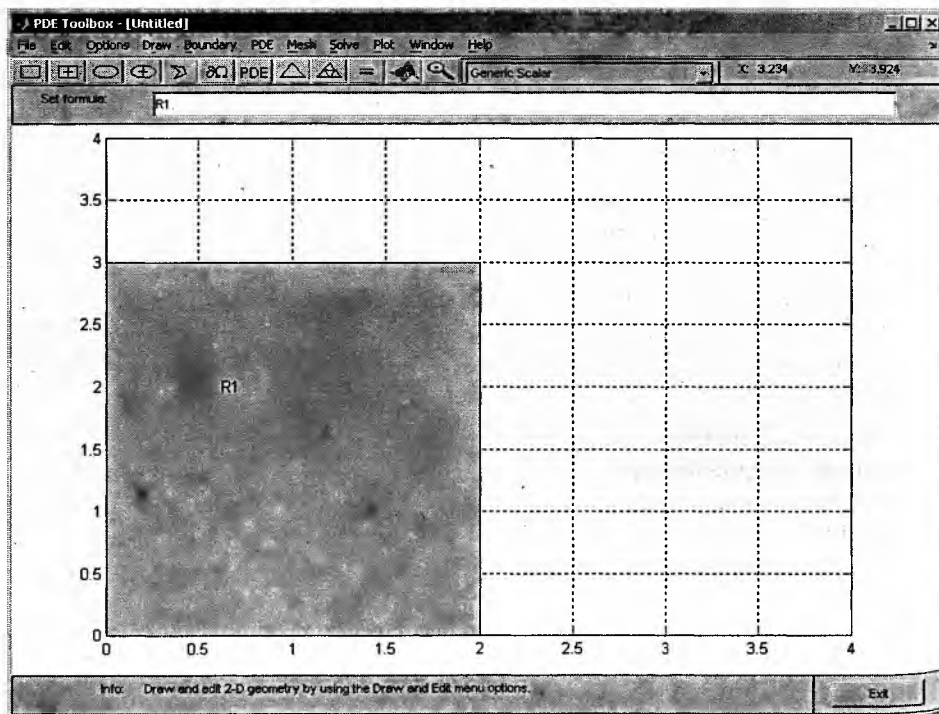


Рис. 8.58 ▾ Область решения задачи 8.7

Далее определяем гиперболическое уравнение и его коэффициенты (рис. 8.59).

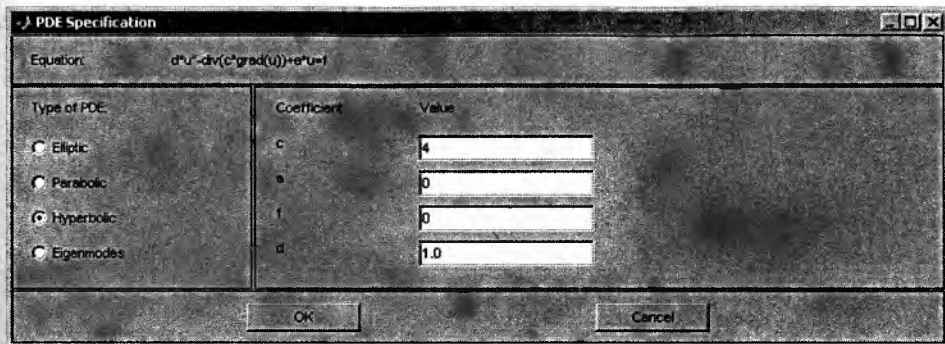


Рис. 8.59 ▼ Определение параметров гиперболического уравнения задачи 8.7

Определяем нулевые условия на границе области (рис. 8.60).

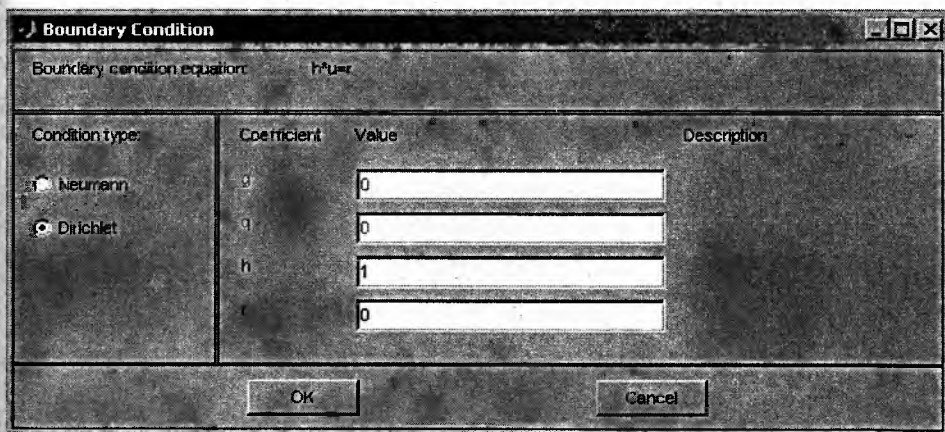


Рис. 8.60 ▼ Граничные условия задачи 8.7

В отличие от рассмотренных ранее эллиптических уравнений в этой задаче необходимо с помощью команды **Solve ► Solve Parameters** (Решение ► Параметры решения) определить интервал изменения времени, начальные условия и точность решения (рис. 8.61)¹.

¹ Обратите внимание на использование операций поэлементного умножения в соответствии с синтаксисом MATLAB.

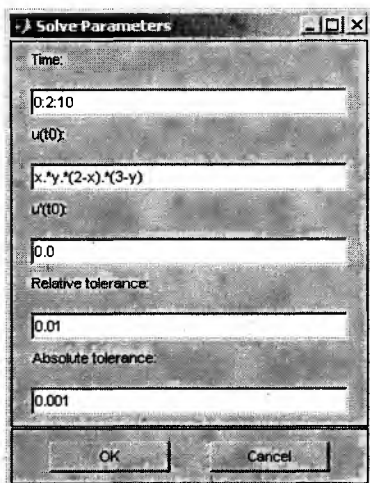


Рис. 8.61 ▼ Начальные условия задачи 8.7

После проведения триангуляции можно запускать задачу на выполнение, но для того, чтобы увидеть процесс колебания мембраны в динамике, нужно включить параметр **Animation** (Анимация) в окне **Plot Selection** (Выделение графика). Далее с помощью кнопки **Options** (Настройки) в окне **Animation Options** (Настройки анимации) – рис.8.62 – установим число кадров в секунду (параметр **Animation rate (fps)**) и число повторов анимации (параметр **Number of repeats**). Теперь при решении задачи пользователь будет видеть динамику процесса колебания мембраны в виде анимации. График решения при $t = 10$ приведен на рис. 8.63.

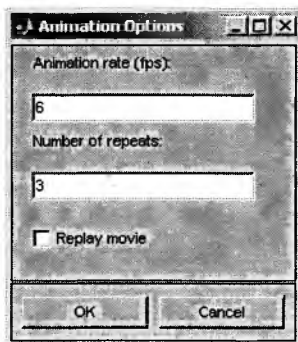


Рис. 8.62 ▼ Параметры анимации

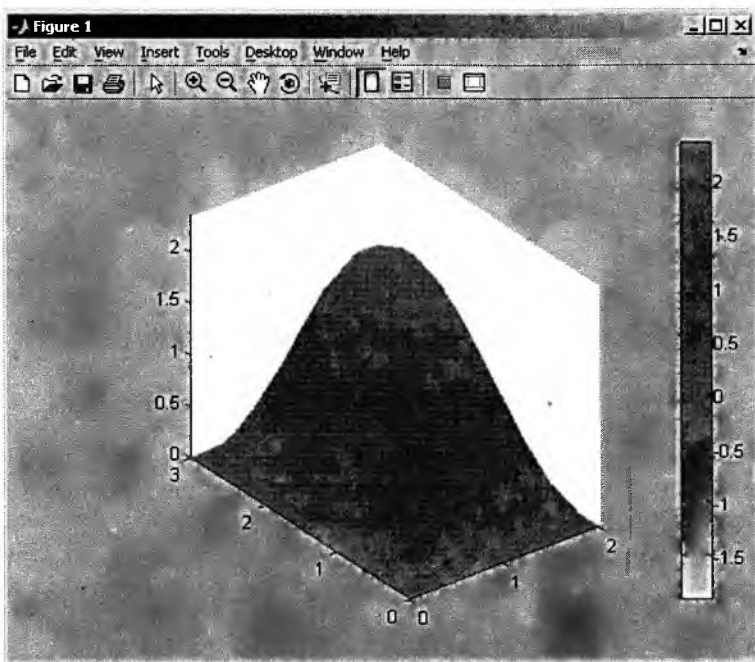


Рис. 8.63 ▾ График решения задачи о колебаниях мембраны

PDE Specification

Equation: $\partial^2 u - \text{div}(c \cdot \text{grad}(u)) + a \cdot u = f$

Type of PDE:	Coefficient	Value	Value
☐ Elliptic	c11, c12	1.0	0.0
☐ Parabolic	c21, c22	0.0	1.0
☒ Hyperbolic	a11, a12	0.0	0.0
☐ Eigenmodes	a21, a22	0.0	0.0
	f1, f2	1.0	1.0
	d11, d12	1.0	0.0
	d21, d22	0.0	1.0

OK Cancel

Рис. 8.64 ▾ Окно определения коэффициентов систем дифференциальных уравнений в частных производных

Отметим, что `pdeTool` позволяет решать и системы двух дифференциальных уравнений в частных производных. Для этого с помощью пункта меню **Options ► Application** (Настройки ► Приложение) необходимо установить режим решения систем **Generic System** (Обобщенная система). Окно для ввода уравнений и коэффициентов представлено на рис. 8.64.

На этом мы заканчиваем знакомство с `pdeTool`. Для более полного описания параметров графической среды `pdeTool` авторы рекомендуют читателю обратиться к книге [4] или к англоязычной документации к пакету.

8.6. Использование Maple для решения уравнений математической физики

Maple менее, чем Mathcad и MATLAB, приспособлен для численного решения дифференциальных уравнений в частных производных, зато с его помощью можно получить аналитическое решение ряда уравнений математической физики методом разделения переменных.

На рис. 8.65 в качестве примера представлено решение уравнения

$$\frac{\partial^2 u}{\partial t^2} = b^2 \frac{\partial^2 u}{\partial x^2} + x(t - a)^2$$

методом разделения переменных. Здесь `_C1`, `_C2`, `_C3`, `_c1` – переменные интегрирования.

Для более детального изучения возможностей Maple при аналитическом решении уравнений математической физики авторы рекомендуют ознакомиться с работами [8, 10].

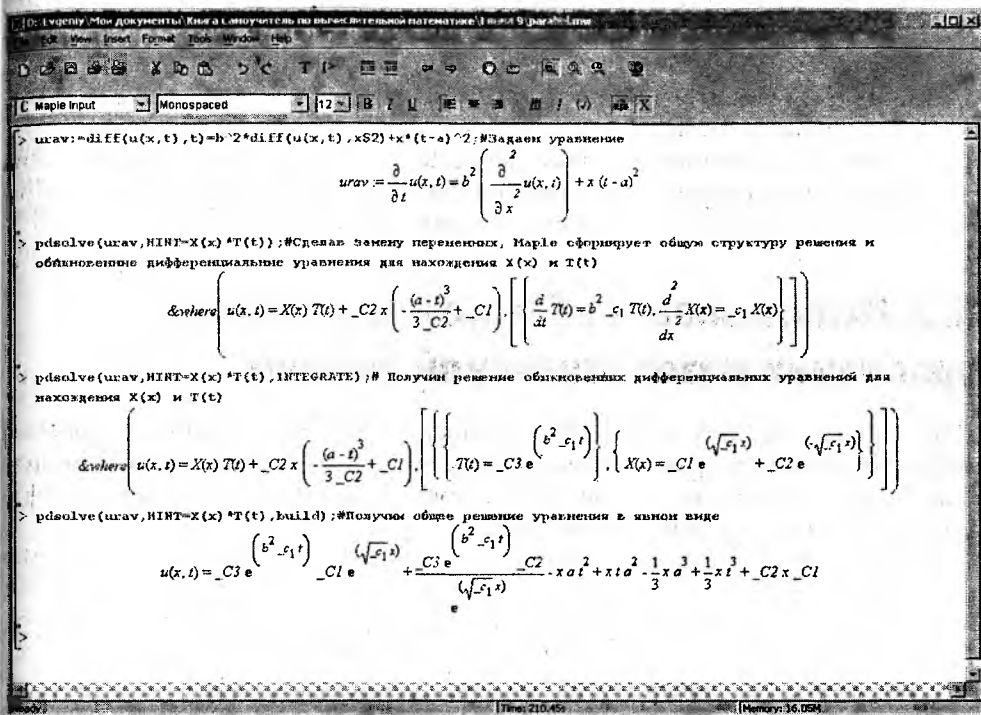


Рис. 8.65 ▼ Решение уравнения $\frac{\partial^2 u}{\partial t^2} = b^2 \frac{\partial^2 u}{\partial x^2} + x(t-a)^2$ методом разделения переменных в Maple

8.7. Советы по использованию математических пакетов при решении дифференциальных уравнений в частных производных

Все три пакета можно использовать для решения уравнений математической физики. Однако, на взгляд авторов, для решения гиперболических и параболических уравнений в частных производных с одной пространственной переменной лучше пользоваться встроенными функциями Mathcad `pdsolve` и `numol`. По мнению авторов, функция `pdsolve` обладает более наглядным интерфейсом, поэтому рекомендуем использовать именно ее для решения уравнений математической

физики с одной пространственной переменной. Метод сеток можно реализовать как в пакете Mathcad, так и в MATLAB. Для решения уравнений в частных производных на плоскости (с двумя пространственными переменными) лучше пользоваться графической средой pdetool, входящей в состав MATLAB. Для аналитического решения уравнений математической физики с постоянными коэффициентами его можно предварительно получить в пакете Maple. При решении трехмерных задач лучше использовать специализированные пакеты для решения уравнений в частных производных FemLab и FlexPDE. Интерфейс пакета Femlab аналогичен описанной в разделе 8.5 графической среде pdetool.

8.8. Задания для самостоятельного решения

1. Решить задачу 8.4 методом сеток.
2. Решить начально-граничную задачу:

$$\begin{aligned}u_{tt} &= a^2 u_{xx}, \\ u(0, t) &= 0, \quad u_x(l, t) = \frac{A}{ES} \sin \omega t, \\ u(x, 0) &= 0, \quad u_t(x, 0) = 0\end{aligned}$$

с помощью функции pdsolve.

3. Решить начально-граничную задачу:

$$\begin{aligned}u_{tt} &= a^2 u_{xx}, \\ u(0, t) &= 0, \quad u_x(l, t) = \frac{A}{ES} \sin \omega t, \\ u(x, 0) &= 0, \quad u_t(x, 0) = 0\end{aligned}$$

методом сеток, запрограммировав его в MATLAB.

4. Решить начально-граничную задачу:

$$\begin{aligned}u_{tt} &= a^2 u_{xx} + \omega^2(x + u) + g \sin(\omega t), \\ u(0, t) &= 0, \quad u_x(l, t) = 0, \\ u(x, 0) &= 0, \quad u_t(x, 0) = 0\end{aligned}$$

с помощью функции pdsolve.

5. Решить начально-граничную задачу:

$$\begin{aligned}u_{tt} &= a^2 u_{xx} + \omega^2(x + u) + g \sin(\omega t), \\ u(0, t) &= 0, \quad u_x(l, t) = 0, \\ u(x, 0) &= 0, \quad u_t(x, 0) = 0\end{aligned}$$

методом сеток, запрограммировав его в MATLAB.

6. Решить начально-граничную задачу:

$$\begin{aligned} u_{tt} &= a^2 u_{xx}, \\ u_x(0, t) &= 0, \quad u_x(l, t) + hu(l, t) = 0, \\ u(x, 0) &= \varphi(x), \quad u_t(x, 0) = \psi(x) \end{aligned}$$

с помощью неявной разностной схемы.

7. Решить начально-граничную задачу:

$$\begin{aligned} u_t &= a^2 u_{xx} + f(x, t), \\ u_x(0, t) - hu(0, t) &= \psi_1(t), \quad u_x(l, t) + hu(l, t) = \psi_2(t), \\ u(x, 0) &= \varphi(x) \end{aligned}$$

методом сеток, запрограммировав его в MATLAB.

8. Решить начально-граничную задачу:

$$\begin{aligned} u_t &= a^2 u_{xx}, \\ u(0, t) &= 0, \quad u(l, t) = A \cos(\omega t), \\ u(x, 0) &= \varphi(x) \end{aligned}$$

с помощью явной разностной схемы, запрограммировав ее в MATLAB.

9. Решить начально-граничную задачу:

$$\begin{aligned} u_t &= a^2 u_{xx}, \\ u(0, t) &= 0, \quad u(l, t) = A \cos(\omega t), \\ u(x, 0) &= \varphi(x) \end{aligned}$$

с помощью неявной разностной схемы, запрограммировав ее в MATLAB

10. Решить начально-граничную задачу:

$$\begin{aligned} u_t &= a^2 u_{xx}, \\ u(0, t) &= 0, \quad u(l, t) = A \cos(\omega t), \\ u(x, 0) &= \varphi(x) \end{aligned}$$

используя функции `numol` и `pdesolve`.



Решение интегральных уравнений

В первом разделе приведены общие сведения об интегральных уравнениях. Метод конечных сумм подробно описан во втором разделе. Там же приведены блок-схемы решения интегральных уравнений Фредгольма и Вольтерра второго рода. Приведены два алгоритма решения уравнения Вольтерра первого рода. Последующие разделы посвящены применению математических пакетов для решения интегральных уравнений методом конечных сумм.

9.1. Общие сведения об интегральных уравнениях

Интегральные уравнения являются важным инструментом математического исследования в задачах техники, теоретической физики и ряда других отраслей науки. *Интегральное уравнение* – уравнение, содержащее неизвестную функцию $y(x)$ под знаком интеграла. В этой книге будут рассматриваться только линейные интегральные уравнения¹.

Рассмотрим основные классы интегральных уравнений [9, 11].

Интегральным уравнением Фредгольма первого рода называется уравнение вида:

$$\int_c^d K(x, s) y(s) ds = f(x). \quad (9.1)$$

¹ В линейные интегральные уравнения функция $y(x)$ входит в первой степени (линейно).

Здесь функции $K(x, s)$, называемая ядром, и $f(x)$ – известные функции. Уравнение

$$y(x) - \lambda \int_c^d K(x, s) y(s) ds = f(x) \quad (9.2)$$

называется *интегральным уравнением Фредгольма второго рода*.

В теории интегральных уравнений параметр λ введен для упрощения анализа существования решения того или иного уравнения. Анализировать параметр λ проще, чем функцию $K(x, s)$ (ядро интегрального уравнения).

Интегральные уравнения Фредгольма первого и второго рода можно обобщить в виде интегрального уравнения *Фредгольма третьего рода*:

$$h(x) y(x) - \lambda \int_c^d K(x, s) y(s) ds = f(x). \quad (9.3)$$

Если $f(x) = 0$, то интегральные уравнения Фредгольма называются *однородными*.

$$h(x) y(x) = \lambda \int_c^d K(x, s) y(s) ds. \quad (9.4)$$

Однородное уравнение Фредгольма второго рода

$$y(x) = \lambda \int_c^d K(x, s) y(s) ds \quad (9.5)$$

допускает тривиальное решение: $y(s) \equiv 0$. Значения λ , при которых уравнение (9.5) имеет нетривиальные решения, называются *собственными числами* (значениями) ядра $K(x, s)$ или уравнения (9.2), а сами нетривиальные решения – *собственными функциями*.

Из теоремы Фредгольма [11] следует:

- если λ является собственным числом ядра $K(x, s)$ то неоднородное уравнение (9.2) не имеет решений или имеет бесконечное множество решений;
- если λ не является собственным числом ядра $K(x, s)$ то неоднородное уравнение (9.2) имеет единственное непрерывное решение $y(x)$.

Важную роль играют уравнения Фредгольма с симметрическим ядром $K(x, s) = K(s, x)$, которое обладает следующими свойствами:

- в уравнениях с симметрическим ядром существует хотя бы одно собственное значение λ ;

- все собственные значения являются действительными;
- собственные функции $\varphi_i(x)$ и $\varphi_j(x)$ симметрического ядра $K(x, s)$ соответствующие различным собственным значениям λ_i и λ_j , ортогональны между собой на промежутке $[c; d]$

$$\int_c^d \varphi_i(x) \varphi_j(x) dx = 0.$$

Интегральным уравнением Вольтерра первого рода называется уравнение вида:

$$\int_c^x K(x, s) y(s) ds = f(x). \quad (9.6)$$

Уравнение

$$y(x) = \lambda \int_c^x K(x, s) y(s) ds - f(x) \quad (9.7)$$

называется *интегральным уравнением Вольтерра второго рода*.

Интегральным уравнением Вольтерра третьего рода называется уравнение вида:

$$h(x) y(x) - \lambda \int_c^x K(x, s) y(s) ds = f(x). \quad (9.8)$$

Уравнения Вольтерра можно свести к уравнениям Фредгольма, введя функцию

$$Q(x, s) = \begin{cases} K(x, s), & \text{если } s \in [a, x], \\ 0, & \text{если } s > x. \end{cases} \quad (9.9)$$

Уравнения Вольтерра можно записать в виде уравнений Фредгольма с ядром $Q(x, s)$. Таким образом, теория уравнений Вольтерра сводится к теории уравнений Фредгольма с ядром $Q(x, s)$ (см. (9.9)). Однако в ряде случаев уравнения Вольтерра исследуются своими методами. Рассмотрим численные методы решения интегральных уравнений.

9.2. Численные методы решения интегральных уравнений

Одним из основных численных методов решения интегральных уравнений является *метод конечных сумм* [11], который основывается на вычислении интеграла с помощью одной из приближенных квадратурных формул (см. главу 4):

$$\int_c^d f(x) dx = \sum_{i=1}^n A_i f(x_i). \quad (9.10)$$

¹ Это могут быть и другие узлы из интервала $[c, d]$. Необязательно, что $x_1 = c$ и $x_n = d$.

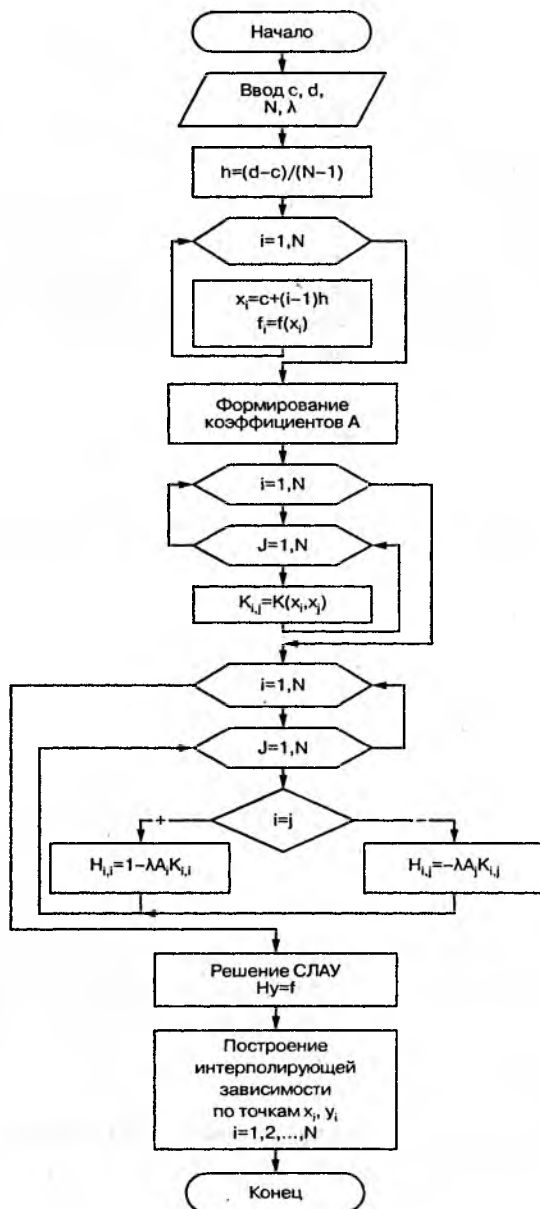


Рис. 9.1 ▽ Блок-схема решения Фредгольма второго рода методом конечных сумм

Решив систему (9.14), найдем значение уравнения (9.2) в узловых точках x_i . Для нахождения значения y в других точках можно аппроксимировать или интерполировать таблицу значений $(x_i, y_i, i = 1, 2, \dots, n)$.

На рис. 9.1 представлена блок-схема решения уравнения Фредгольма второго рода методом конечных сумм.

На рис. 9.3 представлена реализация этого алгоритма в Mathcad, а в листинге 9.1 – в MATLAB.

Рассмотрим, какой вид примет система (9.13)–(9.14) в случае применения в качестве (9.10) формул прямоугольников, трапеций и Симпсона.

В случае использования в качестве метода интегрирования метода прямоугольников система (9.14) принимает вид:

$$\begin{aligned} Hy &= f \\ H_{i,j} &= \begin{cases} 1 - \lambda h K_{i,i}, & i = j \\ -\lambda h K_{i,j}, & i \neq j \end{cases}, i, j = 1, 2, \dots, n-1. \end{aligned} \quad (9.15)$$

Для метода трапеций систему (9.14) можно записать следующим образом:

$$\begin{aligned} Hy &= f \\ H_{i,j} &= \begin{cases} 1 - \lambda h K_{i,i}, & i = j \\ -\lambda h K_{i,j}, & i \neq j \end{cases}, i, j = 2, 3, \dots, n-1 \\ &\quad \begin{cases} 1 - \lambda \frac{h}{2} K_{i,i}, & i = j \\ -\lambda \frac{h}{2} K_{i,j}, & i \neq j \end{cases}, i, j = 1 \text{ или } i, j = n \end{aligned} \quad (9.16)$$

В случае использования в качестве (9.10) формулы Симпсона следует учитывать, что

$$n = 2m + 1; A_1 = A_{2m+1} = \frac{h}{3}; A_2 = A_4 = \dots A_{2m} = \frac{4h}{3}; A_3 = A_5 = \dots A_{2m-1} = \frac{2h}{3}. \quad (9.17)$$

Для решения уравнения Фредгольма первого рода методом конечных сумм можно подставить формулу (9.10) в уравнение (9.1). Однако получаемая система может оказаться некорректной. Поэтому к исходному интегральному уравнению необходимо предварительно применить регуляризацию. Конкретные алгоритмы регуляризации описаны в [9].

9.2.2. Использование метода конечных сумм при решении интегральных уравнений Вольтерра

Уравнение Вольтерра можно решать как уравнение Фредгольма с ядром $Q(t,s)$ (9.9) [9, 11]. Рассмотрим этот процесс подробнее.

Разобьем интервал $[c, d]$ с помощью последовательности значений $x_1, x_2, \dots, x_n$.

$$x_1 \geq c, x_2 = x_1 + h, \dots, x_i = x_{i-1} + h, \dots, x_n \leq d, h = \frac{x_n - x_1}{n-1}. \quad (9.18)$$

Рассмотрим последовательность интегральных равенств, получаемых после подстановки последовательности значений $x_1, x_2, \dots, x_n$ в качестве верхнего предела интегрирования в уравнения Вольтерра второго рода [9, 11]:

$$y(x_i) = \lambda \int_c^{x_i} K(x_i, s) y(s) ds - f(x_i). \quad (9.19)$$

Заменим интеграл в формуле (9.19) интегральной суммой (9.10) и получим следующее соотношение:

$$y_i = \lambda \sum_{j=1}^{n_i} A_{i,j} K_{i,j} y_j - f_i. \quad (9.20)$$

Здесь n_i – количество точек, на которые разбит интервал, $A_{i,j}$ – числовые коэффициенты квадратурной формулы на интервале $[c, x_i]$, $K_{i,j}$ – значения ядра в точке (x_i, s_j) , s_j – узловые точки при интегрировании на интервале $[c, x_i]$.

Если взять последовательность x_i , такую что

$$x_1 = c, x_2 = x_1 + h, \dots, x_i = x_{i-1} + h, \dots, x_n = d, h = \frac{d-c}{n-1},$$

и значения s_j , совпадающие с x_i $\left(s_1 = c, s_2 = x_1 + h, \dots, s_i = x_{i-1} + h, \dots, s_n = d, h = \frac{d-c}{n-1} \right)$, то система уравнений (9.20) значительно упростится и примет вид [9, 11]:

$$y_i = \lambda \sum_{j=1}^i A_{j,j} K_{i,j} y_j - f_i; i = 1, 2, \dots, n. \quad (9.21)$$

Система (9.21) является треугольной и решается методом, аналогичным обратному ходу метода Гаусса (см. главу 1). Перепишем систему (9.21) в компонентной форме:

$$\begin{cases} (1 - \lambda A_1 K_{1,1} y_1) = -f_1, \\ -\lambda A_1 K_{2,1} y_1 + (1 - \lambda A_2 K_{2,2}) y_2 = -f_2, \\ \dots \\ -\lambda A_1 K_{n,1} y_1 - \lambda A_2 K_{n,2} y_2 - \dots + (1 - \lambda A_n K_{n,n}) y_n = -f_n. \end{cases} \quad (9.22)$$

Решение треугольной системы (9.22) можно записать с помощью формулы [9, 11]:

$$y_i = \frac{-f_i + \lambda \sum_{j=1}^{i-1} A_i K_{i,j} y_j}{1 - \lambda A_i K_{i,i}}; \quad i = 1, 2, \dots, n. \quad (9.23)$$

Совокупность всех коэффициентов A_i в каждой строке системы (9.21)–(9.22) должна быть полной для используемой квадратурной формулы (9.10). Таким образом, для каждого интервала $[c, x_i]$, $i = x_1, x_2, \dots, x_n$ должно выполняться соотношение $\sum_{j=1}^i A_j = t_i - a$ [9].

При выборе конкретного вида формулы (9.10) для решения уравнения Вольтерра второго рода предпочтительнее использовать квадратурные формулы замкнутого типа, например формулы прямоугольников или трапеций [9].

После решения системы уравнений (9.22) по формулам (9.23) для нахождения значения функции $y(x)$ в других точках необходимо аппроксимировать или интерполировать полученную таблицу значений $(x_i, y_i, i = 1, 2, \dots, n)$.

На рис 9.2 представлена блок-схема решения уравнения Вольтерра второго рода.

На рис. 9.4 представлена реализация этого алгоритма в Mathcad, а в листинге 9.2 – в MATLAB.

Рассмотрим особенности решения уравнения Вольтерра первого рода методом конечных сумм. Сформируем последовательность $x_1, x_2, \dots, x_n$ такую, что $x_2 = x_1 + h, \dots, x_i = x_{i-1} + h, \dots, x_n$. Сформируем последовательность интегральных сумм:

$$\lambda \int_c^{x_i} K(x_i, s) y(s) ds = f(x_i), \quad (9.24)$$

заменяв интеграл, входящий в (9.24), квадратурной формулой (9.10) и выбирая значения s_j , не совпадающие с x_i . Решения y_j будем находить именно в узлах s_j . В качестве квадратурной формулы необходимо выбрать формулу именно с узлами s_j . Поэтому, если в качестве узлов s выбрать $s_i = x_i - \frac{h}{2}$, то в качестве квадратурной формулы можно будет использовать формулу средних прямоугольников¹. Подставляя формулу средних прямоугольников в качестве квадратурной формулы (9.10) в формулу (9.24), получим следующую систему уравнений:

¹ Это может быть и другая формула; единственное требование к ней – она должна использовать значение не в узлах x_i , а между ними.

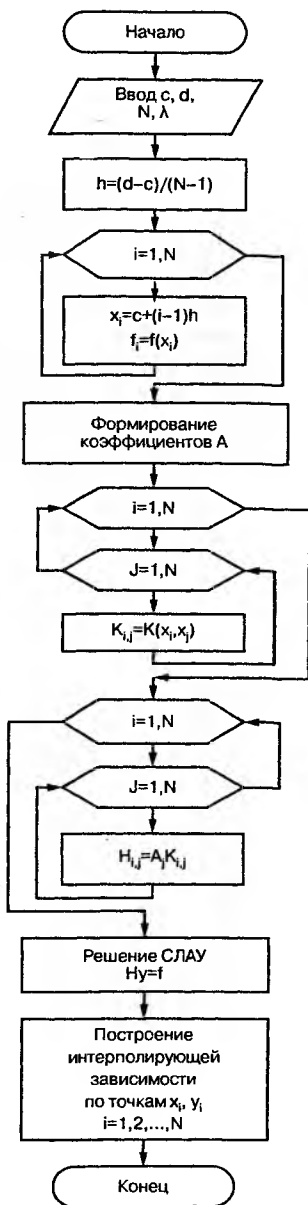


Рис. 9.2 ▼ Блок-схема решения уравнения Вольтерра второго рода

$$\begin{cases} h K_{1,1} y_1 = F_1, \\ h K_{2,1} y_1 + h K_{2,2} y_2 = F_2, \\ h K_{3,1} y_1 + h K_{3,2} y_2 + h K_{3,3} y_3 = F_3, \\ \dots \\ h K_{n,1} y_1 + h K_{n,2} y_2 + \dots + h K_{n,n} y_n = F_n. \end{cases} \quad (9.25)$$

Здесь $K_{i,j} = K(x_i, s_j)$, $F_i = \frac{f(x_i)}{\lambda}$, исключение λ из явного вида системы (9.25) связано с тем, что в уравнении Вольтерра первого рода коэффициент λ не играет столь важной роли, как в уравнении второго рода. Полученная система (9.25), как и система (9.21), является треугольной и решается методом, аналогичным обратному ходу метода Гаусса. Решение ее представлено формулой (9.26):

$$y_i = \frac{F_i - h \sum_{j=1}^{i-1} K_{i,j} y_j}{h K_{i,i}}. \quad (9.26)$$

Найдем значение функции y_i в узловых точках s_p , после чего для нахождения значения u в других точках можно аппроксимировать или интерполировать таблицу значений (s_p, y_p) .

В случае если узлы s_j и x_i совпадают $s_1 = x_1 = c$, $s_2 = x_2 = x_1 + h$, ..., $s_i = x_i = x_{i-1} + h$, ..., $s_n = x_n = d$, $h = \frac{d-c}{n-1}$, то при использовании рассмотренной методики расчета по формулам (9.24)–(9.26) возникает проблема вычисления значения y_1 . Это связано с тем, что формула (9.24) при $x_1 = c$ теряет смысл и не позволяет найти y_1 . В этом случае можно продифференцировать уравнение (9.7) по x и при $x = c$ можно записать следующую приближенную формулу для нахождения y_1 [9]:

$$y_1 = \frac{F'(c)}{K_{1,1}}; \quad F(t) = \frac{f(t)}{\lambda}. \quad (9.27)$$

После чего, подставив в (9.24) при $i = 2, 3, \dots, n$ конкретную квадратурную формулу (9.10), получим систему линейных алгебраических уравнений:

$$\sum_{j=1}^{i-1} A_j K_{i,j} y_j = F_i; \quad i = 2, 3, \dots, n, \quad (9.28)$$

решение которой с учетом соотношения (9.27) имеет вид:

$$y_i = \frac{F_i - \sum_{j=1}^{i-1} A_j K_{i,j} y_j}{A_i K_{i,i}}; \quad i = 2, 3, \dots, n. \quad (9.29)$$

Уравнения Вольтерра первого рода можно решить двумя способами. Первый способ состоит из следующих этапов:

1. Формируем последовательность значений

$$x_1 = c + h, x_2 = x_1 + h, \dots, x_i = x_{i-1} + h, \dots, x_n, h = \frac{d - c}{n}.$$

2. Формируем последовательность значений $s_i = x_i - \frac{h}{2}$.

3. Рассчитываем коэффициенты $K_{i,j} = K(x_i, s_j)$, $F_i = \frac{f(x_i)}{\lambda}$.

4. Вычисляем значения $y_i = \frac{F_i - h \sum_{j=1}^{i-1} K_{i,j} y_j}{h K_{i,i}}$.

5. Интерполируем таблицу значений (s_i, y_i) .

Реализация этого способа решения интегрального уравнения Вольтерра первого рода в Maple представлена на рис. 9.5.

Для решения интегрального уравнения Вольтерра первого рода вторым способом необходимо:

1. Сформировать последовательность значений

$$x_1 = c, x_2 = x_1 + h, \dots, x_i = x_{i-1} + h, \dots, x_n = d, h = \frac{d - c}{n - 1}.$$

2. Рассчитать значения коэффициентов $K_{i,j} = K(x_i, s_j)$, $F_i = \frac{f(x_i)}{\lambda}$.

3. Вычислить приближенное значение $y_1 = \frac{F'(a)}{K_{1,1}}$; $F(t) = \frac{f(t)}{\lambda}$.

4. Вычислить остальные значения y по формуле:

$$y_i = \frac{F_i - \sum_{j=1}^{i-1} A_j K_{i,j} y_j}{A_i K_{i,i}}; i = 2, 3, \dots, n.$$

6. Построить интерполирующую функцию по таблице значений (x_i, y_i) .

Рассмотрим решение интегральных уравнений в математических пакетах.

9.3. Решение интегральных уравнений в Mathcad

В Mathcad отсутствуют встроенные средства для решения интегральных уравнений, однако встроенные средства пакета позволяют реализовать описанный в предыдущем разделе метод конечных сумм. Рассмотрим решение интегральных уравнений в Mathcad на примере конкретных задач.

ЗАДАЧА 9.1. Решить интегральное уравнение:

$$y'(x) - 4 \int_0^{\frac{\pi}{2}} \sin^2(x) y(s) ds = 2x - \pi. \quad (9.26)$$

ORIGIN := 1 Определяем интервал интегрирования [c,d] и коэффициент λ

$$\lambda := 4 \quad c := 0 \quad d := \frac{\pi}{2}$$

Определяем ядро $k(x,s)$ и правую часть $f(x)$ интегрального уравнения Фредгольма второго рода

$$k(x,s) := \sin(x)^2 \quad f(x) := 2 \cdot x - \pi$$

Определяем количество точек $N := 11$ на интервале

$$\text{интегрирования и шаг } h := \frac{d - c}{N - 1}$$

Формируем массив узловых точек x интервала [c,d]

$$i := 1..N \quad x_i := c + (i - 1) \cdot h$$

и коэффициенты $F(i)$ $F_i := f(x_i)$

Для интегрирования будем использовать метод трапеций, поэтому коэффициенты A будут такими

$$A_1 := \frac{h}{2} \quad A_N := \frac{h}{2} \quad i := 2..10 \quad A_i := h$$

Формируем матрицы коэффициентов K и H

$$i := 1..N \quad j := 1..N \quad K_{i,j} := k(x_i, x_j) \quad H_{i,j} := \text{if}(i = j, 1 - \lambda \cdot A_i \cdot K_{i,i}, -\lambda \cdot A_j \cdot K_{i,j})$$

Решая СЛАУ $H y = F$, вычисляем значение искомой функции в узловых точках $y := H^{-1} \cdot F$

$$y^T = (-3.1416 \quad -2.7147 \quad -2.0732 \quad -1.2493 \quad -0.2927 \quad 0.7335 \quad 1.7597 \quad 2.7162 \quad 3.5401 \quad 4.1816 \quad 4.6085)$$

$$\text{Точное решение } \text{resh}(t) := \frac{\pi^2 \cdot \sin(t)^2}{\pi - 1} + 2 \cdot t - \pi$$

$$\text{resh}(x)^T =$$

$$(-3.1416 \quad -2.7147 \quad -2.0732 \quad -1.2493 \quad -0.2927 \quad 0.7335 \quad 1.7597 \quad 2.7162 \quad 3.5401 \quad 4.1816 \quad 4.6085)$$

Рис. 9.3 ▽ Решение уравнения (9.26) методом конечных сумм в Mathcad

Это уравнение является интегральным уравнением Фредгольма второго рода. Оно имеет точное решение: $y(x) = \frac{\pi^2 \sin^2(x)}{\pi - 1} + 2x - \pi$. На рис. 9.3 представлено решение уравнения (9.26) в Mathcad методом конечных сумм (блок-схема алгоритма приведена на рис. 9.1). Там же приведены точные значения в узловых точках. В качестве квадратурной формулы (9.10) использовался метод трапеций.

Решение уравнения Вольтерра рассмотрим на примере следующей задачи.

ЗАДАЧА 9.2. Решить интегральное уравнение:

$$y(x) - \int_0^x x \cos^2(xs^3) y(s) ds = x^2 - \frac{1}{3} \lg(x^4). \quad (9.27)$$

ORIGIN := 1 Определяем интервал интегрирования [c,d] и коэффициент λ

$$\lambda := 1 \quad c := 0 \quad d := 0.6$$

Определяем ядро $k(x,s)$ и правую часть $f(x)$ интегрального уравнения Фредгольма второго рода

$$k(x,s) := x \cdot \cos(xs^3)^2 \quad f(x) := -x^2 + \frac{1}{3} \cdot \tan(x^4)$$

Определяем количество точек $N := 9$ на интервале

$$\text{интегрирования и шаг } h := \frac{d - c}{N - 1}$$

Формируем массив узловых точек x интервала $[c,d]$

$$i := 1..N \quad x_i := c + (i - 1) \cdot h$$

и коэффициенты $F(i)$ $F_i := f(x_i)$

Для интегрирования будем использовать метод Симпсона, поэтому коэффициенты A будут такими

$$A_1 := \frac{h}{3} \quad A_N := \frac{h}{3} \quad i := 2, 4..N - 1 \quad A_i := 4 \cdot \frac{h}{3} \quad j := 3, 5..N - 2 \quad A_j := 2 \cdot \frac{h}{3}$$

Формируем матрицу коэффициентов K и H

$$i := 1..N \quad j := 1..N \quad K_{i,j} := k(x_i, x_j)$$

$$y_i := \frac{-F_i - \lambda \cdot \sum_{j=1}^N \text{if}(j \geq i, 0, A_j \cdot K_{i,j} \cdot y_j)}{1 - \lambda \cdot A_i \cdot K_{i,i}}$$

$$y^T = (0 \quad 5.657 \times 10^{-3} \quad 0.022 \quad 0.051 \quad 0.087 \quad 0.135 \quad 0.182 \quad 0.245 \quad 0.286)$$

$$\text{Точное решение } \text{resh}(t) := t^2 \quad \text{resh}(x)^T =$$

$$(0 \quad 5.625 \times 10^{-3} \quad 0.0225 \quad 0.0506 \quad 0.09 \quad 0.1406 \quad 0.2025 \quad 0.2756 \quad 0.36)$$

Рис. 9.4 ▼ Решение уравнение (9.27) методом конечных сумм в Mathcad

Уравнение (9.27) является уравнением Вольтерра второго рода. Точное решение уравнения: $y(x) = x^2$. На рис. 9.4 представлено решение уравнения (9.27) в Mathcad методом конечных сумм (блок-схема алгоритма приведена на рис. 9.2). Там же приведены точные значения в узловых точках. В качестве квадратурной формулы (9.10) использовался метод Симпсона. Обращаем внимание читателя на то, что при решении уравнений Вольтерра (в отличие от уравнений Фредгольма) с удалением от точки c точность замены интеграла квадратурной формулы уменьшается и вычисленные значения все больше отличаются от точного решения.

9.4. Решение интегральных уравнений в MATLAB

Рассмотрим реализацию алгоритмов, описанных в разделе 9.2, в MATLAB, в виде функций пользователя. В листингах 9.1–9.2 представлены функции, предназначенные для решения задач 9.1 и 9.2 соответственно.

Листинг 9.1

```
% Функция решает интегральное уравнение Фредгольма 2-го рода
function y=fredgolm(c,d,N,lyam)
% Вычисляем шаг по x
h=(d-c)/(N-1);
% Формируем массив x и значения коэффициентов F
for i=1:N
    x(i)=c+(i-1)*h;
    F(i)=f(x(i));
end
% Вычисляем коэффициенты квадратурной формулы, в данном случае метод
% трапеций.
A(1)=h/2;A(N)=h/2;
for i=2:N-1
    A(i)=h;
end
%Формируем матрицу K
for i=1:N
    for j=1:N
        K(i,j)=k(x(i),x(j));
    end
end
%Формируем матрицу H
for i=1:N
    for j=1:N
```

```

    if i==j
        H(i,j)=1-lyam*A(i)*K(i,i);
    else
        H(i,j)=-lyam*A(j)*K(i,j);
    end
end

```

```
end
```

```

% Решив СЛАУ, вычисляем значение y в узловых точках
y=H^(-1)*F';
% Строим график решения
plot(x,y);
title("Решение интегрального уравнения второго рода");
grid on;
end

```

Листинг 9.2

```

% Функция решает интегральное уравнение Вольтерра 2-го рода
function y=voltera(c,d,N,lyam)
% Вычисляем шаг по x
h=(d-c)/(N-1);
% Формируем массив x и значения коэффициентов F
for i=1:N
    x(i)=c+(i-1)*h;
    F(i)=f(x(i));
end
% Вычисляем коэффициенты квадратурной формулы, в данном случае метод
% трапеций.
A(1)=h/2;A(N)=h/2;
for i=2:N-1
    A(i)=h;
end
%Формируем матрицу K
for i=1:N
    for j=1:N
        K(i,j)=k(x(i),x(j));
    end
end
end
% Вычисляем значение y в узловых точках по формуле (9.23)
for i=1:N
    s=0;
    for j=1:i-1
        s=s+A(j)*K(i,j)*y(j);
    end
    y(i)=(-F(i)-s*lyam)/(1-lyam*A(i)*K(i,i))
end
plot(x,y);
title("Решение интегрального уравнения второго рода");
grid on;
end

```

9.5. Решение интегральных уравнений в Maple

Рассмотрим использование Maple для решения интегральных уравнений на примере следующей задачи [9].

ЗАДАЧА 9.3. Решить интегральное уравнение [9]:

$$\int_1^x \left(x^2 + s^2 + 1 \right) x(s) ds = x^2 - \frac{1}{x}$$

на интервале [1; 1.5]. Это уравнение Вольтерра первого рода с точным решением уравнения $y(x) = \frac{1}{x^2}$.

Решение этой задачи в Maple представлено на рис. 9.5.

```

> restart;
> with(linalg):#Подключаем пакет линейной алгебры
> c:=1;d:=1.5:N:=11:lyam:=1:h:=(d-c)/N:
> x:=array(1..N):F:=array(1..N):s:=array(1..N):y:=array(1..N,[0,0,0,0,0,0,0,0,0,0,0]):
> for i to N do x[i]:=c+i*h: s[i]:=x[i]-h/2: F[i]:=x[i]^2-1/x[i]: od:# Формируем массивы x, s и F
> K:=array(1..N,1..N):
> for i from 1 to N do for j from 1 to N do K[i,j]:=1+x[i]^2+s[j]^2: od: od:# Формируем матрицу K
> y[1]:=F[1]/(h*K[1,1]):#Вычислим первое значение y
> for i from 2 to N do r[i]:=0: for j from 1 to i-1 do r[i]:=x[i]+K[i,j]*y[j]: od: y[i]:=(F[i]-h*r[i])/
[i,i]): od:#Вычисляем остальные значения y по формуле (9.26)
> for i to N do tochn_y[i]:=1/s[i]^2: od:# Точное решение уравнения
> for i to N do print(y[i],tochn_y[i]): od:# Вывод найденного методом конечных сумм и точного решения
интегрального уравнения
0.9521921751, 0.9518143962
0.8656461151, 0.8653326122
0.7903869090, 0.7901234568
0.7245336090, 0.7243096424
0.6665814392, 0.6663890046
0.6153149882, 0.6151480200
0.5697438886, 0.5695977216
0.5290546394, 0.5289256198
0.4925739506, 0.4924592182
0.4597407709, 0.4596380350
0.4300845111, 0.4299919377

```

Рис. 9.5 ▼ Решение интегрального уравнения Вольтерра первого рода в Maple

Как видно из рис. 9.5, точное решение и решение, найденное методом конечных сумм, практически совпадают.

9.6. Советы по применению математических пакетов при решении интегральных уравнений

Рассматриваемые математические пакеты не имеют встроенных функций для численного решения интегральных уравнений. Поэтому решение интегральных уравнений заключается в реализации в пакетах метода конечных сумм. На взгляд авторов, это лучше делать в Mathcad или MATLAB, но больших сложностей нет и в реализации этого метода в Maple. Мы рекомендуем читателю использовать тот пакет, в котором читатель привык работать.

9.7. Задания для самостоятельного решения

Решить интегральные уравнения:

$$1. \quad y(x) - \int_{0,1}^1 e^{xs} y(s) ds = x - \frac{e^x}{x} + \frac{e^x - 1}{x^2}.$$

$$2. \quad y(x) - 4 \int_0^1 xs y(s) ds = 1.$$

$$3. \quad y(x) - \frac{1}{2} \int_0^1 xs y(s) ds = \frac{5}{6}x.$$

$$4. \quad y(x) - 3 \int_0^2 xs y(s) ds = 3x - 2.$$

$$5. \quad y(x) - 6 \int_0^{\frac{\pi}{3}} \sin^2(x) y(s) ds = 3x - \frac{10}{11}\pi.$$

$$6. \quad y(x) - 4 \int_0^{\frac{\pi}{2}} \sin(x) \cos(s) y(s) ds = \sin(x).$$

$$7. \quad y(x) + \int_0^x e^{x-s} y(s) ds = e^x.$$

$$8. \quad y(x) - \int_0^x e^{-2(x-s)} y(s) ds = 1 + x.$$

$$9. \quad \int_0^x e^{2(x-s)} y(s) ds = \sin(x).$$

$$10. \quad \int_0^x e^{x-s} y(s) ds = sh(x).$$

Приложение 1

Введение в Mathcad

Mathcad – один из самых популярных математических пакетов, который позволяет проводить различные вычисления в принятых в математике обозначениях. Кроме того, Mathcad предоставляет широкие возможности для создания и редактирования различных графиков и является идеальным инструментом для выполнения инженерных и математических расчетов.

После запуска Mathcad на экран выводится основное окно приложения (рис. П1.1).

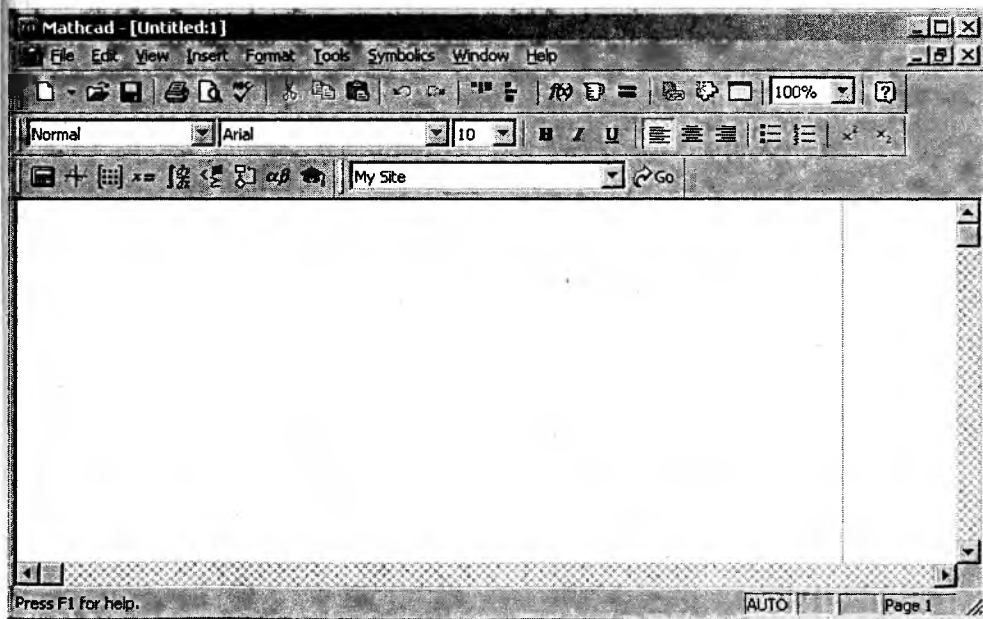


Рис. П1.1 ▼ Окно приложения Mathcad 12

Курсор в виде красного перекрестия указывает место, куда будет вводиться математическое выражение или текст. Курсор передвигается с помощью клавиш $\leftarrow$, $\uparrow$, $\rightarrow$, $\downarrow$ или мыши. В выбранном месте рабочего листа документа можно

ввести математическое выражение и нажать знак $=$ ¹, после чего на экране появится результат вычислений.

Рассмотрим общие принципы *редактирования выражений* в Mathcad. При вводе математических выражений важно понимать роль курсора. Курсор выделяет символ или группу символов, с которыми будет проводиться операция. Для управления курсором служат клавиши **Insert**, **Пробел**, $\leftarrow$, $\rightarrow$.

Оператор воздействует на последний введенный операнд. Для расширения операнда служит клавиша **Пробел**². Курсор может иметь вид $\downarrow$ (в этом случае оператор будет дописываться справа от операнда) или $\uparrow$ (оператор дописывается слева от операнда). Переключение между двумя курсорами осуществляется с помощью клавиши **Insert**.

Клавиша $\leftarrow$ снимает объединение с группы символов, если курсор имеет форму $\downarrow$, а клавиша $\rightarrow$ – с группы символов, когда курсор имеет форму $\uparrow$.

Клавиша **Delete** удаляет символ после курсора, если последний имеет форму $\downarrow$; если же курсор выглядит как $\uparrow$, то эта клавиша выделяет расширенный операнд или удаляет текущий символ. Клавиша **Backspace** удаляет символ перед курсором, если курсор имеет форму $\downarrow$; если же курсор имеет форму $\uparrow$, то она выделяет расширенный операнд или удаляет символ. Выделенный операнд может быть удален с помощью клавиш **Delete** и **Backspace**. Рассмотрим пример.

ПРИМЕР 1. Вычислить значение выражения $\frac{2.3}{4.278 + 21.687} - 8$.

В рабочем листе Mathcad щелчком мышью для обозначения места ввода формулы. После этого последовательно наберем:

1. Число **2.3**³.

2. Символ $\div$.

3. Число **4.278** – на рабочем листе Mathcad возникнет выражение $\frac{2.3}{4.278}$.

4. Символ $+$.

5. Число **21.687** – появится выражение $\frac{2.3}{4.278 + 21.687}$. Далее необходимо,

чтобы оператор вычитания воздействовал на всю дробь. В нашем случае

¹ Для вычисления результата знак $=$ можно набирать не только в конце корректного математического выражения, но и в любом месте этого выражения.

² Расширенным назовем операнд, состоящий более чем из одного числа или более чем из одной переменной.

³ Обратите внимание, что целая и дробная часть числа в Mathcad отделяется точкой независимо от настроек Windows.

курсор выделяет число 21.687. Надо объединить всю дробь в один операнд. Для этого выполните следующие действия:

6. Наберите символ **Пробел**, который объединит $4.278 + 21.687$ – после чего

выражение станет таким:
$$\frac{2.3}{4.278 + 21.687}$$

7. Объедините всю дробь, нажав **Пробел** еще раз
$$\frac{2.3}{4.278 + 21.687}$$

8. Нажмите клавишу вычитания ($-$).

9. Введите число 8.

10. Нажмите клавишу равенства.

В результате рабочий лист Mathcad примет вид, представленный на рис. П1.2.

$$\frac{2.3}{4.78 + 21.687} - 8 = -7.913$$

Рис. П1.2 ▼ Результат решения примера 1 в Mathcad

При более сложных расчетах возникает необходимость использования переменных. *Переменные* задают в рабочем листе документа Mathcad, а затем используют в выражениях, то есть любая переменная до введения в формулы и выражения должна быть определена. Для этого необходимо набрать имя переменной, символ $:=$ ¹ и значение переменной. Пример определения переменных x и y приведен на рис. П1.3.

$$x := 5.32$$

$$y := \frac{\pi}{12}$$

Рис. П1.3 ▼ Пример определения переменных x и y в Mathcad

Имя переменной в Mathcad – совокупность букв (латинских и греческих), цифр и некоторых специальных символов ($_, \%, ", \infty$), начинающаяся с буквы или символа подчеркивания.

Mathcad читает документ сверху вниз и слева направо. Определив переменную, ее можно использовать везде ниже и правее равенства, которым она определена. Чтобы в любой момент увидеть значение переменной, достаточно

¹ Символ $:=$ на экране будет отображаться как $:=$.

набрать ее имя и символ $=$. Если ввести $=$ после еще не определенной переменной, то этот символ отобразится как $:=$.

Mathcad различает прописные и строчные буквы в именах; например, abc , ABC и AbC – это разные переменные.

В Mathcad можно использовать *переменные с индексом*. Для их определения перед индексом следует набрать символ точки $..$. Например, чтобы определить переменную x_{min} , необходимо набрать символ x , затем точку, после чего min . На экране появится x_{min} .

Кроме того, пакет допускает создание так называемой ранжированной переменной. *Ранжированная переменная* – это особый вид переменной, которая принимает множество значений. Она определяется начальным и конечным значениями, а также шагом изменения значения. Для определения ранжированной переменной (например, переменной h , которая изменяется от -3 до 2 с шагом 0.25) в документе Mathcad необходимо выполнить следующее:

1. Ввести имя ранжированной переменной (в нашем случае h) и символ $:=$.
2. Указать начальное значение переменной (в нашем случае это число -3).
3. Через запятую ввести следующее значение переменной (в нашем примере необходимо набрать -2.75).
4. Вставить символ $;$ ¹ и конечное значение переменной (в нашем примере 2).
5. Нажать **Enter**. После этого Mathcad вычисляет все значения, входящие в диапазон. Чтобы увидеть значения созданной ранжированной переменной, достаточно набрать $h=$ (рис. П1.4).

$h := -3, -2.75.. 2$

$h =$

-3
-2.75
-2.5
-2.25
-2
-1.75
-1.5
-1.25
-1
-0.75
-0.5
-0.25
0
0.25
0.5
0.75

Рис. П1.4 ▼ Ранжированная переменная h

¹ Mathcad изображает символ точки с запятой как два символа точки подряд.

Если шаг изменения ранжированной переменной равен 1, то достаточно ввести начальное значение этой переменной, символ ; и конечное значение переменной.

Расчеты в Mathcad производятся в области ввода формул¹. Но кроме этого в Mathcad можно вводить текстовые комментарии и целые области с текстовой информацией. *Текстовая область* создается в том месте, где находится курсор мыши. Вставить текстовую область можно, выполнив последовательность команд **Insert ► Text Region** или нажав символ двойной кавычки.

В созданной области можно вводить текст. Если это русский текст, то необходимо на панели инструментов выбрать шрифт, поддерживающий кириллицу. В Mathcad 12 это должен быть шрифт, явно содержащий слово *Сyr* (например, **Arial Cyr, Times New Roman Cyr**).

В текстовой области используются все принятые в форматированном тексте возможности: изменение шрифта, его размера, установка выравнивания, формирование списков и т.д. Для форматирования текста кроме панели инструментов можно воспользоваться командой **Format ► Text**.

Чтобы выйти из текстовой области, надо щелкнуть мышью за ее пределами.

Управление Mathcad 12 осуществляется с помощью меню и *панели инструментов*. Для вставки операторов и функций удобно воспользоваться *математической панелью* (рис. П1.5), которая включается (выключается) командой **View ► Toolbars ► Math**.



Рис. П1.5 ▼ Математическая панель инструментов Mathcad 12

Первый значок математической панели () открывает *панель простейших вычислений Calculator* (рис. П1.6), которая предназначена для ввода функций, символов, цифр или операторов.

При щелчке по имени функции или оператору в рабочем листе появляется шаблон, в котором надо заполнить поля ввода (рис. П1.7).

Понятно, что панель **Calculator** содержит далеко не все функции Mathcad. Для того чтобы получить доступ к *встроенным* функциям пакета, можно обратиться к мастеру функций, воспользоваться командой **Insert ► Function**.

¹ Область ввода формул еще называют математической областью. Вообще в документе Mathcad существуют области трех видов: ввода формул, текстовая и построения графиков.



Рис. П1.6 ▼ Панель инструментов Calculator

- ! Факториал
- $\sqrt[n]{}$ Корень n-й степени
- | Модуль
- $^{}$ Возведение в степень
- ln(■) Натуральный логарифм
- :=■ Оператор присваивания

Рис. П1.7 ▼ Некоторые шаблоны панели Calculator

Кроме того, в Mathcad существует возможность *определять новые функции* как одного, так и нескольких аргументов. Для этого необходимо указать имя функции, в скобках перечислить ее аргументы, набрать символ : и определить новую функцию. Например:

```
g(x):=sin(x)+cos(x)
f(x,y,z):=sin(x)+cos(z*y^2).
```

После определения новую функцию можно использовать в документе наравне со встроенными (рис. П1.8).

$gg(x,y) := \sin(x \cdot \cos(y))$

$$gg\left(\frac{\pi}{7}, \frac{\pi}{11}\right) = 0.417$$

Рис. П1.8 ▼ Примеры использования функций

Для ввода греческих букв в Mathcad 12 предназначена панель **Greek Symbol** (рис. П1.9), которую можно вызвать, щелкнув по значку $\alpha\beta$ на математической



Рис. П1.9 ▼ Панель инструментов Greek Symbol

палитре или набрав соответствующую латинскую букву совместно с нажатой комбинацией клавиш **Ctrl+G**.

Все *вычислительные операторы* размещены на панели инструментов **Calculus** (рис. П1.10), которая вызывается из панели инструментов **Math** посредством кнопки .



Рис. П1.10 ▼ Панель инструментов **Calculus**

Рассмотрим последовательно все вычислительные операторы:

➤ *оператор дифференцирования*  предназначен для вычисления производной. Результатом его использования может быть как число (значение функции в точке), так и функция (в символьном виде). Чтобы получить результат в символьном виде, надо нажать комбинацию клавиш **Ctrl+.**, в документе Mathcad операция получения символьного результата отобразится в виде стрелки $\rightarrow$ (рис. П1.11);

$$g1(x) := \sin(x^3) \quad \frac{d}{dx} g1(x) \rightarrow 3 \cdot \cos(x^3) \cdot x^2$$

$$f1(x) := \sin(\cos(\sqrt{x})) \quad f2(x) := \frac{d}{dx} f1(x)$$

$$f2(x) \rightarrow \frac{-1}{2} \cdot \cos\left(\cos\left(x^{\frac{1}{2}}\right)\right) \cdot \frac{\sin\left(\frac{1}{x^2}\right)}{\frac{1}{x^2}}$$

$$f2(0.5) = -0.333$$

$$\frac{d}{dx} e^{\sin(x^2)} \rightarrow 2 \cdot \cos(x^2) \cdot x \cdot e^{\sin(x^2)}$$

Рис. П1.11 ▼ Примеры использования оператора дифференцирования

- оператор нахождения производной n -го порядка  позволяет вычислить производную любого порядка (рис. П1.12);

$$\frac{d^3}{dx^3} \sin(x^3) \rightarrow -27 \cdot \cos(x^3) \cdot x^6 - 54 \cdot \sin(x^3) \cdot x^3 + 6 \cdot \cos(x^3)$$

$$\text{ff}(x) := \cos(\sin(x))$$

$$\text{ff2}(x) := \frac{d^2}{dx^2} \text{ff}(x)$$

$$\text{ff2}(x) \rightarrow -\cos(\sin(x)) \cdot \cos(x)^2 + \sin(\sin(x)) \cdot \sin(x)$$

$$\text{ff2}\left(\frac{\pi}{5}\right) = -0.219$$

Рис. П1.12 ▼ Примеры использования оператора нахождения производной n -го порядка

- оператор определенного интеграла  служит для точного и приближенного вычисления определенного интеграла. Для приближенного (численного) вычисления после появления оператора на экране и ввода подынтегрального выражения следует нажать **=**, для точного – **Ctrl+**. (рис. П1.13);

$$\int_0^{\frac{\pi}{6}} \sin(x) \, dx = 0.134$$

$$\int_0^{\frac{\pi}{6}} \sin(x) \, dx \rightarrow \frac{-1}{2} \cdot 3^{\frac{1}{2}} + 1$$

Рис. П1.13 ▼ Примеры использования оператора определенного интеграла

- оператор неопределенного интеграла  вычисляет неопределенный интеграл. Для этого после появления оператора и ввода подынтегрального выражения следует нажать **Ctrl+**. (рис. П1.14);

$$\int x \sin(x^2) \, dx \rightarrow \frac{-1}{2} \cdot \cos(x^2)$$

$$\int \sin(x)^3 \, dx \rightarrow \frac{-1}{3} \cdot \sin(x)^2 \cdot \cos(x) - \frac{2}{3} \cdot \cos(x)$$

Рис. П1.14 ▼ Примеры использования оператора неопределенного интеграла

- *оператор суммирования*  (рис. П1.15) выводится на экран комбинацией клавиш **Ctrl+Shift+4**;

$$\sum_{k=1}^{\infty} \frac{1}{k^2} \rightarrow \frac{1}{6} \cdot \pi^2 \quad \sum_{k=0}^{10} \sin(k) = 1.411 \quad \sum_{k=1}^5 k! = 153$$

Рис. П1.15. Примеры использования оператора суммирования

- *оператор суммирования по ранжированной переменной*  (рис. П1.16) вызывается нажатием на клавиши **Shift+4**;

$$k := 1, 2 \dots 500$$

$$\sum_k \frac{1}{k^3} = 1.202 \quad \sum_k \frac{1}{k^2} = 1.643$$

$$p := 1, 2 \dots 300$$

$$\sum_p \frac{1}{(13\sqrt{p})^{19}} = 2.621$$

Рис. П1.16 ▼ Примеры использования оператора суммирования по ранжированной переменной

- *оператор произведения*  (рис. П1.17) вызывается посредством комбинации клавиш **Ctrl+Shift+3**;

$$\prod_{k=1}^{\infty} \left(1 + \frac{1}{k^2}\right) \rightarrow \frac{1}{\pi} \cdot \sinh(\pi) \quad \prod_{m=1}^{20} \left(1 + \frac{1}{m^3}\right) = 2.425$$

$$\prod_{m=1}^{20} \left(1 + \frac{1}{m^3}\right) \rightarrow \frac{14936861968691671606597909}{61587578582728704000000000}$$

Рис. П1.17 ▼ Примеры использования оператора произведения

- оператор произведения по ранжированной переменной $\prod_p$ (рис. П1.18) выводится комбинацией клавиш **Shift+3**;

$$p := 1, 2..10$$

$$\prod_p \left(2 - \frac{1}{p^2} \right) = 384.673 \quad \prod_p \sqrt{p} = 1.905 \times 10^3$$

$$u := 1, 2..150$$

$$\prod_u \sqrt[2u]{u} = 517.263$$

Рис. П1.18 ▼ Примеры использования оператора произведения по ранжированной переменной

- оператор предела $\lim_{\rightarrow a}$ (рис. П1.19) появляется при нажатии клавиш **Ctrl+L**;

$$\lim_{n \rightarrow \infty} \frac{1}{n^2} \rightarrow 0 \quad \lim_{n \rightarrow \infty} \left(\frac{n^2 + 5n - 7}{5n^2 + 16n - 100} \right) \rightarrow \frac{1}{5}$$

$$\lim_{x \rightarrow 0} \frac{\sin(x)}{x} \rightarrow 1 \quad \lim_{x \rightarrow 0} \operatorname{atan}\left(\frac{1}{x}\right) \rightarrow \text{undefined}$$

$$\lim_{x \rightarrow 0} \operatorname{atan}(\tan(x)) \rightarrow 0 \quad \lim_{x \rightarrow \frac{\pi}{2}} \operatorname{atan}(\tan(x)) \rightarrow \text{undefined}$$

Рис. П1.19 ▼ Примеры использования оператора предела

- оператор предела справа $\lim_{\rightarrow a^+}$ (рис. П1.20) вызывается комбинацией клавиш **Ctrl+Shift+A**;

$$\lim_{x \rightarrow 0^+} \frac{\sin(x)}{x} \rightarrow 1 \quad \lim_{x \rightarrow 0^+} \operatorname{atan}\left(\frac{1}{x}\right) \rightarrow \frac{1}{2} \cdot \pi$$

$$\lim_{x \rightarrow 0^+} \operatorname{atan}(\tan(x)) \rightarrow 0 \quad \lim_{x \rightarrow \frac{\pi}{2}^+} \operatorname{atan}(\tan(x)) \rightarrow -\frac{1}{2} \cdot \pi$$

Рис. П1.20 ▼ Примеры использования оператора предела справа

➤ оператор предела слева  (рис. П1.21) выводится на экран с помощью клавиш **Ctrl+Shift+B**.

$$\lim_{x \rightarrow 0^-} \frac{\sin(x)}{x} \rightarrow 1$$

$$\lim_{x \rightarrow 0^-} \operatorname{atan}\left(\frac{1}{x}\right) \rightarrow -\frac{1}{2} \cdot \pi$$

$$\lim_{x \rightarrow 0^-} \operatorname{atan}(\tan(x)) \rightarrow 0$$

$$\lim_{x \rightarrow 0^-} \operatorname{atan}(\tan(x)) \rightarrow 0$$

Рис. П1.21 ▼ Примеры использования оператора предела слева

На панели инструментов **Boolean** (рис. П1.22), которая управляется кнопкой  математической палитры, расположены *логические операторы и операторы отношения*.



Рис. П1.22 ▼ Панель инструментов **Boolean**

Панель **Boolean** содержит следующие операторы отношения: $>$, $<$, $=$, $\neq$, $\leq$, $\geq$, которые возвращают одно из двух логических значений: 1 (истина) или 0 (ложь). Над логическими операндами в Mathcad определены операторы:

- И (And) $\wedge$;
- ИЛИ (Or) $\vee$;
- исключающее ИЛИ (Exclusive or) $\oplus$;
- отрицание (Not) $\neg$.

Панель **Matrix** предназначена для *операций с матрицами и векторами* (рис. П1.23) и открывается щелчком по кнопке  в панели математических инструментов.



Панель операций с матрицами и векторами содержит следующие кнопки:



– определение размеров матрицы;



– ввод элемента массива;



– вычисление матрицы, обратной к данной: если A – исходная матрица, то A^{-1} – обратная к ней;



– вычисление определителя матрицы: $|A| = \det A$;



– оператор векторизации (поэлементные операции с векторами и матрицами): если $V = \{v_{ij}\}$, то $f(V) = \{f(v_{ij})\}$ или если $A = \{a_{ij}\}$ и $B = \{b_{ij}\}$, то $AB = \{a_{ij}b_{ij}\}$;



– определение столбца матрицы: $A^{\text{col } j}$ – j -й столбец матрицы;



– транспонирование матрицы: $A = \{a_{ij}\}$, $A^T = \{a_{ji}\}$;



– определение ранжированной переменной: $j = m, \dots, n$;



– вычисление скалярного произведения векторов: если $x(x_1, x_2, \dots, x_n)$ и $y(y_1, y_2, \dots, y_n)$, то $x \cdot y = x_1y_1 + x_2y_2 + \dots + x_ny_n$;



– вычисление векторного произведения векторов: если $x(x_1, x_2, x_3)$ и $y(y_1, y_2, y_3)$, то $x \times y = (x_2y_3 - x_3y_2, x_3y_1 - x_1y_3, x_1y_2 - x_2y_1)$;



– вычисление суммы компонент вектора: если $x(x_1, x_2, \dots, x_n)$, то $\sum x = x_1 + x_2 + \dots + x_n$;



– визуализация цифровой информации, сохраненной в матрице: обработка рисунков, которые представлены матрицами, задающими координаты их точек; изображения могут быть получены с помощью сканеров, цифровых фотоаппаратов, графических редакторов и т.п.

Итак, для ввода матрицы в рабочий документ необходимо выполнить следующие действия:

1. Ввести имя матрицы, знак присваивания ($:$) и щелкнуть по кнопке  на панели **Matrix**.
2. Заполнить диалоговое окно (рис. П.24), указав число строк (**Rows**) и число столбцов (**Columns**) будущей матрицы.
3. Закрыть диалоговое окно, щелкнув по кнопке **ОК**.
4. Ввести элементы матрицы в шаблон, который появится справа от знака присваивания (рис. П1.25).

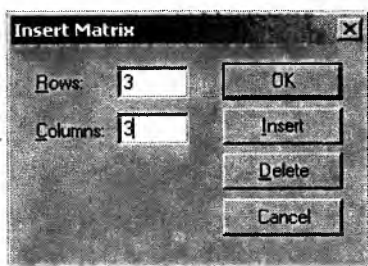


Рис. П1.24 ▼ Диалоговое окно определения размеров матрицы

$$A := \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Рис. П1.25 ▼ Фрагмент рабочего листа в момент ввода элементов матрицы

Для того чтобы выполнить какую-либо операцию над матрицей с помощью кнопок панели инструментов, необходимо щелкнуть по соответствующей кнопке и ввести в поле ввода имя матрицы. Например, щелчок по кнопке транспонирования матрицы M^T вызовет конструкцию M^T , а конструкция M^{-1}

$$\begin{aligned} & \text{ORIGIN} := 1 \\ & A := \begin{pmatrix} 1 & 2 & 3 & 2 \\ 0 & 3 & 2 & 1 \\ 2 & 1 & 0 & 3 \\ 4 & 3 & 0 & 1 \end{pmatrix} \quad \begin{array}{l} \text{Определение матрицы } A \\ \text{размерностью } 4 \times 4 \end{array} \\ & \quad A_{2,2} = 3 \quad \begin{array}{l} \text{Значение элемента матрицы } A, \\ \text{(2-я строка, 2-й столбец)} \end{array} \\ & A^{-1} = \begin{pmatrix} 0.25 & -0.375 & -0.125 & 0.25 \\ -0.313 & 0.469 & 0.031 & 0.063 \\ 0.5 & -0.25 & -0.25 & 0 \\ -0.063 & 0.094 & 0.406 & -0.188 \end{pmatrix} \quad \begin{array}{l} \text{Обратная матрица} \end{array} \\ & A^T = \begin{pmatrix} 1 & 0 & 2 & 4 \\ 2 & 3 & 1 & 3 \\ 3 & 2 & 0 & 0 \\ 2 & 1 & 3 & 1 \end{pmatrix} \quad \begin{array}{l} |A| = 64 \quad \text{Определитель матрицы } A \\ \text{Транспонирование матрицы } A \end{array} \\ & L := \frac{1}{2} \cdot A \quad L = \begin{pmatrix} 0.5 & 1 & 1.5 & 1 \\ 0 & 1.5 & 1 & 0.5 \\ 1 & 0.5 & 0 & 1.5 \\ 2 & 1.5 & 0 & 0.5 \end{pmatrix} \quad \begin{array}{l} \text{Умножение матрицы} \\ \text{на число} \end{array} \\ & \begin{array}{l} \text{Умножение матриц} \end{array} \quad \begin{array}{l} \text{Сложение матриц} \end{array} \\ & A \cdot L = \begin{pmatrix} 7.5 & 8.5 & 3.5 & 7.5 \\ 4 & 7 & 3 & 5 \\ 7 & 8 & 4 & 4 \\ 4 & 10 & 9 & 6 \end{pmatrix} \quad A + L = \begin{pmatrix} 1.5 & 3 & 4.5 & 3 \\ 0 & 4.5 & 3 & 1.5 \\ 3 & 1.5 & 0 & 4.5 \\ 6 & 4.5 & 0 & 1.5 \end{pmatrix} \end{aligned}$$

Рис. П1.26 ▼ Примеры операций над матрицами

появится, если щелкнуть по кнопке вычисления обратной матрицы $\times^{-1}$. Результат вычислений появится после введения знака равенства. Примеры некоторых операций над матрицами, выполненных с использованием панели **Matrix**, можно увидеть на рис. П1.26.

Работа с векторами в Mathcad также не вызывает затруднений. На рис. П1.27 приведены примеры векторных операций.

$$\begin{array}{ll}
 \text{ORIGIN} := 1 & \\
 \underline{H} := \begin{pmatrix} 1 & 2 & 0 \\ 2 & 0 & 3 \\ 3 & 1 & 2 \end{pmatrix} & \text{Определение матрицы H} \\
 & \text{размерностью 3x3} \\
 \underline{M} := \underline{H}^{(2)} & \underline{M} = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \quad \text{Выделение второго} \\
 & \text{столбца матрицы A} \\
 \underline{K} := 2 \cdot \underline{M} & \underline{K} = \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix} \quad \text{Умножение вектора на константу} \\
 \underline{D} := \underline{M} + \underline{K} - \frac{\underline{H}^{(3)}}{5} & \underline{D} = \begin{pmatrix} 6 \\ -0.6 \\ 2.6 \end{pmatrix} \quad \text{Сложение векторов} \\
 \underline{U} := \underline{D}^T & \underline{U} = (6 \quad -0.6 \quad 2.6) \quad \text{Транспонирование вектора} \\
 d := |\underline{D}| & d = 6.567 \quad \text{Длина вектора D} \\
 \underline{S} := \sum \underline{D} & S = 8 \quad \text{Сумма элементов вектора} \\
 \underline{M} \cdot \underline{D} = 14.6 & \text{Скалярное произведение} \\
 \underline{M} \times \underline{D} = \begin{pmatrix} 0.6 \\ 0.8 \\ -1.2 \end{pmatrix} & \text{Векторное произведение}
 \end{array}$$

Рис. П1.27 ▼ Примеры векторных операций

Отметим, что нижняя граница индексации в Mathcad определена системной переменной **ORIGIN**. По умолчанию значение этой переменной равно 0. Обратиться к элементу матрицы или вектора (или присвоить ему значение) можно, если после имени переменной использовать символ прямой открывающейся скобки [, то есть ввод символов $X[$ дает конструкцию $X_{\square}$.

Чтобы выполнить построение графика того или иного вида, необходимо указать его тип. Перечень основных типов графиков, предлагаемый Mathcad,

можно увидеть, если обратиться к пункту главного меню **Insert** (Вставить) и выбрать подпункт **Graph** (График). Кроме того, создать график можно и при помощи панели инструментов **Graph** (рис. П1.28). Для этого надо выбрать кнопку  на математической палитре.



Рис. П1.28 ▼ Панель инструментов для создания различных графиков

Команды, представленные в меню **Graph**, и соответствующие им кнопки на панели инструментов позволяют построить следующие типы графиков:

- **X-Y Plot** (декартов график ) – двумерный график в декартовой системе координат;
- **Polar Plot** (полярный график ) – график в полярных координатах;
- **Surface Plot** (график поверхности ) – трехмерный график (поверхность);
- **Contour Plot** (контурный график ) – контурный график трехмерной поверхности;
- **3D Scatter Plot** (точечный график ) – график в виде точек в трехмерном пространстве;
- **3D Bar Chart** (трехмерная гистограмма ) – график в виде совокупности столбиков в трехмерном пространстве;
- **Vector Field Plot** (векторное поле ) – график векторного поля на плоскости;
- **Plot Wizard...** (мастер трехмерной графики) – трехмерный график с заданными свойствами, созданный при помощи мастера.

Вообще, вызов любой из этих команд, кроме последней, приведет к вставке в рабочий документ специальной графической области, называемой *шаблоном*. Шаблон может содержать одно или несколько полей ввода. Заполнение этих полей соответствующими исходными данными завершит процесс построения графика.

Отметим, что шаблон для построения появляется в обозначенном месте рабочего документа, то есть там, где находится курсор. Но если надо, графическая область перетаскивается в другое место при помощи мыши. Меняются также и размеры области построения графика. Их можно увеличивать или уменьшать по вертикали, по горизонтали и по диагонали, удерживая курсор мыши на специальных маркерах. Графические области, как и любые другие объекты Mathcad, при необходимости выделяют, помещают в буфер обмена, копируют, перемещают или удаляют.

На рис П1.29 изображен график *функции, заданной двумя векторами*. Для его построения надо выполнить последовательность команд:

1. Сформировать векторы данных X и Y .
2. Вставить шаблон для построения графика в декартовой системе координат (кнопка  в панели инструментов **Graph** или выполнить команды **Insert > Graph > X-Y Plot**).
3. Заполнить поля ввода в графическом шаблоне.
4. Нажать клавишу **Enter** или щелкнуть мышью вне области построения.

В результате получится график, заданный соответствующими парами элементов определенных ранее векторов.

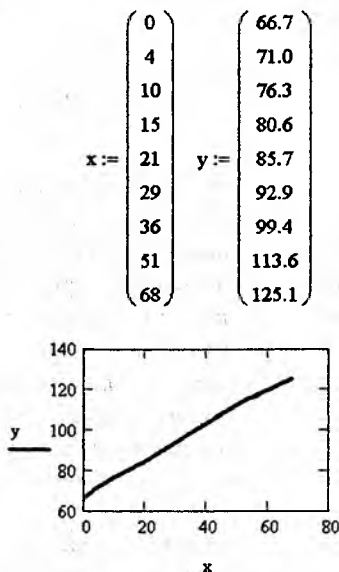


Рис. П1.29 ▼ Построение графика функции, заданной в виде таблицы

График функции, изображенный на рис П1.30, создан так называемым *быстрым способом построения*. В шаблон декартова графика () сразу вводится имя аргумента в поле по оси абсцисс и функция в соответствующее поле оси ординат. График создается автоматически, поскольку значения аргумента не были определены иначе. В Mathcad пределы значения аргумента по умолчанию изменяются в диапазоне от -10 до 10 .

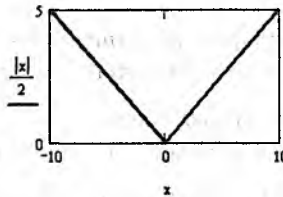


Рис. П1.30 ▾ Быстрое построение графика функции

Если значение функции и значение аргумента будут использованы в дальнейших вычислениях, то лучше построить *график с ранжированным аргументом*. Это делается следующим образом:

1. Аргумент x определите как ранжированную переменную.
2. Задайте функцию $y(x)$.
3. Вставьте шаблон для построения графика (.
4. Введите x по оси абсцисс, а $y(x)$ по оси ординат.
5. Нажмите **Enter**.

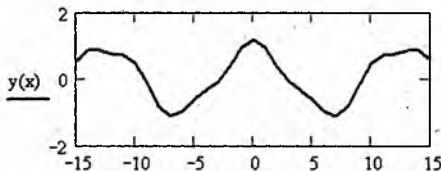
Несколько вариантов построения графика с ранжированным аргументом представлены на рис. П1.31. В первом случае аргумент, заданный ранжированной переменной, определен небольшим количеством значений x $-15, -14, \dots, 0, 1, 2, \dots, 14, 15$ (то есть с шагом 1). Повторное определение переменной x (с шагом 0.2) значительно увеличило количество аргументов функции, в результате чего форма графика стала более точной.

Заметим, что если в рабочем документе была определена некая ранжированная переменная, а впоследствии создан график функции методом быстрого построения, то зависимость будет построена с учетом значения этой ранжированной переменной, а не по параметрам, установленным в Mathcad по умолчанию.

Рис. П1.32 содержит пример построения графиков нескольких функций в одной графической области. Выполнить это несложно – достаточно отложить

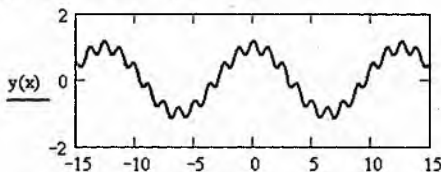
по оси абсцисс аргумент x , а по оси ординат через запятую перечислить все заданные функции¹.

$$x := -15..15 \quad y(x) := \cos\left(\frac{x}{2}\right) + \frac{1}{5} \cdot \cos(5x)$$



Аргумент
изменяется
с шагом 1

$$x := -15, -14.8..15 \quad x$$



Аргумент
изменяется
с шагом 0.2

Рис. П1.31 ▼ Построение графика с ранжированным аргументом

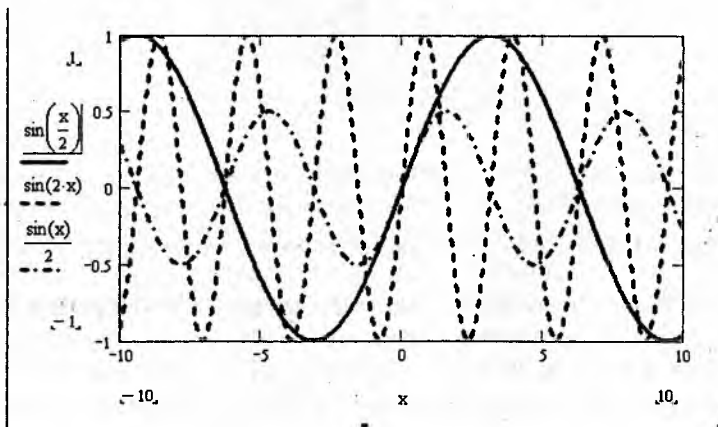


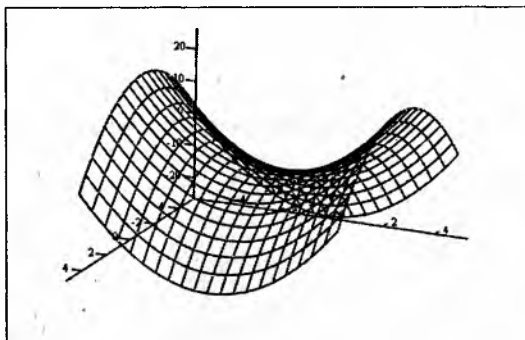
Рис. П1.32 ▼ Построение графика нескольких функций

¹ При перечислении функций по оси ординат запятая ставится перед вводом следующего выражения только тогда, когда курсор полностью захватывает предыдущее.

Форматирование двумерного графика можно выполнить с помощью команд **Format ► Graph ► X-Y Plot** при выделенной графической области. В результате откроется диалоговое окно, в котором можно установить параметры отображения осей (вкладка **X-Y Axes**), линий графика (вкладка **Traces**), надписей на графике (вкладка **Labels**) и параметры по умолчанию (вкладка **Defaults**). Добиться появления этого же окна можно, дважды щелкнув по области построения графика. Заметим, что все параметры форматирования относятся к выделенной графической области.

Для построения трехмерных графиков достаточно вызвать шаблон требуемого типа, например выполнить команды **Insert ► Graph ► Surface Plot** или воспользоваться кнопкой  на панели инструментов **Graph**, и ввести в него имя соответствующей матрицы или функции. График, заданный функцией, изображен на рис. П1.33.

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



z

Рис. П1.33 ▼ Быстрое построение графика функции $z(x, y) = y^2 - x^2$

Окно форматирования трехмерного графика **3D Plot Format** вызывается двойным щелчком по области построения графика или командой основного меню **Format ► Graph ► 3D Plot...** при выделенном графике.

Панель инструментов **Programming** (рис. П1.34) предназначена для программирования в Mathcad и вызывается кнопкой  математической палитры.

Панель **Programming** содержит следующие операторы:

- Add Line – создания блока для ввода команд Mathcad-программы;
- ← – присваивания;

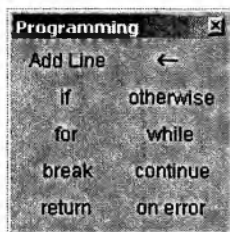


Рис. П1.34 ▼ Средства программирования

- If – условный;
- Otherwise – альтернативного выбора (применяется вместе с условным);
- For – цикла с известным числом повторений;
- While – условного цикла;
- Break – прерывания;
- Continue – продолжения;
- Return – возврата;
- on error – обработки ошибок.

Рассмотрим перечисленные средства программирования.

Оператор Add Line вставляет в рабочий документ конструкцию, показанную на рис. П1.35 (*программный блок или программный модуль*). Блок ограничен жирной вертикальной линией, справа от которой расположены поля ввода команд программного модуля.

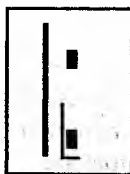


Рис. П1.35 ▼ Создание программного модуля

Чтобы расширить программный блок путем добавления новых полей ввода, надо повторно вызвать оператор Add Line, но уже внутри программного модуля. Причем можно не только сделать блок длиннее, но и создавать всевозможные вложенные структуры, например такие, как на рис. П1.36.

Обратите внимание, что операторы программного модуля одного уровня объединяются вертикальной линией. Если необходимо добавить поле ввода в конструкцию if, то курсор должен быть установлен так, как показано на рис. П1.36. Если же требуется вставить новое поле ввода вне условного оператора, то следует

• if • Конструкция условного оператора

$$F(a, b) := \begin{cases} c \leftarrow 0 \\ c \leftarrow a + b \text{ if } a < b \\ c \leftarrow c + 12 \end{cases}$$

$F(3, 6) = 21$ $3 < 6$, следовательно $c := 3 + 6 + 12$ $c = 21$

$F(8, 1) = 12$ $3 > 6$, следовательно $c = 12$

Рис. П1.37 ▼ Применение условного оператора

Оператор альтернативного выбора позволяет запрограммировать условную конструкцию вида:

Оператор1 if условие
Оператор2 otherwise.

Другими словами, если условие истинно, то выполнить оператор1, иначе – оператор2. Пример применения подобной конструкции вы видите на рис. П1.38. Функция выводит текстовое сообщение о том, какое из двух заданных чисел больше.

• if • Конструкция оператора
• otherwise

$$F(a, b) := \begin{cases} "a > b" \text{ if } a > b \\ "a < b" \text{ otherwise} \end{cases}$$

$F(3, 7) = "a < b"$
 $F(8, 1) = "a > b"$

Рис. П1.38 ▼ Применение условного оператора с альтернативным выбором

Оператор for предоставляет возможность организовать цикл по некоторой переменной, изменяющейся в заданном диапазоне.

Оператор цикла с известным числом повторений записывают так:

for переменная 0 начальное значение, шаг .. конечное значение
оператор.

Эта запись означает, что оператор выполнится для значений переменной, изменяющейся в диапазоне от начального значения до конечного, с заданным шагом.

На рис. П1.39 показаны примеры создания циклов с известным числом повторений.

for $i \in 1..10$ Конструкция оператора

Сумма цифр от 1 до 10

```
SUM := | S ← 0
      | for i ∈ 1..10
      |   S ← S + i
SUM = 55
```

Сумма нечетных цифр от 1 до 10

```
SUM_N := | S ← 0
          | for i ∈ 1,3..10
          |   S ← S + i
SUM_N = 25
```

Рис. П1.39 ▼ Использование оператора цикла с известным числом повторений

Рис. П1.40 содержит программный модуль с применением цикла, в теле которого выполняются два оператора. Результатом работы функции является вектор, первый элемент которого – сумма чисел от 1 до n , второй – произведение этих чисел.

$P(n) := \begin{array}{ l} S \leftarrow 0 \\ P \leftarrow 1 \\ \text{for } i \in 1..n \\ \quad \begin{array}{ l} S \leftarrow S + i \\ P \leftarrow P \cdot i \end{array} \\ V_0 \leftarrow S \\ V_1 \leftarrow P \\ V \end{array}$	Присвоение начальных значений Вычисляется значение суммы чисел и произведения от 1 до n. Формируется вектор из значений суммы и произведения
$P(5) = \begin{pmatrix} 15 \\ 120 \end{pmatrix}$	Результат работы функции

Рис. П1.40 ▼ Создание функции с применением циклического оператора

С помощью оператора **while** создается цикл, количество повторений которого неизвестно, но предусмотрен выход из него по некоторому логическому условию. Запись *условного цикла* или *цикла с неизвестным числом повторений* имеет вид:

while условие
оператор.

Это означает, что оператор будет выполняться в цикле до тех пор, пока истинно условие.

Рассмотрим использование оператора **While** в задаче определения наибольшего общего кратного двух чисел. По алгоритму Евклида числа сравниваются

друг с другом и большее уменьшается на величину меньшего до тех пор, пока они не станут равными. Программный модуль на рис. П1.41 реализует эту задачу.

while Конструкция оператора

$$P(a, b) := \left| \begin{array}{l} \text{while } a \neq b \\ \quad \left| \begin{array}{l} a \leftarrow a - b \text{ if } a > b \\ b \leftarrow b - a \text{ otherwise} \end{array} \right. \\ a \end{array} \right.$$

$$P(5, 15) = 5 \quad P(21, 18) = 3 \quad P(1, 13) = 1$$

Рис. П1.41 ▾ Функция для определения наибольшего общего делителя

Фрагмент рабочего документа на рис. П1.42 демонстрирует, как с помощью этой функции можно определить наибольший общий делитель множества чисел, заданных в виде массива x .

$$R(V) := \left| \begin{array}{l} a \leftarrow V_0 \\ \text{for } i \in 1.. \text{rows}(V) - 1 \\ \quad a \leftarrow P(a, V_i) \\ a \end{array} \right. \quad x := \begin{pmatrix} 21 \\ 14 \\ 49 \\ 35 \end{pmatrix} \quad R(x) = 7$$

Рис. П1.42 ▾ Определение наибольшего общего делителя множества чисел

Иногда бывает необходимо завершить цикл досрочно. Для этого предназначен *оператор прерывания* break. Применение этого оператора показано на рис. П1.43. Задача заключается в определении положения первого нулевого элемента в массиве чисел. Понятно, что если такой элемент найден, то нет смысла проверять массив до конца. Посредством оператора прерывания управление в этом случае передается на конец цикла.

$$\begin{array}{l} \text{ORIGIN} := 1 \\ G(X) := \left| \begin{array}{l} k \leftarrow 0 \\ \text{for } i \in 1.. \text{rows}(X) \\ \quad \text{if } X_i = 0 \\ \quad \quad \left| \begin{array}{l} k \leftarrow i \\ \text{break} \end{array} \right. \\ k \end{array} \right. \end{array} \quad Y := \begin{pmatrix} 1 \\ 2 \\ 0 \\ 3 \\ 5 \end{pmatrix} \quad Z := \begin{pmatrix} 1 \\ 8 \\ 9 \end{pmatrix}$$

$$G(Y) = 3 \quad G(Z) = 0$$

Рис. П1.43 ▾ Использование оператора прерывания цикла

Оператор `continue` – это также оператор прерывания, но в отличие от `break`, прерывающего цикл, он останавливает выполнение только текущей итерации. Пример вычисления ненулевых элементов массива оператором `continue` приведен на рис. П1.44.

$$F(x) := \begin{cases} P \leftarrow 1 \\ \text{for } i \in 0.. \text{rows}(x) - 1 \\ \quad \left| \begin{array}{l} \text{continue if } x_i = 0 \\ P \leftarrow P \cdot x_i \end{array} \right. \\ P \end{cases} \quad X := \begin{pmatrix} 1 \\ 5 \\ 0 \\ 3 \\ 0 \\ 1 \end{pmatrix} \quad F(X) = 15$$

Рис. П1.44 ▼ Применение оператора продолжения цикла

Оператор `return` прерывает выполнение программного модуля в любой точке и возвращает значение выражения или переменной либо текстовое сообщение, стоящее следом за ним. Пример работы этого оператора вы видите на рис. П1.45. Функция определяет, является ли вектор единичным. При наличии хотя бы одного элемента вектора, отличного от единицы, программный модуль прерывается.

$$E(V) := \begin{cases} \text{for } i \in 0.. \text{rows}(V) - 1 \\ \quad \text{return "NO" if } V_i \neq 1 \\ \text{"YES"} \end{cases} \quad a := \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad b := \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

$E(a) = \text{"YES"} \quad E(b) = \text{"NO"}$

Рис. П1.45. Применение оператора прерывания программного модуля

Если при составлении программного модуля предполагается, что какая-либо команда может вызвать ошибку (например, деление на ноль), то можно воспользоваться оператором `on error` для перехвата этой ошибки.

Оператор перехвата ошибок имеет следующую конструкцию:

выражение1 `on error` выражение2,

где выражение2 – действие, которое должно выполниться в данной строке программного модуля, но если при этом произойдет ошибка, то вместо него будет выполнено выражение1.

Из рис. П1.46 ясно, как функционирует этот оператор.

$$f(x) := \begin{cases} y \leftarrow x \\ \text{"divide by zero"} \text{ on error } \frac{1}{y^2 - 1} \end{cases}$$

$$f(2) = 0.333 \quad f(1) = \text{"divide by zero"} \quad f(-1) = \text{"divide by zero"}$$

Рис. П1.46 ▼ Использование оператора обработки ошибок

Документ, созданный в среде пакета, хранится в отдельном файле с расширением `mcd`. Рассмотрим основные средства, которыми располагает Mathcad для работы с файлами.

Каждый раз при запуске Mathcad открывает новое пустое окно под именем **Untitled:1** (Без названия).

При необходимости создать новое окно, находясь непосредственно в среде пакета, надо выполнить команду основного меню **File ► New** или щелкнуть по кнопке **New** (Создать) на панели инструментов. Любая из этих операций приведет к созданию пустого документа в окне Mathcad. Каждый новый документ выводится с условным названием **Untitled:1**, **Untitled:2**, **Untitled:3** и т.д. – в зависимости от того, сколько документов было открыто до него.

Для сохранения документа в Mathcad нужно выполнить команду основного меню **File ► Save** или выбрать кнопку **Save** (Сохранить) на панели инструментов. Если документ сохраняется впервые, то будет выведено диалоговое окно **Save As**, в котором определяется имя и тип сохраняемого файла. Окно сохранения практически ничем не отличается от таких же окон в других приложениях.

Открывается существующий документ командой **File ► Open** или кнопкой **Open** (Открыть) на панели инструментов.

Закрыть документ можно командой **File ► Close** или свернуть окно документа как любое стандартное окно Windows. Завершить сеанс работы в Mathcad можно, закрыв окно пакета либо выполнив команду **File ► Exit**. Если файл не был сохранен, то появится окно **Save As**, в котором можно сохранить файл и выйти, отказаться от изменений и выйти (то есть выйти без сохранения) или вернуться к редактированию.

Редактирование документа в Mathcad мало чем отличается от подобных операций в стандартных приложениях Windows.

Чтобы выделить фрагмент документа, удерживая левую кнопку мыши, обведите области, расположенные непосредственно друг за другом. Все содержимое документа выделяется командой **Edit ► Select All**. Снять выделение можно щелчком мыши в невыделенной части документа.

Для удаления фрагмента документа его необходимо выделить, а затем нажать клавишу **Del**. Пустые строки, образовавшиеся при удалении, будут убраны, если установить курсор в начало строки и нажать клавишу **Del**.

Фрагмент документа, который требуется скопировать (переместить), надо выделить и поместить в буфер обмена командой главного меню **Edit ► Copy** (**Edit ► Cut**). Вставляется фрагмент из буфера с помощью команды главного меню **Edit ► Paste**.

Кроме того, копировать и перемещать фрагменты документа удобно путем перетаскивания. Если поместить указатель мыши на один из выделенных участков, он приобретет форму ладони. Удерживая левую кнопку мыши, фрагмент можно перетащить в любое место рабочего документа, а при нажатой клавише **Ctrl** – скопировать.

Математические и текстовые области выравнивают либо по правому, либо по верхнему краю. Для этого их необходимо предварительно выделить и выполнить команду **Format ► Align Regions**.

Если в процессе работы складывается впечатление, что в документе появилась лишняя информация и/или удаляемая информация не исчезает, то все станет на место, если выполнить команду обновления документа **View ► Refresh**.

Операция отмены последнего действия работает в Mathcad так же, как и в других приложениях Windows. Ранее произведенное действие отменяется командой **Edit ► Undo**, повторить ранее отмененную операцию позволяет команда **Edit ► Redo** основного меню.

Справочная система Mathcad включает в себя несколько электронных книг:

- **Tutorials** – электронные учебники по Mathcad, построенные в форме обучающих курсов от простого к сложному;
- **QuickSheets** – документы Mathcad, организованные в виде электронной книги, которую удобно использовать в качестве шаблонов, часто называемые «быстрыми шпаргалками».
- **Reference Tables** – справочники, содержащие фундаментальные константы, единицы измерения и т.п.;
- **User's Guide** – руководство пользователя Mathcad;
- **Release Notes** – комментарии к текущей версии Mathcad;
- **E-book** – электронные книги, представленные в виде дополнительных библиотек.

Справочная система вызывается из основного меню командой **Help** (Помощь).

Каждая из электронных книг открывается в специфическом окне и имеет свои элементы управления, которые похожи на строение окна браузера. Перемещаться по содержимому электронных книг несложно. Как и Web-страницы, они снабжены гиперссылками, которые позволяют переходить со страницы на страницу. Кроме того, существуют специальные кнопки для поиска, перехода на предыдущую или домашнюю страницу и т.д.

Приложение 2

Введение в MATLAB

В этом приложении приведены общие сведения о пакете MATLAB 7 и встроенном языке программирования (М-языке).

После запуска MATLAB 7 (Пуск ► Программы ► MATLAB 7.0 ► MATLAB 7.0) на экране появится среда приложения MATLAB (рис. П2.1).

Команды MATLAB можно вводить в командном окне (Command Window). В командном окне можно ввести обычное арифметическое выражение и нажать клавишу Enter. MATLAB вычислит значение выражения, выведет его на экран и поместит в системную переменную ans. Если после ввода выражения набрать

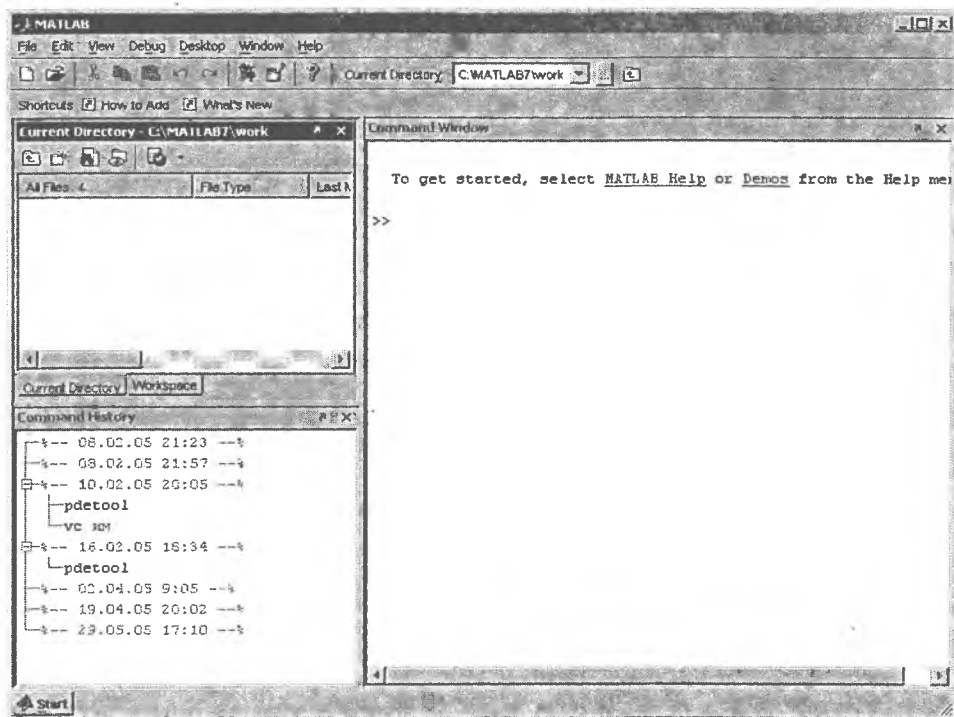
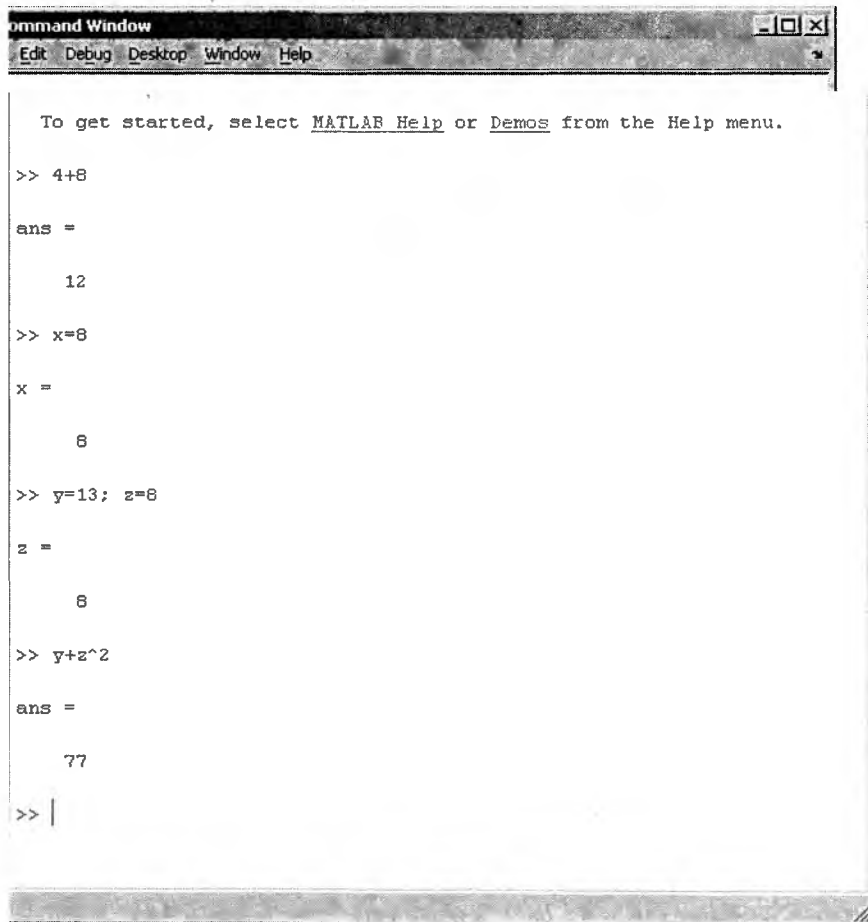


Рис. П2.1 ▼ Окно приложения MATLAB 7

«», то результат будет вычислен, помещен в переменную `ans`, но переменная экран выводиться не будет. В MATLAB определены следующие арифметические операции: `+` – сложение, `-` – вычитание, `*` – умножение, `/` – деление, `^` – возведение в степень). В одной строке командного окна можно вводить несколько команд, разделенных символом «;». В командном окне MATLAB есть возможность работать с переменными, для этого их необходимо определить. Оператор присваивания в MATLAB имеет вид: `<имя переменной> = <выражение>`.

ис. П2.2 приведены примеры простейших вычислений в MATLAB.



```
command Window
Edit  Debug  Desktop  Window  Help

To get started, select MATLAB Help or Demos from the Help menu.

>> 4+8

ans =

    12

>> x=8

x =

     8

>> y=13; z=8

z =

     8

>> y+z^2

ans =

    77

>> |
```

Рис. П2.2 ▾ Простейшие вычисления в MATLAB 7

При работе с переменными следует помнить, что в MATLAB различаются большие и маленькие буквы (переменные с именами AB , aB , Ab и ab – это разные переменные).

В MATLAB определены следующие часто встречающиеся алгебраические функции: $\sin(Z)$, $\sinh(Z)$, $\operatorname{asin}(Z)$, $\cos(Z)$, $\cosh(Z)$, $\operatorname{acos}(Z)$, $\tan(Z)$, $\tanh(Z)$, $\operatorname{atan}(Z)$, $\cot(Z)$, $\coth(Z)$, $\operatorname{acot}(Z)$, $\exp(Z)$, $\log(Z)$, $\log_{10}(Z)$, $\sqrt{Z}$, $\operatorname{abs}(Z)$, $\operatorname{sign}(Z)$, $\exp(Z)$, π^i . Комплексные переменные вводятся следующим оператором присваивания:

`<имя переменной> = <Действительная часть> + i|j* Мнимая часть`

MATLAB является специализированным пакетом для работы с векторами и матрицами, поэтому все объекты в нем матрицы, даже простая переменная – это матрица размером 1×1 . Матрицу (массив) можно определить с помощью оператора присваивания. Для этого необходимо ввести имя переменной, символ `=` и в квадратных скобках значения элементов матрицы через пробел, одну строку от другой отделяют символом «;». На рис. П2.3 определена матрица A размером 2×6 .

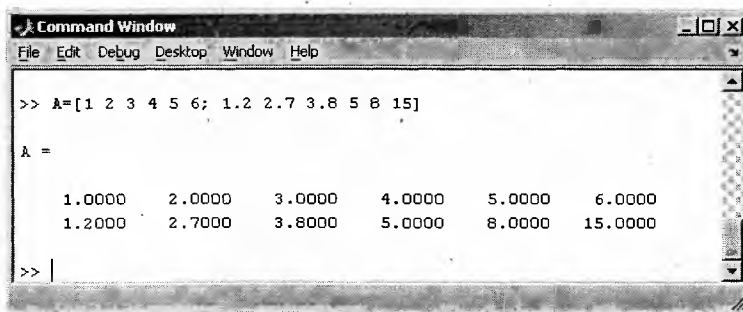


Рис. П2.3 ▼ Пример определения матрицы A в MATLAB 7

На рис. П2.4 определены матрица-строка и матрица-столбец.

Несколько векторов (матриц) можно объединить в строку следующим образом: $v = [v1; v2]$ (рис. П2.5).

В MATLAB можно определить *массив* как последовательность значений, изменяющуюся от начального (XN) до конечного значения (XX) с некоторым шагом (HX)². Для этого можно воспользоваться оператором присваивания следующего вида:

`X = XN[ :HX ]:XX,`

¹ Переменная Z может принимать как действительные, так и комплексные значения.

² Аналог ранжированной переменной в Mathcad.

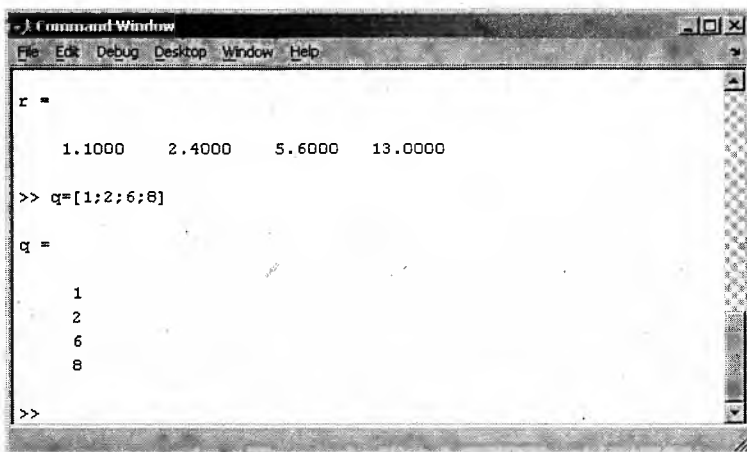


Рис. П2.4 ▼ Пример определения матрицы-строки и матрицы-столбца

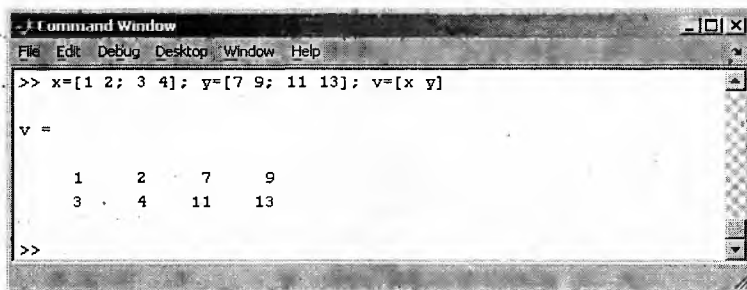


Рис. П2.5 ▼ Пример объединения матриц в MATLAB 7

где X – имя массива, XN – начальное значение, XK – конечное значение, HX – шаг изменения переменной. Если HX отсутствует, то шаг равен единице. Примеры определения массивов как последовательности значений представлены на рис. П2.6.

Аналогично оператору «:» в MATLAB существует функция `linspace`, которая формирует массив равноотстоящих точек, эта функция применяется в двух формах:

- ▶ `linspace (a, b)` – возвращает массив из 100 точек, равномерно распределенных между a и b;
- ▶ `linspace (a, b, n)` – возвращает массив из n точек, равномерно распределенных между a и b.

```

J Command Window
File Edit Debug Desktop Window Help

>> x=1:0.3:4

x =

Columns 1 through 6

    1.0000    1.3000    1.6000    1.9000    2.2000    2.5000

Columns 7 through 11

    2.8000    3.1000    3.4000    3.7000    4.0000

>> K=1:7

K =

     1     2     3     4     5     6     7

>>

```

Рис. П2.6 ▾ Пример массива как последовательности значений в MATLAB 7

Для обращения к элементу массива (матрицы) необходимо указать имя и в круглых скобках индексы элемента. Например, $A(2, 4)$ или $x(3)$. Элементы массивов нумеруются с нуля. Можно обратиться к строке ($A(i, :)$, которая возвращает i -ю строку матрицы A), или к столбцу матрицы ($A(:, j)$, который возвращает j -й столбец матрицы A).

Операции над массивами и матрицами. Над матрицами в MATLAB определены следующие операции:

- + – сложение матриц;
- – – вычитание матриц;
- * – умножение матриц;
- .* – поэлементное умножение матриц;
- ^ – возведение в степень;
- .^ – поэлементное возведение в степень;
- ' – транспонирование матриц;
- ./ – поэлементное деление матриц.

Построение двумерных графиков в MATLAB. Для построения двумерных графиков можно воспользоваться функцией `plot` следующей структуры:

```
plot (x1, y1, s1, x2, y2, s2, x3, y3, s3, ...),
```


где $x_1, x_2, x_3, \dots$ – аргументы функций, $y_1, y_2, y_3, \dots$ – имена функций или массивов значений ординат функций, строковые константы $s_1, s_2, s_3, \dots$ определяют тип линий графика (цвет и тип линии, а также символ, отображающий точки на графике). В табл. П2.1 приведены значения параметра si , определяющего параметры линии выводимого графика.

Таблица П2.1 ▼ Свойства линий графика

Цвет	
y	желтый
m	розовый
c	голубой
r	красный
g	зеленый
b	синий
w	белый
k	черный
Тип символа	
.	точка
o	окружность
x	крестик
+	знак «+»
*	звездочка
s	квадрат
d	ромб
v	треугольник вершиной вниз
^	треугольник вершиной вверх
<	треугольник вершиной влево
>	треугольник вершиной вправо
p	пятиконечная звезда
h	шестиконечная звезда
Тип линии	
-	сплошная линия
:	пунктирная
-.	штрихпунктирная
—	штриховая

Для подписи осей можно воспользоваться командами `xlabel(text)`, `ylabel(text)`, для подписи графика следует использовать команду `title(text)`, легенды на график выводятся с помощью команды `legend(text1, text2, ...)`¹.

¹ Здесь `text`, `text1`, `text2` – любой заключенный в кавычки текст (текстовая строка).

В качестве примера можно рассмотреть построение графиков функций $y = e^{\sin(x)} \cos(x)$, $z = e^{\sin^2(x)}$ на интервале $x \in [-\pi, \pi]$. Для построения графика в командном окне MATLAB можно ввести¹:

```
>> x=-pi:pi/20:pi;
>> y=exp(sin(x)).*cos(x);
>> z=exp(sin(x).^2);
>> plot(x,y,'r-',x,z,'k+');
>> xlabel("X");ylabel("Y");title("Графики функций");legend("Y","Z");
```

Полученный график представлен на рис. П2.7.

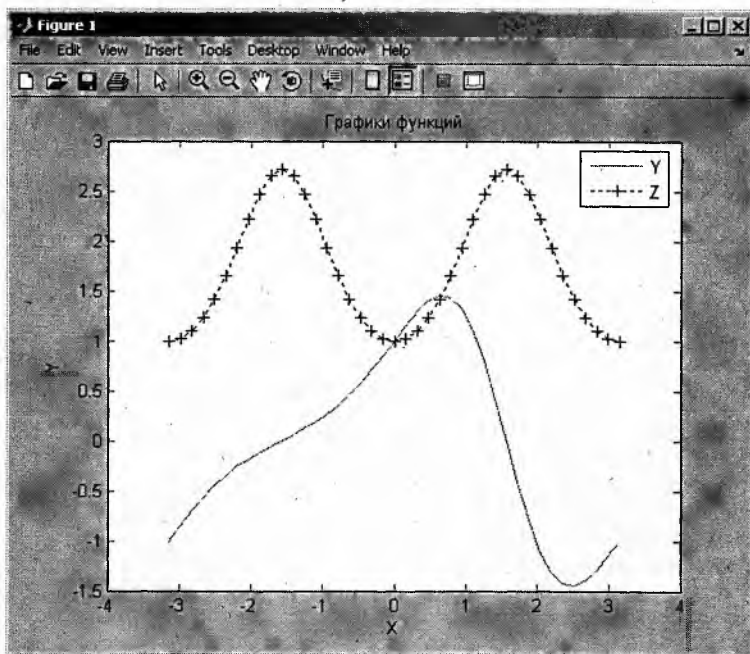


Рис. П2.7 ◀ Окно графиков функций $y = e^{\sin(x)} \cos(x)$, $z = e^{\sin^2(x)}$ в MATLAB 7

Построение трехмерных графиков в MATLAB. Перед построением графика функций двух переменных необходимо разбить область построения прямоугольной сеткой с помощью функции `meshgrid` и вычислить значения функции в узлах сетки. После этого можно построить график поверхности с помощью

¹ Обратите внимание на использование операций `.*` и `.^`.

функций `mesh(X, Y, Z)` или `surf(X, Y, Z)`. Для подписей осей координат можно воспользоваться функциями `xlabel(text)`, `ylabel(text)`, `zlabel(text)`.

Построение трехмерного графика рассмотрим на примере функции $z(x, y) = y^2 - x^2$ на интервале $x \in [-2, 2]$, $y \in [-2, 2]$. В листинге приведены команды MATLAB, которые можно использовать для построения каркасной поверхности (функция `mesh`).

```
>> [X,Y]=meshgrid(-2:0.2:2,-2:0.2:2);  
>> Z=Y.^2-X.^2;  
>> mesh(X,Y,Z);  
>> xlabel("X");ylabel("Y");zlabel("Z");title("Поверхность");
```

Полученный график представлен на рис. П2.8.

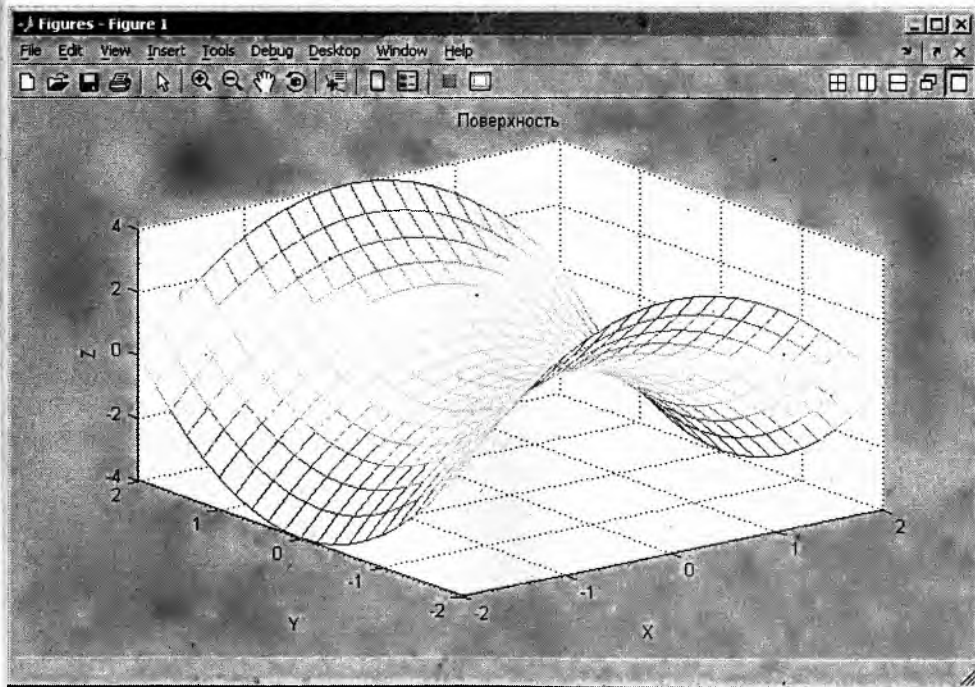


Рис. П2.8 ▾ График поверхности $z(x, y) = y^2 - x^2$ в MATLAB 7

Если команду `mesh(X, Y, Z)` заменить командой `surf(X, Y, Z)`, то будет построена каркасная поверхность с заливкой цветом каждой клетки поверхности (рис. П2.9).

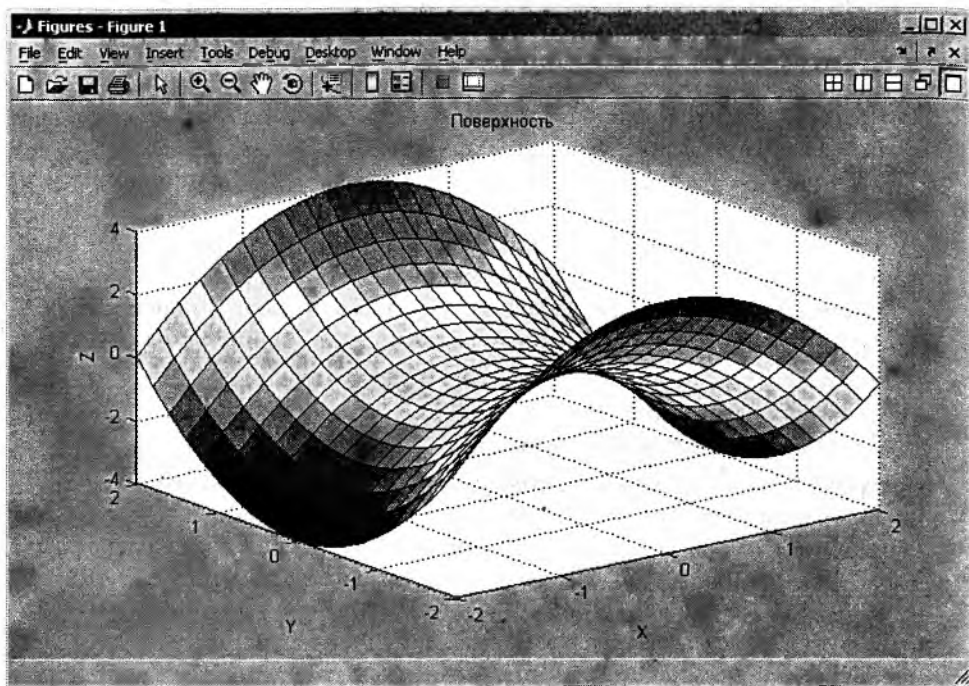


Рис. П2.9 ▾ График $z(x, y) = y^2 - x^2$ с заливкой цветом (surf) в MATLAB 7

Сохранение вычисленных значений. Одним из самых простых способов сохранения вычисленных значений является сохранение рабочей среды (всех переменных) с помощью команды **File ► Save Workspace As ...**. При этом открывается диалоговое окно, в котором необходимо указать папку и имя сохраняемого файла. Файл рабочей среды имеет расширение *mat*. После выполнения этой команды все переменные и значения рабочей среды будут храниться в одном файле с расширением *mat*. Для загрузки рабочей среды необходимо выполнить команды **File ► Open**.

Кроме сохранения переменных в файле рабочей среды в MATLAB есть возможности работы с текстовым файлом, аналогичные языку Си.

Для записи данных в текстовый файл необходимо выполнить следующие действия:

1. Открыть файл с помощью функции `fopen`. Функция `fopen(filename, mode)` имеет два параметра: `filename` – строка, в которой хранится полное имя открываемого файла; `mode` – строка, которая определяет режим работы

с файлом; возможны следующие значения: "rt" – открываем текстовый файл в режиме чтения; "wt" – создаем пустой текстовый файл для записи; "rt+" – открываем текстовый файл в режиме чтения и записи; "wt+" – открываем пустой текстовый файл, в этом в режиме можно записывать и читать данные из файла. Функция возвращает два параметра: файловая переменная и строка символов, к которой хранится сообщение о результатах открытия файла.

2. Записать данные в файл с помощью функции `fprintf`. Обращение к функции `printf` имеет следующий вид: `printf(F, s1, s2)`, где `F` – имя файловой переменной, `s1` – строка вывода, `s2` – список выводимых переменных. В строке вывода могут использоваться некоторые специальные символы:
 - `\b` – сдвиг текущей позиции влево;
 - `\n` – перевод строки;
 - `\t` – горизонтальная табуляция.

В строке вывода вместо выводимых переменных указывается строка преобразования следующего вида: `%n.мтип`, где `n` – ширина поля вывода; если `n` позиций недостаточно, то поле вывода расширяется до минимально необходимого, незаполненные позиции заполняются пробелами; `м` – количество знаков после десятичной точки. В качестве типа могут использоваться следующие латинские буквы: `c` – символ, `d` – целое десятичное число, `o` – целое восьмеричное число, `x` – целое шестнадцатеричное число, `f` – вещественное число вида `[±]dddd.dddd`, `e` – вещественное число в экспоненциальной форме вида `±d.dddd[e][±]ddd`, `g` – вещественное число вида `e` или `f` в зависимости от значения и точности, `s` – строка символов.

3. Закрывать файл с помощью функции `fclose(F)`, `F`.

Для чтения данных из файла необходимо:

1. Открыть файл с помощью функции `fopen`.
2. Считать данные из файла с помощью функции `fscanf`. Обращение к функции `fscanf` имеет вид: `A=fscanf(F, s1)`¹.
3. Закрывать файл (функция `fclose`).

В качестве примера рассмотрим запись в файл значений: $x = \frac{\pi}{7}$ и $y = \cos(x)$.

На рис. П2.10 представлено вычисление x и y и запись их в текстовый файл `abc.txt`.

На рис. П2.11 представлено содержимое файла `abc.txt`.

¹ В переменную `A` считываются данные из файла в соответствии со строкой `s1`.

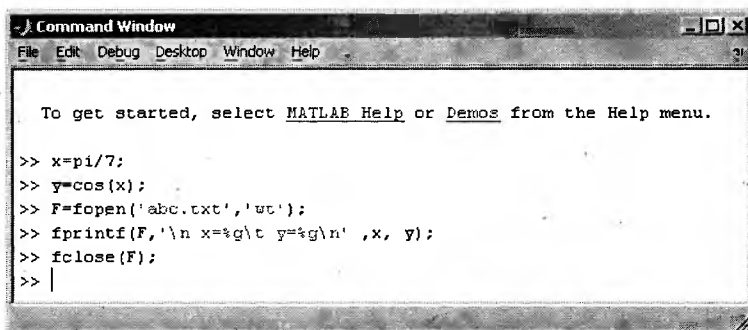


Рис. П2.10 ▼ Вычисление $x = \frac{\pi}{7}$ и $y = \cos(x)$ и запись их в текстовый файл abc.txt

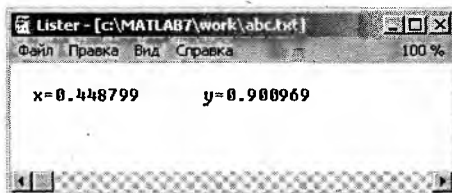


Рис. П2.11 ▼ Содержимое текстового файла abc.txt

Теперь рассмотрим два способа чтения данных из файла abc.txt. На рис. П2.12 представлено чтение содержимого файла abc.txt в переменную *h*. На рис. П2.13 оказано, как можно считать данные из файла abc.txt в две переменные *p* и *s*.

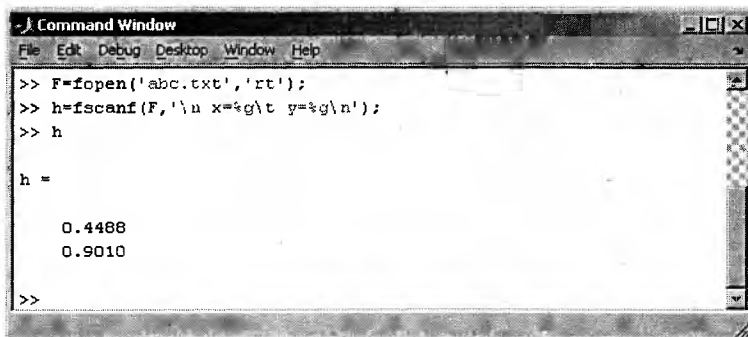


Рис. П2.12 ▼ Чтение содержимого файла abc.txt в переменную *h*

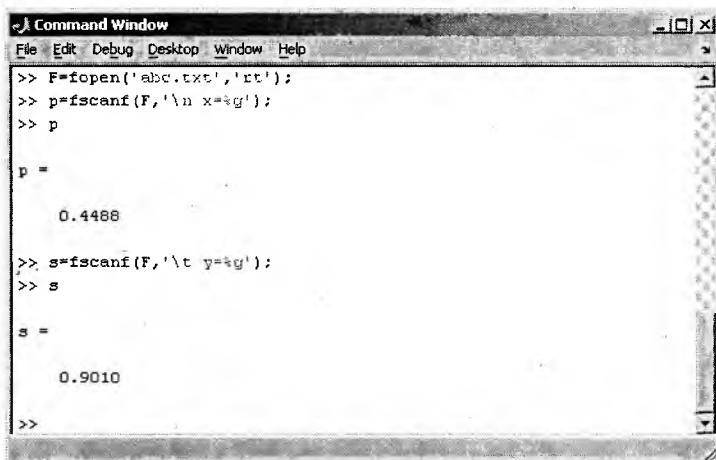


Рис. П2.13 ▼ Чтение содержимого файла abc.txt в переменные *p* и *h*

Работа с М-файлами в MATLAB. Режим командной строки в MATLAB не всегда удобен даже при довольно простых расчетах. Зачастую удобнее записать несколько команд в файл, а затем выполнить их последовательно. Файлы с командами MATLAB имеют расширение *.m*, для их создания необходимо выполнить команду **File ► New ► M-File**. На экране появится окно редактора М-файлов (рис. П2.14).

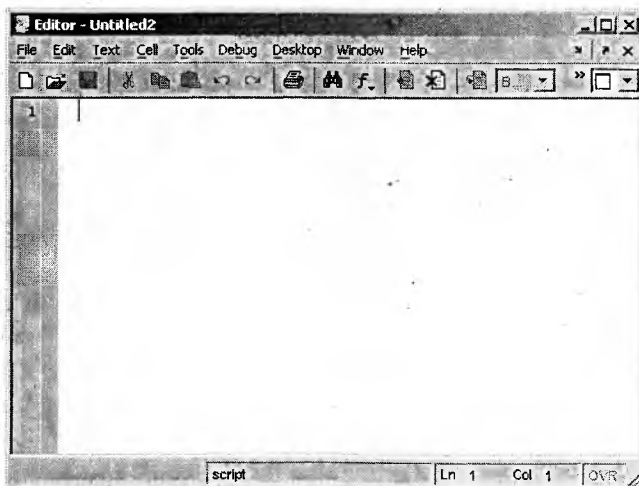


Рис. П2.14 ▼ Окно редактора М-файлов в MATLAB 7

После сохранения файла его можно вызвать по имени (без расширения .m) из командной строки MATLAB.

В М-файле могут находиться не только отдельные команды MATLAB, но и целые программы на М-языке. В программах на М-языке могут использоваться любые функции MATLAB (в том числе и пользовательские) и операторы для организации циклов и разветвлений. Если строка начинается с символа %, то это строка *комментария*.

Основные операторы М-языка MATLAB. Операторы М-языка MATLAB аналогичны операторам других языков программирования. Рассмотрим их последовательно. Оператор if имеет следующую структуру:

```
if (условие)
операторы1
elseif (условие)
операторы2
else
операторы 3
end
```

В операторе if условие – логическое выражение. В логических выражениях можно использовать логические операторы: ~ (отрицание), &(и), | (или); операторы отношений == (равно), ~= (не равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно).

Оператор while в MATLAB можно записать следующим образом:

```
while (условие)
операторы
end
```

Оператор for в MATLAB имеет следующую форму:

```
for (x=xn:dx:xk)
операторы
end
```

Здесь x – переменная цикла, xn, xk, dx – начальное, конечное значение и шаг переменной цикла.

Для аварийного выхода из цикла можно воспользоваться оператором break.

М-файлы можно оформлять как отдельные функции. В этом случае имя функции должно совпадать с именем файла, в котором она хранится (функция с именем primer должна храниться в файле primer.m), и первая строка этой функции должна иметь вид: function [y1, y2, ..., yn] = name_function(x1, x2, ..., xm). Здесь name_function – имя функции, x1, x2, ..., xm – список входных параметров функции, y1, y2, ..., yn – список выходных параметров функции. Функция, как и любой оператор, заканчивается словом end.

В качестве примера рассмотрим функцию `kol_max`, которая вычисляет количество максимумов в матрице X и представлена на рис. П2.15. А на рис. П2.16 показано, как вызывать функцию `kol_max` из командного окна MATLAB.

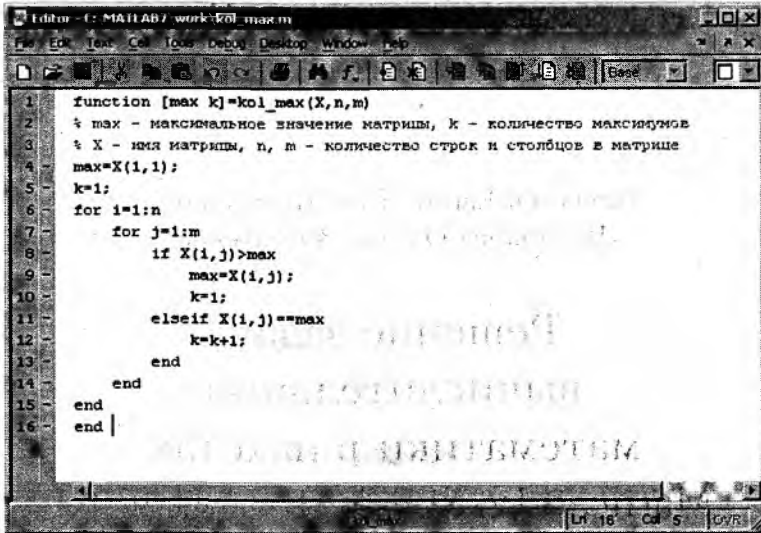


Рис. П2.15 ▼ Функция `kol_max`

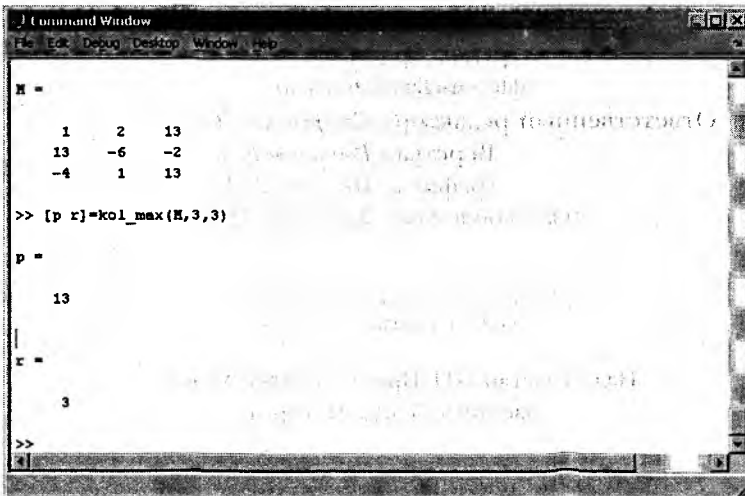


Рис. П2.16 ▼ Вызов функции `kol_max` из командного окна MATLAB

Приложение 3

Введение в Maple

Maple – математический пакет, который смело можно назвать одним из лидеров среди пакетов компьютерной алгебры. Maple одинаково успешно справляется как с символьными, так и с численными расчетами, но многие команды разработаны так, что даже при численном характере вычислений выдают аналитический результат. То есть основной акцент в Maple сделан на символьные вычисления. Кроме того, мощная графическая база, отлично сочетающаяся с вычислительными возможностями, позволяет использовать пакет как для простых, например студенческих задач, так и для более сложных – инженерных или научных.

Запуск Maple производится любым приемлемым для приложений Windows способом. Рис. П3.1 отображает рабочее окно системы, которое появится после запуска Maple.

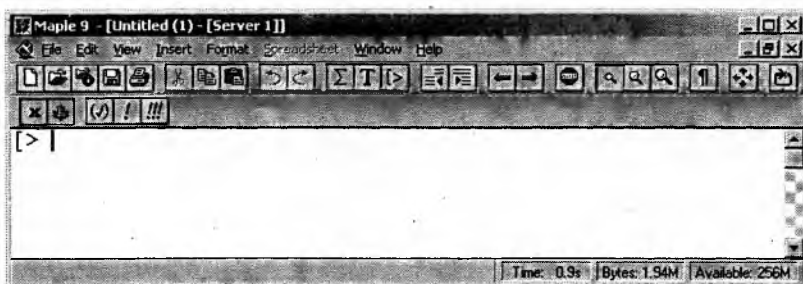


Рис. П3.1 ▼ Окно приложения Maple 9

Диалог с системой протекает в стиле «вопрос–ответ». Приглашением к вводу команд является символ `>` и мигающий курсор. Все команды в Maple отображаются шрифтом типа Courier красного цвета, а результаты выполнения команд – шрифтом Times синего цвета. Изменить цвет и тип шрифта можно с помощью пункта меню **Format ► Styles....** Команды и результаты их работы объединяются по группам. Группы отделены друг от друга квадратными скобками в левой части рабочей области.

Рассмотрим общие принципы редактирования команд в Maple. Ввод символов, так же как и в любом текстовом редакторе, выполняется при помощи

клавиатуры. Перемещение курсора по строке осуществляется клавишами $\leftarrow$ и $\rightarrow$ по введенным строкам рабочей области – $\uparrow$ и $\downarrow$. Клавиша **Backspace** удаляет символ слева от курсора, а **Del** – справа. Клавиша **Ins** задает режимы вставки и замены. Переключением регистров управляют клавиши **Shift** (включает верхний регистр) и **CapsLock** (меняет местами верхний и нижний регистры).

Особенности ввода команд заключаются в следующем. Знак точка с запятой ; в конце выражения (команды) указывает, что его результат должен быть выведен на экран, в отличие от знака двоеточия :, который отменяет вывод и может использоваться в качестве разделителя при записи нескольких выражений в одной строке. Для выполнения команды необходимо нажать клавишу **Enter**. Символ # предназначен для ввода текстового комментария.

Для записи математических выражений используют операторы сложения (+), вычитания (–), умножения (*), деления (/), возведения в степень (^). На рис. П3.2 приведены примеры вычисления элементарных арифметических выражений, из которых видно, что Maple работает с целыми, рациональными и вещественными числами. Признаком вещественного числа является разделительная точка, которая автоматически переводит вычисления из символьных в вещественные.

```

> (2+3)/10;
                                     1
                                     2
> (2+3)/10.;
0.50000000000
> 7^2*(4-1);
147
> 10^(-6)*3;
      3
     ---
    1000000
> 10^(-6)*3.;
>
0.3000000000 10^-5

```

Рис. П3.2 ▼ Элементарные вычисления в Maple

В Maple встроено более трех тысяч различных функций, обратиться к которым можно, указав имя и список параметров. Доступ к информации о той или иной функции осуществляется с помощью команды ?<имя функции>. На рис. П3.3 приведены примеры вычислений с использованием элементарных математических функций. Обратите внимание, что результат вычисления функции с

целочисленным или рациональным аргументом, если возможно, будет представлен через математическую константу (например, π или e), в противном случае функция повторится в строке вывода.

```
> 5*exp(2);
                                     5 e2
> arccos(1/2);
                                      $\frac{\pi}{3}$ 
> ln(5);
                                     ln(5)
> sin(3/4.);
0.6816387600
```

Рис. ПЗ.3 ▼ Элементарные математические функции в Maple

Имя переменной в Maple может состоять из букв, цифр и некоторых специальных символов (например, можно использовать знак подчеркивания), но обязательно должно начинаться с буквы. Запрещается использовать в качестве имен переменных ключевые слова Maple. Команда ?<имя переменной> позволяет проверить, уникальна ли группа символов, составляющая <имя переменной>. Если результатом работы команды станет появление окна справки, это значит, что <имя переменной> зарезервировано и лучше его не использовать¹. Ограничений на длину имени в Maple практически нет (не более 524275 символов). Кроме того, Maple различает прописные и строчные буквы, то есть переменные Name и NAME разные.

Для определения переменных в Maple существует оператор присваивания :=, предназначение которого не отличается от функций подобных операторов в других математических пакетах или языках программирования: некоторой переменной присваивается конкретное значение. Но особенность пакета Maple заключается в том, что переменные в нем рассматриваются как символьные объекты и могут использоваться в математических выражениях и функциях без предварительного определения. Более того, в Maple существуют функции, которые работают только с символьными или необъявленными переменными. Примеры определения и использования переменных приведены на рис. ПЗ.4. Так, для отыскания производной функции $\lg(x)$ пришлось переопределить переменную x как строковую при помощи оператора $x := 'x'$. Проинтегрировать

¹ Maple, в отличие от большинства языков программирования и математических пакетов, сообщения об ошибке не выдает, но функция или команда, связанная с параметром <имя переменной>, перестанет работать.

функцию $\sin^2(y)$ удалось, присвоив переменной y статус неопределенной оператором $y := \text{evaln}(y)$. Отменить все присваивания можно командой `restart`, но делать это можно лишь в том случае, если вы уверены, что все определения, сделанные в документе, причем не только в текущем, уже не нужны.

```
> sqrt(x^2-y^2);#Переменные x и y не определены

$$\sqrt{x^2 - y^2}$$

> x:=3:y:=2.1: #Определение переменных x и y
> sqrt(x^2-y^2);#Вычисление значения выражения
2.142428529
> diff(log[10](x),x);#Вычисление производной от функции
lg(x), переменная x определена выше как число
Error, wrong number (or type) of parameters in function diff
> x:='x':diff(log[10](x),x);#Вычисление производной от
функции lg(x), переменная x определена как символьная.

$$\frac{1}{x \ln(10)}$$

> int(sin(y)^2,y);#Вычисление интеграла, переменная
определена.
Error, (in int) wrong number (or type) of arguments
> y:=evaln(y);#Переменная не определена

$$y=y$$

> int(sin(y)^2,y);#Вычисление интеграла, переменная не
определена.

$$-\frac{1}{2} \cos(y) \sin(y) + \frac{y}{2}$$

```

Рис. П3.4 ▼ Примеры определения переменных и вычисления выражений

Оператор присваивания можно использовать и для определения новых функций как одного, так и нескольких аргументов.

Один из способов задания функции заключается в присваивании переменной некоторого выражения:

<имя переменной>:=<выражение>.

Однако подобную конструкцию функцией назвать можно с трудом, хотя бы потому, что в ней нет списка аргументов (параметров) и их значения нужно определять заранее с помощью оператора присваивания (рис. П3.5).

Следующий способ задания функции близок к обычной математической записи:

<имя функции>(<список параметров>):=<выражение>.

```
[> restart;
> Y:=sin(x)+cos(x);#Переменной Y присваивается значение
    выражения sin(x)+cos(x), при этом Y - не функция, а
    выражение.
                                
$$Y = \sin(x) + \cos(x)$$

> x:=3/4; Y;#Значение выражения Y в символьном виде:
                                
$$\sin\left(\frac{3}{4}\right) + \cos\left(\frac{3}{4}\right)$$

> evalf(Y);#Значение выражения Y в вещественном
    формате:
                                1.413327629
```

Рис. ПЗ.5 ▼ Пример создания и использования функции без параметров

На рис. ПЗ.6 приведен пример использования функции, заданной таким образом, для построения графика (подробно о построении графиков см. далее).

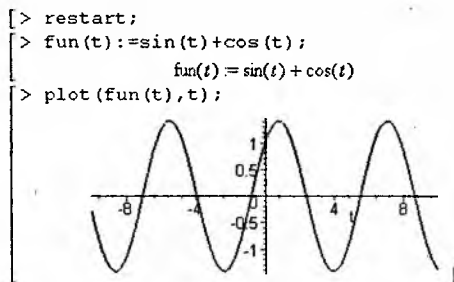


Рис. ПЗ.6 ▼ Примеры создания и использования функции с параметрами

Но все-таки наиболее полноценным способом задания функции в Maple будет запись, в которой используется оператор соответствия в виде стрелки $\rightarrow$:

$\langle \text{имя функции} \rangle := (\langle \text{список параметров} \rangle) \rightarrow \langle \text{выражение} \rangle$.

В приведенной конструкции $\langle \text{список параметров} \rangle$ – это формальные параметры функции, $\langle \text{имя функции} \rangle$ – команда, которая воздействует на аргументы, указанные в списке параметров, $\langle \text{выражение} \rangle$ – математическое выражение, которое соответствует этой команде. После определения функции таким способом ее вызов осуществляется так:

$\langle \text{имя функции} \rangle (\langle \text{список параметров} \rangle)$,

где $\langle \text{список параметров} \rangle$ – локальные параметры, которые сохраняют свои значения только в теле функции $\langle \text{имя функции} \rangle$. Примеры, приведенные на рис. ПЗ.7, иллюстрируют сказанное.

```

> restart;
> f:=t->t*exp(t); #Задание функции f с параметром t

$$f = t \rightarrow t e^t$$

> t:=0;f(0.7); #Глобальная переменная t не изменила
своего значения после вызова функции f

$$t = 0$$

1.409626895
> z:=(q,p)->q^2+p^2; #Задание функции z с параметрами
q и p

$$z = (q, p) \rightarrow q^2 + p^2$$

> z(1,2); #Обращение к функции z
5

```

Рис. П3.7 ▼ Примеры создания функций при помощи оператора соответствия

Рассмотрим, как в Maple проходит работа с данными множественного типа. Важным типом данных в пакете являются *списки*. Задаются они с помощью квадратных скобок []. На рис. П3.8 видно, что списки преобразуются и выводятся в том порядке, в котором были заданы.

```

> [2, 4, 6, 8, 10]; #Список из пяти целых чисел
[2, 4, 6, 8, 10]
> [a, b, c]; #Список из трех символов
[a, b, c]
> s:=[1/2, 3/4+3/2, 7/8-1/4]; #Список выражений

$$S = \left[ \frac{1}{2}, \frac{9}{4}, \frac{5}{8} \right]$$

> F:=[x^2, x^3, x^4, x^5]; #Список функций

$$F = [x^2, x^3, x^4, x^5]$$

> x:=2:F; #Список значений функций
[4, 8, 16, 32]

```

Рис. П3.8 ▼ Создание списков

Одним из применений списков является задание векторов и матриц. Для создания вектора используют функцию `array(in..ik, V)`, которая возвращает одномерный массив с индексами от `in` до `ik` и значениями из списка `V`. Матрица формируется командой `array(in..ik, jn..jk, M)`, результатом работы которой является двумерный массив с номерами строк от `in` до `ik`, столбцов от `jn` до `jk` и значениями из списка `M`. Примеры ввода массивов приведены на рис. П3.9. Обратите внимание, что обращение к элементу массива происходит с указанием индексов элемента в квадратных скобках.

На рис. П3.10 и П3.11 приведены основные операции над матрицами и векторами, которые допустимы в Maple. Все они выполнены при помощи функции

```

> array(1..5, [2, 4, 6, 8, 10]); #Вектор
      [2, 4, 6, 8, 10]
> V:=[a+b, c-d]; #Одномерный список
> v:=array(1..2, V); #Вектор
      v=[a+b, c-d]
> M:=array(1..3, 1..3, [[1, 2, 3], [4, 5, 6], [7, 8, 9]]); #Матрица
      M=
      [ 1  2  3 ]
      [ 4  5  6 ]
      [ 7  8  9 ]
> v[1]; #Обращение к элементу вектора v
      a+b
> M[2,2]; #Обращение к элементу матрицы M
      5

```

Рис. П3.9 ▼ Создание векторов и матриц

`evalm(<матричное выражение>)`. Эта команда вычисляет значение матричного выражения, в котором могут быть использованы матричные операторы: `+` (сложение), `-` (вычитание), `&*` (умножение), `^` (возведение в степень).

```

> V:=array(1..3, [1, 3, 5]);
      V=[1, 3, 5]
> V[2]:=0:evalm(V);
      [1, 0, 5]
> W:=evalm(2+V);
      W=[3, 2, 7]
> evalm(V+W);
      [4, 2, 12]
> evalm(V*2-W);
      [-1, -2, 3]
> evalm(sin(V/2.));
      [0.4794255386, 0., 0.5984721441]

```

Рис. П3.10 ▼ Операции с векторами

Нетрудно заметить, что в основную библиотеку Maple включены минимальные средства для работы с векторами и матрицами. Основные функции содержатся в пакете `linalg`, специально предназначенном для решения задач линейной алгебры. Пакет содержит более ста функций и вызывается командой `with(linalg)` так, как показано на рис. П3.12.

Перейдем к вопросам построения графиков в Maple. Для построения *двумерных графиков* используют функцию

`plot(<функция>, X [, Y, <опции>])`,

для которой `<функция>` – имя функции или выражение, описывающее функцию, график которой нужно построить, `X` – имя аргумента, возможно задание области


```

> A:=array(1..2,1..2,[[3,2],[4,1]]);#Матрица A

$$A = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$$

> M:=evalm(A*3);#Матрица M=A*3

$$M = \begin{bmatrix} 9 & 6 \\ 12 & 3 \end{bmatrix}$$

> evalm(M^(-1));#Матрица, обратная к M

$$\begin{bmatrix} -\frac{1}{15} & \frac{2}{15} \\ \frac{4}{15} & -\frac{1}{5} \end{bmatrix}$$

> evalm(A+M);#Сложение матриц

$$\begin{bmatrix} 12 & 8 \\ 16 & 4 \end{bmatrix}$$

> evalm(A&*M);#Умножение матриц A и M, где &*
оператор матричного умножения

$$\begin{bmatrix} 51 & 24 \\ 48 & 27 \end{bmatrix}$$

> evalm(M^2);#То же, что M умножить на M по
правилу умножения матриц

$$\begin{bmatrix} 153 & 72 \\ 144 & 81 \end{bmatrix}$$


```

Рис. П3.11 ▼ Матричные операции

```

> with(linalg);
Warning, the protected names norm and trace have been redefined and
unprotected

```

```

[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian,
addecol, addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout,
blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan, companion,
concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge,
dotprod, eigenvals, eigenvalues, eigenvectors, eigenvects, enltermatrix, equal,
exponential, extend, ffgausselim, fibonacci, forwardsub, frobenius, gausselim,
gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian, hlibert,
htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith, issimilar, iszero,
jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix, minor,
minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent,
pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspace,
rowspan, rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector,
sumbasis, swapcol, swaprow, sylveste, toeplitz, trace, transpose, vandermonde,
vecpotent, vectdim, vector, wronskian]

```

Рис. П3.12 ▼ Загрузка пакета для решения задач линейной алгебры

его изменения по оси абсцисс, Y – необязательный параметр, задает область изменения графика по оси ординат, <опции> – необязательный параметр или список параметров, задает стиль построения графика (толщину и цвет линий, тип линий и т.п.).

На рис П3.13 изображен график функции, заданной парами точек. Он построен при помощи функции `plot`, в которой <функция> – это имя списка данных XY , определенного выше, параметры X и Y показывают ограничения по оси абсцисс от 0 до 70, а по оси ординат – от 60 до 130.

```
> XY:=[0,66.7],[4,71.0],[10,76.3],[15,80.6],[21,85.7],
      [29,92.9],[36,99.4],[51,113.6],[68,125.1]];
> plot(XY,x=0..70,y=60..130);
```

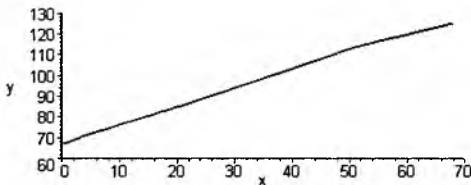


Рис. П3.13 ▼ Построение графика функции, заданной парами точек

Построение графика функции $y = |x| / 2$, изображенного на рис. П3.14, было выполнено самым простым способом. Команда `plot` выполнена только с двумя обязательными параметрами – выражением, описывающим функцию, и аргументом от которого эта функция зависит. И поскольку значения аргумента неопределенны, график создается автоматически в диапазоне от -10 до 10 .

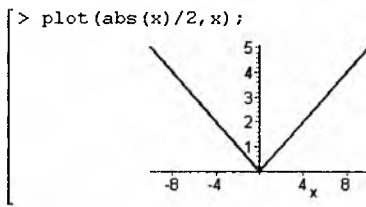


Рис. П3.14 ▼ Простейший способ построения графика функции

Рис. П3.15 содержит пример построения графиков нескольких функций в одной графической области. Обратите внимание на то, что имена функций заданы в виде списка, а параметр X определен по умолчанию. Кроме того, введены опции, изменяющие цвет, стиль, тип и толщину линий. Заметим, что форматирование двумерного графика можно выполнить, задавая параметры

непосредственно в функции `plot`, а можно при помощи контекстного меню, которое вызывается щелчком правой кнопкой мыши по области построения графика.

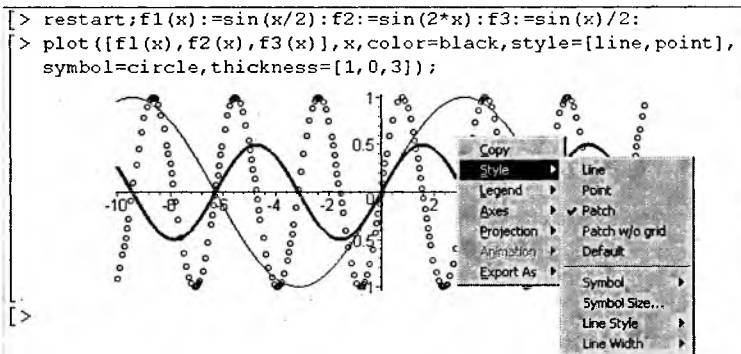


Рис. ПЗ.15 ▼ Построение графика нескольких функций

На рис. ПЗ.16 приведен пример быстрого построения трехмерного графика с помощью функции `plot3d`, которая имеет такие же параметры, как и функция `plot`. Поверхности также можно форматировать, изменяя параметры функции `plot3d` и используя контекстное меню.

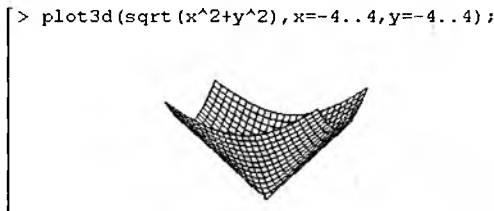


Рис. ПЗ.16 ▼ Быстрое построение поверхности

Рассмотрим основные *функции математического анализа*, встроенные в пакет. *Дифференцирование* в Maple выполняется при помощи функции `diff(f(x1[, x2, ..., xn]), x1[, x2, ..., xn])`, где $f(x_1, x_2, \dots, x_n)$ – дифференцируемая функция или алгебраическое выражение, а $x_1, x_2, \dots, x_n$ – переменные, по которым производится дифференцирование. Функция `Diff` нужна для записи операции дифференцирования в стандартной математической форме. Примеры дифференцирования приведены на рис. ПЗ.17.

```

> diff(sin(x), x);
cos(x)
> f:=t->diff(cos(t), t); #Функция от производной
f:=t->diff(cos(t), t)
> f(pi); #Значение производной в точке
-sin(pi)
> diff(sin(x)^4, x, x, x); #Производная третьего порядка,
оператор diff(sin(x)^4, x$3) выдаст тот же результат
24 sin(x) cos(x)^3 - 40 sin(x)^3 cos(x)
> z(x, y) := (y^2 - x^2) * (1 + x)^2:
> Diff(z(x, y), y) = diff(z(x, y), y);

$$\frac{\partial}{\partial y} ((y^2 - x^2)(1 + x)^2) = 2y(1 + x)^2$$

> Diff(z(x, y), x$2) = diff(z(x, y), x$2);

$$\frac{\partial^2}{\partial x^2} ((y^2 - x^2)(1 + x)^2) = -2(1 + x)^2 - 8x(1 + x) + 2y^2 - 2x^2$$


```

Рис. П3.17 ▼ Примеры дифференцирования

Вычислить *интеграл* можно, если воспользоваться функцией `int(f(x), x[= a..b])`, в которой $f(x)$ – подынтегральная функция, x – переменная интегрирования, $a..b$ – пределы интегрирования. Функция `Int` предназначена для вывода интегральных выражений в более привычном, математическом виде. На рис. П3.18 приведены примеры вычисления интегралов.

```

> int(sin(x), x);
-cos(x)
> Int(ln(x)^2, x);

$$\int \ln(x)^2 dx$$

> Int(sin(x)*exp(2*x), x) = int(sin(x)*exp(2*x), x);

$$\int e^{(2x)} \sin(x) dx = -\frac{1}{5} e^{(2x)} \cos(x) + \frac{2}{5} e^{(2x)} \sin(x)$$

> int(cos(3.*x), x=0..3/2);
-0.3258433725
> Int(ln(x)^2, x=1..2) = int(ln(x)^2, x=1..2);

$$\int_1^2 \ln(x)^2 dx = 2 \ln(2)^2 - 4 \ln(2) + 2$$


```

Рис. П3.18 ▼ Примеры интегрирования

Операция суммирования выполняется в Maple при помощи функции `sum(f, k[= m..n])`, f – функция, задающая члены суммируемого ряда, k –

рования, $m..n$ – пределы изменения индекса k . Как видно на рис. П3.19, функция `sum` вычисляет сумму членов ряда, а `Sum` – записывает сумму, так как это принято в математике. Обратите внимание, что `infinity` – встроенная константа, обозначающая бесконечность, а функция `evalf` (<выражение>) вычисляет значение <выражения> в формате с плавающей точкой.

```
> Sum(n!, n=1..5);
```

$$\sum_{n=1}^5 n!$$

```
> sum(n!, n=1..5);
```

$$153$$

```
> sum(sin(k), k=1..3)=evalf(sum(sin(k), k=1..3));
```

$$\sin(1) + \sin(2) + \sin(3) = 1.891888420$$

```
> Sum(1/p^2, p=1..infinity)=sum(1/p^2, p=1..infinity);
```

$$\sum_{p=1}^{\infty} \frac{1}{p^2} = \frac{\pi^2}{6}$$

```
> A:=array([7, 3, 9, 1, 2]);
```

$$A = [7, 3, 9, 1, 2]$$

```
> Sum(A[i], i=1..5)=sum(A[i], i=1..5);
```

$$\sum_{i=1}^5 A_i = 22$$

Рис. П3.19 ▾ Примеры суммирования

Произведение членов последовательности позволяет функция `product` ($m..n$), параметры которой аналогичны параметрам функции `sum`. Действие функций `product` и `Product` понятно из примеров, на рис. П3.20.

```
> product(1+1/m^3, m=1..20);
```

$$\frac{14936861968691671606597909}{6158757858272870400000000}$$

```
> Product(1+1/m^3, m=1..20)=product(1+1./m^3, m=1..20);
```

$$\prod_{m=1}^{20} \left(1 + \frac{1}{m^3}\right) = 2.425304311$$

```
> Product(sqrt(p), p);
```

$$\prod_p \sqrt{p}$$

Рис. П3.20 ▾ Примеры вычисления произведений

Для вычисления *предела* существует функция `limit(f, x = a[, <опция>])`. Здесь f – алгебраическое выражение, $x = a$ – точка, в которой будет вычислен предел (может принимать значение `infinity` – бесконечность), `<опция>` указывает направление поиска предела (`left` – слева, `right` – справа). Функция `Limit` используется для представления вычисляемого предела в стандартном математическом виде (рис. П3.21).

```
> Limit(1/n^2, n=infinity)=limit(1/n^2, n=infinity);
```

$$\lim_{n \rightarrow \infty} \frac{1}{n^2} = 0$$

```
> Limit(sin(x)/x, x=0)=limit(sin(x)/x, x=0);
```

$$\lim_{x \rightarrow 0} \frac{\sin(x)}{x} = 1$$

```
> Limit(arctan(1/x), x=0)=limit(arctan(1/x), x=0);
```

$$\lim_{x \rightarrow 0} \arctan\left(\frac{1}{x}\right) = \text{undefined}$$

```
> Limit(arctan(1/x), x=0, right)=limit(arctan(1/x), x=0, right);
```

$$\lim_{x \rightarrow 0+} \arctan\left(\frac{1}{x}\right) = \frac{\pi}{2}$$

```
> Limit(arctan(1/x), x=0, left)=limit(arctan(1/x), x=0, left);
```

$$\lim_{x \rightarrow 0-} \arctan\left(\frac{1}{x}\right) = -\frac{\pi}{2}$$

Рис. П3.21 ▼ Примеры вычисления пределов

Начиная с пятой версии в Maple появились *математические палитры*. Они значительно облегчают ввод математических выражений, что, безусловно, очень

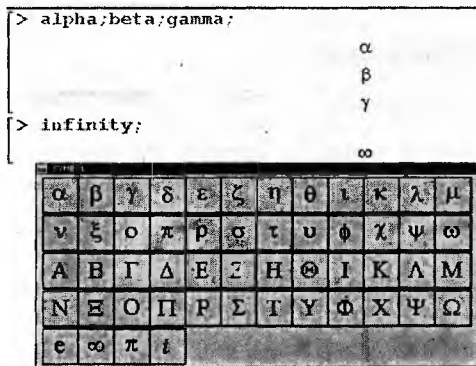


Рис. П3.22 ▼ Палитра греческих и специальных символов


```

<имя процедуры>:=proc(<список формальных параметров>))
<тело процедуры>
end;

```

С помощью конструкции <имя переменной>::<тип> можно определять тип переменных из списка формальных параметров. Фактические параметры подставляются на место формальных при вызове процедуры:

```

<имя процедуры>([список фактических параметров]).

```

Несоответствие типов формальных и фактических параметров ведет к сообщению об ошибке. Пример создания и вызова процедуры приведен на рис. П3.25. Здесь решается задача вычисления площади треугольника по формуле Герона. Обратите внимание, что конструкция end; вызвала повторение текста процедуры в строке вывода. Символ двоеточия в конце процедуры end: позволит избежать этого.

```

> S:=proc(a,b,c)#Описание процедуры
> P:=(a+b+c)/2;
> S:=sqrt(P*(P-a)*(P-b)*(P-c));
> end:
Warning, 'P' is implicitly declared local to procedure 'S'
Warning, 'S' is implicitly declared local to procedure 'S'

S:=proc(a,b,c)
local P,S;
    P:=1/2*a+1/2*b+1/2*c; S:=sqrt(P*(P-a)*(P-b)*(P-c))
end proc
> S(5,4,3);#Вызов процедуры
6
> S(2.,8,7);#Вызов процедуры
6.437196595

```

Рис. П3.25 ▼ Создание и вызов процедуры

Maple-процедура возвращает значение последнего выражения в ее теле. Вернуть в качестве результата любое выражение позволяет специальный оператор RETURN(<выражение>). Пример применения *оператора возврата* приведен на рис. П3.26.

Условный оператор применяется, если в зависимости от некоторого условия требуется выполнить либо одно, либо другое действие, и имеет следующую структуру:

```

if <условие1> then <выражение1>
[elif <условие2> then <выражение2>]
[else <выражение3>]
fi;

```



```

> S:=proc(a,b,c) #Описание процедуры
> P:=(a+b+c)/2: RETURN(P); #Результатом работы процедуры
    будет значение выражения P
> S:=sqrt(P*(P-a)*(P-b)*(P-c));
> end:
Warning, 'P' is implicitly declared local to procedure 'S'

Warning, 'S' is implicitly declared local to procedure 'S'

> S(2,8,7); #Вызов процедуры

```

$$\frac{17}{2}$$

Рис. ПЗ.26 ▼ Применение оператора возврата

Но чаще всего используют такие конструкции:

if <условие1> then <выражение1> fi;

ИЛИ

if <условие1> then <выражение1> else <выражение2> fi;

В условиях могут использоваться логические конструкции, содержащие знаки сравнения (<, >, <=, >=, =, <>) и логические операторы and, or, not.

На рис. ПЗ.27 приведены примеры использования условных операторов.

```

> F:=proc(a,b)
> c:=0:
> if a<b then c:=a+b fi: c:=c+12;
> end:
Warning, 'c' is implicitly declared local to procedure 'F'

> F(3,6); F(8,1); #

```

$$\begin{array}{r} 21 \\ 12 \end{array}$$

```

> FU:=proc(x,y)
> if x>y then print("x>y") else print("x<y") fi;
> end:
> FU(3,7); FU(8,1);

```

$$\begin{array}{r} \text{"x<y"} \\ \text{"x>y"} \end{array}$$

```

> FUN:=proc(c,d)
> if c>d then print("c>d")
> elif c<d then print("c<d")
> else print("c=d")
> fi;
> end:
> FUN(3,7); FUN(8,1); FUN(1,1);

```

$$\begin{array}{r} \text{"c<d"} \\ \text{"c>d"} \end{array}$$

Циклический оператор представлен в Maple следующей конструкцией:

```
[for <имя переменной>] [from <выражение1>] [to <выражение3>]
[by <выражение2>] [while <выражение4>] do <тело цикла> od;
```

Эта запись означает, что <тело цикла> выполнится для значений <имя переменной> (параметр цикла), изменяющейся в диапазоне от <выражение1> до <выражение3>, с шагом <выражение2>. Параметр <выражение4> задает условие продолжения <тела цикла>.

В упрощенном варианте циклические конструкции могут выглядеть так:

```
for <имя переменной> from <выражение1> to <выражение3> do <тело цикла> od;
for <имя переменной> from <выражение1> to <выражение3> by <выражение2> do
<тело цикла> od;
```

или

```
while <выражение4> do <тело цикла> od;
```

На рис. П3.28 рассмотрены примеры создания циклов.

```
> SUM:=proc();#Сумма чисел 1+1.5+2+...+10
> S:=0;
> for i from 1 to 10 by 0.5 do
> S:=S+i od;
> end;
>
Warning, 'S' is implicitly declared local to procedure 'SUM'
Warning, 'i' is implicitly declared local to procedure 'SUM'
> SUM();
104.5
> SU:=proc(m,n)#Сумма целых чисел от m до n
> S:=0;for i from m to n do S:=S+i od;
> end;
Warning, 'S' is implicitly declared local to procedure 'SU'
Warning, 'i' is implicitly declared local to procedure 'SU'
> SU(10,20);
165
> S:=proc()#Сумма чисел 10+8+6+4+2
> s:=0;i:=10;while i>=2 do s:=s+i: i:=i-2;od;
> RETURN(s);
> end;
Warning, 's' is implicitly declared local to procedure 'S'
Warning, 'i' is implicitly declared local to procedure 'S'
> S();
```

Иногда бывает необходимо завершить цикл досрочно. Для этого предназначен оператор прерывания *break*. Для прерывания выполнения текущей операции используется оператор *next*. Применение этих операторов показано на рис. ПЗ.29.

```
> G:=proc(X,n)#Вычисляет номер первого нулевого
    элемента в массиве
> for i from 1 to n do
> if X[i]=0 then k:=i: break: fi: od:RETURN(k);
> end:
Warning, 'i' is implicitly declared local to procedure 'G'

Warning, 'k' is implicitly declared local to procedure 'G'

> x:=[1,2,3,0,4,5,6]:G(x,7);

4

> F:=proc(X,n)#Вычисление произведения ненулевых
    элементов в массиве
> P:=1;for i from 1 to n do
> if X[i]=0 then next else P:=P*X[i] fi: od:
    RETURN(P);
> end:
Warning, 'P' is implicitly declared local to procedure 'F'

Warning, 'i' is implicitly declared local to procedure 'F'

> F(x,7);|
```

720

Рис. ПЗ.29 ▼ Использование операторов прерывания

Операторы *quit*, *done* и *stop* прерывают выполнение текущей программы. Документ, созданный в среде пакета, хранится в отдельном файле с расширением *mws*. Рассмотрим основные средства, которыми располагает Maple для работы с файлами. Они практически ничем не отличаются от аналогичных приложений:

- создать новое окно (каждый новый документ выводится с условным названием *Untitled:1*, *Untitled:2*, *Untitled:3* и т.д.) – **File** ➤ **New**;
- сохранить документ – **File** ➤ **Save**;
- открыть существующий документ – **File** ➤ **Open**;
- закрыть документ – **File** ➤ **Close**;
- завершить сеанс работы (если файл не был сохранен, то появится окно *Save As*) – **File** ➤ **Exit**.

Редактирование документа в Maple осуществляется так же, как в любом текстовом редакторе. Выделять фрагменты документа можно мышью, команда

Edit ► Select All выделяет все содержимое документа. Для удаления выделенного фрагмента документа можно использовать клавишу **Del**. Команды **Edit ► Copy**, **Edit ► Cut**, **Edit ► Past** предназначены для копирования и перемещения фрагментов документа.

Важную роль в работе с пакетом Maple играет *справочная система*. Вызывается она командой **Help** и содержит следующие разделы:

- **Introduction** – введение в справочную систему;
- **Help on context** – контекстная справка;
- **New User's Tour** – обучающая система;
- **What's New** – новые возможности системы;
- **Using Help** – правила использования справочной системы;
- **Glossary** – словарь терминов;
- **Topic Search** – предметный поиск по заданному слову;
- **Full Text Search** – предметный поиск по всему тексту справки;
- **History** – история поиска;
- **Save to Database** – запись данных в базу;
- **Remove Topic** – восстановление базы данных предметного поиска;
- **Balloon Help** – включение всплывающих подсказок;
- **Register Maple** – регистрация Maple;
- **About Maple** – информация о системе Maple.

Уникальность справочной системы Maple заключается в том, что только в ней можно найти полную информацию обо всех функциях пакета. Кроме того, в ней содержится огромное количество примеров использования функций и операторов. Однако следует отметить, что справка написана на английском языке и достаточно объемна.

Заключение

На этом первое знакомство с методами вычислительной математики можно считать законченным. Авторы надеются, что, прочитав эту книгу, читатель сможет применять численные методы и математические пакеты при решении прикладных задач. Если после прочтения книги у читателя возникнет желание исправить что-нибудь в блок-схемах и программах, приведенных в самоучителе, то авторы сочтут свою задачу выполненной.

Список используемой литературы

1. Аладьев В. З., Гершгорн Н. А. Вычислительные задачи на персональном компьютере. – К.: Техника, 1991. – 245 с.
2. Алексеев Е. Р., Чеснокова О. В. Mathcad 12. – М.: НТ Пресс, 2005. – 345 с.
3. Алексеев Е. Р., Чеснокова О. В., Павлыш В. Н., Славинская Л. В. Турбо Паскаль 7.0. – М.: НТ Пресс, 2004. – 270 с.
4. Ануфриев И. Е., Смирнов А. Б., Смирнова Е. Н. MATLAB 7. – СПб.: БХВ-Петербург, 2005. – 1104 с.
5. Березин И. С., Жидков Н. П. Методы вычислений. – М.: Наука, 1966. – 632 с.
6. Бронштейн И. Н., Семендяев К. А. Справочник по математике для инженеров и учащихся вузов. – М.: Наука, 1981. – 720 с.
7. Будак Б. М., Самарский А. А., Тихонов А. Н. Сборник задач по математической физике. – М.: ФИЗМАТЛИТ, 2003. – 688 с.
8. Васильев А. Н. Maple 8. Самоучитель. – М.: Издательский дом «Вильямс», 2003. – 352 с.
9. Вержбицкий В. М. Основы численных методов. – М.: Высшая школа, 2002. – 840 с.
10. Голосков Д. П. Уравнения математической физики. Решение задач в системе Maple. – СПб.: Питер, 2004. – 539 с.
11. Демидович Б. П., Марон И. А., Шувалова В. З. Численные методы анализа. – М.: Наука, 1967. – 368 с.
12. Дьяконов В. Mathcad 2000: Учебный курс. – СПб.: Питер, 2000. – 592 с.
13. Дьяконов В. Maple 6: Учебный курс. – СПб.: Питер, 2001. – 608 с.
14. Кирьянов Д. В. Самоучитель Mathcad 12. – СПб.: БХВ-Петербург, 2004. – 576 с.
15. Мартынов Н. Н., Иванов А. П. MATLAB 5.x. Вычисления, визуализация, программирование. – М.: Кудиц-образ, 2000. – 336 с.
16. Метьюз Джон Г., Финк Куртис Д. Численные методы. Использование MATLAB: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 720 с.
17. Плис А. И., Сливина Н. А. Mathcad: Математический практикум для экономистов и инженеров. – М.: Финансы и статистика, 1999. – 656 с.

18. Салманов О. Н. Математическая экономика с применением Mathcad и Excel. – СПб.: БХВ-Петербург, 2003. – 464 с.
19. Семененко М. Г. Математическое моделирование в Mathcad. – М.: Альтекс-А, 2003. – 208 с.
20. Сигорский В. П. Математический аппарат инженера. – К.: Техніка, 1975. – 768 с.
21. Тихонов А. Н., Самарский А. А. Уравнения математической физики. – М.: Наука, 1966. – 724 с.

Предметный указатель

В

Вычисление

- значения многочлена 184
- производных многочлена 186

Г

Графическая среда pdetool 378

Д

Деление многочлена на линейный множитель 184

Дифференциальное уравнение в частных производных

- волновое уравнение 345
- Гельмгольца 344
- гиперболическое 344
- Лапласа 344
- метод сеток (конечных разностей) 351
- метод Фурье (метод разделения переменных) 346
- начальные и граничные условия 345
- неоднородное 344
- однородное 344
- параболическое 344
- принцип суперпозиции 346
- Пуассона 344
- телеграфное 345
- теплопроводности 345
- эллиптическое 344

Дифференциальные уравнения 289

- n -го порядка 289
- решение 289
 - метод Адамса 307
 - метод Кутты–Мерсона 303

метод Милна 309

метод Рунге–Кутты второго порядка 299

метод Рунге–Кутты первого порядка 299

метод Рунге–Кутты четвертого порядка 300

метод Хорна 299

метод Эйлера 292

методы Рунге–Кутты 299

модифицированный метод Милна 310

модифицированный метод Эйлера 296

З

Задача

Коши 290

Штурма–Лиувилля 347

собственные значения 347

собственные функции 347

И

Индекс корреляции 147

Интегральное уравнение 408

Вольтерра второго рода 410

Вольтерра первого рода 410

Вольтерра третьего рода 410

метод конечных сумм 410

Фредгольма второго рода 409

Фредгольма первого рода 408

Фредгольма третьего рода 409

Интерполяция 110

канонический полином 112

линейная 122
полином Лагранжа 116
полином Ньютона 113
сплайн-интерполяция 117

К

Квадратичная форма
 приведение к каноническому
 виду 70
Коэффициент
 корреляции 145
 регрессии 145
Краевая задача 290
Критерий Стьюдента 145

Л

Линия регрессии 145

М

Матрица 11
 LU-разложение 30, 53
 QR-разложение 32, 53
 верхняя треугольная 11
 вырожденная 12
 диагональная 12
 единичная 12
 идемпотентная 13
 инволютивная 13
 квадратная 11
 кососимметрическая 13
 невырожденная 12
 нижняя треугольная 12
 норма 45, 52, 72
 нулевая 11
 обратная 12, 20
 обратная данной 82, 93
 операции сравнения 12
 определитель 12, 24, 80, 90
 ортогональная 13, 34
 преобразование по Гауссу 53, 82
 произведение матриц 12

прямоугольная 11
разложение по Халецкому 53, 82
ранг 14, 52, 90
симметрическая 13
след 52
собственные значения 53
собственный вектор 53
сумма 12
транспонированная 12, 80
число обусловленности 46, 52, 72
Матричное уравнение 57, 82
Метод
 наименьших квадратов 138
 скалярных произведений 46
 стрельбы 335

Н

Нелинейные уравнения 167
 алгебраические 167, 183
 отделение корней 168
 случай комплексных корней 169
 трансцендентные 167

Р

Разностная схема
 неявная двухслойная 358
 явная двухслойная 353
Решение нелинейных уравнений 232
 в Maple 206
 в Mathcad 194
 в MATLAB 215
 графический метод 168
 комбинированный метод 177
 метод итерации 180
 метод касательных 174
 метод парабол для алгебраических
 уравнений 191
 метод половинного деления 170
 метод спуска для алгебраических
 уравнений 188
 метод хорд 172
 случай комплексных корней 183

Решение систем линейных уравнений 13, 53, 82, 87, 94, 233
 базисное 14
 итерационные методы 15
 метод LU-разложения 29, 61, 83, 96, 101
 метод QR разложения 97
 метод QR-разложения 31, 75, 83
 метод Гаусса 15, 60, 82, 83, 95
 метод градиента 43, 105
 метод Зейделя 38, 75
 метод итерации 36, 102
 метод квадратных корней 25, 98
 метод обратной матрицы 59, 82
 метод ортогонализации 32, 73
 метод прогонки 34, 75
 метод релаксации 41, 103
 метод скорейшего спуска 43
 метод Халецкого 95
 нетривиальное 13
 общее 14, 66
 правило Крамера 58
 схема Халецкого 27, 99
 точные методы 15
 тривиальное 13
 частное 14, 66

Решение систем нелинейных уравнений 229, 231, 232, 239, 240, 242, 243, 244
 метод градиента 224, 244
 метод итерации 222, 237
 метод Ньютона 219, 234
 модифицированный метод Ньютона 223, 237

С

Система
 дифференциальных уравнений 290
 жесткая 315

линейных дифференциальных уравнений 290
 Система линейных уравнений 13
 исследование на
 совместность 14, 64
 неоднородная 13
 неопределенная 14
 несовместная 14
 однородная 13
 определенная 14
 расширенная матрица системы 13
 совместная 14
 Собственное значение 45, 67, 85, 93
 метод итерации 48
 метод скалярных
 произведений 77, 105
 обобщенная задача 45, 71
 частичная проблема 46
 Собственный вектор 45, 67, 85, 93
 приведение матрицы
 к диагональному виду 69
 Схема Горнера 185

Ф

Функции Maple
 augment 78
 cholesky 82
 col 80
 coldim 80
 crossprod 80
 delcols 78
 delrows 79
 det 80
 diag 80
 diff 285
 dotprod 80
 dsolve 337
 entermatrix 79
 fsolve 238

Int 281
 int 281
 interp 134
 inverse 82
 LeastSquares 160
 linsolve 82
 matrix 78
 mulcol 80
 norm 81
 rank 81
 row 80
 rowdim 80
 rref 82
 scalarmul 80
 solve 238
 spline 136
 submatrix 79
 subs 285
 trace 81
 transpose 80
 vectdim 80
 vector 80

Функции Mathcad

Bulstoer 320, 325
 bulstoer 320
 cols 52
 cond1 52
 csort 53
 cspline 125
 diag 51
 eigenvals 53
 eigenvect 53
 eigenvects 53
 expfit 154
 Find 228
 genvals 53
 genvecs 53
 identity 51
 Im 51

intercept 148
 interp 125, 150
 last 52
 length 52
 lgsfit 154
 line 148
 linfit 153
 Infit 154
 logfit 154
 lsolve 53
 lspline 125
 lu 53
 matrix 50
 max 52
 mean 52
 median 52
 min 52
 Minerr 228
 multigrid 376
 norm1 52
 numol 371
 Odesolve 321
 odesolve 317
 pdesolve 369
 pspline 125
 pwrfit 154
 qr 53
 Radau 328
 rank 52
 Re 51
 relax 376
 reverse 53
 Rkadapt 319, 322
 rkadapt 320, 325
 rkfixed 318, 322
 rows 52
 rref 53
 rsort 53
 sbval 335