

**А. М. Половко
П. Н. Бутусов**

МАТЛАВ

ДЛЯ СТУДЕНТА

Санкт-Петербург

«БХВ-Петербург»

2005

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73
П52

Половко А. М., Бутусов П. Н.

П52 MATLAB для студента. — СПб.: БХВ-Петербург,
2005. — 320 с.: ил.

ISBN 5-94157-595-5

Содержится описание компьютерных технологий решения математических задач с помощью системы MATLAB. Приводятся примеры на все методы, изложенные в книге. Представлены варианты задач для индивидуального обучения. Описаны методики решения задач управления и создания приложений для решения типовых задач.

*Для студентов, аспирантов, преподавателей технических вузов
и специалистов, применяющих математические вычисления
в профессиональной деятельности*

УДК 681.3.06(075.8)
ББК 32.973.26-018.2я73

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Людмила Еремеевская</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн обложки	<i>Игоря Цырульников</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 12.07.05.

Формат 60×90^{1/16}. Печать офсетная. Усл. печ. л. 20.

Тираж 4000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-595-5

© Половко А. М., Бутусов П. Н., 2005
© Оформление, издательство "БХВ-Петербург", 2005

Оглавление

Введение	11
Глава 1. Основы интерфейса MATLAB.....	15
1.1. Окна системы MATLAB.....	16
1.1.1. Окно <i>Command Window</i>	16
1.1.2. Окно <i>Workspace</i>	18
1.1.3. Окно <i>Current Directory</i>	19
1.1.4. Окно <i>Command History</i>	20
1.1.5. Окно <i>Launch Pad</i>	20
1.2. Главное меню системы	21
1.2.1. Меню <i>File</i>	21
1.2.2. Меню <i>Edit</i>	23
1.2.3. Меню <i>View</i>	24
1.2.4. Меню <i>Web</i>	24
1.2.5. Меню <i>Window</i>	25
1.2.6. Меню <i>Help</i>	25
1.3. Панель инструментов.....	25
Глава 2. Язык общения с MATLAB.....	27
2.1. Символы и операторы языка	27
2.1.1. Специальные символы	27
2.1.2. Операторы отношения	29
2.1.3. Логические операторы	31
2.2. Числа, переменные, функции языка	32
2.2.1. Числа в MATLAB.....	32
2.2.2. Переменные и константы	34

2.3. Функции и команды общения	35
2.3.1. Команды управления окном.....	36
2.3.2. Сообщение об ошибках и их исправление.....	36
2.3.3. Сохранение результатов вычислений.....	37
2.3.4. Завершение работы.....	38

Глава 3. Создание приложений для решения типовых задач..... 39

3.1. Постановка задачи.....	39
3.2. Знакомство с инструментом.....	41
3.2.1. Меню и панель инструментов.....	43
Меню <i>Layout</i>	43
Меню <i>Tools</i>	44
3.2.2. Панель элементов управления	44
3.3. Файлы, генерируемые системой в процессе создания приложения	45
3.4. Работа над приложением	47
3.4.1. Первый этап.....	48
3.4.2. Второй этап	51
3.4.3. Третий этап.....	54

Глава 4. Специальные вычисления 61

4.1. Табулирование функции.....	61
4.2. Вычисление суммы элементов массива чисел.....	64
4.3. Вычисление произведения элементов чисел	65
4.4. Вычисление производных	66
4.5. Вычисление пределов	69
4.6. Разложение функции в степенной ряд.....	72
4.7. Определение экстремумов функции.....	77
4.7.1. Функция <i>fmin</i> (' <i>fun</i> ', <i>x1</i> , <i>x2</i>)	80
4.7.2. Функция <i>fmin</i> (' <i>fun</i> ', <i>x1</i> , <i>x2</i> , <i>options</i>)	82
4.8. Интегральные преобразования.....	83
4.8.1. Преобразование Лапласа	83
Функция <i>Laplace</i> (<i>F</i>)	85
Функция <i>Laplace</i> (<i>F,s</i>).....	87
Функция <i>Laplace</i> (<i>F,w,s</i>).....	87
4.8.2. Решение дифференциальных уравнений с помощью преобразования Лапласа.....	88
4.8.3. Обратное преобразование Лапласа	90

Глава 5. Вычисление математических функций	93
5.1. Элементарные функции.....	93
5.1.1. Алгебраические и арифметические функции	94
5.1.2. Тригонометрические функции.....	98
5.1.3. Обратные тригонометрические функции.....	99
5.1.4. Гиперболические функции.....	100
5.1.5. Обратные гиперболические функции.....	101
5.1.6. Функции комплексного аргумента	102
5.2. Специальные математические функции.....	104
5.2.1. Гамма-функция.....	104
5.2.2. Бета-функция (Эйлеров интеграл первого рода).....	106
5.2.3. Функции ошибок.....	107
5.2.4. Интегральная показательная функция	109
5.2.5. Функции Эйри.....	109
5.2.6. Функции Лежандра	112
5.2.7. Функции Бесселя.....	113
5.3. Функции пользователя.....	114
 Глава 6. Алгебра векторов и матриц	 116
6.1. Создание векторов и матриц.....	116
6.2. Преобразование матриц.....	118
6.2.1. Вызов на экран и замена элементов матрицы	118
6.2.2. Изменение размера вектора или матрицы	119
6.2.3. Математические операции с векторами и матрицами	121
Определитель матрицы	121
Транспонирование матрицы	122
След матрицы.....	122
Обратная матрица	123
Единичная матрица.....	123
Образование матрицы с единичными элементами	124
Образование матрицы с нулевыми элементами.....	125
Вектор равноотстоящих точек.....	126
Перестановка элементов матрицы	126
Создание матрицы с заданной диагональю	127
Создание массивов со случайными элементами.....	129
Поворот матрицы.....	132
Выделение треугольных частей матрицы.....	133
Вычисление магического квадрата	134
6.3. Математические операции над векторами и матрицами	135
6.3.1. Примеры образования функций от векторов и матриц	138

Глава 7. Визуализация вычислений.....	140
7.1. Двухмерная графика	140
7.1.1. Функция <i>plot(x,y)</i>	141
7.1.2. Функция <i>plot(x,y,s)</i>	142
7.1.3. Функция <i>plot(x1,y1,s1, x2,y2,s2, ...,xn,yn,sn)</i>	146
7.1.4. Функции построения графиков в логарифмическом масштабе.....	150
7.1.5. Графики в полярной системе координат.....	151
7.1.6. Создание гистограмм.....	152
7.2. Трехмерная графика.....	153
Глава 8. Алгоритмы и технологии решения уравнений	156
8.1. Алгоритмы решения алгебраических и трансцендентных уравнений.....	156
8.1.1. Метод дихотомии (половинного деления).....	156
8.1.2. Метод хорд	157
8.1.3. Метод касательных	159
8.1.4. Комбинированный метод (метод хорд и касательных)	161
8.1.5. Метод итераций.....	162
8.2. Технология решения алгебраических и трансцендентных уравнений в среде MATLAB	166
8.2.1. Технология решения уравнений с помощью функции <i>solve()</i>	166
8.2.2. Технология определения вещественных корней уравнения с помощью функции <i>fzero()</i>	169
8.2.3. Технология определения корней многочлена с помощью функции <i>roots()</i>	174
8.2.4. Варианты алгебраических и трансцендентных уравнений для индивидуальных заданий или решений.....	175
8.3. Методы решения систем алгебраических уравнений	175
8.3.1. Решение систем линейных алгебраических уравнений.....	182
Выбор начальных приближений	186
Условия сходимости итерационного процесса	186
Признак окончания вычислений	188
8.3.2. Алгоритмы метода итерации	188
8.3.3. Сравнительная оценка точных и итерационных методов.....	190
8.4. Компьютерные технологии решения систем линейных алгебраических уравнений в среде MATLAB.....	190
8.4.1. Решение системы линейных уравнений с помощью определителей	191

8.4.2. Матричный метод решения систем линейных уравнений....	192
8.4.3. Решение систем линейных уравнений с помощью функции <i>solve()</i>	194
8.5. Компьютерные технологии решения систем нелинейных уравнений	196
Функция <i>cgs()</i>	198
8.6. Варианты уравнений для индивидуального решения	199
Задание 1. Решение систем линейных алгебраических уравнений.....	199
Задание 2. Решение систем нелинейных алгебраических уравнений.....	204

Глава 9. Решение дифференциальных уравнений..... 205

9.1. Формулировка задачи	205
9.2. Приближенные аналитические методы решения дифференциальных уравнений	206
9.2.1. Метод последовательного дифференцирования	206
9.2.2. Метод неопределенных коэффициентов.....	208
9.2.3. Метод последовательных приближений	209
9.3. Численные методы решения дифференциальных уравнений	210
9.3.1. Метод Эйлера	210
9.3.2. Усовершенствованные методы Эйлера.....	212
Метод Эйлера — Коши.....	212
Усовершенствованный метод Эйлера.....	212
Усовершенствованный метод Эйлера — Коши с итерационной обработкой результатов.....	213
9.3.3. Метод Рунге — Кутты	213
9.4. Компьютерные технологии решения дифференциальных уравнений.....	216

Глава 10. Алгоритмы и технологии вычисления интегралов 220

10.1. Методы и алгоритмы вычисления интегралов	220
10.1.1. Формулы прямоугольников	221
10.1.2. Формула трапеций	221
10.1.3. Формула парабол (Симпсона).....	222
10.2. Численные методы вычисления интеграла в системе MATLAB	223
10.2.1. Метод трапеций.....	223
Функция <i>cumtrapz(x,y)</i>	226
Функция <i>trapz(y)</i>	228
Функция <i>trapz(x,y)</i>	229

10.2.2. Численное интегрирование с помощью квадратурных формул	231
Метод парабол (Симпсона).....	231
Функция <i>quad('fun',a,b)</i>	232
Функция <i>quad('fun',a,b,tol)</i>	232
Функция <i>ablquad('fun',a,b,c,d)</i>	233
Функция <i>quad8('fun',a,b)</i>	234
10.3. Аналитические методы вычисления интеграла	236
10.3.1. Функция <i>int()</i> вычисления неопределенного и определенного интегралов	236
10.3.2. Вычисление кратных интегралов	239
10.3.3. Вычисление несобственных интегралов.....	240
10.4. Примеры вычисления интегралов.....	241

Глава 11. Методы и компьютерные технологии интерполяции..... 248

11.1. Элементы теории.....	248
11.1.1. Выбор вида функции интерполяции.....	250
Графоаналитический способ.....	250
Способ линеаризации нелинейных функций.....	254
Анализ табличных разностей.....	255
Использование специальных программ автоматизации интерполяции	256
11.1.2. Определение коэффициентов функции интерполяции	257
11.1.3. Проверка адекватности модели	257
11.2. Интерполяция точная в узлах. Универсальный метод.....	258
11.2.1. Интерполяция линейными функциями.....	258
11.2.2. Интерполяция нелинейными функциями.....	262
11.2.3. Сплайн-интерполяция	264
11.2.4. Интерполяция точная в узлах	265
11.3. Интерполяция, приближенная в узлах (аппроксимация).....	267
11.3.1. Функция <i>lsqcurvefit()</i>	267
11.3.2. Полиномиальная аппроксимация.....	270
11.3.3. Интерполяция кубическими полиномами	272

Глава 12. Компьютерные технологии решения задач управления 274

12.1. Задачи управления	274
12.2. Функции MATLAB для создания передаточных функций звеньев системы	275
12.2.1. Функция <i>tf()</i>	275
12.2.2. Функции <i>pole()</i> и <i>zero()</i>	277

12.2.3. Функции <i>roots()</i> и <i>poly()</i>	279
12.2.4. Функция <i>conv()</i>	280
12.2.5. Функция <i>polyval()</i>	281
12.3. Операции с передаточными функциями звеньев	283
12.3.1. Сложение передаточных функций	283
12.3.2. Функция <i>pzmap()</i>	284
12.3.3. Функция <i>series()</i>	286
12.3.4. Функция <i>parallel()</i>	287
12.3.5. Функция <i>feedback()</i>	288
12.3.6. Функция <i>minreal()</i>	292
12.4. Исследование переходных процессов в системах управления	293
12.4.1. Функция <i>step()</i>	295
12.5. Частотные характеристики системы	296
12.5.1. Амплитудно-фазовая характеристика системы	299
12.5.2. Диаграмма Никольса	300
12.6. Пример анализа динамики системы управления	302
12.6.1. Образование передаточной функции разомкнутой системы	303
12.6.2. Определение нулей и полюсов передаточной функции $G(S)$	303
12.6.3. Расположение нулей и полюсов на комплексной плоскости	304
12.6.4. Анализ устойчивости системы	304
12.6.5. Исследование качества переходного	304
12.6.6. Получение передаточной функции замкнутой системы ...	306
12.6.7. Определение нулей и полюсов передаточной функции замкнутой системы и расположение их на комплексной плоскости	306
12.6.8. Переходные процессы замкнутой системы с жесткой отрицательной обратной связью	308
12.6.9. Исследование устойчивости и качества переходных процессов системы управления при гибкой отрицательной обратной связи	308
12.7. Индивидуальные задания для исследования динамики систем управления	309
12.7.1. Задание 1	309
12.7.2. Задание 2	314
Постановка задачи	314
Варианты индивидуальных заданий и передаточных функций	315

Введение

Компьютерная алгебра — новое научное направление в информатике. Его появление тесно связано с созданием универсальных математических программных средств символьной математики, таких как Mathematica, Maple, Derive, Mathcad, MATLAB и др.

Каждая из этих систем является уникальной. В ней имеется свой язык общения, наборы математических функций, алгоритмы и методы решения математических задач.

Уникальность системы MATLAB определяется следующими ее особенностями:

- ◆ система ориентирована на матричные операции;
- ◆ наличие большого числа библиотечных функций, делающих ее одновременно специализированной математической системой, предназначенной для решения ряда научных и инженерных задач (анализ и синтез систем управления, теория нечетких множеств, планирование эксперимента и многих других задач);
- ◆ возможность диалога с другими математическими системами (Maple, Mathcad, MS Excel) расширяет возможности MATLAB, ликвидирует один из ее недостатков — слабую, по сравнению с другими системами, символьную математику.

В результате этих особенностей MATLAB — одна из наиболее мощных математических систем, пользующаяся большой популярностью пользователей.

За рубежом изданы десятки книг по системе MATLAB. В нашей стране их очень мало. При этом большинство из них недоступны широкому читателю, т. к. изданы малыми тиражами в основном внутривузовскими издательствами для своих внутренних потребностей.

Настоящая книга предназначена для широкого круга пользователей. Ее особенностями являются:

- ◆ подробное изложение компьютерных технологий решения математических задач, а не только перечисление функций MATLAB, предназначенных для решения математических задач;
- ◆ большое количество примеров: практически на каждый из математических методов;
- ◆ краткое изложение сущности математических методов решения задач в среде MATLAB;
- ◆ отсутствие в книге сведений о системе MATLAB, не имеющих прямого отношения к решению математических задач (история создания, инсталляция системы, специальная графика и т. п.);
- ◆ краткость и одновременно ясность и достаточная полнота изложения компьютерных технологий решения математических задач;
- ◆ ориентация на студента технического вуза.

Книгу целесообразно использовать при проведении упражнений на персональном компьютере, например, по вычислительной математике, информатике и многим другим предметам.

При проведении лабораторных работ по общетехническим и специальным дисциплинам, требующим элементов научных исследований, без универсальных программных систем не обойтись. Здесь будет полезна система MATLAB.

Систему MATLAB целесообразно использовать при обработке результатов лабораторных работ по любому из предметов, где они проводятся.

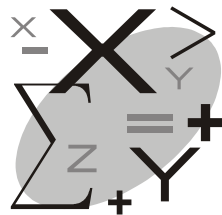
Книга просто необходима студенту при курсовом и дипломном проектировании. Она должна занять достойное место в библио-

теке студента уже с первого курса, где он проходит обучение по математике и информатике.

Не следует только думать, что если книга адресована студенту, то она ограничена по содержанию. Это далеко не так. Ее специфика лишь в методике изложения компьютерных технологий решения математических задач: большое число примеров, индивидуальных заданий, наличие сведений из теории решения математических задач, простота и краткость изложения.

По полноте излагаемых методов и особенно компьютерных технологий решения математических задач в среде MATLAB она не уступает другим, например, приведенным в списке литературы в конце книги.

Книга будет полезна преподавателям, аспирантам и соискателям, научным работникам и всем, кому по роду деятельности придется решать математические задачи.



ГЛАВА 1

Основы интерфейса MATLAB

Общение пользователя с системой MATLAB, как и с любой другой математической системой, происходит с помощью клавиатуры и мыши. Набирая соответствующие символы, являющиеся кодами операций, и щелкая мышью по кнопкам, пользователь вводит данные, обращается к функциям и командам, получает решение.

Обширный пользовательский интерфейс включает множество операций и требует знания (вернее запоминания) большого количества функций, команд и кнопок. К ним, в частности, относятся:

- ◆ главное меню системы;
- ◆ кнопки панели инструментов;
- ◆ окна системы;
- ◆ редактор файлов;
- ◆ общение с приложениями;
- ◆ графическая система;
- ◆ справочная система

и многое другое.

Все это можно изучить и активно им пользоваться только в процессе решения задачи. Изучать элементы интерфейса в полном объеме в начале работы с системой смысла не имеет.

В данной главе в конспективной форме описываются только изначально необходимые окна системы, главное меню и панель инструментов.

1.1. Окна системы MATLAB

Общение пользователя с системой происходит посредством пяти окон. Рассмотрим каждое из них.

1.1.1. Окно *Command Window*



Окно **Command Window** (Окно команд) является для пользователя наиболее важным. Посредством этого окна (рис. 1.1) вводятся математические выражения, получаются результаты вычислений, а также выдаются сообщения, посылаемые системой. Данное окно становится доступным пользователю сразу же после запуска программы. Математические выражения пишутся в командной строке после знака приглашения `>>`.

Наберем в строке ввода выражение

```
>> x=2+3
```

Для выполнения действия нажмем клавишу `<Enter>`. Результат виден на рис. 1.1.

Если мы захотим исправить одну из цифр, то у нас ничего не получится. Невозможность редактирования ранее введенной команды простой установкой курсора в нужную строку является одной из особенностей системы MATLAB. Для того чтобы отредактировать ранее введенную команду, необходимо установить курсор в строку ввода и воспользоваться клавишами `<↑>` и `<↓>`. Эти клавиши позволяют пролистать стек введенных ранее команд и оставить в строке ту команду, которая необходима. Команду можно выполнить сразу (нажав клавишу `<Enter>`) или после редактирования.

Окно команд имеет еще два управляющих элемента, в правом верхнем углу. Это кнопка закрытия окна , а расположенная левее — кнопка отделения окна от интерфейса системы . По-

сле нажатия на эту кнопку окно становится автономным, имеющим собственное меню. Возврат окна в общий интерфейс происходит посредством выбора из его главного меню пунктов **View | Dock Command Window** (Вид | Пристыковать окно команд). Заметим, что такими же кнопками снабжены оставшиеся четыре окна, поэтому о них (кнопках) в дальнейшем мы упоминать не будем.

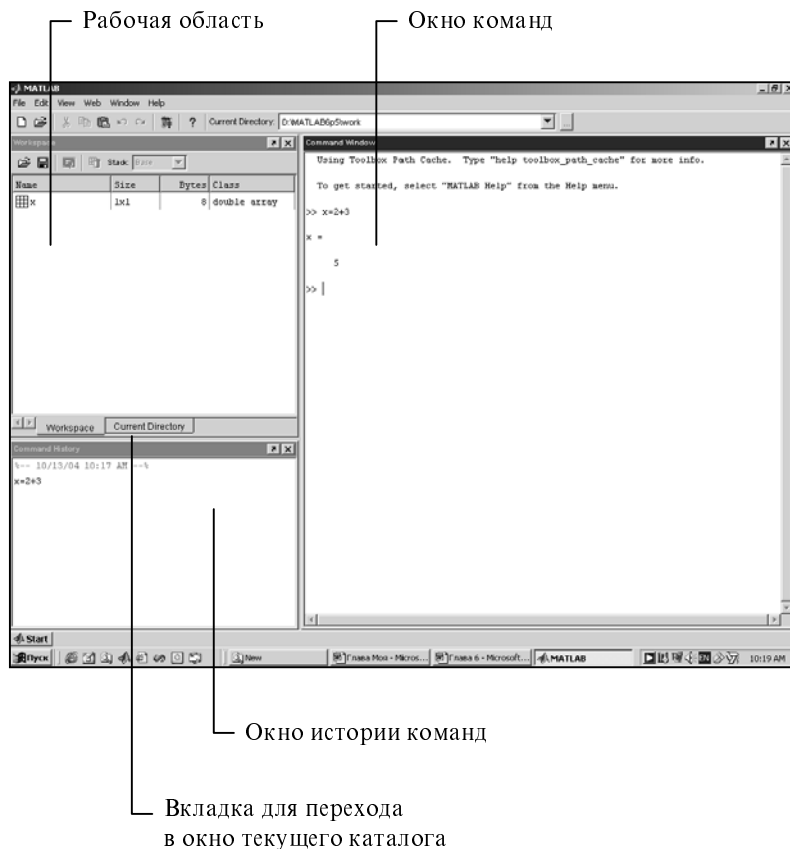


Рис. 1.1. Главное окно системы

Рассмотренное окно является основным и, в принципе, при работе с системой MATLAB можно было бы обойтись без других.

Однако наличие остальных окон делают диалог с компьютером легким и комфортным.

1.1.2. Окно *Workspace*

В процессе работы используются переменные различных типов. Созданные переменные хранятся в специально отведенной области памяти компьютера. Они не исчезают сами по себе, а только при выходе из программы или с помощью специальных команд.

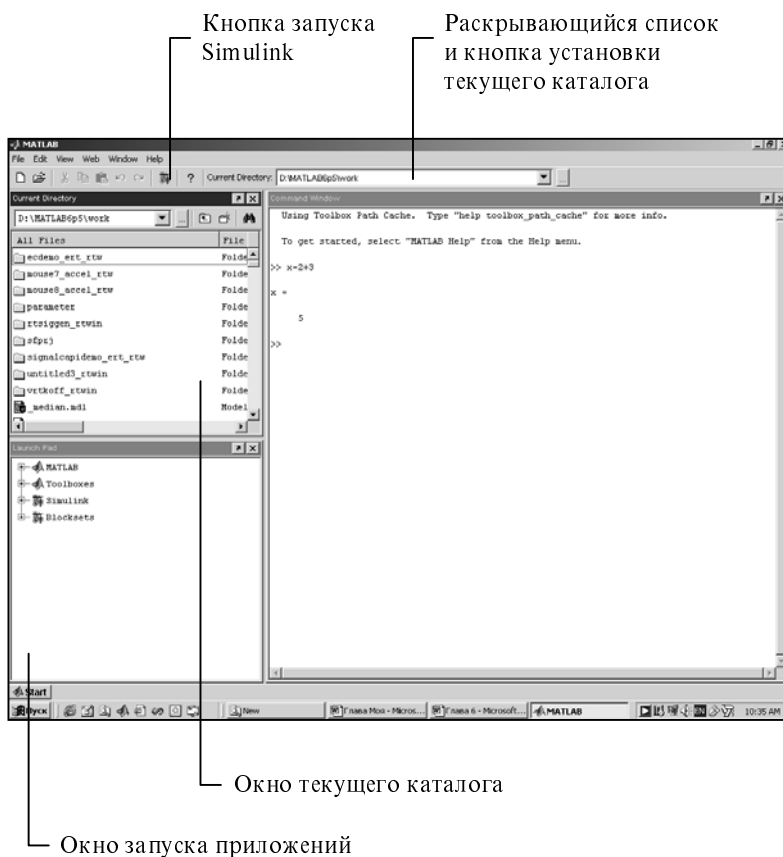


Рис. 1.2. Окно *Workspace*

При этом переменные (точнее их значения) можно использовать в любом вводимом нами математическом выражении. Окно **Workspace** (Рабочая область) предоставляет пользователю список всех переменных, хранящихся в рабочем пространстве (рис. 1.2). Выбрать можно любую переменную, просмотреть ее содержимое или выполнить какие-либо другие действия.

Упомянутые действия выполняются посредством контекстного меню (нужно щелкнуть правой кнопкой мыши по имени переменной в списке).

1.1.3. Окно *Current Directory*

Окно **Current Directory** (Текущий каталог) является аналогом известной программы Проводник, но имеет для MATLAB свое особое предназначение (рис. 1.3).

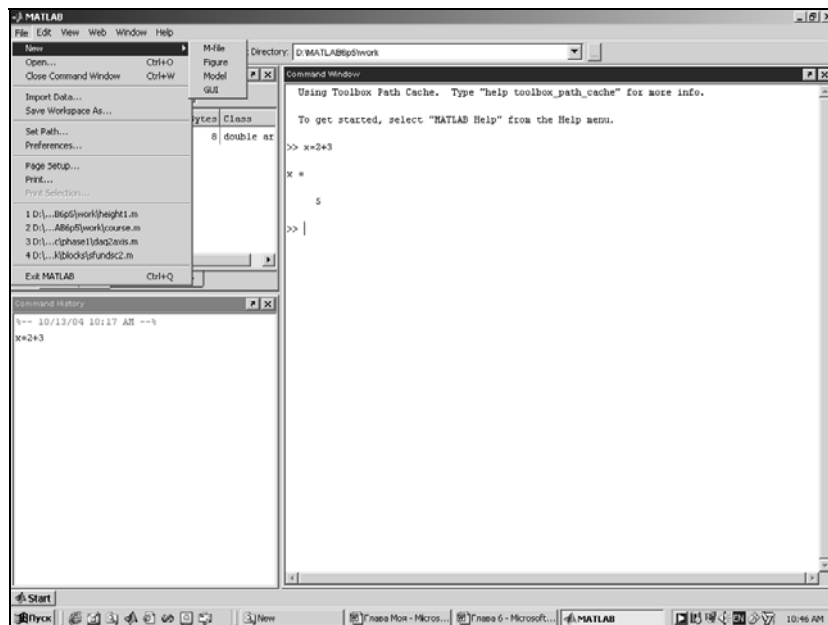


Рис. 1.3. Окно **Current Directory**

Дело в том, что, кроме работы с математическими выражениями из командного окна, пользователь также может работать с файлами (об этом речь пойдет далее). К тому же математические функции, которые мы используем, физически представляют собой файлы, названные по именам функций. В этих файлах записаны программы, реализующие функции. Таким образом, пользователь постоянно использует файлы. Например, указывая встроенную функцию, мы фактически пишем имя файла (без расширения), в котором хранится текст программы. И где в таком случае система должна искать требуемый файл? Она будет искать его в текущем каталоге или в пути доступа.

1.1.4. Окно *Command History*

Все команды, которые набираются в командной строке **Command Window** (Окно команд), автоматически образуют список, который и выводится в окне **Command History** (История команд). Чем полезен этот список? Если появилась необходимость повторить ранее выполненную команду, то она отыскивается в списке **Command History** (История команд), и, дважды щелкнув по ней левой кнопкой мыши, можно команду выполнить.

Можно выполнить нужную последовательность команд из командной строки и получить соответствующую последовательность команд в **Command History** (История команд). Несколькими простыми действиями полученная последовательность может быть преобразована в программу. Содержимое данного окна не теряется после выхода из системы и выключения компьютера. Удалить список команд можно только с помощью меню.

1.1.5. Окно *Launch Pad*

Окно **Launch Pad** (Панель запуска) содержит дерево файловой системы, где отображены только установленные на компьютере программные продукты, входящие в систему MATLAB. С помощью этого окна можно запустить любой из них. Те же действия легко осуществить посредством кнопки **Start** (Пуск), расположенной в левом нижнем углу окна MATLAB.

1.2. Главное меню системы

Посредством меню выполняются наиболее общие действия системы MATLAB. Меню имеет стандартный вид и организацию, присущие многим программным продуктам. Умение работать с меню может существенно облегчить диалог пользователя с компьютером. В этом разделе мы рассмотрим более подробно только те пункты меню, которые необходимы для решения поставленных в книге задач. При этом нужно иметь в виду, что глубоко изучить и активно использовать меню можно только в процессе решения задач.

Главное меню MATLAB содержит следующие шесть пунктов:

- ◆ **File** (Файл) — работа с файлами;
- ◆ **Edit** (Правка) — редактирование;
- ◆ **View** (Вид) — управление окнами;
- ◆ **Web** — связь с фирмой-разработчиком через Интернет;
- ◆ **Window** (Окно) — связь с окнами системы;
- ◆ **Help** (Справка) — связь со справочной системой MATLAB.

Рассмотрим главное меню системы, его команды и операции.

1.2.1. Меню *File*

Меню содержит большое число команд. Будем их называть в дальнейшем пунктами главного меню системы.

Пункт **New** (Создать) предоставляет возможность создать новый объект, а именно:

- ◆ **M-File** (М-файл) — файл с расширением *m*, в который записываются программы;
- ◆ **Figure** (Фигура) — специальное окно для вывода графической информации. При решении математических задач обычно используются функции, которые создают такие окна автоматически, поэтому данный пункт меню при визуализации вычислений используется редко;
- ◆ **Model** (Модель) — модель Simulink;

- ♦ **GUI** — графический интерфейс пользователя (Graphical User interface), используется для создания собственных приложений.

Пункт **Open** (Открыть) позволяет выполнить открытие существующего объекта посредством стандартного диалогового окна. Открытие объекта можно также осуществить из окна **Current Directory** (Текущий каталог).

Пункт **Close Current Directory** (Заккрыть текущий каталог) закрывает окно текущего каталога.

Пункт **Import Data** (Импортировать данные) производит импорт в среду MATLAB разнородных данных (анимационные ролики, звуковые файлы, числовые данные в различных форматах и т. д.).

Пункт **Save Workspace As** (Сохранить рабочую область как) выполняет сохранение рабочей области. Созданные и используемые переменные хранятся в специальной рабочей области (Workspace). При выходе из программы рабочая область автоматически уничтожается вместе со всеми переменными. Чтобы этого не происходило, необходимо сохранить рабочую область на диске в виде файла с расширением *mat*. Проще всего это сделать посредством пункта меню **Save Workspace As** (Сохранить рабочую область как), можно также использовать команду `save имя_файла.mat`. В дальнейшем сохраненная рабочая область загружается либо с помощью пункта меню **Import Data** (Импортировать данные), либо командой `load имя_файла`.

Пункт **Set Path** (Задать путь) организывает работу с путями доступа. При обращении к конкретному файлу (например, к функции) система MATLAB не требует указания пути. Это происходит потому, что MATLAB использует специальный список папок с путями доступа к ним. Когда вводится имя файла (с целью его поиска), система автоматически просматривает все известные ей папки, имеющиеся в списке. При инсталляции MATLAB список путей доступа строится автоматически. Для сохранения файлов во вновь созданной папке необходимо внести имя папки и путь доступа в общий список с помощью пункта меню **Set Path** (Задать путь). При вызове данного пункта появляется диалоговое окно с элементами управления (кнопками и списком).

Наиболее актуальными являются кнопки **Add Folder** (Добавить папку), **Add With Subfolders** (Добавить вместе с вложенными папками). Нажав одну из кнопок, можно указать путь к желаемой папке, после чего она будет добавлена к списку путей доступа. Кнопка **Save** сохраняет сделанные изменения.

Пункт **Preferences** (Настройка) изменяет некоторые свойства рабочей среды системы MATLAB.

Следующие три пункта меню: **Page Setup** (Параметры страницы), **Print** (Печать) и **Print Selection** (Печать выделенной области) служат для вывода информации на принтер, являются стандартными для многих пакетов и объяснений не требуют.

Меню **File** (Файл) имеет список последних открывавшихся файлов. Он содержит имена файлов с путями доступа и позволяет загрузить эти файлы двойным щелчком мыши.

Пункт **Exit MATLAB** (Выход) позволяет завершить работу с программой.

1.2.2. Меню *Edit*

Пункты **Undo** (Отменить), **Redo** (Повторить), **Cut** (Вырезать), **Copy** (Копировать), **Paste** (Вставить), **Select All** (Выделить все) и **Find** (Найти) полностью соответствуют своему стандартному предназначению и в комментариях не нуждаются. Первые пять из перечисленных пунктов меню продублированы кнопками на панели инструментов.

Пункт **Paste Special** (Специальная вставка) используется для обмена с внешними программами (например, MS Excel), числовыми данными посредством буфера обмена.

Пункт **Clear Command Window** (Очистить окно команд) очищает командное окно.

Пункт **Clear Command History** (Очистить историю команд) очищает окно предыстории.

Пункт **Clear Workspace** (Очистить рабочую область) очищает рабочую область от хранящихся в ней переменных.

1.2.3. Меню *View*

Среда системы MATLAB располагает несколькими рабочими окнами. Поэтому необходимо иметь инструмент, позволяющий управлять появлением, расположением и содержимым этих окон. Таким инструментом является меню **View** (Вид).

Пункт **Desktop Layout** (Разметка рабочего стола) помогает задать количество и расположение окон путем исполнения пунктов подменю.

Пункт **Undock** (Отстыковать) позволяет сделать автономным (отделить окно от интерфейса системы) выделенное в данный момент (активное) окно. После выбора данного пункта надпись меняется на **Dock** (Пристыковать) с названием активного окна. Меняется также на противоположную и функция пункта меню. Теперь при его выборе автономное окно снова прикрепляется к общему окну системы.

Следующая группа пунктов меню с названиями окон является группой переключателей. Каждый из этих пунктов может сделать видимым или невидимым соответствующее окно.

Пункт **Current Directory Filter** (Фильтр текущего каталога) имеет подменю пунктов-переключателей. С помощью этих переключателей можно выводить в окно **Current Directory** (Текущий каталог) определенные типы файлов.

Пункт **Workspace View Options** (Параметры отображения рабочей области) позволяет менять состав информации о переменных в списке окна **Workspace** (Рабочая область). Здесь можно также отсортировать список переменных по различным критериям.

1.2.4. Меню *Web*

Меню **Web** позволяет, при наличии подключения к Интернету, заходить в различные разделы сайта фирмы-изготовителя программного продукта. Здесь можно ознакомиться с новой информацией о системе, скачать ее, получить консультацию по интересующим вопросам. Для осуществления последних двух возможностей необходимо быть зарегистрированным пользователем.

Кроме того, выбрав пункт **MATLAB File Exchange** (Расширения), можно иметь свободный доступ к огромной библиотеке открытых примеров использования MATLAB для решения большого спектра задач.

1.2.5. Меню *Window*

Отображает список всех открытых дополнительных окон и служит для оперативного перехода к нужному окну. Это могут быть графические окна или окна, содержащие текст программы (рабочие окна в этом списке не отображаются). Пункт меню **Close All** (Заккрыть все) позволяет закрыть все окна, кроме рабочих.

1.2.6. Меню *Help*

Содержит большое количество полезной информации о MATLAB на английском языке.

1.3. Панель инструментов

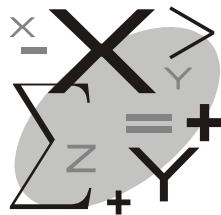
Кнопки панели инструментов обеспечивают выполнение большинства необходимых команд решения математических задач. Всплывающие подсказки при обращении к кнопкам сообщают об их содержании.

Кнопки панели инструментов имеют следующие назначения:

- ◆ **New file** (Создать) — выводит окна редактора файлов;
- ◆ **Open file** (Открыть) — открывает окна загрузки файлов;
- ◆ **Cut** (Вырезать) — вырезает выделенный файл и помещает в буфер обмена;
- ◆ **Copy** (Копировать) — копирует выделенный файл в буфер обмена;
- ◆ **Paste** (Вставить) — переносит фрагмент из буфера обмена в строку ввода;
- ◆ **Undo** (Отменить) — отменяет результат предыдущей операции;

- ◆ **Workspace Browser** (Просмотр рабочей области) — выводит окно ресурсов рабочей области;
- ◆ **Path Browser** (Просмотр пути) — выводит окно файловой структуры;
- ◆ **New Simulink Model** (Создать модель Simulink) — создает модель Simulink;
- ◆ **Help Window** (Справка) — открывает окна справки.

ГЛАВА 2



Язык общения с MATLAB

2.1. Символы и операторы языка

Операторы языка — это символы операций над данными, называемыми *операндами*. В MATLAB применяются все общепринятые операнды. Однако некоторые из них имеют ряд особенностей. Следует всегда иметь в виду, что большинство операторов языка MATLAB относится к матричным операциям. Например, операторы `*` и `/` вычисляют произведение и частное от деления двух массивов векторов и матриц. Если же необходимо вычислить почленное умножение или деление массивов, то следует применять операторы `.*` и `./`. В MATLAB также используется оператор деления справа налево (`\` или `.\`).

Подробно эти и другие, часто используемые операторы языка с большим числом примеров, рассмотрены в *главе 6*.

Полный список операторов можно получить по команде `help ops`.

2.1.1. Специальные символы

Специальными являются следующие символы языка MATLAB:

- ◆ `()` — круглые скобки;
- ◆ `[]` — квадратные скобки;
- ◆ `{ }` — фигурные скобки;
- ◆ `.` — десятичная точка;

- ◆ ; — точка с запятой;
- ◆ : — двоеточие;
- ◆ , — разделитель (запятая);
- ◆ .. — родительский каталог;
- ◆ ... — продолжение строки;
- ◆ % — комментарий;
- ◆ ! — вызов команды операционной системы;
- ◆ = — присвоение;
- ◆ ' — кавычка.

Рассмотрим назначение специальных символов.

- ◆ : — оператор образования массива данных переменной; формирует из векторов и матриц подвекторы и подматрицы. Представляется в следующих формах:
 - $i:k$ — аналог вектора $[i, i+1, i+2, \dots, k]$, например, $1:5$ — $[1\ 2\ 3\ 4\ 5]$;
 - $i:j:k$ — аналог оператора $i:k$, существует при $j > 0$, $k > i$ или при $j < 0$, $i > k$;
 - $M(:, i)$ — выбирается i -й столбец из матрицы M ;
 - $M(i, :)$ — выбирается i -я строка из матрицы M ;
 - $M(:, :)$ — аналогичен $M(:)$;
 - $M(i:k)$ — аналогичен $M(i), M(i+1), M(i+2), \dots, M(k)$;
 - $M(:, i, k)$ — аналогичен $M(:, i), M(:, i+1), M(:, i+2), \dots, M(:, k)$;
 - $M(:)$ — представление массива M в виде столбца;
 - $M(:, :, k)$ — k -я строка трехмерного массива M .
- ◆ Оператор круглые скобки () используется для задания аргументов функции, порядка выполнения операций в математических выражениях, указания индексов элементов вектора или матрицы. Например: $\sin(x)$, $(x-1)/(x+1)$, $x(v)$, $x(1)$, $M(A, B)$, $M(:, i)$.

- ◆ Оператор квадратные скобки [] формирует векторы и матрицы, например: [1 2 3 4], [1,2,3; 3 5 2].
- ◆ Оператор фигурные скобки { } применяется для формирования массивов ячеек.
- ◆ Десятичная точка (.) служит для отделения целой части числа от дробной. Кроме того, она применяется как знак указания операций над элементами символьных переменных. Примеры: 3.2, .15, 2.*log(x)+x.^2-x./cos(x).
- ◆ Точка с запятой (;) применяется в конце операторов для подавления вывода информации на экран, а также внутри круглых скобок для отделения строк матрицы.
- ◆ Запятая (,) используется для разделения элементов вектора и матрицы.
- ◆ Знак процента (%) воспринимается программой как начало комментария.
- ◆ Символ равенства (=) является знаком присваивания имени математическому выражению: x=[1 2 3 4 5], x=cos(a), x=2.5.
- ◆ Одиночная кавычка (') применяется для указания того, что математическое выражение содержит символьные переменные, например: Y='x+exp(-a)+1=0'.
- ◆ Знак восклицания (!) указывает, что за ним следует команда операционной системы.
- ◆ Знак из двух точек (..) определяет, что осуществляется переход на один уровень вверх по дереву каталогов (родительский каталог).
- ◆ Знак три точки (и более) (...) — продолжение строки. Его используют при переносе текста на другую строку.

2.1.2. Операторы отношения

Операторы отношения служат для сравнения двух операндов. Если операнды одинаковы, то программа возвращает 1 (True), в противном случае — 0 (False).

Правила записи операндов приведены в табл. 2.1.

Таблица 2.1. Операторы отношения

Функция	Имя оператора	Обозначение	Примеры
eq	Равно	==	a=b
ne	Не равно	~=	a~=b
lt	Меньше	<	x<y
gt	Больше	>	x>y
le	Меньше или равно	<=	x<=y
ge	Больше или равно	>=	x>=y

Операторы = и ~= сравнивают действительные и комплексные переменные. При этом сравниваются действительные и комплексные части числа.

Операторы <, <=, >, >= при сравнении комплексных чисел сравнивают только действительные части числа.

Примеры представлены в табл. 2.2.

Таблица 2.2. Примеры использования операторов отношения

Выражение	Функция	Результат
>> 5==5	>> eq(5, 5)	ans = 1
>> 3~=3	>> ne(3, 3)	ans = 0
>> 2+3i==2+i	>> eq(2+3i, 2+i)	ans = 0
>> 2+3i==2+3i	>> eq(2+3i, 2+3i)	ans = 1
>> 2+3i~=2+3i	>> ne(2+3i, 2+3i)	ans = 0

Таблица 2.2 (окончание)

Выражение	Функция	Результат
>> 3.2<3.21	>> lt(3.2<3.21)	ans = 1
>> 2.3+8i<2.4+i	>> lt(2.3+8i, 2.4+i)	ans = 1
>> 3.8-3i>5+i	>> gt(3.8-3i, 5+i)	ans = 0
>> 3<2.999	>> le(3, 2.999)	ans = 0
>> 3>=2.999	>> ge(3, 2.999)	ans = 1

2.1.3. Логические операторы

Логические операторы предназначены для реализации логических операций: дизъюнкции (ИЛИ), конъюнкции (И), инверсии (НЕ), исключающего ИЛИ.

Правила записи операторов приведены в табл. 2.3.

Таблица 2.3. Логические функции и операторы

Функция	Имя	Оператор
and	И	&
or	ИЛИ	
not	НЕ	~
xor	ИЛИ НЕ	-

Пример 2.1

```
>> 3 | 'x'  
ans =  
1
```

```

>> 'x' | ~'x'
ans =
    1

>> 'x' & ~'x'
ans =
    0

>> x1=[1,2,a,4];
>> x2=[1,0,0,1];
>> and(x1,x2)           или           >> x1&x2
ans =
    1    0    0    1

>> or(x1,x2)           или           >> x1|x2
ans =
    1    1    1    1

>> not(x1)             или           >> ~x1
ans =
    0    0    0    0

>> not(x2)             или           >> ~x2
ans =
    0    1    1    0

>> xor(x1,x2)
ans =
    0    1    1    0

```

Из примеров видно, что программа числа и символы, отличные от нуля, воспринимает как 1.

2.2. Числа, переменные, функции языка

2.2.1. Числа в MATLAB

Числа в MATLAB могут быть положительными и отрицательными, целыми и дробными, действительными и комплексными. Они могут представляться с фиксированной и плавающей точкой, с мантиссой и порядком (в научной форме).

Особенности представления чисел в MATLAB:

- ◆ мнимая единица кодируется с помощью двух символов: i или j ;
- ◆ целая часть числа от дробной отделяется точкой;
- ◆ отделение порядка числа от мантиисы осуществляется символом e ;
- ◆ знак $+$ положительного числа не ставится, знаки $+$ и $-$ положительного и отрицательного чисел называются *унарными*.

Формат чисел определяет их вид на экране монитора. А все вычисления в MATLAB осуществляются в формате двойной точности. Ввод чисел выполняется в любом формате по желанию пользователя.

Устанавливается формат чисел с помощью следующих команд:

- ◆ `format short` — короткое представление (5 знаков числа);
- ◆ `format short e` — короткое представление в экспоненциальной форме (5 знаков мантиисы, 3 знака порядка);
- ◆ `format long` — длинное представление числа (15 знаков);
- ◆ `format long e` — длинное представление в экспоненциальной форме (15 знаков мантиисы, 3 знака порядка);
- ◆ `format hex` — шестнадцатеричный формат;
- ◆ `format bank` — представление в денежном выражении (2 знака после точки).

Пример 2.2

Необходимо представить число $5/7$ во всех форматах.

Решение:

- ◆ `format short` — 0.7143;
- ◆ `format short e` — 7.1429e-001;
- ◆ `format long` — 0.71428571428571;
- ◆ `format long e` — 7.142857342857143e-001;
- ◆ `format bank` — 0.71.

2.2.2. Переменные и константы

Переменные — это символы, используемые для обозначения некоторых хранимых данных. Переменная имеет имя, называемое *идентификатором*. Данные могут менять свои значения, а идентификатор остается прежним. В MATLAB количество символов идентификатора ограничено и равно 31.

Имя переменной начинается с буквы и может состоять из букв, цифр и некоторых (допустимых) символов.

В процессе решения задач переменные могут занимать много памяти компьютера. Для очистки памяти от переменных в MATLAB используется функция `clear`, которая имеет синтаксис:

- ◆ `clear` — уничтожение всех переменных;
- ◆ `clear x` — уничтожение переменной `x`;
- ◆ `clear a b c` — уничтожение семейства переменных.

Константа — это численное значение уникального имени, имеющего математический смысл. Наиболее часто в MATLAB используются следующие константы:

- ◆ `pi` — число π ;
- ◆ `inf` — машинная бесконечность;
- ◆ `ans` — имя переменной, хранящей результат вычисления и высвечиваемой на экране в следующем виде: `ans=`;
- ◆ `NaN` — нечисловой характер данных (Not a Number);
- ◆ `eps` — погрешность операций с числами с плавающей точкой (2^{-52});
- ◆ `realmin` — минимальное число с плавающей точкой (2^{-1022});
- ◆ `realmax` — максимальное число с плавающей точкой (2^{1022}).

Пример 2.3

```
>> pi/2
ans =
    1.5708
```

```
>> sin(1.2)/sin(0)
ans =
    inf
>> (2-2)/(1-1)
ans =
    NaN
>> eps
ans =
    2.2204e-016
>> realmin
ans =
    2.2251e-308
>> realmax
ans =
    1.7977e308
```

Математическое выражение, взятое в одиночные кавычки, не вычисляется. Оно воспринимается как сочетание символов. Такое выражение может быть преобразовано в числовое. Об этом мы узнаем в дальнейшем при решении задач.

2.3. Функции и команды общения

Понятие "функция" весьма обширно. В математике — это любое выражение, у которого имеется имя, например $y = 2x + \sin(x) - 1$. Математические функции могут быть *элементарными*, *специальными* и *функциями пользователя*. В компьютерной алгебре функцией часто называют процедуру решения стандартной задачи, например, `solve` — имя функции решения уравнений. Математические функции достаточно подробно описаны в *главе 5*. Функции компьютерной алгебры описываются в процессе решения математических задач. В следующих разделах будут представлены функции общения с компьютером, независимо от того, какая решается задача. Такими функциями являются команды управления окнами, редактирования, сохранения результатов решения задачи, завершения работы.

2.3.1. Команды управления окном

Для очистки экрана набирается команда и нажимается клавиша <Enter>. Такими командами в MATLAB являются:

- ◆ `clc` — очищает окно, оставляя лишь знак приглашения `>>` в верхнем левом углу экрана;
- ◆ `home` — аналогична команде `clc`;
- ◆ `echo on all` — вывод на экран текста m-файлов;
- ◆ `echo off all` — отключение вывода на экран текста m-файлов;
- ◆ `echo <имя_файла> on` — вывод на экран файла сценария;
- ◆ `echo <имя_файла> of` — выключение режима вывода на экран файла сценария;
- ◆ `echo <имя_файла>` — смена режима вывода на противоположный;
- ◆ `more on` — режим постраничного вывода;
- ◆ `more off` — отключение режима постраничного вывода.

2.3.2. Сообщение об ошибках и их исправление

При наличии ошибок в выражениях или командах MATLAB не только не выдает решение, но и указывает на наличие ошибки. Из текста иногда можно понять сущность ошибки, но чаще комментарии бывают настолько общими, что трудно установить место и содержание ошибки.

В MATLAB используются два вида информации об ошибке: предупреждение и сообщение о ней.

При предупреждении процесс вычислений не прекращается, но на экран выдается текст, предупреждающий о том, что ответ может быть ошибочным.

При сообщении об ошибке вычисления прекращаются.

В случае неопределенности результатов вычисления появляется сообщение `NaN`, что означает неопределенность, например, вида $0/0$ или ∞/∞ .

При делении числа на ноль появляется сообщение "Warning: Devide by Zero".

Следует при этом иметь в виду, что машинный ноль и бесконечность имеют значения 10^{-308} и 10^{308} соответственно.

Напомним, что устранение ошибки наиболее целесообразно не путем набора нового правильного выражения, а редактированием ошибочного.

В MATLAB применяется оригинальный способ редактирования посредством повторного исполнения команд. С помощью клавиш $\langle \uparrow \rangle$ и $\langle \downarrow \rangle$ можно, перелистывая строки ранее введенных выражений или команд, найти необходимое выражение и либо отредактировать его, либо выполнить повторно. Например, пусть необходимо вычислить выражение $xe^{-x} + x^2 - 1$ при значении x от 0 до 2 с шагом 0.2.

Программа будет иметь вид:

```
>> x=0:0.2:2;  
>> y=x*exp(-x)+x^2-1
```

Пусть теперь необходимо вычислить при тех же значениях аргумента следующую функцию: $xe^x - x^2 + 1$.

Для решения задачи достаточно, нажимая многократно клавишу $\langle \uparrow \rangle$, вызвать исходное выражение и отредактировать его. После нажатия клавиши $\langle \text{Enter} \rangle$ получим ответ.

Клавиши $\langle \rightarrow \rangle$ и $\langle \leftarrow \rangle$ перемещают курсор вправо и влево на один символ соответственно. Комбинация клавиш $\langle \text{Ctrl} \rangle + \langle \rightarrow \rangle$, $\langle \text{Ctrl} \rangle + \langle \leftarrow \rangle$ перемещает курсор на одно слово вправо и влево соответственно.

2.3.3. Сохранение результатов вычислений

В рабочей области памяти хранятся результаты решения задачи. Их сохранение осуществляется с помощью функции `save`, которая имеет вид:

- ◆ `save fname` — сохранение рабочей области всех переменных в файле с именем `fname.mat`;

- ◆ `save fname x` — сохранение переменной `x`;
- ◆ `save fname x, y, z` — сохранение переменных `x, y, z`.

С помощью функции `save` нельзя сохранить весь текст, который находится на экране после решения задачи. Запрет объясняется просто. При решении задачи экран, как правило, загружен ошибочными функциями, неверными данными, сообщениями об ошибках и т. п. Все это хранить нет надобности, поэтому функция этого и не делает.

Для того чтобы сохранить необходимые строки решения задачи, пользователь должен убрать с экрана все ненужное. Для этого он должен воспользоваться редактором и отладчиком и получить текст без синтаксических и прочих ошибок. Такой текст сохраняется в виде файла с расширением `m`.

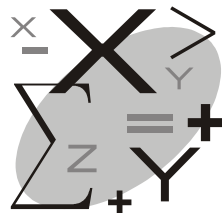
Впрочем, сохранить неотредактированный текст в полном объеме тоже можно. Для этого необходимо воспользоваться командой `diary`.

Сохраненный с помощью функции `save` текст можно вызвать для продолжения решения задачи. Для этого служит функция `load`.

2.3.4. Завершение работы

Завершение работы с системой MATLAB осуществляется с помощью команд `quit`, `exit` или нажатием комбинации клавиш `<Ctrl>+<0>`.

Все, что написано в этой главе, заучивать не нужно. Процедуры, команды и функции, приведенные здесь, запоминаются, а главное понимаются, только в процессе решения задачи. К этой главе читатель сможет обращаться в процессе решения задачи, как к справочному материалу.



ГЛАВА 3

Создание приложений для решения типовых задач

Данную главу при первом чтении целесообразно пропустить. К ней следует вернуться после изучения остальных глав книги, когда может потребоваться собственное приложение для решения пользовательских задач.

3.1. Постановка задачи

С помощью MATLAB можно не только работать с окном **Command Window** (Окно команд) и создавать собственные функции, но и разрабатывать свои приложения и даже делать их независимыми. Такие независимые приложения не требуют установки системы MATLAB, что очень удобно. В данной главе излагаются способы и описываются инструменты, позволяющие достаточно просто и быстро разрабатывать программы с пользовательским интерфейсом, предназначенным для решения определенного класса задач. На протяжении всей главы в качестве примера будем создавать приложение, позволяющее получить график аналитического выражения в заданном интервале, найти и обозначить точку пересечения с осью абсцисс, найти локальный минимум в заданном интервале. Описанная задача содержит следующие основные шаги:

- ◆ ввод аналитического выражения;
- ◆ ввод значений границ интервала;

- ◆ построение графика выражения;
- ◆ визуальный анализ графика;
- ◆ автоматический расчет и обозначение на графике искомых точек.

Задача намеренно упрощена для большего акцента именно на инструментальных средствах.

Технология решения любой задачи тесно связана с конструкцией интерфейса компьютерной программы, предназначенной для решения задачи.

Опишем желаемый вид интерфейса более подробно.

1. Для решения поставленной задачи прежде всего необходимо обеспечить ввод аналитического выражения и значений границ интервала. Подразумевается, что выражение будет представлять собой функцию одной переменной. Для указанных целей предусмотрим поля ввода: одно для выражения, другое — для значения границ.

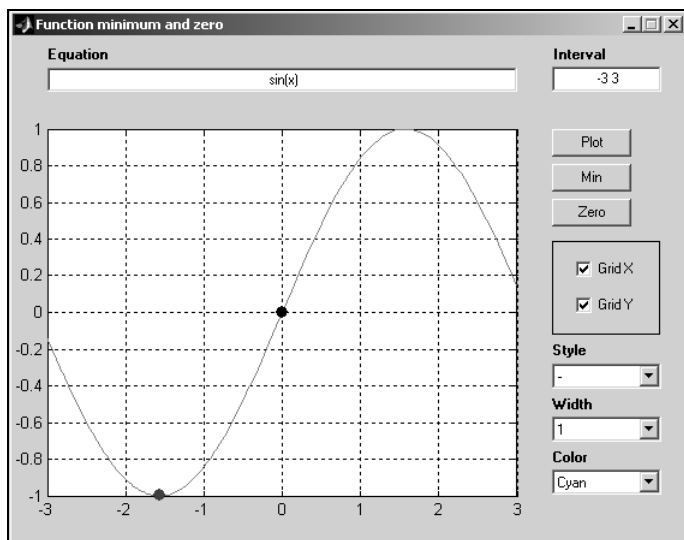


Рис. 3.1. Элементы интерфейса

2. Построение графика, расчет и визуализацию точки локального минимума, расчет и визуализацию точки пересечения с нулевой осью обеспечим тремя соответствующими кнопками.
3. Предусмотрим графическое окно для визуализации данных и аналитических выражений.
4. Предусмотрим управляющие элементы вспомогательного характера, такие как элемент управления координатной сеткой, изменения некоторых характеристик графика (стиля, толщины линий, цвета).

В результате элементы интерфейса будут иметь вид, показанный на рис. 3.1.

3.2. Знакомство с инструментом

Для того чтобы приступить к работе по созданию приложения, выберем пункт меню **File | New | GUI** (Файл | Создать | GUI¹). Появится диалоговое окно, изображенное на рис. 3.2.

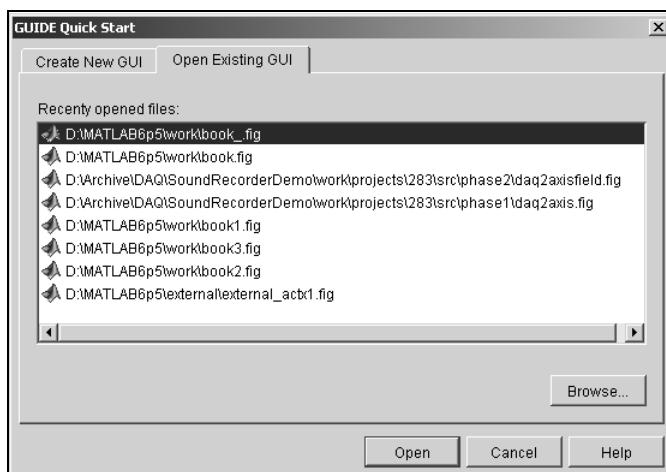


Рис. 3.2. Стартовое диалоговое окно

¹ Graphical User Interface — графический интерфейс пользователя.

Окно имеет две вкладки: **Create New GUI** (Создать GUI) и **Open Existing GUI** (Открыть GUI). Поскольку мы только приступаем к работе, нас интересует первая вкладка, а именно пункт **Blank GUI (Default)** (Новый GUI (по умолчанию)) в списке **GUIDE Templates** (Шаблоны GUI). После выбора указанных опций нажмем кнопку **OK**. Откроется программа GUIDE (рис. 3.3). Именно этот инструмент и предназначен для визуального конструирования интерфейса приложения, а также для программирования функций, размещаемых на нем элементов.

Познакомимся с элементами управления самой среды GUIDE.

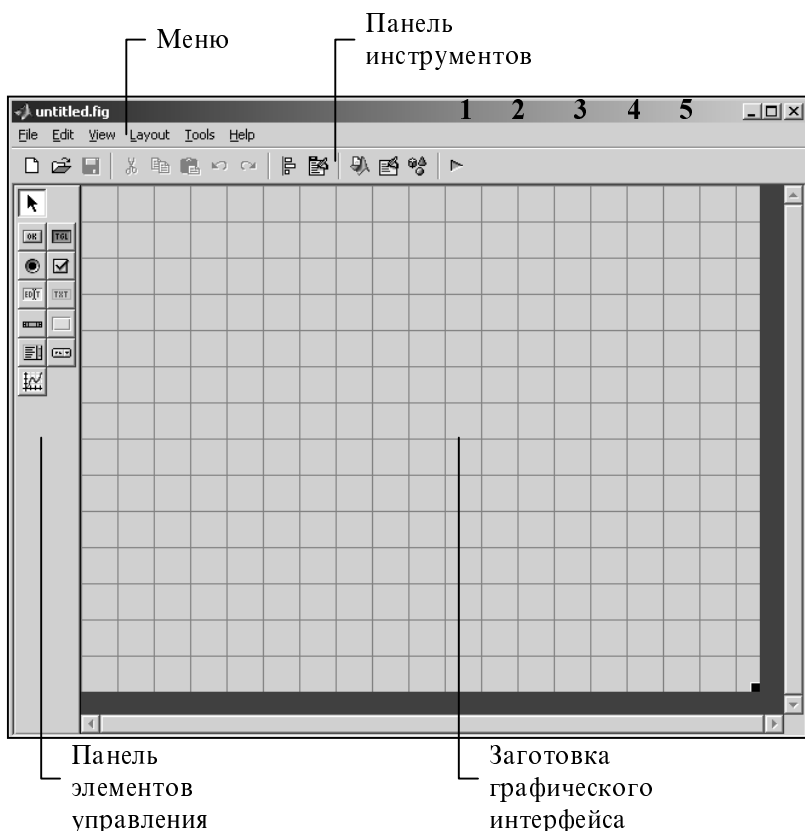




Рис. 3.3. Окно среды GUIDE


3.2.1. Меню и панель инструментов

Меню имеет стандартное назначение. Наиболее используемые его пункты продублированы кнопками на панели инструментов. Многие пункты меню имеют те же названия и назначения, что и в меню системы MATLAB, которое описано в *главе 1*. Рассмотрим только те пункты, которые нам еще неизвестны.

Пункт **Edit | Duplicate** (Правка | Дублировать) позволяет быстро получить копию выделенного элемента управления и разместить его в области заготовки графического интерфейса.

Пункт **View | Property Inspector** (Вид | Инспектор свойств) выводит на экран список свойств выделенного элемента. Позволяет редактировать значения свойств. Продублирован на панели инструментов кнопкой .

Пункт **View | Object Browser** (Вид | Просмотр объектов) выводит на экран дерево управляющих элементов (объектов), расположенных на графическом интерфейсе. Двойной щелчок мышью по названию объекта позволяет вызвать список свойств этого объекта. Продублирован на панели инструментов кнопкой .


Пункт **View | M-file Editor** (Вид | Редактор m-файлов) позволяет просматривать и редактировать файл, в котором содержится текст программ, реализующих функцию элементов управления. Продублирован на панели инструментов кнопкой . При взаимодействии пользователя с элементами графического интерфейса происходят действия, которые реализуются запрограммированными функциями.


Меню *Layout*

Пункт **Layout | Snap to Grid** (Разметка | Привязать к сетке) наделает объекты свойством притягиваться к сетке, нанесенной на графический интерфейс.


Оставшиеся четыре пункта меню **Layout** (Разметка) переносят объекты на задний или передний план в случае, если эти объекты взаимно перекрываются.

Меню *Tools*

Пункт **Tools | Run** (Сервис | Запуск) запускает программу на выполнение. Позволяет оперативно проверить правильность функционирования создаваемой программы. Продублирован на панели инструментов кнопкой .

Пункт **Tools | Align Objects** (Сервис | Выворнять объекты) вызывает диалоговое окно, позволяющее выровнять взаимное расположение объектов. Продублирован на панели инструментов кнопкой .

Пункт **Tools | Grid and Rulers** (Сервис | Сетка и линейки) вызывает диалоговое окно, позволяющее показать или скрыть сетку, вертикальную и горизонтальную линейки. Позволяет установить размер ячейки сетки и включить/выключить режим притягивания объектов к сетке.



Пункт **Tools | Menu Editor** (Сервис | Редактор меню) позволяет посредством диалогового окна конструировать основное и контекстные меню. Продублирован на панели инструментов кнопкой .











Пункт **Tools | Tab Order Editor** (Сервис | Редактор перехода по <Tab>) вызывает диалоговое окно, позволяющее установить порядок активизации элементов интерфейса при помощи нажатия клавиши <Tab> в работающем приложении.

Пункт **Tools | GUI Options** (Сервис | Параметры GUI) позволяет задавать некоторые свойства приложения. Нас вполне устроят установки по умолчанию.

3.2.2. Панель элементов управления

Панель предназначена для выбора и размещения на поле графического интерфейса следующих элементов управления (см. рис. 3.3):

- ◆  — переход в режим выделения объектов, расположенных на графическом интерфейсе;
- ◆  — кнопка;


- ◆  — переключатель;
- ◆  — поле ввода текстовой информации;
- ◆  — полоса прокрутки;
- ◆  — список;
- ◆  — вывод информации в виде графиков и диаграмм;
- ◆  — кнопка-переключатель;
- ◆  — флажок;
- ◆  — метка (текстовое поле без возможности редактирования);
- ◆  — рамка;
- ◆  — раскрывающийся список.

3.3. Файлы, генерируемые системой в процессе создания приложения

В момент выбора опции **Blank GUI (Default)** (Новый GUI (по умолчанию)) в окне **GUIDE Templates** (Шаблоны GUI), появляющемся после выбора пункта меню **File | New | GUI** (Файл | Создать | GUI), система автоматически создает два файла, имеющие имя приложения с расширениями `fig` и `m`. Первый из них содержит информацию об элементах управления, размещенных в области графического интерфейса, второй — программы, которые будут реагировать на действия пользователя с элементами управления. Упомянутые файлы не только создаются автоматически, они также изменяются самой системой в процессе работы пользователя над приложением. Что необходимо знать об этих файлах? Если о `fig`-файле пользователю вполне достаточно знать, что такой файл просто существует, то с `m`-файлом необходимо разобраться подробнее.

В списке свойств каждого управляющего элемента, размещенного в области графического интерфейса, имеется свойство `Callback`.

В качестве значения этого свойства записывается имя функции (подпрограммы), которая выполняется при возникновении на управляющем элементе его основного события. Для кнопки таким событием является ее движение вверх при отпускании после щелчка мышью. Для полосы прокрутки — перемещение движка, и т. д. Это событие вызывает на исполнение функцию (подпрограмму), записанную в m-файле приложения. Каждый управляющий элемент может реагировать на несколько событий, но мы рассмотрим только основное — `Callback`.

В качестве упражнения сделаем следующее. Создадим управляющий элемент "кнопка". Для этого щелкнем мышью по кнопке  на панели элементов управления, а затем — в желаемом месте области графического интерфейса. В ней появится изображение собственно кнопки в обрамлении. Обрамление свидетельствует о том, что данный элемент выделен и позволяет производить с ним определенные действия: перемещать, изменять размеры и т. д. Мы произведем на кнопке двойной щелчок мышью, чем вызовем на экран список свойств данного элемента. Крайне важно сразу же задать значение свойства `Tag` элемента. В дальнейшем это очень упростит нам работу с данным элементом из других подпрограмм. Установим значение свойства `Tag` равным `myButton`. После этого заглянем в m-файл нашего приложения и найдем там следующие строки:

- ◆ `% --- Executes on button press in myButton — выполняется при нажатии на кнопку (если точнее, то при отпускании ее);`
- ◆ `function myButton_Callback(hObject, eventdata, handles);`
- ◆ `% hObject handle to myButton — описатель кнопки;`
- ◆ `% eventdata reserved — зарезервировано и не используется;`
- ◆ `% handles structure with handles — структура, содержащая описатели всех элементов.`

Самой важной здесь является строка с заголовком функции (подпрограммы):

```
function myButton_Callback(hObject, eventdata, handles)
```

Заголовок этой функции был записан системой автоматически в момент создания управляющего элемента "кнопка". Заголовок

сопровождается строками комментариев, начинающихся с символа %.

Название функции `myButton_Callback` говорит о том, что функция обслуживает событие `Callback` элемента `myButton`. Функция содержит три параметра, краткое описание которых дано в комментариях. Мы же обсудим их подробнее.

Функция представляет собой запись какого-либо алгоритма на языке программирования. Практически во всех функциях мы будем обращаться к элементам управления (объектам) и их свойствам. Изменяя соответствующие свойства различных объектов, можно их скрывать и показывать, изменять цвет, строить графики и многое другое. Для того чтобы все это проделывать, необходимо знать, как обратиться к объекту и какими он обладает свойствами. Информацию о свойствах объекта можно получить из списка свойств в GUI справочной системы MATLAB, а также из источников [8, 9]. Обратиться к объекту из тела функции можно двумя способами.

- ◆ **Способ первый.** Обращение к текущему объекту (объекту, событие `Callback` которого обрабатывается в данный момент). Обращение производится посредством аргумента `hObject`, который является описателем текущего объекта.
- ◆ **Способ второй.** Использование аргумента `handles`, который является структурой, содержащей описатели всех элементов управления (объектов).

Поясним вышесказанное в процессе построения нашего приложения.


3.4. Работа над приложением

Работу над приложением начнем с "чистого листа". Для этого закроем GUI, если он был открыт. Затем выполним действия по созданию заготовки нового интерфейса, как это было описано в *разд. 3.2*. Для облегчения понимания выполняемых действий предлагается разбить нашу задачу на несколько этапов:

- ◆ первый этап — построение графика аналитического выражения в заданном интервале;

- ◆ второй этап — решение задач поиска корня и локального минимума;
- ◆ третий этап — создание дополнительных элементов управления.

3.4.1. Первый этап

На этом этапе создадим две области ввода текста: область ввода выражения и область границ интервала, а также оси для вывода графика и кнопки для построения графика. Перед созданием указанных элементов рекомендуется открыть список свойств текущего (выделенного) элемента, для чего следует нажать кнопку  (**Property Inspector**). Далее создание каждого из элементов советуем осуществлять в такой последовательности:

1. Щелкнуть мышью на панели элементов управления по нужному элементу.
2. Щелкнуть мышью в нужном месте в области заготовки интерфейса (появится собственно элемент в обрамлении).
3. Более точно позиционировать элемент с помощью мыши или клавиш управления курсором (если это необходимо).
4. Потянуть мышью в нужную сторону за соответствующий маркер в углу обрамления элемента и изменить его размеры (если это необходимо).
5. Установить значение свойства `Tag` в списке свойств элемента. (Значение свойства `Tag` является идентификатором (уникальным именем) элемента. Система автоматически присваивает каждому элементу свое уникальное имя, но нам нужно, чтобы имя это было более осмысленным.)
6. Удалить присвоенное по умолчанию значение свойства `String` и установить новое (если это необходимо).

Теперь поочередно создадим две области ввода текста и области ввода оси графика. Для первой области ввода текста свойству `Tag` присвоим значение `edEquation`, для второй — `edInterval`, для осей — `axMy`, одновременно очистив значения свойства `String` каждого из элементов (рис. 3.4).

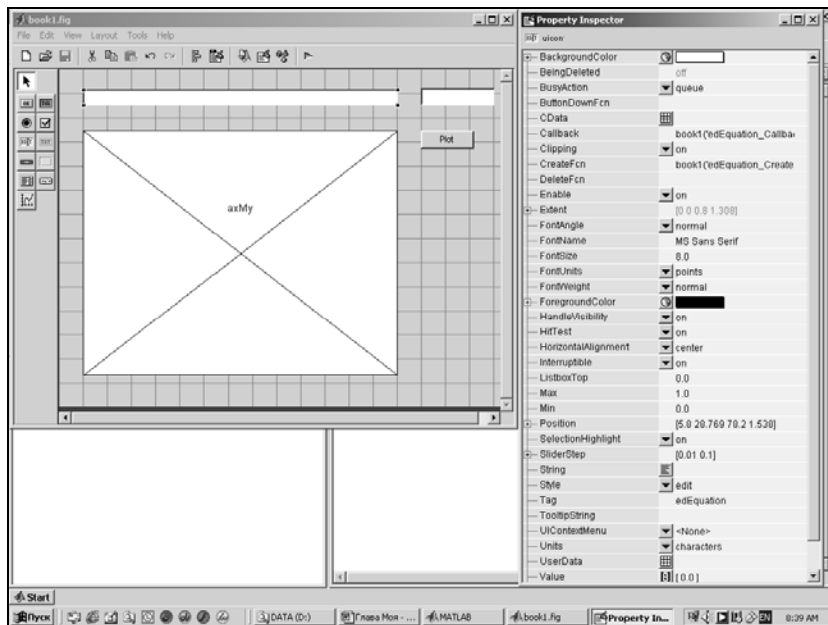




Рис. 3.4. Создание областей ввода

Работа над первыми тремя элементами закончена. Теперь создадим кнопку. Выполним все шаги рекомендуемой последовательности действий, присвоив свойству `Tag` кнопки значение `btnPlot`, а свойству `String` — значение `Plot`. Запустим приложение кнопкой  панели инструментов. Перед запуском система предложит присвоить имя приложению и сохранить его (по умолчанию в папке `...\Work`), что мы и сделаем. После запуска приложения проверим функциональность его элементов. Введем выражение (например, $\sin(x)$), границы $(-3, 3)$ и нажмем кнопку. Мы увидим, что элементы функционируют, но при этом ничего не происходит.

Для построения графика необходимо, чтобы при нажатии кнопки выполнялась определенная последовательность команд. Для записи такой последовательности надо вызвать на редактирование `m`-файл нашего приложения, нажав кнопку  панели инструментов. Тот же результат можно получить, щелкнув правой

кнопкой мыши по кнопке и выбрав в появившемся контекстном меню **View Callbacks | Callback** (Отобразить Callback-функции | Callback). В открывшемся содержимом m-файла нужно найти имя функции, которая обслуживает событие Callback, возникающее при нажатии кнопки `btnPlot`. Далее, под найденным именем функции запишем необходимую последовательность команд. В результате получим:

```
% --- Executes on button press in btnPlot.
function btnPlot_Callback(hObject, eventdata, handles);
% hObject      handle to btnPlot (see GCBO);
% eventdata    reserved — to be defined in a future version of
MATLAB;
% handles      structure with handles and user data (see GUIDATA
interval=str2num(get(handles.edInterval, 'String'));
f=inline(get(handles.edEq, 'String'));
fplot(f, interval);
```

Со структурой заголовка мы уже знакомы. Прокомментируем введенную последовательность команд.

```
get(handles.edInterval, 'String')
```

Функция `get()` позволяет считывать конкретное свойство (в нашем случае `String`) конкретного объекта (в нашем случае поле ввода интервала). Обратите внимание на способ обращения к объекту — через структуру `handles`, которая передается в функцию в качестве параметра. После `handles` через точку записывается идентификатор нужного объекта — `handles.edInterval`.

Функция `str2num()` преобразует строковое значение в числовое, полученное значение присваивается переменной `interval`.

В качестве аргумента функции `inline()` используется функция `get()`:


```
inline(get(handles.edEq, 'String'))
```

которая позволяет получить значение поля ввода аналитического выражения (собственно выражение). Функция `inline()` оперативно преобразует строку, содержащую выражение, в функцию. После такого преобразования к полученной функции можно обращаться так же, как и к любой встроенной. Например, можно

получить значение функции при заданном значении аргумента — $f(2)$, поскольку результат работы `inline()` присвоен переменной `f`.

Выражение `fplot(f, interval)` позволяет построить график функции `f` в заданном интервале (`interval`) в текущих осях.

Точки с запятой в конце каждой из строк позволяют подавить вывод результата в командное окно MATLAB.

Запустим наше приложение кнопкой  панели инструментов, введем выражение (например, $\sin(x)$) и границы интервала (например, $-3, 3$), разделенные пробелом. Получим результат, изображенный на рис. 3.1.

Основа приложения готова. Далее мы будем шаг за шагом вносить усовершенствования и узнавать новые возможности MATLAB. Переходим к следующему этапу разработки нашего приложения.

3.4.2. Второй этап

Начнем с создания кнопок **Min** и **Zero** (см. рис. 3.1). Как создаются кнопки, мы уже знаем. Напомним только, что обязательно надо присвоить значения свойствам `Tag` и `String` каждой из кнопок. Для первой из кнопок это будут, соответственно, значения `Min` и `btnMin`, для второй — `Zero` и `btnZero`. После того, как все это проделано, запустим наше приложение и убедимся, что кнопки присутствуют и нажимаются.

Теперь пришло время наделить наши кнопки необходимыми функциями. Займемся кнопкой **Min**. При нажатии этой кнопки наша программа должна рассчитать и нанести на построенный график точку локального минимума исследуемого аналитического выражения.

Войдем в режим редактирования `m`-файла приложения и найдем заголовок функции, обслуживающей событие `Callback` управляющего элемента `btnMin`. Под заголовком запишем последовательность команд. Результат будет выглядеть следующим образом:

```
% --- Executes on button press in btnMin.
function btnMin_Callback(hObject, eventdata, handles)
% hObject    handle to btnMin (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

interval=str2num(get(handles.edInterval,'String'));
x1=interval(1);
x2=interval(2);
f=inline(get(handles.edEq,'String'));
x=fminbnd(f,x1,x2);
y=f(x);
plot(x,y,'r.','MarkerSize',25);
```

Первая строка нам уже знакома.

```
interval=str2num(get(handles.edInterval,'String'));
```

С ее помощью мы получаем вектор-строку, содержащую численные значения границ интервалов.

Во второй и третьей строках мы обращаемся к элементам вектора-строки `interval` по индексам и присваиваем значения первого и второго элементов переменным `x1` и `x2` соответственно:

```
x1=interval(1);
x2=interval(2);
```

Выражение

```
f=inline(get(handles.edEq,'String'));
```

получает содержимое поля `edEquation` и преобразует его в функцию:

```
x=fminbnd(f,x1,x2)
```

Здесь мы используем встроенную функцию `fminbnd()`, которая осуществляет поиск локального минимума функции одной переменной на заданном интервале. Полученный результат присваиваем переменной `x`:

```
y=f(x);
```

Подставляем найденную абсциссу локального минимума в качестве аргумента функции `f()` и находим соответствующую ординату.

Описание функции `plot()`:

```
plot(x,y,'r.','MarkerSize',25)
```

Строим точку с заданными координатами x , y . Точка имеет форму круга красного цвета, что определено аргументом встроенной функции `plot()` — `'r.'` (r — от англ. *red*, $.$ — круг). Размер точки задан числовым значением свойства `MarkerSize`.

При нажатии кнопки `btnZero` программа должна рассчитать и нанести на построенный график точку локального минимума исследуемого аналитического выражения.

Программа обработки свойства `Callback` кнопки `btnZero` выглядит аналогично предыдущей:

```
% --- Executes on button press in btnZero
function btnZero_Callback(hObject, eventdata, handles)
% hObject      handle to btnZero (see GCBO)
% eventdata    reserved — to be defined in a future version of
MATLAB
% handles      structure with handles and user data (see
GUIDATA)
interval=str2num(get(handles.edInterval,'String'));
x1=interval(1);
x2=interval(2);
f=inline(get(handles.edEq,'String'));
x=fzero(f, (x1+x2)/2);
y=f(x);
plot(x,y,'g.','MarkerSize',25)
```

Программа должна рассчитать и нанести на построенный график точку пересечения с нулем (корень) исследуемого аналитического выражения.

Программа содержит строку

```
x=fzero(f, (x1+x2)/2);
```

со встроенной функцией `fzero()`. Функция используется для приближенного вычисления корня уравнения по заданному началь-

ному приближению (см. главу 8). В качестве параметров `fzero()` задаются функция, построенная на основе аналитического выражения, и середина заданного интервала.

После внесения описанных изменений в программу проверим ее работоспособность.

3.4.3. Третий этап

Создадим элементы управления, выводящие и убирающие координатную сетку и изменяющие стиль, толщину и цвет построенной кривой. Для управления появлением и удалением координатной сетки воспользуемся элементами управления `CheckBox` (т. е. флажками). Создадим два таких элемента (рис. 3.5) и присвоим каждому из них значения свойствам `Tag` и `String`. Для первого элемента `cbX` и `GridX` соответственно; для второго — `cbY` и `GridY` соответственно.

После внесения изменений в текст программы получим:

```
% --- Executes on button press in cbX.
function cbX_Callback(hObject, eventdata, handles)
% hObject    handle to cbX (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles     structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of cbX
if get(hObject,'Value')
    set(gca,'XGrid','on')
else
    set(gca,'XGrid','off')
end
% --- Executes on button press in cbY.
function cbY_Callback(hObject, eventdata, handles)
% hObject    handle to cbY (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles     structure with handles and user data (see
GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of cbY
```

```
if get(hObject,'Value')
    set(gca,'YGrid','on')
else
    set(gca,'YGrid','off')
end
```

Как видно, программы полностью идентичны. Используя конструкцию `if...else`, мы осуществляем выбор между выражениями `set(gca,'XGrid','on')` и `set(gca,'XGrid','off')`. Выбор происходит на основании результата, полученного выполнением команды `get(hObject,'Value')`. С помощью этой команды получаем значение свойства `Value` текущего элемента. Это свойство может иметь только два значения — 0 или 1. Если получена единица, то выполняется первая команда — `set(gca,'XGrid','on')` (выводится координатная сетка), если ноль — команда `set(gca,'XGrid','off')` и `else` (скрывается координатная сетка).

Рассмотрим подробнее конструкцию `set()`. Функция `set()` позволяет установить значение конкретного свойства указанного объекта. В данном случае мы обращаемся к функции `gca()` (осуществляет доступ к свойствам текущих осей) и присваиваем свойству осей `XGrid` (координатная сетка вдоль оси *x*) значение `'on'` (включено) или `'off'` (выключено).

Наиболее универсальным управляющим элементом автор считает раскрывающийся список (в системе MATLAB он называется `Popup Menu`). Данный элемент более удобен, чем простой список `List Box`, т. к. занимает меньше места в области разрабатываемого интерфейса приложения. Раскрывающийся список по своей функциональности может заменить элементы управления `Radio Button` (переключатель) и `Slider` (полоса прокрутки), что с успехом используют авторы демонстрационных примеров MATLAB. В нашем приложении присутствуют сразу три раскрывающихся списка — элементы изменения стиля, толщины и цвета.

Создание элемента начинается, как обычно, с его размещения и подгонки размеров известным нам способом. Затем задается свойство `Tag` элемента (в нашем случае — `pmStyle`, `pmWidth` и `pmColor`). Задание свойства `String` раскрывающегося списка имеет

свои особенности. При щелчке мышью на кнопке рядом со свойством `String` в окне свойств элемента возникает следующее окно (рис. 3.5), в котором необходимо ввести название каждого пункта списка. Название записывается одной строкой и отделяется от следующей нажатием клавиши `<Enter>`.

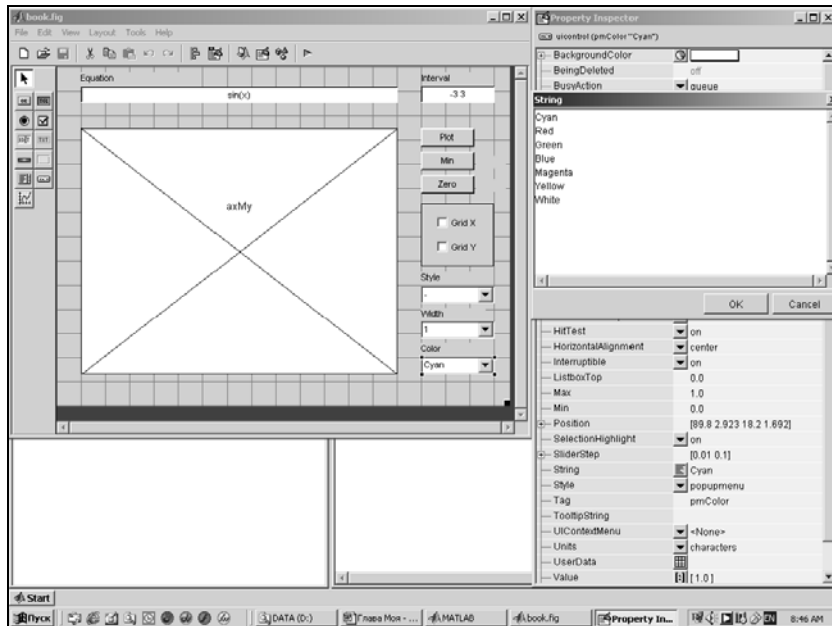


Рис. 3.5. Окно свойств элемента

После создания трех раскрывающихся списков внесем изменения в программу. Программы обработки события `Callback` каждого из списков идентичны. Они имеют вид:

```
% --- Executes on selection change in pmStyle.
function pmStyle_Callback(hObject, eventdata, handles)
% hObject    handle to pmStyle (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = get(hObject,'String') returns pmStyle
contents as cell array
% contents{get(hObject,'Value')} returns selected item from
pmStyle
Num=get(hObject,'Value');
switch Num
    case 1
        set(handles.line,'LineStyle','-');
    case 2
        set(handles.line,'LineStyle','--');
    case 3
        set(handles.line,'LineStyle',':');
    case 4
        set(handles.line,'LineStyle','-.'');
end
% --- Executes on selection change in pmWidth.
function pmWidth_Callback(hObject, eventdata, handles)
% hObject    handle to pmWidth (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns pmWidth
contents as cell array
% contents{get(hObject,'Value')} returns selected item from
pmWidth
Num=get(hObject,'Value');
switch Num
    case 1
        set(handles.line,'LineWidth',1);
    case 2
        set(handles.line,'LineWidth',2);
    case 3
        set(handles.line,'LineWidth',3);
    case 4
        set(handles.line,'LineWidth',4);
end
% --- Executes on selection change in pmColor.
function pmColor_Callback(hObject, eventdata, handles)
% hObject    handle to pmColor (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns pmColor
contents as cell array
% contents{get(hObject,'Value')} returns selected item from
pmColor
```

```
Num=get(hObject,'Value');
switch Num
    case 1
        set(handles.line,'Color','cyan');
    case 2
        set(handles.line,'Color','red');
    case 3
        set(handles.line,'Color','green');
    case 4
        set(handles.line,'Color','blue');
    case 5
        set(handles.line,'Color','magenta');
    case 6
        set(handles.line,'Color','yellow');
    case 7
        set(handles.line,'Color','white');
end
```

Выражение

```
Num=get(hObject,'Value');
```

получает значение свойства `Value` текущего элемента и присваивает его переменной `Num`. Значение свойства `Value` содержит номер строки списка, выбранной пользователем. Конструкция `switch...case` позволяет выполнить команду, которая соответствует номеру выбранной строки. А именно:

```
switch Num
    case 1
        set(handles.line,'LineStyle','-');
    case 2
        set(handles.line,'LineStyle','--');
    case 3
        set(handles.line,'LineStyle',':');
```

```
case 4
    set(handles.line,'LineStyle','-');
end
```

Действие команды заключается в присвоении свойствам `LineStyle`, `LineWidth` и `Color` нужных значений. К объекту, свойство которого изменяется, мы обращаемся посредством структуры `handles.line`, которую получили в качестве параметра функции обработки события `Callback`. Идентификатор `line` указывает на кривую графика. Тонкость заключается в том, что этот идентификатор-указатель мы должны создать, активизировать и внести в структуру `handles` самостоятельно на этапе построения графика. Для этого внесем изменения в текст программы функции обработки события `Callback` кнопки `btnPlot`.

```
% --- Executes on button press in btnPlot.
function btnPlot_Callback(hObject, eventdata, handles)
% hObject    handle to btnPlot (see GCBO)
% eventdata  reserved – to be defined in a future version of
MATLAB
% handles structure with handles and user data (see GUIDATA)
cla
interval=str2num(get(handles.edInterval,'String'));
f=inline(get(handles.edEq,'String'));
[x,y]=fplot(f,interval);
handles.line=plot(x,y,'c-');
guidata(gcbo,handles);
hold on
```

Поясним суть изменений.

```
[x,y]=fplot(f,interval);
```

Такая форма использования функции `fplot()` позволяет получить значения координат точек графика без вывода самого графика.

Выражение

```
handles.line=plot(x,y,'c-');
```


присваивает указатель на кривую, построенную с помощью функции `plot()` идентификатору `line` структуры `handles`. Дело в том, что функция `fplot()` на это не способна.

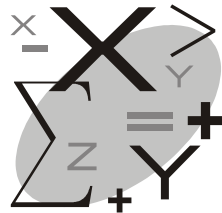
В строке

```
guidata(gcbo,handles);
```

обновленная структура `handles` сохраняется, что дает возможность использовать эти обновления в других функциях.

В строке `cla` очищаются текущие оси.

Наше приложение почти готово. Осталось только нанести некоторые поясняющие надписи, например, **Equation**. Для этого воспользуемся элементом `Static Text` (кнопка ). Создадим элемент обычным способом и присвоим его свойству `String` нужное текстовое значение, при этом можно довольствоваться значением свойства `Tag`, присвоенным по умолчанию. Заметим, что данный элемент имеет множество изменяемых свойств (например, `FontName`, `FontSize`), манипулируя которыми можно придать надписи нужный вид.



ГЛАВА 4

Специальные вычисления

В данной главе излагаются способы и компьютерные технологии вычисления сумм и произведений ряда чисел, представление функций в табличной форме, вычисление производных, пределов, особых точек, разложение функций в степенные ряды.

Заканчивается глава описанием преобразования Лапласа и применением его для решения линейных дифференциальных уравнений.

Способы иллюстрируются большим количеством примеров.

4.1. Табулирование функции

Математическая функция может быть представлена в виде формулы, таблицы, графика. Табличное представление функции необходимо в следующих случаях:

- ♦ определение погрешности интерполяции;
- ♦ вычисление табличных разностей с целью определения степени интерполяционного полинома;
- ♦ определение области изоляции корня;
- ♦ оценка численных значений функции в широком диапазоне аргументов.

В системе MATLAB табулирование функции осуществляется с помощью функции `subs()`, которая имеет вид:

```
subs(f, x, x1)
```

где:

- ◆ f — функция, заданная аналитически;
- ◆ x — аргумент функции f ;
- ◆ $x1$ — вектор значений аргумента x , при которых определяется значение функции f .

Переменная $x1$ может представляться в виде вектора или при постоянном шаге в виде: $x_n : \Delta x : x_k$ где x_n — начальное значение $x1$, Δx — шаг, x_k — конечное значение $x1$.

Технология табулирования функции $f(x)$:

1. Определение группы символьных переменных с помощью функции `syms`.
2. Образование вектора $x1$.
3. Ввод функции табулирования $y = f(x)$.
4. Образование функции табулирования `subs`.
5. Получение решения путем нажатия клавиши <Enter>.

Пример 4.1

Приведем примеры табулирования функции $y = e^x$ для $x1$ в диапазоне $[0; 1]$ с постоянным шагом $h = 0.2$ и в случае, когда $x1 = [0, 0.5, 1, 2, 5]$.

Программы решения задачи и ответы имеют вид:

```
>> syms x, x1, y;
>> x1 = 0 : 0.2 : 1;
>> y = exp(x);
>> subs(y, x, x1)
ans =
    1.0000    1.2214    1.4918    1.8221    2.2255    2.7183
>> x1 = [ 0, 0.5, 1, 2, 5 ];
>> y = exp(x);
>> subs(y, x, x1)
ans =
    1.0000    1.6487    2.7183    7.3891   148.4132
```

Функция `subs()` позволяет табулировать одновременно несколько функций. Для этого необходимо функцию y представить в виде матрицы табулируемых функций, например,

```
y = [exp(x); x.^2; sin (x)].
```

MATLAB позволяет табулировать функции, используя матричные операции и не обращаясь к функции `subs`. Технология вычислений в этом случае состоит в выполнении следующих операций:

1. Определение символьных переменных с помощью функции `syms`.
2. Образование вектора аргумента x .
3. Образование матрицы, элементами которой являются аргумент x и табулируемые функции.
4. Получение решения путем нажатия клавиши <Enter>.
5. При необходимости получение решения в столбик используется функция y' (транспонирование).

Покажем технологию на примере.

Пример 4.2

Пусть необходимо табулировать функции e^x , $\sin x$, $\cos x$ в диапазоне изменения x , равном $[0; 1]$, с шагом 0.2. Решение следует получить в виде матрицы.

Решение:

```
>> syms x, y;
>> x = 0 : 0.2 : 1;
>> y = [ x; exp(x); sin(x); cos(x) ]
```

$y =$

0	0.2000	0.4000	0.6000	0.8000	1.0000
1.0000	1.2214	1.4918	1.8221	1.2255	2.7183
0	0.1987	0.3984	0.5646	0.7174	0.8415
1.0000	0.9801	0.9211	0.8253	0.6967	0.5403

y'

0	1.0000	0	1.0000
0.2000	1.2214	0.1987	0.9801

0.4000	1.4918	0.3984	0.9211
0.6000	1.8221	0.5646	0.8253
0.8000	1.2255	0.7174	0.6967
1.0000	2.7183	0.8415	0.5403

4.2. Вычисление суммы элементов массива чисел

Вычисление суммы элементов массива чисел осуществляется в MATLAB с помощью функций `sum()` и `cumsum()`, которые имеют следующий синтаксис:

```
sum(x)
cumsum(x)
```

где x — вектор или матрица элементов суммирования.

Если x — вектор, то функция `sum(x)` выдает значение суммы элементов вектора. Если x — матрица, то откликом будет вектор, элементами которого являются суммы каждого столбца матрицы.

Функция `cumsum(x)` возвращает суммы элементов и все промежуточные результаты суммирования.

Приведем примеры суммирования на все указанные выше случаи.

Пример 4.3

```
>> x = 1 : 1000;
>> s = sum(x)
s =
    500500

>> x = [ 1, 4, 9, 16, 25 ] .^2;
>> s = sum(x)
s =
    979

>> x=[ 1, 2, 3, 4, ; 2, 3, 4, 5 ; 3, 4, 5, 6 ; 4, 5, 6, 7 ];
>> s = sum(x)
```

```
s =  
    10    14    18    22  
>> s = cumsum(x)  
s =  
    1     2     3     4  
    3     5     7     9  
    6     9    12    15  
   10    14    18    22
```

4.3. Вычисление произведения элементов чисел

Вычисление произведения элементов массива осуществляется с помощью функций `prod()` и `cumprod()`, имеющих следующий синтаксис:

```
prod(x)  
cumprod(x)
```

где x — вектор или матрица элементов.

Если x — вектор, то функция `prod(x)` вычисляет произведение элементов вектора. Если x — матрица, то откликом будет вектор, элементами которого являются произведения элементов каждого столбца матрицы.

Функция `cumprod(x)` дополнительно возвращает частичные произведения элементов вектора или столбцов матрицы.

Ниже приведены примеры вычисления произведения ряда чисел.

Пример 4.4

Необходимо вычислить:

- ◆ произведение чисел от 1 до 10;
- ◆ произведение элементов вектора $[1, 4, 9, 16, 25]$;
- ◆ произведение квадратов элементов вектора $[1, 4, 9, 16, 25]$;
- ◆ частичные произведения столбцов матрицы $[1, 2, 3, 4; 2, 3, 4, 5; 3, 4, 5, 6; 4, 5, 6, 7]$.

Решение.

```
>> x = 1 : 10;
>> P = prod(x)
P =
    3628800

>> x = [ 1, 4, 9, 16, 25 ];
>> P = prod(x)
P =
    14400

>> x = [ 1, 4, 9, 16, 25 ].^2;
>> P = prod(x)
P =
    207360000

>> x = [ 1, 2, 3, 4 ; 2, 3, 4, 5 ; 3, 4, 5, 6 ; 4, 5, 6, 7 ];
>> P = cumprod(x)
P =
     1     2     3     4
     2     6    12    20
     6    24    60   120
    24   120   360   840
```

4.4. Вычисление производных

В MATLAB производная находится с помощью следующей встроенной функции:

```
diff(f, x, n)
```

где:

- ◆ f — дифференцируемая функция;
- ◆ x — аргумент функции (переменная дифференцирования);
- ◆ n — порядок производной (по умолчанию $n=1$).

Технология вычисления производной:

1. Определение символьных переменных с помощью функции `syms()`.

2. Ввод функции дифференцирования f .
3. Ввод функции $\text{diff}(f, x, n)$ с конкретными значениями x и n .
4. Получение решения после нажатия клавиши <Enter>.

Будем иллюстрировать методику на примерах.

Пример 4.5

Пусть необходимо найти первую и третью производные функции $x \cos x$. Процедуры вычисления производных имеют вид:

```
>> syms x, n;
>> y = x * cos (x);
>> diff(y, x)
ans =
    - sin (x) * x + cos (x)
>> diff(y, x, 3)
ans =
    sin (x) * x - 3 cos (x)
```

Функция $\text{diff}(f, x, n)$ позволяет вычислять производные функций, содержащих символьные переменные.

Пример 4.6

Далее приведены решения для следующих трех функций:

$$y_1 = ax^2,$$

$$y_2 = n^x,$$

$$y_3 = e^{-ax^5} + \ln(a^n + x^a) - \frac{an}{x^3}.$$

Для функции y_2 вычислена третья производная.

```
>> syms a x n;
>> y1 = a * x^2;
>> y2 = n^x;
>> y3 = exp (- a * x^5) + log (a^n + x^a) - a * n / (x^3);
```

```
>> z1 = diff (y1, x)
z1 =
    2 * a * x
>> z2 = diff (y2, x, 3)
z2 =
    n^x * log (n)^3
>> z3 = diff (y3, x)
z3 =
    -5 * a * x^4 * exp (- a * x^5) + a * x^(a-1) / (a^n + x^a)
    + 3 * a * n/x^4
```

Функция дифференцирования имеет следующие особенности. Если переменная дифференцирования x в выражении `diff` отсутствует, а функция имеет вид `diff(f)`, то программа не выдает ошибки. Она осуществляет дифференцирование по переменной функции f в порядке, обратном алфавиту.

Например, если функция f содержит переменные a, b, c , то дифференцирование будет выполнено по переменной c . Если при этом в составе аргументов содержится переменная x , то она имеет абсолютный приоритет, независимо от ее положения в алфавите переменных.

Пример 4.7

Приведем примеры на все перечисленные случаи.

```
>> syms a b c x w;
>> diff(a + b^2)
ans =
    2 * b
>> diff(a + c * b^3)
ans =
    b^3
>> diff(a * w + c * b^3)
ans =
    a
>> diff(x * a * w + b^3)
ans =
    a * w
```

Функция f может быть вектором и матрицей. В таких случаях откликом будет также вектор или матрица, элементами которой будут производные от исходных функций, образующих вектор или матрицу.

Пример 4.8

```
>> syms a x;
>> y = [ x * sin(x) ; x^5 ; exp(a * x) ];
>> diff (y, x)
ans =
    cos (x) * x + sin (x)
    5 * x^4
    a * exp (a * x)
```

4.5. Вычисление пределов

Рассмотрим конкретный пример.

Пример 4.9

Вычислим значение функции

$$y = \frac{\sin x}{x}$$

в диапазоне значений x , равном $[0; 1]$, с шагом 0.2. Решение имеет вид:

```
>> x = 0 : 0.2 : 1;
>> y = sin (x) ./ x
y =
    NaN    0.9933    0.9735    0.9411    0.8967    0.8415
```

Ответ ошибочный. Программа не выдала значение функции при $x=0$, восприняв процедуру деления на ноль как недопустимую. Между тем, здесь имеет место неопределенность вида $0/0$ и функция

$$\frac{\sin 0}{0} = 1.$$

В подобных случаях при практических расчетах пользователю приходится анализировать результат программы. Искать предел функции, используя *правило Лопиталя*. Системы компьютерной алгебры позволяют находить пределы функции, в том числе и в случаях, когда имеют место неопределенности вида $0/0$, $0/\infty$, $\infty/0$, ∞/∞ .

В MATLAB пределы вычисляются с помощью функции `limit()`, имеющей синтаксис:

```
limit(f, x, x0)
```

где:

- ◆ f — функция, предел которой определяется;
- ◆ x — аргумент функции f ;
- ◆ x_0 — предельное значение x .

Наиболее часто пределы вычисляются при $x=0$ или при $x=\infty$. Символ ∞ кодируется в MATLAB словом `inf`.

Пример 4.10

Найти пределы функций:

$$y_1 = \frac{\sin x}{x} \quad \text{при } x \rightarrow 0,$$

$$y_2 = \frac{(1 - e^{-x})}{x} \quad \text{при } x \rightarrow \infty,$$

$$y_3 = \frac{(1 - x)}{\ln x} \quad \text{при } x \rightarrow 1.$$

Решения будут иметь вид:

```
>> syms x;
>> limit(sin(x)/x, x, 0)
ans =
    1
```

```
>> limit((1 - exp(-x))./x, x, inf)
ans =
    0
>> limit((1 - x)./log (x)), x, 1)
ans =
   -1
```

В табл. 4.1 приведены задачи на определение пределов функции. Примеры иллюстрируют возможности системы MATLAB.

Таблица 4.1. Функции и значения их пределов

№	Функция	Предельное значение аргумента	Ответ
1	$(1+x)^{\frac{1}{x}}$	∞	1
2	$\left(\frac{a^x + b^x}{2}\right)^{\frac{1}{x}}$	0	$\sqrt{a} \cdot \sqrt{b}$
3	$\left(1 + \frac{1}{x}\right)^{\frac{1}{x}}$	∞	1
4	$(1-x)^{\frac{1}{1-x}}$	1	NaN
5	$(1-x)^{\frac{1}{1-x}}$	∞	1
6	$\frac{1 - e^{-at}}{\log(1-t)}$	0	$-a$
7	$\frac{1 - a^n}{a n}$	0	$-\frac{\ln a}{a}$
8	$2a \frac{1 - e^{-ax}}{2 - e^{-ax}}$	∞	a

Таблица 4.1 (окончание)

№	Функция	Предельное значение аргумента	Ответ
9	$\frac{x-a}{\ln(x-a+1)}$	a	1
10	$\frac{n^2 - n^x}{x-2}$	2	$-n^2 \ln n$

4.6. Разложение функции в степенной ряд

Разложение функции $y = f(x)$ в степенной ряд осуществляется по формуле Тейлора:

$$f(x) = f(a) + (x-a) \frac{f'(a)}{1!} + (x-a)^2 \frac{f''(a)}{2!} + \dots + (x-a)^n \frac{f^{(n)}(a)}{n!} + \dots$$

В формуле используются следующие обозначения:

- ◆ a — значение аргумента x функции $y = f(x)$, вокруг которого происходит разложение в ряд;
- ◆ $f(a)$, $f'(a)$, $f''(a)$, ..., $f^{(n)}(a)$ — значения функции и ее производных в точке a .

При $a = 0$ формула называется *рядом Маклорена* и имеет вид:

$$f(x) = f(0) + x \frac{f'(0)}{1!} + x^2 \frac{f''(0)}{2!} + \dots + \frac{x^n}{n!} f^{(n)}(0).$$

Разложение функции в степенной ряд реализуется в системе MATLAB с помощью функции `Taylor()`, которая имеет вид:

`Taylor(f(x), x, x0, n)`

где:

- ◆ $f(x)$ — функция, разлагаемая в степенной ряд;
- ◆ x — аргумент функции $f(x)$;
- ◆ x_0 — значение x , вокруг которого происходит разложение функции $f(x)$;
- ◆ n — число членов разложения.

Технология разложения функции в ряд:

1. Определение символьных переменных с помощью функции `syms()`.
2. Ввод функции $y = f(x)$.
3. Ввод функции `Taylor(y, x, x0, n)`.
4. Получение решения путем нажатия клавиши <Enter>.

В качестве примера разложим в ряд Тейлора функции

$$y_1 = \sin x,$$

$$y_2 = e^x,$$

$$y_3 = \sinh x,$$

$$y_4 = \frac{3x+1}{2x^2+5x+1},$$

$$y_5 = \ln x,$$

$$y_6 = \frac{\sin x}{x}.$$

Пример 4.11

Выполним разложение вокруг $x_0=0$ с числом членов $n=5$.

Решение имеет вид:

```
>> syms x, y1, y2, y3, y4, y5, y6, n;
>> y1 = sin(x); y2 = exp(x); y3 = sinh(x);
>> y4 = (3 * x + 1) (2 * x^2 + 5 * x + 1);
```

```
>> y5 = log(x); y6 = sin(x)./x;
>> z1 = Taylor(y1, x, 0, 5)
z1 =
    x - 1/6 * x^3
>> z2 = Taylor(y2, x, 0, 5)
z2 =
    1 + x + 1/2 * x^2 + 1/6 * x^3 + 1/24 * x^4
>> z3 = Taylor(y3, x, 0, 5)
z3 =
    x + 1/6 * x^3
>> z4 = Taylor(y4, x, 0, 5)
z4 =
    1 - 2 * x + 8 * x^2 - 36 * x^3 + 164 * x^4
>> z5 = Taylor(y5, x, 0, 5)
z5 =
```

Решения нет.

```
>> z6 = Taylor(y6, x, 0, 5)
z6 =
    1 - 1/6 * x^2 + 1/120 * x^4
```

Обратим внимание на то, что степенные ряды функций $\sin x$ и $\sinh x$ имеют только два члена, в то время как функция `Taylor()` содержит $n=5$. Это объясняется тем, что четные производные функции $\sin x$ равны нулю.

Функцию $y_5 = \ln x$ программа не разложила в степенной ряд вокруг $x=0$ потому, что $\ln 0$ не существует.

При разложении функции в степенной ряд всегда возникают следующие вопросы:

- ◆ Возможно ли разложение данной функции в степенной ряд?
- ◆ Какое число членов ряда должно быть для обеспечения требуемой точности?
- ◆ Как найти погрешности степенного ряда?

Ответим на эти вопросы.

Разложить в ряд Тейлора можно только такую функцию, которая имеет n производных и ряд является сходящимся. При разложении функции в степенной ряд ограничиваются определенным числом членов ряда. Поэтому всегда необходимо знать, сколько членов ряда нужно иметь, чтобы обеспечить заданную точность вычисления функции.

Необходимое число членов ряда зависит от вида функции и значения аргумента.

Рассмотрим типичные случаи.

Случай 1. Ряд знакопеременный, а члены ряда убывают по величине.

В этом случае ряд обладает следующим свойством: сумма отброшенных членов ряда не превосходит последнего оставленного члена. В таком случае определить число членов ряда можно на основании анализа общего члена ряда. В качестве примера рассмотрим убывающий знакопеременный ряд функции e^{-x} . Общий член этого ряда имеет вид:

$$(-1)^n \frac{x^n}{n!}.$$

Так как все отброшенные члены ряда не превосходят этого, то, задавшись погрешностью ε , можно найти число n из условия:

$$\frac{x^n}{n!} \leq \varepsilon.$$

Пусть, например, $x = 1.5$, $\varepsilon = 0.01$. Тогда

$$n! = \frac{x^n}{\varepsilon} \text{ или } n! = 1.5^n \times 100.$$

Решая это уравнение, получим $n \approx 6$.

Случай 2. Ряд знакопеременный, x большое. В этом случае свойство частичной суммы ряда сохраняется, но ряд сходится медленно и число членов ряда может быть очень большим.

Случай 3. Ряд не знакопеременный. В этом случае общих правил выбора числа членов не существует.

Погрешность ряда можно оценить при компьютерных технологиях, по крайней мере, следующими способами:

- ◆ табулированием исходной функции и функции, полученной в результате разложения в степенной ряд;
- ◆ построением графиков функций;
- ◆ вычислением ошибок.

Рассмотрим первый способ на примере.

Пример 4.12

Разложим функцию $y = e^x$ в ряд Тейлора при $n = 3, 4, 5$:

$$y_3 = 1 + x + \frac{x^2}{2} + \frac{x^3}{6},$$

$$y_4 = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24},$$

$$y_5 = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120}.$$

Выполним табуляцию функций $y = e^x$, y_3 , y_4 и y_5 при изменении x в диапазоне от 0 до 2 с шагом $h=0.2$.

Программа на языке системы MATLAB имеет вид:

```
>> syms x, y, y3, y4, y5;
>> x = 0 : 0.2 : 2;
>> y = exp(x);
>> y3 = 1 + x + x.^2./2 + x.^3./6;
>> y4 = 1 + x + x.^2./2 + x.^3./6 + x.^4./24;
>> y5 = 1 + x + x.^2./2 + x.^3./6 + x.^4./24 + x.^5./120;
>> z = [ x; y; y3; y4; y5 ];
>> z'
```

z =

x	y	y3	y4	y5
0	1.0000	1.0000	1.0000	1.0000
0.2000	1.2214	1.2213	1.2214	1.2214

0.4000	1.4918	1.4907	1.4917	1.4918
0.6000	1.8221	1.8160	1.8214	1.8220
0.8000	2.2255	2.2053	2.2224	2.2251
1.0000	2.7183	2.6667	2.7083	2.7167
1.2000	3.3201	3.2080	3.2944	3.3151
1.4000	4.0552	3.8373	3.9974	4.0422
1.6000	4.9530	4.5627	4.8357	4.9231
1.8000	6.0496	5.3920	5.8294	5.9869
2.0000	7.3891	6.3330	7.0000	7.2667

Из решения видно, что погрешности рядов при $n = 3, 4, 5$ достаточно большие, они уменьшаются с увеличением числа членов n .

Наглядное представление о погрешностях рядов можно получить, если представить функции в виде графиков. Этот способ позволяет визуально установить область значений x , в которой разложение допустимо.

Наиболее эффективным методом оценки погрешности ряда является определение среднеквадратических погрешностей. Этот метод подробно рассмотрен в *главе 11*.

4.7. Определение экстремумов функции

Классическим способом определения экстремума (максимума или минимума) функции $y = f(x)$ является определение корня ее производной. В системе MATLAB этот метод может быть реализован с помощью функций

```
diff(f, x)
solve('fun', x)
```

Первая определит производную, а вторая значение x , при котором находится экстремум функции $f(x)$.

Покажем технологию определения максимума функции системой MATLAB по этому методу на примере.

Пример 4.13

Пусть функция имеет вид: $y = xe^{-x}$. Требуется определить координаты ее максимума.

Решение задачи.

1. Определение области $x_1 \leq x \leq x_2$ нахождения максимума функции.

Представим функцию в виде графика:

```
>> x = 0 : 0.1 : 3 ;  
>> y = x. * exp(-x);  
>> plot(x,y)
```

График функции приведен на рис. 4.1.

Из рис. 4.1 видно, что областью изоляции максимума функции может быть $0.5 \leq x \leq 1.5$.

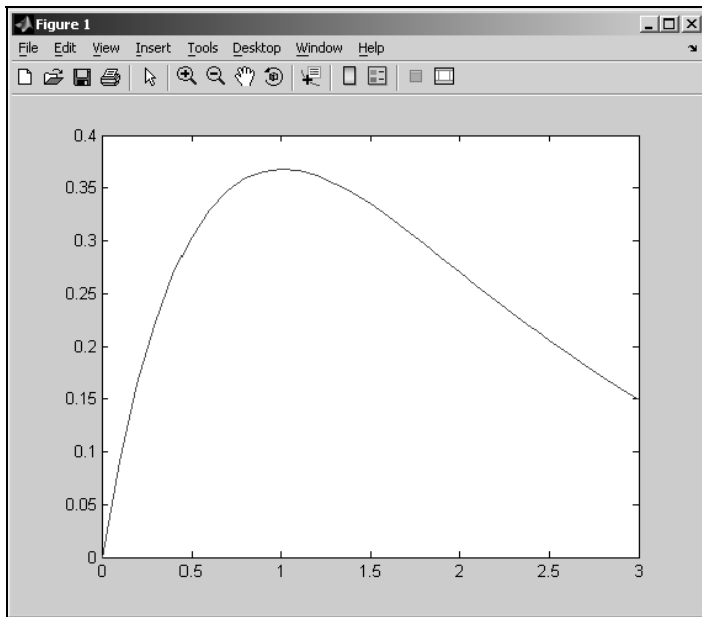


Рис. 4.1. График функции $y = xe^{-x}$

2. Вычисление производной функции $y = xe^{-x}$:

```
>> syms x, y, z;  
>> z = diff(y, x)  
z =  
    exp (-x) - x exp (-x)
```

3. Определение корня производной:

```
>> solve('exp(-x) - x * exp(-x) = 0', x)  
ans =  
    1
```

4. Определение значения функции:

```
>> x = 1;  
>> y = x. * exp(-x)  
y =  
    0.3679
```

MATLAB имеет встроенную функцию, позволяющую определить координаты экстремума функции, не вычисляя корня производной. Такой функцией является `fmin()` в нескольких вариантах реализации.

Вычисление минимума функции в MATLAB осуществляется с помощью следующих встроенных функций:

```
fmin ('fun', x1, x2)  
fmin ('fun', x1, x2, options)  
fmin ('fun', x1, x2, options, P1, ..., P10)
```

В этих функциях приняты следующие обозначения:

- ◆ 'fun' — взятая в одинарные кавычки функция, минимум которой определяется;
- ◆ x1, x2 — область значений аргумента, в которой находится минимум функции 'fun';
- ◆ options — вектор, содержащий компоненты управления процессом решения задачи, например: `options(1)` — вывод значений промежуточных итераций, `options(2)` — задание погреш-

ностей итераций (по умолчанию $1e-04$), `options(14)` — задание числа итераций (по умолчанию 500).

4.7.1. Функция *fmin* ('fun', x1, x2)

Эта функция определяет значение аргумента x минимума функции 'fun' из диапазона $x_1 \leq x \leq x_2$.

Приведем примеры определения локального минимума функции.

Пример 4.14

Пусть дана нелинейная функция

$$y = 2^x - 4x + 6,$$

координаты локального минимума которой необходимо определить.

Решение. Определим область значений $x_1 \leq x \leq x_2$, для чего создадим график функции:

```
>> x = 0 : 0.1 : 6;
>> y = 2.^x - 4 * x + 6;
>> plot(x, y)
```

График показан на рис. 4.2.

Из рис. 4.2 видно, что областью значений аргумента x , в которой находится минимум функции, может быть $2 \leq x \leq 4$. Теперь для определения x воспользуемся функцией `fmin('fun', x1, x2)`:

```
>> x = fmin ('2^x - 4 * x + 6', 2, 4)
x =
    2.5288
```

Найдем значение локального минимума:

```
>> x = 2.5288;
>> Y = 2.^x - 4 * x + 6
Y =
    1.6557
```

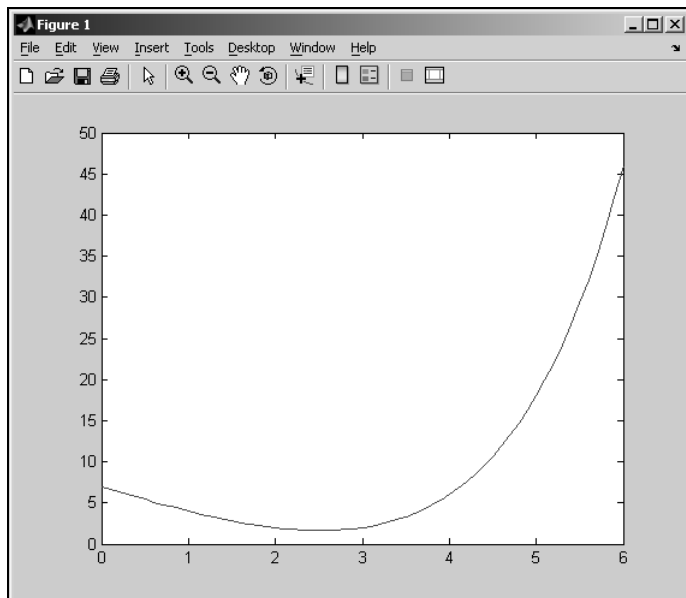


Рис. 4.2. График функции $y = 2^x - 4x + 6$

Таким образом, координатами локального минимума функции являются $(2.5288, 1.6557)$.

Функция `fmin('fun',x1,x2)` позволяет определять также максимум функции $y = f(x)$. Для этого необходимо перед функцией $y = f(x)$ поставить знак минус.

Пример 4.15

В качестве примера определим максимум рассмотренной выше функции $y = xe^{-x}$:

```
>> syms x;
>> fmin('-x * exp (-x)', 0.5, 1.5)
ans =
    1
```

4.7.2. Функция *fmin* ('fun', x1, x2, options)

Покажем результаты этой функции на примере.

Пример 4.16

Нужно определить минимум функции

$$y = 2^x - 4x + 6.$$

Решение:

```
>> syms x;
>> [x, options] = fmin('2.^x - 4 * x + 6', 2, 4, [0.1 e - 8])
x =
    2.5288
options =
    columns 1 thround 10
    0  0.0001  0.0001  0.0000  0  0  0  1.6557  0  11.0000
    columns 11 thround 18
    0  0  0  500.000  0  0.0000  0.10000  0
```

На восьмом месте приведено максимальное значение функции, равное 1.6557, на десятом показано число выполненных итераций (11), на четырнадцатом — заданное максимальное число итераций (500 по умолчанию).

В MATLAB имеется возможность решения задач минимизации функций нескольких переменных. Такими встроенными функциями являются:

```
fmins('fun', x0)
[x, options] = fmins('fun', x0)
fmins('fun', x0, options)
```

В этих функциях приняты следующие обозначения:

- ◆ 'fun' — многопараметрическая функция $y = f(x_1, x_2, \dots, x_n)$;
- ◆ x0 — вектор начальных значений;
- ◆ options — вектор компонентов управления решением задачи.

♦ Используется три параметра:

- `options(1)` — вывод результатов промежуточных итераций;
- `options(2)` — задание погрешностей вычисления аргумента (по умолчанию $1e-04$);
- `options(14)` — задание максимального количества итераций (по умолчанию $200n$, где n — число переменных).

4.8. Интегральные преобразования

Интегральные преобразования находят широкое применение при решении дифференциальных уравнений, вычислении предельных значений функции $f(x)$, исследовании динамики систем управления, систем массового обслуживания и во многих других технических и научных задачах.

Наиболее популярными являются преобразования Лапласа, Карсона и z -преобразование.

Здесь рассматривается только преобразование Лапласа, реализованное в системе MATLAB.

4.8.1. Преобразование Лапласа

Преобразование Лапласа функции $f(x)$ имеет вид:

$$L\{f(x)\} = \int_{-\infty}^{\infty} f(x)e^{-sx} dx, \quad (4.1)$$

где $f(x)$ — функция, преобразование Лапласа которой необходимо найти.

Если аргументом функции является время t , то преобразование Лапласа имеет вид:

$$L\{f(t)\} = \int_0^{\infty} f(t)e^{-st} dt. \quad (4.2)$$

Получим преобразование Лапласа для некоторых простых функций.

Пусть

$$\begin{aligned}f(t) &= a, \\f(t) &= e^{-at}, \\f(t) &= \sin(\omega t).\end{aligned}$$

Тогда на основании (4.2) имеем:

$$\begin{aligned}L(a) &= \int_0^{\infty} a e^{-st} dt = \frac{a}{s}, \\L(e^{-at}) &= \int_0^{\infty} e^{-at} e^{-st} dt = \frac{1}{s+a}, \\L(\sin(\omega t)) &= \int_0^{\infty} \sin(\omega t) e^{-st} dt = \frac{1}{s^2 + \omega^2}.\end{aligned}$$

Преобразование Лапласа производной n -го порядка имеет вид:

$$L\left(\frac{d^n x(t)}{dt^n}\right) = s^n x(s) - x_{(0)}^{(n-1)} - x_{(0)}^{(n-2)} - \dots - x(0),$$

где $x_{(0)}^i$ — значение i -й производной при $t = 0$.

Преобразование Лапласа для интеграла имеет вид:

$$L\left(\int_0^t f(t) dt\right) = \frac{1}{s} f(s).$$

С помощью преобразования Лапласа можно существенно упростить решение ряда задач, связанных с определением пределов функции. Для этого служат следующие предельные теоремы:

$$\begin{aligned}\lim_{t \rightarrow 0} f(t) &= \lim_{s \rightarrow \infty} s f(s), \\ \lim_{t \rightarrow \infty} f(t) &= \lim_{s \rightarrow 0} s f(s).\end{aligned}\tag{4.3}$$

В книгах и справочниках приводятся преобразования Лапласа многих функций. Эти результаты легко получить с помощью систем компьютерной алгебры.

В системе MATLAB преобразование Лапласа функции $f(t)$ осуществляется с помощью следующих встроенных функций:

- ◆ `Laplace(F)` — преобразование Лапласа символьной переменной F ;
- ◆ `Laplace(F,s)` — преобразование Лапласа по формуле (4.2);
- ◆ `Laplace(F,omega,s)` — преобразование Лапласа по переменной ω .

Функция *Laplace(F)*

Если функция F является аргументом t , то преобразование Лапласа осуществляется по формуле (4.2). Если же в F аргумент t отсутствует, то преобразование Лапласа осуществляется по переменной в соответствии с алфавитом переменных функции F .

Рассмотрим этот случай на примере.

Пример 4.17

Пусть необходимо найти преобразование Лапласа функции $F = a$.

Решение:

```
>> syms a;
>> Laplace(a)
ans =
```

$$\frac{1}{s^2}$$

Этот ответ не совпадает с известными значениями преобразования Лапласа постоянной a . Известно, что

$$L(a) = \frac{a}{s},$$

который получается в результате преобразования по формуле (4.2):

$$f(s) = \int_0^{\infty} a e^{-st} dt = \frac{a}{s}.$$

В данном примере функция F не содержит переменной t , поэтому функция `Laplace(F)` вычисляет преобразование Лапласа по переменной a :

$$L(a) = \int_0^{\infty} a e^{-at} dt = \frac{1}{s^2}.$$

Если необходимо найти преобразование Лапласа переменной n , представляющей собой число, например, $n=2$, то функция `Laplace(F)` решения не дает. Это объясняется тем, что в данном случае в выражении F отсутствует переменная интегрирования.

Пример 4.18

Пусть функция $f(t) = a + bc$, т. е. не содержит в качестве аргумента t . Найдем преобразование Лапласа этой функции с помощью функции `Laplace(F)`:

```
>> syms a b c t s w;
```

```
>> Laplace(a + b * c)
```

```
ans =
```

$$\frac{a}{s} + \frac{b}{s^2}$$

```
>> Laplace(a + d * c)
```

```
ans =
```

$$\frac{a}{s} + \frac{c}{s^2}$$

```
>> Laplace(a + d * w)
```

```
ans =
```

$$\frac{a}{s} + \frac{d}{s^2}$$

```
>> Laplace(a + w * t)
```

```
ans =
```

$$\frac{a}{s} + \frac{w}{s^2}$$

Из примера видно, что функция $\text{Laplace}(F)$ не воспринимает преобразуемые выражения как постоянные величины. Преобразования Лапласа получаются путем интегрирования выражений соответственно по переменным c, d, w, t , т. е. в обратном алфавитном порядке буквенных символов. При этом абсолютным приоритетом обладает переменная t .

Функция $\text{Laplace}(F, s)$

Эта функция аналогична функции $\text{Laplace}(F)$. Ее отличие в том, что она позволяет находить преобразование Лапласа для случая численных значений функции F .

Пример 4.19

```
>> Laplace(3.5, s)
ans =
    3.5
    s
```

Функция $\text{Laplace}(F, w, s)$

Функция отличается от предыдущих тем, что в интегрируемой функции F указывается переменная интегрирования ω . Функция обеспечивает прямое преобразование Лапласа по формуле:

$$L(s) = \int_0^{\infty} f(\omega) e^{-\omega s} d\omega. \quad (4.4)$$

Пример 4.20

```
>> syms a b c x s t;
>> Laplace(a, t, s)
ans =
    a
    s

>> Laplace(t * exp (-a * t), t, s)
```

ans =

$$\frac{1}{(s+a)^2}$$

```
>> laplace (a + b * c, b, s)
```

ans =

$$\frac{a}{s} + \frac{c}{s^2}$$

4.8.2. Решение дифференциальных уравнений с помощью преобразования Лапласа

Преобразование Лапласа находит широкое применение при решении задач, связанных с анализом дифференциальных уравнений. Приведем один из таких примеров.

Пусть имеется многоканальная система массового обслуживания с отказами. Интенсивность потока заявок на обслуживание λ , интенсивность обслуживания заявки μ , число обслуживающих каналов $n = 2$.

Такая система может находиться в следующих состояниях:

- ◆ s_0 — начальное состояние системы, когда она свободна от обслуживания (заявок нет);
- ◆ s_1 — обслуживается одна заявка, второй обслуживающий орган свободен от обслуживания;
- ◆ s_2 — обслуживаются две заявки, оба обслуживающих органа заняты обслуживанием, очередной заявке будет отказано в обслуживании.

Так как время поступления заявки на обслуживание и время обслуживания являются величинами случайными, то эффективность такой системы характеризуется вероятностями состояний. Эти вероятности могут быть получены путем решения следующей системы дифференциальных уравнений:

$$\frac{d P_0(t)}{d t} = -\lambda P_0(t) + \mu P_1(t),$$

$$\frac{d P_1(t)}{d t} = \lambda P_0(t) - (\lambda + \mu) P_1(t) + 2\mu P_2(t),$$

$$\frac{d P_2(t)}{d t} = \lambda P_1(t) - 2\mu P_2(t).$$

Будем определять вероятности состояний при следующих начальных условиях:

$$P_0(0) = 1, P_1(0) = P_2(0) = 0.$$

Представим систему дифференциальных уравнений в преобразованиях Лапласа:

$$sP_0(s) - P_0(0) = -\lambda P_0(s) + \mu P_1(s),$$

$$sP_1(s) - P_1(0) = \lambda P_0(s) - (\lambda + \mu) P_1(s) + 2\mu P_2(s),$$

$$sP_2(s) - P_2(0) = \lambda P_1(s) - 2\mu P_2(s).$$

Так как $P_0(0) = 1$, а $P_1(0) = P_2(0) = 0$, то после очевидных преобразований получим:

$$(s + \lambda)P_0 - \mu P_1 = 1$$

$$-\lambda P_0 + (\lambda + \mu + s)P_1 - 2\mu P_2 = 0$$

$$-\lambda P_1 + (s + 2\mu)P_2 = 0.$$

Аргумент s в выражениях для P опущен для краткости уравнений.

Полученная система уравнений является алгебраической с постоянными коэффициентами.

Состояние s_2 является отказовым для очередной заявки на обслуживание. Тогда вероятность $P_c(t)$ того, что заявка будет при-

нята на обслуживание в произвольный момент времени t , равна сумме вероятностей состояний s_0 и s_1 , т. е.

$$P_c(t) = P_0(t) + P_1(t).$$

Система MATLAB позволяет решать системы алгебраических уравнений в аналитическом виде (см. главу 8). Полученное нами решение имеет вид:

$$P_0(s) = \frac{s^2 + (\lambda + 3\mu)s + 2\mu^2}{\Delta},$$

$$P_1(s) = \frac{\lambda s + 2\lambda\mu}{\Delta},$$

$$P_c(s) = P_0(s) + P_1(s) = \frac{s^2 + (2\lambda + 3\mu)s + 2\lambda\mu + 2\mu^2}{\Delta},$$

$$\Delta = s(s^2 + (2\lambda + 3\mu)s + \lambda^2 + 2\lambda\mu + 2\mu^2).$$

Найдем теперь финальные вероятности состояний, воспользовавшись предельными теоремами (4.3)

$$\lim_{t \rightarrow \infty} P_c(t) = \lim_{s \rightarrow 0} sP_c(s).$$

На основании этой теоремы имеем:

$$P_c = \frac{2\lambda\mu + 2\mu^2}{\lambda^2 + 2\lambda\mu + 2\mu^2}.$$

Полученная формула позволяет определить готовность системы принять заявку на обслуживание в любой момент времени t . Так, например, если интенсивность потока заявок равна интенсивности обслуживания, т. е. $\lambda = \mu$, то $P_c = \frac{4}{5} = 0.8$.

4.8.3. Обратное преобразование Лапласа

Для получения решения системы дифференциальных уравнений во временной области необходимо полученное решение в преобразованиях Лапласа представить в функции t .

Обратное преобразование Лапласа имеет вид:

$$f(t) = \frac{1}{2\pi i} \int_{\delta-i\infty}^{\delta+i\infty} F(s) e^{st} ds. \quad (4.5)$$

Существуют таблицы обратных преобразований различных функций. Однако при наличии универсальных программных средств символьной математики обращаться к ним нет необходимости.

В MATLAB обратное преобразование Лапласа находится с помощью функции `iLaplace()`, которая имеет вид:

`iLaplace(L(s), t)`

где:

- ◆ $L(s)$ — прямое преобразование Лапласа;
- ◆ t — аргумент искомой функции $f(t)$.

Технология получения обратного преобразования Лапласа такова:

1. Создание группы символьных объектов с помощью функции `syms()`.
2. Набор и ввод функции $L(s)$.
3. Набор и ввод функции `iLaplace(L(s), t)`.
4. Получение решения путем нажатия клавиши <Enter>.

Рассмотрим технологию обратного преобразования Лапласа на примере.

Пример 4.21

Необходимо получить оригинал функции

$$L(s) = \frac{a + b s}{s^2}.$$

Решение:

```
>> syms a b s t L;
>> L = (a + b * s) / s^2;
>> iLaplace(L, t)
```

```
ans =
    a t + b
```

Пример 4.22

Получим теперь оригинал функции $P_c(s)$ решения системы дифференциальных уравнений массового обслуживания.

Пусть $\lambda = \mu = 1$. Тогда

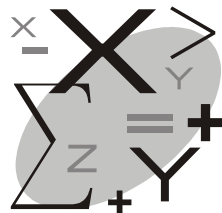
$$P_c(s) = \frac{s^2 + 5s + 4}{s(s^2 + 5s + 5)}.$$

Получение оригинала функции $P_c(s)$:

```
>> syms s t P;
>> P = (s^2 + 5 * s + 4) / (s * (s^2 + 5 * s + 5));
>> iLaplace (P, t)
ans =
```

$$\frac{4}{5} + \frac{1}{5} e^{-\frac{5}{2}t} \cos h\left(\frac{1}{2}\sqrt{5}t\right) + \frac{\sqrt{5}}{5} e^{-\frac{5}{2}t} \sin h\left(\frac{1}{2}\sqrt{5}t\right)$$

Преобразование Лапласа позволило найти решение в виде формулы. В этом его главное достоинство.



ГЛАВА 5

Вычисление математических функций

Основными видами математических функций являются:

- ◆ элементарные функции;
- ◆ специальные функции;
- ◆ функции пользователя.

Приведем перечень функций и покажем способы их вычисления.

5.1. Элементарные функции

Математические функции представляются в виде $\text{fun}(x)$, где fun — имя функции, x — аргумент в виде числа, вектора или матрицы. Числа, элементы вектора или матрицы могут быть вещественными или комплексными. Если число комплексное, то при вычислении абсолютного значения x откликом будет модуль комплексного числа.

Технология вычисления элементарных функций предельно проста: осуществляется ввод функции с заданным значением аргумента x и нажимается клавиша <Enter>. На экране появится ответ в виде числа, вектора или матрицы.

Далее приводятся элементарные функции и их вычисления в системе MATLAB.

5.1.1. Алгебраические и арифметические функции

◆ $\text{abs}(x)$ — абсолютное значение x .

Переменная x может быть вещественным или комплексным числом, вектором или матрицей. Если x — число комплексное, то ответом будет модуль комплексного числа.

Пример 5.1

Найти с помощью функции $\text{abs}(x)$ абсолютное значение чисел следующих векторов и матриц:

$Y1 = [-3, 5]$, $Y2 = [2, -3, 2+3i, i]$,

$Y3 = [2, -3; 1, 2+3i; -2, -5]$.

Решение:

```
>> Y1=[-3,5];
>> Y2=[2, -3, 2+3i, i];
>> Y3=[2, -3; 1, 2+3i; -2, -5];
>> abs(Y1)
ans =
     3     5
>> abs(Y2)
ans =
     2.0000     3.0000     3.6056     1.0000
>> abs(Y3)
ans =
     2.0000     3.000
     1.0000     3.6056
     2.0000     5.0000
```

◆ $\exp(x)$ — экспоненциальная функция.

Вычисляет значение e^x , если аргумент x — вещественное число. Если x — комплексное, то вычисляет так называемую комплексную экспоненту:

$$e^z = e^x (\cos y + i \sin y),$$

где $z = x + iy$.

Пример 5.2

Найти экспоненциальную функцию для следующих аргументов:

[1 2 3 4 5], [2.5+7i, -1, 1],
[-1, 0.1, i; 3+1.2i, -i, 5; -0.5, 0.5, 2].

Решение:

```
>> Y1 = [1, 2, 3, 4, 5];
>> Y2 = [2.5+ 7i, -1, 1];
>> Y3 = [-1, 0.1, i; 3+1.2i, -i, 5; -0.5, 0.5, 2];
>> exp(Y1)
ans =
    2.7183    7.3891   20.0855   54.5982   148.4132
>> exp(Y2)
ans =
    9.1844 + 8.0037i    0.3679    2.7183
>> exp(Y3)
ans =
    1.0e+002*
    0.0037            0.0111            0.0054+0.0084i
    0.0728+0.1872i    0.0054-0.0084i    1.4841
    0.0061            0.0165            0.0739
```

◆ **Логарифмические функции** $\log(x)$, $\log_{10}(x)$, $\log_2(x)$.

Вычисляются логарифмы чисел с основанием e , 10, 2.

Аргумент x может быть числом положительным и отрицательным, вектором и матрицей. В случае вектора или матрицы вычисляют логарифм каждого элемента. Если число комплексное $z = x + iy$ или отрицательное, то вычисляют так называемый комплексный логарифм:

$$\log(z) = \log(\text{abs}(z)) + i \cdot \text{atan2}(\text{imag}(z), \text{real}(z))$$

Пример 5.3

Найти значения $\log(x)$, $\log_{10}(x)$, $\log_2(x)$ при следующих значениях x :

$x=[1 \ 2 \ 3 \ 4 \ 5]$, $x=1+2i$, $x=-5$.

Решение:

```
>> Y1 = [1 2 3 4 5];
>> Y2 = 1+2i;
>> Y3 = -5;
>> log(Y1)
ans =
    0    0.6931    1.0986    1.3863    1.6094
>> log10(Y1)
ans =
    0    0.3010    0.4771    0.6021    0.6990
>> log2(Y1)
ans =
    0    1.0000    1.5850    2.0000    2.3219
>> log(Y2)
ans =
    0.8047 + 1.1071i
>> log10(Y2)
ans =
    0.3495 + 0.4808i
>> log2(Y2)
ans =
    1.1610 + 1.5973i
>> log(Y3)
ans =
    1.6094 + 3.1416i
>> log10(Y3)
ans =
    0.6990 + 1.3644i
>> log2(Y3)
ans =
    2.3219 + 4.5324i
```

◆ `sqrt(x)` — корень квадратный из x .

При этом аргумент x может быть числом, вектором или матрицей. Если x — вектор или матрица, то функция возвращает корень квадратный из каждого элемента вектора или матрицы. Если число отрицательное или комплексное, то результатом будет комплексное число.

Пример 5.4

Найти корень квадратный из следующих значений x :

$x_1 = -7$, $x_2 = [1, 2, 3+4.5i]$,
 $x_3 = [1, 3, 5; -2, i, 4; 7, 3, 1]$.

Решение:

```
>> x1 = -7;
>> x2 = [1 2 3+4.5i];
>> x3 = [1 3 5; -2 i 4; 7 3 1];
>> sqrt(x1)
ans =
    0+2.6458i
>> sqrt(x2)
ans =
    1.0000    1.4142    2.0504+1.0973i
>> sqrt(x3)
ans =
    1.0000          1.7321          2.2361
    0 + 1.4142i    0.7071 + 0.7071i    2.0000
    2.6458          1.7321          1.0000
```

◆ $\text{mod}(x, y)$ — результатом является остаток от деления x на y .

Пример 5.5

Необходимо найти остаток от деления x на y , если

$x_1=5, y_1=3$; $x_2=-12, y_2=7$; $x_3=8+4i, y_3=2$.

Решение:

```
>> z1=mod(x1,y1)
ans =
    2
>> z2=mod(x2,y2)
ans =
    2
>> z3=mod(x3,y3)
ans =
```

Ответа нет

5.1.2. Тригонометрические функции

Тригонометрические функции $\sin x$, $\cos x$, $\operatorname{tg} x$, $\operatorname{ctg} x$, $\sec x$, $\operatorname{cosec} x$ представляются в MATLAB в следующем виде:

<code>sin(x)</code>	<code>tg(x)</code>	<code>sec(x)</code>
<code>cos(x)</code>	<code>cot(z)</code>	<code>csc(x)</code>

Аргумент x может быть числом положительным и отрицательным, вещественным и комплексным, вектором и матрицей.

Покажем процедуры вычисления тригонометрических функций на примерах.

Пример 5.6

```
>> sin([1,2,0,pi/2])
ans =
    0.8415    0.9093    0    1.0000
>> cos([-0.6, pi/6, pi/2])
ans =
    0.8253    0.8660    0.0000
>> tan([2+3i, pi/2, 0])
ans =
    1.0e+16
   -0.0000 + 0.0000i    1.6363    0
>> cot([1, pi/2, 3, 4])
ans =
    0.6421    0.0000   -7.0153    0/8637
>> sec([1, 2, 2.5; -1, 3, 2.1; 0.5, 6, -2])
ans =
    1.8508    -2.4030    -1.2482
    1.8508    -1.0101    -1.9808
    1.1395     1.0415    -2.4030
>> csc([i, 0, -2, pi/4])
ans =
    0   -0.8509i    inf    -1.0998    1.4142
```

5.1.3. Обратные тригонометрические функции

Обратные тригонометрические функции $\arcsin x$, $\arccos x$, $\arctg x$, $\operatorname{arccotg} x$, $\operatorname{arcsec} x$, $\operatorname{arccosec} x$ в системе MATLAB представляются в следующем виде:

<code>asin(x)</code>	<code>atan(x)</code>	<code>asec(x)</code>
<code>acos(x)</code>	<code>acot(x)</code>	<code>acsc(x)</code>

Аргумент x может быть числом положительным и отрицательным, представлять собой вектор или матрицу. Если x — вектор или матрица, то откликом будет также вектор или матрица, элементами которых являются значения обратных тригонометрических функций. Следует иметь в виду, что действительными значениями x функций `asin(x)` и `acos(x)` являются числа из диапазона $[-1; 1]$. При этом значения `asin(x)` находятся в диапазоне $\left[-\frac{\pi}{2}; \frac{\pi}{2}\right]$, а `acos(x)` в диапазоне $[0; \pi]$. Для случая действительных значений x , находящихся вне области $[-1; 1]$, функции `asin(x)` и `acos(x)` являются комплексными.

Если x — комплексное число, то обратная тригонометрическая функция также будет числом комплексным.

Пример 5.7

```
>> asin([1, 0, -2, -0.5])
ans =
    1.5708    0    1.5708 - 1.3170i    -0.5236
>> acos([1, 0, -2, -0.5,i])
ans =
    0    1.5708    3.1416 - 1.3170i    2.0944    1.5708 - 0.8814i
>> atan([1, 0, 2,-0.5, i])
ans =
   -0.7854    0    1.1071   -0.4636   NaN + NaNi
>> acot([-1, 0, 2, -0.5+i])
ans =
   -0.7854    1.5708    0.4636   -1.1071   NaN + NaNi
```

```
>> asec([1, 0, 2, -0.5])
ans =
    0          0 + infi      1.0472    3.1416-1.3170i
>> acsc([1, 0, 2, -0.5, i])
ans =
    1.5708    NAN-infi    0.5236   -1.5708 + 1.3170i    0 - 0.8814i
```

5.1.4. Гиперболические функции

Гиперболические функции $\operatorname{sh} x$, $\operatorname{ch} x$, $\operatorname{th} x$, $\operatorname{cth} x$, $\operatorname{sch} x$, $\operatorname{csch} x$ имеют вид:

$\sinh(x)$	$\tanh(x)$	$\operatorname{sech}(x)$
$\cosh(x)$	$\operatorname{coth}(x)$	$\operatorname{csch}(x)$

Они выражаются через экспоненциальные функции и имеют вид следующих формул:

$$\operatorname{sh} x = \frac{e^x - e^{-x}}{2},$$

$$\operatorname{ch} x = \frac{e^x + e^{-x}}{2},$$

$$\operatorname{th} x = \frac{e^{2x} - 1}{e^{2x} + 1},$$

$$\operatorname{cth} x = \frac{e^{2x} + 1}{e^{2x} - 1},$$

$$\operatorname{sch} x = \frac{2e^x}{e^{2x} + 1},$$

$$\operatorname{csch} x = \frac{2e^x}{e^{2x} - 1}.$$

Аргумент x может быть положительным и отрицательным, вещественным и комплексным числом, вектором и матрицей. Элементы вектора и матрицы могут быть вещественными или комплексными числами.

Пример 5.8

```
>> sinh([0, 1, -1, 0.7, 5])
ans =
    0          1.1752      -1.1752    0.7586    74.2032
>> cosh([0, 1, -1, 2, 3+0.5i])
ans =
    1.0000    1.5431    1.5431    3.7622    8.8352 + 4.8028i
>> tanh([0, 1, -1, 0.7, 5])
ans =
    0          0.7616   -0.7616    0.6044    0.9999
>> coth([0, 1, -1, 2, 3+0.5i])
ans =
    inf          1.3130   -1.3130    1.0373    1.0027 - 0.0042i
>> sech([0, 1, -1, 0.7, 5])
ans =
    1.0000    0.6481    0.6481    0.7967    0.0135
>> csch([0, 1, -1, 2, 3+0.5i])
ans =
    inf          0.8509   -0.8509    0.2757    0.0874 - 0.0480i
```

5.1.5. Обратные гиперболические функции

Обратные гиперболические функции $\operatorname{arsh} x$, $\operatorname{arch} x$, $\operatorname{arth} x$, $\operatorname{arcth} x$, $\operatorname{arsch} x$, $\operatorname{arcsch} x$ представляются в системе MATLAB в следующем виде:

$\operatorname{asinh}(x)$	$\operatorname{atanh}(x)$	$\operatorname{asech}(x)$
$\operatorname{acosh}(x)$	$\operatorname{acoth}(x)$	$\operatorname{acsch}(x)$

Обратные гиперболические функции выражаются через логарифмические и гиперболические функции и имеют вид:

$$\operatorname{arsh} x = \ln \left(\sqrt{x^2 + 1} + x \right),$$

$$\operatorname{arch}(x) = \operatorname{iarccos}(x),$$

$$\operatorname{arth}(x) = -\operatorname{iarctan}(ix),$$

$$\operatorname{arcth}(x) = -\operatorname{iarccot}(-ix),$$

$$\operatorname{arsch}(x) = \operatorname{iarccos}(1/x),$$

$$\operatorname{arcsch} x = \ln \frac{\sqrt{x^2 + 1} \operatorname{sh}(x) + 1}{x}.$$

Аргумент x может быть числом вещественным и комплексным, вектором и матрицей.

Если x представлен в виде вектора или матрицы, то результат вычислений также представляется в виде вектора или матрицы.

Пример 5.9

```
>> asinh([0, 1, -3, 2+3i])
ans =
    0          0.8814        -1.8184        1.9686 + 0.9647i
>> acosh([0, 1, -3, 2+3i])
ans =
    0 + 1.5708i    0          1.7627 + 3.1416i    1.9834 + 1.0001i
>> atanh([0, 1, -3, 2+3i])
ans =
    0    NaN + NaNi    -0.3466 + 1.5708i    0.1469 + 1.3390i
>> acoth([0, 1, -3, 2+3i])
ans =
    0 + 1.5708i    NaN + NaNi    -0.3466    0.1469 - 0.2318i
>> asech([0, 1, -3, 2+3i])
ans =
    inf    0    0 + 1.9106i    0.2313 - 1.4204i
>> acsc([0, 1, -3, 2+3i])
ans =
    NaN + NaNi    0.8814    -0.3275    0.1574 - 0.2300i
```

5.1.6. Функции комплексного аргумента

Пусть комплексное число представлено в виде:

$$z = a + bi.$$

Основными функциями комплексного аргумента в MATLAB являются:

<code>abs(z)</code>	<code>imag(z)</code>	<code>phase(z)</code>
<code>real(z)</code>	<code>conj(z)</code>	

Функция `abs(z)` вычисляет модуль комплексного числа:

$$M = \sqrt{a^2 + b^2}.$$

Откликами функций `real(z)` и `imag(z)` являются соответственно вещественная и мнимая часть комплексного числа.

Функция `conj(z)` определяет комплексно-сопряженное число аргумента `z`.

Функция `phase(z)` вычисляет фазу комплексного числа

$$\varphi = \operatorname{arctg} \frac{a}{b}.$$

Аргумент `z` может быть числом, вектором или матрицей. Если `z` является вектором или матрицей, то функция комплексного аргумента также выдает вектор или матрицу, элементы которой являются реализациями соответствующей функции.

Пример 5.10

```
>> abs([2, 3+2i, 5i, -1+i])
ans =
     2     3.6056     5     1.4142
>> real([-3+2i, 2-3i])
ans =
    -3     2
>> imag([-3+2i, 2-3i])
ans =
     2    -3
>> conj([-3+2i, 2-3i])
ans =
   -3.0000-2.0000i    2.0000+3.0000i
>> phase([-3+2i, 2-3i])
ans =
    2.5536    5.3004
```

5.2. Специальные математические функции

Специальные математические функции часто приходится использовать при решении широкого класса научных и практических задач. Это решение дифференциальных уравнений специального вида, вычисление интегралов, задачи вероятностного характера и многое другое. Существует большое число специальных функций. Их подробное описание приводится в книгах [9, 15, 16]. В системе MATLAB реализованы не все функции. Из имеющихся мы ограничимся изучением только тех, которые наиболее часто могут использоваться студентами вузов. Технология вычисления специальных функций в системе MATLAB практически не отличается от технологии вычисления элементарных функций. Пользователь вводит имя функции и значения аргументов. Нажимает клавишу <Enter> и получает значение специальной функции. При этом аргументом может быть число, вектор или матрица. Если аргументом является вектор или матрица, то откликом также будет вектор или матрица того же размера. При этом функция вычисляется для каждого элемента вектора или матрицы.

5.2.1. Гамма-функция

Гамма-функция имеет много интегральных представлений. Вот одно из них:

$$\Gamma(n) = \int_0^x e^{-t} t^{n-1} dt.$$

Это представление справедливо при целом n . Гамма-функция в этом случае отождествляется с факториалом целого числа, при этом справедливыми являются следующие выражения:

$$\Gamma(n+1) = n\Gamma(n),$$

$$\Gamma(1) = \Gamma(2) = 1,$$

$$\Gamma(n) = (n-1)!,$$

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi},$$

$$\Gamma\left(-\frac{1}{2}\right) = -2\sqrt{\pi},$$

$$\Gamma\left(n + \frac{1}{2}\right) = \left(n - \frac{1}{2}\right)!.$$

Гамма-функция существует для случая n целого и дробного, положительного и отрицательного, действительного и комплексного.

Гамма-функция в MATLAB имеет представление для случая только действительного положительного n .

Синтаксис гамма-функции таков:

`gamma(n)`,

где n — действительное положительное число.

Пример 5.11

Необходимо вычислить гамма-функцию вектора $[0, 1, 2, 6, -3, 4.2]$.

Решение:

```
>> n=[0, 1, 2, 6, -3, 4.2])
ans =
    inf    1.0000    1.0000   120.0000    inf    7.7567
```

Из примера видно, что гамма-функцию чисел 0 и -3 программа не вычисляет (в результате получается `inf` — бесконечность). Следует иметь в виду, что при вычислении факториала целого числа

`gamma(n)=(n-1)!`

Если, например, необходимо вычислить $5!$, то функция `gamma()` будет иметь вид: `gamma(6)`.

В системе MATLAB гамма-функция имеет следующие два варианта:

```
gammainc(x, n)
gammaln(n)
```

Функция `gammainc(x, n)` вычисляет неполную гамма-функцию элементов `x`, `n`.

$$F(x, n) = \frac{1}{\Gamma(n)} \int_0^x e^{-t} t^{n-1} dt.$$

Элементы `x` и `n` должны быть вещественными и положительными.

Функция `gammaln(n)` возвращает логарифм числа `gamma(n)`.

Пример 5.12

```
>> gammainc(3, 2)
ans =
    0.8009

>> gammaln([2, 3.5, 7.3, 15])
ans =
    0    1.2010    7.1479   25.1912

>> gammaln([2, 3; 7.3, 1; 4, 12.5])
ans =
    0          0.6931
    7.1479     0
    1.7918    18.7343
```

5.2.2. Бета-функция (Эйлеров интеграл первого рода)

Бета-функция представляется в виде следующего интеграла:

$$B(x, y) = \int_0^1 t^{x-1} (1-t)^{y-1} dt.$$

Она имеет большое число различных интегральных представлений. При практических расчетах наиболее часто бета-функция вычисляется через гамма-функции:

$$B(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}.$$

Система MATLAB имеет следующие варианты этой функции:

```
beta(x, y)
betainc(z, x, y)
betaln(x, y)
```

Функция `beta(x, y)` возвращает бета-функцию при положительных значениях x, y .

Функция `betainc(z, x, y)` возвращает неполную бета-функцию действительных аргументов x, y . Аргумент z должен быть в интервале $[0; 1]$.

Функция `betaln(x, y)` вычисляет натуральный логарифм бета-функции.

Пример 5.13

```
>> beta(2, 4)
ans =
    0.0500
>> betainc(0.5, 2, 4)
ans =
    0.0125
>> betaln(2, 4)
ans =
   -2.9957
```

5.2.3. Функции ошибок

Система MATLAB имеет следующие варианты функций ошибок:

◆ `erf(x)` — функция ошибок, определяемая по выражению:

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt;$$

- ◆ $\text{erfc}(x)$ — остаточная функция ошибок, определяемая выражением:

$$\text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt;$$

- ◆ $\text{erfcx}(x)$ — масштабированная функция ошибок, вычисляемая по следующему выражению:

$$\text{erfcx}(x) = e^{x^2} \text{erfc}(x);$$

- ◆ $\text{erfinv}(x)$ — вычисляет обратную функцию ошибок при условии $-1 < x < 1$.

Аргумент x в функциях ошибок может быть числом, вектором и матрицей. Элементами вектора и матрицы являются действительные числа (положительные и отрицательные).

Пример 5.14

Пусть функция x является вектором $x=[2 \ -0.3 \ 4.5 \ 0.6]$. Необходимо определить все виды ошибок.

Решение и его результаты будут иметь вид:

```
>> erf(x)
ans =
    0.9953   -0.3286    1.0000    0.6039
>> erfc(x)
ans =
    0.0047    1.3286    0.0000    0.3961
>> erfcx(x)
ans =
    0.2554    1.4537    0.1225    0.5678
>> erfinv(x)
ans =
    NaN   -0.2725    NaN    0.5951
```

В последнем случае решения нет (NaN) для значений $x=2$ и $x=4.5$. Эти числа находятся вне диапазона допустимых значений.

5.2.4. Интегральная показательная функция

Существует большое число интегральных показательных функций. В MATLAB реализована только одна из них, которая имеет вид:

$$E(x) = \int_x^{\infty} \frac{e^{-t}}{t} dt.$$

В системе MATLAB эта функция представляется так:

```
expint(x)
```

где x — число, вектор или матрица, элементами которых могут быть положительные и отрицательные, действительные и комплексные числа.

Пример 5.15

```
>> expint([1, 2, -3, 6.2, 2+3i])
ans =
    0.2194    0.0489   -9.9338-3.1416i    0.0003    0.0248 + 0.0203i
>> expint([1.2, 3; 2, 7.5; 1-2i, 0.7])
ans =
    0.1564            0.0130
    0.0489            0.0001
   -0.1268 + 0.0351i    0.3738
```

5.2.5. Функции Эйри

Функция Эйри является решением дифференциального уравнения второго порядка

$$\frac{d^2 y}{dx^2} - xy = 0.$$

Существуют два вида функции Эйри — первого и второго рода.

Функция первого рода $Ai(x, n)$ при $x = 0$ имеет вид:

$$Ai(0, n) = \frac{3^{5/6} \times \left(\frac{1}{3}\right)!}{2\pi},$$

в противном случае

$$Ai(x, n) = \frac{C \times 3^{-\frac{1}{6}} - D \times 3^{\frac{1}{6}}}{2\pi},$$

где:

$$C = \sum_{k=0}^n \frac{3^k x^{3k} \left(k - \frac{2}{3}\right)!}{(3k)!},$$

$$D = x \sum_{k=0}^n \frac{3^k x^{3k} \left(k - \frac{1}{3}\right)!}{(3k+1)!}.$$

Функция второго рода $Bi(x, n)$ при $x = 0$ имеет вид:

$$Bi(0, n) = \frac{3^{4/3} \times \left(\frac{1}{3}\right)!}{2\pi},$$

в противном случае

$$Bi(x, n) = \frac{C \times 3^{1/3} + D \times 3^{2/3}}{2\pi},$$

где:

$$C = \sum_{k=0}^n \frac{3^k x^{3k} \left(1 - \frac{2}{3}\right)!}{(3k)!},$$

$$D = x \sum_{k=0}^n \frac{3^k x^{3k} \left(k - \frac{1}{3}\right)!}{(3k+1)!}.$$

В системе MATLAB функции Эйри представляются в следующем виде:

- ◆ `airy(z)` — функция Эйри первого рода;
- ◆ `airy(k, z)` — функция Эйри второго рода.

Аргументы функции имеют значения:

- ◆ `z` — число, вектор или матрица, элементы которых являются положительными или отрицательными, действительными или комплексными числами;
- ◆ `k` — определяет следующие варианты результатов:
 - `k=0` выдает результат тот же, что и функция `airy(z)`;
 - `k=1` выдает производную функции Эйри первого рода;
 - `k=2` выдает функцию Эйри второго рода;
 - `k=3` выдает производную функции Эйри второго рода.

В системе MATLAB функции Эйри вычисляются при одном фиксированном значении суммы ряда n .

Пример 5.16

```
>> y=[0, 3.2, -3.2, 2+3i];
>> airy(y)
ans =
    0.3550    0.0046    0.4174 + 0.0000i    0.0081 + 0.1312i
>> airy(0,y)
ans =
    0.3550    0.0046    0.4174 + 0.0000i    0.0081 + 0.1312i
>> airy(1,y)
ans =
   -0.2500   -0.0085    0.0650 - 0.0000i    0.0967 - 0.2320i
>> airy(2,y)
ans =
    0.6149   19.5870   -0.0539 - 0.0000i   -0.3964 - 0.5697i
>> airy(3,y)
ans =
    0.4483   33.2577   -0.7541 + 0.0000i    0.3495 - 1.1053i
```

5.2.6. Функции Лежандра

Функция Лежандра имеет вид:

$$P_n^m(x) = (-1)^m \sqrt{(1-x^2)^m} \frac{d^m P_n(x)}{dx^m},$$

где

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n (x^2 - 1)}{dx^n}.$$

Вычисление функции в системе MATLAB осуществляется с помощью встроенной функции, имеющей вид:

`legendre(n, x)`,

где:

- ♦ `n` — целое число, не превосходящее 256;
- ♦ `x` — аргумент, значение которого удовлетворяет условию $-1 < x < 1$.

Функция возвращает массив чисел размерности $n + 1$ для каждого аргумента `x`.

Аргумент `x` может быть числом или вектором.

Пример 5.17

```
>> legendre(3, 0.5)
ans =
    -0.4375
    -0.3248
     5.6250
    -9.7428

>> legendre(3, [-0.5, 0.2, 0.1])
ans =
    0.4375    -0.2800    -0.1475
   -0.3248     1.1758     1.4179
   -5.6250     2.8800     1.4850
   -9.7428   -14.1091   -14.7756
```

5.2.7. Функции Бесселя

Функции Бесселя являются решением следующего дифференциального уравнения:

$$Z^2 \frac{d^2 y(z)}{dz^2} + Z \frac{dy(z)}{dz} + (Z^2 - L^2)y(x) = 0,$$

где L — неотрицательное значение постоянной.

Функции $J_L(x)$ и $J_L(z)$ образуют множество решений дифференциального уравнения. Эти решения имеют вид:

♦ функция Бесселя первого рода

$$J_L(z) = \left(\frac{z}{2}\right)^L \sum_{k=0}^{\infty} \frac{\left(-\frac{z^2}{4}\right)^k}{k! \Gamma(L+k+1)},$$

где $\Gamma(L+k+1)$ — гамма-функция;

♦ функция Бесселя второго рода

$$Y_v(z) = \frac{J_L(z) \cos(L\pi) - J_{-L}^{(z)}}{\sin(L\pi)};$$

♦ функции Бесселя третьего рода можно вычислить, используя функции первого и второго рода:

$$H_L^{(1)}(z) = J_L(z) + iY_L(z),$$

$$H_L^{(2)}(z) = J_L(z) - iY_L(z).$$

В системе MATLAB функции Бесселя определяются с помощью следующих встроенных функций:

♦ `besselj(n, z)` — определяет функцию Бесселя первого рода;

♦ `bessely(n, z)` — определяет функцию Бесселя второго рода.

В выражениях z — массив чисел, которые могут быть вещественными и комплексными, n — порядок массива, является числом вещественным и положительным.

Решение выдается для каждого элемента массива z .

Если z положительно, то результат будет вещественным.

Пример 5.18

```
>> A=[2, 1,5];
>> B=[1-2i, 3,5];
>> besselj(A,B)
ans =
    -0.2611 - 0.7623i    0.3391    0.2611
>> bessely(A,B)
ans =
    -0.7512    0.1240i    0.3247   -0.4537
```

Существуют другие функции Бесселя и их модификации. С ними можно познакомиться в книгах [9, 15, 16].

5.3. Функции пользователя

Решение практических задач, как правило, связано с вычислениями по одним и тем же алгоритмам при различных исходных данных. В таких ситуациях полезно иметь в памяти компьютера функции пользователя, обращаясь к которым получать решение, подобно тому, как мы обращаемся к элементарным или специальным функциям. Создание функции пользователя осуществляется так, как описано ниже.

1. Вызов окна редактора m -файлов путем нажатия кнопки **New M-File** (Создать m -файл).
2. Ввод строки

```
function Z = expxp(x)
```

Ключевое слово `function` объявляет новую функцию, имя которой `expxp`, а ее параметр — x . Символ Z определяет значение функции при аргументе x . На экране образуется введенное выражение.

3. Задание новой функции (функции пользователя). Пусть

```
Z = exp(x)/x
```

Наберем это выражение на клавиатуре. Оно отобразится на экране окна редактора m-файлов.

4. Сохранение функции пользователя осуществляется нажатием кнопки **Save** (Сохранить) на панели инструментов. В результате появится новое окно. В поле **File name** (Имя файла) отобразится имя созданной функции `exrpr`. Для сохранения функции на диске достаточно щелкнуть мышью по кнопке **Save** (Сохранить).

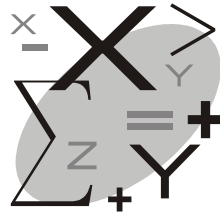
5. Закрытие окна редактора m-файлов.

Функция пользователя $z = \exp(x) / x$ создана.

Для вычисления функции при данном аргументе x достаточно набрать имя функции и значение аргумента в круглых скобках: $z = \text{exrpr}(1)$. На экране получим значение функции $z = \exp(1) / 1$.

* * *

Система MATLAB имеет ряд функций, не относящихся к элементарным, специальным и функциям пользователя. К ним принадлежат функции поразрядной обработки, обработки множеств, времени и даты. Они достаточно подробно описаны в [9].



ГЛАВА 6

Алгебра векторов и матриц

6.1. Создание векторов и матриц

Вектор или матрица состоят из имени и элементов, заключенных в квадратные скобки. Элементы вектора отделяются запятыми или пробелами. Элементами вектора могут быть числа положительные и отрицательные, действительные и комплексные.

Пример 6.1

```
>> v=[1,2,-3,-7,12]
>> v=[3 2+3i 1-2i -5]
```

Для вывода вектора на экран нажимается клавиша <Enter>. Откликом будут элементы вектора без квадратных скобок, отделенные друг от друга пробелами. Для нашего примера они будут иметь вид:

```
v =
    1     2    -3    -7    12
v =
    3.0000    2.0000 + 3.0000i    1.0000 - 2.0000i    -5.0000
```

Элементами матрицы, так же как вектора, могут быть числа положительные и отрицательные, действительные и комплексные. В строках матрицы они отделяются запятыми или пробелами, а строки отделяются точкой с запятой (;).

Пример 6.2

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

После нажатия клавиши <Enter> на экране появится следующая матрица:

M =

1	-2	3
2	3	-4
-3	4	5

```
>> M=[7-2i,1+i,12;1,2,7;i,2,-i]
```

А теперь матрица выглядит так:

M =

7.0000 - 2.0000i	1.0000 + 1.0000i	12.0000
1.0000	2.0000	7.0000
1.0000i	2.0000	-1.0000i

Если элементы вектора или матрицы являются числами, отличающимися друг от друга на постоянный шаг, то вектор или матрицу можно образовать проще.

Пример 6.3

```
>> V=[1:4]
```

V =

1	2	3	4
---	---	---	---

```
>> M=[1:3;2:4;7:9]
```

M =

1	2	3
2	3	4
7	8	9

Здесь решение получено для случая постоянного шага, равного 1.

При постоянном шаге, отличном от единицы, процедуры образования вектора и матрицы и отклики имеют вид:

```
>> V=[1:0.2:2]
V =
    1.0000    1.2000    1.4000    1.6000    1.8000    2.0000
>> M=[1:0.2:1.8; 2:0.4:3.6; 1:5]
V =
    1.0000    1.2000    1.4000    1.6000    1.8000
    2.0000    2.4000    2.8000    3.2000    3.6000
    1.0000    2.0000    3.0000    4.0000    5.0000
```

6.2. Преобразование матриц

Система MATLAB позволяет:

- ◆ заменить элементы вектора или матрицы без их редактирования;
- ◆ изменить размер вектора или матрицы;
- ◆ преобразовать матрицу в иной вид;
- ◆ образовать матрицу специального вида.

6.2.1. Вызов на экран и замена элементов матрицы

Для вызова на экран элементов вектора или матрицы достаточно указать их имя и координаты в круглых скобках (номер строки и номер столбца вектора).

Пример 6.4

```
>> V=[1 2 3 7 12];
>> V(4)
V =
    7
>> M=[1,3,7; 2 6 12; -4 8 3];
>> M(1,3)
M =
    7
```

Для замены элемента необходимо указать имя элемента или матрицы, его координаты и присвоить этому имени новое значение

элемента. После нажатия клавиши <Enter> на экране будет отображен вектор или матрица с новым значением элемента.

Пример 6.5

Пусть вектор и матрица — те же, что и в предыдущем примере. Заменим третий элемент вектора (со значением 3) на 12, а элемент матрицы, находящийся во второй строке и третьем столбце (со значением 12), — на -7.

Решение:

```
>> V(3) = 12
V(3) =
    1     2     12     7     12
>> M(2,3) = -7
M =
    1     3     7
    2     6    -7
```

6.2.2. Изменение размера вектора или матрицы

Изменение размера вектора проще всего осуществить путем его редактирования. Изменение размера матрицы легче выполнить посредством удаления или добавления строк и столбцов матрицы.

Удаление строки или столбца осуществляется с помощью знака двоеточия (:), который ставится в круглых скобках после имени матрицы:

- ◆ $M(:, n)$ — удалить строку n ;
- ◆ $M(m, :)$ — удалить столбец m .

Пример 6.6

Пусть матрица имеет вид:

```
M =
    1     2     3
```

```

2      7      4
3      12     -7

```

Необходимо удалить вторую строку и третий столбец.

Решение будет иметь вид:

```

>> M(:,2)
M =
      1      2      3
      3     12     -7
>> M(3,:)
M =
      1      2
      3     12

```

Увеличить размер матрицы можно посредством объединения малых матриц в большую. Эта процедура называется *конкатенацией*. Она осуществляется путем образования матрицы из имен малых матриц. При этом допускаются алгебраические операции над именами. Рассмотрим эти методы на примерах.

Пример 6.7

Пусть имеется три следующих вектора:

```
v1=[1 2 3], v2=[3 -2 1], v3=[7 6 2].
```

Образуем матрицу из этих векторов. Векторы следует рассматривать как элементы матрицы. Тогда получим:

```
>> M=[v1; v2; v3]
```

На экране матрица из векторов v1, v2, v3:

```

M =
      1      2      3
      3     -2      1
      7      6      2

```

Выполним теперь операцию конкатенации. Создадим из полученной матрицы матрицу размером 6×6. Для этого образуем три новых матрицы: M+3, M-5 и M*2.

Процедуры имеют вид:

```
>> Z=[M, M+3; M-5, M*2]
```

```
Z =
```

```

     1     2     3     4     5     6
     3    -2     1     6     1     4
     7     6     2    10     9     5
    -4    -3    -2     2     4     6
    -2    -7    -4     6    -4     2
     2     1    -3    14    12     4
```

6.2.3. Математические операции с векторами и матрицами

Определитель матрицы

Определитель матрицы вычисляется с помощью функции

```
det (M)
```

где M — матрица, элементами которой могут быть вещественные и комплексные числа.

Пример 6.8

```

>> M=[2 3 -1;1 -6 2;1 3 5];
>> det (M)
ans =
    -90

>> M=[1+2i, 3, -2.5; i, -1, 5; 3, 5, 0];
>> det (M)
ans =
    12.5000 - 62.5000i
```

Рассмотрим функции системы MATLAB, позволяющие преобразовывать векторы и матрицы, создавать новые матрицы, выполнять математические операции над элементами векторов и матриц. При практических расчетах такие действия бывают необходимы, если расчеты сводятся к матричным операциям.

Транспонирование матрицы

Транспонированной называется матрица, у которой строки стали столбцами, а столбцы строками исходной матрицы.

Транспонирование осуществляется следующим представлением исходной матрицы: M' , где M — исходная матрица.

Пример 6.9

Пусть исходная матрица имеет вид:

$$M = \begin{bmatrix} 1 & 2 & 7 \\ 3 & -4 & 2 \\ 5 & 1 & -5 \end{bmatrix}.$$

Получим транспонированную матрицу:

```
>> M=[1 2 7; 3 -4 2; 5 1 -5];
>> Z=M'
```

```
Z =
     1     3     5
     2    -4     1
     7     2    -5
```

След матрицы

Следом матрицы называется сумма ее диагональных элементов. Вычисляется с помощью функции `trace()`, которая имеет вид:

```
trace(M)
```

где M — матрица.

Пример 6.10

Пусть матрица имеет вид: $M=[2 \ 6 \ -1; \ 2 \ 4 \ 8; \ 1 \ -2 \ 3]$.

Ее диагональными элементами являются 2, 4, 3, а их сумма равна 9.

Решение имеет вид:

```
>> M=[2, 6, -1; 2, 4, 8; 1, -2, 3];  
>> Z=trace(M)  
Z =  
9
```

Обратная матрица

Обратной называется матрица, полученная в результате деления единичной матрицы E на исходную:

$$M^{-1}=E/M$$

Получают обратную матрицу с помощью функции `inv()`, имеющей вид:

```
inv(M)
```

где M — исходная квадратичная матрица.

Пример 6.11

```
>> M=[1, -1, 3; 2, 11, 7; -3, 5, 4];  
>> Z=inv(M)  
Z =  
0.0539    0.1138   -0.2395  
-0.1737    0.0778   -0.0060  
0.2575   -0.0120    0.0778
```

Единичная матрица

Функциями создания *единичной матрицы* являются:

- ◆ `eye(n)` — определяет единичную матрицу размером $n \times n$;
- ◆ `eye(m,n)` — определяет единичную матрицу размером $m \times n$ с единицами в диагонали и с нулями в остальных элементах матрицы;
- ◆ `eye(size(M))` — определяет единичную матрицу с тем же размером, что и матрица M .

Пример 6.12

```
>> M=eye(3)
M =
     1     0     0
     0     1     0
     0     0     1

>> M=eye(3,4)
M =
     1     0     0     0
     0     1     0     0
     0     0     1     0

>> M=[1 1 2 3; 2, 2, -3, 8; 0, 1, 3, 0; 1, 2, 3, 4];
>> Z=eye(size(M))
Z =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

Образование матрицы с единичными элементами

Матрица с единичными элементами реализуется следующими функциями:

- ◆ `ones(n)` — образует матрицу размером $n \times n$, все элементы которой равны единице;
- ◆ `ones(m,n)` — образует единичную матрицу размером $m \times n$;
- ◆ `ones(size(M))` — образует единичную матрицу такого же размера, как и матрица `M`.

Пример 6.13

```
>> M=ones(3)
M =
     1     1     1
     1     1     1
     1     1     1
```

```
>> M=ones(3,4)
M =
     1     1     1     1
     1     1     1     1
     1     1     1     1
>> M=[3, 2, 1, 7; 6, 1, -2, 4];
>> Z= ones(size(M))
Z =
     1     1     1     1
     1     1     1     1
```

Образование матрицы с нулевыми элементами

Матрицы с нулевыми элементами формируются следующими функциями:

- ◆ `zeros(n)` — создает матрицу размером $n \times n$ с нулевыми элементами;
- ◆ `zeros(m,n)` — образует матрицу размером $m \times n$ с нулевыми элементами;
- ◆ `zeros(size(M))` — возвращает матрицу с нулевыми элементами того же размера, что и матрица `M`.

Пример 6.14

```
>> M=zeros(3)
M =
     0     0     0
     0     0     0
     0     0     0
>> M=zeros(3,4)
M =
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> M=[1, 2, 3; 2, 3, 4; 3, 4, 5];
>> M=zeros(size(M))
M =
     0     0     0
     0     0     0
     0     0     0
```

Вектор равноотстоящих точек

Вектор равноотстоящих точек формирует массив точек в диапазоне $[a; b]$. Реализуется следующими функциями:

- ◆ `linspace(a,b)` — создает массив из 100 точек, распределенных равномерно в диапазоне $[a; b]$;
- ◆ `linspace(a,b,n)` — создает массив из n точек, равномерно распределенных в диапазоне $[a; b]$.

Пример 6.15

```
>> R=linspace(1,2)
R =
    1.0000    1.0101    1.0202.....1.9899    2.0000
>> R=linspace(1,10,5)
R =
    1.0000    3.2500    5.5000    7.7500   10.0000
```

Перестановка элементов матрицы

Перестановка столбцов и строк матрицы осуществляется с помощью следующих функций:

- ◆ `fliplr(M)` — осуществляет перестановку столбцов матрицы M относительно вертикальной оси;
- ◆ `flipud(M)` — осуществляет перестановку строк матрицы относительно горизонтальной оси.

Пример 6.16

```
>> M=[1, 2, 3, 4; 3, 1, 4, 2]
M =
     1     2     3     4
     3     1     4     2
>> Z=fliplr(M)
Z =
     4     3     2     1
     2     4     1     3
```

```
>> W=flipud(M)
```

```
W =
```

```
    3    1    4    2
    1    2    3    4
```

В MATLAB имеется также функция перестановки элементов, которая из вектора V , состоящего из n элементов, образует матрицу, состоящую из $n!$ столбцов, т. е. число строк соответствует всем перестановкам элементов вектора V . Функция имеет вид:

```
perms(V)
```

где V — вектор, состоящий из n элементов.

Пример 6.17

```
>> V=[2,1,7];
```

```
>> P=perms(V)
```

```
P =
```

```
    7    1    2
    7    2    1
    1    7    2
    1    2    7
    2    1    7
    2    7    1
```

Создание матрицы с заданной диагональю

Система MATLAB имеет функции, позволяющие из элементов вектора V создавать матрицу с заданной диагональю.

Таковыми функциями являются:

- ◆ $M=\text{diag}(V,K)$ — образует матрицу M , в одной из диагоналей которой находятся элементы вектора V . При $K=0$ элементы вектора находятся в главной диагонали, при $K>0$ — в K -й верхней диагонали, при $K<0$ — в K -й нижней диагонали. Остальные элементы квадратной матрицы M являются нулями;
- ◆ $M=\text{diag}(V)$ — образуется квадратная матрица, в которой на главной диагонали стоят элементы вектора V ;
- ◆ $V=\text{diag}(M,K)$ — образует вектор-столбец V из K -й диагонали матрицы M ;

- ◆ `V=diag(M)` — возвращает главную диагональ матрицы `M` в виде вектора `V`.

Пример 6.18

```
>> V=[1, 0, 2, -7];
>> M=diag(V,0)
M =
    1     0     0     0
    0     0     0     0
    0     0     2     0
    0     0     0    -7

>> Z=diag(V, -2)
Z =
    0     0     0     0     0     0
    0     0     0     0     0     0
    1     0     0     0     0     0
    0     0     0     0     0     0
    0     0     2     0     0     0
    0     0     0    -7     0     0

>> V=diag(M,0)
V =
    1
    0
    2
   -7

>> M=[1, 2, -3, 5; -2, 6, 4, 7; 1, 0, 4, -2];
>> V=diag(M,1)
V =
    2
    4
   -2

>> v=diag(M)
v =
    1
    6
    4
```

Создание массивов со случайными элементами

Генерирование случайных чисел в системе MATLAB осуществляется с помощью функции, которая создает случайные числа, равномерно распределенные на интервале $[0; 1]$. Функция имеет следующие варианты:

- ◆ `rand(n)` — создает матрицу случайных чисел размером $n \times n$;
- ◆ `rand(m,n)` — создает матрицу случайных чисел размером $m \times n$;
- ◆ `rand(m,n,p,...)` — формирует массив случайных чисел с нормальным законом распределения;
- ◆ `rand(size(A))` — генерирует массив случайных чисел размера A ;
- ◆ `rand` — генерирует единственное случайное число, распределенное по равномерному закону, нажимая многократно клавиши $\langle \uparrow \rangle$ (стек функции `rand()`) и $\langle \text{Enter} \rangle$, получим множество случайных чисел из диапазона $[0; 1]$, распределенных по равномерному закону;
- ◆ `rand('state')` — генерирует вектор случайных чисел, состоящий из 35 значений, с равномерным законом распределения.

Состояние генератора случайных чисел можно менять. Для этого существуют следующие варианты функции:

- ◆ `rand('state',0)` — генератор устанавливается в исходное начальное состояние;
- ◆ `rand('state',s)` — генератор устанавливается в некоторое состояние s .

Пример 6.19

```
>> Z=rand(4)
Z =
    0.4449    0.9568    0.9797    0.7373
    0.6946    0.5226    0.2714    0.1365
    0.6213    0.8801    0.2523    0.0118
    0.7948    0.1730    0.8757    0.8929
>> M=[1, 2, 3; 2, 4, 1];
>> Z=rand(size(M))
```

Z =

0.6614	0.4692	0.9883
0.2844	0.0648	0.5828

Покажем график, образованный случайными числами. Для этого сформируем координаты точек на плоскости в виде матрицы случайных чисел с большим числом строк и одним столбцом по осям x и y .

Программа будет иметь следующий вид:

```
>> X=rand(800, 1);  
>> Y=rand(800, 1);  
>> plot(X, Y, '.')
```

График показан на рис. 6.1.

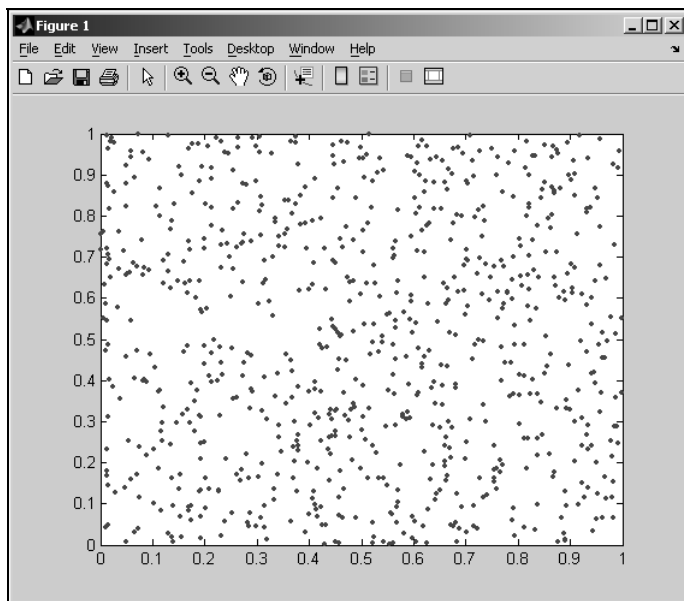


Рис. 6.1. Точки со случайными значениями координат x и y , распределенными по равномерному закону

Система MATLAB имеет датчик случайных чисел с нормальным законом распределения, с математическим ожиданием, равным

нулю, и среднеквадратическим отклонением, равным 1. Генерирование случайных чисел осуществляется с помощью следующих функций:

- ◆ `randn` — генерирует одно случайное число, нажимая последовательно клавиши `<↑>` и `<Enter>`, можно получить семейство случайных чисел;
- ◆ `randn(n)` — матрица случайных чисел размером $n \times n$;
- ◆ `randn(m,n)` — матрица случайных чисел размером $m \times n$;
- ◆ `randn(m,n,p,...)` — массив случайных чисел с нормальным законом распределения;
- ◆ `randn(size(B))` — генерирует случайные числа размером B ; B может быть вектором или матрицей;
- ◆ `randn('state')` — возвращает вектор с двумя элементами при данном состоянии генератора случайных чисел;
- ◆ `randn('state',0)` — возвращает генератор в начальное (нулевое) состояние;
- ◆ `randn('state',s)` — устанавливает состояние генератора s .

Пример 6.20

```
>> Z=randn(2, 3)
Z =
    0.7853    0.7104    0.7073
    0.4353    0.9508    0.1381
```

Система позволяет по данным датчика случайных чисел построить график — гистограмму нормального распределения случайных чисел.

Программа имеет следующий вид:

```
>> Y=randn(n,1);
>> hist(Y,m)
```

где n — количество случайных чисел, m — число интервалов в диапазоне случайных чисел.

Пример 6.21

Построить гистограмму случайных чисел при $n = 20000$ и $m = 100$.

В этом случае программа будет иметь вид:

```
>> Y=randn(20000,1);  
>> hist(Y,100)
```

Ответ представлен на рис. 6.2.

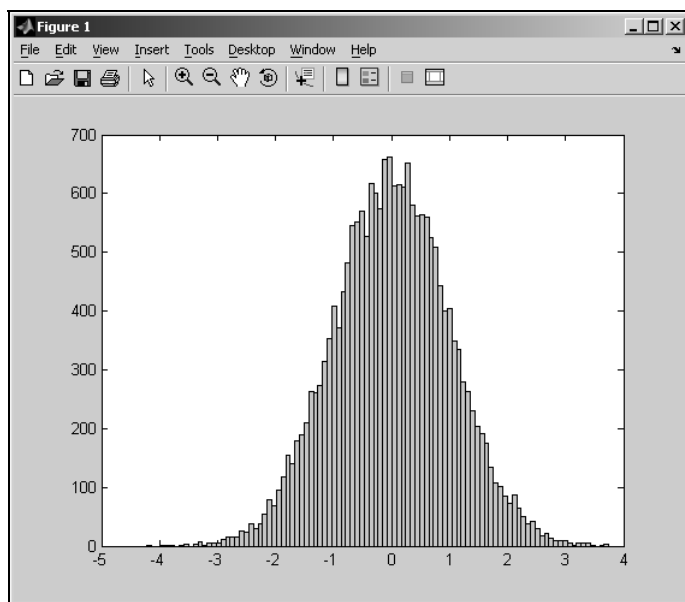


Рис. 6.2. График нормального закона распределения случайных чисел

Поворот матрицы

Поворот матрицы позволяет создать новую матрицу, у которой меняются не только значения элементов в строках и столбцах, но также размер матрицы.

Поворот матрицы осуществляет функция `rot()`, имеющая вид:

```
rot90(M, K)
```

где:

- ◆ M — матрица;
- ◆ K — число, указывающее на величину поворота матрицы в градусах, кратных 90° .

Если $K = 1$, то поворот осуществляется на 90° , при $K = 2$ — на 180° и т. д.

Поворот выполняется против часовой стрелки.

При $K = 1$ функция имеет вид:

`rot90(M)`

Пример 6.22

```
>> M=[1, 2, 3; 2, 3, 4; 3, 4, 5];
```

```
>> Z=rot90(M,2)
```

Z =

5 4 3

4 3 2

3 2 1

Выделение треугольных частей матрицы

Выделение треугольных частей матрицы выполняется с помощью следующих функций:

- ◆ `tril(M)` — создает нижнюю треугольную часть матрицы, остальные элементы являются нулями;
- ◆ `tril(M,K)` — создает нижнюю часть диагонали, начиная с K -й;
- ◆ `triu(M)` — создает верхнюю часть матрицы M ;
- ◆ `triu(M,K)` — создает верхнюю часть матрицы M , начиная с K -й диагонали.

Пример 6.23

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

M =

1 4 3

```
      7   2   2
      6   1   5
>> Z=tril(M)
Z =
      1   0   0
      7   2   0
      6   1   5
>> Z=tril(M,1)
Z =
      0   4   3
      0   0   2
      0   0   0
```

Вычисление магического квадрата

MATLAB имеет большое число функций, позволяющих образовывать специальные матрицы, матричные функции. В качестве примера приведем только одну из них.

Матрица, называемая *магическим квадратом*, представляет собой квадратную матрицу размером $n \times n$ ($n \geq 3$), у которой сумма строк, столбцов и главных диагоналей является одной и той же величиной.

Функция имеет вид:

```
magic(n)
```

где n — размер квадратной матрицы.

Пример 6.24

```
>> M=magic(3)
M =
      8   1   6
      3   5   7
      4   9   2
```

6.3. Математические операции над векторами и матрицами

Над векторами и матрицами можно выполнять практически все те операции, что и над числами: сложение и вычитание, умножение и деление, вычисление элементарных функций, таких как возведение в степень, извлечение квадратного корня, вычисление логарифма, вычисление тригонометрических функций. При этом матричными операторами являются почти все арифметические операторы (табл. 6.1).

Таблица 6.1. Таблица матричных операторов MATLAB

Функция	Название	Оператор	Синтаксис
plus	Плюс (сложение матриц)	+	M1+M2
minus	Минус (вычитание матриц)	-	M1-M2
times	Почленное умножение массивов чисел	.*	M1.*M2
mtimes	Матричное умножение	*	M1*M2
mpower	Возведение матрицы в степень	^	M1^X
power	Почленное возведение элементов матрицы в степень	.^	M1.^X
mzdivide	Деление матриц слева направо	/	M1/M2
mldivide	Обратное деление матриц	\	M1\M2
rdivide	Почленное деление элементов матрицы слева направо	./	M1./M2
ldivide	Почленное деление элементов матрицы справа налево	.\	M1.\M2

Приведем примеры выполнения матричных операций.

Даны следующие матрицы:

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 5 \\ 2 & 3 & 1 \\ 2 & 1 & 4 \end{bmatrix}, \quad \mathbf{N} = \begin{bmatrix} 2 & 0 & 3 \\ 1 & 5 & 4 \\ 3 & 3 & 1 \end{bmatrix}.$$

Воспользуемся матричными функциями, приведенными в табл. 6.1, и выполним операции, соответствующие этим функциям:

$$\text{plus}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} 3 & 2 & 8 \\ 3 & 8 & 5 \\ 5 & 4 & 5 \end{bmatrix} \quad \text{minus}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} -1 & 2 & 2 \\ 1 & -2 & -3 \\ -1 & -2 & 3 \end{bmatrix}$$

$$\text{mtimes}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} 19 & 25 & 16 \\ 10 & 18 & 19 \\ 17 & 17 & 14 \end{bmatrix} \quad \text{times}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} 2 & 0 & 15 \\ 2 & 15 & 4 \\ 6 & 3 & 4 \end{bmatrix}$$

$$\text{mpower}(\mathbf{M}, 2) = \begin{bmatrix} 15 & 13 & 27 \\ 10 & 14 & 17 \\ 12 & 11 & 27 \end{bmatrix} \quad \text{power}(\mathbf{M}, 2) = \begin{bmatrix} 1 & 4 & 25 \\ 4 & 9 & 1 \\ 4 & 1 & 16 \end{bmatrix}$$

$$\text{mrdivide}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} 0.9 & 0.7 & -0.5 \\ -0.14 & 0.18 & 0.7 \\ 1.02 & 0.26 & -0.1 \end{bmatrix}$$

$$\text{mldivide}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} 0.9524 & 2.5714 & -0.3810 \\ -0.4286 & 0.1429 & 1.5714 \\ 0.3810 & -0.5714 & 0.0476 \end{bmatrix}$$

$$\text{rdivide}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} 0.5 & \text{inf} & 1.667 \\ 2 & 0.6 & 0.25 \\ 0.6667 & 0.3333 & 4 \end{bmatrix}$$

$$\text{ldivide}(\mathbf{M}, \mathbf{N}) = \begin{bmatrix} 2 & 0 & 0.6 \\ 0.5 & 1.6667 & 4 \\ 1.5 & 3 & 0.25 \end{bmatrix}$$

Все эти действия будут успешно выполнены, если вместо функций воспользоваться соответствующими операторами.

Пример 6.25

```
>> M=[1 2 5; 2 3 1; 2 1 4];
>> N=[2 0 3; 1 5 4; 3 3 1];
```

$$\mathbf{M} \cdot \mathbf{N} = \begin{bmatrix} 2 & 0 & 15 \\ 2 & 15 & 4 \\ 6 & 3 & 4 \end{bmatrix}$$

$$\mathbf{M}^2 = \begin{bmatrix} 15 & 13 & 27 \\ 10 & 14 & 17 \\ 12 & 11 & 27 \end{bmatrix}$$

Аналогичные операции выполняются над векторами. Покажем это на примере.

Пример 6.26

Пусть даны два вектора:

```
>> v1=[1, 2, 4, 7];
>> v2=[-2, 3, 1, 5]
```

Процедуры вычислений и результаты будут иметь вид:

```
>> v1+v2
=
```

```
[-1 5 5 12]
```

```
v1-v2=[3 -1 3 2]
```

```
v1.*v2=[-2 6 4 35]
```

```
v1.^2=[1 4 16 49]
```

$$V1/V2=1.1026$$

$$V1 \setminus V2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -0.2857 & 0.4286 & 0.1429 & 0.7143 \end{bmatrix}$$

$$V1./V2=[-0.5000 \ 0.6667 \ 4.0000 \ 1.4000]$$

$$V1.\setminus V2=[-2.0000 \ 1.5000 \ 0.2500 \ 0.7143]$$

6.3.1. Примеры образования функций от векторов и матриц

Приведем примеры образования элементарных функций $\ln x$, e^x , $\sin x$ из вектора и матрицы.

Пример 6.27

Пусть вектор N имеет вид: $N=[1 \ 3 \ 5 \ 7 \ 9]$.

Образуем функции $\ln N$, e^N , $\sin N$.

Решение будет иметь вид:

```
>> N=[1, 3, 5, 7, 9];
>> Z=log(N)
Z =
    0    1.0986    1.6094    1.9459    2.1972
>> Z=exp(N)
Z =
    1.0e + 003
    0.0027    0.0201    0.1484    1.0966    8.1031
>> Z=sin(N)
Z =
    0.8415    0.1411   -0.9589    0.6570    0.4121
```

Пример 6.28

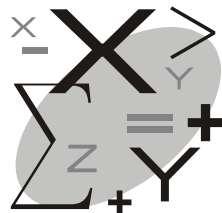
Пусть матрица имеет вид: $K=[1, \ 2, \ 7; \ 3, \ -2, \ 6]$.

Необходимо вычислить функции: $\ln K$, e^{-K} , $e^K + 2K + K^2$.

Процедуры ввода и результаты вычисления имеют вид:

```
>> K=[1, 2, 7; 3, -2, 6];  
>> Z=log(K)  
Z =  
    0          0.6931          1.19459  
    1.0986    0.6931 + 3.1416i    1.7918  
>> Z=exp(-K)  
Z =  
    0.3679    0.1353    0.0009  
    0.0498    7.3891    0.0025  
>> Z=exp(K)+2*K+K.^2  
Z =  
    1.0e + 003  
    0.0057    0.0154    1.1596  
    0.0351    0.0001    0.4514
```

ГЛАВА 7



Визуализация вычислений

Система MATLAB имеет богатые возможности графического представления информации. Она позволяет строить двухмерные и трехмерные графики функций, заданных в аналитическом виде, в виде векторов и матриц; дает возможность построения множества функций на одном графике; позволяет представлять графики разными цветами, типами точек и линий и в различных системах координат.

Система способна строить диаграммы, гистограммы и графики специальных функций.

Мы ограничимся рассмотрением лишь двухмерной и трехмерной графики универсальных математических функций.

Более детально с графической системой MATLAB можно ознакомиться по литературным источникам, например, [9].

7.1. Двухмерная графика

Основными функциями двухмерной графики являются:

```
plot(x, y)
plot(x, y, s)
plot(x1, y1, s1, x2, y2, s2, ..., xn, yn, sn)
```

где:

- ◆ x — аргумент функции, задаваемой в виде вектора;
- ◆ y — функция, представленная в аналитическом виде или в виде вектора или матрицы;
- ◆ s — вектор стилей графика; константа, определяющая цвет линий графика, тип точек и тип линии;
- ◆ x_1, x_2, \dots, x_n — аргументы n функций, изображаемых на одном графике;
- ◆ y_1, y_2, \dots, y_n — функции, изображаемые на одном графике.

Рассмотрим более подробно функции двухмерной графики и приведем примеры.

7.1.1. Функция $\text{plot}(x,y)$

Функция позволяет строить график при задании функции $y = f(x)$ в аналитическом виде, в виде вектора или матрицы. В математических расчетах находит широкое применение. Наиболее часто используется в следующих случаях:

- ◆ выбор области изоляции корня уравнения $f(x) = 0$;
- ◆ определение координат особых точек функции (максимумов, минимумов, точек перегиба, разрывов непрерывностей);
- ◆ проверка достоверности выбора функции интерполяции;
- ◆ качественная оценка точности представления функции степенным рядом.

Пример 7.1

Дана функция

$$y = 3^x - 9x + 6.$$

Определить область изоляции корня уравнения

$$3^x - 9x + 6 = 0$$

и другие особые точки функции.

Решение:

```
>> x = 0 : 0.1 : 3.5;  
>> y = 3.^x - 9 * x + 6;  
>> plot(x,y)
```

График функции приведен на рис. 7.1. Из рисунка видно, что функция имеет два корня и минимум. При этом областями изоляции корней может быть: $0.5 \leq x_1 \leq 1.5$, $2 \leq x_2 \leq 3$. Минимум функции расположен в области $1.5 \leq x \leq 2.5$.

Теперь можно искать корни уравнения и минимум функции.

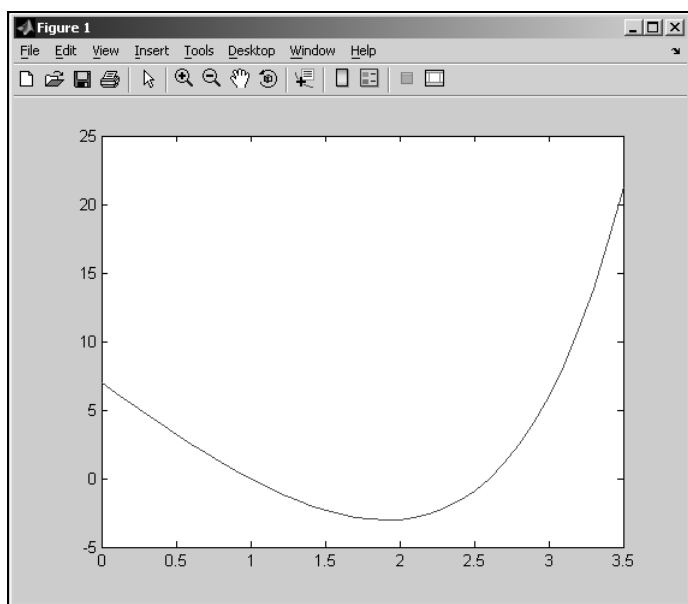


Рис. 7.1. График функции $y = 3^x - 9x + 6$

7.1.2. Функция *plot(x,y,s)*

Функция аналогична функции `plot(x,y)` и отличается лишь наличием вектора констант `s`, определяющего цвет линий графика, тип точек и линий функции, т. е. стиль графика. Стиль графика `s` можно не задавать.

В табл. 7.1 приведены стили графиков системы MATLAB.

Таблица 7.1. Стили графиков

Тип точки		Цвет линии		Тип линии	
•	Точка	Y	Желтый	–	Сплошная
О	Окружность	M	Фиолетовый	:	Двойной пунктир
×	Крест	C	Голубой	–.	Штрих-пунктир
+	Плюс	R	Красный	--	Штриховая
*	Звездочка	G	Зеленый		
S	Квадрат	B	Синий		
D	Ромб	W	Белый		
∨, ∧, <, >	Треугольник вверх, вниз, влево, вправо	K	Черный		
P	Пятиугольник				
H	Шестиугольник				

При задании стиля символ s представляется в виде вектора, элементами которого являются: тип точки, цвет линии и тип линии, разделенные запятыми и выделенные одиночными кавычками. Например:

```
plot(x, y, ['R','*','-'])
```

Это график красного цвета (R), точки графика в виде звездочек (*), линии штрихпунктирные (–.).

На рис. 7.2 показан график функции $y = 3^x - 9x + 6$, выполненный в этом стиле.

Пример 7.2

Астроном Хаббл обнаружил, что галактики удаляются от Земли тем быстрее, чем дальше они от нее расположены. Данные опыта Хаббла приведены в табл. 7.2.

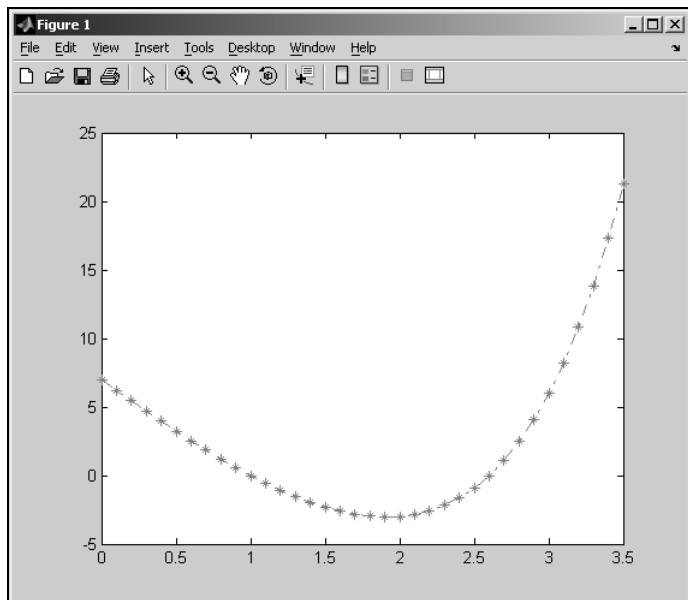


Рис. 7.2. График функции в стиле s

Таблица 7.2. Данные расширения Вселенной

Название галактики	R	V
Дева	22	7,5
Пегас	68	24
Персей	108	32
Волосы Вероники	137	47
Большая медведица 1	255	93
Лев	315	120
Северная корона	390	134
Близнецы	405	144
Волопас	685	245
Большая медведица 2	700	260
Гидра	1100	380

В таблице обозначены:

- ◆ R — расстояние галактики от Земли, миллионов световых лет;
- ◆ V — скорость удаления галактики, сотни миль в секунду.

Необходимо выбрать вид функции интерполяции с целью определения математической модели расширения Вселенной.

Решение задачи. Воспользуемся графоаналитическим методом. Представим функцию $V = f(R)$ в виде графика и по его виду подберем подходящую функцию. График создадим с помощью функции `plot(x,y,s)`.

Программа будет иметь вид:

```
>> x=[22, 68, 108, 137, 255, 315, 390, 405, 685, 700, 1100];  
>> y=[7.5, 24, 32, 47, 93, 120, 134, 144, 254, 260, 380];  
>> plot(x,y,['k','s','-.'])
```

График функции приведен на рис. 7.3.

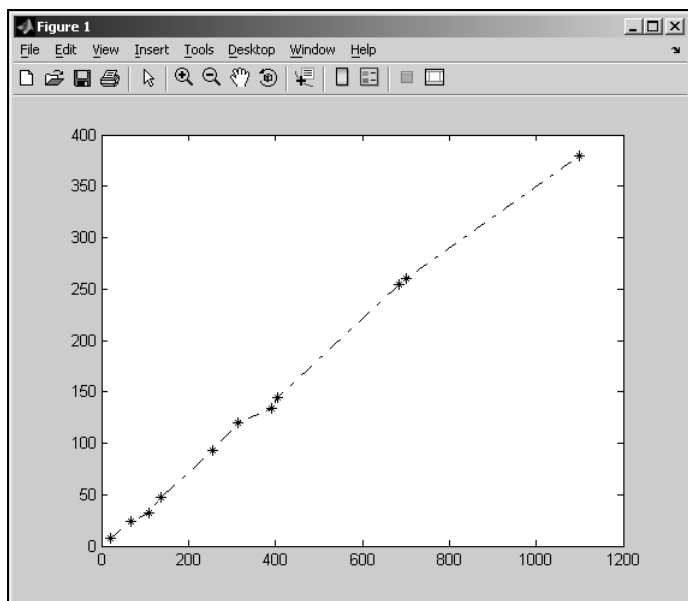


Рис. 7.3. График функции расширения Вселенной

Из рис. 7.3 видно, что функция интерполяции может быть линейной: $V = a + bR$. Остается определить коэффициенты a и b , и математическая модель расширения Вселенной будет найдена.

7.1.3. Функция

plot(x1,y1,s1, x2,y2,s2, ...,xn,yn,sn)

Эта функция позволяет строить большое число математических функций на одном графике. Обозначения имеют следующий смысл:

- ◆ x_i — i -й массив аргументов, заданный в виде вектора;
- ◆ y_i — i -й массив значений функции для заданного массива аргументов;
- ◆ s_i — стиль графика для i -й функции.

Стиль можно не задавать. В этом случае MATLAB выбирает стиль самостоятельно.

Функция $y_i(x_i)$ может задаваться в аналитическом виде. Если эта функция встроенная, то она вводится по общим правилам представления функции, имеющей символьные переменные. Если функция пользовательская, то необходимо создавать m -файл программы на языке программирования системы MATLAB.

В качестве примера вначале построим график одиночной функции и укажем на некоторые особенности его создания.

Пусть функция задана в виде табл. 7.3.

Таблица 7.3. Таблица функции $y = f(x)$

x	1	2	3	4
y	6.2	4.1	1.9	0.6

Необходимо построить график функции с комментариями.

Последовательность команд будет иметь вид:

```
>> x=[1 2 3 4];
>> y=[6.2, 4.1, 1.9, 0.6];
>> plot(x, y, '-q')
```

После нажатия клавиши <Enter> получим график функции.

При необходимости можно поместить в окне монитора заголовки и надписи, а также нанести координатную сетку, выполнив следующие команды:

```
>> title ('Our plot')
>> xlabel ('X axis')
>> ylabel ('Y axis')
>> grid on
```

Все описанные выше команды построения графика приведены на рис. 7.4.

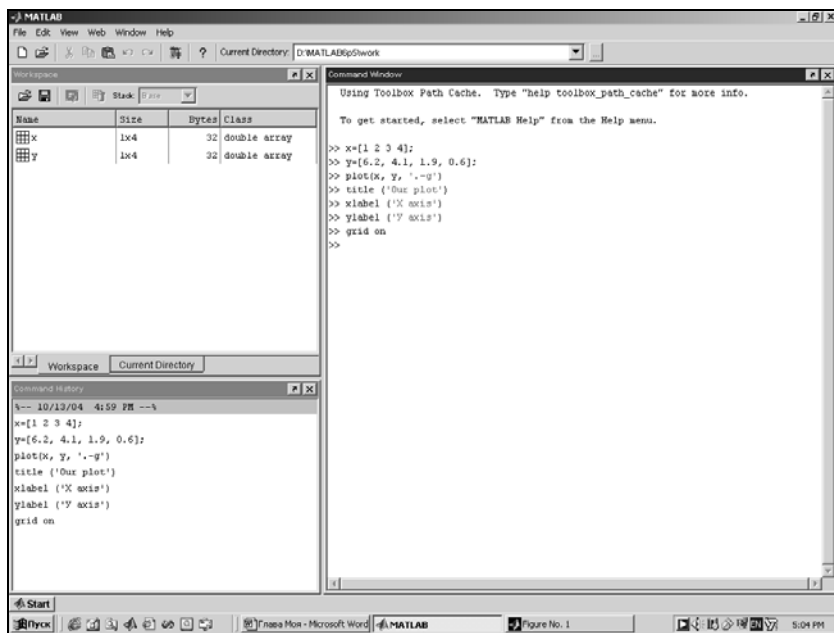


Рис. 7.4. Команды построения графика функции по табл. 7.3

В результате выполнения приведенной последовательности команд получим график, представленный на рис. 7.5.

Если на одном графике необходимо поместить две функции $y(x)$ и $z(x)$, то команда `plot()` будет иметь вид:

```
plot(x, y, x, z)
```

например:

```
>> x=[1, 2, 3, 4, 5];  
>> y=cos(x);  
>> z=exp(-x);  
>> plot(x, y, x, z).
```

Вывести две кривые в одно окно можно также, используя команду `hold on`. После выполнения данной команды все графики будут выводиться в одно окно. Отменить этот режим можно командой `hold off`.

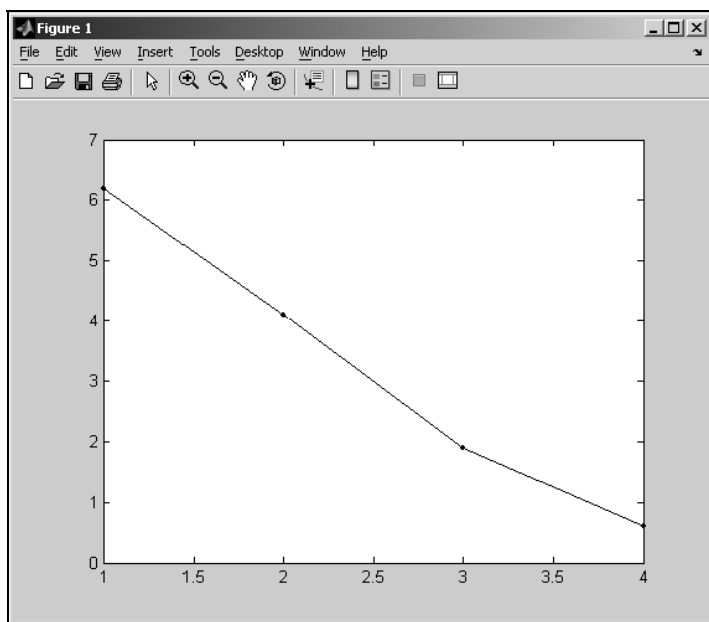


Рис. 7.5. График функции, заданной табл. 7.3

Если функция представляется в символьном виде, то для построения ее графика используется функция `ezplot()`, имеющая вид:

```
f='2*x^2+3*x+1'  
ezplot(f, xн, xк)
```

где f — функция, график которой необходимо построить, $x_{\text{из}}$, $x_{\text{к}}$ — диапазон изменения аргумента.

Откликом здесь является гладкая кривая с представлением на экране вида функции.

Функция `plot(x1,y1,s1,x2,y2,s2,...,sn,yn,sn)` необходима при проверке достоверности решения задачи интерполяции, когда сравниваются две функции: исходная, заданная в виде таблицы, и аналитическая, полученная в результате интерполяции.

Пример 7.3

В результате решения задачи аппроксимации получена следующая математическая модель расширения Вселенной:

$$V = 0.33R + 0.37.$$

Результат получен по данным астронома Хаббла, приведенным в табл. 7.2.

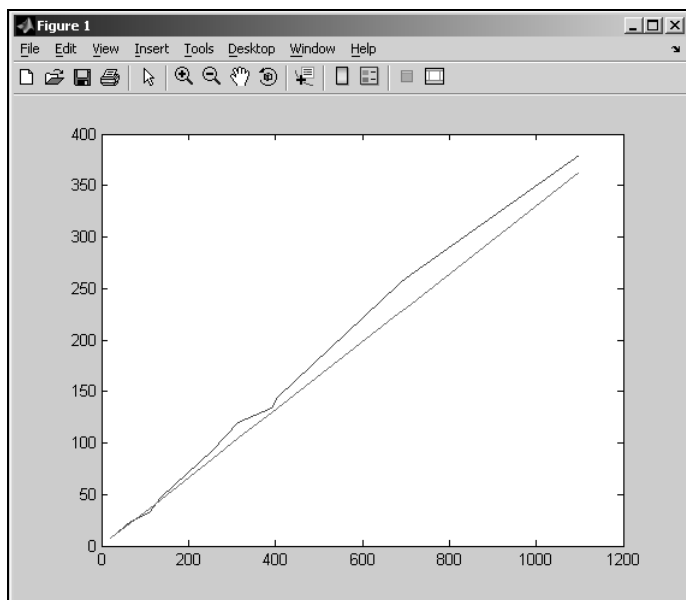


Рис. 7.6. Графики функции расширения Вселенной

Построим зависимость $V = f(R)$ по исходным данным и полученной математической модели и проверим достоверность результатов моделирования.

Программа построения графиков имеет вид:

```
>> R=[22, 68, 108, 137, 255, 315, 390, 405, 685, 700, 1100];  
>> V=[7.5, 24, 32, 47, 93, 120, 134, 144, 254, 260, 380];  
>> F = 0.33 * R + 0.37;  
>> plot(R,V,R,F)
```

Откликом будут графики, показанные на рис. 7.6. Из рисунка видно, что графики существенно отличаются друг от друга. Достоверность математической модели сомнительна.

7.1.4. Функции построения графиков в логарифмическом масштабе

Построение графиков в логарифмическом масштабе необходимо в следующих случаях:

- ◆ исследование устойчивости систем управления частотными методами;
- ◆ исследование качества переходных процессов на основе логарифмических амплитудно-частотных характеристик;
- ◆ анализ помехозащищенности технических объектов;
- ◆ наглядность результатов при их графическом представлении.

В системе MATLAB построение графиков в логарифмическом масштабе осуществляется с помощью функций:

```
loglog(...)  
semilogx(...)  
semilogy(...)
```

Функция `loglog()` строит график в логарифмическом масштабе по обеим осям, функция `semilogx()` — по оси x , функция `semilogy()` — по оси y .

Синтаксис этих функций аналогичен соответствующим функциям `plot()`.

7.1.5. Графики в полярной системе координат

Построение графиков в полярной системе координат осуществляется в MATLAB с помощью следующих функций:

```
polar(θ, r)
```

```
polar(θ, r, s)
```

где:

- ♦ θ — угол функции $r(\theta)$;
- ♦ r — функция, представляющая собой радиус $r(\theta)$;
- ♦ s — вектор стилей, аналогичный функции `plot()`.

Пример 7.4

Построить график в полярной системе координат функции

$$y = \frac{2}{\sin(5t)}.$$

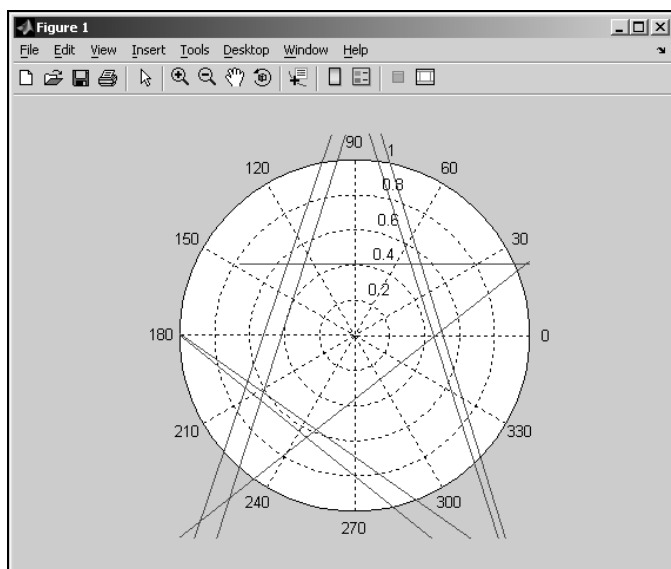


Рис. 7.7. График функции в полярной системе координат

Решение имеет вид совокупности следующих команд:

```
>> t = 0 : pi/40:2 * pi;  
>> polar(t,2./sin(5*t))
```

График функции показан на рис. 7.7.

7.1.6. Создание гистограмм

Гистограмма представляет собой столбиковую диаграмму, характеризующую число попаданий элементов вектора v в каждый из k интервалов. Данные для гистограммы получают с помощью следующих функций:

```
N = hist(Y)  
N = hist(Y, M)  
N = hist(Y, X)  
[N,X] = hist(...)
```

В функциях приняты обозначения:

- ◆ Y — вектор чисел, возвращаемых для десяти интервалов, выбираемых автоматически;
- ◆ M — количество интервалов;
- ◆ X — вектор.

Команда `hist(...)` с синтаксисом, описанным в функциях выбора чисел, строит график гистограммы.

Пример 7.5

Построить гистограмму случайных чисел, распределенных по нормальному закону. Десять тысяч чисел сгенерировать с помощью функции `randn(m,n)`.

Программа решения задачи имеет вид:

```
>> x = -3.5 : 0.2 : 3.5;  
>> y = randn(10000,1);  
>> hist(y,x)
```

График приведен на рис. 7.8.

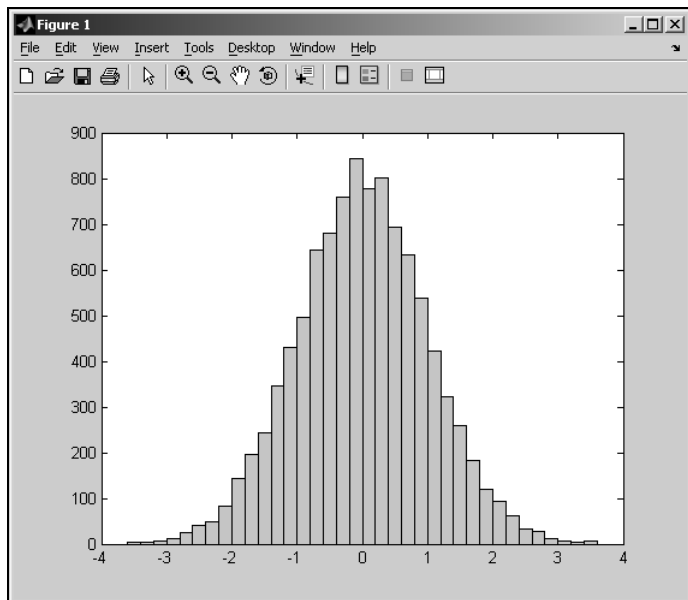


Рис. 7.8. Гистограмма чисел, распределенных по нормальному закону

Из графика видно, что сгенерированные числа действительно подчиняются нормальному закону.

7.2. Трехмерная графика

Система MATLAB имеет богатые возможности построения трехмерных графиков. Мы рассмотрим только несколько функций, позволяющих создавать трехмерные графики.

Для создания трехмерного графика $z = f(x, y)$ необходимо иметь матрицы значений переменных x , y .

Для этого предназначены следующие функции:

```
[X,Y] = meshgrid(x, y)
```

```
[X,Y] = meshgrid(x)
```

```
[X,Y,Z] = meshgrid(x, y, z)
```

Функция `meshgrid(x, y)` — преобразует область векторов x , y в массивы X , Y , которые используются для вычисления функции $z = f(x, y)$ и построения графиков.

Строки массива X являются копиями вектора x , а столбцы массива Y — копиями вектора y . Это видно на следующем примере:

```
>> [X,Y] = meshgrid (1:0.2:1.6, 12:0.5:14)
```

```
X =
```

```

1      1.2    1.4    1.6
1      1.2    1.4    1.6
1      1.2    1.4    1.6
1      1.2    1.4    1.6
```

```
Y =
```

```

12      12      12      12
12.5    12.5    12.5    12.5
13      13      13      13
13.5    13.5    13.5    13.5
14      14      14      14
```

Функция

```
[X,Y,Z] = meshgrid(x, y, z)
```

возвращает трехмерный массив для построения трехмерного графика.

Графики трехмерных поверхностей строятся с помощью следующих функций:

```

plot3(x, y, z)
plot3(X, Y, Z)
plot3(X, Y, Z, S)
plot3(x1, y1, z1, s1, x2, y2, z2, s2, ..., xn, yn, zn, sn)
```

где:

- ◆ x, y, z — векторы аргументов функции;
- ◆ X, Y, Z — матрицы одинакового размера;
- ◆ s — стили линий и точек графика, аналогично функции `plot()`.

Приведенные функции строят точки трехмерного графика и соединяют их отрезками прямых в соответствии с заданным стилем.

Функция `plot3(x1,y1,z1,s1,x2,y2,z2,s2,...,xn,yn,zn,sn)` строит на одном рисунке n функций.

Пример 7.6

Построить график функции

$$z = \ln x + \ln y$$

в диапазоне аргументов $[-4; 4]$ с шагом $h = 0.1$.

Решение имеет вид:

```
>> [X,Y] = meshgrid([-4:0.1:4]);  
>> Z = log (X) + log (Y);  
>> plot3(X,Y,Z)
```

График трехмерной поверхности приведен на рис. 7.9.

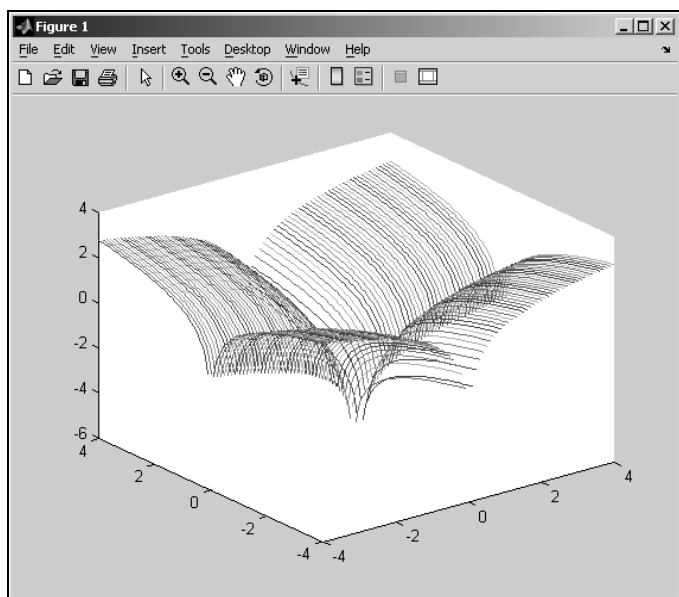
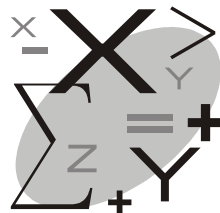


Рис. 7.9. График функции $z = \ln x + \ln y$



ГЛАВА 8

Алгоритмы и технологии решения уравнений

8.1. Алгоритмы решения алгебраических и трансцендентных уравнений

Алгоритм любого метода является совокупностью условий выбора начального приближения, расчетных соотношений и признака окончания вычислительного процесса.

Рассмотрим ряд алгоритмов численных методов определения корней уравнений.

8.1.1. Метод дихотомии (половинного деления)

Сущность метода состоит в следующем. Предположим, что областью изоляции корня уравнения $f(x) = 0$ является $[a; b]$. Тогда за первое приближение к искомому корню x принимается:

$$x_1 = \frac{a + b}{2}.$$

Затем вычисляются значения функции $f(x)$ в точках a и x_1 (или b и x_1). Если $f(a) \cdot f(x_1) < 0$, то новой областью изоляции корня

является $[a; x_1]$, в противном случае — $[b; x_1]$. Равносильным является условие $f(b) \cdot f(x_1) < 0$. Если это условие выполняется, то новой областью изоляции будет $[b; x_1]$, в противном случае — $[a; x_1]$.

Вторым приближением к искомому корню является:

$$x_2 = \frac{a + x_1}{2},$$

если $f(a) \cdot f(x_1) < 0$ или

$$x_2 = \frac{b + x_1}{2},$$

если $f(b) \cdot f(x_1) < 0$. Затем вычисляются значения функции b при $x = x_2$ и проверяется условие $f(x_1) \cdot f(x_2) < 0$ и т. д.

Признаком окончания вычислительного процесса в этом методе является одно из следующих условий:

$$f(x_n) \leq \varepsilon \text{ или } |f(x_n) - f(x_{n-1})| \leq \varepsilon,$$

где ε — допустимая погрешность вычисления корня.

Достоинством метода является простота алгоритма и высокая точность определения корня. Медленная сходимость итераций — основной недостаток метода.

8.1.2. Метод хорд

Алгоритмом метода хорд является совокупность следующих соотношений:

◆ Условие выбора начального приближения:

$$x_0 = \begin{cases} a, & \text{если } f(a)f''(a) < 0 \text{ или } f(b)f''(b) > 0, \\ b, & \text{если } f(a)f''(a) > 0 \text{ или } f(b)f''(b) < 0. \end{cases}$$

♦ Расчетные соотношения:

$$x_n = \begin{cases} x_{n-1} - \frac{(b - x_{n-1})f(x_{n-1})}{f(b) - f(x_{n-1})}, & \text{если } x_0 = a, \\ x_{n-1} - \frac{(a - x_{n-1})f(x_{n-1})}{f(a) - f(x_{n-1})}, & \text{если } x_0 = b. \end{cases}$$

♦ Признак окончания вычислительного процесса:

$$|x_n - x_{n-1}| \leq \varepsilon \quad \text{или} \quad |f(x_n)| \leq \varepsilon.$$

В алгоритме метода хорд приняты следующие обозначения:

- ♦ $[a; b]$ — область изоляции корня;
- ♦ $f(a)$, $f(b)$ — значения функции уравнения $f(x) = 0$ в точках a и b ;
- ♦ $f''(a)$, $f''(b)$ — значения вторых производных функции $f(x)$ в точках a и b ;
- ♦ x_n — приближение корня уравнения $f(x) = 0$, $n = 1, 2, \dots$;
- ♦ ε — погрешность вычисления корня уравнения.

Как видно из описания алгоритма, он является итерационным. Для его реализации в виде программы для ЭВМ необходимо знать:

- ♦ область $[a; b]$ изоляции корня;
- ♦ вторую производную $f''(x)$;
- ♦ значение начального приближения, если не существует аналитическое выражение второй производной функции $f(x)$.

Получить эти данные можно с помощью универсальных программных средств символьной математики, существенно облегчив труд учащегося.

Метод хорд дает возможность получить решение с необходимой точностью с меньшим числом итераций по сравнению с методом половинного деления.

Его недостатками являются:

- ◆ сложность метода в связи с необходимостью вычисления второй производной;
- ◆ неудовлетворительный признак окончания вычислительного процесса.

Последний недостаток объясняется тем, что уточнение корня на каждой из итераций происходит по признаку $|x_n - x_{n-1}| \leq \varepsilon$, в то время как сам корень \bar{x} при этом не находится в области $[x_n; x_{n-1}]$, т. к. приближение к корню идет только от начального приближения a или b . Может оказаться, что абсолютная разность $|x_n - x_{n-1}|$ мала и удовлетворяет условию окончания вычислительного процесса, но при этом корень \bar{x} далеко расположен от x_n .

Если признаком окончания вычислительного процесса считать условие $|f(x)| \leq \varepsilon$, то при этом может оказаться, что значение функции $f(x)$ мало, а абсцисса x_n далеко находится от корня \bar{x} .

Отсутствие хорошего признака окончания вычислительного процесса может привести к вычислению корня уравнения $f(x) = 0$ с погрешностью, превышающей ε , хотя оба условия окончания вычислительного процесса выполнены.

8.1.3. Метод касательных

Идея метода состоит в следующем. Выбирается произвольно значение x , принадлежащее функции $f(x)$ уравнения $f(x) = 0$. Проводится касательная к функции в этой точке до пересечения ее с осью абсцисс. Точка пересечения касательной с осью абсцисс (обозначим ее x_1) принимается за первое приближение корня.

Вычисляется значение функции $f(x_1)$ в точке x_1 и вновь проводится касательная в точке с координатами $(x_1, f(x_1))$. Точка x_2 пересечения касательной с осью абсцисс принимается за второе

приближение корня уравнения $f(x) = 0$ и т. д. Признаком окончания вычислительного процесса, как и в методе хорд, является выполнение одного из условий:

$$|x_n - x_{n-1}| \leq \varepsilon \text{ или } |f(x)| \leq \varepsilon.$$

Легко получить следующую рекуррентную формулу вычисления приближений:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}, \quad (8.1)$$

где $f'(x_{n-1})$ — производная функции $f(x)$ в точке x_{n-1} .

Начальное приближение x_0 , как и в методе хорд, зависит от вида функции $f(x)$ и области изоляции корня $[a; b]$. При этом оказывается, что оно будет противоположным значению x_0 в методе хорд. Если в методе хорд $x_0 = a$, то в методе касательных $x_0 = b$ и наоборот.

Алгоритмом метода касательных является совокупность следующих соотношений:

◆ Условие выбора начального приближения:

$$x_0 = \begin{cases} a, & \text{если } f(a)f''(a) > 0 \text{ или } f(b)f''(b) < 0, \\ b, & \text{если } f(a)f''(a) < 0 \text{ или } f(b)f''(b) > 0. \end{cases}$$

◆ Расчетное соотношение:

$$x_n = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})}.$$

◆ Признак окончания вычислительного процесса:

$$|x_n - x_{n-1}| \leq \varepsilon \text{ или } |f(x_n)| \leq \varepsilon.$$

Из алгоритма видно, что для его реализации в виде компьютерной программы необходимо знать:

- ◆ область $[a; b]$ изоляции корня;
- ◆ аналитические выражения первой и второй производных;

- ◆ начальное приближение, если не существует аналитического выражения второй производной. Их определение возможно с помощью компьютерных технологий, реализуемых в универсальных программных средствах символьной математики.

Ограничения метода касательных: метод нельзя реализовать на практике, если функция $f(x)$ уравнения $f(x) = 0$ не имеет первой производной. Например, уравнение $2x! - e^{-x} + 5 = 0$ не может быть решено, т. к. функция $x!$ не имеет производной.

Метод касательных имеет те же недостатки, что и метод хорд. По сравнению с методом хорд он более трудоемкий, т. к. требует вычисления в точках $x_0, x_1, x_2, \dots, x_n$ не только значений функции $f(x)$, но и ее производной. Его достоинство: во многих случаях дает высокую точность результата при малом числе итераций.

8.1.4. Комбинированный метод (метод хорд и касательных)

Существенным недостатком методов хорд и касательных является неудовлетворительный признак окончания вычислительного процесса. Условия

$$|x_n - x_{n-1}| \leq \varepsilon \quad \text{или} \quad |f(x_n)| \leq \varepsilon$$

не всегда обеспечивают необходимую точность определения корня уравнения $f(x) = 0$.

Комбинированный способ позволяет устранить этот недостаток. Из описания методов хорд и касательных следует, что если один из них дает значение корня с недостатком, то другой — обязательно с избытком. Эта особенность методов предоставляет возможность выработать хороший признак окончания вычислений и обеспечить необходимую точность результата.

Обозначим $x_n^{(x)}$ — n -е приближение корня, вычисленное по методу хорд, $x_n^{(k)}$ — по методу касательных.

Тогда для оценки погрешности вычисления корня целесообразно воспользоваться условием $\left| x_n^{(x)} - x_n^{(k)} \right| \leq \varepsilon$, т. к. достоверно известно, что в диапазоне $\left[x_n^{(x)} - x_n^{(k)} \right]$ обязательно находится искомый корень.

Алгоритмом вычисления корней комбинированным методом является совокупность следующих соотношений:

◆ Условия выбора начального приближения:

$$x_0 = \begin{cases} a, & \text{если } f(a)f''(a) > 0 \text{ или } f(b)f''(b) < 0, \\ b, & \text{если } f(a)f''(a) < 0 \text{ или } f(b)f''(b) > 0. \end{cases}$$

◆ Расчетные соотношения:

$$x_n^{(k)} = x_{n-1} - \frac{f(x_{n-1})}{f'(x_{n-1})};$$

$$x_n^{(x)} = \begin{cases} x_{n-1} - \frac{(b - x_{n-1})f(x_{n-1})}{f(a) - f(x_{n-1})}, & \text{если } x_0 = a, \\ x_{n-1} - \frac{(a - x_{n-1})f(x_{n-1})}{f(a) - f(x_{n-1})}, & \text{если } x_0 = b. \end{cases}$$

◆ Признак окончания вычислительного процесса:

$$\left| x_n^{(x)} - x_n^{(k)} \right| \leq \varepsilon.$$

Недостатком метода хорд и касательных является большая его трудоемкость. Однако при высокой производительности компьютера этот недостаток значения не имеет.

Существенное преимущество метода заключается в его способности обеспечить высокую точность определения корня при конечном числе итераций.

8.1.5. Метод итераций

Исходное уравнение $f(x) = 0$ преобразуется к виду $x = \varphi(x)$. Берется из области изоляции корня $[a, b]$ произвольное значе-

ние x_0 , которое принимается за начальное приближение корня. Приближения x_1, x_2, \dots, x_n вычисляются по соотношениям $x_1 = \varphi(x_0), x_2 = \varphi(x_1), \dots, x_n = \varphi(x_{n-1})$.

Повторяя эти процедуры многократно, можно вычислить значение корня с заданной точностью.

При практическом использовании метода итераций возникают следующие вопросы:

- ◆ Каковы условия сходимости итерационного процесса?
- ◆ Если итерационный процесс расходится, то каким образом можно обеспечить его сходимость?
- ◆ Как определить погрешность вычисления корня?

Существуют две теоремы, которые отвечают на первый вопрос. Приведем их без доказательства.

Теорема 1

Если в итерационном процессе $x_n = \varphi(x_{n-1})$ последовательность x_1, x_2, \dots, x_n имеет предел, т. е. $\lim_{n \rightarrow \infty} x_n = \bar{x}$, то значение \bar{x} является корнем уравнения $f(x) = 0$.

Теорема 2

Итерационный процесс сходится, если на всем интервале области изоляции корня $[a; b]$ выполняется условие $|\varphi'(x)| < 1$. При этом за значение x_0 принимается любое число из области $[a, b]$.

Теперь ответим на второй вопрос. Известны несколько способов обеспечения сходимости итераций.

Способ 1. Если итерационный процесс $x_n = \varphi(x_{n-1})$ не сходится, то следует представить исходное уравнение $f(x) = 0$ в иной возможной форме и выбрать такое из них, при котором обеспечивается сходимость итерационного процесса.

Способ 2. Переход к обратной функции.

Представим исходное уравнение $x = \varphi(x)$ в виде $y = \varphi(x)$ и решим его относительно x . Получим $x = \Psi(y)$. Найдем производную по y функции $x = \Psi(y)$:

$$\frac{d\Psi(y)}{dy} = \frac{dx}{dy} = \frac{1}{\frac{dy}{dx}} = \frac{1}{\varphi'(x)}.$$

Так как при расходящемся итерационном процессе

$$|\varphi'(x)| > 1, \text{ то } \left| \frac{1}{\varphi'(x)} \right| < 1$$

и итерационный процесс $y_n = \varphi(y_{n-1})$ сходится. Очевидно, что если x_k — корень уравнения $y = \Psi(y)$, то x_k также будет корнем уравнения $x = \varphi(x)$.

Способ 3. Подбор множителя.

Предположим, что исходное уравнение $f(x) = 0$ преобразовано к виду $x = \varphi(x)$ и $|\varphi'(x)| > 1$, т. е. процесс расходится. Выберем произвольно функцию $g(x) \neq 0$ и умножим исходное уравнение на $g(x)$. Тогда получим:

$$f(x) \cdot g(x) = 0$$

или

$$x = x - f(x) \cdot g(x).$$

Теперь

$$\varphi(x) = x - f(x) \cdot g(x).$$

Подберем функцию $g(x)$ такую, чтобы удовлетворялось условие $|\varphi'(x)| < 1$ во всей области изоляции корня $[\alpha, b]$.

Признаком окончания вычислительного процесса во всех предыдущих методах было одно из условий

$$|x_n - x_{n-1}| \leq \varepsilon \quad \text{или} \quad |f(x_n)| \leq \varepsilon.$$

В методе итераций условием сходимости итерационного процесса и обеспечения необходимой точности определения корня является

$$|x_n - x_{n-1}| \leq \varepsilon \quad \text{при} \quad |\phi'(x)| < \frac{1}{2}.$$

Из описания метода итераций можно сформулировать следующий алгоритм решения уравнения $f(x) = 0$ методом итераций:

- ♦ условие выбора начального приближения

$$a \leq x_0 \leq b;$$

- ♦ расчетное соотношение

$$x_n = \phi(x_{n-1}) \quad \text{при условии} \quad |\phi'(x)| < \frac{1}{2};$$

- ♦ признак окончания вычислений

$$|x_n - x_{n-1}| \leq \varepsilon.$$

Из описания метода итераций видно, что основным его недостатком является сложность обеспечения сходимости итерационного процесса и точности определения корня.

Из описания методов дихотомии, хорд, касательных и итераций следует, что разработка программ определения корней уравнений требует в ряде случаев от программиста знания области изоляции корня, первой и второй производных функций $f(x)$, значения начального приближения, проверки условия и обеспечения сходимости итерационного процесса. Только при этих условиях могут быть составлены программы перечисленных методов и проведены исследования алгоритмов. Определить область изоляции корня и вычислить значения производных и тем более проверить и обеспечить сходимость итерационного процесса "вручную" чрезвычайно трудно.

Время, затраченное на эти процедуры, будет превосходить время, необходимое для составления программ.

Здесь следует использовать универсальные программные средства символьной математики, в частности систему MATLAB.

8.2. Технология решения алгебраических и трансцендентных уравнений в среде MATLAB

Решение алгебраических и трансцендентных уравнений в среде MATLAB осуществляется с помощью следующих встроенных функций: `solve()`, `fzero()`, `roots()`.

Технология решения уравнений с помощью встроенных функций предельно проста. Рассмотрим ее и приведем примеры.

8.2.1. Технология решения уравнений с помощью функции *solve()*

Функция `solve()` представляется в следующем виде:

```
solve('f(x)', x)
```

где:

- ◆ `'f(x)'` — решаемое уравнение, записанное в одиночных кавычках;
- ◆ `x` — искомое неизвестное.

Уравнение $f(x)=0$ записывается в произвольной форме. При этом если знак равенства отсутствует, то программа воспринимает уравнение в виде $f(x)=0$.

Аргумент `x` при решении уравнения можно опустить.

Функция `syms()`, определяющая имя символьной переменной и обязательная при решении систем уравнений, здесь может быть опущена.

Рассмотрим технологию определения корня уравнения с помощью функции `solve()` на примерах.

Пример 8.1

Пусть необходимо решить следующее уравнение:

$$\sin x + x - 1 = 0.$$

Программа решения уравнения имеет вид:

```
>> Y = solve('sin(x) + x - 1 = 0')
```

После нажатия клавиши <Enter> получим следующее решение:

```
Y =  
0.510973
```

Функция `solve()` в ряде случаев позволяет определить все корни уравнения $f(x)=0$ без указания начальных значений x или областей изоляции корней.

Пример 8.2

Необходимо определить корни уравнения

$$2^x - 4x + 3 = 0.$$

Программа решения уравнения и результат имеют вид:

```
>> Y = solve('2.^x - 4 * x + 3 = 0')  
Y =  
1.418  
3.413
```

Найдены оба корня уравнения.

Функция `solve()` позволяет найти не только вещественные, но и комплексные корни уравнения $f(x)=0$. Покажем это на примере.

Пример 8.3

Пусть уравнение имеет вид:

$$\sin x + \ln x + e^x - 1 = 0.$$

Необходимо найти корни уравнения.

Программа имеет вид:

```
>> Y = solve('sin(x) + log(x) + exp(x) - 1 = 0')
```

После нажатия клавиши <Enter> получим следующее решение:

```
Y =  
3.055 - 1.71447 i
```

Обратите внимание на то обстоятельство, что, найдя комплексный корень уравнения, функция не выдала вещественного корня, который имеет значение: $x=0.407$.

Большое достоинство функции `solve()` в том, что она позволяет решать уравнения, представленные в аналитическом виде.

Пример 8.4

Необходимо решить следующее уравнение:

$$2^x - 3(a - b) = 0.$$

Решение будет иметь вид:

```
>> Y=solve('2^x-3*(a-b)=0')  
Y =  
Log((3*a-3*b)/log(2))
```

Функция `solve()` имеет следующий недостаток. Она не требует информации о начальном значении корня или области его изоляции. Поэтому в случае трансцендентного уравнения и в ряде других случаев она не находит всех корней уравнения.

В качестве примера в табл. 8.1 приведены уравнения и количество корней в каждом из них. Функция `solve()` нашла только один корень, находящийся в столбце x .

Таблица 8.1. Уравнения и их корни

Уравнение	Число корней	X
$x! + 2x - 2 = 0$	4	0.555
$x^2 + 4 \sin x - 2 = 0$	2	0.463
$2 \sin(\ln x) = 0$	6	1
$\ln(4 - 2x) + x^2 - 2 = 0$	3	1.2774

8.2.2. Технология определения вещественных корней уравнения с помощью функции *fzero()*

Функция *fzero()* имеет следующие реализации:

```
fzero('f(x)', x)
fzero('f(x)', [x1, x2])
fzero('f(x)', x, tol, trace)
fzero('f(x)', [x1, x2], tol)
fzero('f(x)', [x1, x2], tol, trace)
```

В выражениях функции приняты следующие обозначения:

- ◆ 'f(x)' — решаемое уравнение, взятое в одиночные кавычки;
- ◆ x — начальное приближение (значение) искомого корня;
- ◆ [x1, x2] — область изоляции корня;
- ◆ tol — заданная погрешность вычисления корня;
- ◆ trace — значение корня в каждой итерации.

Технологию определения корней уравнения покажем на примерах.

Пример 8.5

Необходимо найти корни уравнения

$$2^x - 4x + x \sin x = 0,$$

если известно, что корни находятся вблизи значений $x=1$ и $x=4$.

Решение:

```
>> Y = fzero('2^x - 4 * x + x * sin(x)', 1)
Y =
    0.3478
>> Y = fzero('2^x - 4 * x + x * sin(x)', 4)
Y =
    4.4761
```

Пример 8.6

Нужно определить вещественные корни уравнения

$$\ln(4 - 2x) + x^2 - 2 = 0,$$

используя функцию

```
fzero('y(x)', [x1,x2])
```

На рис. 8.1 приведен график функции.

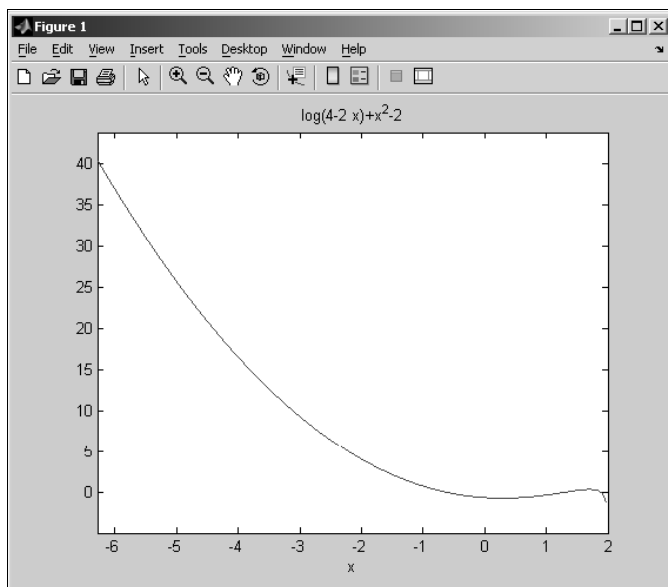


Рис. 8.1. График функции $\ln(4 - 2x) + x^2 - 2 = 0$

Из рис. 8.1 видно, что областями изоляции корней могут быть: $[0; -1]$, $[1; 1.5]$, $[1.5; 1.95]$.

Тогда программа определения корней и результаты решения задачи будут иметь вид:

```
>> X1 = fzero('log(4 - 2 * x) + x^2 - 2', [0, -1]);  
>> X2 = fzero('log(4 - 2 * x) + x^2 - 2', [1, 1.5]);  
>> X2 = fzero('log(4 - 2 * x) + x^2 - 2', [1.5, 1.95]);  
>> X = [x1, x2, x3]
```

После нажатия клавиши <Enter> получим следующий ответ:

```
X =  
    0.594    1.2774    1.9001
```

Это очень поучительный пример. При определении корня из области $[1.5; 1.95]$ мы не получали решения, выбрав область изоляции $[1.5; 2]$, затем повторно $[1.5; 1.9]$, хотя по виду графика корень должен находиться в этих областях изоляции.

Причинами этого являются:

- ♦ в случае области изоляции $[1.5; 2]$ $\log(4 - 2x)$ при $x=2$ не существует ($\log 0$);
- ♦ в случае области изоляции $[1.5; 1.9]$ решения нет потому, что корень равен 1.9001, т. е. находится вне выбранной области изоляции.

При задании области изоляции корня необходимо не только правильно выбрать область $[x_1; x_2]$, но также согласовать ее с решаемым уравнением. Значение функции $f(x)$ должно существовать во всей области $[x_1; x_2]$.

Пример 8.7

Необходимо вычислить корень уравнения

$$\ln(4 - 2x) + x^2 - 2 = 0$$

из области изоляции $[0; -1]$ с точностью

```
tol = [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001]
```

Программа решения задачи имеет вид:

```
>> X1 = fzero('log(4 - 2 * x) + x^2 - 2', [0, -1, 0.1]);
>> X2 = fzero('log(4 - 2 * x) + x^2 - 2', [0, -1], 0.01);
...
>> X5 = fzero('log(4 - 2 * x) + x^2 - 2', [0, -1], 0.00001);
>> X = [x1, x2, x3, x4, x5]
```

После нажатия клавиши <Enter> получим следующий ответ:

```
X =
    -0.6367    -0.5906    -0.5994    -0.5994    -0.5945    -0.5945
```

По результатам расчета видно влияние погрешности на значение корня. Так, например, если необходимо вычислить значение корня с точностью два знака после запятой, то достаточно задать $\text{tol}=0.01$, а если с точностью четыре знака, то $\text{tol}=0.00001$.

Пример 8.8

Необходимо определить корень уравнения

$$\ln(4 - 2x) + x^2 - 2 = 0$$

из области $[0; -1]$ с выдачей информации о каждой итерации (число итераций не более 50).

Программа решения задачи и результат имеют вид:

```
>> fzero('log(4 - 2 * x) + x^2 - 2', [0, -1], 50)
ans =
```

Funt-count	x	f (x)
1	0	-0.613706
2	-1	0.791759
3	-0.436657	-0.225557
4	-0.634142	0.0638411
5	-0.590577	-0.00619106

```
6          -0.594428    -0.0.000141692
7          -0.594518     1.45892 e 008
8          -0.594516    -3.13434 e 006
ans =
    -0.5945
```

Корень найден с точностью 10^{-6} (данные седьмой и восьмой итераций) за 8 итераций.

Таким образом, функция `fzero()` позволяет найти все корни уравнения $f(x)=0$. Для этого необходимо знать области изоляции корней. Для их определения достаточно построить график функции $y=f(x)$ и визуально определить области изоляции всех корней. Существенный недостаток функции `fzero()` состоит в том, что она не определяет комплексных корней уравнения.

Из изложенного следует важный вывод: система MATLAB не имеет функции, которая позволила бы определить все корни трансцендентного уравнения. Функция `solve()` не определяет всех корней уравнения, т. к. она не имеет информации об области их изоляции. Функция `fzero()` не находит комплексных корней. Только совместно эти функции могут найти все вещественные и комплексные корни. При этом необходимо знать область изоляции всех вещественных корней.

С учетом этих особенностей системы MATLAB компьютерная технология решения алгебраических и трансцендентных уравнений состоит в выполнении процедур, перечисленных ниже:

- ♦ определение области изоляции корня графоаналитическим методом;
- ♦ определение вещественных корней уравнения с помощью функции `fzero()` при заданных значениях погрешности `tol`;
- ♦ определение комплексных корней уравнения с помощью функции `solve('f(x)')`.

Следует при этом иметь в виду, что функция `solve('f(x)')` может выдать некоторые, а иногда и все вещественные корни уравнения $f(x)=0$.

8.2.3. Технология определения корней многочлена с помощью функции *roots()*

Функция `roots()` имеет вид:

`roots(z)`

где z — вектор коэффициентов многочлена.

Рассмотрим технологию решения задачи на примерах.

Пример 8.9

Пусть необходимо определить корни многочлена

$$y = 2x^5 - 3x^4 + 5x^3 + x^2 + 7x + 3.$$

Решение имеет вид:

```
>> Y = roots([2 -3 5 1 7 3])
```

Y =

```
1.2189 + 1.4110 i
1.2189 - 1.4110 i
-0.2719 + 1.0105 i
-0.2719 - 1.0105 i
-0.3940.
```

В случае когда в многочлене отсутствует член x^k , то принимается $a_k = 0$.

Пример 8.10

Определить корни уравнения

$$x^{10} - 1 = 0.$$

Будем рассматривать выражение $x^{10} - 1$ как многочлен десятой степени. Тогда программа вычисления корней будет иметь вид:

```
>> z = [1 0 0 0 0 0 0 0 0 0 -1];
>> y = roots(z)
```

После нажатия клавиши <Enter> получим следующий ответ:

```
Y =  
-1  
-0.8090 + 0.5878 i  
-0.8090 - 0.5878 i  
-0.3090 + 0.8511 i  
-0.3090 - 0.8511 i  
0.3090 + 0.8511 i  
0.3090 - 0.8511 i  
1  
0.8090 + 0.5878 i  
0.8090 - 0.5878 i
```

Замечание

Функция `roots(z)` не позволяет получить решение, если коэффициенты многочлена заданы в виде символьных переменных.

8.2.4. Варианты алгебраических и трансцендентных уравнений для индивидуальных заданий или решений

Алгебраические уравнения для решения аналитическими методами представлены в табл. 8.2.

Алгебраические и трансцендентные уравнения для решения численными методами перечислены в табл. 8.3.

8.3. Методы решения систем алгебраических уравнений

Системы уравнений могут быть линейными и нелинейными, с постоянными и переменными коэффициентами. Решение таких уравнений возможно аналитическими и численными методами. Аналитические методы дают решение в общем виде. Такое решение, в силу своей общности, всегда более предпочтительно, чем численное, которое дает ответ лишь для конкретных числовых данных коэффициентов.

Таблица 8.2. Алгебраические уравнения для решения их аналитическими методами

№ п/п	Уравнение	Ответы
1	$x^3 - a = 0$	<ul style="list-style-type: none"> $x = a^{1/3}$ $x_1 = a^{1/3}, x_{2,3} = a^{1/3} \left(-\frac{1}{2} \pm \frac{\sqrt{3}}{2} i \right)$
2	$x^5 + (b-a)x^4 - abx^3 + ax^2 + ax(b-a) - a^2b = 0$	<ul style="list-style-type: none"> $x_1 = a, x_2 = -a^{1/3}, x_3 = b$ $x_1 = a, x_2 = -a^{1/3}, x_3 = -b, x_{4,5} = a^{1/3} \left(\frac{1}{2} \pm \frac{\sqrt{3}}{2} i \right)$
3	$x^5 + bx^4 - a^4x - a^4b = 0$	<ul style="list-style-type: none"> $x_1 = -a, x_2 = -a, x_3 = -b$ ($x = 0, a = 0$) $x_1 = -a, x_2 = -a, x_3 = -b, x_{4,5} = \pm ai$
4	$x^4 + 2x^3(1+a) + x^2(4a-1) - 2x(1+a) - 4a = 0$	$x_1 = 1, x_2 = 1, x_3 = -2, x_4 = -2a$

Таблица 8.2 (продолжение)

№ п/п	Уравнение	Ответы
5	$x^5 - 3x^4 + (a - b^3)x^3 + 3(b^3 - a)x^2 - ab^3x + 3ab^3 = 0$	<ul style="list-style-type: none"> $x_1 = -\sqrt{-a}, x_2 = \sqrt{-a}, x_3 = 3$ $x_1 = -\sqrt{a}, x_2 = \sqrt{-a}, x_3 = 3, x_{4,5} = \pm i(-b)^{3/2}$ <p>При $a > 0, b > 0$:</p> <ul style="list-style-type: none"> $x = 3$ $x_1 = 3, x_{2,3} = \pm i\sqrt{a}, x_{4,5} = i(-b)^{3/2}$
6	$x^5 - ax^3 + 2x^2 - 2a = 0$	<ul style="list-style-type: none"> $x_{1,2} = \pm\sqrt{a}, x_3 = -2^{1/3}$ $x_{1,2} = \pm\sqrt{a}, x_3 = -2^{1/3}, x_{4,5} = \frac{2^{1/3}}{2} \pm \frac{108^{1/6}}{2}i$
7	$x^7 + ax^5 - x^2 - a = 0$	<ul style="list-style-type: none"> $x_{1,2} = \pm\sqrt{-a}, x_3 = 1$ $x_{1,2} = \pm\sqrt{-a}, x_3 = 1, x_{4,5} = -\frac{\sqrt{5}}{4} - \frac{1}{4}\pm i\sqrt{\frac{5}{8}} - \frac{\sqrt{5}}{8},$ $x_{6,7} = \frac{\sqrt{5}}{4} - \frac{1}{4}\pm i\sqrt{\frac{5}{8}} + \frac{5}{8}$

Таблица 8.2 (продолжение)

№ п/п	Уравнение	Ответы
8	$2^x - 2(a+b) = 0$	<ul style="list-style-type: none"> $x = \frac{\ln(a+b)}{\ln 2} + 1$ $x_1 = \frac{\ln(a+b)}{\ln 2} + 1, x_{2,3} = \frac{\ln(a+b)}{\ln 2} + 1 \pm \frac{2\pi}{\ln 2} i$
9	$\ln \sin x - 2a = 0$	$x_1 = a \sin e^{2a}, x_{2,3} = \pm \pi - a \sin e^{2a}$
10	$e^{-2x} - 2a = 0$	<ul style="list-style-type: none"> $x = -\frac{\ln 2a}{2}$ $x_1 = -\frac{\ln 2a}{2}, x_{2,3} = -\frac{\ln 2a}{2} \pm \pi i$
11	$\sin ax + \cos ax = 0$	$x_1 = -\frac{5\pi}{4a}, x_2 = \frac{3\pi}{4a}, x_3 = -\frac{\pi}{4a}$
12	$\sin ax + \operatorname{tg} ax = 0$	$x_1 = -\frac{3\pi}{a}, x_{2,3} = \pm \frac{\pi}{a}$
13	$e^{-ax} + e^{ax} = 0$	<ul style="list-style-type: none"> False $x_{1,2} = \pm \frac{5\pi}{2a} i, x_{3,4} = \pm \frac{3\pi}{2a} i, x_{5,6} = \pm \frac{\pi}{2a} i$

Таблица 8.2 (окончание)

№ п/п	Уравнение	Ответы
14	$e^{-\alpha} + e^{\alpha} - 1 = 0$	<ul style="list-style-type: none"> • False • $x_{1,2} = \pm \frac{7\pi}{3a}i$, $x_{3,4} = \pm \frac{5\pi}{3a}i$, $x_{5,6} = \pm \frac{\pi}{3a}i$
15	$e^{-\alpha} + e^{\alpha} - \ln a = 0$	$x_{1,2} = \frac{\ln \left(\frac{\ln a}{2} \pm \frac{\sqrt{\ln^2 a - 4}}{2} \right)}{a}$
16	$ae^{-x} + be^x = 0$	<ul style="list-style-type: none"> • $x = \frac{\ln \left(-\frac{a}{b} \right)}{2}$ • $x_1 = \frac{\ln \left(-\frac{a}{b} \right)}{2}$, $x_{2,3} = \frac{\ln \left(-\frac{a}{b} \right)}{2} \pm \pi i$
17	$a \sin x + b \cos x = 0$	$x_{1,2} = -a \tan \left(\frac{b}{a} \right) \pm \pi$, $x_3 = -a \tan \left(\frac{b}{a} \right)$

Таблица 8.3. Алгебраические и трансцендентные уравнения для решения их численными методами

№ п/п	Уравнения		
	Алгебраические	Трансцендентные	Число корней
1	$x^4 - 4x^3 - 10x^2 + 3x - 4 = 0$	$2\sin(\ln x) = 0$	6
2	$x^5 - 5x^2 + 4,5 = 0$	$\operatorname{arctg}(\operatorname{tg} x) = 0$	∞
3	$x^5 - x + 0,2 = 0$	$4e^{-1/ x } - 2$	3
4	$x^3 - 0,2x^2 + 0,5x + 1,5 = 0$	$\sin(\sin x)$	∞
5	$x^{10} - 1 = 0$	$2^x - 4x = 0$	2
6	$x^8 + 2x - 1,5 = 0$	$x! + 2x - 2 = 0$	4
7	$x^4 - 2x^2 + 8x + 1 = 0$	$\ln x + (x + 1)^3 = 0$	1
8	$x^3 + 4x^2 - 5x - 2 = 0$	$\operatorname{arctg}(x - 2) + x = 0$	1
9	$6x^8 - 2x^2 + 3 = 0$	$x^2 + 4\sin x - 2 = 0$	2
10	$3,5x^5 - 2,8x^3 + 7,5x - 2,5 = 0$	$x - \ln(7 - 4x)$	1

Таблица 8.3 (окончание)

№ п/п	Уравнения		
	Алгебраические	Трансцендентные	Число корней
11	$x^5 + x^4 + x^3 + x^2 + x + 1 = 0$	$e^{-2x} + \frac{3}{x} - 1 = 0$	2
12	$1.5x^5 + 17x - 21 = 0$	$e^{-6x} + 3x^2 - 18 = 0$	2
13	$2x^4 - 3.5x^2 + 3 = 0$	$\ln(4 - 2x) + x^2 - 2 = 0$	3
14	$17x^9 - 15x^7 + 13x^4 + 11x^3 + 9x - 7 = 0$	$3^x - 9x + 1 = 0$	2
15	$x^4 + 2x^3 - 1 = 0$	$2^x + \ln 2x - 5.6 = 0$	1
16	$x^x - 21x^2 + 55 = 0$	$2^x + e^{-x} - 5x + 1 = 0$	2
17	$x^7 + 2x^6 + 3x^5 + 4x^4 + 5x^3 + 6x^2 + 7x + 8 = 0$	$3x = e^x + 1.5 = 0$	2
18	$8x^7 - 7x^6 + 6x^5 - 5x^4 + 4x^3 - 3x^2 + 2x - 1 = 0$	$\frac{e^x}{2} - (x - 1)^2 = 0$	1
19	$x^{50} - 1 = 0$	$x + \lg x - 0.5 = 0$	1
20	$x^7 + 6x^6 + x^5 - 4x^4 + x^3 - 2x^2 + x - 1 = 0$	$2x! - e^{-x} + 5 = 0$	4

имеет хотя бы одно решение, в противном случае она называется *несовместной*. Совместная система может иметь единственное решение или бесконечное их число. Если система имеет бесконечное число решений, ее называют *неопределенной*. Рассмотрим примеры таких систем.

Система уравнений

$$\begin{cases} x_1 + 2x_2 - x_3 = 1 \\ 2x_1 + x_2 + x_3 = 2 \\ 3x_1 + 2x_2 + 3x_3 = 5 \end{cases}$$

имеет единственное решение: $x_1 = 0$, $x_2 = 1$, $x_3 = 1$, т. е. она является совместной и определенной.

Система

$$\begin{cases} x_1 + 2x_2 - x_3 = 1 \\ 2x_1 + 4x_2 - 2x_3 = 2 \\ 3x_1 + 2x_2 + 3x_3 = 5 \end{cases}$$

имеет бесконечное число решений, которые удовлетворяют следующим равенствам:

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

т. е. система является совместной и неопределенной. Определены лишь условия решения. В этой системе первое и второе уравнения идентичны.

Система

$$\begin{cases} x_1 + 2x_2 - x_3 = 1 \\ 2x_1 + 4x_2 - 2x_3 = 2 \\ 3x_1 + 6x_2 - 3x_3 = 3 \end{cases}$$

не имеет ни одного решения и является несовместной.

Действительно, в этой системе будет три одинаковых уравнения после деления второго уравнения на 2, а третьего на 3.

Существует большое число различных методов решения систем линейных уравнений. Все они могут быть разделены на две группы: точные методы и методы последовательных приближений. Следует при этом иметь в виду, что точными методами являются только аналитические методы. Если с помощью этих методов решать систему уравнений с числовыми коэффициентами, то точных решений можно не получить за счет ошибок вычислений, связанных с конечной памятью компьютера.

Наиболее популярным из точных методов решения линейных алгебраических уравнений является метод Гаусса. Метод Гаусса изучается в математике, и нет надобности его здесь описывать. Напомним только теорему Крамера.

Если определитель матрицы коэффициентов $|A|$ системы n линейных уравнений с n неизвестными отличен от нуля, то система имеет решение и притом единственное.

При решении системы n линейных уравнений необходимо выполнить следующее количество операций:

$$N = \frac{2n(n+1)(n+2)}{3} + n(n-1).$$

Если, например, $n=10$, то число операций будет $N=970$.

Метод Гаусса может привести к существенным ошибкам при определении неизвестных x_1, x_2, \dots, x_n в случае плохо обусловленных систем. *Плохо обусловленной* называется такая система, у которой модуль определителя матрицы коэффициентов мал по сравнению с какой-либо из норм матрицы. *Нормой матрицы* может быть: максимальная из сумм модулей коэффициентов строк или столбцов. Плохо обусловленные системы чувствительны к ошибкам округления, которые неизбежны при компьютерных методах реализации алгоритма Гаусса.

Рассмотрим решение линейных алгебраических уравнений методом простой итерации.

Разрешим исходную систему уравнений (8.2) относительно неизвестных:

Ответы на эти вопросы совместно с расчетными соотношениями и будут алгоритмом решения систем линейных уравнений методом итерации. Ответим на поставленные вопросы для случая линейных систем алгебраических уравнений.

Выбор начальных приближений

Если область, в которой находятся неизвестные x_i ($i = 1, 2, \dots, n$), известна, то начальные значения выбираются произвольно из этой области. Если область неизвестна, то за начальные приближения можно взять свободные члены $\frac{b_1}{a_{11}}, \frac{b_2}{a_{22}}, \dots, \frac{b_n}{a_{nn}}$.

Условия сходимости итерационного процесса

Условием сходимости итерационного процесса является: сумма абсолютных значений отношений коэффициентов в каждом уравнении системы к диагональному должна быть меньше единицы.

Обеспечить сходимость итерационного процесса можно путем преобразования исходной системы к эквивалентной. Эти преобразования можно выполнить путем перестановки уравнений, операций сложения и вычисления уравнений, умножения на постоянный коэффициент.

Рассмотрим пример. Необходимо решить методом итераций следующую систему уравнений:

$$\begin{cases} 2x_1 + 3x_2 + x_3 = 1 \\ -7x_1 - 2x_2 + 4x_3 = 6 \\ 8x_1 + x_2 - 3x_3 = 12 \end{cases}$$

Эта система имеет решение: $x_1 = 5$, $x_2 = -\frac{11}{2}$, $x_3 = \frac{15}{2}$.

Однако решать эту систему уравнений методом итераций опасно, т. к. здесь не обеспечены условия сходимости итераций.

Действительно, в первом уравнении

$$\left| \frac{3}{2} \right| + \left| \frac{1}{2} \right| > 1,$$

во втором

$$\left| \frac{7}{2} \right| + \left| \frac{4}{2} \right| > 1,$$

в третьем

$$\left| \frac{8}{3} \right| + \left| \frac{1}{3} \right| > 1.$$

Для обеспечения условий сходимости преобразуем исходную систему уравнений. Второе уравнение поставим в первую строку. Тогда

$$\left| \frac{2}{7} \right| + \left| \frac{4}{7} \right| < 1.$$

Заметим, что первое уравнение можно заменить также третьим.

Умножим первое уравнение на 4 и сложим его со вторым. Тогда получим

$$+x_1 + 10x_2 + 8x_3 = 4.$$

Теперь

$$\left| \frac{1}{10} \right| + \left| \frac{8}{10} \right| < 1.$$

Это уравнение можно сделать вторым.

Для получения третьего уравнения выполним следующие преобразования: сложим все уравнения, полученное уравнение умножим на 2 и сложим со вторым. Тогда получим:

$$-x_1 + 2x_2 + 8x_3 = 44.$$

Теперь

$$\left| \frac{1}{8} \right| + \left| \frac{2}{8} \right| < 1,$$

условие сходимости итераций выполнено. В результате всех этих операций получена следующая эквивалентная система уравнений:

$$\begin{cases} -7x_1 - 2x_2 + 4x_3 = 6 \\ x_1 + 10x_2 + 8x_3 = 10 \\ -x_1 + 2x_2 + 8x_3 = 44 \end{cases}$$

Теперь условия сходимости итераций выполнены полностью. Решение системы уравнений итерационным методом возможно при начальных условиях

$$x_1^{(0)} = -\frac{6}{7}, \quad x_2^{(0)} = 1, \quad x_3^{(0)} = \frac{44}{8} = 5,5.$$

Признак окончания вычислений

Признаком окончания итерационного процесса из условий точно-сти можно в первом приближении считать условие:

$$|x_k^{(v+1)} - x_k^{(v)}| \leq \varepsilon,$$

где $x_k^{(v+1)}$, $x_k^{(v)}$ — значение k -го неизвестного соответственно на $(v+1)$ и (v) итерациях, ε — допустимая погрешность вычисления неизвестных.

Существуют два способа решения уравнений методом итераций: метод простой итерации и метод Зейделя. Пользователь не имеет возможности выбирать метод итераций, т. к. функции и команды системы MATLAB не различают этих методов. Метод решения уравнений в системах символьной математики пользователю неизвестен, следовательно, выбора не существует.

8.3.2. Алгоритмы метода итерации

Алгоритм метода простой итерации состоит из совокупности условий выбора начальных приближений, расчетных соотношений и признака окончания вычислений.

8.3.3. Сравнительная оценка точных и итерационных методов

В методе простой итерации на одну итерацию необходимо выполнить приблизительно $2n^2$ арифметических операций типа сложения и умножения, в то время как по методу Гаусса для решения системы n уравнений необходимо выполнить $\frac{2}{3}n^3$ операций. Тогда очевидно, что метод итераций более целесообразен для случая, когда возможно получить решение задачи не более чем за $\frac{1}{3}n$ итераций, т. е. он выгоден при решении уравнений больших размерностей.

Логическая схема итерационных методов очень проста, поэтому компьютерные программы короче, чем в методе Гаусса.

Итерационные методы позволяют распараллеливать алгоритм, что дает возможность эффективно решать уравнения на много-процессорных компьютерах.

Недостатки итерационных методов в том, что они требуют от пользователя проверки условий сходимости итераций, и если условия не выполняются, то преобразовывать исходные уравнения к виду, когда обеспечивается сходимость итерационного процесса. Кроме этого, итерационные методы требуют выбора начальных приближений. Все это существенно усложняет компьютерные технологии решения уравнений.

8.4. Компьютерные технологии решения систем линейных алгебраических уравнений в среде MATLAB

Рассмотрим следующие способы решения систем линейных уравнений средствами MATLAB:

- ♦ способ вычисления определителей матрицы коэффициентов системы уравнений;

- ◆ матричный способ;
- ◆ с помощью функции `solve()`;
- ◆ с помощью встроенной функции `nnls()`.

8.4.1. Решение системы линейных уравнений с помощью определителей

Пусть D — главный определитель матрицы коэффициентов системы уравнений, d_k — частный определитель, образованный заменой коэффициентов при k -м неизвестном системы уравнений на коэффициенты правых частей уравнений (свободных членов). Тогда неизвестное x_k вычисляется по выражению:

$$x_k = \frac{d_k}{D}.$$

Приведем примеры решения уравнений методом определителей.

Пример 8.11

Необходимо решить следующую систему линейных уравнений:

$$\begin{cases} 2x_1 + x_2 - 3x_3 = 1 \\ x_1 - x_2 + 2x_3 = 18 \\ 7x_1 + 5x_2 + x_3 = 3 \end{cases}$$

Методом определителей решение системы будет иметь вид:

$$x_1 = \frac{\begin{vmatrix} 1 & 1 & -3 \\ 18 & -1 & 2 \\ 3 & 5 & 1 \end{vmatrix}}{D}, \quad x_2 = \frac{\begin{vmatrix} 2 & 1 & -3 \\ 1 & 18 & 2 \\ 7 & 3 & 1 \end{vmatrix}}{D}, \quad x_3 = \frac{\begin{vmatrix} 2 & 1 & 1 \\ 1 & -1 & 18 \\ 7 & 5 & 3 \end{vmatrix}}{D},$$

$$D = \begin{vmatrix} 2 & 1 & -3 \\ 1 & -1 & 2 \\ 7 & 5 & 1 \end{vmatrix}.$$

Теперь представим определители в виде матриц:

```
dx1 = [1, 1, -3; 18, -1, 2; 3, 5, 1]
```

```
dx2 = [2, 1, -3; 1, 18, 2; 7, 3, 1]
```

```
dx3 = [2, 1, 1; 1, -1, 18; 7, 5, 3]
```

```
D = [2, 1, -3; 1, -1, 2; 7, 5, 1]
```

Тогда программа вычисления неизвестных будет иметь вид:

```
>> x1 = det(dx1) / det(D);
```

```
>> x2 = det(dx2) / det(D);
```

```
>> x3 = det(dx3) / det(D);
```

```
>> X = [x1, x2, x3]
```

```
X =
```

```
6.7111 -9.0222 1.1333
```

Недостаток метода определителей состоит в том, что он требует образования числа матриц на одну больше, чем число неизвестных.

8.4.2. Матричный метод решения систем линейных уравнений

Пусть A — матрица коэффициентов системы уравнений, B — вектор свободных членов, X — вектор неизвестных. Тогда неизвестные определяются по одному любому из следующих выражений:

$$X = A^{-1} * B$$

$$X = A \setminus B$$

$$X = \text{inv}(A) * B$$

Пример 8.12

Решить систему уравнений предыдущего примера:

$$\begin{cases} 2x_1 + x_2 - 3x_3 = 1 \\ x_1 - x_2 + 2x_3 = 18 \\ 7x_1 + 5x_2 + x_3 = 3 \end{cases}$$

В данном случае

$$\mathbf{A} = \begin{vmatrix} 2 & 1 & -3 \\ 1 & -1 & 2 \\ 7 & 5 & 1 \end{vmatrix}, \quad \mathbf{B} = [1, 18, 3], \quad \mathbf{X} = [x_1, x_2, x_3].$$

Тогда решение будет иметь вид:

```
>> A = [2,1,-3; 1,-1,2; 7,5,1];  
>> B = [1;18;3];  
>> X = inv(A) * B  
ans =  
    6.7111  
   -9.0222  
    1.1333
```

MATLAB позволяет решать системы линейных уравнений в аналитическом виде, когда коэффициенты уравнений являются символьными переменными. Для этого необходимо определить символьные переменные с помощью встроенной функции `syms X`, где `X` — перечень символьных переменных, которые отделяются пробелами.

Пример 8.13

Необходимо решить следующую систему уравнений:

$$\begin{cases} ax_1 + bx_2 = 1 \\ cx_1 + dx_2 = 3 \end{cases}$$

Решение:

```
>> syms a b c d;  
>> A = [a,b; c,d];  
>> B = [1;3];  
>> X = inv(A) * B
```

Запишем ответ в виде, удобном для чтения:

$$x_1 = \frac{d}{ad-bc} - \frac{3b}{ad-bc},$$

$$x_2 = \frac{c}{ad-bc} + \frac{3a}{ad-bc}.$$

Программа позволяет решать уравнения в случае, когда коэффициенты являются числами комплексными.

Пример 8.14

Решить следующую систему уравнений:

$$\begin{cases} 2x_1 + (3+i)x_2 = -1 \\ -x_1 + 2x_2 = 3-2i \end{cases}$$

Решение имеет вид:

```
>> A = [2, 3+i; -1, 2];
>> B = [-1; 3 - 2i];
>> X = inv(A) * B
X =
    -1.76 + 0.68 i
     0.62 - 0.66 i
```

8.4.3. Решение систем линейных уравнений с помощью функции *solve()*

Функция `solve()` в случае решения систем уравнений имеет вид:

```
solve ('f1', 'f2', ..., 'fn')
solve ('f1', 'f2', ..., 'fn', x1, x2, ..., xn)
```

где:

- ♦ 'f_i' — *i*-е уравнение системы, $i = 1, 2, \dots, n$;
- ♦ x_i — *i*-е неизвестное, $i = 1, 2, \dots, n$.

Каждое уравнение системы берется в одинарные кавычки и отделяется от предыдущего запятой.

Перед функцией `solve()` необходимо с помощью функции `syms` определить символьные переменные.

Технологию решения системы уравнений рассмотрим на примере.

Пример 8.15

Пусть необходимо решить следующую систему уравнений:

$$\begin{cases} 3x + y - z = 3 \\ -5x + 3y + 4z = 1 \\ x + y + z = 0.5 \end{cases}$$

Программа решения системы уравнений имеет вид:

```
>> syms x y z;  
>> Y = solve('3*x+y-z=3', '-5*x+3*y+4*z=1', 'x+y+z=0,5')
```

После нажатия клавиши <Enter> получим ответ в следующем виде:

```
x : [ 1x1 sym]  
y : [ 1x1 sym]  
z : [ 1x1 sym]
```

Программа задачу решила, но не выдала значения неизвестных x , y , z . Для их получения необходимо воспользоваться командой `Y.k`, где k — имя неизвестного. В нашем случае решение будет иметь вид:

```
>> Y.x  
ans =  
    -1.10714  
>> Y.y  
ans =  
    1.96428  
>> Y.z  
ans =  
   -1.35714
```

Можно также использовать функцию

```
vpa(Y.k,n)
```

где:

◆ k — искомое неизвестное;

◆ n — число знаков ответа.

Получим решение с числом знаков $n=5$:

```
>> vpa(Y.x, 5)
```

```
ans =
```

```
0.10714
```

```
>> vpa(Y.y, 5)
```

```
ans =
```

```
1.9642
```

```
>> vpa(Y.z, 5)
```

```
ans =
```

```
1.3571
```

8.5. Компьютерные технологии решения систем нелинейных уравнений

Решение систем нелинейных уравнений в MATLAB осуществляется функцией `fsolve()`, которая имеет вид:

```
fsolve('file', x0)
```

где `file` — система уравнений, сохраненная в `m`-файле.

Пример 8.16

Пусть необходимо решить следующую систему нелинейных уравнений:

$$\begin{cases} x_1 x_2 + x_3 = 6.5 \\ x_1 x_2^4 + x_3 = 167 \\ x_1 x_2^6 + x_3 = 1470 \end{cases}$$

Представим систему уравнений в виде функции пользователя с именем `myfun` и сохраним ее в файле `myfun.m`.

Пусть содержимое файла имеет вид:

```
Func F = myfun (x)
F = [x(1)*x(2)+x(3)-6.5; x(1)*x(2)^4+x(3)-167;
x(1)*x(2)^6+x(3)-1470]
```

Программа и результаты решения имеют вид:

```
>> x0 = [1;1;1];
>> X = fsolve('myfun', x0)
```

После нажатия клавиши <Enter> получим следующее решение:

```
X =
    2.1512
    2.9678
    0.1157
```

MATLAB имеет большое число функций решения специальных уравнений. В них реализованы итерационные методы решения уравнений и метод наименьших квадратов. Некоторые из этих функций позволяют решать также обычные системы линейных уравнений.

Таковыми функциями являются: `bicq()`, `cqs()`, `qmrres()`, `qmr()`. Опишем кратко эти функции и приведем примеры решения систем линейных уравнений.

Функция `bicq()` имеет вид:

```
bicq(A, B)
```

где:

- ◆ A — матрица коэффициентов системы уравнений;
- ◆ B — вектор свободных членов.

Функция вычисляет неизвестные методом итераций.

За начальные приближения по умолчанию принимает нулевой вектор длиной n , где n — число неизвестных. Количество итераций определяется либо по максимальному их числу (20 по

умолчанию), либо по достижению относительной погрешности (по умолчанию 10^{-6}).

Функция имеет несколько модификаций. Приведем две:

- ◆ `bicq(A,B,tol)` — выдает решение с погрешностью `tol`;
- ◆ `bicq(A,B,tol,maxit)` — выдает решение с погрешностью `tol` при заданном числе итераций.

Пример 8.17

Пусть необходимо решить следующую систему уравнений:

$$\begin{cases} 1.5x_1 + 2.7x_2 + 0.75x_3 = 1.8 \\ 6x_1 - x_2 + 3x_3 = -1 \\ x_1 + 7x_2 - 4x_3 = 3 \end{cases}$$

Решение выполним с помощью функции `bicq()` и ее модификаций при `tol=1.e-4` и `maxit=15`.

Решение:

```
>> A = [1.5  2.7  0.75 ;  6  -1  3 ;  1  7  -4 ];
>> B = [1.8; -1; 3];
>> y1 = bicq(A,B);
>> y2 = bicq(A,B,0.0001);
>> y3 = bicq(A,B,0.0001,15);
>> Y = [y1, y2, y3]
Y =
    -0.2524    -0.2524    -0.2524
     0.6949     0.6949     0.6949
     0.4030     0.4030     0.4030
```

Во всех случаях ответ одинаков. Это объясняется тем, что данный ответ при используемом числе итераций является точным.

Функция `cqs()`

Функция `cqs()` представляется в следующем виде:

```
cqs(A, B)
cqs(A, B, tol)
cqs(A, B, tol, maxit)
```

Этой функцией реализуется так называемый *квадратичный метод сопряженных аргументов*. Итерационный процесс начинается с нулевой итерации (нулевой вектор начальных значений). Число итераций определяется по одному из следующих признаков:

- ◆ число итераций задается пользователем;
- ◆ сходимость итерационного процесса;
- ◆ погрешность результата.

Пример 8.18

Необходимо решить с помощью функции `cqs()` систему уравнений предыдущего примера. Решение необходимо получить, используя функцию

`cqs(A, B)`

и ее модификации.

Программа решения задачи такая же, как и в примере 8.16, только имя функции будет не `bicq()`, а `cqs()`. Ответ получается прежним, поэтому нет смысла его приводить.

Функции `qmrres()` и `qmr()` решают систему линейных уравнений методом итераций, используя соответственно метод минимизации обобщенной невязки и метод квазимиимизации невязки [9]. Они имеют те же модификации, что и функция `bicq()`. Технология решения уравнений не отличается от рассмотренных выше.

8.6. Варианты уравнений для индивидуального решения

Далее приводятся два задания для индивидуального решения систем уравнений с помощью MATLAB.

Задание 1. Решение систем линейных алгебраических уравнений

Решить приведенные в табл. 8.4 системы уравнений матричным методом, методами Гаусса и Ньютона, методом итераций.

Таблица 8.4. Системы линейных уравнений

№	Система уравнений	№	Система уравнений
1	$\begin{cases} 1.5x_1 - 0.8x_2 + 4.25x_3 = 5.1 \\ 1.2x_1 + 7.18x_2 - 3.2x_3 = 4.2 \\ 0.5x_1 - 1.5x_2 + 7.1x_3 = -1.2 \end{cases}$	2	$\begin{cases} 6.7x_1 - 0.6x_2 + 0.83x_3 = 6.8 \\ 0.8x_1 + 1.1x_2 - 7.2x_3 = 5.2 \\ 1.2x_1 - 5.4x_2 + 0.54x_3 = -3.2 \end{cases}$
3	$\begin{cases} -1.32x_1 + 2.15x_2 + 7.6x_3 = -1.4 \\ 2.62x_1 + 6.1x_2 - 4.12x_3 = 5.6 \\ 8.3x_1 - 2.84x_2 - 1.5x_3 = -6.5 \end{cases}$	4	$\begin{cases} 0.51x_1 - 10.2x_2 - 3.62x_3 = -2.05 \\ 3.09x_1 + 1.23x_2 - 4.64x_3 = -5.6 \\ 3.2x_1 - 2.31x_2 - 8.4x_3 = 6.1 \end{cases}$
5	$\begin{cases} 7.12x_1 - 6.66x_2 + 2.6x_3 = -3.1 \\ -1.76x_1 + 6.5x_2 - 0.87x_3 = 2.85 \\ 0.65x_1 + 0.87x_2 - 8.7x_3 = 5.56 \end{cases}$	6	$\begin{cases} 6.4x_1 - 0.73x_2 + 2.1x_3 = 3.8 \\ -1.07x_1 + 3.8x_2 - 1.5x_3 = -1.2 \\ 2.7x_1 - 3.1x_2 + 4.2x_3 = -7.5 \end{cases}$
7	$\begin{cases} 9.21x_1 - 1.84x_2 + 0.7x_3 = -3.2 \\ -6.17x_1 + 8.5x_2 - 2.87x_3 = -3.75 \\ 0.7x_1 + 0.87x_2 - 8.7x_3 = 2.64 \end{cases}$	8	$\begin{cases} 4.3x_1 - 1.2x_2 + 10.3x_3 = 4.2 \\ 0.21x_1 + 6.2x_2 + 3.54x_3 = 5.1 \\ -0.31x_1 - 0.52x_2 + 3.6x_3 = -2.1 \end{cases}$
9	$\begin{cases} 6.9x_1 + 2.3x_2 + 1.21x_3 = 3.1 \\ -x_1 + 2.3x_2 - 3.4x_3 = -2.3 \\ 0.21x_1 + 0.43x_2 - 6.3x_3 = 3.6 \end{cases}$	10	$\begin{cases} 12.4x_1 - 0.56x_2 + 4.2x_3 = 6.3 \\ -0.65x_1 + 4.4x_2 + 1.5x_3 = 1.5 \\ 1.5x_1 + 2.1x_2 - 2.8x_3 = 1.7 \end{cases}$

Таблица 8.4 (окончание)

№	Система уравнений	№	Система уравнений
11	$\begin{cases} 1.2x_1 + 1.06x_2 + 6.7x_3 = 2.12 \\ 4.2x_1 - 6.3x_2 - 0.9x_3 = -1.1 \\ 0.6x_1 + 6.8x_2 - 0.82x_3 = 0.83 \end{cases}$	12	$\begin{cases} 9.7x_1 + 0.35x_2 - 1.84x_3 = 2.15 \\ 4.64x_1 - 7.1x_2 - 4.3x_3 = 1.5 \\ 0.32x_1 + 3.48x_2 - 3.3x_3 = -3.1 \end{cases}$
13	$\begin{cases} 6.5x_1 + 2.34x_2 + 1.4x_3 = 2.8 \\ 0.5x_1 + 7.3x_2 - 2.4x_3 = -3.8 \\ 8.6x_1 + 0.34x_2 - 6.4x_3 = 0.64 \end{cases}$	14	$\begin{cases} 2.8x_1 + 4.3x_2 - 3.7x_3 = 5.1 \\ -0.45x_1 + 8.24x_2 + 4.8x_3 = 5.4 \\ 0.54x_1 + 2.3x_2 + 3.7x_3 = 1.54 \end{cases}$
15	$\begin{cases} 6x_1 + 0.13x_2 - 0.67x_3 = 1.9 \\ 3.8x_1 + 1.25x_2 - 4.3x_3 = 6.4 \\ 0.38x_1 - 0.64x_2 - 3.2x_3 = 5.4 \end{cases}$	16	$\begin{cases} 1.5x_1 - 2.6x_2 + 7x_3 = -11.2 \\ 6.6x_1 + 1.3x_2 - 1.24x_3 = 5.3 \\ 0.85x_1 - 8.4x_2 + 4.7x_3 = 1.6 \end{cases}$
17	$\begin{cases} 6.2x_1 - 0.52x_2 + 2.3x_3 = -1.8 \\ -4.2x_1 + 3.4x_2 - 0.5x_3 = 0.7 \\ 0.2x_1 + 0.8x_2 + 3.6x_3 = 3.2 \end{cases}$	18	$\begin{cases} 0.63x_1 - 0.54x_2 + 1.7x_3 = 3.6 \\ 0.65x_1 + 4.4x_2 + 0.15x_3 = 2.3 \\ 1.5x_1 + 0.2x_2 + 4.1x_3 = 2.8 \end{cases}$
19	$\begin{cases} 8.4x_1 - 0.25x_2 + 3.1x_3 = -5.7 \\ -0.3x_1 + 6.1x_2 - 1.54x_3 = 3.3 \\ -6.8x_1 + 1.2x_2 - 7x_3 = 4.5 \end{cases}$	20	$\begin{cases} 12x_1 + 4.2x_2 - 0.8x_3 = -5.4 \\ -4.1x_1 + 2.2x_2 - 0.16x_3 = 1.6 \\ -1.6x_1 - 4.3x_2 + 8.4x_3 = 12.2 \end{cases}$

Таблица 8.5. Системы нелинейных уравнений

№	Система уравнений	Начальные приближения	№	Система уравнений
1	$\begin{cases} \sin(x_1 + x_2) - 1.24x_1 = 0.1 \\ x_1^2 + x_2^2 = 1 \end{cases}$	$\begin{aligned} x_1 &= 0.74 \\ x_2 &= 0.67 \end{aligned}$	2	$\begin{cases} \sin(x_2 + 1) - x_1 = 1.2 \\ 2x_2 + \cos x_1 = 2 \end{cases}$
3	$\begin{cases} \operatorname{tg}(x_1 x_2 + 0.2) = x_1^2 \\ 0.6x_1^2 + 2x_2^2 = 1 \end{cases}$	$\begin{aligned} x_1 &= 0.88 \\ x_2 &= 0.52 \end{aligned}$	4	$\begin{cases} \cos(x_2 - 1) + x_1 = 0.5 \\ x_2 - \cos x_1 = 3 \end{cases}$
5	$\begin{cases} \sin x_1 + 2 \sin x_2 = 1 \\ 2 \sin 3x_1 + 3 \sin 3x_2 = 0.3 \end{cases}$	$\begin{aligned} x_1 &= 1.08 \\ x_2 &= 0.06 \end{aligned}$	6	$\begin{cases} x_1^2 x_2 - x_2 - 9 = 0 \\ x_1 x_2 - x_1^2 + 10 = 0 \end{cases}$
7	$\begin{cases} \operatorname{tg}(x_1 - x_2) - 4x_1 = 0 \\ x_1^2 + 2x_2^2 = 1 \end{cases}$	$\begin{aligned} x_1 &= -0.5 \\ x_2 &= 0.6 \end{aligned}$	8	$\begin{cases} x_1^2 + x_2^2 - x_1 x_2 = 0 \\ \sin(x_1 + x_2) - 2.4x_1 + 3.2 = 0 \end{cases}$
9	$\begin{cases} x_1^4 + x_2^2 - 3 = 0 \\ x_1^3 + x_2^3 - 4 = 0 \end{cases}$	$\begin{aligned} x_1 &= 0.95 \\ x_2 &= 1.4 \end{aligned}$	10	$\begin{cases} 2x_1 x_2^2 - 4x_2 - 7.5 = 0 \\ x_1^2 - 3x_1 x_2 + 4.5 = 0 \end{cases}$
11	$\begin{cases} \sin x_1 - x_2 = 1.3 \\ \cos x_2 - x_1 = -0.82 \end{cases}$	$\begin{aligned} x_1 &= 1.8 \\ x_2 &= -0.35 \end{aligned}$	12	$\begin{cases} \sin x + 2 \cos x - 0.8 = 0 \\ x_1 x_2^2 + 3x_1 - 4.5 = 0 \end{cases}$

Таблица 8.5 (окончание)

№	Система уравнений	Начальные приближения	№	Система уравнений
13	$\begin{cases} x_1^3 + x_2^3 - 6x_1 + 3 = 0 \\ x_1^3 - x_2^3 - 6x_2 = 2 \end{cases}$	$\begin{aligned} x_1 &= 0.52 \\ x_2 &= -0.37 \end{aligned}$	14	$\begin{cases} x_1^2 x_2^2 - x_2^2 - 18.75 = 0 \\ x_1 + x_2^2 - 8.25 = 0 \end{cases}$
15	$\begin{cases} x_2 + e^{x_1 - x_2} = 0 \\ x_1 + e^{x_1 + x_2} = 0 \end{cases}$	$\begin{aligned} x_1 &= -0.7 \\ x_2 &= -0.35 \end{aligned}$	16	$\begin{cases} \operatorname{tg}(x_1 - x_2) - 4x_1 = 0 \\ x_1^2 + 3x_2^3 - 4 = 0 \end{cases}$
17	$\begin{cases} x_1^2 x_2 - 8x_1 + 5.5 = 0 \\ x_1 x_2 + 3x_2 - 10 = 0 \end{cases}$	—	18	$\begin{cases} x_1^2 \sin x_2 + x_2^2 \sin x_1 + 1 = 0 \\ 2x_1 + e^{(x_1 + x_2)} - 4 = 0 \end{cases}$
19	$\begin{cases} e^{x_1} + 2x_2 \ln x_1 - 3 = 0 \\ x_1^2 x_2 - 3x_1 + 5.4 = 0 \end{cases}$	—	20	$\begin{cases} \sin x_1 + 3.5 \sin x_2 - 1 = 0 \\ 2 \sin 3x_1 + 3 \sin 2x_2 - 0.4 = 0 \end{cases}$

Сравнить результаты этих методов, указав их достоинства и недостатки.

Примечание

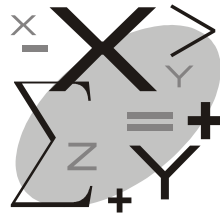
При решении системы уравнений методом итераций преобразуйте исходную систему к виду, когда итерационный процесс сходится. Решите задачу при точности определения неизвестных 3, 10 и 20 знаков после запятой. Объясните результаты решения.

Задание 2. Решение систем нелинейных алгебраических уравнений

Решить нелинейные системы уравнений методами Ньютона и итераций (табл. 8.5).

Примечание

В уравнениях с нечетными номерами указаны значения начальных приближений (кроме № 17 и 19). В уравнениях с четными номерами начальные приближения должны быть определены учащимися. Так как система уравнений имеет второй порядок, то области начальных приближений легко найти графическим способом.



ГЛАВА 9

Решение дифференциальных уравнений

9.1. Формулировка задачи

Представим дифференциальное уравнение первого порядка в следующем виде:

$$y' = f(x), \quad (9.1)$$

и пусть $y(x_0) = y_0$ — начальные условия его решения.

Тогда решением уравнения является функция $y = \varphi(x)$, которая, будучи подставленной в исходное уравнение, обратит его в тождество и одновременно будут выполняться начальные условия. Эта задача в математике называется *задачей Коши*.

Задача Коши при решении дифференциального уравнения n -го порядка

$$y^{(n)} = f(x, y, y', \dots, y^{(n-1)}) \quad (9.2)$$

формулируется аналогично, при этом должны быть удовлетворены следующие n начальных условий:

$$\begin{aligned} y(x_0) &= y_0, \\ y'(x_0) &= y'_0, \\ &\dots\dots\dots \\ y^{(n-1)}(x_0) &= y_0^{(n-1)}. \end{aligned} \quad (9.3)$$

При численных методах решения дифференциального уравнения n -го порядка уравнение представляется в виде системы дифференциальных уравнений первого порядка, разрешенных относительно производной. Такое представление осуществляется введением новых переменных.

Обозначим $y' = y_1$, $y_1' = y_2$, ..., $y_{n-2}' = y_{n-1}$ и подставим новые переменные в исходное дифференциальное уравнение.

Получим следующую систему дифференциальных уравнений:

$$\begin{aligned} y' &= y_1, \\ y_1' &= y_2, \\ &\dots\dots\dots \\ y_{n-2}' &= y_{n-1} \\ y^{(n)} &= f(x, y, y_1, \dots, y_{n-1}). \end{aligned} \tag{9.4}$$

Общим решением дифференциального уравнения (9.1) является функция $y = \varphi(x, c)$, где c — произвольная постоянная, определяемая начальными условиями. Если удастся найти общее решение, то решение задачи Коши сводится к нахождению произвольной постоянной c . В большинстве практических задач общее решение найти не удастся. Поэтому дифференциальные уравнения решают приближенными методами.

Существуют две группы приближенных методов: аналитические и численные. Рассмотрим кратко эти методы.

9.2. Приближенные аналитические методы решения дифференциальных уравнений

9.2.1. Метод последовательного дифференцирования

Решение дифференциального уравнения n -го порядка в виде (9.2) при начальных условиях (9.3) можно разложить в ряд Тейло-

ра. Можно доказать, что при определенных условиях ряд Тейлора является приближенным решением исходного дифференциального уравнения.

Ряд Тейлора функции $y = \varphi(x)$ имеет вид:

$$\begin{aligned} y(x) = & y(x_0) + \frac{y'(x_0)}{1!}(x-x_0) + \frac{y''(x_0)}{2!}(x-x_0)^2 + \dots + \\ & + \frac{y^{(n-1)}(x_0)}{(n-1)!}(x-x_0)^{n-1} + \\ & + \frac{y^n(x_0)}{n!}(x-x_0)^n + \frac{y^{n+1}(x_0)}{(n+1)!}(x-x_0)^{n+1} + \dots + \\ & + \frac{y^k(x_0)}{k!}(x-x_0)^k + \dots \end{aligned} \quad (9.5)$$

Из выражения (9.5) видно, что n его первых членов $y(x_0)$, $y'(x_0)$, ..., $y^{(n-1)}(x_0)$ известны, т. к. они являются начальными условиями решения задачи. Это означает, что искомое приближенное решение дифференциального уравнения может быть записано в виде степенного ряда степени $n-1$ непосредственно (не решая уравнения) по известным начальным условиям.

Для определения остальных членов степенного ряда (9.5) необходимо знать производные $y^{(n)}(x_0)$, $y^{(n+1)}(x_0)$, ..., $y^{(k)}(x_0)$, ...

Производная $y^{(n)}(x_0)$ вычисляется из исходного дифференциального уравнения (9.2), если подставить в его правую часть значения x , y , y' , ..., $y^{(n-1)}$ из начальных условий. Производные $y^{(n+1)}(x_0)$, ..., $y^{(k)}(x_0)$, ... можно вычислить путем последовательного дифференцирования исходного уравнения (9.2) и вычисления правых частей производных в точках

$$x = x_0,$$

$$y = y_0,$$

$$\begin{aligned}
 y' &= y'_0, \\
 &\dots\dots\dots \\
 y^{(n)} &= y_0^{(n)}, \\
 &\dots\dots\dots \\
 y^{(k-1)} &= y_0^{(k-1)}.
 \end{aligned}$$

Число членов степенного ряда (9.5) определяется из условия требуемой точностью решения дифференциального уравнения.

9.2.2. Метод неопределенных коэффициентов

Сущность метода покажем на примере решения дифференциального уравнения второго порядка вида:

$$y'' + P(x)y' + Q(x)y = R(x) \quad (9.6)$$

при начальных условиях $y(x_0) = y_0$, $y'(x_0) = y'_0$.

Пусть следующий степенной ряд является решением дифференциального уравнения (9.6):

$$y(x) = \sum_{n=0}^{\infty} c_n x^n. \quad (9.7)$$

Тогда неизвестными решения являются коэффициенты c_n . Для их определения разложим в степенной ряд функции $P(x)$, $Q(x)$, $R(x)$:

$$\begin{aligned}
 P(x) &= \sum_{n=0}^{\infty} p_n x^n, \\
 Q(x) &= \sum_{n=0}^{\infty} q_n x^n, \\
 R(x) &= \sum_{n=0}^{\infty} r_n x^n.
 \end{aligned} \quad (9.8)$$

Продифференцируем дважды искомое решение (9.7):

$$y'(x) = \sum_{n=1}^{\infty} n c_n x^{n-1}, \quad (9.9)$$

$$y''(x) = \sum_{n=2}^{\infty} n(n-1) c_n x^{n-2}. \quad (9.10)$$

Подставим теперь (9.7), (9.8), (9.9), (9.10) в исходное уравнение (9.6). В результате получим:

$$\begin{aligned} \sum_{n=2}^{\infty} n(n-1) c_n x^{n-2} + \sum_{n=0}^{\infty} p_n x^n \sum_{n=1}^{\infty} n c_n x^{n-1} + \sum_{n=0}^{\infty} q_n x^n \sum_{n=0}^{\infty} c_n x^n = \\ = \sum_{n=0}^{\infty} r_n x^n. \end{aligned} \quad (9.11)$$

Для определения коэффициентов c_n достаточно приравнять коэффициенты при одинаковых степенях x и решить систему алгебраических уравнений. Коэффициенты c_0 и c_1 при этом определяются из начальных условий. В выражении (9.11) находятся бесконечные степенные ряды. При решении практических задач ограничиваются определенным числом членов рядов из условий точности результатов.

9.2.3. Метод последовательных приближений

Приближенное решение дифференциального уравнения (9.1) можно найти по следующей рекуррентной формуле:

$$y_n(x) = y_0(x) + \int_{x_0}^x f(x, y_{n-1}) dx, \quad (9.12)$$

где $y_0(x)$ — начальное приближение, $f(x, y_{n-1})$ — правая часть исходного дифференциального уравнения, полученная на $(n-1)$ -м приближении.

В качестве начального приближения можно взять любую функцию, достаточно близкую к точному решению. В качестве начального приближения можно взять частичную сумму степенного ряда или начальное условие $y(x_0)$.

9.3. Численные методы решения дифференциальных уравнений

9.3.1. Метод Эйлера

Численное решение дифференциального уравнения первого порядка $y' = f(x, y)$ при начальных условиях $y(x_0) = y_0$ по методу Эйлера находится в виде табл. 9.1.

Таблица 9.1. Таблица решения дифференциального уравнения

x_0	x_1	x_2	...	x_n
y_0	y_1	y_2	...	y_n

Значения x_0, y_0 являются начальными условиями решения уравнения. Первая строка таблицы при известном x_0 и шаге h вычисляется по соотношениям:

$$x_1 = x_0 + h,$$

$$x_2 = x_1 + h,$$

.....

$$x_n = x_{n-1} + h.$$

Значения функции y_0, y_1, \dots, y_n вычисляются по рекуррентной формуле Эйлера.

Разложим функцию $y = \varphi(x)$, являющуюся решением исходного уравнения, в ряд Тейлора:

$$y(x) = y(x_0) + \frac{y'(x_0)}{1!}(x - x_0) + \frac{y''(x_0)}{2!}(x - x_0)^2 + \dots + \frac{y^{(n)}(x_0)}{n!}(x - x_0)^n.$$

Вычислим значение функции в точке $x = x_1 = x_0 + h$:

$$y(x) = y(x_0) + \frac{y'(x_0)}{1!}(x_1 - x_0) + \frac{y''(x_0)}{2!}(x_1 - x_0)^2 + \dots + \frac{y^{(n)}(x_0)}{n!}(x_1 - x_0)^n.$$

Ограничиваясь первыми двумя членами разложения, получим:

$$y(x_1) = y(x_0) + \frac{y'(x_0)}{1!}(x_1 - x_0).$$

Так как $x_1 - x_0 = h$, а $y'(x_0) = f(x_0, y_0)$, то

$$y(x_1) = y(x_0) + hf(x_0, y_0).$$

Принимая x_1, y_1 за начальные условия и приводя те же рассуждения, что при вычислении функции $y(x_1)$, получим:

$$y(x_2) = y(x_1) + hf(x_1, y_1),$$

$$y(x_3) = y(x_2) + hf(x_2, y_2),$$

.....

$$y(x_n) = y(x_{n-1}) + hf(x_{n-1}, y_{n-1}).$$

Из этих выражений видно, что значения y_1, y_2, \dots, y_n табл. 9.1 могут быть вычислены по следующей рекуррентной формуле Эйлера:

$$y_{i+1} = y_i + hf(x_i, y_i). \quad (9.13)$$

Из вывода формулы (9.13) следует, что метод Эйлера основан на линейном разложении функции $f(x, y)$ на участке h с последующей экстраполяцией решения за пределы этого участка.

Ошибка метода возникает за счет отбрасывания членов ряда Тейлора и имеет порядок h^2 .

Метод Эйлера позволяет решать системы уравнений и дифференциальные уравнения высокого порядка.

9.3.2. Усовершенствованные методы Эйлера

Метод Эйлера дает большие погрешности, возрастающие от итерации к итерации. Усовершенствованные методы Эйлера дают решение с большей точностью. Все они относятся к так называемым *методам прогноза и коррекции*.

Метод Эйлера — Коши

По этому методу дифференциальное уравнение первого порядка вида $y' = f(x, y)$ при начальных условиях $y(x_0) = y_0$ решается по следующей рекуррентной формуле:

$$y_{i+1} = y_i + h \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2}, \quad (9.14)$$

где y_{i+1} вычисляется по методу Эйлера, т. е.

$$y_{i+1} = y_i + hf(x_i, y_i).$$

Погрешность метода — порядка h^3 .

Усовершенствованный метод Эйлера

По этому методу дифференциальное уравнение первого порядка решается по формуле:

$$y_{i+1} = y_i + hf \left(x_{i+\frac{1}{2}}, y_{i+\frac{1}{2}} \right), \quad (9.15)$$

где

$$x_{i+\frac{1}{2}} = x_i + \frac{h}{2}, \quad y_{i+\frac{1}{2}} = y_i + \frac{h}{2} f(x_i, y_i).$$

Погрешность этого метода такая же, как и усовершенствованного метода Эйлера — Коши.

Усовершенствованный метод Эйлера — Коши с итерационной обработкой результатов

По этому методу дифференциальное уравнение первого порядка решается по формуле:

$$y_{i+1}^{(k)} = y_i + h \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1}^{(k-1)})}{2}, \quad k = 1, 2, \dots \quad (9.16)$$

В формуле (9.16) $y_{i+1}^{(k-1)}$ является начальным приближением итерационного процесса и определяется по формуле Эйлера, т. е.

$$y_{i+1}^0 = y_i + hf(x_i, y_i).$$

Признаком окончания итерационного процесса является условие:

$$|y_{i+1}^k - y_{i+1}^{k-1}| < \varepsilon. \quad (9.17)$$

Из выражения (9.16) и условия (9.17) видно, что идея метода состоит в следующем. По формуле Эйлера (9.13) находится решение на $(i+1)$ -м шаге, которое является на этом шаге первым приближением. Данное приближение используется для образования итерационного процесса на основе формулы усовершенствованного метода Эйлера — Коши. Признаком окончания итерационного процесса является условие (9.17). Погрешность метода — порядка h^3 на каждом шаге итераций.

9.3.3. Метод Рунге — Кутты

Метод Рунге — Кутты обладает более высокой точностью, чем методы Эйлера за счет уменьшения методических ошибок. Идея метода состоит в следующем.

По методу Эйлера решение дифференциального уравнения первого порядка определяется из соотношения:

$$y_{i+1} = y_i + \Delta y_i,$$

где

$$\Delta y_i = hf(x_i, y_i) = hy'(x_i, y_i).$$

Тогда приращение Δy_i может быть найдено путем интегрирования:

$$\Delta y_i = \int_{x_i}^{x_{i+1}} y'(x, y) dx = \int_{x_i}^{x_{i+1}} f(x, y) dx.$$

Или окончательно

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx.$$

Вычислим теперь интеграл по методу прямоугольников:

$$y_{i+1} = y_i + (x_{i+1} - x_i) f(x_i, y_i) = y_i + hf(x_i, y_i).$$

Из полученного выражения видно, что вычисление интеграла по методу прямоугольников приводит к формуле Эйлера.

Вычислим интеграл по формуле трапеций:

$$y_{i+1} = y_i + \frac{h}{2} (f(x_i, y_i) + f(x_{i+1}, y_{i+1})).$$

Из выражения видно, что оно совпадает с расчетной формулой усовершенствованного метода Эйлера — Коши.

Для получения более точного решения дифференциального уравнения следует воспользоваться более точными методами вычисления интеграла.

В методе Рунге — Кутты искомый интеграл представляется в виде следующей конечной суммы:

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y) dx = y_i + \sum_{i=1}^q p_i K_i(h), \quad (9.18)$$

где p_i — некоторые числа, зависящие от q ; $K_i(h)$ — функции, зависящие от вида подынтегральной функции $f(x, y)$ и шага интегрирования h ; они вычисляются по следующим формулам:

$$K_1(h) = hf(x, y),$$

$$K_2(h) = hf(x + \alpha_2 h, y + \beta_{21} K_1(h)),$$

$$K_3(h) = hf(x + \alpha_3 h, y + \beta_{31} K_1(h) + \beta_{32} K_2(h)),$$

.....

$$K_q(h) = hf(x + \alpha_q h, y + \beta_{q1} K_1(h) + \dots + \beta_{q,q-1} K_{q-1}(h)).$$

Значения p , α , β получают из соображений высокой точности вычислений.

Формулы Рунге — Кутты третьего порядка ($q = 3$) имеют следующий вид:

$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 4K_2 + K_3),$$

$$K_1 = hf(x_i, y_i),$$

$$K_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{K_1}{2}\right),$$

$$K_3 = hf(x_i + h, y_i + K_1 + 2K_2).$$

Наиболее часто используется метод Рунге — Кутты четвертого порядка, для которого расчетные формулы имеют вид:

$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4),$$

$$K_1 = hf(x_i, y_i),$$

$$K_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{K_1}{2}\right), \quad (9.19)$$

$$K_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{K_2}{2}\right),$$

$$K_4 = hf(x_i + h, y_i + K_3).$$

Формулы Рунге — Кутты имеют погрешности порядка h^{q+1} . Погрешность метода Рунге — Кутты четвертого порядка имеет порядок h^5 .

9.4. Компьютерные технологии решения дифференциальных уравнений

Система MATLAB позволяет решать дифференциальные уравнения и системы высокого порядка с табличным и графическим представлением результатов.

Функциями решения дифференциальных уравнений являются `ode23()` и `ode45()`, имеющие вид:

```
[t, x]=ode23('fun', t0, tf, x0)
[t, x]=ode45('fun', t0, tf, x0)
[t, x]=ode23('fun', t0.tf.x0, tol, trace)
[t, x]=ode45('fun', t0.tf.x0, tol, trace)
```

где:

- ◆ 'fun' — имя m-файла, в котором содержатся правые части системы дифференциальных уравнений;
- ◆ `t0` — начальное значение аргумента;
- ◆ `tf` — конечное значение аргумента;
- ◆ `x0` — вектор начальных условий;
- ◆ `tol` — задаваемая точность, по умолчанию, для `ode23()` — $1.e-3$, для `ode45()` — $1.e-6$;
- ◆ `trace` — выдача промежуточных результатов.

Наиболее часто дифференциальными уравнениями описываются процессы, протекающие во времени. Тогда переменной интегрирования является время.

Функции `ode()` реализуют численные методы Рунге — Кутты третьего, четвертого, пятого и шестого порядков с автоматическим выбором шага.

Технология решения дифференциальных уравнений в системе MATLAB такова:

1. Создание новой функции, представляющей собой m-файл вычисления правых частей системы дифференциальных уравнений.
2. Ввод функции `ode()`.
3. Получение решения нажатием клавиши <Enter>.

На экране получим решение в виде таблицы с числом столбцов равном числу неизвестных.

Система позволяет получить график функции. Для этого необходимо воспользоваться функцией `plot[t,x]`.

Пример 9.1

Необходимо определить вероятности состояний многоканальной системы массового обслуживания (СМО) с отказами. Задача формулируется следующим образом. Система массового обслуживания состоит из трех обслуживающих органов. Интенсивность потока заявок на обслуживание $a = 0.1$ 1/ час, интенсивность обслуживания заявок равна $m = 1.2$ 1/час. Необходимо определить вероятности всех состояний системы.

Число состояний системы в данном случае равно четырем:

- ♦ 1 — система свободна от обслуживания;
- ♦ 2 — один обслуживающий орган занят обслуживанием, остальные свободны;
- ♦ 3 — два обслуживающих органа заняты, один свободен;
- ♦ 4 — все три обслуживающих органа заняты, очередной заявке будет отказано в обслуживании.

Функционирование системы можно описать следующей системой дифференциальных уравнений:

$$\frac{dP_1(t)}{dt} = -aP_1(t) + mP_2(t),$$

$$\frac{dP_2(t)}{dt} = aP_1(t) - (a+m)P_2(t) + 2mP_3(t),$$

$$\frac{dP_3(t)}{dt} = aP_2(t) - (a+2m)P_3(t) + 3mP_4(t),$$

$$\frac{dP_4(t)}{dt} = aP_3(t) - 3mP_4(t).$$

В системе уравнений P_i — вероятность i -го состояния системы, $i = 1, 2, 3, 4$.

Будем анализировать функционирование системы, начиная с состояния, когда система свободна от обслуживания. В этом случае начальными условиями решения системы дифференциальных уравнений будут:

$$P_1(0) = 1,$$

$$P_2(0) = P_3(0) = P_4(0) = 0.$$

Будем решать систему уравнений по описанной выше технологии.

1. Создание m-файла функции вычисления правых частей дифференциальных уравнений.

Пусть имя файла — `sisdu.m`, тогда функция может иметь следующий вид:

```
Function CMO=sisdu(t,p)
y1=-0.1.*P(1)+1.2.*P(2);
y2=0.1.*P(1)-1.3.*P(2)+2.4.*P(3);
y3=0.1.*P(2)-2.5.*P(3)+3.6.*P(4);
y4=0.1.*P(3)-3.6.*P(4);
CMO=[y1,y2,y3,y4];
```

2. Ввод функции `ode()`:

```
>> t0=0; tf=0;x0=[1,0,0,0];
>> [t,P]=ode23('sisdu',t0,tf,x0)
```

3. Решение.

После нажатия клавиши <Enter> получим решение в виде следующей таблицы:

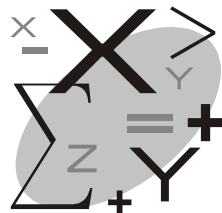
t	P (1)	P (2)	P (3)	P (4)
0	1	0	0	0
0.0008	0.9999	0.0001	0.0000	0.0000
0.0048	0.9995	0.0005	0.0000	0.0000
0.0248	0.9976	0.0024	0.0000	0.0000
0.0907	0.9914	0.0085	0.0000	0.0000
0.1858	0.9885	0.0164	0.0001	0.0000
.....				
2.2571	0.9210	0.0758	0.0031	0.0001
3.6429	0.9202	0.0765	0.0032	0.0001
4.7109	0.9201	0.0767	0.0032	0.0001
5.9984	0.9200	0.0767	0.0032	0.0001

При желании можно воспроизвести решение в виде графика, для этого нужно воспользоваться функцией `plot()`:

```
>> plot(t,P)
```

Решение системы дифференциальных уравнений позволяет сделать ряд важных выводов:

- ◆ время переходного процесса в системе мало, оно равно 6 часам;
- ◆ система обладает высокой готовностью в любой момент времени принять заявку на обслуживание;
- ◆ вероятность обслуживания заявки в любой произвольный момент времени равна $P(1) + P(2) + P(3) = 0.9999$.



ГЛАВА 10

Алгоритмы и технологии вычисления интегралов

10.1. Методы и алгоритмы вычисления интегралов

Система MATLAB позволяет вычислять неопределенные и определенные интегралы, первообразные которых заданы в виде аналитических выражений. Она также имеет большое число способов численного интегрирования.

Численное интегрирование необходимо в следующих случаях:

- ♦ первообразная не выражается через элементарные функции;
- ♦ аналитическое выражение интеграла слишком сложное;
- ♦ подынтегральная функция задана в табличной форме или в виде матрицы.

При вычислениях интегралов численными методами подынтегральную функцию целесообразно представлять в наиболее простом виде. Это может ускорить вычисления. Упрощение подынтегральной функции можно выполнить, воспользовавшись функциями `simplify()` и `factor()`. Имеют место случаи, когда система до упрощения не может вычислить неопределенный интеграл и легко его определяет после упрощения.

Метод вычисления интеграла выбирает пользователь. В этом особенность системы MATLAB. С помощью MATLAB студент имеет возможность сравнивать различные методы численного интегрирования.

Существует ряд способов численного интегрирования. Во всех таких способах вычисление осуществляется по приближенным формулам, называемым *квадратурными*. Приведем некоторые из них.

10.1.1. Формулы прямоугольников

Формулы прямоугольников представляются в следующем виде:

$$\int_a^b f(x) dx = \left\{ \begin{array}{l} h \sum_{k=0}^{n-1} y_k \\ h \sum_{k=1}^n y_k \end{array} \right\}, \quad (10.1)$$

где:

- ◆ h — шаг интегрирования;
- ◆ y_k — значение подынтегральной функции при аргументе x_k , $k = 0, 1, 2, \dots, n$;
- ◆ $n = \frac{b-a}{h}$ — число частей, на которые разбивается область интегрирования a, b .

Одна из формул дает значение интеграла с избытком, другая — с недостатком. Какая из них выдает решение с избытком или с недостатком, зависит от вида подынтегральной функции.

10.1.2. Формула трапеций

Эта формула имеет вид:

$$\int_a^b f(x) dx = h \left(\frac{y_0}{2} + \sum_{k=1}^{n-1} y_k + \frac{y_n}{2} \right), \quad (10.2)$$

где:

- ◆ y_0 — значение подынтегральной функции при $x = a$;
- ◆ y_n — значение подынтегральной функции при $x = b$;
- ◆ h — шаг интегрирования.

10.1.3. Формула парабол (Симпсона)

Эта формула имеет вид:

$$\int_a^b f(x)dx = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + 4y_5 + \dots + 4y_{n-1} + y_n). \quad (10.3)$$

В этой формуле ординаты с нечетными индексами умножаются на 4, а с четными — на 2. Предполагается, что n — число четное. При нечетном n формула имеет вид:

$$\int_a^b f(x)dx = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + 4y_5 + \dots + 2y_{n-1} + y_n).$$

Крайние ординаты имеют коэффициент, равный 1.

Существует много других квадратурных формул вычисления интегралов: Котеса, Чебышева, Гаусса и др.

Вычисление интеграла с помощью квадратурных формул требует знания числа n области интегрирования $[a, b]$. Значение n выбирается из условий требуемой точности вычисления интеграла. Зависимости погрешности ε от n в общем случае не существует. Поэтому для обеспечения точности программы вычисления интегралов используют следующий алгоритм. Определенный интеграл вычисляется дважды: с произвольными шагами h и $\frac{h}{2}$. Если

при этом разность интегралов мала, то вычислительный процесс заканчивается, а значением интеграла считается то, которое вычислено с шагом $\frac{h}{2}$, в противном случае шаг уменьшается вдвое

и сравниваются значения интеграла с шагами $\frac{h}{2}$ и $\frac{h}{4}$ и т. д.

Вычисление интеграла численными методами посредством MATLAB может оказать большую помощь учащемуся при изучении численных методов и программирования. Применяя

MATLAB, пользователь может анализировать подынтегральную функцию, что необходимо в случае, когда разработанная программа не дает возможности вычислить интеграл с требуемой точностью (например, при наличии разрывов непрерывности подынтегральной функции). Сравнение результатов, полученных с помощью программы учащегося и с помощью MATLAB, позволяет убедиться в достоверности решения и отсутствии ошибок в программе учащегося.

10.2. Численные методы вычисления интеграла в системе MATLAB

В системе MATLAB вычисление интегралов реализовано численными методами трапеций, парабол (Симпсона) и Ньютона — Котеса.

Опишем технологию вычисления этими методами.

10.2.1. Метод трапеций

Метод трапеций реализован в MATLAB несколькими функциями, приведенными ниже.

Функция

`cumtrapz(y)`

осуществляет вычисление интеграла в случае, когда значения функции y заданы в виде вектора или матрицы неограниченных размеров. Откликом этой функции является n интегралов, где n — число элементов вектора или число элементов в каждом столбце матрицы.

Интегралы вычисляются по методу трапеций, когда значения функции y состоят из одного элемента вектора (первого), двух элементов вектора (первого и второго) и т. д. до последнего интеграла, когда число значений y равно числу элементов вектора или элементов столбцов матрицы (количество элементов в каждом столбце одинаково).

Такое вычисление интеграла называется *интегрированием с накоплением*. Программа вычисляет интеграл с шагом $h=1$. Если

же ординаты получены с иным шагом, то результат вычислений интеграла нужно умножить на h .

Пример 10.1

Пусть функция $y(x)$ имеет значения, представленные в виде следующего вектора: $y=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$. Необходимо вычислить

$$\int_a^b y(x) dx.$$

При этом $a=1$, $b=1, 2, 3, \dots, 10$.

Функция вычисления интеграла методом трапеций будет иметь вид:

```
>> y=[1,2,3,4,5,6,7,8,9,10];
>> cumtrapz(y)
```

После нажатия клавиши <Enter> получим следующий ответ:

```
ans =
    0    1.5000    4.0000    7.5000   12.0000   17.5000   24.0000
   31.5000   40.0000   49.5000
```

Проверим правильность вычисления интеграла:

$$\int_1^1 1 dx = 0.$$

Далее воспользуемся формулой трапеций (10.2):

$$\int_1^2 y(x) dx = \frac{y_0}{2} + \frac{y_1}{2} = \frac{1}{2} + \frac{2}{2} = 1.5,$$

$$\int_1^3 y(x) dx = \frac{y_0}{2} + y_1 + \frac{y_2}{2} = \frac{1}{2} + 2 + \frac{3}{2} = 4,$$

$$\int_0^{10} y(x) dx = \frac{y_0}{2} + y_1 + y_2 + \dots + y_9 + \frac{y_{10}}{2} =$$

$$= \frac{1}{2} + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + \frac{10}{2} = 49.5.$$

Возможности этой функции весьма ограничены.

Действительно, пусть необходимо вычислить интеграл вида

$$\int_0^{10} (xe^{-x} + \ln x + 1) dx.$$

Чтобы вычислить этот интеграл с помощью функции `cumtrapz()`, следует сначала вычислить 10 ординат подынтегральной функции, представив их в виде вектора.

Программа вычисления интеграла с накоплением будет иметь вид:

```
>> x=1:1:10;
>> y=x*exp(x)+log(x)+1;
>> cumtrapz(y)
```

После нажатия клавиши <Enter> получим следующий ответ:

```
ans =
    1.0e+005
    0          0.0001    0.0005    0.0019    0.0067    0.0226
    0.0731    0.2307    0.7147    2.1806
```

Основным недостатком метода трапеций является большая погрешность результата вычисления интеграла.

Покажем это на предыдущем примере.

Неопределенный интеграл имеет следующее точное решение:

$$\int (xe^x + \ln x + 1) dx = e^x (x - 1) + x \ln x.$$

Тогда определенный интеграл с накоплением, в соответствии с методом Ньютона (разность значений интеграла при верхнем и нижнем пределах), будет иметь значения:

0	8.775	43.467	169.34	601.7	2027.89
6593.42	20883.34	64844.45	198261.22		

Результат существенно отличается от полученного с помощью функции `cumtrapz()`.

Пример 10.2

Функция $y(x)$ является матрицей.

```
>> y=[1,2,3;2,3,4;3,4,5;4,5,6];
>> cumtrapz(y)
ans =
```

0	0	0
1.5	2.5	3.5
4	6	8
7.5	10.5	13.5

В каждом столбце исходной матрицы получены значения интеграла с накоплением.

Функция *cumtrapz(x,y)*

Эта функция выполняет интегрирование с накоплением функции $y(x)$ также методом трапеций. При этом x и y могут быть либо векторами одной и той же размерности, либо x — это вектор-столбец, а y — матрица.

Откликом является значение интеграла с диапазоном значений x .

Пример 10.3

Функция $y(x)$ задана в виде следующих векторов:

```
x=[1,3,7,9,10], y=[1,3,5,7,9].
```

Необходимо вычислить методом трапеций значение интеграла с накоплением.

Решение:

```
>> x=[1,3,7,9,10];
>> y=[1,3,5,7,9];
>> cumtrapz(x,y)
```

После нажатия клавиши <Enter> получим:

```
ans =
    0    4   20   32   40
```

Здесь вычисления значений интеграла с накоплением на i -м шаге S_i осуществляется по формуле:

$$S_i = S_{i-1} + \left(\frac{y_i}{2} + \frac{y_{i+1}}{2} \right) \Delta x,$$

где:

- ◆ S_{i-1} — предыдущее значение интеграла;
- ◆ y_i, y_{i+1} — значения функции в начале и в конце интервала Δx ;
- ◆ $[9; 10]$ — шаг интегрирования на участке $[y_i; y_{i+1}]$.

Для нашего примера вычисления имеют вид:

- ◆ на участке вектора $[1; 3]$:

$$S_1 = 0 + \left(\frac{1}{2} + \frac{3}{2} \right) \cdot 2 = 4;$$

- ◆ на участке вектора $[3; 7]$:

$$S_2 = S_1 + \left(\frac{3}{2} + \frac{5}{2} \right) \cdot 4 = 20;$$

- ◆ на участке вектора $[7; 9]$:

$$S_3 = 20 + \left(\frac{5}{2} + \frac{7}{2} \right) \cdot 2 = 32;$$

- ◆ на участке вектора $[9; 10]$:

$$S_4 = 32 + \left(\frac{7}{2} + \frac{9}{2} \right) \cdot 1 = 40.$$

Как видно из примера, функция `cumtrapz(x,y)` вычисляет интеграл с накоплением при переменном шаге h .

Пример 10.4

Функция $y(x)$ задана в виде матрицы:

```
y(x) = [1 3 5; 3 5 7; 4 6 8; 4 7 9; 5 7 10]
```

При этом аргумент x представляет собой вектор следующего вида: $x=[1,3,7,9,10]$.

Вычислить значения интеграла.

Программа и результат вычисления интеграла с помощью функции `cumtrapz(x,y)` имеют вид:

```
>> x=[1 3 7 9 10];
>> y=[1 3 5; 3 5 7; 4 6 8; 4 7 9; 5 7 10];
>> cumtrapz(x,y)
```

После нажатия клавиши <Enter> получим:

```
ans =
    0         0         0
    4         8        12
   18        30        42
   26        43        59
  30.5       50       68.5
```

Функция *trapz(y)*

Функция возвращает значение интеграла

$$\int_a^b y(x) dx .$$

При этом $y(x)$ может быть вектором или матрицей. Если $y(x)$ — матрица, то функция возвращает вектор значений интеграла каждого столбца матрицы.

Пример 10.5

Необходимо вычислить методом трапеций значение интеграла, если функция $y(x)$ является следующим вектором:

```
y=[1,3,5,7,9].
```

Решение:

```
>> y=[1 3 5 7 9];  
>> trapz(y)  
ans =  
      20
```

Пример 10.6

Пусть функция $y(x)$ является матрицей, тогда программа решения будет иметь вид:

```
>> y=[1 3 5; 3 5 7; 4 6 8; 4 7 9; 5 7 10];  
>> trapz(y)  
ans =  
      14      23     31.5
```

Функция $\text{trapz}(x,y)$

Вычисляет интеграл функции $y(x)$ по x методом трапеций. Аргумент и функция задаются в виде векторов или x — в виде вектора, а y — в виде матрицы.

Пример 10.7

Пусть аргумент x и функция $y(x)$ заданы в виде следующих векторов: $x=[1\ 3\ 7\ 9\ 10]$, $y=[1\ 3\ 5\ 7\ 9]$.

Необходимо вычислить интеграл методом трапеций.

Решение:

```
>> x=[1 3 7 9 10];  
>> y=[1 3 5 7 9];  
>> trapz(x,y)
```

После нажатия клавиши <Enter> получим ответ:

```
ans =  
    40
```

Пример 10.8

Пусть аргументом функции $y(x)$ является вектор $x=[1\ 3\ 7\ 9\ 10]$, а функция — матрицей $y= [1\ 3\ 5; 3\ 5\ 7; 4\ 6\ 8; 4\ 7\ 9; 5\ 7\ 10]$. Вычислим значение интеграла методом трапеций, используя функцию `trapz(x,y)`.

Решение:

```
>> x=[1 3 7 9 10];  
>> y=[1 3 5; 3 5 7; 4 6 8; 4 7 9; 5 7 10];  
>> trapz(x,y)  
ans =  
    30.5    50    68.5
```

Функции `trapz(y)` и `trapz(x,y)` допускают аналитическое задание подынтегральной функции $y(x)$. Однако в этом случае аргументу x должны быть присвоены численные значения во всем диапазоне интегрирования.

Пример 10.9

Пусть подынтегральная функция имеет вид

$$y(x) = xe^x + \ln x + 1.$$

Необходимо вычислить определенный интеграл в диапазоне от 1 до 10 с шагом 0.5.

Решение:

```
>> x=1: 0.5: 10;  
>> y=x.*exp(x)+log(x)+1;  
>> trapz(y)  
ans=  
    4.0657e + 005
```

```
>> trapz(x,y)
ans =
    2.0328e + 005
```

Следует иметь в виду, что при вычислении интеграла с помощью функции `trapz(x,y)` его значение зависит от шага интегрирования.

10.2.2. Численное интегрирование с помощью квадратурных формул

Метод парабол (Симпсона)

Вычисление интеграла *методом парабол* осуществляется по формуле (10.3). Для его реализации в системе MATLAB используются следующие функции:

```
quad('fun', a, b)
quad('fun', a, b, tol)
quad('fun', a, b, tol, trace)
ablquad('fun', a, b, c, d)
ablquad('fun', a, b, c, d, tol)
ablquad('fun', a, b, c, d, tol, trace)
```

В этих функциях приняты обозначения:

- ◆ 'fun' — подынтегральная функция, взятая в одинарные кавычки;
- ◆ a, b — пределы интегрирования;
- ◆ tol — относительная погрешность, задаваемая пользователем, по умолчанию $\text{tol}=10\text{e-}3$;
- ◆ c, d — пределы интегрирования по другой переменной (внешней) при вычислении двойного интеграла;
- ◆ trace — число, отличное от нуля, по которому система показывает ход вычислительного процесса.

Рассмотрим перечисленные функции и приведем примеры.

Функция *quad('fun',a,b)*

Функция вычисляет определенный интеграл

$$\int_a^b f(x) dx$$

с погрешностью, не превышающей 10^{-3} .

Подынтегральная функция $f(x)$ представляется в аналитическом виде с соблюдением правил записи функций в системе MATLAB.

Пример 10.10

Подынтегральная функция имеет вид:

$$f(x) = e^x + x^2 + 2 \sin x - 5.$$

Необходимо вычислить интеграл

$$\int_1^5 f(x) dx.$$

Решение:

```
>> y='exp(x)+x.^2+2*sin(x)-5';  
>> quad(y,1,5)
```

После нажатия клавиши <Enter> получим следующий ответ:

```
ans =  
    167.5415
```

Функция может быть представлена одной строкой:

```
>> quad('exp(x)+x.^2+2*sin(x)-5',1,5)
```

После нажатия клавиши <Enter> получим тот же ответ.

Функция *quad('fun',a,b,tol)*

В функции `quad('fun',a,b,tol)` параметр `tol` является желаемой погрешностью, которая представляется в виде $1e-n$. По умолчанию `tol=1.e-3`.

Пример 10.11

Пусть подынтегральная функция имеет вид:

$$f(x) = e^x + x^2 + 2\sin x - 5.$$

Необходимо вычислить интеграл в диапазоне от 1 до 5 с погрешностью не выше 10^{-7} .

Решение:

```
>> quad('exp(x)+x.^2+2*sin(x)-5',1,5,1e-7)
```

После нажатия клавиши <Enter> получим ответ:

```
ans =  
167.5415
```

Ответ тот же, что и в примере 10.10, когда вычисление интеграла выполнялось с погрешностью не более 10^{-3} . Это объясняется тем, что в ответе содержится только 4 значащих цифры после запятой. Отличие можно будет видеть при большем числе цифр после запятой.

Функция *ablquad('fun',a,b,c,d)*

В функции

```
ablquad('fun',a,b,c,d)
```

приняты следующие обозначения:

- ◆ 'fun' — это функция с двумя переменными;
- ◆ a, b — пределы по внутренней переменной;
- ◆ c, d — пределы по внешней переменной.

Пример 10.12

Функция двух переменных имеет вид:

$$z = x^2 + y^2 - 2.$$

Необходимо вычислить интеграл

$$\int_1^2 \int_0^3 z(x, y) dx dy.$$

Решение:

```
>> z='x.^2+y.^2-2';  
>> ablquad(z, 1, 2, 0, 3)
```

После нажатия клавиши <Enter> получаем ответ:

```
ans =  
      10
```

Решение получено с погрешностью, не превышающей 10^{-3} .

Повысим точность вычисления интеграла. Пусть $\text{tol}=1e-7$.

Тогда функция будет иметь вид:

```
>> ablquad(z, 1, 2, 0, 3, 1·e-7)
```

и после нажатия клавиши <Enter> получим:

```
ans =  
      10
```

Ответ прежний потому, что интеграл и полученный ответ является точным решением.

Функция *quad8('fun',a,b)*

Функция реализует метод Ньютона — Котеса восьмого порядка. Дает наиболее точное решение по сравнению с предыдущими методами и функциями. Имеет те же опции, что и функция `quad('fun',a,b, tol, trace)`.

Большое число методов и функций вычисления интегралов является достоинством системы MATLAB.

При изучении методов интегрирования студент имеет возможность сравнить методы по их точности.

Сравним точность методов на примере.

Пример 10.13

Пусть подынтегральная функция имеет вид:

$$y(x) = x^3 + x^2 e^{-x} + \sinh(x) - 12.$$

Необходимо вычислить интеграл в пределах от 0 до 10 методом трапеций, парабол (Симпсона) и Ньютона — Котеса. Использовать функции `trapz(x,y)`, `quad('fun',a,b)`, `quad8('fun',a,b)`. Точность вычисления интеграла — по умолчанию (10^{-3}).

Результаты вычисления интеграла приведены в табл. 10.1.

Таблица 10.1. Значения интеграла, вычисленные различными функциями

Функция	Значения интеграла
<code>trapz(x,y)</code>	1.3404×10^4 при $h = 1$
	1.3629×10^4 при $h = 0.5$
	1.3404×10^4 при $h = 0.1$
	1.3394×10^4 при $h = 0.01$
<code>quad('fun',0,10)</code>	1.3394×10^4
<code>quad8('fun',0,10)</code>	1.3394×10^4

Точное значение неопределенного интеграла имеет вид:

$$\frac{e^x}{2} - \frac{e^{-x}(2x^2 + 4x + 3)}{2} + \frac{x^4}{4} - 12x.$$

Численное значение интеграла, вычисленное по точной формуле в диапазоне от 0 до 10, равно 1.339422738×10^4 .

Из примера видно, что значение интеграла практически одно и то же.

Правда, в методе трапеций шаг интегрирования должен быть не менее $h = 0.01$.

10.3. Аналитические методы вычисления интеграла

10.3.1. Функция *int()* вычисления неопределенного и определенного интегралов

Вычисление интеграла аналитическими методами осуществляется в системе MATLAB с помощью функций `int()`, которые имеют вид:

```
int(y(x))  
int(y(x), a, b)
```

где:

- ◆ $y(x)$ — подынтегральная функция;
- ◆ a, b — пределы интегрирования.

Эти функции вычисляют:

- ◆ неопределенный интеграл;
- ◆ неопределенный интеграл с символьными переменными;
- ◆ определенный интеграл с символьными значениями пределов интегрирования;
- ◆ определенный интеграл от алгебраических функций;
- ◆ кратные интегралы;
- ◆ несобственные интегралы.

Технология вычисления интегралов достаточно проста и состоит в следующем:

1. Определение символьных переменных с помощью функции `syms`.
2. Ввод подынтегрального выражения с присвоением ему имени;
`y=f(x)`;
3. Ввод функции `int(y)`, если вычисляется неопределенный интеграл, или функции `int(y,a,b)`, если вычисляется определенный интеграл в пределах $[a; b]$.
4. Получение решения путем нажатия клавиши `<Enter>`.

Будем иллюстрировать методику на примерах решения задач.

Пример 10.14

Необходимо вычислить следующий интеграл:

$$\int \frac{x}{1+x^2} dx .$$

Решение:

```
>> syms x;  
>> y=x/(1+x^2);  
>> int(Y)  
ans =  
1/2*log(1+x^2)
```

Пример 10.15

Вычислить интеграл:

$$\int \frac{x}{a+bx^2} dx .$$

Здесь имеем случай, когда подынтегральная функция задана в аналитическом виде с символьными переменными.

Решение имеет вид:

```
>> syms x a b;  
>> y=x/(a+b*x^2);  
>> int(y)  
ans =  
1/(2*b)*log(a+b*x^2)
```

Пример 10.16

Вычислить значение определенного интеграла

$$\int_a^b \frac{x}{1+x^2} dx .$$

Здесь пределы интегрирования заданы в символьных переменных.

Решение:

```
>> syms x a b;
>> y=x/(1+x^2);
>> int(y,a,b)
ans =
    1/2*log(1+b^2)-1/2*log(1+a^2)
```

Пример 10.17

Вычислить определенный интеграл:

$$\int_a^b \frac{x}{c+dx^2} dx.$$

В данном случае подынтегральное выражение представлено в аналитическом виде, а пределы интегрирования — в виде символьных переменных. Это наиболее общий случай аналитического вычисления интеграла.

Решение представляется в следующем виде:

```
>> syms x a b c d;
>> y=x/(c+d*x^2);
>> int(y,a,b)
ans =
    1/2d*(log(c+b^2*d)-log(c+a^2*d))
```

Пример 10.18

Вычислить определенный интеграл

$$\int_1^5 \frac{x}{1+x^2} dx.$$

Решение:

```
>> syms x;
>> y=x/(1+x^2);
```

```
>> int(y,1,5)
ans =
    1/2*log(13)
```

Для получения решения в естественной форме достаточно активизировать с помощью мыши строку ответа и нажать клавишу <Enter>. Будет получен следующий ответ:

```
ans =
    1.2825
```

10.3.2. Вычисление кратных интегралов

Наиболее просто кратный интеграл вычислить путем интегрирования ответа, полученного от предыдущего значения интеграла.

Пример 10.19

Пусть необходимо вычислить двойной неопределенный интеграл вида:

$$\iint \frac{x}{1-x^2} dx.$$

Решение путем двойного интегрирования имеет вид:

```
>> syms x;
>> y=x/(1-x^2);
>> Z=int(y)
Z =
    -1/2*log(x-1)-1/2*log(x+1)
>> int(Z)
ans =
    -1/2*log(x-1)*(x-1)+x-1/2*log(x+1)*(x+1)
```

Программа будет иметь более простой вид, если функцию `int()` повторить n раз при n -кратном интегрировании.

Для нашего примера решение будет иметь вид:

```
>> syms x;
>> y=x/(1-x^2);
```

```
>> int(int(y))
ans =
-1/2*log(x-1)*(x-1)+x-1/2*log(x+1)*(x+1)
```

10.3.3. Вычисление несобственных интегралов

Вычисления интегралов с бесконечными пределами с помощью функции `int()` имеют особенности, которые увидим при решении примеров.

Пример 10.20

Необходимо вычислить следующий определенный интеграл:

$$\int_0^{\infty} \frac{x}{\sinh(5x)} dx.$$

Решение:

```
>> syms x;
>> y=x/sinh(5*x);
>> int(y,0,inf)
ans =
pi^2/100
```

А теперь вычислим тот же интеграл, введя символьную переменную a :

$$\int_0^{\infty} \frac{x}{\sinh(ax)} dx.$$

Повторим вычисление интеграла:

```
>> syms x a;
>> y=x/sinh(a*x);
>> int(y,0,inf)
ans =
```

Решение в явном виде не получено. Причина в том, что переменная a не определена. Это может быть число положительное или

отрицательное, действительное или комплексное. Предположим, что это число действительное и положительное. Тогда программа будет иметь вид:

```
>> syms x a;  
>> y=x/sinh(abs(a)*x);  
>> int(y,0,inf)  
ans =  
      pi^2/(4*a^2)
```

Функция `int()` позволяет вычислять достаточно сложные интегралы. Вот один из примеров.

Пример 10.21

Требуется вычислить следующий несобственный интеграл:

$$\int_0^{\infty} \frac{dx}{(x + \sqrt{x^2 + a^2})^3}.$$

Решение имеет вид:

```
>> syms x a;  
>> y=1/(x+sqrt(x^2+a^2))^3;  
>> int(y,0,inf)  
ans =  
      3/(8*a^2)
```

Не нужно указывать программе на знак числа a и писать `abs(a)`, т. к. число a возводится в квадрат и является положительным.

10.4. Примеры вычисления интегралов

Далее приводятся задачи на интегрирование с помощью системы MATLAB. Эти задачи будут способствовать изучению компьютерных технологий вычисления интегралов. Кроме того, они весьма оригинальны и вызывают восхищение своими ответами. Не следует огорчаться, если система выдает иное решение, чем

указано в ответе. Вероятнее всего, они идентичны. Попробуйте преобразовать полученный ответ с помощью функций `simplify()` и `factor()`.

Проверить идентичность ответов можно, если подставить одно из численных значений переменных в подынтегральное выражение и вычислить значение интеграла, сравнив его с численным значением ответа, приведенного в таблице интегралов (табл. 10.2 и 10.3).

Возможны случаи, когда система вообще не выдает решения, а оно имеется. Это доказывает лишь то, что компьютер не может быть более интеллектуален, чем человек; его программа не совершенна.

Таблица 10.2. Таблица определенных интегралов

№	Интеграл	Ответ
1	$\int_0^1 \frac{x}{\sqrt{1-x}} dx$	$\frac{4}{3}$
2	$\int_0^1 \frac{1}{(1-ax)^n} dx$	$\frac{(1-a)^{1-n} - 1}{a(n-1)}$
3	$\int_0^1 x^{n-1}(x+1) dx$	$\frac{1}{n+1} + \frac{1}{n}$
4	$\int_0^1 \left(\frac{x^2}{x+1} \right) dx$	$\ln 2 - \frac{1}{2}$
5	$\int_0^1 \frac{x^5}{1+x} dx$	$\frac{47}{60} - \ln 2$
6	$\int_0^1 \frac{x^2}{\sqrt{1-x^2}} dx$	$\frac{\pi}{4}$
7	$\int_0^1 x \sin x dx$	$\sin 1 - \cos 1 = 0.301$

Таблица 10.2 (окончание)

№	Интеграл	Ответ
8	$\int_0^1 x \cos x \, dx$	$\cos 1 - \sin 1 - 1$
9	$\int_0^1 x^n \ln x \, dx$	$-\frac{1}{(n+1)^2}$
10	$\int_0^1 \frac{x^3 \ln x}{1+x} \, dx$	-0.0386

Таблица 10.3. Таблица неопределенных интегралов

№	Интеграл	Ответ
1	$\int \frac{x}{1-x^2} \, dx$	$\frac{1}{2} \ln(1+x^2)$
2	$\int \frac{x}{1+x^4} \, dx$	$\frac{1}{2} \operatorname{arctg} x^2$
3	$\int \frac{x}{1+x^2} \, dx$	$\operatorname{arctg} x$
4	$\int \frac{x}{1-x^4} \, dx$	$-\frac{1}{4} \ln \frac{-1+x^2}{1+x^2}$
5	$\int \frac{1}{x(1+x^2)} \, dx$	$\ln \frac{x}{\sqrt{1+x^2}}$
6	$\int \frac{1}{\sqrt{(a+bx)^3}} \, dx$	$-\frac{2}{b\sqrt{a+bx}}$
7	$\int \frac{x}{\sqrt{(a+bx+cx^2)^3}} \, dx$	$-\frac{2(2a+bx)}{(4ac-b^2) \cdot \sqrt{a+bx+cx^2}}$

Таблица 10.3 (продолжение)

№	Интеграл	Ответ
8	$\int \frac{x^2}{\sqrt{(a+bx)^3}} dx$	$\frac{2(b^2x^2 - 4abx - 8a^2)}{3b^2\sqrt{a+bx}}$
9	$\int \frac{x^4}{\sqrt{(a+cx^2)^7}} dx$	$\frac{1}{5} - \frac{x^5}{a(\sqrt{a+cx^2})^5}$
10	$\int \frac{1}{x\sqrt{a+cx^2}} dx$	$\frac{1}{\sqrt{a}} \ln \frac{\sqrt{a+cx^2} - \sqrt{a}}{x}$
11	$\int \frac{1}{\operatorname{sh} x \cdot \operatorname{ch} x} dx$	$\ln(\operatorname{th} x)$
12	$\int \frac{1}{\operatorname{sh}^2 x \cdot \operatorname{ch}^2 x} dx$	$\frac{4}{1-e^{4x}}$
13	$\int \frac{\operatorname{ch} x}{\operatorname{sh} x \cdot \operatorname{ch} x} dx$	$\frac{x}{2} - \frac{1}{4}e^{-2x}$
14	$\int xe^{ax} \operatorname{sh}(ax) dx$	$\frac{e^{2ax}}{4a} \left(x - \frac{1}{2a} \right) - \frac{x^2}{4} + \frac{1}{8a^2}$
15	$\int \sin x \cos^2 x dx$	$-\frac{\cos^3 x}{3}$
16	$\int \sin^4 x \cos x dx$	$\frac{\sin^5 x}{5}$
17	$\int \frac{1}{\sin x} dx$	$\ln \left(\operatorname{tg} \frac{x}{2} \right)$
18	$\int \frac{x + \sin x}{1 + \cos x} dx$	$x \operatorname{tg} \frac{x}{2}$
19	$\int \frac{1}{x^2(x-1)} dx$	$\ln(x-1) - \ln x + \frac{1}{x}$

Таблица 10.3 (окончание)

№	Интеграл	Ответ
20	$\int \frac{x \sin x}{\cos^2 x} dx$	$\frac{x}{\cos x} - \ln \left(\operatorname{tg} \left(\frac{x}{2} + \frac{\pi}{4} \right) \right)$
21	$\int \frac{1}{\sin^3 x \cos x} dx$	$-\frac{1}{2 \sin^2 x} + \ln(\operatorname{tg} x)$
22	$\int x \cos x^2 dx$	$\frac{\sin x^2}{2}$
23	$\int \frac{\sin^2 x}{\cos^4 x} dx$	$\frac{1}{3} \operatorname{tg}^3 x$
24	$\int a^x e^{-x} dx$	$\frac{e^{-x} a^x}{\ln a - 1} - \frac{1}{\ln a - 1}$

Таблица 10.4. Таблица несобственных интегралов

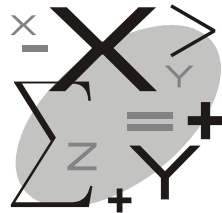
№	Интеграл	Ответ
1	$\int_0^{\infty} \frac{1+x}{(x+a)^{p+1}} dx$	$(x+1)(x+a)^{-p-1}$
2	$\int_0^{\infty} x^a e^{-x} dx$	$a!$
3	$\int_{-1}^{\infty} \frac{e^{-qx}}{\sqrt{1+x}} dx$	$e^q \sqrt{\frac{\pi}{q}}, q > 0$
4	$\int_0^{\infty} \frac{x^2}{\sinh(ax)} dx$	$\frac{4.2}{a^3}$
5	$\int_1^{\infty} \frac{1}{(a-bx)(x-1)} dx$	$\ln \frac{2b-a}{b}$ $a-b$

Таблица 10.4 (продолжение)

№	Интеграл	Ответ
6	$\int_0^{\infty} x^a e^{-ax} dx$	$a^a (a-1)!$
7	$\int_0^{\infty} \frac{x}{\sinh(ax)} dx$	$\frac{\pi^2}{4a^2}$
8	$\int_0^{\infty} x^{2n+1} p e^{-px^2} dx$	$\frac{n!}{2p^n}$
9	$\int_0^{\infty} \frac{1}{1+e^{px}} dx$	$\frac{\ln 2}{p}$
10	$\int_0^{\infty} \frac{ax}{\sinh(ax)} dx$	$\frac{\pi^2}{2a}$
11	$\int_0^{\infty} e^{-q^2 x^2} dx$	$\frac{\sqrt{\pi}}{2q}, q > 0$
12	$\int_0^{\infty} x^n e^{-kx} dx$	$n! k^{-n-1}, k > 0$
13	$\int_0^1 \frac{x e^x}{(1+x)^2} dx$	$\frac{e}{2} - 1$
14	$\int_{-\infty}^{+\infty} \frac{e^x}{(\beta + e^x)^2} dx$	$\frac{1}{\beta}$
15	$\int_0^{\infty} \frac{x}{1-x^3} dx$	$\frac{2\sqrt{3}}{9} \pi$
16	$\int_0^{\infty} \frac{1}{(x^2 + a^2)^3} dx$	$\frac{3\pi}{16a^5}$

Таблица 10.4 (окончание)

№	Интеграл	Ответ
17	$\int_{-\infty}^{\infty} \frac{1}{(ax^2 + 2bx + c)^2} dx$	$\frac{\pi a}{2(ac - b^2)^{3/2}}$
18	$\int_0^{\infty} \frac{1}{(x + \sqrt{x^2 + a^2})^2} dx$	$\frac{2}{3a}$
19	$\int_0^{\infty} \frac{e^{-p}}{1+x^3} dx$	$\frac{2\sqrt{3}}{9} \pi e^{-3}$
20	$\int_0^{\infty} \frac{x}{\sqrt{e^x - 1}} dx$	$2\pi \ln 2$
21	$\int_0^{\infty} \frac{x e^{-x}}{\sqrt{e^x - 1}} dx$	$\frac{\pi}{2} (2 \ln 2 - 1)$
22	$\int_0^{\infty} \frac{x e^{-2x}}{\sqrt{e^x - 1}} dx$	$\frac{3}{4} \pi \left(\ln 2 - \frac{7}{12} \right)$
23	$\int_0^{\infty} \frac{x^2 e^{-x}}{\sqrt{(e^x - 1)^3}} dx$	$8\pi \ln 2$
24	$\int_0^{\infty} x^{2n+1} e^{-px^2} dx$	$\frac{n!}{2p^{n+1}}$



ГЛАВА 11

Методы и компьютерные технологии интерполяции

11.1. Элементы теории

Основы теории интерполяции и компьютерные технологии решения задач достаточно полно изложены в [13]. Здесь приводятся лишь элементы теории.

Интерполяцией называется представление функции $y = f(x)$ функцией $\varphi(x)$, идентичной в некоторой области значений аргумента.

Функция $f(x)$ может быть задана аналитически или в виде таблицы. В зависимости от метода интерполяции функция $\varphi(x)$ в MATLAB может быть получена в аналитическом виде или в виде чисел из диапазона узлов интерполяции.

Интерполяция — научная основа моделирования. Функция интерполяции $\varphi(x)$, полученная в аналитическом виде, есть математическая модель изучаемого объекта или явления.

Основными видами интерполяции являются: точная в узлах и приближенная в узлах.

При интерполяции, точной в узлах, значения функции $y = \varphi(x)$ совпадают со значениями исходной функции $y = f(x)$ в узлах интерполяции (рис. 11.1).

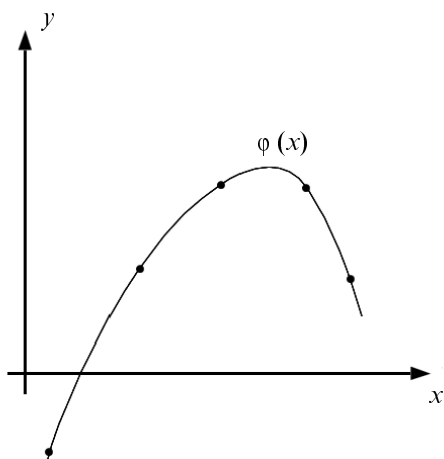


Рис. 11.1. Интерполяция, точная в узлах

На рис. 11.1 значения функции $y = f(x)$ обозначены точками, совпадающими в узлах интерполяции с функцией $y = \varphi(x)$.

При интерполяции, приближенной в узлах, значения функции $y = \varphi(x)$ не совпадают со значениями исходной функции $y = f(x)$ (рис. 11.2).

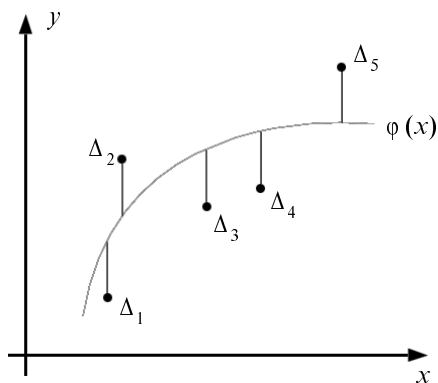


Рис. 11.2. Интерполяция, приближенная в узлах

Интерполяция, точная в узлах, используется в тех случаях, когда исходная функция $y = f(x)$ задана аналитически или моделируется физическое явление при высокой точности исходных данных.

Интерполяцию, приближенную в узлах, часто называют *аппроксимацией*. Она применяется в случаях, когда исходные данные функции $y = f(x)$, полученные экспериментально, могут содержать ошибки. Тогда аппроксимация позволяет получить математическую модель с более высокой точностью, чем интерполяция, точная в узлах. Высокая точность аппроксимации обеспечивается за счет сглаживания неточностей исходных данных.

Компьютерные технологии интерполяции любого вида состоят из следующих основных этапов:

- ◆ выбор вида функции интерполяции;
- ◆ определение коэффициентов функции интерполяции;
- ◆ проверка адекватности математической модели.

Далее описываются эти этапы.

11.1.1. Выбор вида функции интерполяции

Этот этап является наиболее важным при практическом решении задач. Вид функции интерполяции во многом определяет адекватность полученной модели. Ошибка в выборе вида функции $\varphi(x)$ приводит во многих случаях к отрицательному результату моделирования.

Рассмотрим способы выбора вида функции интерполяции.

Графоаналитический способ

Функция $y = f(x)$ представляется в виде графика. График сравнивается с графиками типичных математических функций и на основании их сравнения выбирается подходящая. Перечислим часто используемые функции и их графики.

◆ *Степенная функция*

$$y = a^x.$$

График функции показан на рис. 11.3.

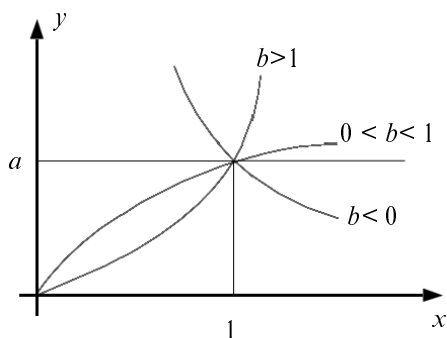


Рис. 11.3. Степенная функция $y = a^x$

◆ *Логарифмическая функция*

$$y = a + b \ln x.$$

График функции показан на рис. 11.4.

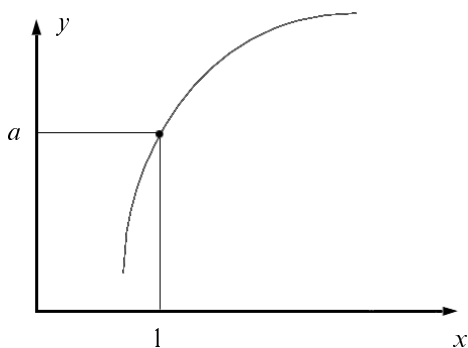


Рис. 11.4. Логарифмическая функция $y = a + b \ln x$

◆ Дробно-линейная функция

$$y = \frac{x}{a + bx}.$$

График функции показан на рис. 11.5.

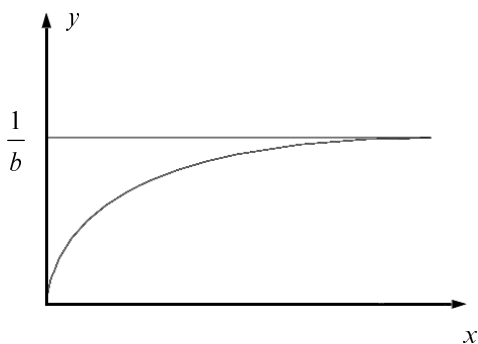


Рис. 11.5. Дробно-линейная функция $y = \frac{x}{a + bx}$

◆ Показательная функция

$$y = ab^x.$$

График функции показан на рис. 11.6.

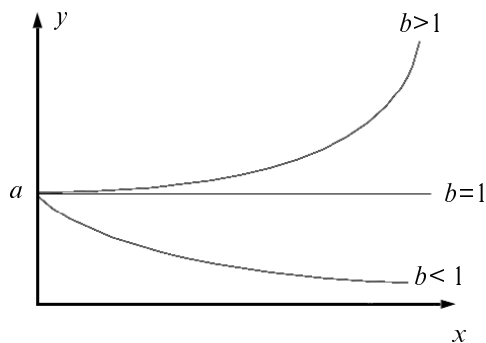


Рис. 11.6. Показательная функция $y = ab^x$

Приведем пример выбора функции интерполяции графоаналитическим методом.

Пример 11.1

Пусть функция задана в виде табл. 11.1. Необходимо установить вид функции интерполяции $y = \varphi(x)$.

Таблица 11.1. Исходная функция $y = f(x)$

x	y	$\varphi(x)$	x	y	$\varphi(x)$
1	25	25	6	1.2	1.19
2	7.7	7.69	7	0.9	0.915
3	3.9	3.86	8	0.73	0.729
4	2.4	2.37	9	0.6	0.59
5	1.6	1.62	10	0.5	0.499

График в виде координат точек таблицы приведен на рис. 11.7.

Из рис. 11.7 видно, что функция подобна степенной $\varphi(x) = ax^b$ при $b < 0$. Решая задачу интерполяции, получим: $\varphi(x) = 2.5x^{1.7}$. Результаты табулирования функции $\varphi(x)$ приведены в табл. 11.1.

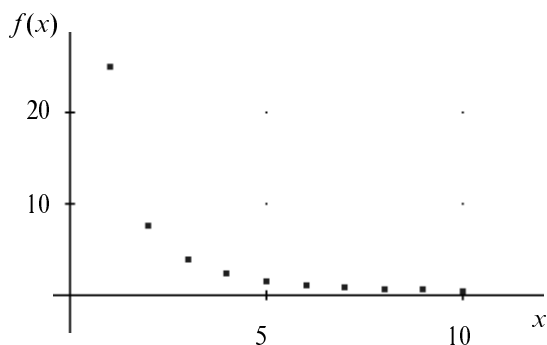


Рис. 11.7. Функция $y = f(x)$, соответствующая данным табл. 11.1

На рис. 11.8 показаны функция интерполяции $\varphi(x)$ и исходная функция $y(x)$, заданная в виде координат точек табл. 11.1.

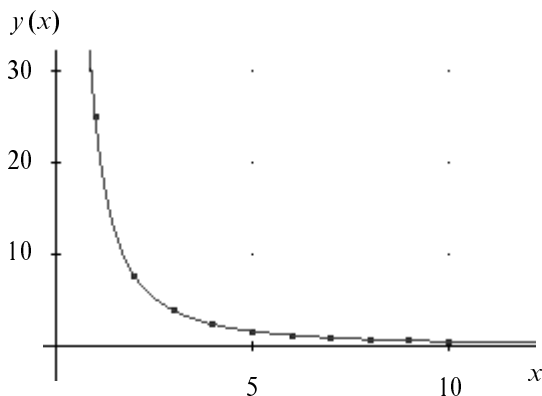


Рис. 11.8. Исходная функция (точки) и функция интерполяции (сплошная кривая)

Из таблицы и рисунка видим, что полученная функция интерполяции может быть математической моделью объекта: сплошная кривая проходит через точки, являющиеся координатами функции $f(x)$.

Способ линеаризации нелинейных функций

Линеаризация нелинейных функций осуществляется путем представления их в иной системе координат методом замены переменных x и y . Покажем способы линеаризации на примерах рассмотренных выше функций.

♦ Показательная функция

$$y = ab^x.$$

Прологарифмируем показательную функцию:

$$\ln y = \ln a + x \ln b.$$

Заменой переменных $\ln y = Y$, $x = X$ получим линейную функцию

$$Y = \ln a + X \ln b.$$

◆ *Степенная функция*

$$y = ax^b.$$

Представим функцию в виде:

$$\ln y = \ln a + b \ln x.$$

Тогда введением новых переменных $\ln y = Y$, $\ln x = X$ получим линейную функцию

$$Y = \ln a + bX.$$

◆ *Логарифмическая функция*

$$y = a + b \ln x.$$

Выравнивание осуществляется заменой переменных $y = Y$, $\ln x = X$. После замены переменных получим:

$$Y = a + bX.$$

◆ *Дробно-линейная функция*

$$y = \frac{x}{a + bx}.$$

Представим функцию в виде:

$$\frac{x}{y} = a + bx.$$

Введя новые переменные $Y = \frac{x}{y}$, $X = x$, получим линейную функцию

$$Y = a + bX.$$

Анализ табличных разностей

Этот способ позволяет выбрать степень многочлена при полиномиальной интерполяции. Если n -е табличные разности функции $y = f(x)$ одинаковы, то степень многочлена будет не выше n .

Рассмотрим этот способ на примере.

Пример 11.2

Предположим, что функция интерполяции является многочленом и представлена в виде табл. 11.2.

Таблица 11.2. Таблица функции $f(x)$

x	1	2	3	4	5	6	7
y	2.2	6.5	12.8	21.1	31.4	43.7	58

В табл. 11.3 приведены значения табличных разностей.

Таблица 11.3. Значения табличных разностей

x	1	2	3	4	5	6	7
y	2.2	6.5	12.8	21.1	31.4	43.7	58
$\Delta^{(1)}$		4.3	6.3	8.3	10.3	12.3	14.3
$\Delta^{(2)}$		2	2	2	2	2	

Из таблицы видно, что вторые табличные разности постоянны. Это значит, что интерполяционный полином должен быть не выше второй степени, т. е.

$$y = a_0 + a_1x + a_2x^2.$$

В результате решения задачи интерполяции получим:

$$y = x^2 + 1.3x - 0.1.$$

Использование специальных программ автоматизации интерполяции

Существуют универсальные программные средства, позволяющие выбрать функцию интерполяции, если исходная функция представлена в табличной форме. Такими программными средствами являются: SIMPLE FORMULA, TableCurve, Curve Expert.

Эти программы выдают более тысячи различных функций с указанием их погрешностей. Их описание и примеры использования приведены в [13].

11.1.2. Определение коэффициентов функции интерполяции

Существует большое число различных способов определения коэффициентов функции интерполяции. Выбор способа зависит от формулировки задачи и метода интерполяции. Система MATLAB позволяет определить коэффициенты функции интерполяции следующими способами:

- ◆ использованием встроенных функций системы;
- ◆ решением алгебраических линейных или нелинейных уравнений;
- ◆ непосредственным вычислением коэффициентов полиномов.

11.1.3. Проверка адекватности модели

Проверку правильности решения задачи можно осуществить следующими способами:

- ◆ табулированием функции аппроксимации и сравнением ее результатов с исходными данными;
- ◆ использованием графических образов;
- ◆ вычислением погрешности математической модели.

Первые два способа будут подробно рассмотрены при решении задач интерполяции различными методами.

Наиболее эффективным способом проверки адекватности модели является вычисление погрешностей функции интерполяции.

Погрешность интерполяционной формулы оценивается абсолютной среднеквадратической погрешностью ε и максимальной относительной δ , которые имеют вид:

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^n \Delta_i^2}{n}}, \quad \delta = \frac{\varepsilon}{y_{\min}} \cdot 100\%, \quad (11.1)$$

где:

- ◆ $\Delta_i = y(x_i) - \varphi(x_i)$;
- ◆ $y(x_i)$ — функция, заданная в виде таблицы;
- ◆ $\varphi(x_i)$ — функция интерполяции;
- ◆ n — число узлов исходной функции, заданной таблично;
- ◆ y_{\min} — минимальное значение функции.

Задача считается решенной, если погрешность модели не превосходит заданной.

Вычисления погрешностей ε и δ в системе осуществляются с помощью встроенных функций. Процедуры расчета погрешностей рассмотрим в дальнейшем на примерах.

11.2. Интерполяция точная в узлах. Универсальный метод

11.2.1. Интерполяция линейными функциями

Универсальный метод требует решения систем алгебраических уравнений. Система позволяет решать уравнения матричным методом и с помощью функции `solve()`. Эти методы описаны в главе 8.

Процедуры решения задачи интерполяции универсальным методом в системе покажем на примере.

Пример 11.3

Пусть функция $y = f(x)$ задана в виде табл. 11.4, представляющей собой зависимость высоты березы H от возраста L [13]. Необходимо найти математическую модель роста березы $H = f(L)$.

Решим задачу методом интерполяции, точной в узлах, при ограниченном их числе.

Таблица 11.4. Зависимость высоты березы от возраста

	Зависимость высоты березы от возраста								
L , лет	20	30	40	50	60	70	80	90	100
H , м	9.8	12.8	15.8	18.6	21.3	23.5	25.3	26.5	27.4

Выберем вид функции интерполяции, воспользовавшись графо-аналитическим методом.

Значения функция $H = f(L)$ в виде точек графика приведены на рис. 11.9.

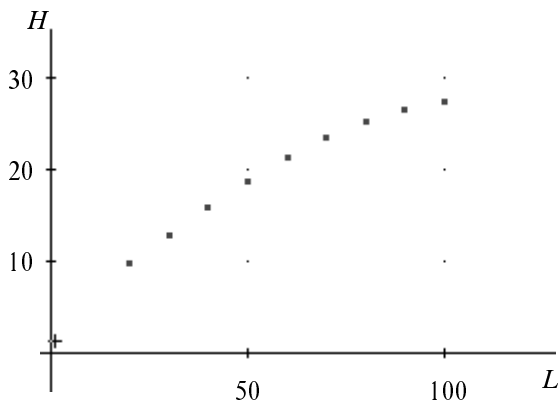


Рис. 11.9. Координаты точек функции, заданной табл. 11.4

Из графика видно, что функцию $H = f(L)$ можно представить полиномом степени, не выше второй. Предположим, что интерполяционный многочлен является многочленом второй степени:

$$L(H) = a_0 + a_1 H + a_2 H^2.$$

Для определения коэффициентов многочлена составим систему уравнений. Выберем из таблицы координаты (20; 9.8), (50; 18.6), (90; 26.5).

Тогда система уравнений будет иметь вид:

$$\begin{cases} 9.8 = a + 20b + 20^2 c \\ 18.6 = a + 50b + 50^2 c \\ 26.5 = a + 90b + 90^2 c \end{cases}$$

Решение системы уравнений в среде MATLAB.

Создадим группу символьных объектов:

```
>> syms b1 b2 b3 a b c t;
```

Опишем уравнения системы на языке MATLAB:

```
>> y1='9.8=a+20b+ 20^2c';
```

```
>> y2='18.6=a+50b+50^2c';
```

```
>> y3='26.5=a+90b+90^2c';
```

Команда решения системы:

```
>> [b1,b2,b3]=solve(y1,y2,y3)
```

После нажатия клавиши <Enter> получим ответ в следующем виде:

```
b1=
    2.56
b2=
    0.389
b3=
   -0.00137
```

Проверка решения:

```
>> subs(y1,{a,b,c},{b1,b2,b3});
```

```
>> subs(y2,{a,b,c},{b1,b2,b3});
```

```
>> subs(y3,{a,b,c},{b1,b2,b3})
```

После нажатия клавиши <Enter> в каждой строке оператора subs получим ответы:

```
9.8 = 9.8    18.6 = 18.6    26.5 = 26.5
```

Решение системы уравнений верно.

Таким образом, функция интерполяции имеет вид:

$$H(L) = -0.00137L^2 + 0.389L + 2.56.$$

Она показана на рис. 11.10.

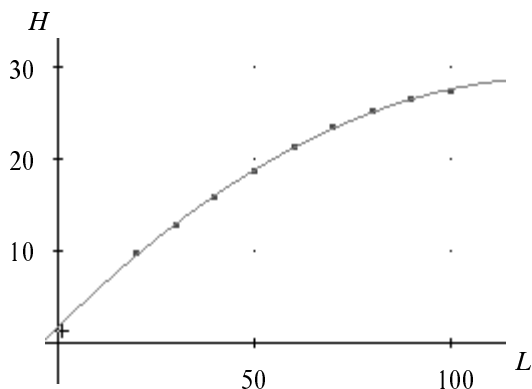


Рис. 11.10. Функция интерполяции

Из рис. 11.10 видно, что функция интерполяции и координаты функции, заданной в виде таблицы, практически совпадают.

Вычислим значения функции в узлах интерполяции, решив задачу табулирования функции $y = f(x)$:

```
>> p1=subs(2.56+0.389*t-...
0.00137*t^2,t,[20,30,40,50,60,70,80,90,100])
```

После нажатия клавиши <Enter> получим ответ:

```
p1 =
    9.79    12.99    15.92    18.585    20.968    23.077    24.91
    26.47    27.76
```

Исходные данные функции $y(x)$ представим в виде вектора:

```
>> P=[9.8, 12.8, 15.8, 18.6, 21.3, 23.5, 25.3, 26.5, 27.4];
```

Вычислим значения погрешностей интерполяции.

```
>> e=sum((p-p1).^2);
>> E=sqrt(e)/length(P)
E =
    0.0879
>> ymin=min(P);
>> d=E/ymin*100
d =
    0.8967
```

Малые абсолютная и относительная погрешности (примерно 0,9%) означают, что полученная в результате интерполяции функция может быть математической моделью роста березы.

11.2.2. Интерполяция нелинейными функциями

Интерполяция точная в узлах нелинейными функциями осуществляется в системе с помощью функции `fsolve()`, которая имеет вид:

```
fsolve('file', x0)
```

где:

- ◆ x_0 — вектор начальных приближений;
- ◆ 'file' — имя m-файла, содержащего систему уравнений.

Рассмотрим нелинейную интерполяцию на примере.

Пример 11.4

Пример заимствован из [13]. Пусть функция $f(x)$ задана в виде табл. 11.5.

Таблица 11.5. Таблица функции $y = f(x)$

x	1	2	3	4	5	6
y	6,5	20	53,5	167	473	1470

Необходимо функцию представить в виде формулы.

Выберем вид функции интерполяции, воспользовавшись графо-аналитическим методом. График функции в виде точек таблицы показан на рис. 11.11.

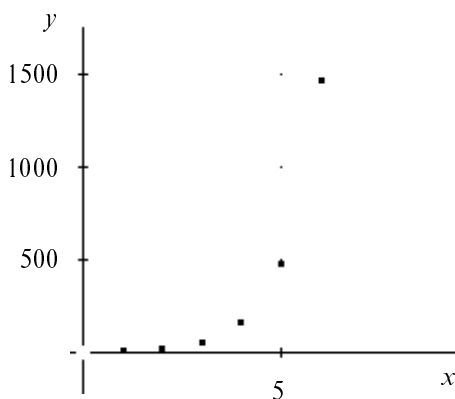


Рис. 11.11. Вид функции, представленной в виде табл. 11.5

Из рис. 11.11 видно, что функция $y = f(x)$ похожа на степенную. Выберем в качестве интерполяционной функцию вида

$$y = ab^x + c.$$

Так как функция содержит три неизвестных, то составим систему трех нелинейных уравнений, выбрав в качестве аргументов узлы интерполяции $x = 1, 4, 6$. Тогда система уравнений будет иметь вид:

$$\begin{cases} ab + c = 6.5 \\ ab^4 + c = 167 \\ ab^6 + c = 1470 \end{cases}$$

Представим эту систему уравнений в виде функции пользователя с именем файла `myfun.m` и сохраним его с помощью команды **Save** (Сохранить) меню **File** (Файл) или соответствующей кнопки панели инструментов.

Содержимое файла может иметь вид:

```
Function F=myfun(x)
F = [x(1)*x(2)+x(3)-6.5; x(1)*x(2)^4+x(3)-167;
x(1)*x(2)^6+x(3)-1470];
```

Теперь воспользуемся функцией `fsolve()`:

```
>> x0=[1; 1; 1];
>> x=fsolve('myfun', x0)
```

После нажатия клавиши <Enter> получим решение:

```
x =
    2.1512
    2.9678
    0.1157
```

Тогда функция интерполяции будет иметь вид:

$$\varphi(x) = 2.1512 \cdot 2.9678^x + 0.1157.$$

11.2.3. Сплайн-интерполяция

Интерполяция кубическими сплайнами в среде MATLAB осуществляется с помощью функции `spline()`. Функция имеет вид:

```
Yi = spline(x, y, xi)
```

где:

- ◆ x — вектор узлов интерполяции;
- ◆ y — вектор значений функции в узлах интерполяции;
- ◆ x_i — вектор аргументов функции $y = f(x)$ из области ее определения, задаваемый пользователем.

Вместо вектора функция $y = f(x)$ может быть задана в виде формулы.

Функция `spline()` не позволяет получить функцию интерполяции в виде формулы. В этом ее существенный недостаток.

Рассмотрим технологию интерполяции на примерах.

Пример 11.5

Пусть функция задана в виде табл. 11.5. Необходимо найти ее значения при $x = 1.5, 2.5, 3.5, 4.5, 5.5$.

Процедуры интерполяции будут иметь вид:

```
>> x=[1,2,3,4,5,6];  
>> y=[6.5,20,53.5,167,473,1470];  
>> xi=[1.5,2.5,3.5,4.5,5.5];  
>> yi=spline(x, y, xi)
```

После нажатия клавиши <Enter> на экране появится ответ:

```
yi =  
    15.233    29.7667    98.7    270.9958    847.7542
```

Пример 11.6

Пусть функцией является

$$y = \sin x,$$

заданная при $x = 1, 2, 3, 4, 5$. Необходимо найти ее значения при $x = 1.5, 1.8, 2.3, 3, 3.5$.

Программа решения задачи имеет вид:

```
x=[1,2,3,4,5];  
xi=[1.5, 1.8, 2.3, 3, 3.5];  
yi=spline(x, sin(x),xi)
```

После нажатия клавиши <Enter> на экране ответ:

```
1.022    0.9843    0.7358    0.1411   -0.3414.
```

Из ответа видны погрешности интерполяции: значение $\sin x$ не может быть больше единицы.

11.2.4. Интерполяция точная в узлах

Функция `interp1()` позволяет решать задачи интерполяции несколькими методами. Она имеет вид:

```
yi=interp1(x, y, xi, метод)
```

где:

- ◆ x, y — векторы значений узлов и функции;
- ◆ x_i — вектор значений аргументов, задаваемый пользователем;
- ◆ *метод* — аргумент, позволяющий пользователю выбрать метод интерполяции.

Методами интерполяции являются:

- ◆ 'nearest' — ступенчатая;
- ◆ 'linear' — линейная;
- ◆ 'cubic' — кубическая;
- ◆ 'spline' — кубическими сплайнами.

Если метод не указан, то реализуется линейная интерполяция.

Приведем пример решения задачи этими методами.

Пример 11.7

Функция $y = f(x)$ задана в виде табл. 11.6.

Таблица 11.6. Значения функции $y = f(x)$

x	2.5	3.7	8.4	11.7	20	27	38
y	1.4	2.7	5.6	7.5	9.1	13.2	15.3

Необходимо определить значения функции при значениях аргументов $x = 3, 4, 6, 10, 25, 30$.

Далее приводятся процедуры решения задачи на ЭВМ:

```
>> x=[2.5, 3.7, 8.4, 11.7, 20, 27, 38];
>> y=[1.4, 2.7, 5.6, 7.5, 9.1, 13.2, 15.3];
>> xi=[3,4, 6, 10, 25, 30];
>> yi=interp1(x, y, xi);           % линейная интерполяция
>> yi=interp1(x, y, xi, 'linear'); % линейная интерполяция
>> yi=interp1(x, y, xi, 'nearest'); % ступенчатая интерполяция
>> yi=interp1(x, y, xi, 'cubic');  % кубическая интерполяция
```

```
>> yi=interp1(x, y, xi, 'spline'); % интерполяция кубическими  
>>                                     % сплайнами
```

В результате решения задачи получим следующие ответы:

◆ линейная интерполяция

1.9417	2.8851	4.1191	6.5212	12.0286	13.7727
--------	--------	--------	--------	---------	---------

◆ ступенчатая интерполяция

1.4	2.7	2.7	5.6	13.2	13.2
-----	-----	-----	-----	------	------

◆ кубическая

1.9884	2.9419	4.2606	6.3449	12.2020	14.0582
--------	--------	--------	--------	---------	---------

◆ кубическими сплайнами

1.9912	2.9666	4.3323	6.5662	11.8137	15.1007
--------	--------	--------	--------	---------	---------

Из результатов решения видно, что ответы разные и зависят от метода интерполяции. Какой же из них более точный? Ответ на этот вопрос можно дать только при анализе погрешностей интерполяции.

Функция `interp1()` не позволяет получить функцию интерполяции $\varphi(x)$ в виде формулы. В этом ее существенный недостаток.

11.3. Интерполяция, приближенная в узлах (аппроксимация)

11.3.1. Функция *lsqcurvefit()*

Осуществим сглаживание неточностей исходных данных, используя функции аппроксимации. Одной из таких функций в MATLAB является функция `lsqcurvefit()`, которая имеет вид:

```
coef=lsqcurvefit(f, x0, a, b)
```

где:

- ◆ a, b — векторы узлов интерполяции и значений функции;
- ◆ x_0 — стартовое значение параметров функции;
- ◆ f — функция аппроксимации, задаваемая пользователем.

Технологию решения задачи интерполяции рассмотрим на примере.

Пример 11.8

Стоимость G перевозки леса автопоездом в зависимости от расстояния S представлена в табл. 11.7.

Таблица 11.7. Стоимость перевозки леса

S , км	30	40	50	60	70	80	90	100	110	120
G , тыс. руб.	6.36	6.85	7.34	7.84	8.08	8.32	8.57	8.70	8.82	8.94

Требуется найти математическую модель стоимости перевозки леса методом интерполяции, приближенной в узлах.

Функция $G = f(S)$, полученная по данным табл. 11.7, приведена на рис. 11.12.

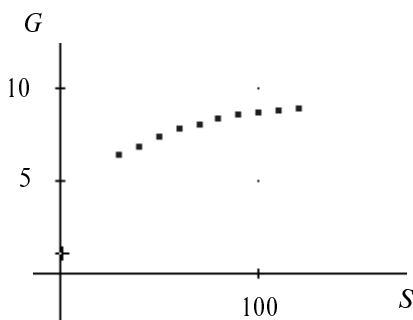


Рис. 11.12. Зависимость стоимости перевозки леса от расстояния

Из рис. 11.12 следует, что в качестве функции интерполяции можно выбрать полином второй степени:

$$G = a + bS + cS^2.$$

Решим задачу аппроксимации полиномом второй степени, используя функцию `lsqcurvefit()`.

Ниже приводятся процедуры решения задачи аппроксимации и вычисления погрешностей в среде MATLAB с применением функции `lsqcurvefit()`:

```
>> a=[30,40,50,60,70,80,90,100,110,120];  
>> b=[6.36,6.85,7.34,7.84,8.08,8.32,8.57,8.70,8.82,8.94];  
>> x0=[1,1,1];  
>> f = inline('x(3)*a.^2+x(2)*a+x(1)', 'x', 'a');  
>> coef=lsqcurvefit(f,x0,a,b)
```

После нажатия клавиши <Enter> на экране появится ответ в виде значений полинома:

```
coef =  
    4.4747    0.0719   -0.0003
```

Функция интерполяции найдена и имеет вид:

$$G = 4.4747 + 0.0719S - 0.0003S^2.$$

Определим абсолютную E и максимальную относительную d погрешности интерполяции, воспользовавшись формулами (11.1):

```
>> x=[30,40,50,60,70,80,90,100,110,120];  
>> y=subs('4.4747+0.0719*t+0.0003*t^2','t',x);  
>> f1=y;  
>> f2=b;  
>> e=f1-f2;  
>> s=sum(e.^2);  
>> E=sqrt(s)/length(f2);  
>> d=E/min(f2)*100
```

После нажатия клавиши <Enter> получаем ответ:

```
E =  
    0.0231    d=0.3633%
```

Низкая погрешность интерполяции является доказательством адекватности математической модели.

Функция `lsqcurvefit()` достаточно универсальна. Вид функции интерполяции может быть не только полином n -й степени, но и другими функциями, приведенными в настоящей главе.

11.3.2. Полиномиальная аппроксимация

Аппроксимация полиномами в среде MATLAB осуществляется с помощью функции `polyfit()`, которая имеет вид:

```
polyfit(x, y, n)
```

где:

- ◆ `x` — вектор узлов интерполяции;
- ◆ `y` — вектор значений функции в узлах интерполяции;
- ◆ `n` — степень полинома.

Откликом при реализации функции `polyfit()` является вектор коэффициентов полинома.

Функция $y = f(x)$ может быть также представлена в аналитическом виде.

Приведем пример интерполяции с помощью функции `polyfit()`.

Пример 11.9

Требуется решить задачу о стоимости вывозки леса из предыдущего примера, используя функцию `polyfit()`.

Функция задана в виде табл. 11.7. Тогда процедуры интерполяции будут иметь вид:

```
>> x=[30,40,50,60,70,80,90,100,110,120];  
>> y=[6.36,6.85,7.34,7.84,8.08,8.32,8.57,8.7,8.82,8.94];  
>> p=polyfit(x, y, 2)
```

После нажатия клавиши <Enter> ответ получим в следующем виде:

```
p =  
    -0.003    0.0719    4.4747
```

Тогда функцией интерполяции будет следующий полином второй степени:

$$\varphi(G) = 4.4747 + 0.0719S - 0.003S^2.$$

Решение совпадает с полученным в предыдущем примере.

Протабулируем функцию $\varphi(G)$ с целью сравнения ее значений с исходными данными.

В MATLAB имеется функция вычисления математического выражения при заданных значениях аргументов. Функция имеет вид:

```
polyval(p, x)
```

где:

- ◆ p — вычисляемая функция;
- ◆ x — вектор аргументов функции.

Воспользуемся этой функцией для проверки достоверности результатов аппроксимации.

Введем функцию $f = \text{polyval}(p, x)$ и нажмем клавишу <Enter>. Откликом будет следующее решение:

$f =$

6,3695	6.884	7.34	7.7373	8.0761	8.3564
8.5780	8.7412	8.8457	8.8917		

Сравнивая решение с вектором y исходных данных, убеждаемся в том, что во всем диапазоне аргументов значения интерполяционного полинома $G(S)$ второй степени мало отличаются от значений исходных данных. Это еще раз доказывает, что полученная функция интерполяции может быть математической моделью оценки стоимости вывозки леса.

Система MATLAB позволяет обоснованно выбрать степень полинома при полиномиальной интерполяции путем вычисления табличных разностей. Для этой цели служит функция $\text{diff}()$, имеющая вид:

```
diff(v, n)
```

где:

- ◆ v — вектор функции $y(x)$;
- ◆ n — порядок конечных разностей.

Пример 11.10

Необходимо определить методом табличных разностей вид функции интерполяции в случае решения предыдущей задачи о стоимости вывозки леса.

Определим степень полинома, воспользовавшись функцией `diff()`. Процедуры в системе MATLAB имеют вид:

```
>> y=[6.36,6.85,7.34,7.84,8.08,8.32,8.57,8.7,8.82,8.94];
>> diff(y,1)
ans =
    0.49    0.49    0.5    0.24    0.24    0.25    0.13    0.12    0.12
>> diff(y, 2)
ans =
    0    0.01   -0.26    0    0.01   -0.12   -0.01    0
```

Табличные разности второго порядка малы и практически одинаковы, поэтому интерполяционный полином должен быть не выше второй степени. Этот же результат нами был получен в примере 11.8.

11.3.3. Интерполяция кубическими полиномами

Интерполяция кубическими полиномами реализуется с помощью функции `icubic()`, имеющей вид:

```
y_i=icubic(x, y, x_i)
```

где:

- ◆ x, y — аргумент и функция, заданные в виде векторов, или x — в виде вектора, а y — в виде формулы;
- ◆ x_i — вектор значений аргумента, для которых вычисляется значение функции; x_i должно быть в диапазоне значений x .

Пример 11.11

Функция задана в виде табл. 11.11

Необходимо найти значение функции при значениях аргумента $x_i = 2, 5, 7, 13, 17$.

Таблица 11.8. Значения функции $y = f(x)$

x	1	3	6	9	12	15	18
y	1	2	5	8	10	14	19

Процедуры решения задачи в среде MATLAB имеют вид:

```
>> x=[1 3 6 9 12 15 18];  
>> y=[1 2 5 8 10 14 19];  
>> xi=[2 5 7 13 17];  
>> yi=icubic(x, y, xi)
```

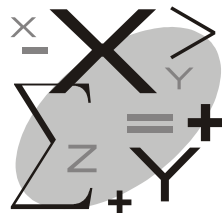
После нажатия клавиши <Enter> получим решение в следующем виде:

```
yi =  
    1.1246    3.0928    5.3590   10.7824   17.1211
```

Функция `icubic()` отличается от функции `interp1()` с опциями 'cubic' и 'spline' методами интерполяции. В этом можно убедиться по результатам решения одних и тех же задач.

При решении предыдущей задачи с помощью функции `interp1()` с опцией 'cubic' получим следующий ответ:

```
yi =  
    1.3524    3.8953    6.1111   11.0778   17.3666
```



ГЛАВА 12

Компьютерные технологии решения задач управления

12.1. Задачи управления

При анализе систем управления задача формулируется следующим образом.

Дано:

- ♦ структурная схема (блок схема) системы;
- ♦ передаточные функции звеньев системы;
- ♦ значения переменных передаточных функций.

Необходимо определить:

- ♦ устойчивость системы управления;
- ♦ качество переходных процессов;
- ♦ точность системы.

При синтезе системы управления задача формулируется иначе. Необходимо создать из имеющихся звеньев структурную схему системы, которая бы удовлетворяла условиям устойчивости (запас по фазе и амплитуде), качеству переходных процессов (форма переходного процесса, длительность, величина перерегулирования) и точности.

В такой постановке решение задач анализа и синтеза весьма целесообразно с помощью математической системы MATLAB.

Характерными особенностями исследований с помощью MATLAB являются:

- ◆ простота;
- ◆ высокая наглядность;
- ◆ возможность получения характеристик системы практически любой сложности.

MATLAB позволяет:

- ◆ исследовать устойчивость системы управления (запасы устойчивости по амплитуде и фазе);
- ◆ получать переходные и частотные характеристики системы;
- ◆ исследовать качество переходных процессов (вид переходного процесса и его длительность, величину перерегулирования);
- ◆ выбрать параметры звеньев системы, вид и характеристики обратной связи с целью обеспечения требуемых динамических свойств системы управления.

Простота решения задач управления определяется наличием в MATLAB специальных функций. Ниже приводятся функции MATLAB, позволяющие воспроизводить передаточные функции звеньев и системы в целом и исследовать ее динамические свойства.

12.2. Функции MATLAB для создания передаточных функций звеньев системы

12.2.1. Функция *tf()*

Функция имеет вид:

`tf(n, m)`

где:

- ◆ *n* — вектор коэффициентов числителя передаточной функции;
- ◆ *m* — вектор коэффициентов знаменателя передаточной функции.

Она служит для образования передаточной функции звеньев и системы в целом.

Пример 12.1

Необходимо образовать передаточную функцию

$$G(S) = \frac{2S + 5}{S^3 + 2S + 1}.$$

В нашем случае векторы коэффициентов числителя и знаменателя передаточной функции имеют вид: $n=[2, 5]$, $m=[1, 0, 2, 1]$.

Ноль в векторе m ставится потому, что в знаменателе передаточной функции член S^2 отсутствует.

Процедуры образования передаточной функции $G(S)$ имеют вид:

```
>> n=[2,5];
>> m=[1,0,2,1];
>> qs=tf(n,m)
```

После нажатия клавиши <Enter> на экране появится передаточная функция в виде:

```
Transfer function:
   2 s + 5
-----
s^3 + 2 s + 1
```

Функцию $qs=tf(n,m)$ можно также представить в следующем виде:

```
>> qs=tf([2 5],[1 0 2 1])
```

Числа в векторах n и m отделяются друг от друга либо запятыми, либо пробелами (как в нашем примере), а сами векторы заканчиваются символом (;). Символ точка с запятой подавляет вывод на экран векторов при нажатии клавиши <Enter>.

Процедуры получения передаточной функции приведены на рис. 12.1.

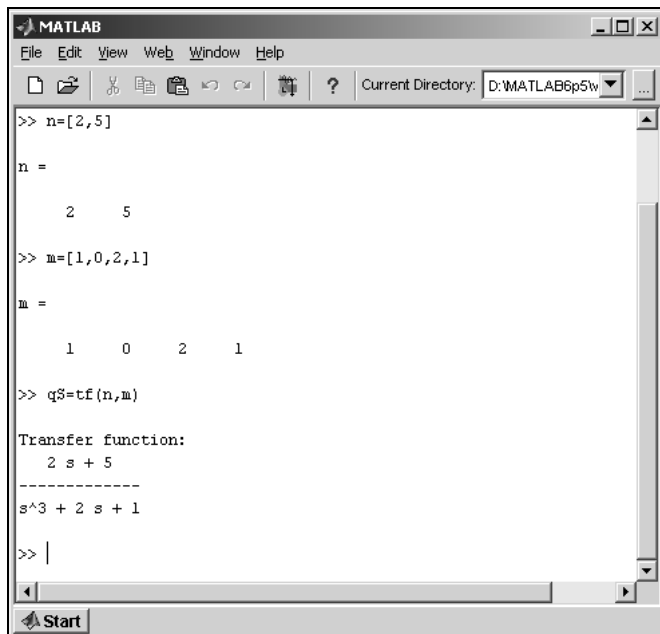


Рис. 12.1. Образование передаточной функции

12.2.2. Функции *pole()* и *zero()*

Функции предназначены для определения, соответственно, полюсов и нулей передаточной функции $G(S)$. Они имеют вид:

`pole(qS)`

`zero(qS)`

где `qS` — имя передаточной функции, заданной оператором `tf`.

Напомним, что *нулями* передаточной функции называются корни числителя, а полюсами — корни знаменателя.

Пример 12.2

Определить полюсы и нули передаточной функции, полученной в примере 12.1.

Процедуры в MATLAB будут иметь вид:

```
>> qs=tf([2 5],[1 0 2 1]);  
>> P=pole(qs)
```

После нажатия клавиши <Enter> на экране появится ответ:

```
P =  
    0.2267 + 1.4677i  
    0.2267 - 1.4677i  
   -0.4534  
>> z=zero(qs)  
Z =  
   -2.5
```

Процедуры определения нулей и полюсов показаны на рис. 12.2.

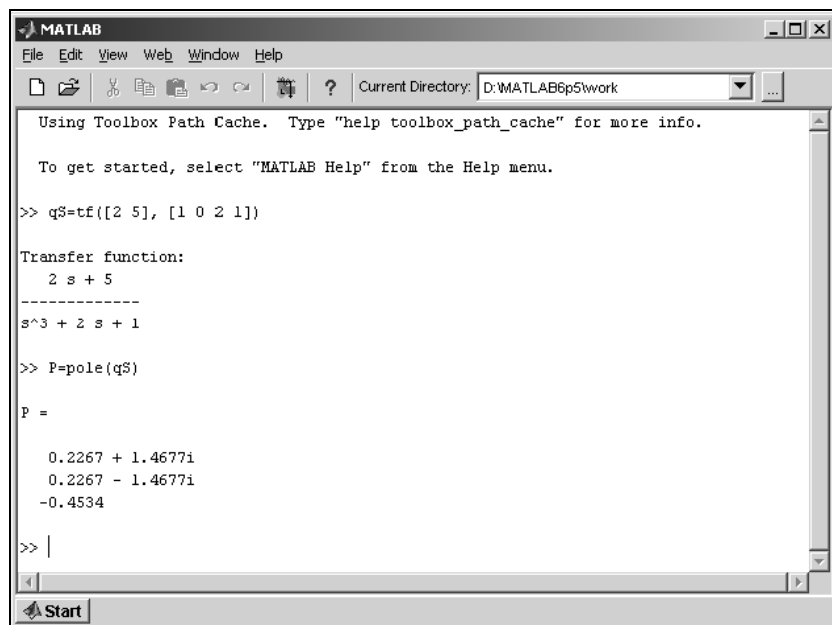


Рис. 12.2. Определение нулей и полюсов передаточной функции

12.2.3. Функции *roots()* и *poly()*

Функции предназначены, соответственно, для вычисления корней полинома и его восстановления по значениям корней. Эти функции имеют вид:

```
roots(P)
```

```
poly(r)
```

где:

◆ P — вектор коэффициентов полинома;

◆ r — вектор корней полинома.

Пример 12.3

Найти корни уравнения $S^3 + 3S^2 + 4$ и по корням восстановить полином.

В данном случае $P=[1 \ 3 \ 0 \ 4]$ и процедуры решения будут иметь вид:

```
>> P = [1 3 0 4];  
>> r = roots(P)  
r =  
    -3.3553  
    0.1777 + 1.0773i  
    0.1777 - 1.0773i  
>> P = poly(r)  
P =  
    1.0000    3.0000    0.0000    4.0000
```

Решение показано на рис. 12.3.

Функции `roots()` и `poly()` полезно использовать для определения полюсов и нулей в условиях, когда по каким-либо причинам функции `pole()` и `zero()` не могут дать решения.

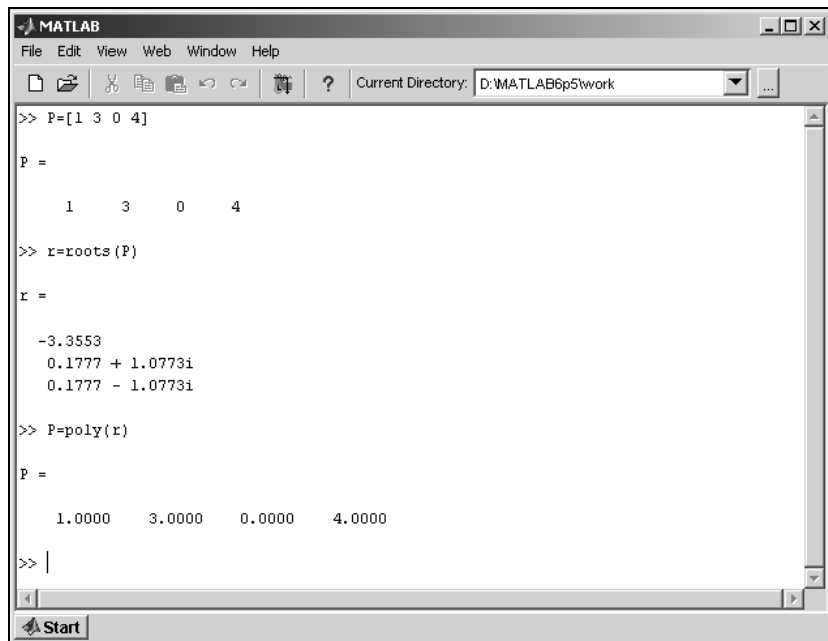


Рис. 12.3. Функции определения корней полинома и его восстановления

12.2.4. Функция *conv()*

Функция применяется для умножения полиномов. Она имеет вид:

`conv(P, q)`

где P , q — векторы коэффициентов полиномов $P(S)$ и $q(S)$.

Пример 12.4

Умножить полиномы

$$P(S) = 3S^2 + 2S + 1 \text{ и } q(S) = S + 4.$$

Процедуры в MATLAB имеют вид:

```
>> P=[3 2 1];
```

```
>> q=[1 4];
```

```
>> G=conv(P, q)
G =
      3      14      9      4
```

Или

$$G = 3S^3 + 14S^2 + 9S + 4.$$

Процедуры умножения полиномов показаны на рис. 12.4

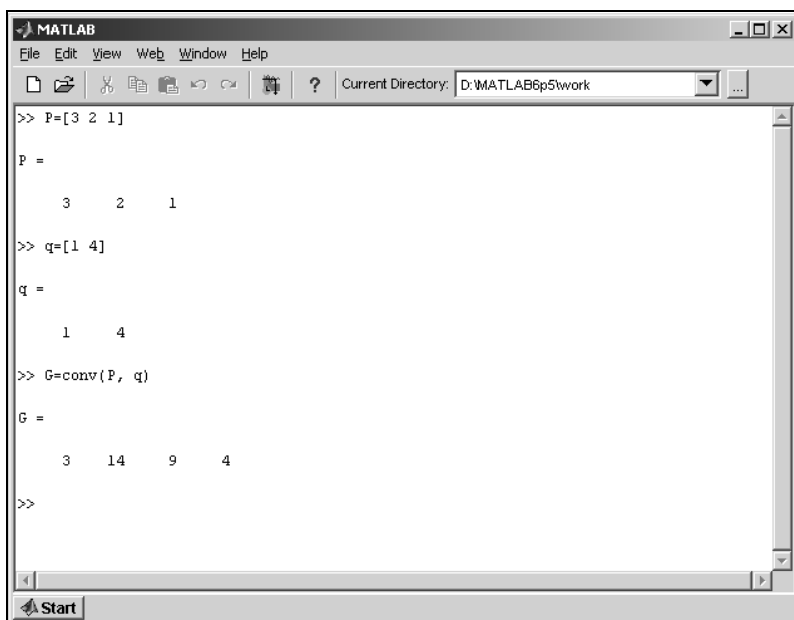


Рис. 12.4. Процедуры умножения полиномов

12.2.5. Функция *polyval()*

Функция предназначена для вычисления значений полинома при заданном значении переменной. Она имеет вид:

`polyval(n, k)`

где:

- ♦ n — вектор коэффициентов полинома;
- ♦ k — значение переменной S .

Пример 12.5

Необходимо вычислить значение полинома

$$P(S) = 3S^2 + 2S + 1$$

при $S = -2$.

Решение:

```
>> n=[3 2 1];  
>> z=polyval(n, -2)  
Z =  
    9
```

Вычисление значения полинома показано на рис. 12.5.

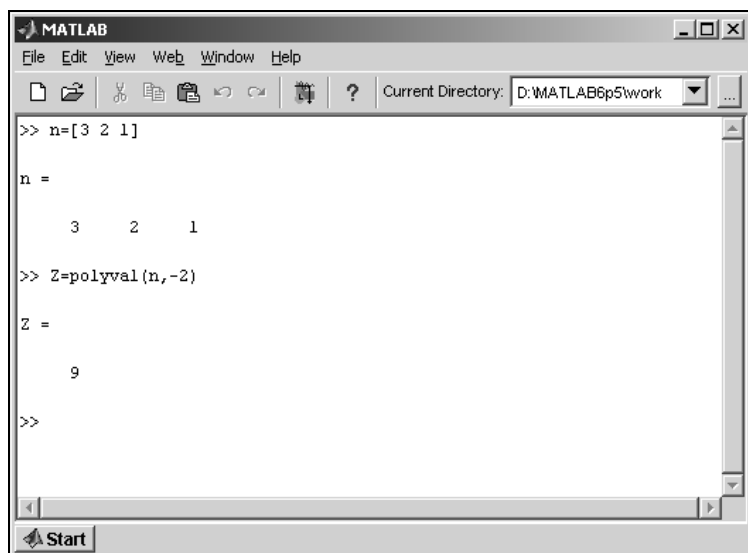


Рис. 12.5. Вычисление значений полинома

12.3. Операции с передаточными функциями звеньев

12.3.1. Сложение передаточных функций

Сложение передаточных функций осуществляется с помощью оператора +.

Пример 12.6

Сложить передаточные функции

$$G_1(S) = \frac{10}{S^2 + 2S + 5} \text{ и } G_2(S) = \frac{2S^2 + 12S + 15}{S^3 + 3S^2 + 7S + 5}.$$

Решение:

```
>> n1=[10];
>> m1=[1 2 5];
>> z1=tf(n1,m1)
Transfer function:
      10
-----
s^2 + 2 s + 5
>> n2=[2 12 15];
>> m2=[1 3 7 5];
>> z2=tf(n2,m2)
Transfer function:
      2 s^2 + 12 s + 15
-----
s^3 + 3 s^2 + 7 s + 5
>> G=z1+z2
Transfer function:
      2 s^4 + 26 s^3 + 79 s^2 + 160 s + 125
-----
s^5 + 5 s^4 + 18 s^3 + 34 s^2 + 45 s + 25
```

Процедуры сложения передаточных функций показаны на рис. 12.6.

Аналогично осуществляются операции вычитания, умножения и деления передаточных функций с помощью операторов -, *, /.

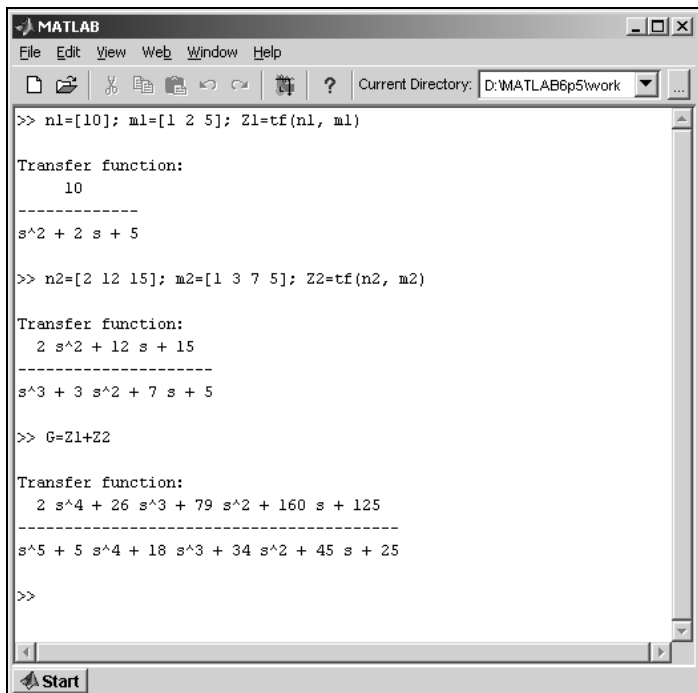


Рис. 12.6. Сложение передаточных функций

12.3.2. Функция *pzmap()*

Функция `pzmap()` показывает расположение полюсов и нулей передаточной функции на комплексной плоскости S . Функция имеет вид:

`pzmap(G)`

где G — имя передаточной функции.

Пример 12.7

Представить на плоскости S нули и полюсы функции

$$G(S) = \frac{6S^5 + 18S^4 + 25S^3 + 75S^2 + 4S + 12}{S^5 + 6S^4 + 14S^3 + 16S^2 + 9S + 2}.$$

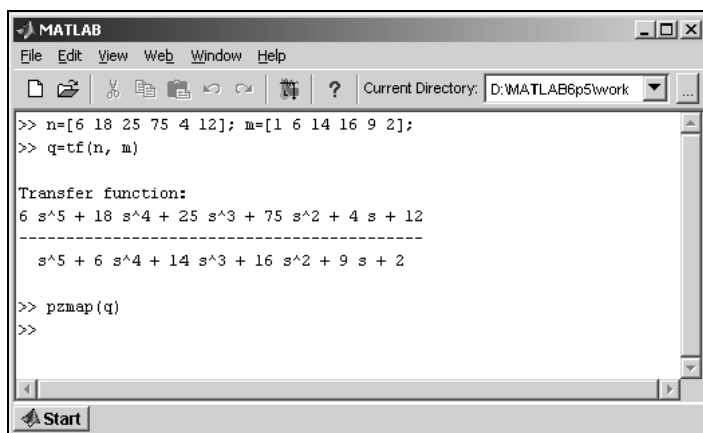


Рис. 12.7. Определение нулей и полюсов передаточной функции

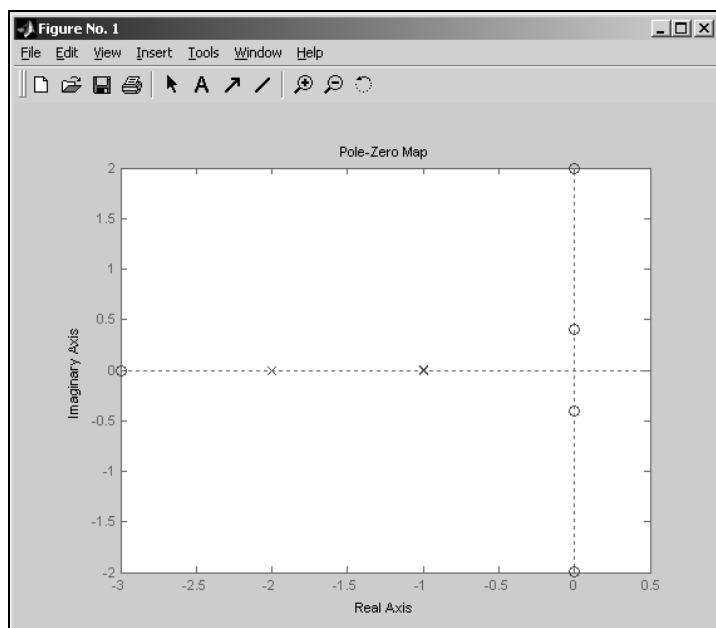


Рис. 12.8. Нули и полюсы передаточной функции

Решение:

```
>> n=[6 18 25 75 4 12];
>> m=[1 6 14 16 9 2];
>> q=tf(n,m)
Transfer function:
6 s^5 + 18 s^4 + 25 s^3 + 75 s^2 + 4 s + 12
-----
s^5 + 6 s^4 + 14 s^3 + 16 s^2 + 9 s + 2
>> pzmap(q)
```

Процедуры определения нулей и полюсов передаточной функции показаны на рис. 12.7, а ответ представлен в виде рис. 12.8 с расположением нулей (кружки) и полюсов (звездочки) на плоскости S . Обратите внимание, что на рисунке указаны только два полюса, хотя в знаменателе функции $G(S)$ полином пятой степени. Такой результат получен потому, что в нашем случае четыре корня знаменателя равны -1 .

12.3.3. Функция *series()*

Функция `series()` используется для образования передаточной функции системы, состоящей из последовательного соединения звеньев. Она имеет вид:

```
series(q1, q2)
```

где $q1$ и $q2$ — передаточные функции последовательно соединенных звеньев.

Пример 12.8

Структурная схема системы управления показана на рис. 12.9.

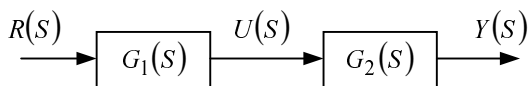


Рис. 12.9. Структурная схема системы

Необходимо получить передаточную функцию системы

$$G(S) = \frac{Y(S)}{R(S)},$$

если передаточные функции звеньев имеют вид:

$$G_1(S) = \frac{U(S)}{R(S)} = \frac{S+1}{S+2}, \quad G_2(S) = \frac{Y(S)}{U(S)} = \frac{1}{5S^2}.$$

Решение:

```
>> n1=[1 1];
>> m1=[1 2];
>> q1=tf(n1,m1);
>> n2=[1];
>> m2=[5 0 0];
>> q2=tf(n2,m2);
>> G=series(q1,q2)
Transfer function:
      s + 1
-----
5 s^3 + 10 s^2
```

Ответом является, как и следовало ожидать, произведение передаточных функций звеньев.

12.3.4. Функция *parallel()*

Функция `parallel()` используется для образования передаточной функции системы, состоящей из параллельных звеньев, и имеет вид:

```
parallel(q1, q2)
```

где `q1` и `q2` — передаточные функции параллельно соединенных звеньев.

Пример 12.9

Структурная схема системы управления приведена на рис. 12.10.

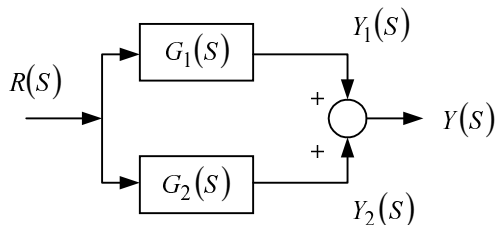


Рис. 12.10. Структурная схема системы, состоящая из параллельных звеньев

Необходимо получить передаточную функцию системы

$$G(S) = \frac{Y(S)}{R(S)},$$

если передаточные функции звеньев имеют вид:

$$G_1(S) = \frac{Y_1(S)}{R(S)} = \frac{S+1}{S^2+3S+1}, \quad G_2(S) = \frac{Y_2(S)}{R(S)} = \frac{S+2}{S^2+S+3}.$$

Решение:

```
>> n1=[1 1];
>> m1=[1 3 1];
>> q1=tf(n1,m1);
>> n2=[1 2];
>> m2=[1 1 3];
>> q2=tf(n2,m2);
>> G=parallel(q1,q2)
Transfer function:
      2 s^3 + 7 s^2 + 11 s + 5
-----
s^4 + 4 s^3 + 7 s^2 + 10 s + 3
```

12.3.5. Функция *feedback()*

Функция `feedback()` применяется для образования передаточной функции замкнутой системы по известным передаточным функциям разомкнутой системы и цепи обратной связи.

Она имеет вид:

$\text{feedback}(q, q_{oc}, \pm 1)$

где:

- ◆ q_{oc} — передаточная функция цепи обратной связи;
- ◆ ± 1 — указывает вид обратной связи (-1 — положительная, $+1$ — отрицательная).

Пример 12.10

Структурная схема системы управления приведена на рис. 12.11.

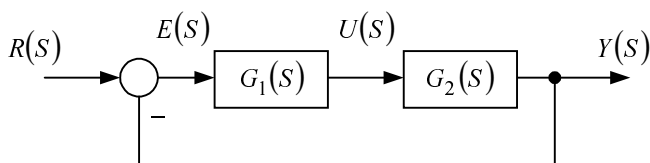


Рис. 12.11. Структурная схема системы управления

Передаточные функции звеньев имеют вид:

$$G_1(S) = \frac{S+1}{S+2}, \quad G_2(S) = \frac{1}{5S^2}.$$

Передаточная функция цепи обратной связи образует отрицательную обратную связь с коэффициентом передачи, равным 1.

Необходимо получить передаточную функцию замкнутой системы управления:

$$G(S) = \frac{Y(S)}{R(S)}.$$

Передаточная функция $G(S)$ определяется по выражению

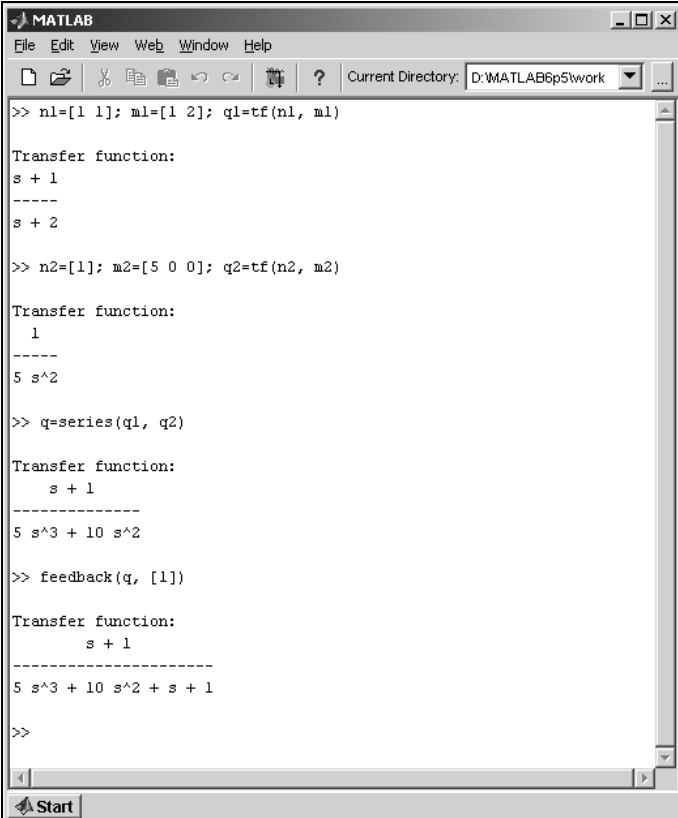
$$G(S) = \frac{G_1(S) \cdot G_2(S)}{1 + G_1(S) \cdot G_2(S)}.$$

Из этого выражения и структурной схемы видно, что для получения передаточной функции замкнутой системы необходимо вначале образовать с помощью функции `tf()` звенья $G_1(S)$ и $G_2(S)$, затем посредством функции `series()` образовать передаточную функцию разомкнутой системы и после этих процедур использовать функцию `feedback()` для образования передаточной функции замкнутой системы.

Программа образования передаточной функции замкнутой системы управления имеет вид:

```
>> n1=[1 1];
>> m1=[1 2];
>> q1=tf(n1,m1)
Transfer function:
s + 1
-----
s + 2
>> n2=[1];
>> m2=[5 0 0];
>> q2=tf(n2,m2)
Transfer function:
1
-----
5 s^2
>> q=series(q1,q2)
Transfer function:
s + 1
-----
5 s^3 + 10 s^2
>> feedback(q,[1])
Transfer function:
s + 1
-----
5 s^3 + 10 s^2 + s + 1
```

Решение задачи приведено на рис. 12.12.



```
MATLAB
File Edit View Web Window Help
Current Directory: D:\MATLAB6p5\work

>> n1=[1 1]; m1=[1 2]; q1=tf(n1, m1)

Transfer function:
s + 1
-----
s + 2

>> n2=[1]; m2=[5 0 0]; q2=tf(n2, m2)

Transfer function:
1
-----
5 s^2

>> q=series(q1, q2)

Transfer function:
s + 1
-----
5 s^3 + 10 s^2

>> feedback(q, [1])

Transfer function:
s + 1
-----
5 s^3 + 10 s^2 + s + 1

>>
```

Рис. 12.12. Образование передаточной функции системы с жесткой отрицательной обратной связью

Пример 12.11

Структурная схема системы управления приведена на рис. 12.13. Необходимо получить передаточную функцию замкнутой системы

$$G(S) = \frac{Y(S)}{R(S)}.$$

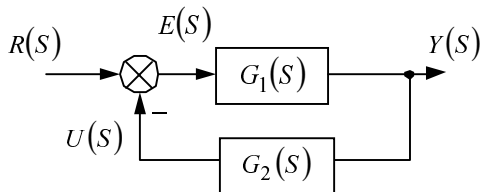


Рис. 12.13. Структурная схема системы с гибкой отрицательной обратной связью

Передаточные функции звеньев имеют вид:

$$G_1(S) = \frac{S+1}{S+2}, \quad G_2(S) = S+0,5.$$

Решение:

```
>> n1=[1 1];
>> m1=[1 2];
>> q1=tf(n1,m1);
>> n2=[1 0.5];
>> m2=[1];
>> q2=tf(n2,m2);
>> freeback(q1,q2,-1)
Transfer function:
      s + 1
-----
s^2 + 2.5 s + 2.5
```

12.3.6. Функция *minreal()*

Функция `minreal()` позволяет выполнить сокращения передаточной функции при наличии одинаковых сомножителей в числителе и знаменателе. Она имеет вид:

`minreal(G)`

где G — передаточная функция системы.

Пример 12.12

Передаточная функция системы управления имеет вид:

$$G(S) = \frac{S^2 + 4S + 3}{S^3 + S^2 + 2S + 2}.$$

Необходимо выполнить сокращение, если имеются одинаковые сомножители в числителе и знаменателе.

Решение. Представим функцию $G(s)$ на языке MATLAB и воспользуемся функцией `minreal()`. Программа будет иметь вид:

```
>> n=[1 4 3];  
>> m=[1 1 2 2];  
>> q=tf(n,m);  
>> mineral(q)  
Transfer function:  
    s + 3  
-----  
s^2 + 2
```

12.4. Исследование переходных процессов в системах управления

Исследовать переходные процессы в системах управления можно следующими методами:

- ◆ непосредственным решением дифференциальных уравнений, описывающих динамику системы управления;
- ◆ с помощью преобразования Лапласа передаточной функции системы;
- ◆ с помощью встроенной функции `step()`.

Все эти методы могут быть реализованы в системе MATLAB.

Для исследования переходных процессов с помощью преобразования Лапласа необходимо получить обратное преобразование Лапласа передаточной функции звена $Y(s)$ и представить его графически, а затем по виду графика определить вид переходного процесса (апериодический, колебательный) и его длительность.

Для графического воспроизведения результата в MATLAB используется функция `ezplot()`, имеющая вид:

```
ezplot(Y(t), xn, xk)
```

где:

- ◆ $Y(t)$ — функция, записанная в символьном виде (взята в кавычки);
- ◆ x_n, x_k — диапазон изменения аргумента, в нашем случае диапазон изменения t .

Пример 12.13

Пусть обратное преобразование функции имеет вид:

$$Y(t) = 0.5e^{-0.5t}.$$

Тогда программа воспроизведения графика в MATLAB будет иметь вид:

```
>> Y='0.5*exp(0.5*t)';  
>> ezplot(Y,0,3)
```

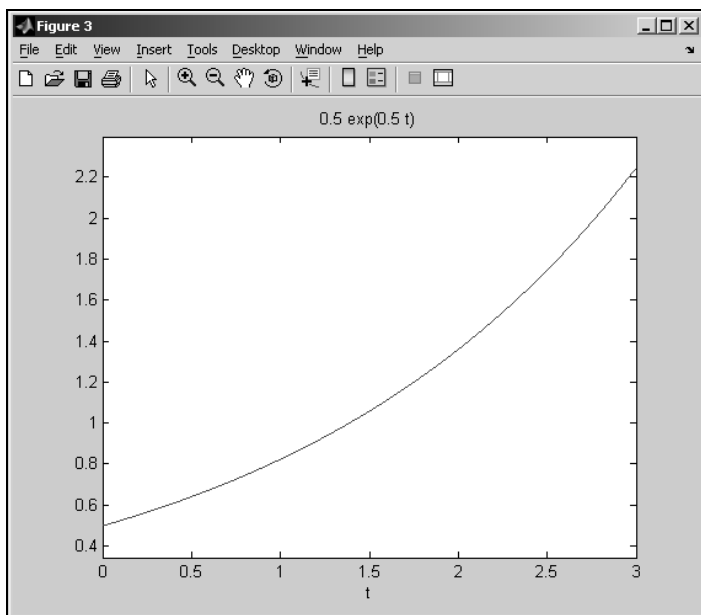


Рис. 12.14. График переходного процесса системы

После нажатия клавиши <Enter> на экране график функции (рис. 12.14) в диапазоне t , равном $[0—3]$.

12.4.1. Функция *step()*

Функция `step()` вычисляет реакцию системы управления на единичное ступенчатое воздействие. Если целью исследования является получение графика, то функция записывается в следующем виде:

```
step(q, t)
```

где:

- ♦ q — передаточная функция системы;
- ♦ t — время функционирования системы управления.

При этом график будет получен автоматически с указанием переменных по осям. Если же график необходим для иных целей с его сохранением, то функция записывается с указанием аргументов в левой части, например,

```
[y, t]= step(q, t)
```

После этого для образования графика применяется функция `plot(t,y)`. При этом перед функцией `step(q,t)` необходимо указать диапазон изменения t , например, в таком виде:

```
t=[0:0.1:3]
```

т. е. от 0 до 3 с шагом 0.1.

Пример 12.14

Определить переходную характеристику системы управления, передаточная функция которой имеет вид:

$$Y(S) = \frac{5400}{2S^2 + 2.5S + 5402}.$$

Решение:

```
>> n1=[5400];  
>> m1=[2 2.5 5402];
```

```
>> q=tf(n1,m1)
Transfer function:
      5400
-----
2 s^2 + 2.5 s + 5402
>> t=[0:0.005:3];
>> [y,t]=step(q,t);
>> plot(t,y)
```

Ответом будет график переходной характеристики, показанной на рис. 12.15.

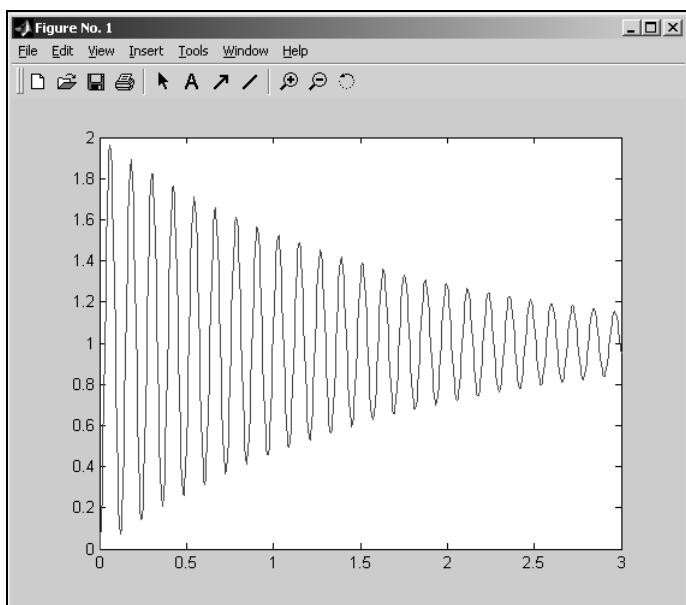


Рис. 12.15. Переходная характеристика системы

12.5. Частотные характеристики системы

Амплитудно-частотная и фазочастотная характеристики в системе MATLAB строятся с помощью функции `bode()`, имеющей вид:

```
bode(sys)
```

где *sys* — имя передаточной функции.

Полученные частотные характеристики называются *функциями Боде*.

Пример 12.15

Необходимо построить частотные характеристики звена, передаточная функция которого имеет вид:

$$Y(S) = \frac{0.5S + 1}{S(2S + 1)}.$$

Программа решения задачи имеет вид:

```
>> n=[0.5 1];  
>> m=[2 1 0];  
>> sys=tf(n,m);  
>> bode(sys)
```

После нажатия клавиши <Enter> на экране появится амплитудно-частотная и фазочастотная характеристики звена (рис. 12.16). Частота имеет размерность "рад./с." и представляется в логарифмическом масштабе. Амплитуда измеряется в децибелах, фаза — в градусах.

При построении диаграммы Боде в области желаемых частот используется функция

```
logspace(a, b, n)
```

где:

- ◆ *a* — начальное значение частоты;
- ◆ *b* — конечное значение частоты;
- ◆ *n* — число точек в диапазоне $[a; b]$.

Функция `bode()` при этом записывается в следующем виде:

```
bode(sys, w)
```

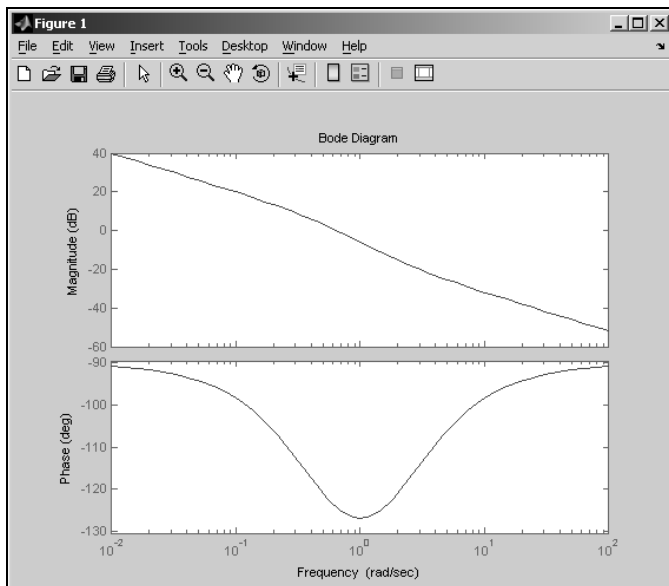


Рис. 12.16. Частотные характеристики звена

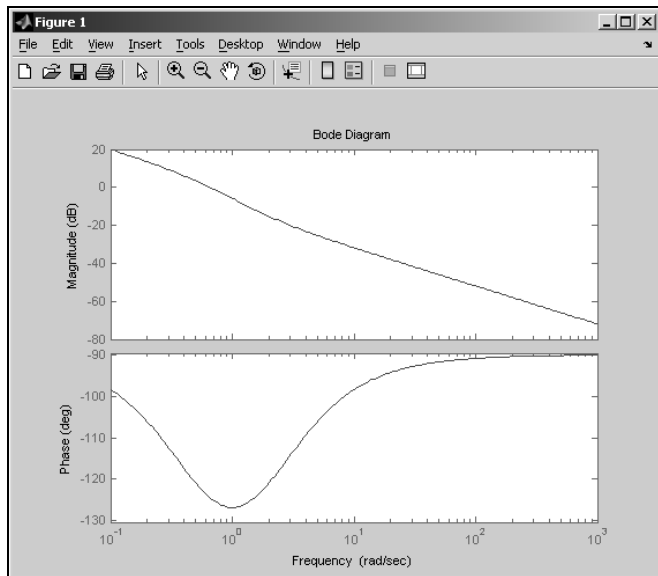


Рис. 12.17. График функции Боде

Для нашего примера программа будет иметь вид:

```
>> N=[0.5  1];  
>> M=[2  1  0];  
>> sys=tf(N, M);  
>> W=logspace(-1,3,200);  
>> bode(sys,W)
```

После нажатия клавиши <Enter> на экране отобразятся характеристики в заданном диапазоне частот (рис. 12.17).

Следует иметь в виду, что в функции `logspace()` значения a и b (в нашем случае $a=-1$, $b=3$) — это степени 10, т. е. 10^{-1} , 10^3 , что соответствует диапазону частот 0,1—3 рад/с.

12.5.1. Амплитудно-фазовая характеристика системы

Амплитудно-фазовую характеристику называют *диаграммой Найквиста*. Она применяется для анализа устойчивости по *критерию Найквиста*. Реализуется в системе MATLAB с помощью функции

```
nyquist(sys)
```

где `sys` — имя передаточной функции.

Пример 12.16

Необходимо построить диаграмму Найквиста звена, передаточная функция которого имеет вид:

$$Y(S) = \frac{0.5}{S^3 + 2S^2 + S + 0.5}.$$

Программа построения диаграммы имеет вид:

```
>> N=[0.5];  
>> M=[1  2  1  0.5];  
>> sys=tf(N, M);  
>> nyquist(sys)
```

После нажатия клавиши <Enter> на экране появится диаграмма Найквиста, приведенная на рис. 12.18.

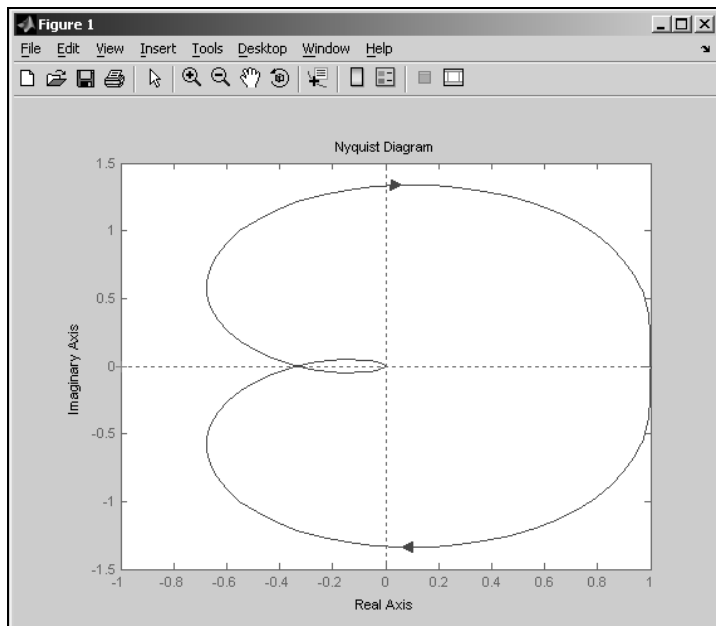


Рис. 12.18. Диаграмма Найквиста

12.5.2. Диаграмма Никольса

Сетка кривых линий на логарифмической амплитудно-фазовой диаграмме называется *диаграммой Никольса*. Линиям постоянных значений амплитуды M соответствуют децибелы, а линиям постоянных значений $N = \operatorname{tg} \varphi$ — градусы.

Диаграмма Никольса используется для исследования вопросов устойчивости автоматических систем. Для построения диаграммы Никольса MATLAB имеет специальную функцию:

```
nichols(sys, w)
```

где:

- ◆ sys — имя передаточной функции;
- ◆ w — частота, задаваемая пользователем в логарифмическом масштабе.

Пример 12.17

Передаточная функция системы имеет вид:

$$Y(S) = \frac{1}{S(S+1)(0.2S+1)}.$$

Программа построения диаграммы Никольса:

```
>> n=[1];  
>> m=[0.2 1.2 1 0];  
>> sys=tf(n,m);  
>> w=logspace(-1,1,400);  
>> nichols(sys,w);  
>> ngrid
```

После нажатия клавиши <Enter> на экране — диаграмма Никольса (рис. 12.19).

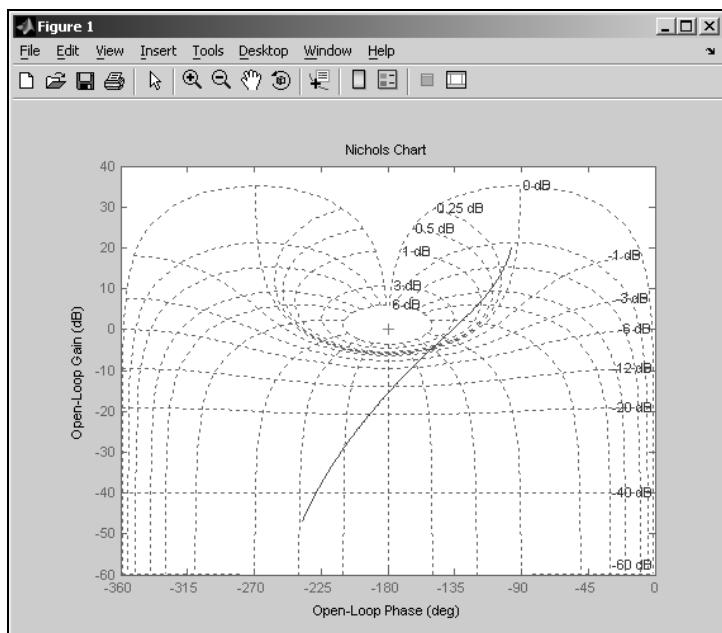


Рис. 12.19. Диаграмма Никольса

Функция `ngrid` вызывается для нанесения криволинейной сетки координат.

12.6. Пример анализа динамики системы управления

Структурная схема системы управления приведена на рис. 12.20.

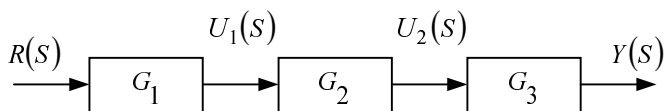


Рис. 12.20. Структурная схема разомкнутой системы управления

Передаточные функции звеньев имеют вид:

$$G_1(S) = K_1, \quad G_2(S) = \frac{K_2}{S(T_1S + 1)}, \quad G_3(S) = \frac{T_2S + 1}{T_3S + 1}.$$

Переменные имеют значения:

$$K_1 = 10, \quad K_2 = 5, \quad T_1 = 1.5, \quad T_2 = 3.5, \quad T_3 = 4.7.$$

Необходимо исследовать:

- ◆ динамические свойства разомкнутой системы, определив устойчивость системы и качество переходных процессов;
- ◆ влияние обратной связи на устойчивость и качество переходных процессов.

Решать поставленные задачи будем в такой последовательности:

1. Получение передаточной функции системы управления

$$G(S) = \frac{Y(S)}{R(S)}.$$

2. Определение нулей и полюсов передаточной функции разомкнутой системы.
3. Определение расположения нулей и полюсов на плоскости S .
4. Исследование качества переходных процессов.

5. Выбор, на основании предыдущих исследований, вида обратной связи.
6. Исследование устойчивости и качества переходных процессов в системе с обратной связью.

12.6.1. Образование передаточной функции разомкнутой системы

Образовать передаточную функцию системы можно лишь в том случае, если определены ее переменные. Программа MATLAB с символьными переменными K и T не работает. Присвоение переменным численных значений осуществляется оператором = (равно):

```
>> K1=10;
>> K2=5;
>> T1=1.5;
>>T2=3.5;
>> T3=4.7;
>> n1=[K1]; m1=[1]; z1=tf(n1,m1);
>> n2=[K2]; m2=[T1 1 0]; z2=tf(n2,m2);
>> n3=[T2 1]; m3=[T3 1]; z3=tf(n3,m3);
>> G=z1*z2*z3
```

Transfer function:

175 s + 50

7.05 s^3 + 6.2 s^2 + s

Обратите внимание на то, что в ответе имени функции $G(S)$ нет. Ее имя совпадает с именем произведения $z1*z2*z3$.

12.6.2. Определение нулей и полюсов передаточной функции $G(S)$

Программа имеет вид:

```
>> P=pole(G)
```

P =

0

```
-0.6667  
-0.2128  
>> N0=zero(G)
```

Ответа не последовало. Программа не нашла нулей передаточной функции. В целях экономии времени не будем искать причины отказа.

Для определения корня воспользуемся функцией `roots()`. Представим числитель передаточной функции $G(S)$ полином $175S + 50$ в векторном виде и воспользуемся функцией `roots()`:

```
>> q1=[175 50];  
>> N0=roots(q1)  
N0 =  
-0.2857
```

12.6.3. Расположение нулей и полюсов на комплексной плоскости

Выполним:

```
>> pzmap(G)
```

Результатом выполнения функции является комплексная плоскость S с расположением нулей (кружки) и полюсов (крестики). Значения нулей и полюсов совпадают с полученными в п. 3, что свидетельствует о правильности нашего решения. Комплексная плоскость с нулями и полюсами показана на рис. 12.21.

12.6.4. Анализ устойчивости системы

Анализ полюсов и нулей передаточной функции позволяет сделать вывод, что исследуемая система не устойчива, т. к. один из полюсов равен нулю.

12.6.5. Исследование качества переходного

Вычислим реакцию системы управления на единичную ступенчатую функцию, воспользовавшись функцией `step()`. На данном

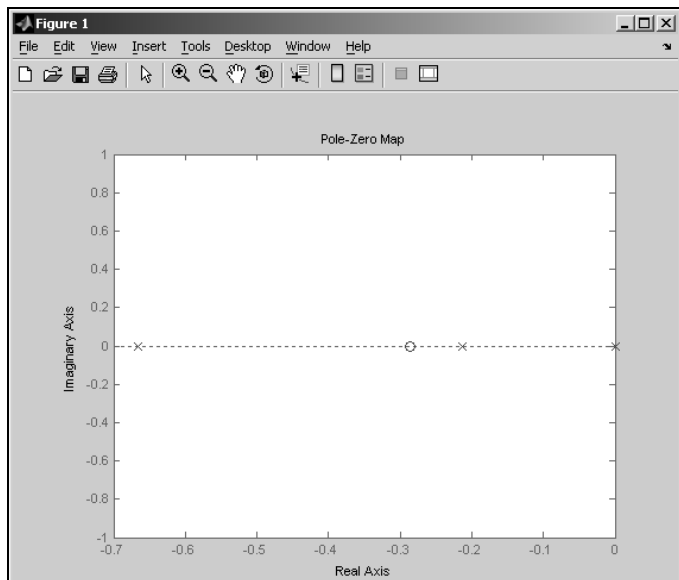


Рис. 12.21. Расположение нулей и полюсов на комплексной плоскости

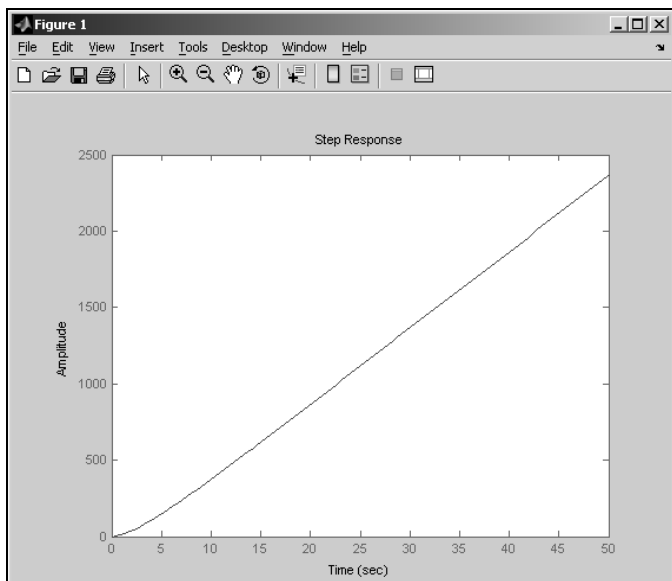


Рис. 12.22. Переходный процесс системы

этапе исследования нам достаточно получить лишь график реакции системы, поэтому не будем указывать временную область графика.

```
>> step(G)
```

На экране (рис. 12.22) — переходный процесс, представляющий собой возрастающую амплитуду выходного сигнала от времени. Система неустойчива.

12.6.6. Получение передаточной функции замкнутой системы

Исследуем теперь влияние обратной связи на динамику системы управления.

Передаточная функция замкнутой системы GOS определяется через передаточную функцию разомкнутой системы $G(S)$ при отрицательной обратной связи в соответствии с выражением:

$$GOS = \frac{G(S)}{1 + G(S)}.$$

В MATLAB это выражение реализуется с помощью функции `feedback()`, которая в нашем случае имеет вид:

```
>> GOS=feedback(G, [1])
Transfer function:
      175 s + 50
-----
7.05 s^3 + 6.2 s^2 + 176 s + 50
```

12.6.7. Определение нулей и полюсов передаточной функции замкнутой системы и расположение их на комплексной плоскости

Так как числители передаточной функции замкнутой и разомкнутой систем совпадают, то определим лишь полюсы функции $GOS(S)$ и отразим нули и полюсы на плоскости S .

```
>> PO=pole(GOS)
PO =
    -0.2967 + 4.9706i
    -0.2967 - 4.9706i
    -0.2860
>> pzmap(GOS)
```

На экране (рис. 12.23) появится комплексная плоскость S с нулями и полюсами передаточной функции $GOS(s)$.

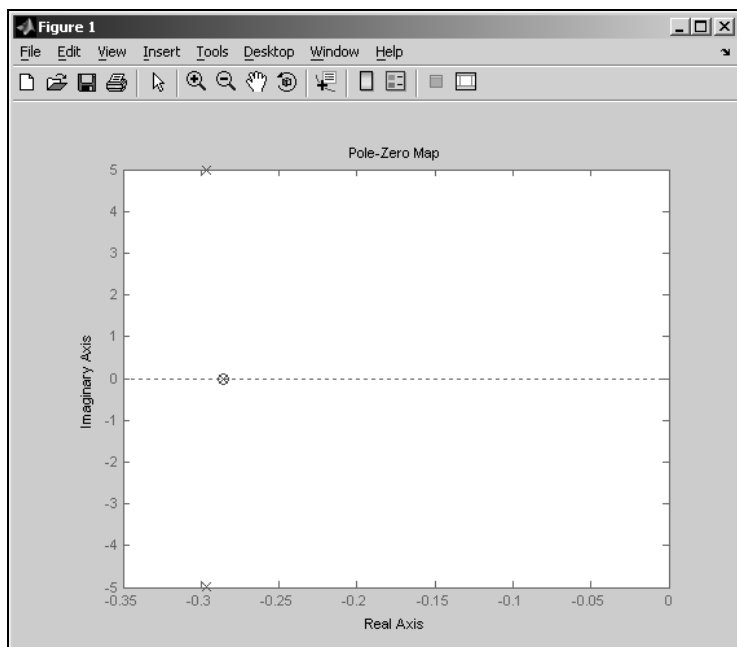


Рис. 12.23. Нули и полюсы передаточной функции

Анализ показал, что замкнутая система управления является устойчивой, ее нули и полюсы расположены в левой полуплоскости.

12.6.8. Переходные процессы замкнутой системы с жесткой отрицательной обратной связью

График переходного процесса получаем после реализации функции

`step(GOS)`

На экране — график, представляющий собой колебательный затухающий процесс (рис. 12.24).

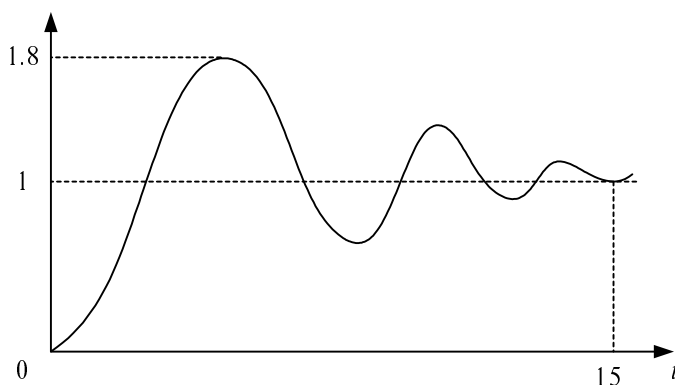


Рис. 12.24. График переходного процесса замкнутой системы

Длительность переходного процесса $\tau \approx 15$ с, величина перерегулирования $A \approx 1.8$.

12.6.9. Исследование устойчивости и качества переходных процессов системы управления при гибкой отрицательной обратной связи

Улучшить динамику системы управления можно, используя гибкую обратную связь по производной.

В качестве обратной связи применим блок с передаточной функцией

$$G_{oc} = TS + 1.$$

Результаты расчетов при значениях $T = 2, 0.5, 0.1$ таковы:

- ◆ при $T = 2$ процесс аperiodический с длительностью $\tau \approx 12$ с и отсутствием перерегулирования;
- ◆ при $T = 0.5$ процесс аperiodический с длительностью $\tau \approx 2.5$ с;
- ◆ при $T = 0.1$ процесс колебательный с длительностью $\tau \approx 3.3$ с и величиной перерегулирования $A \approx 1.35$.

Эти исследования при желании читатель выполнит самостоятельно.

12.7. Индивидуальные задания для исследования динамики систем управления

В следующих разделах приводятся два индивидуальных задания по исследованию динамики систем управления.

В первом из них ставятся задачи образования передаточной функции системы, определение условий устойчивости по значениям нулей и полюсов передаточной функции и образования переходных характеристик системы.

Второе задание посвящено исследованиям устойчивости и качества переходных процессов по переходным и частотным характеристикам системы.

12.7.1. Задание 1

Блок-схема системы управления приведена на рис. 12.25.

Необходимо исследовать устойчивость и качество переходных процессов разомкнутой системы управления, системы с жесткой и гибкой обратной связью. Решение задачи выполнить в последовательности, приведенной в примере *разд. 12.6*.

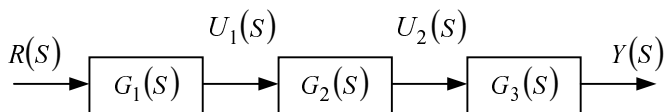


Рис. 12.25. Исходная структурная схема системы

При исследовании динамики системы управления с гибкой обратной связью передаточную функцию цепи обратной связи G_{oc} можно выбрать из следующих вариантов:

$$G_{oc}(S) = K,$$

$$G_{oc}(S) = TS + 1,$$

$$G_{oc}(S) = \frac{T_1 S + 1}{T_2 S + 1},$$

$$G_{oc}(S) = \frac{K}{TS + 1}.$$

Далее приведены передаточные функции звеньев системы для вариантов заданий.

◆ Вариант 1

$$G_1(S) = \frac{K_1}{T_1 S + 1}, \quad G_2(S) = \frac{K_2}{S(T_2 S + 1)}, \quad G_3(S) = \frac{T_3 S + 1}{T_4 S + 1},$$

$$K_1 = 20, \quad K_2 = 2, \quad T_1 = 1.5, \quad T_2 = 2, \quad T_3 = 5.4, \quad T_4 = 2.5.$$

◆ Вариант 2

$$G_1(S) = K_1, \quad G_2(S) = \frac{K_2}{S(T_1 S + 1)}, \quad G_3(S) = \frac{K_3}{T_2 S + 1},$$

$$K_1 = 30, \quad K_2 = 5, \quad K_3 = 12, \quad T_1 = 2.5, \quad T_2 = 0.8.$$

◆ Вариант 3

$$G_1(S) = K_1(T_1 S + 1), \quad G_2(S) = \frac{K_2}{S(T_2 S + 1)}, \quad G_3(S) = \frac{K_3(T_3 S + 1)}{T_4 S + 1},$$

$$K_1 = 32, \quad K_2 = 16, \quad K_3 = 2.5, \quad T_1 = 1.5, \quad T_2 = 5, \quad T_3 = 1.5, \quad T_4 = 2.$$

◆ Вариант 4

$$G_1(S) = K_1, \quad G_2(S) = \frac{K_2(T_1S+1)}{S(T_2S+1)}, \quad G_3(S) = \frac{T_3S+1}{T_4S+1},$$

$$K_1 = 20, \quad K_2 = 25, \quad T_1 = 1.5, \quad T_2 = 3, \quad T_3 = 0.5, \quad T_4 = 1.4.$$

◆ Вариант 5

$$G_1(S) = \frac{K_1}{T_1S+1}, \quad G_2(S) = \frac{K_2}{S(T_2S+1)}, \quad G_3(S) = \frac{K_3}{T_3S+1},$$

$$K_1 = 20, \quad K_2 = 10, \quad K_3 = 15, \quad T_1 = 2, \quad T_2 = 1.5, \quad T_3 = 0.5.$$

◆ Вариант 6

$$G_1(S) = \frac{K_1}{T_1S+1}, \quad G_2(S) = \frac{K_2(T_2S+1)}{S(T_3S+1)}, \quad G_3(S) = \frac{K_3}{T_4S+1},$$

$$K_1 = 15, \quad K_2 = 20, \quad K_3 = 7, \quad T_1 = 2, \quad T_2 = 0.5, \quad T_3 = 3, \quad T_4 = 1.4.$$

◆ Вариант 7

$$G_1(S) = \frac{K_1(T_1S+1)}{T_2S+1}, \quad G_2(S) = \frac{K_2}{S(T_3S+1)}, \quad G_3(S) = \frac{T_4S+1}{T_5S+1},$$

$$K_1 = 12, \quad K_2 = 15, \quad T_1 = 1.4, \quad T_2 = 0.4, \quad T_3 = 2, \quad T_4 = 5.$$

◆ Вариант 8

$$G_1(S) = K_1(T_1S+1), \quad G_2(S) = \frac{K_2}{S(T_2S+1)}, \quad G_3(S) = \frac{K_3}{T_3S+1},$$

$$K_1 = 5, \quad K_2 = 15, \quad K_3 = 12, \quad T_1 = 0.8, \quad T_2 = 0.5, \quad T_3 = 2.$$

◆ Вариант 9

$$G_1(S) = K_1, \quad G_2(S) = \frac{K_2(T_1S+1)}{S(T_2S+1)}, \quad G_3(S) = \frac{T_3S+1}{T_4S+1},$$

$$K_1 = 20, \quad K_2 = 25, \quad T_1 = 1.5, \quad T_2 = 3, \quad T_3 = 0.5, \quad T_4 = 1.4.$$

◆ Вариант 10

$$G_1(S) = \frac{K_1}{S(T_1S+1)}, G_2(S) = \frac{T_2S+1}{T_3S+1}, G_3(S) = \frac{K_2(T_4S+1)}{T_5S+1},$$

$$K_1 = 5, K_2 = 35, T_1 = 0.5, T_2 = 2, T_3 = 3.2, T_4 = 1.5, T_5 = 0.2.$$

◆ Вариант 11

$$G_1(S) = \frac{T_1S+1}{T_2S+1}, G_2(S) = \frac{K_1}{T_3S+1}, G_3(S) = \frac{K_2(T_4S+1)}{S(T_5S+1)},$$

$$K_1 = 100, K_2 = 5.6, T_1 = 3, T_2 = 0.5, T_3 = 2.5, T_4 = 5, T_5 = 2.$$

◆ Вариант 12

$$G_1(S) = \frac{K_1}{S(T_1S+1)}, G_2(S) = K_2, G_3(S) = \frac{K_3(T_2S+1)}{T_3S+1},$$

$$K_1 = 10, K_2 = 20, K_3 = 6.7, T_1 = 2, T_2 = 0.5, T_3 = 0.1.$$

◆ Вариант 13

$$G_1(S) = \frac{K_1}{S(T_1S+1)}, G_2(S) = \frac{K_2(T_2S+1)}{T_3S+1}, G_3(S) = K_3(T_4S+1),$$

$$K_1 = 5, K_2 = 7, K_3 = 12, T_1 = 0.5, T_2 = 1.5, T_3 = 2, T_4 = 1.8.$$

◆ Вариант 14

$$G_1(S) = \frac{K_1(T_1S+1)}{S(T_2S+1)}, G_2(S) = K_2, G_3(S) = \frac{K_3(T_3S+1)}{T_4S+1},$$

$$K_1 = 5, K_2 = 7, K_3 = 10, T_1 = 1.2, T_2 = 3.2, T_3 = 1.8, T_4 = 2.$$

◆ Вариант 15

$$G_1(S) = \frac{K_1}{T_1S+1}, G_2(S) = \frac{K_2}{T_2S+1}, G_3(S) = \frac{K_3(T_3S+1)}{S(T_4S+1)},$$

$$K_1 = 2, K_2 = 8, K_3 = 12, T_1 = 2, T_2 = 3.5, T_3 = 0.2, T_4 = 0.5.$$

◆ Вариант 16

$$G_1(S) = \frac{K_1}{T_1 S + 1}, \quad G_2(S) = \frac{K_2}{T_2 S + 1}, \quad G_3(S) = \frac{K_3(T_3 S + 1)}{S(T_4 S + 1)},$$

$$K_1 = 20, \quad K_2 = 12, \quad K_3 = 8, \quad T_1 = 0.5, \quad T_2 = 0.1, \quad T_3 = 0.1, \quad T_4 = 2.$$

◆ Вариант 17

$$G_1(S) = \frac{(T_1 S + 1)K_1}{T_2 S + 1}, \quad G_2(S) = \frac{K_2}{S(T_3 S + 1)}, \quad G_3(S) = K_3(T_4 S + 1),$$

$$K_1 = 5, \quad K_2 = 100, \quad K_3 = 1.5, \quad T_1 = 0.2, \quad T_2 = 2, \quad T_3 = 1.2, \quad T_4 = 4.2.$$

◆ Вариант 18

$$G(S) = \frac{K_1(T_1 S + 1)}{T_2 S + 1}, \quad G_2(S) = \frac{K_2}{S(T_3 S + 1)}, \quad G_3(S) = \frac{K_3}{T_4 S + 1},$$

$$K_1 = 12, \quad K_2 = 50, \quad K_3 = 2, \quad T_1 = 0.5, \quad T_2 = 1.5, \quad T_3 = 2, \quad T_4 = 2.5.$$

◆ Вариант 19

$$G_1(S) = \frac{K_1(T_1 S + 1)}{T_2 S + 1}, \quad G_2(S) = K_2(T_3 S + 1), \quad G_3(S) = \frac{K_3}{S(T_4 S + 1)},$$

$$K_1 = 12, \quad K_2 = 5, \quad K_3 = 50, \quad T_1 = 2, \quad T_2 = 0.5, \quad T_3 = 3.5, \quad T_4 = 2.5.$$

◆ Вариант 20

$$G_1(S) = \frac{K_1(T_1 S + 1)}{T_2 S + 1}, \quad G_2(S) = \frac{K_2}{S(T_3 S + 1)}, \quad G_3(S) = K_3,$$

$$K_1 = 5, \quad K_2 = 70, \quad K_3 = 12, \quad T_1 = 0.5, \quad T_2 = 2, \quad T_3 = 1.5.$$

◆ Вариант 21

$$G_1(S) = \frac{K_1}{T_1 S + 1}, \quad G_2(S) = \frac{K_2(T_2 S + 1)}{S(T_3 S + 1)}, \quad G_3(S) = K_3(T_4 S + 1),$$

$$K_1 = 5, \quad K_2 = 50, \quad K_3 = 7, \quad T_1 = 0.5, \quad T_2 = 1.5, \quad T_3 = 2, \quad T_4 = 0.8.$$

♦ Вариант 22

$$G_1(S) = \frac{K_1(T_1S+1)}{T_2S+1}, \quad G_2(S) = \frac{K_2}{S(T_3S+1)}, \quad G_3(S) = \frac{K_3}{T_4S+1},$$

$$K_1 = 5, \quad K_2 = 30, \quad K_3 = 8, \quad T_1 = 2, \quad T_2 = 0.5, \quad T_3 = 0.8, \quad T_4 = 1.5.$$

♦ Вариант 23

$$G_1(S) = \frac{K_1(T_1S+1)}{T_2S+1}, \quad G_2(S) = \frac{K_2}{S(T_3S+1)}, \quad G_3(S) = K_3,$$

$$K_1 = 3, \quad K_2 = 40, \quad K_3 = 12, \quad T_1 = 0.5, \quad T_2 = 2.5, \quad T_3 = 3.5.$$

♦ Вариант 24

$$G_1(S) = \frac{K_1(T_1S+1)}{S(T_2S+1)}, \quad G_2(S) = \frac{K_2(T_3S+1)}{T_4S+1}, \quad G_3(S) = K_3,$$

$$K_1 = 7, \quad K_2 = 28, \quad K_3 = 12, \quad T_1 = 2, \quad T_2 = 0.8, \quad T_3 = 0.5, \quad T_4 = 3.$$

♦ Вариант 25

$$G_1(S) = \frac{K_1}{T_1S+1}, \quad G_2(S) = \frac{K_2(T_2S+1)}{S(T_3S+1)}, \quad G_3(S) = K_3(T_4S+1),$$

$$K_1 = 3.5, \quad K_2 = 32, \quad K_3 = 8, \quad T_1 = 1.5, \quad T_2 = 2, \quad T_3 = 0.5, \quad T_4 = 3.$$

12.7.2. Задание 2

Постановка задачи

Предлагаются четыре звена системы управления в виде передаточных функций. Варианты звеньев приведены в индивидуальных заданиях. Необходимо исследовать их характеристики, определив:

- ♦ переходные процессы с помощью преобразования Лапласа;
- ♦ реакцию звена на единичное ступенчатое воздействие;
- ♦ амплитудно-частотную и фазочастотную характеристики;
- ♦ амплитудно-фазовую характеристику;

- ◆ диаграмму Никольса;
- ◆ показатели качества переходного процесса (вид переходного процесса и его длительность, величину перерегулирования);
- ◆ запас устойчивости по амплитуде и фазе.

Исследования выполнить с помощью универсального программного средства MATLAB и специализированного пакета прикладных программ Control System Toolbox. Переходную характеристику звена следует получить путем обратного преобразования Лапласа передаточной функции звена.

Варианты индивидуальных заданий и передаточных функций

Варианты заданий приведены в табл. 12.1.

Цифры в графе "звенья" являются номерами передаточных функций звеньев.

Решения необходимо получить в виде формул, графиков и детального анализа полученных результатов.

Таблица 12.1. Варианты заданий

Номер					
Вариант	1	2	3	4	5
Звенья	1, 3, 7, 11	2, 5, 8, 10	3, 6, 9, 11	4, 2, 5, 8	1, 4, 5, 9
Вариант	6	7	8	9	10
Звенья	1, 4, 6, 10	2, 6, 8, 9	3, 5, 7, 8	1, 4, 8, 11	2, 3, 6, 9
Вариант	11	12	13	14	15
Звенья	3, 6, 8, 10	4, 5, 9, 11	1, 4, 7, 10	2, 5, 8, 11	3, 6, 7, 9
Вариант	16	17	18	19	20
Звенья	4, 7, 9, 11	1, 4, 6, 10	2, 6, 7, 9	3, 5, 8, 11	4, 7, 8, 10
Вариант	21	22	23	24	25
Звенья	1, 5, 6, 9	2, 7, 9, 11	3, 4, 7, 10	4, 5, 7, 9	1, 6, 9, 11

Далее приведены передаточные функции звеньев системы.

◆ Вариант 1

$$Y(S) = \frac{TS}{TS + 1}$$

а) $T = 0.5$;

б) $T = 5$.

◆ Вариант 2

$$Y(S) = \frac{T_2 S}{T_1 S + 1}$$

а) $T_1 = 0.2$, $T_2 = 1$;

б) $T_1 = 1$, $T_2 = 0.2$.

◆ Вариант 3

$$Y(S) = \frac{T_1 S + 1}{T_2 S + 1}$$

а) $T_1 = 0.3$, $T_2 = 1.5$;

б) $T_1 = 1.5$, $T_2 = 0.3$.

◆ Вариант 4

$$Y(S) = \frac{K}{TS + 1}$$

а) $K = 10$, $T = 0.2$;

б) $K = 50$, $T = 1$.

◆ Вариант 5

$$Y(S) = \frac{1}{TS + 1}$$

а) $T = 0.5$;

б) $T = 4$.

◆ Вариант 6

$$Y(S) = \frac{K}{S(TS + 1)}$$

а) $K = 10$, $T = 0.2$;

б) $K = 20$, $T = 1$.

◆ Вариант 7

$$Y(S) = \frac{K}{S}$$

а) $K = 10$;

б) $K = 100$.

◆ Вариант 8

$$Y(S) = \frac{KS}{(T_1S + 1)(T_2S + 1)}$$

а) $K = 10, T_1 = 0.1, T_2 = 1$;

б) $K = 100, T_1 = 0.1, T_2 = 1$.

◆ Вариант 9

$$Y(S) = \frac{K(T_1S + 1)}{S(T_2S + 1)(T_3S + 1)}$$

а) $K = 10, T_1 = 0.1, T_2 = 0.5, T_3 = 1$;

б) $K = 10, T_1 = 0.8, T_2 = 0.5, T_3 = 1$.

◆ Вариант 10

$$Y(S) = \frac{K(T_1S + 1)}{S^2(T_2S + 1)}$$

а) $K = 20, T_1 = 0.5, T_2 = 1$;

б) $K = 20, T_1 = 1, T_2 = 0.5$.

◆ Вариант 11

$$Y(S) = \frac{K}{(T_1S + 1)(T_2S + 1)(T_3S + 1)}$$

а) $K = 10, T_1 = 0.1, T_2 = 0.7, T_3 = 1.5$;

б) $K = 20, T_1 = 1, T_2 = 2, T_3 = 3$.

Литература

1. Андриевский Б., Фрадков А. Избранные главы теории автоматического управления с примерами на языке Matlab. — СПб.: Наука, 1999.
2. Воробьева Г. Н., Данилова А. Н. Практикум по вычислительной математике. — М.: Высшая школа, 1990.
3. Гнеденко Б. В., Коваленко И. Н. Введение в теорию массового обслуживания. — М.: Наука, 1987.
4. Гультияев А. Визуальное моделирование в среде Matlab. Учебный курс. — СПб.: Питер, 2000.
5. Гутер Р. С., Резниковский П. Т. Программирование и вычислительная математика, выпуск 2. — М.: Наука, 1971.
6. Демидович Б. П., Марон И. А. Основы вычислительной математики. — М.: Наука, 1970.
7. Дорф Р., Бишоп Р. Современные системы управления. — М.: Лаборатория базовых знаний, 2002.
8. Дьяконов В., Круглов В. Математические пакеты расширения Matlab. Специальный справочник. — СПб.: Питер, 2001.
9. Дьяконов В. Учебный курс. — СПб.: Питер, 2001.
10. Егоренков Д. Л., Фрадков А. Л., Харламов В. Ю. Основы математического моделирования. — СПб.: БГТУ, 1996.
11. Компьютер для студентов, аспирантов и преподавателей. Самоучитель, под редакцией В. Б. Комягина. — Можайск: Триумф, 2002.

12. Копченова Н. В., Марон И. А. Вычислительная математика в примерах и задачах. — М.: Наука, 1972.
13. Половко А., Бутусов П. Интерполяция. Методика и компьютерные технологии их реализации. — СПб.: БХВ-Петербург, 2004.
14. Половко А. Derive для студента. — СПб.: БХВ-Петербург, 2005.
15. Потемкин В. Г. Инструментальные средства Matlab 5.x. — М.: Диалог-МИФИ, 2000.
16. Потемкин В. Система инженерных и научных расчетов Matlab 5.x. — М.: Диалог-МИФИ, 1999.
17. Пулькин С. П. Вычислительная математика. — М.: Просвещение, 1972.
18. Форсайт Дж., Малькольм М., Моулер К. Машинные методы математических вычислений: Пер. с англ. — М.: Мир, 1980.