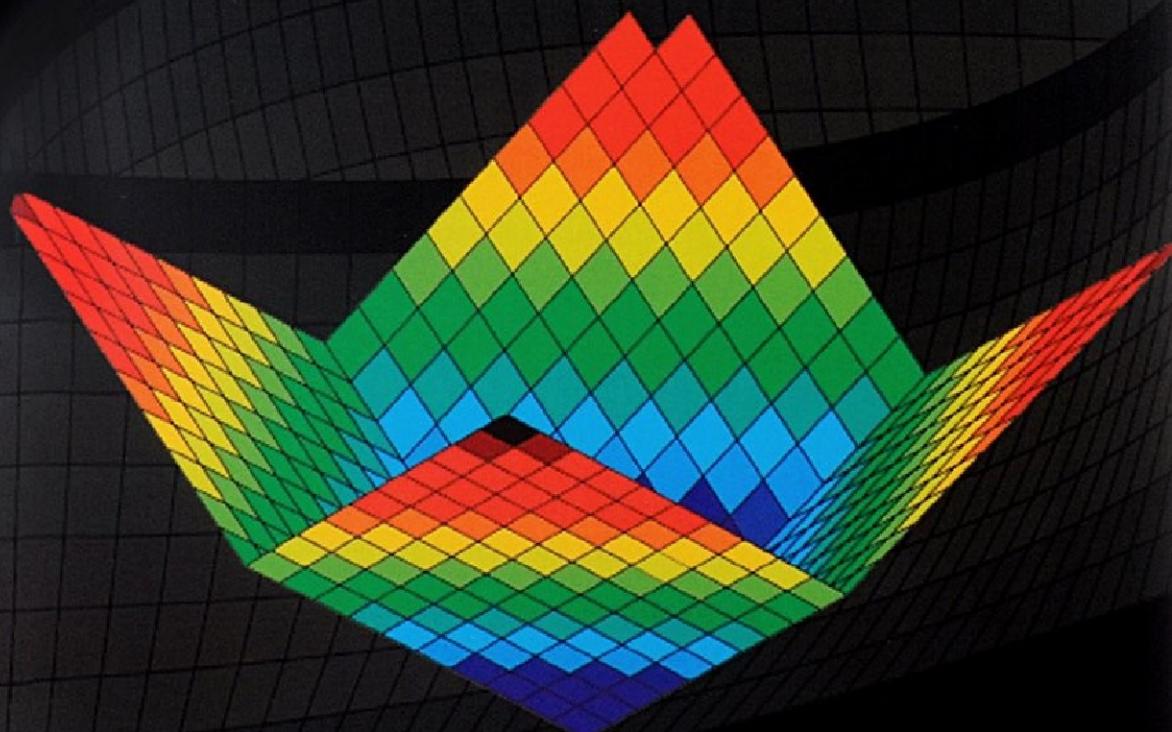


MATLAB

Теория и практика



Амос Гилат

Амос Гилат

MATLAB

Теория и практика

MATLAB[®]

An Introduction with Applications

Fifth Edition

Amos Gilat

Department of Mechanical and Aerospace Engineering
The Ohio State University

WILEY

MATLAB[®]

Теория и практика

5-е издание

Амос Гилат

Факультет механики и космической техники
Университет штата Огайо



Москва, 2016

УДК 51-37:004.9MATLAB

ББК 22.1с

Г47

Г47 Амос Гилат

MATLAB. Теория и практика. 5-е изд. / Пер. с англ. Смоленцев Н. К. – М.: ДМК Пресс, 2016. – 416 с.: ил.

ISBN 978-5-97060-183-9

Данная книга предлагает практическое введение в MATLAB – пакет прикладных программ для решения задач технических вычислений и однотипный язык программирования. Издание охватывает все, что необходимо для эффективного использования MATLAB, от простых арифметических действий со скалярами до создания и использования массивов, трехмерных графиков и решения дифференциальных уравнений. Снимки экранов, учебные примеры, работающие примеры программ и домашние задания с вопросами по математике, физике и инженерным наукам – все это делает освоение программы MATLAB эффективным и основательным.

Издание предназначено в первую очередь студентам техническим вузов, а также инженерам и научным работникам, использующим MATLAB в своей работе.

УДК 51-37:004.9MATLAB

ББК 22.1с

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

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-11862-986-4 (англ.)
ISBN 978-5-97060-183-9 (рус.)

© 2015, 2011 John Wiley & Sons, Inc.
© Оформление, издание, ДМК Пресс, 2016

Посвящается моим родителям Шошане и Хаиму Гельбваксам



ОГЛАВЛЕНИЕ

Предисловие	11
Введение	13
Глава 1.	
Начало работы с MATLAB	17
1.1. Запуск MATLAB, окна MATLAB	17
1.2. Работа в командном окне.....	21
1.3. Арифметические операции со скалярами	23
1.3.1. Приоритет операций.....	23
1.3.2. Использование MATLAB как калькулятор.....	24
1.4. Форматы вывода	24
1.5. Встроенные элементарные математические функции.....	25
1.6. Определение скалярных переменных	28
1.6.1. Оператор присвоения.....	28
1.6.2. Правила для имен переменных	30
1.6.3. Предопределенные переменные и зарезервированные слова.....	30
1.7. Полезные команды для управления переменными.....	31
1.8. Файлы сценария	32
1.8.1. Замечания о файлах сценариях	32
1.8.2. Создание и сохранение файлов сценариев.....	33
1.8.3. Выполнение файла сценария.....	34
1.8.4. Текущий каталог	34
1.9. Примеры применений MATLAB.....	36
1.10. Задачи	39
Глава 2.	
Создание массивов	46
2.1. Создание одномерных массивов (векторов)	46
2.2. Создание двумерных массивов (матриц)	49
2.2.1. Команды zeros, ones и eye.....	51
2.3. Замечания о переменных в MATLAB	52
2.4. Оператор транспонирования	52
2.5. Адресация (индексация) массива	53
2.5.1. Вектор	53



2.5.2. Матрица	54
2.6. Использование двоеточия : в адресации массивов	55
2.7. Добавление элементов к существующим переменным	57
2.8. Удаление элементов	59
2.9. Встроенные функции для управления массивами	60
2.10. Строки символов и строки как переменные	63
2.11. Задачи	65

Глава 3.

Математические операции с массивами **74**

3.1. Сложение и вычитание.....	75
3.2. Умножение массивов.....	76
3.3. Деление массивов	79
3.4. Поэлементные операции	83
3.5. Использование массивов во встроенных математических функциях MATLAB.....	85
3.6. Встроенные функции для анализа массивов	86
3.7. Генерация случайных чисел.....	88
3.8. Примеры приложений MATLAB.....	91
3.9. Задачи	96

Глава 4.

Использование файлов сценариев и управление данными .. **105**

4.1. Рабочее пространство MATLAB и окно рабочего пространства.....	106
4.2. Входные параметры файла сценария.....	107
4.3. Команды вывода	110
4.3.1. Команда disp.....	111
4.3.2. Команда fprintf	113
4.4. Команды save и load	120
4.4.1. Команда save	121
4.4.2. Команда load	122
4.5. Импорт и экспорт данных	123
4.5.1. Команды для импорта и экспорта данных	124
4.5.2. Использование Мастера импорта	126
4.6. Примеры приложений MATLAB.....	127
4.7. Задачи	132

Глава 5.

Двумерные графики..... **143**

5.1. Команда plot	143
5.1.1. График определенных данных.....	148

5.1.2. График функции	148
5.2. Команда fplot	150
5.3. Графическое изображение нескольких графиков в одном окне	151
5.3.1. Использование команды plot	152
5.3.2. Использование команд hold on и hold off	153
5.3.3. Использование команды line	153
5.4. Форматирование окна графика.....	154
5.4.1. Использование команд форматирования окна графика	154
5.4.2. Форматирование графика используя редактор графиков	158
5.5. Графики с логарифмическими осями.....	158
5.6. Графики с планками погрешностей.....	160
5.7. Графики специального вида.....	162
5.8. Гистограммы	163
5.9. Графики в полярных координатах.....	166
5.10. Расположение нескольких окон графиков на одной странице	167
5.11. Несколько окон графиков Figure.....	167
5.12. Построение графиков с использованием ленты инструментов PLOTS	169
5.13. Примеры приложений MATLAB	170
5.14. Задачи	175

Глава 6.

Программирование в MATLAB **185**

6.1. Операторы сравнения и логические операторы	186
6.2. Условные операторы.....	193
6.2.1. Структура if-end	194
6.2.2. Структура if-else-end.....	195
6.2.3. Структура if-elseif-else-end	195
6.3. Оператор переключения switch-case.....	198
6.4. Циклы	201
6.4.1. Циклы for-end	201
6.4.2. Циклы while-end	205
6.5. Вложенные циклы и вложенные условные операторы	208
6.6. Команды break и continue.....	210
6.7. Примеры приложений MATLAB	211
6.8. Задачи	219

Глава 7.

Определенные пользователем функции и файлы функций .. **229**

7.1. Создание файла функции	230
7.2. Структура файла функции	231



7.2.1. Стока определения функции	231
7.2.2. Входные и выходные аргументы	232
7.2.3. Стока H1 и текстовые строки справки	234
7.2.4. Тело функции	234
7.3. Локальные и глобальные переменные	235
7.4. Сохранение файла функции	236
7.5. Использование пользовательских функций	236
7.6. Примеры простых пользовательских функций	237
7.7. Сравнение файлов функций и скриптов-файлов	239
7.8. Анонимные функции	240
7.9. Функции от функций	243
7.9.1. Использование дескрипторов функций для передачи функции в функцию от функции	243
7.9.2. Использование имени функции для передачи функции в функцию от функции	246
7.10. Подфункции	249
7.11. Вложенные функции	250
7.12. Примеры приложений MATLAB	253
7.13. Задачи	256

Глава 8.

Многочлены, подбор кривых и интерполяция 270

8.1. Многочлены	270
8.1.1. Значение многочлена	271
8.1.2. Корни многочлена	272
8.1.3. Сложение, умножение и деление многочленов	273
8.1.4. Производные многочленов	275
8.2. Подбор кривой	276
8.2.1. Подбор кривой многочленами; функция polyfit	276
8.2.2. Подбор кривой другими функциями	280
8.3. Интерполяция	284
8.4. Базовый интерфейс для подбора Basic fitting	287
8.5. Примеры приложений MATLAB	290
8.6. Задачи	295

Глава 9.

Приложения в численном анализе 305

9.1. Решение уравнения с одной переменной	305
9.2. Нахождение минимума или максимума функции	308
9.3. Численное интегрирование	310
9.4. Обыкновенные дифференциальные уравнения	312



9.5. Примеры приложений MATLAB	317
9.6. Задачи	323

Глава 10.**Трехмерные графики 333**

10.1. Графики линий	333
10.2. Сети и графики поверхностей	334
10.3. Специальные графики.....	340
10.4. Команда view.....	343
10.5. Примеры приложений MATLAB	345
10.6. Задачи	350

Глава 11.**Символьная математика 356**

11.1. Символьные объекты и символьные выражения.....	357
11.1.1. Создание символьных объектов.....	357
11.1.2. Создание символьных выражений	359
11.1.3. Команда findsym и значение символьной переменной по умолчанию.....	362
11.2. Изменение вида существующего символьного выражения	363
11.2.1. Команды collect, expand и factor	363
11.2.2. Команды simplify и simple	365
11.2.3. Команда pretty	367
11.3. Решение алгебраических уравнений.....	367
11.4. Дифференцирование.....	372
11.5. Интегрирование.....	373
11.6. Решение обыкновенных дифференциальных уравнений	375
11.7. Графическое изображение символьных выражений.....	378
11.8. Численные расчеты с символьными выражениями	381
11.9. Примеры приложений MATLAB	384
11.10. Задачи	392

Приложение.**Сводка символов, команд и функций 402****Предметный указатель 413**



ПРЕДИСЛОВИЕ

MATLAB – это очень популярный язык для технических вычислений, используемых студентами, инженерами и учеными в университетах, научно-исследовательских институтах и в различных отраслях промышленности во всем мире. Это программное обеспечение популярно, потому что оно мощно и удобно. Для студентов младших курсов университетов это может быть следующим инструментом для использования после графических калькуляторов в средней школе.

Эта книга была написана после нескольких лет преподавания этого программного обеспечения студентам младших курсов в качестве вводного прикладного курса. Цель состояла в том, чтобы написать книгу, которая представляет это программное обеспечение дружественным, непугающим образом. Поэтому, книга написана на простом и понятном языке. Во многих местах для перечисления фактов и деталей, которые связаны с определенной темой, используются маркеры и списки вместо длинного текста. Книга включает многочисленные примеры типовых задач математики, науки и техники, которые подобны задачам, с которыми встречаются новые пользователи MATLAB.

Этот пятый выпуск книги обновлен и соответствует выпуску MATLAB 2013b. Кроме того, были исправлены задачи в конце каждой главы. В главах 1–8 около 80 % задач новые или отличаются от задач предыдущих изданий.

Я хотел бы выразить благодарность некоторым из своих коллег в Университете штата Огайо. Профессору Ричарду Фреулеру за его комментарии и доктору Майку Парку за то, что он просмотрел разделы книги и за предложенные изменения. Я также ценю участие и поддержку профессоров Роберта Гастэфсона, Джона Демеля и доктора Джона Мерилла из Технического образовательного инновационного центра в Университете штата Огайо. Особая благодарность профессору Майку Личтенстейджеру (OSU) и моей дочери Тэл Гилат (Университет Маркетт), которая тщательно просмотрела первый выпуск книги и дала ценные комментарии и критические замечания.

Профессор Брайан Харпер (OSU) внес существенный вклад в составление новых задач в конце глав в данном издании книги.

Я хотел бы выразить свою признательность всем тем, кто просмотрел более ранние редакции текста на различных стадиях его развития, включая Бетти Барр, университет Хьюстона; Андрей Г. Чаховской, Калифорнийский университет, Дэвис; Роджер Кинг, университет Толедо; Ричард Куор, университет Колорадо в Колорадо-Спрингсе; Ларри Лэджерстром, Калифорнийский университет, Дэвис;



Ю-Джоу Лин, Акронский университет; Х. Дэвид Шитс, Канзиус-Колледж; Джеб Томас, университет Айовы; Брайан Вик, Политехнический институт и университет штата Вирджиния; Джей Веицен, Массачусетский университет, Лоуэлл; и Джейн Паттерсон Файф, Университет штата Огайо. Кроме того, Я хотел выразить признательность Даниэлю Сэйри и Джойсу По, все из John Wiley&Sons, которые поддерживали работу над пятым изданием.

Я надеюсь, что книга будет полезна и поможет пользователям MATLAB наслаждаться этим программным обеспечением.

Амос Гилат
Колумбия, Огайо
ноябрь, 2013
gilat.1@osu.edu



ВВЕДЕНИЕ

MATLAB – это мощный язык для технических вычислений. Название MATLAB происходит от слов MATrix LABoratory (матричная лаборатория), потому что основной элемент его данных есть матрица (массив). MATLAB может использоваться для математических вычислений, моделирования, анализа и обработки данных, визуализации и графики и разработки алгоритмов.

MATLAB широко используется в университетах и колледжах во вводных и продвинутых курсах математики, в науке и, особенно, в технике. В индустрии это программное обеспечение используется в исследованиях, разработке и проектировании. Стандартная программа MATLAB имеет инструменты (функции), которые могут использоваться для решения типичных проблем. Кроме того, MATLAB имеет дополнительные пакеты расширения (toolboxes), которые являются наборами специализированных программ, созданных для решения определенных типов задач. Например, пакеты расширения для обработки сигналов, символьных вычислений и анализа систем управления.

До недавнего времени большинство пользователей MATLAB были людьми со знаниями языков программирования, такими как ФОРТРАН и С, которые переключались на MATLAB, поскольку это программное обеспечение стало популярным. Следовательно, большая часть литературы, которая была написана о MATLAB, предполагала, что у читателя есть знания компьютерного программирования. Книги о MATLAB зачастую предлагают сложные темы или приложения, которые специализируются в конкретной области. Однако сегодня MATLAB предлагается студентам колледжа как первая (и часто единственная) компьютерная программа, которую они изучат. Для этих студентов имеется необходимость в книге, которая излагает MATLAB, не предполагая предшествующего опыта в компьютерном программировании.

Цель этой книги

Книга «MATLAB. Теория и практика» предназначена для студентов, которые используют MATLAB впервые и не имеют, или имеют немного опыта в компьютерном программировании. Она может быть использована в качестве учебника первокурсников для инженерных курсов или семинаров в которых используется MATLAB. Книга может также служить в качестве справочного материала в более продвинутых разделах науки и технических курсах, где MATLAB используется в качестве инструмента для решения задач. Она также может использоваться для

самостоятельного изучения MATLAB студентами и практикующими инженерами. Кроме того, книга может быть дополнением или вторичной книгой в курсах, где MATLAB используется, но у преподавателя нет времени, чтобы изложить его в достаточной мере.

Затронутые темы

MATLAB – это огромная программа, и поэтому невозможно изложить все это в одной книге. Эта книга фокусируется прежде всего на основах MATLAB. Мы предполагаем, что если эти основы хорошо поняты, студент будет в состоянии изучить продвинутые темы легко с использованием меню справочной информации MATLAB.

Порядок, в котором темы представлены в этой книге, был выбран тщательно и основан на многолетнем опыте преподавания MATLAB во вводном техническом курсе. Темы представлены в порядке, который позволяет студенту следовать глава за главой в этой книге. Каждая тема представлена полностью в одном месте и затем используется в следующих главах.

Первая глава описывает базовую структуру и функции MATLAB и как использовать эту программу для простых арифметических действий со скалярами как калькулятор. В конце главы представлены скрипт-файлы (файлы сценарии). Они позволяют студенту написать, сохранить и выполнять простые программы MATLAB. Следующие две главы посвящены теме массивов. Основной элемент данных MATLAB – это массив, который не требует определения его размеров. Эта концепция, которая делает MATLAB очень мощной программой, может быть немного трудна для понимания студентов, у которых есть только ограниченные знания и опыт в линейной алгебре и векторном анализе. Понятие массивов представляется постепенно и затем объясняется достаточно широко в деталях. Глава 2 описывает, как создать массивы, а глава 3 покрывает математические действия с массивами.

После этих основ в главе 4 представлены более продвинутые темы, которые связаны со скрипт-файлами и вводом и выводом данных. Затем следует изложение темы двумерного графического изображения в главе 5. Программирование с MATLAB представлено в главе 6. Оно включает управление потоком выполнения команд с условными операторами и циклами. Пользовательские функции, анонимные функции и функции от функции изложены затем в главе 7. Изложение файлов функций (пользовательские функции) преднамеренно проведено отдельно от темы скрипт-файлов. Как показывает опыт, это легче понять студентам, которые не знакомы с подобными понятиями из других компьютерных программ.

Следующие три главы затрагивают более продвинутые темы. Глава 8 описывает, как MATLAB может использоваться для вычислений с многочленами и как использовать MATLAB для подгонки кривой и интерполяции. Глава 9 покрывает применения MATLAB в численном анализе. Она включает решение нелинейных уравнений, нахождение минимума или максимума функции, численного интегрирования и решения обыкновенных дифференциальных уравнений первого порядка. Глава 10 описывает, как сделать трехмерные графики – это продолжение гла-

вы по двумерным графикам. Глава 11 излагает очень подробно, как использовать MATLAB в символьных операциях.

Схема обычной главы

В каждой главе темы представляются постепенно в порядке, который делает понятия легкими для понимания. Использование MATLAB широко демонстрируется в тексте и примерами. Некоторые из более длинных примеров в главах 1–3 названы учебными программами. Каждое использование MATLAB печатается другим шрифтом на сером фоне, с дополнительными пояснениями к тексту программы. Идея состоит в том, чтобы читатель выполнил эти демонстрационные и учебные программы для того, чтобы получить опыт в использовании MATLAB. Кроме того, каждая глава включает формальные примеры типовых задач, которые являются примерами применений MATLAB для решения задач в математике, науке и технике. Каждый пример включает постановку задачи и детальное решение. Некоторые типовые задачи представлены в середине главы. У всех глав (кроме главы 2) есть раздел в конце с несколькими типовыми задачами. Необходимо отметить, что задачи могут быть решены с MATLAB многими различными способами. Решения типовых задач записаны так, что им легко следовать. Это означает, что во многих случаях задача может быть решена значительно короче, или иногда «более хитрой» программой. Студентам рекомендуется попытаться записать их собственные решения и сравнить конечные результаты. В конце каждой главы есть ряд задач для домашней работы. Они включают общие задачи математики и науки и задачи из различных дисциплин техники.

Символьные вычисления

MATLAB – это по существу программное обеспечение для численных расчетов. Однако, могут быть выполнены и символьные математические операции, если установлен пакет расширения Symbolic Math. Пакет инструментов Symbolic Math включен в студенческую версию этого программного обеспечения MATLAB и может быть добавлен к стандартной программе.

Программное и аппаратное обеспечение

Программа MATLAB, как и большинство другого программного обеспечения, непрерывно разрабатывается и часто выпускаются новые версии. Эта книга соответствует MATLAB версии 8.2.0.701, выпуска 2013b. Нужно подчеркнуть, однако, что книга покрывает основы MATLAB, которые почти не меняются от версии до версии. Эта книга излагает использование MATLAB на компьютерах, которые используют операционную систему Windows. Когда MATLAB используется на других машинах, все по существу то же самое. Пользователь отсылается к документации MATLAB для деталей относительно использования MATLAB на других операционных системах. Предполагается, что это программное обеспечение установлено на компьютере и у пользователя есть элементарные знания о работе с компьютером.

Порядок тем в книге

Вероятно, невозможно написать учебник, где все вопросы представлены в подходящем для всех порядке. Порядок тем в этой книге такой, что сначала излагаются основы MATLAB (массивы и операции над массивом) и, как уже упоминалось выше, каждая тема разобрана полностью в одном месте, что делает книгу удобной для ссылок. Порядок тем в этом пятом выпуске тот же самый, как и в предыдущем выпуске. Программирование представлено перед пользовательскими функциями. Это позволяет использовать программирование в пользовательских функциях. Кроме того, приложения MATLAB в численном анализе следуют за главой 8, которая покрывает многочлены, подбор кривых и интерполяцию.



ГЛАВА 1.

Начало работы с MATLAB

Эта глава начинается с описания характеристик и целей различных окон в MATLAB. Более подробно представлено командное окно. Затем в данной главе показано, как использовать MATLAB для арифметических операций со скалярами способами, похожими на использование калькулятора. Представлено использование элементарных математических функций со скалярами. Показано, как определить скалярные переменные (оператор присвоения) и как использовать эти переменные в арифметических вычислениях. Последний раздел в главе знакомит со скрипт-файлами для записи, сохранения и выполнения простых программ MATLAB. В конце приведены примеры решения нескольких задач и список задач и упражнений для закрепления материала.

1.1. Запуск MATLAB, окна MATLAB

Предполагается, что программное обеспечение установлено на компьютере, и что пользователь может запустить программу. При запуске открывается рабочий стол MATLAB сконфигурированный значениями по умолчанию, рисунок 1.1. Сверху расположена лента инструментов, ниже находится панель инструментов текущего каталога и еще ниже – четыре окна. Верхняя лента инструментов имеет три вкладки: HOME, PLOTS и APPS. При выборе другой вкладки меняются значки ленты инструментов. Обычно MATLAB используется с выбранной вкладкой HOME. Соответствующие значки используются для выполнения различных команд, это показывается ниже в данной главе. Вкладка PLOTS может использоваться для создания графиков, подробнее об этом см. в главе 5 (раздел 5.12), а вкладка APPS может использоваться для открытия дополнительных приложений и панелей инструментов MATLAB.

Конфигурация по умолчанию

При конфигурации по умолчанию (рис. 1.1) основное рабочее окно MATLAB включает следующие четыре окна, которые находятся под лентой инструментов: командное окно (самое большое окно в центре), окно текущего каталога (слева) и окна рабочего пространства и истории команд (справа). Список окон MATLAB и их целей приведен в табл. 1.1. Четыре из этих окон – командное окно, окно Figure,

окно редактора и окно справки будут постоянно использоваться в данной книге и они кратко описаны на следующих страницах. Более подробные описания включены в главы, где они используются. Окно истории команд, окно текущего каталога и окно рабочего пространства описаны в разделах 1.2, 1.8.4, и 4.1, соответственно.

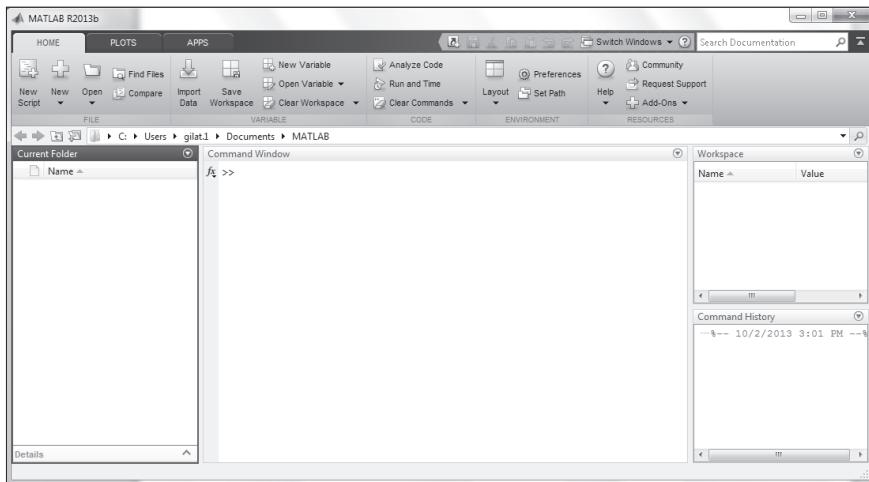


Рис. 1.1. Обычный вид рабочего стола MATLAB

Таблица 1.1. Окна MATLAB

Окно	Назначение
Командное окно (Command Window)	Главное окно, вводит переменные, выполняет команды, программы.
Окно Figure	Для вывода результатов команд графики.
Окно редактора (Editor)	Создает и отлаживает скрипты-файлы и файлы функций.
Окно справки (Help)	Обеспечивает справочную информацию.
Окно истории команд (Command History)	Строки команд, введенных командном окне.
Рабочее пространство (Workspace)	Предоставляет информацию о сохраненных переменных.
Окно текущего каталога (Current Folder)	Показывает файлы в текущей папке.

Командное окно. Это главное окно MATLAB и оно открывается при запуске MATLAB. Удобно иметь командное окно как единственное видимое окно. Это может быть сделано или закрытием всех других окон, или выбирая **Command Window Only** в меню, которое открывается, когда выбран значок **Layout** на ленте инструментов. Для закрытия окна достаточно щелкнуть по выпадающему меню в

верхнем правом углу окна и затем выбрать **Close**. Работа в командном окне описана подробно в разделе 1.2.

Окно графиков (Figure). Открывается автоматически при выполнении команд построения графиков и содержит графики, создаваемые этими командами. Пример окна графиков Figure показан на рис. 1.2. Более подробное описание этого окна дано в главе 5.

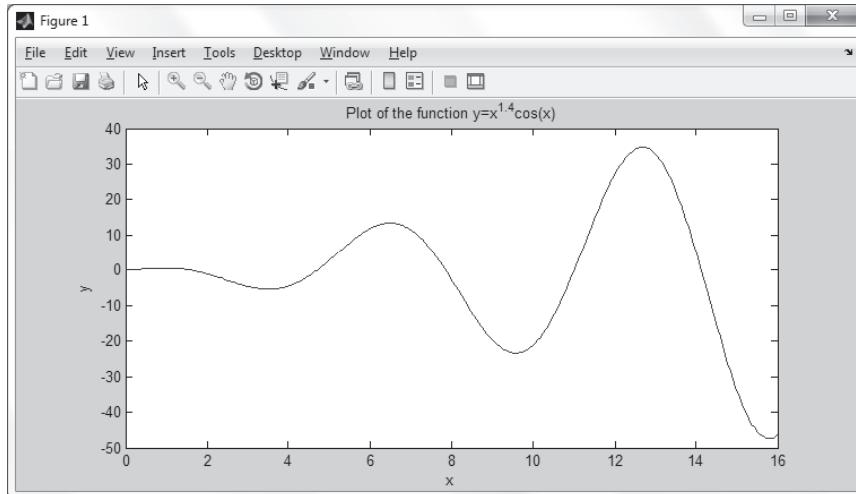


Рис. 1.2. Пример окна графиков

Окно редактора. Используется для записи и редактирования программ. Это окно открывается щелчком значку **New Script** на ленте инструментов, или щелчком по значку **New** и последующим выбором **Script** из открывшегося меню. Пример окна редактора показан на рис. 1.3. Более подробно об окне редактора написано в разделе 1.8.2, где оно используется для записи скриптов-файлов (сценариев) и в главе 7, где оно используется для написания файлов функций.

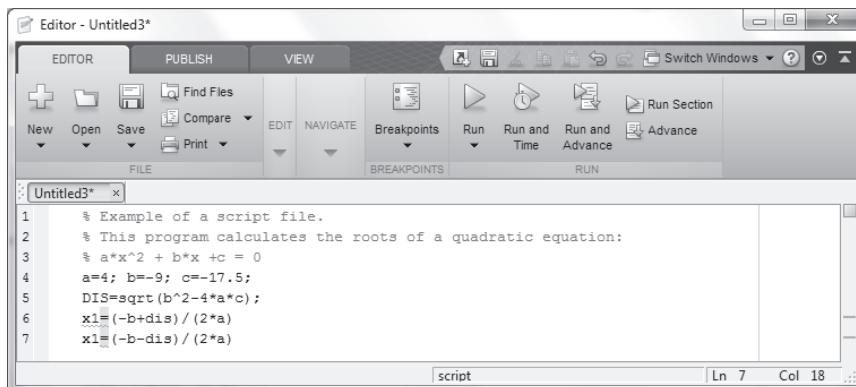


Рис. 1.3. Пример окна редактора

Окно справки (Help). Содержит справочную информацию. Это окно может быть открыто через значок **Help** на ленте инструментов командного окна или панели инструментов любого окна MATLAB. Окно справки является интерактивным и может использоваться для получения информации о любой функции MATLAB. Рис. 1.4 показывает открытое окно справки.

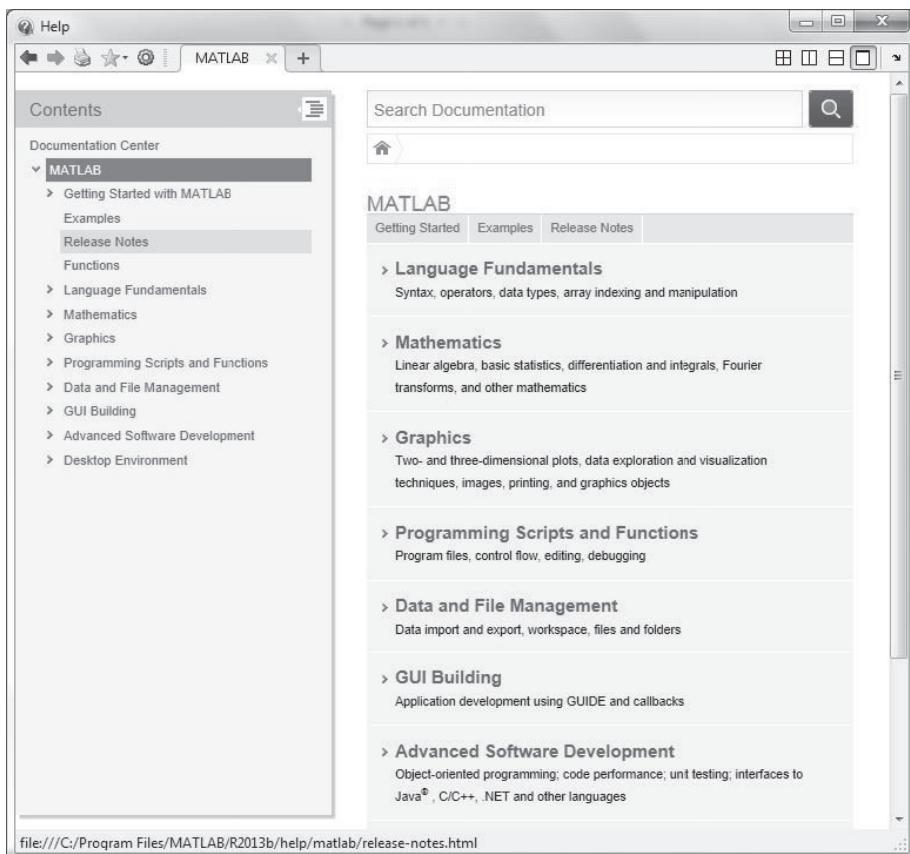


Рис. 1.4. Окно справки

Когда MATLAB запускается первый раз, экран похож на тот, что показан на рис. 1.1. Для большинства новичков, вероятно, будет удобнее закрыть все окна кроме окна команд. Закрытые окна могут быть вновь открыты, для этого их нужно выбрать при помощи значка конфигурации **Layout** на ленте инструментов. Окна, показанные на рис. 1.1, могут быть выведены на экран, щелкнув по значку конфигурации **Layout** и выбирая в открывшемся меню значение по умолчанию **Default**. Различные окна на рис. 1.1 пристыкованы к рабочему столу. Окно может быть отстыковано (станет отдельным, независимым окном) простым вытаскиванием его мышкой за пределы рабочего стола. Независимое окно может быть повторно прикреплено, щелкнув по выпадающему меню в верхней правой части окна и выбирая **Dock**.

1.2. Работа в командном окне

Командное окно – это главное окно MATLAB и оно может использоваться для выполнения команд, открытия других окон, запуска программ, написанных пользователем и управления программным обеспечением. Пример окна команд с несколькими простыми командами, которые будут объяснены позже в этой главе, показан на рис. 1.5.

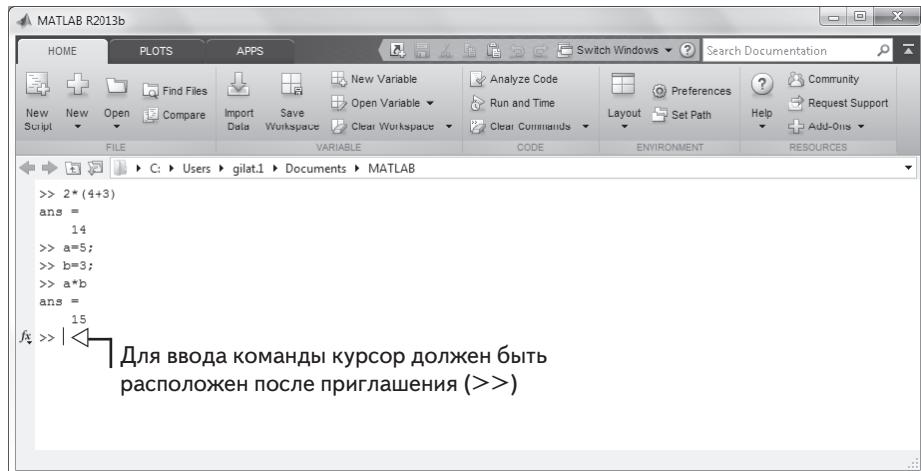


Рис. 1.5. Командное окно

Замечания о работе в командном окне:

- Для ввода команды курсор должен быть расположен после командного приглашения ($>>$).
- Команда выполняется после ввода команды нажатия клавиши **Enter**. Однако, выполняется только последняя команда. Все команды, выполненные ранее (они могут еще отображаться на экране) заново не исполняются.
- На одной строке можно ввести несколько команд, ставя запятую между командами. При нажатии клавиши **Enter**, команды выполняются в порядке слева направо.
- Невозможно вернуться к предыдущей строке, которая остается выведенной на экран в командном окне, сделайте исправление, и затем повторно выполните команду.
- Ранее введенные команды запоминаются и могут быть введены заново при помощи клавиши со стрелкой вверх (\uparrow). Когда такая команда выведена на экран в командной строке, она может быть при необходимости изменена и затем выполнена. Клавиша со стрелкой вниз (\downarrow) может использоваться для перемещения по списку ранее введенных команд вниз.
- Если команда является слишком длинной и не умещается на одной строке, она может быть продолжена на следующей строке при помо-

ши многоточия ... и нажатия клавиши **Enter** для перехода на следующую строку. Тогда продолжение команды вводится в новой строке. Команду может продолжать строка за строкой в общей сложности до 4 096 знаков.

Точка с запятой (;)

Когда команда введена в командном окне и нажата клавиша **Enter**, она выполняется. Любой результат, который создает команда, выводится на экран в командном окне. Однако, если в конце команды поставлена точка с запятой (;), результат команды не выводится на экран. Ввод точки с запятой полезен, когда результат очевиден или известен, или когда результат является очень большим. Если несколько команд введены на одной и той же строке, то результат каждой из них не будет выведен на экран, если после соответствующей команды стоит точка с запятой вместо запятой.

Ввод (%)

Если в начале строки введен символа % (*процент*), то строка определяется как **комментарий**. Это означает, что при нажатии клавиши **Enter**, строка не выполняется. Символ % и следующий за ним текст (комментарий) может также быть введена после команды (на той же самой строке). Это не влияет на выполнение команды. Обычно нет никакой необходимости в комментариях в командном окне. Однако комментарии часто используются в программе, чтобы добавить описания или пояснить программу (см. главы 4 и 6).

Команда clc

Команда **clc** (вводим **clc** и нажимаем **Enter**) очищает командное окно. После работы некоторое время в командном окне, экран этого окна может стать очень длинным. При выполнении команды **clc** на экран выводится чистое окно. Эта команда не изменяет ничего, что было сделано ранее. Например, если некоторые переменные были определены ранее (см. раздел 1.6), они все еще существуют и могут использоваться. Клавиша со стрелкой вверх может также использоваться, чтобы вспомнить команды, которые были введены ранее.

Окно истории команд

Окно истории команд перечисляет команды, которые были введены в командном окне. Оно содержит также команды из предыдущих сеансов. Команда в окне истории команд может снова использоваться в командном окне. Двойным щелчком по этой команде она повторно вводится в командном окне и выполняется. Также можно перетянуть команду в командное окно, произвести изменения если нужно и затем выполнить ее. Список в окне истории команд может быть очищен. Для этого нужно выбрать строки, которые будут удалены, затем щелкнуть правой кнопкой мыши и выбрать **Delete**. Вся история может быть удалена щелчком правой кнопкой по раскрывающемуся меню справа вверху окна и выбором **Clear Command History** от открывающимся меню.

1.3. Арифметические операции со скалярами

В этом разделе мы обсуждаем только арифметические операции со скалярами, которые являются числами. Как будет объяснено ниже в этой главе, числа могут использоваться в арифметических вычислениях непосредственно (как с калькулятором), или они могут быть присвоены переменным, которые могут впоследствии использоваться в вычислениях. Символы арифметических операций:

Операция	Символ	Пример
Сложение	+	5+3
Вычитание	-	5-3
Умножение	*	5*3
Правое деление	/	5/3
Левое деление	\	5\3 = 3/5
Возведение в степень	^	5^3 (означает $5^3 = 125$)

Отметим, что все символы кроме левого деления – те же самые как в большинстве калькуляторов. Для скаляров результат левого деление – это величина обратная к правому делению. Однако левое деление главным образом используется для операций с массивами, которые обсуждаются в Главе 3.

1.3.1. Приоритет операций

MATLAB выполняет вычисления согласно порядку очередности (*приоритету*), приведенному ниже. Этот порядок такой же, что и для большинства калькуляторов.

Приоритет	Математическая операция
Первый	Круглые скобки. Для вложенных круглых скобок, сначала выполняются самые внутренние.
Второй	Возведение в степень.
Третий	Умножение, деление (равный приоритет).
Четвертый	Сложение и вычитание.

В выражении, у которого есть несколько операций, операции более высокого приоритета выполняются перед операциями более низкого приоритета. Если у двух или больше операций один и тот же приоритет, то выражение выполняется слева направо. Для изменения порядка вычислений можно использовать круглые скобки.

1.3.2. Использование MATLAB как калькулятор

Самый простой способ использовать MATLAB – это как калькулятор. Достаточно вводить в командном окне математические выражения и нажимать клавишу **Enter**. MATLAB вычисляет выражение и выводит на экран `ans` = за которым следует числовой результат выражения в следующей строке. Это демонстрируется в учебной программе 1.1.

Учебная программа 1.1. Использование MATLAB как калькулятора

<code>>> 7+8/2</code>	Ведите и нажмите Enter .
<code>ans =</code>	$8/2$ выполняется первым.
<code>11</code>	
<code>>> (7+8)/2</code>	Ведите и нажмите Enter .
<code>ans =</code>	Сначала выполняется $7+8$.
<code>7.5000</code>	
<code>>> 4+5/3+2</code>	Сначала выполняется $5/3$.
<code>ans =</code>	
<code>7.6667</code>	
<code>>> 5^3/2</code>	Сначала выполняется 5^3 , затем $/2$.
<code>ans =</code>	
<code>62.5000</code>	
<code>>> 27^(1/3)+32^0.2</code>	Сначала выполняется $1/3$, затем $27^{(1/3)}$ и $32^{0.2}$
<code>ans =</code>	
<code>5</code>	и $+$ выполняется последним.
<code>>> 27^1/3+32^0.2</code>	Сначала выполняются 27^1 и $32^0.2$, затем $/3$
<code>ans =</code>	
<code>11</code>	и $+$ выполняется последним.
<code>>> 0.7854-(0.7854)^3/(1*2*3)+0.785^5/(1*2*3*4*5)...</code>	←
<code>- (0.785)^7/(1*2*3*4*5*6*7)</code>	Ведите три точки ... (и нажмите Enter)
<code>ans =</code>	для продолжения выражения на следующую строку.
<code>0.7071</code>	
<code>>></code>	Последнее выражение – это первые четыре члена ряда Тейлора для $\sin(\pi/4)$.

1.4. Форматы вывода

Пользователь может управлять *форматами вывода* MATLAB результатов на экран. В учебной программе 1.1 формат результата – с фиксированной точкой с четырьмя десятичными знаками (называемый коротким, *short*), он является форматом по умолчанию для числовых значений. Формат может быть изменен с командой *format*. После введения команды *format*, все последующие результаты выводятся на экран в указанном формате. Некоторые из доступных форматов перечислены и описаны в табл. 1.2.

У MATLAB есть несколько других форматов для вывода чисел на. Детали этих форматов могут быть получены при помощи команды `help format`. Формат, в котором выводятся на экран числа, не влияет на то, как MATLAB вычисляет и сохраняет числа.

Таблица 1.2. Форматы вывода

Команда	Описание	Пример
<code>format short</code>	Фиксированная точка с 4 десятичными цифрами для: $0.001 \leq \text{число} \leq 1000$ Иначе отображается в формате <code>format short e</code> .	<code>>> 290/7</code> <code>ans =</code> <code>41.4286</code>
<code>format long</code>	Фиксированная точка с 15 десятичными цифрами для: $0.001 \leq \text{число} \leq 100$ Иначе отображается в формате <code>format long e</code> .	<code>>> 290/7</code> <code>ans =</code> <code>41.428571428571431</code>
<code>format short e</code>	Экспоненциальное представление с 4 десятичными цифрами.	<code>>> 290/7</code> <code>ans =</code> <code>4.1429e+001</code>
<code>format long e</code>	Экспоненциальное представление с 15 десятичными цифрами.	<code>>> 290/7</code> <code>ans =</code> <code>4.142857142857143e+001</code>
<code>format short g</code>	Лучшее 5-значное с фиксированной или плавающей запятой.	<code>>> 290/7</code> <code>ans =</code> <code>41.429</code>
<code>format long g</code>	Лучшее 15-значное с фиксированной или плавающей запятой.	<code>>> 290/7</code> <code>ans =</code> <code>41.4285714285714</code>
<code>format bank</code>	Два десятичных знака после запятой.	<code>>> 290/7</code> <code>ans =</code> <code>41.43</code>
<code>format compact</code>	Удаляет пустые строки для вывода на экран большего числа строк с информацией.	
<code>format loose</code>	Добавляет пустые линии (противоположность <code>compact</code>).	

1.5. Встроенные элементарные математические функции

В дополнение к основным арифметическим операциям выражения в MATLAB могут включать функции. У MATLAB есть очень большая библиотека *встроенных функций*. Каждая функция имеет имя и аргумент в круглых скобках. Например, функция `sqrt(x)`, которая вычисляет квадратный корень числа. Ее имя – `sqrt`, и аргумент – `x`. При использовании функции аргумент может быть числом,

переменной, которой было присвоено числовое значение (пояснения см. в разделе 1.6), или вычислимое выражение, которое может быть составлено из чисел и/или переменных. Функции могут также быть включены в аргументы, так же как в выражения. Учебная программа 1.2 показывает примеры использования функции `sqrt(x)`, когда MATLAB используется в качестве калькулятора.

Учебная программа 1.2. Использование встроенной функции `sqrt`.

```
>> sqrt(64)                                Аргументом является число.
ans =
    8
>> sqrt(50+14*3)                            Аргументом является выражение.
ans =
    9.5917
>> sqrt(54+9*sqrt(100))                   Аргумент содержит функцию.
ans =
    12
>> (15+600/4)/sqrt(121)                   Функция включена в выражение.
ans =
    15
>>
```

Некоторые обычно используемые элементарные математические встроенные функции MATLAB приведены в табл. 1.3 – 1.5. Полный список функций, организованных по категориям, может быть найден в окне справки.

Таблица 1.3. Элементарные математические функции

Функция	Описание	Пример
<code>sqrt(x)</code>	Квадратный корень.	<pre>>> sqrt(81) ans = 9</pre>
<code>nthroot(x, n)</code>	Действительный корень степени n из вещественного числа x . (Если x отрицательно, то n должно быть нечетным целым числом.)	<pre>>> nthroot(80, 5) ans = 2.4022</pre>
<code>exp(x)</code>	Экспонента (e^x).	<pre>>> exp(5) ans = 148.4132</pre>
<code>abs(x)</code>	Абсолютное значение.	<pre>>> abs(-24) ans = 24</pre>
<code>log(x)</code>	Натуральный логарифм. Основание логарифма e (\ln).	<pre>>> log(1000) ans = 6.9078</pre>

Функция	Описание	Пример
<code>log10(x)</code>	Логарифм по основанию 10.	<code>>> log10(1000)</code> <code>ans =</code> <code>3.0000</code>
<code>factorial(x)</code>	Факториал $x!$ (x должно быть положительным целым.)	<code>>> factorial(5)</code> <code>ans =</code> <code>120</code>

Таблица 1.4. Тригонометрические функции

Функция	Описание	Пример
<code>sin(x)</code>	Синус угла x (x в радианах).	<code>>> sin(pi/6)</code>
<code>sind(x)</code>	Синус угла x (x в градусах).	<code>ans =</code> <code>0.5000</code>
<code>cos(x)</code>	Косинус угла x (x в радианах).	<code>>> cosd(30)</code>
<code>cosd(x)</code>	Косинус угла x (x в градусах).	<code>ans =</code> <code>0.8660</code>
<code>tan(x)</code>	Тангенс угла x (x в радианах).	<code>>> tan(pi/6)</code>
<code>tand(x)</code>	Тангенс угла x (x в градусах).	<code>ans =</code> <code>0.5774</code>
<code>cot(x)</code>	Котангенс угла x (x в радианах).	<code>>> abs(-24)</code>
<code>cotd(x)</code>	Котангенс угла x (x в градусах).	<code>ans =</code> <code>24</code>

Обратные тригонометрические функции для угла в радианах – это `asin(x)`, `acos(x)`, `atan(x)`, `acot(x)`, а для угла в градусах – `asind(x)`, `acosd(x)`, `atand(x)`, `acotd(x)`. Гиперболические функции – `sinh(x)`, `cosh(x)`, `tanh(x)` и `coth(x)`. В табл. 1.4 используется символ `pi`, который равен π (см. раздел 1.6.3).

Таблица 1.5. Функции округления

Функция	Описание	Пример
<code>round(x)</code>	Округляет до самого близкого целого числа.	<code>>> round(17/5)</code> <code>ans =</code> <code>3</code>
<code>fix(x)</code>	Округляет ближе к нулю. Усекает знаки после запятой.	<code>>> fix(13/5)</code> <code>ans =</code> <code>2</code>
<code>ceil(x)</code>	Ближайшее целое справа, т. е. ближайшее большее или равное данному числу.	<code>>> ceil(11/5)</code> <code>ans =</code> <code>3</code>
<code>floor(x)</code>	Целая часть числа. Наибольшее целое не превосходящее данное. Ближайшее целое слева.	<code>>> floor(-9/4)</code> <code>ans =</code> <code>-3</code>

Функция	Описание	Пример
<code>rem(x, y)</code>	Возвращает остаток после деления x на y .	<code>>> rem(13, 5)</code> <code>ans =</code> <code>3</code>
<code>sign(x)</code>	Функция сигнум (знак). Возвращает 1, если $x > 0$, -1 если $x < 0$ и 0, если $x = 0$	<code>>> sign(5)</code> <code>ans =</code> <code>1</code>

1.6. Определение скалярных переменных

Переменная представлена именем, созданным из буквы или комбинации нескольких букв (и цифр), которому присвоено числовое значение. Как только переменной присвоено числовое значение, она может использоваться в математических выражениях, в функциях и в любых операторах MATLAB и командах. Переменная – это фактически имя расположения памяти. Когда определяется новая переменная, MATLAB выделяет соответствующее место в памяти, где хранится присвоенное значение. При использовании переменной используются хранящиеся там данные. Если переменной присвоено новое значение, меняется содержание этого места расположения памяти. (В главе 1 мы рассматриваем только такие переменные, которым присваиваются числовые скалярные значения. Присвоение и адресация переменных, которые являются массивами, обсуждаются главе 2.)

1.6.1. Оператор присвоения

В MATLAB знак = называется оператором присвоения. Оператор присвоения присваивает значение переменной.

Имя_переменной = численное значение, или вычислимое выражение

- Левая часть оператора присвоения может включать только одно имя переменной. Правая часть может быть числом, или вычислимым выражением, которое может включать числа и/или переменные, которым ранее были присвоены числовые значения. Когда нажимается клавиша **Enter**, числовое значение правой части присваивается переменной и MATLAB выводит на экран переменную и ее присвоенное значение в следующих двух строках. Следующий текст показывает как работает оператор присвоения.

<pre>>> x=15 x = 15 >> x=3*x-12 x = 33 >></pre>	<p>Число 15 присвоено переменной x. MATLAB выводит на экран имя переменной и его присвоенное значение. Переменой x присвоено новое значение. Новое значение – это 3-кратное предыдущее значение x минус 12.</p>
---	--

Последний оператор ($x = 3x - 12$) иллюстрирует различие между оператором присвоения и знаком «равно». Если бы в этом операторе знак «==» означал бы «равно» (уравнение), то значение x было бы равно 6 (как решение уравнения на x).

Ниже показывается использование ранее определенных переменных для определения новых переменных.

```
>> a=12          Присвоение 12 переменной а.  
a =  
    12  
>> B=4          Присвоение 12 переменной В.  
B =  
    4  
>> C=(a-B)+40-a/B*10  Присвоение значения выражения  
C =                  правой части переменной С.  
    18
```

- Если в конце команды введена точка с запятой, то при нажатии клавиши **Enter** MATLAB не выводит на экран переменную со своим присвоенным значением (но переменная все еще существует и сохранена в памяти).
- Если переменная уже существует, то вводя имя переменной и нажимая клавишу **Enter** мы выводим на экран переменную и ее значение на следующих двух строках.

В качестве примера показываем приведенные выше команды, но уже с использованием точек с запятой.

```
>> a=12;  
>> B=4;  
>> C=(a-B)+40-a/B*10;  
>> C          ↘  
C =  
    18
```

Переменные а, В и С определены,
но не выведены на экран, так как в конце
каждого оператора стоит точка с запятой.
Значение переменной С выведено на экран,
при помощи ввода имени переменной.

- Несколько присвоений могут быть введены на одной и той же строке. Эти присвоения должны быть разделены запятой (после запятой может быть вставлен пробелы). При нажатии клавиши **Enter** присвоения выполняются слева направо, и переменные, и их присвоения выводятся на экран. Переменная не выводится на экран, если вместо запятой введена точка с запятой. Например, все вышеупомянутые присвоения переменных а, в и с могут быть сделаны на одной и той же строке.

```
>> a=12, B=4; C=(a-B)+40-a/B*10  
a =  
    12  
C =  
    18
```

↑
Переменная В не выводится на экран,
поскольку после нее стоит точка с запятой

- Переменной, которая уже существует, может быть повторно присвоено новое значение. Например:

```
>> ABB=72;           Значение 72 присвоено переменной ABB.  
>> ABB=9;           Новое значение 9 присвоено переменной ABB.  
>> ABB             ← Текущее значение переменной выводится  
ABB =               на экран, когда вводится имя переменной  
    9                и нажимается клавиша Enter.  
>>
```

- После того, как переменная определена, она может использоваться в качестве аргумента в функциях. Например:

```
>> x=0.75;  
>> E=sin(x)^2+cos(x)^2  
E =  
    1  
>>
```

1.6.2. Правила для имен переменных

Переменную можно назвать в соответствии со следующими правилами:

- Имя переменной должно начаться с буквы.
- Может иметь длину до 63 символов.
- Может содержать буквы, цифры и символ подчеркивания.
- Не может содержать символы пунктуации (например, апострофы, запятые, точку с запятой) и буквы русского алфавита.
- MATLAB чувствителен к регистру: он различает прописные и строчные буквы. Например, AA, Aa, aA, и aa – это имена четырех различных переменных.
- Пробелы между символами недопустимы (используйте подчеркивание, если нужен пробел).
- Избегайте использования имен встроенных функций для переменных (то есть, избегайте использования cos, sin, exp, sqrt и т. п.). После того, как Вы использовали имя функции для имени переменной, эта функция уже не может быть вызвана.

1.6.3. Предопределенные переменные и зарезервированные слова

Есть 20 слов, называемых *ключевыми словами* (keywords), которые зарезервированы MATLAB в различных целях и не могут использоваться в качестве имен переменных. Эти слова:

```
break case catch classdef continue else elseif
end for function global if otherwise parfor
persistent return spmd switch try while
```

При вводе таких слов они отображаются синим. Если пользователь пытается использовать ключевое слово в качестве имени переменной, то на экран выводится сообщение об ошибке. (Ключевые слова могут быть выведены на экран командой `iskeyword`.)

Многие часто используемые переменные уже определены при запуске MATLAB. Некоторые из *предопределенных переменных*:

ans переменная, которая принимает значение последнего выражения, которое не было присвоено определенной переменной (см. Учебную программу 1.1). Если пользователь не присваивает значение выражения какой-нибудь переменной, MATLAB автоматически хранит результат в `ans`.

pi это число π .

eps наименьшая разность между двумя числами. Равна $2^{(-52)}$, что приближенно равно $2.2204e-016$.

inf используется для обозначения бесконечности.

i мнимая единица, корень из -1 , считается равным $0 + 1.0000i$.

j то же самое, что и **i**.

NaN нечисловое выражение, неопределенность. Используется, когда MATLAB не может определить числовое значение. Пример: $0/0$.

Предопределенные переменные могут быть переопределены, чтобы иметь любое другое значение. Переменные `pi`, `eps`, и `inf` обычно не переопределяются, так как они часто используются во многих приложениях. Другие предопределенные переменные, такие как `i` и `j`, переопределяются часто (обычно в циклах), когда комплексные числа не используются в приложении.

1.7. Полезные команды для управления переменными

Следующие команды могут использоваться для удаления переменных или для получения информации о созданных переменных. Когда эти команды введены в командном окне и нажата клавиша **Enter**, они или предоставляют информацию, или выполняют задачу как указано ниже.

Команда	Результат
<code>clear</code>	Удаляет все переменные из памяти.
<code>clear x y z</code>	Удаляет только переменные <code>x</code> , <code>y</code> и <code>z</code> из памяти.
<code>who</code>	Выводит на экран список переменных имеющихся в настоящий момент в памяти.
<code>whos</code>	Выводит на экран список переменных имеющихся в настоящий момент в памяти и их размерах вместе с информацией об их байтах и классе (см. раздел 4.1).

1.8. Файлы сценария

До сих пор все команды вводились в командном окне и выполнялись при нажатии клавиши **Enter**. Хотя каждая команда MATLAB может быть выполнена таким образом, но это неудобно, а иногда может быть и невозможно для выполнения ряда команд – особенно, если они связаны друг с другом (программа). Команды в командном окне не могут быть сохранены и выполнены вместе снова. Кроме того, командное окно не является интерактивным. Это означает, что каждый раз при нажатии клавиши **Enter** выполняется только последняя команда, а все что выполнено прежде неизменно. Если в ранее выполненной команде необходимы изменения или исправления, а результат этой команды используется в командах, которые за ней следуют, то все команды должны быть введены и выполнены заново. Другой (лучший) способ выполнить серию команд в MATLAB заключается в том, чтобы сначала создать файл со списком команд (программу), сохранить его и затем запустить (исполнить) этот файл. При исполнении файла, команды, которые он содержит, выполняются в том порядке, как они записаны. Если нужно, команды в файле могут быть исправлены или изменены и файл может быть сохранен и выполнен снова. Файлы, которые используются с этой целью, называются *файлами сценариев*, или *скрипт-файлами*.

Важное замечание: Этот раздел покрывает только минимум, необходимый для выполнения простых программ. Это позволит студенту использовать скрипт-файлы, практикуясь на материале, который представлен в этом и следующих двух главах (вместо того, чтобы многократно вводить команды в командном окне). Файлы сценариев рассматриваются снова в главе 4, где имеется много дополнительных тем, важных для понимания MATLAB и написания программ в виде скрипт-файла.

1.8.1. Замечания о файлах сценариев

- Файл сценария – это последовательность команд MATLAB, он называется также программой.
- Когда работает (выполняется) файл сценария, MATLAB выполняет команды в том порядке, в котором они записаны, как если бы они были введены в командном окне.
- Если у файла сценария есть команда без точки с запятой в конце, которая генерирует результат (например, присвоение значения переменной), ее результат будет выведен на экран в командном окне.
- Использование скриптов удобно потому, что его можно редактировать (корректировать или делать другие изменения) и выполнять много раз.
- Файлы сценарии могут быть написаны и отредактированы в любом текстовом редакторе и затем вставлены в редактор MATLAB.
- Файлы сценарии также называют M-файлами, потому что при их сохранении используется расширение `.m`.

1.8.2. Создание и сохранение файлов сценариев

В MATLAB файлы сценарии создаются и редактируются в окне *редактора/отладчика*. Это окно открывается из командного окна, щелкнув по значку **New Script** в ленте инструментов, или щелкнув по **New** в ленте инструментов и затем выбирая **Script** из открывшегося меню. Открытое окно редактора/отладчика показано на рис. 1.6.

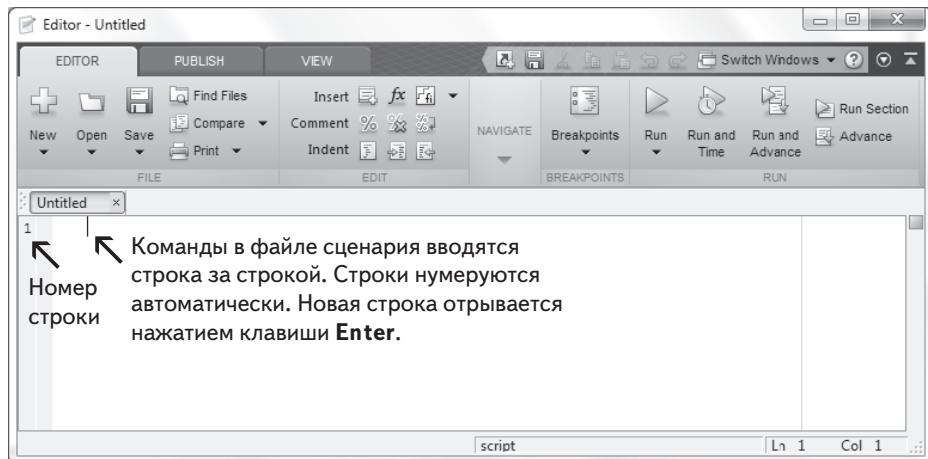


Рис. 1.6. Окно редактора/отладчика

У окна редактора/отладчика сверху есть лента инструментов и три вкладки EDITOR, PUBLISH и VIEW. Щелчок по вкладкам изменяет значки в ленте инструментов. Обычно MATLAB использует EDITOR в качестве выбранной вкладки при открытии окна редактора. Соответствующие значки используются для выполнения различных команд, как будет объяснено ниже. После открытия окна команды файла сценария вводятся строка за строкой. MATLAB автоматически нумерует новую строку каждый раз, когда нажимается клавиша **Enter**. Команды могут быть также введены в любом текстовом редакторе или программе текстового процессора и затем скопированы и вставлены в окно редактора/отладчика. Пример короткой программы введенной в окно редактора/отладчика, показан на рис. 1.7. Несколько первых строк в файле сценария – это обычно комментарии (они не выполняются, так как первый символ на строке – %), которые описывают программу, записанную в файле сценария.

Перед выполнением файла сценария он должен быть сохранен. Это можно сделать, щелкнув по **Save** в ленте инструментов и выбрав **Save As...** из открывшегося меню. При сохранении MATLAB прибавляет расширение **.m** к имени. Правила для выбора имени файла такие же, что и правила именования переменных (должен начинаться с буквы, может включать цифры и подчеркивания, никаких пробелов, кириллицы и не более 63 символов). Имена определяемых пользователем переменных, предопределенных переменных и команд MATLAB или функций не должны использоваться в качестве имен файлов сценария.

```

Editor - C:\MATLAB Book 5th Edition\Chapter 1\Chap1_Examp_1.m
FILE EDIT NAVIGATE BREAKPOINTS RUN
New Open Save Print Find Files Comment Insert Co To Breakpoints Run Run and Time Run and Advance
FILE EDIT NAVIGATE BREAKPOINTS RUN
Chap1_Examp_1.m x
1 % Example of a script file.
2 % This program calculates the roots of a quadratic equation:
3 % a*x^2 + b*x + c = 0
4
5 - a=4; b=-9; c=-17.5; ← Задание трех
6 - DIS = sqrt(b^2-4*a*c); ← переменных
7 - x1 = ( b+DIS )/( 2*a );
8 - x2 = ( -b-DIS )/( 2*a ) ← Вычисление
                                            двух корней
Ln 8 Col 20

```

Рис. 1.7. Введенная программа в окне редактора/отладчика

1.8.3. Выполнение файла сценария

Файл сценария может быть выполнен либо непосредственно из окна редактора, щелкнув по значку **Run** (см. рис. 1.7), или вводя имя файла в командном окне и затем нажимая клавишу **Enter**. При этом MATLAB должен знать, где сохранен этот файл. Файл будет исполнен, если папка, где находится файл, будет текущей папкой MATLAB или если эта папка перечислена в путях поиска, как будет объяснено ниже.

1.8.4. Текущий каталог

Текущий каталог (папка) отображается в поле текущего каталога «Current Folder» на рабочем столе MATLAB, как показано на рис. 1.8. Если предпринята попытка выполнить файл сценария не из текущего каталога щелкнув по значку **Run** (в окне редактора), то открывается диалоговое окно, предлагающее ряд вариантов, как показано в рис. 1.9. Пользователь может изменить текущую папку на ту папку, где находится файл сценария, или добавить ее к путям поиска. Если в сеансе используются две или более различных текущих папок, то можно переключиться от одной к другой в поле **Current Folder** командного окна. Текущая папка также может быть изменена в окне текущего каталога, которое можно открыть на рабочем столе MATLAB, как показано на рис. 1.10. Текущая папка может быть изменена, выбирая диск и папку где хранится файл.

Простой альтернативный способ изменения текущей папки заключается в использовании команды **cd** в командном окне. Чтобы выбрать текущую папку на другом диске необходимо ввести **cd**, пробел, и затем имя каталога, сопровождающего двоеточием : и нажать клавишу **Enter**. Например, чтобы выбрать текущую папку на диске E (например, флеш-карта) вводят **cd E:**. Если файл сценария сохранен в папке на диске, то должен быть определен путь к той папке. Это делается

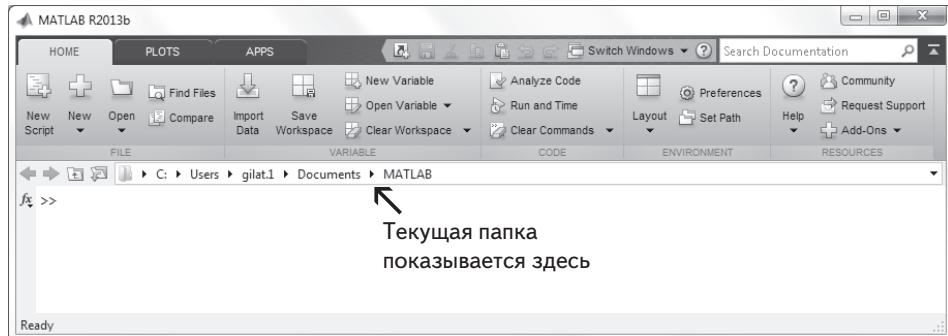


Рис. 1.8. Поле текущего каталога в командном окне

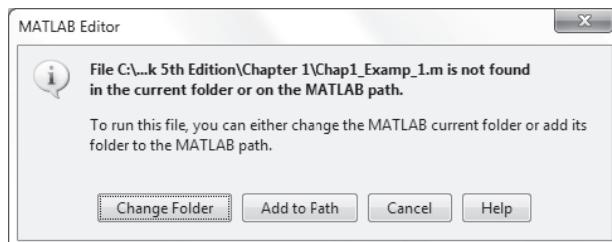


Рис. 1.9. Изменение текущего каталога

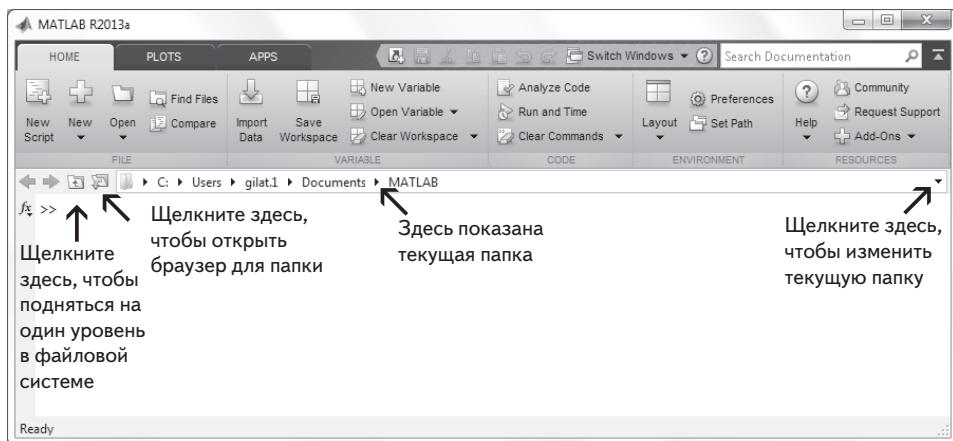


Рис. 1.10. Окно текущего каталога

добавлением строки пути в команде `cd`. Например, `cd E:\Chapter 1` или `cd('E:\Chapter 1')` устанавливает путь к каталогу Chapter 1 на диске E. Следующий пример показывает выбор текущей папки на диске E. Затем выполняется файл сценария Chap1_Examp1.m, который был сохранен на диске E, вводя имя файла и нажимая клавишу **Enter**.

```
>> cd('E:\Chapter 1')
>> Chap1_Examp1      ← Выбор текущего каталога на диске Е.
x1 =                               ← Выполнение файла сценария вводом
    3.5000                         имени файла и нажатием клавиши Enter.
x2 =                               ← Результат полученный файлом сценария
    -1.2500                        (корни x1 и x2) выводятся на экран в командном окне.
```

1.9. Примеры применений MATLAB

Пример задачи 1.1. Тригонометрические тождества

Дано тригонометрическое тождество

$$\cos^2 \frac{x}{2} = \frac{\tan x + \sin x}{2 \tan x}.$$

Проверить его корректность, вычисляя каждую часть равенства, подстановкой конкретных значений.

Решение

Задача решается следующими командами в командном окне.

```
>> x=pi/5;                      Задание x.
>> LHS=cos(x/2)^2              Вычисление левой части.
LHS =
    0.9045
>> RHS=(tan(x)+sin(x))/(2*tan(x))   Вычисление правой части.
RHS =
    0.9045
```

Пример задачи 1.2. Геометрия и тригонометрия

Четыре круга размещены так, как показано на рисунке. В каждой общей точке они касаются друг друга. Определить расстояние между центрами C_2 и C_4 . Радиусы кругов:

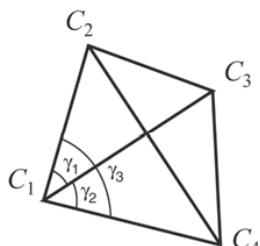
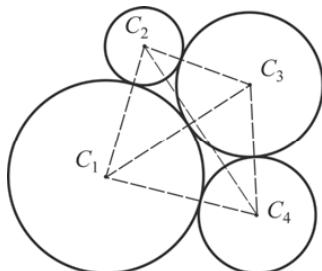
$$R_1 = 16 \text{ мм}, R_2 = 6.5 \text{ мм}, R_3 = 12 \text{ мм} \text{ и } R_4 = 9.5 \text{ мм}.$$

Решение

Линии, которые соединяют центры кругов образуют четырехугольника. В двух из треугольников, $\Delta C_1C_2C_3$ и $\Delta C_1C_3C_4$, длины всех сторон известны. Используем эту информацию для вычисления углов γ_1 и γ_2 в этих треугольниках по теореме косинусов.

Например, γ_1 вычисляется из равенства:

$$(C_2C_3)^2 = (C_1C_2)^2 + (C_1C_3)^2 - 2(C_1C_2)(C_1C_3)\cos(\gamma_1)$$



Далее, длина стороны C_2C_4 вычисляется из треугольника $\Delta C_1C_2C_4$. Это можно сделать снова по теореме косинусов (длины, C_1C_2 и C_1C_4 известны, а угол γ_3 является суммой углов γ_1 и γ_2).

Задача решается созданием следующей программы в виде файла сценария:

```
% Решение Задачи 1.2
R1=16; R2=6.5; R3=12; R4=9.5;
C1C2=R1+R2; C1C3=R1+R3; C1C4=R1+R4;
C2C3=R2+R3; C3C4=R3+R4;
Gama1=acos( (C1C2^2+C1C3^2-C2C3^2) / (2*C1C2*C1C3) );
Gama2=acos( (C1C3^2+C1C4^2-C3C4^2) / (2*C1C3*C1C4) );
Gama3=Gama1+Gama2;

C2C4=sqrt(C1C2^2+C1C4^2-2*C1C2*C1C4*cos(Gama3))
% Вычисление длины стороны C2C4.
```

Задание радиусов
Вычисление длин сторон

Вычисление γ_1, γ_2 и γ_3

При выполнении файл сценария в командном окне выводится следующее (значение переменной $C2C4$):

```
C2C4 =
33.5051
```

Пример задачи 1.3. Теплопередача

Объект с некоторой начальной температурой T_0 помещенный в момент времени $t = 0$ внутрь камеры с постоянной температурой T_s испытает изменение температуры согласно уравнению

$$T = T_s + (T_0 - T_s) e^{-kt},$$

где T – это температура объекта в момент времени t и k – некоторая постоянная. Предположим, что содовая при температуре 120 °F (после того как она была оставлена в автомобиле) помещена в холодильник, где температура 38 °F. Требуется определить с точностью до градуса температуру содовой после трех часов пребывания в холодильнике. Примите значение $k = 0.45$. Сначала определите все переменные и затем вычислите температуру, используя одну команду MATLAB.

Решение

Задача решается следующими командами в командном окне.

```
>> Ts=38; T0=120; k=0.45; t=3;
>> T=round(Ts+(T0-Ts)*exp(-k*t))
```

Округление до ближайшего целого.

Пример задачи 1.4. Начисление процентов

Баланс B сберегательного счета после t лет после вклада в размере P по годовой процентной ставке r и начислении процентов n раз в год находится по формуле:

$$B = P \left(1 + \frac{r}{n}\right)^{nt} \quad (1)$$

Если проценты начисляются один раз в год, то баланс находится по формуле:

$$B = P(1+r)^t \quad (2)$$

Предположим, что 5000 \$ инвестируются на 17 лет на один счет, где проценты начисляются ежегодно. Пусть дополнительно 5000 \$ инвестируются на другой счет, где проценты начисляются ежемесячно. В обеих счетов процентная ставка составляет 8.5%. Используя MATLAB определить, сколько времени (в годах и месяцах) потребуется, чтобы баланс второго счета был равен балансу первого счета через 17 лет.

Решение

Выполняем следующие действия:

- Вычисляем B для суммы 5000 \$, которые вложены на счет с ежегодным начислением процентов по формуле (2).
- Вычисляем t для суммы B , вычисленной в пункте (a), из уравнения (1) с ежемесячным начислением процентов.
- Определяем число лет и месяцев, которые соответствуют t .

Задача решается написанием следующей программы в виде скрипта-файла:

```
% Решение задачи примера 1.4
P=5000; r=0.085; ta=17; n=12;
B=P*(1+r)^ta
t=log(B/P)/(n*log(1+r/n))
years=fix(t)
months=ceil((t-years)*12)
```

Шаг (a): Вычисление B из ур. (2).
 Шаг (b): Решение ур. (1) и вычисление t .
 Шаг (c): Определение числа лет.
 Определение числа месяцев.

При выполнении этого файла сценария на экран в командном окне выводятся следующее (значения переменных B , t , $years$ и $months$):

```
>> format short g
B =
20011
t =
16.374
years =
16
months =
16.374
```

Значения переменных B , t , $years$, и $months$ выведены на экран (так как точка с запятой не была введена в конце ни одной из команд)

1.10. Задачи

Следующие задачи могут быть решены при помощи команд в командном окне, или написанием программы в скрипт-файле и последующим выполнением этого файла.

1. Вычислить:

$$(a) \frac{22+5.1^2}{50-6.3^2}$$

$$(b) \frac{44}{7} + \frac{8^2}{5} - \frac{99}{3.9^2}$$

2. Вычислить:

$$(a) \frac{\sqrt{41^2 - 5.2^2}}{e^5 - 100.53}$$

$$(b) \sqrt[3]{132} + \frac{\ln(500)}{8}$$

3. Вычислить:

$$(a) \frac{14.8^3 - 6.3^2}{(\sqrt{13} + 5)^2}$$

$$(b) 45\left(\frac{288}{9.3} - 4.6^2\right) - 1065e^{-1.5}$$

4. Вычислить:

$$(a) \frac{24.5 + 64/3.5^2 + 8/3 \cdot 12.5^3}{\sqrt{76.4} - 28/15}$$

$$(b) (5.9^2 - 2.4^2)/3 + \left(\frac{\log_{10} 12890}{e^{0.3}}\right)^2$$

5. Вычислить:

$$(a) \cos\left(\frac{7\pi}{9}\right) + \tan\left(\frac{7\pi}{15}\right) \sin(15^\circ)$$

$$(b) \sin^2(80^\circ) - \frac{(\cos 14^\circ \sin 80^\circ)^2}{\sqrt[3]{0.18}}$$

6. Присвоить переменной x значение $x = 6.7$ и вычислить:

$$(a) 0.01x^5 - 1.4x^3 + 80x + 16.7$$

$$(b) \sqrt{x^3 + e^x - 51/x}$$

7. Присвоить переменной t значение $t = 3.9$ и вычислить:

$$(a) 56t - 9.81\frac{t^2}{2}$$

$$(b) 14e^{-0.1t} \sin(2\pi t)$$

8. Присвоить переменным x и y значения $x = 5.1$ и $y = 4.2$ и вычислить:

$$(a) \frac{3}{4}xy - \frac{7x}{y^2} + \sqrt{xy}$$

$$(b) (xy)^2 - \frac{x+y}{(x-y)^2} + \sqrt{\frac{x+y}{2x-y}}$$

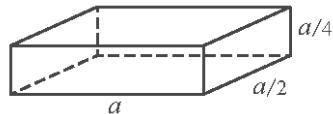
9. Присвоить переменным a, b, c и d значения: $a = 12, b = 5.6, c = \frac{3a}{b^2}$ и $d = \frac{(a-b)^2}{c}$ и вычислить:

$$(a) \frac{a}{b} + \frac{d-c}{d+c} - (d-b)^2$$

$$(b) e^{\frac{d-c}{a-2b}} + \ln\left(\left|c-d+\frac{b}{a}\right|\right)$$

10. Сфера имеет радиус 24 см. Стороны прямоугольной призмы равны a , $a/2$ и $a/4$.

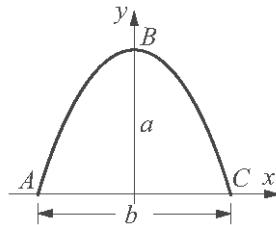
- (a) Найти призму того же объема, что и сфера.
 (b) Найти призму, у которой площадь поверхности равна площади сферы.



11. Длина дуги сегмента ABC эллипса с полуосами a и b приближенно задается формулой:

$$L_{ABC} = \frac{1}{2} \sqrt{b^2 + 16a^2} + \frac{b^2}{8a} \ln \left(\frac{4a + \sqrt{b^2 + 16a^2}}{b} \right).$$

Определить L_{ABC} если $a = 11$ см и $b = 9$ см.



12. Два тригонометрических тождества задаются формулами:

$$(a) \sin 5x = 5 \sin x - 20 \sin^3 x + 16 \sin^5 x; \quad (b) \sin^2 x \cos^2 x = \frac{1 - \cos 4x}{8}.$$

Проверить корректность данных тождеств, вычисляя значения левой и правой частей равенства при $x = \frac{\pi}{12}$.

13. Два тригонометрических тождества задаются формулами:

$$(a) \tan 3x = \frac{3 \tan x - \tan^3 x}{1 - 3 \tan^2 x}; \quad (b) \cos 4x = 8(\cos^4 x - \cos^2 x) + 1.$$

Проверить корректность данных тождеств, вычисляя значения левой и правой частей равенства при $x = 24^\circ$.

14. Определим две переменные: $\alpha = \pi/6$ и $\beta = 3\pi/8$. Используя эти переменные, покажите, что следующее тригонометрическое тождество корректно, вычисляя значения левых и правых сторон равенства.

$$\sin \alpha + \sin \beta = 2 \sin \frac{\alpha + \beta}{2} \cos \frac{\alpha - \beta}{2}.$$

15. Дано: $\int x \sin ax dx = \frac{\sin ax}{a^2} - \frac{x \cos ax}{a}$. Используйте MATLAB для вычисления следующего определенного интеграла: $\int_{\pi/3}^{3\pi/2} x \sin(0.6x) dx$.

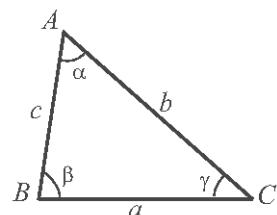
16. В показанном справа треугольнике $a = 5.3$ см, $\gamma = 42^\circ$ и $b = 6$ см. Определить a , b , и γ как переменные, и тогда:

- (a) Вычислить длину b по теореме косинусов.

(Теорема косинусов: $c^2 = a^2 + b^2 - 2ab \cos(\gamma)$).

- (b) Вычислите углы β и γ (в градусах) используя теорему косинусов.

- (c) Проверьте, что сумма углов равна 180° .



17. В показанном справа треугольнике $a = 5$ см, $b = 7$ см и $\gamma = 25^\circ$.

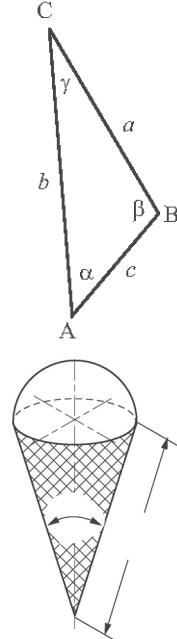
Определить a , b , и γ как переменные, и тогда:

- Вычислить длину c подставляя переменные в теорему косинусов. (Теорема косинусов: $c^2 = a^2 + b^2 - 2ab \cos(\gamma)$)
- Вычислите углы α и β (в градусах) используя теорему синусов.
- Проверить теорему тангенсов подстановкой результатов пункта (b) в правую и левую части равенства.

$$\text{Теорема тангенсов: } \frac{a-b}{a+b} = \frac{\tan\left(\frac{1}{2}(\alpha-\beta)\right)}{\tan\left(\frac{1}{2}(\alpha+\beta)\right)}.$$

18. В показанном справа конусе мороженого $L = 4"$, $\theta = 35^\circ$.

Конус заполнен мороженым так, что часть, которая выше конуса – полушарие. Определите объем этого мороженого.

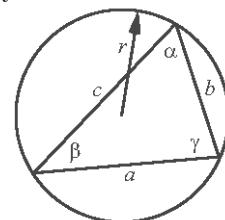


19. Для показанного справа треугольника, $a = 48$ мм, $b = 34$ мм и $\gamma = 83$. Определите a , b и γ как переменные и затем:

- Вычислить c , подставляя переменные в теорему косинусов.
(Теорема косинусов: $c^2 = a^2 + b^2 - 2ab \cos(\gamma)$)
- Вычислить радиус r круга, описанного около треугольника, используя формулу:

$$r = \frac{abc}{4\sqrt{s(s-a)(s-b)(s-c)}},$$

где $s = (a + b + c)/2$



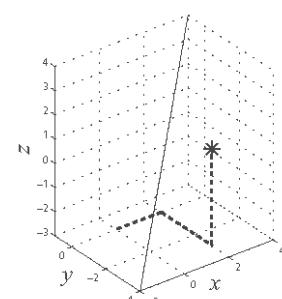
20. Параметрические уравнения линии в пространстве – это: $x = x_0 + at$, $y = y_0 + bt$ и $z = z_0 + ct$.

Расстояние d от точки до линии может быть вычислено по формуле:

$$d = d_{A0} \sin \left[a \cos \left(\frac{(x_A - x_0)a + (y_A - y_0)b + (z_A - z_0)c}{d_{A0}\sqrt{a^2 + b^2 + c^2}} \right) \right]$$

где $d_{A0} = \sqrt{(x_A - x_0)^2 + (y_A - y_0)^2 + (z_A - z_0)^2}$.

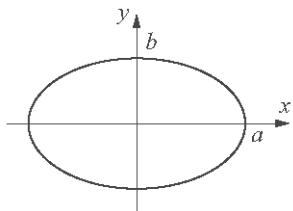
Определить расстояние от точки $A(2, -3, 1)$ до линии $x = -4 + 0.6t$, $y = -2 + 0.5t$, и $z = -3 + 0.7t$. Сначала определить переменные x_0, y_0, z_0, a, b и c , а затем использовать эти переменные (и координаты точки A) для вычисления переменной d_{A0} и, наконец, вычислить d .



21. Длина эллипса может быть аппроксимирована формулой:

$$C = \pi(3(a+b) - \sqrt{(3a+b)(a+3b)}).$$

Вычислить длину эллипса с полуосами $a = 16$ мм и $b = 11$ мм.



22. 315 человек должны быть перевезены автобусами, у которых по 37 посадочных мест. Вводя одну строку (команду) в командном окне, вычислить, сколько мест останется пустыми, если автобусы по очереди будут перевозить всех людей. (Подсказка: использовать встроенную функцию MATLAB ceil.)

23. 739 яблок должны быть упакованы и отгружены так, что в коробку помещается 54 штуки. В одну строку (команду) в командном окне вычислить, сколько яблок останется неупакованными, если могут быть отгружены только полные коробки. (Подсказка: использовать встроенную функцию MATLAB fix.)

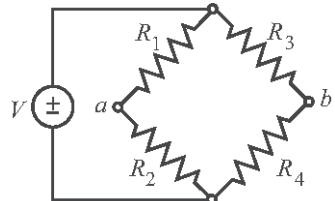
24. Присвойте переменной число 316 501.673 и затем, вводя одну команду, вычислите следующее:

- (a) Округление числа до сотен, к самой близкой сотне.
 (b) Вокруг числа для тысяч, к самой близкой тысяче.

25. Разность потенциалов между точками a и b в цепи моста Уитстона находится по формуле:

$$V_{ab} = V \left(\frac{R_1 R_3 - R_2 R_{41}}{(R_1 + R_2)(R_3 + R_4)} \right).$$

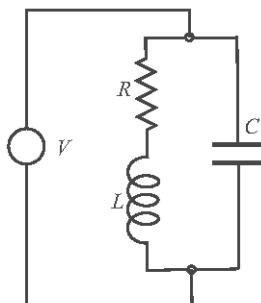
Вычислите разность потенциалов когда
 $V = 14$ В, $R_1 = 120.6$ Ом, $R_2 = 119.3$ Ом, $R_3 = 121.2$ Ом
 и $R_4 = 118.8$ Ом.



26. Резонансная частота f (в Гц) для приведенной цепи определяется формулой:

$$f = \frac{1}{2\pi} \sqrt{\frac{1}{LC} - \left(\frac{R}{L}\right)^2}.$$

Вычислить резонансную частоту, если $L = 0.15$ Гн, $R = 14$ Ом и $C = 2.6 \times 10^{-6}$ Фарад.



27. Число комбинаций выборок по r объектов из n объектов (число сочетаний) определяется формулой:

$$C_{n,r} = \frac{n!}{r!(n-r)!}.$$

- (a) Определить, сколько комбинаций возможно в лотерейной игре для выбора 6 чисел из 49.
- (b) Используя следующую формулу определите вероятность угадывания двух из шести вытянутых чисел.

$$\frac{C_{6,2} C_{43,4}}{C_{49,6}}.$$

(Использовать встроенную функцию factorial.)

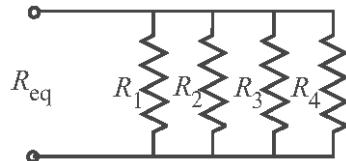
28. Формула для изменения основания логарифма имеет вид:

$$\log_a N = \frac{\log_b N}{\log_b a}.$$

- (a) Используя функцию MATLAB `log(x)` вычислить $\log_4 0.085$.
- (b) Используя функцию MATLAB `log10(x)` вычислить $\log_6 1500$.

29. Эквивалентное сопротивление четырех резисторов R_1, R_2, R_3 и R_4 , которые соединены параллельно определяется формулой:

$$R_{eq} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \frac{1}{R_4}}.$$

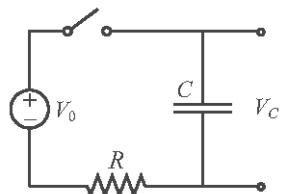


Вычислить R_{eq} , если $R_1 = 120$ Ом, $R_2 = 220$ Ом, $R_3 = 75$ Ом и $R_4 = 130$ Ом.

30. Напряжение V_C спустя t секунд после замыкания переключателя показанной цепи есть:

$$V_C = V_0(1 - e^{-t/(RC)}).$$

Для данных $V_0 = 36$ В, $R = 2500$ Ом и $C = 1600$ мкФ, вычислить V_C в течение 8 секунд после замыкания переключателя.



31. Для оценки возраста органического материала используется радиоактивный распад углерода-14. Это затухание моделируется показательной функцией $f(t) = f(0) e^{kt}$, где t – это время, $f(0)$ – количество материала при $t = 0$, $f(t)$ – количество материала в момент t и k – некоторая константа. Углерод-14 имеет период полураспада приблизительно 5730 лет. Выборка, взятая со следов человека в музее Акахуалинка в Никарагуа, показывает присутствие 77.45 % начального ($t = 0$) содержания углерода-14. Определите оценку возраста следов. Решить задачу записью программы в файле сценария. Программа сначала определяет постоянную k , а затем вычисляет t для $f(t) = 0.7745 f(0)$ и, наконец, округляет ответ на самый близкий год.

- 32.** Наибольший общий делитель – это самое большое положительное целое число, которое делит оба числа без остатка. Например, НОД чисел 8 и 12 равен 4. Используйте окно справки MATLAB для поиска встроенной функции MATLAB, которая определяет общий наибольший делитель двух чисел. Затем используйте функцию, чтобы показать что общий наибольший делитель:
- 91 и 147 есть 7.
 - 555 и 962 есть 37.

- 33.** Шкала моментных магнитуд (MMS), обозначаемая M_w , которая измеряет полную энергию, выделившуюся при землетрясении, задается формулой (магнитуда по Канамори):

$$M_w = \frac{2}{3} \log_{10} M_0 - 10.7,$$

где M_0 – величина сейсмического момента выраженного в динхсм (мера энергии, выделившейся при землетрясении). Определить, во сколько раз больше было выделено энергии M_0 при самом большом землетрясении в мире в Чили ($M_w = 9.5$) в 1960 г., чем при землетрясении вблизи острова Крыс (Rat Island), Аляска ($M_w = 8.7$), в 1965 г.

- 34.** Согласно специальной теории относительности, стержень длины L перемещающийся со скоростью v будет короче на величину δ , заданную формулой:

$$\delta = L \left(1 - \sqrt{1 - \frac{v^2}{c^2}} \right),$$

где c – скорость света (примерно 300×10^6 м/с). Вычислить, насколько сократится стержень длиной 2 м, перемещаясь со скоростью 5 000 м/с.

- 35.** Значение суммы B , полученной в результате размещения суммы P на сберегательный счет с фиксированной годовой процентной ставкой r после n лет, может быть вычислено по формуле:

$$B = P \left(1 + \frac{r}{m} \right)^{nm},$$

где m – это число начислений процентов в году. Рассмотрим депозит в размере 80 000\$ на срок 5 лет. Определить, насколько больше денег будет получено, если проценты будут начисляться ежедневно вместо ежегодного начисления процентов.

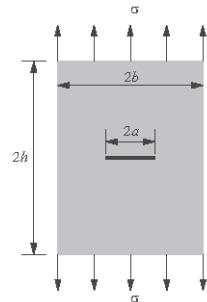
- 36.** Закон охлаждения Ньютона дает температуру объекта $T(t)$ в момент времени t через его температуру $T(0)$ при $t = 0$ и окружающую температуру T_s .

$$T = T_s + (T_0 - T_s) e^{-kt}.$$

Полицейский прибыл на место преступления в гостиничный номер в 21:18, где он нашел труп. Он сразу измерил температуру тела и зафиксировал, что она равна 79.5 °F. Ровно один час спустя он измерил температуру снова и зафиксировал, что, она равна 78.0 °F. Определить время смерти, предполагая, что температура тела жертвы до смерти была нормальной (98.6 °F) и что температура комнаты была постоянной 69 °F.

- 37.** Коэффициент интенсивности напряжения K предсказывает состояние напряжения (интенсивность напряжения) около трещины. Для пластины с трещиной и нагрузкой, показанной на рисунке, K определяется по формуле:

$$K = \sigma \sqrt{\pi a} \left(\frac{1 - \frac{a}{2b} + 0.326 \left(\frac{a}{b} \right)^2}{\sqrt{1 - \frac{a}{b}}} \right).$$



Определить K для случая, когда $\sigma = 12\,000$ пси, $h = 5$ дюйм, $b = 4$ дюйм и $a = 1.5$ дюйм.

- 38.** Распространение компьютерного вируса через компьютерную сеть может быть смоделировано формулой:

$$N(t) = 20e^{0.15t},$$

где $N(t)$ – число зараженных компьютеров и t время в минутах.

- (a) Определить, сколько времени потребуется, чтобы число зараженных компьютеров удвоилось.
 (b) Определить, сколько времени потребуется для заражения 1 000 000 компьютеров.

- 39.** Используйте окно справки, чтобы найти формат, который выводит на экран результат в виде отношение целых чисел. Например, когда число 3.125 будет выведено на экран как 25/8. Измените вывод на этот формат и выполните следующие операции:

$$(a) 5/8 + 16/6; \quad (b) 1/3 - 11/13 + 2.7^2.$$

- 40.** Формула Стерлинга для приближенного значения больших факториалов имеет вид:

$$n! = \sqrt{2\pi n} \left(\frac{n}{e} \right)^n.$$

Используйте эту формулу для вычисления $20!$. Сравните результат с истинным значением, полученным со встроенной функцией MATLAB `factorial` вычислением относительной ошибки

$$\text{ошибка} = (\text{Верноезнач.} - \text{Приблиз.знач.}) / \text{Верноезнач.}$$



ГЛАВА 2.

Создание массивов

Массив – это основная форма, которую использует MATLAB для хранения и управления данными. Массив – это список чисел, расположенных в строках и/или столбцах. Самый простой (одномерный) массив является строкой или столбцом чисел. Более сложный (двумерный) массив является набором чисел, расположенных в строках и столбцах. Одно из возможных использований массивов – это хранение информации и данных как в таблице. В науке и инженерии одномерные массивы часто представляют векторами, а двумерные массивы часто представляют матрицами. Эта глава показывает, как создавать и адресовать массивы, а глава 3 показывает, как использовать массивы в математических операциях. В дополнение к массивам, сделанным из чисел, массивы в MATLAB могут также быть списками символов, которые называют строками. Строки обсуждаются в разделе 2.10.

2.1. Создание одномерных массивов (векторов)

Одномерный *массив* – это список чисел, расположенных в строку или столбец. Рассмотрим в качестве примера представление положения точки в пространстве в трехмерной декартовой системе координат. Как показано на рис. 2.1, положение точки *A* определено списком трех чисел 2, 4 и 5, которые являются координатами точки.

Положение точки *A* может быть выражено радиус-вектором:

$$\mathbf{r}_A = 2 \mathbf{i} + 4 \mathbf{j} + 5 \mathbf{k},$$

где \mathbf{i} , \mathbf{j} и \mathbf{k} являются единичными векторами в направлении осей x , y и z , соответственно. Числа 2, 4 и 5 могут использоваться для определения вектора-строки или вектора-столбца.

Любой список чисел может рассматриваться как вектор. Например, табл. 2.1 содержит данные прироста населения, которые могут использоваться для создания

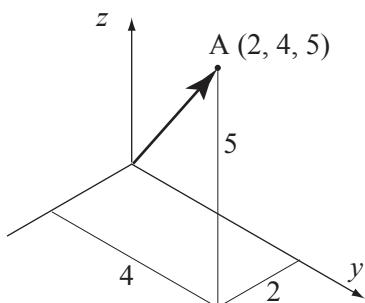


Рис. 2.1. Положение точки

двух списков чисел – один из годов, а другой – из значений популяции. Каждый список может быть занесён в виде элементов вектора с числами, расположеннымными в строку или в столбец.

Табл. 2.1. Данные популяции

Год	1984	1986	1988	1990	1992	1994	1996
Популяция (миллионов)	127	130	136	145	158	178	211

В MATLAB вектор создается присваиванием элементов вектора переменной. Это может быть сделано несколькими способами в зависимости от источника информации, которая используется для элементов вектора. Когда вектор содержит определенные числа, которые известны (как, например, координаты точки A), значение каждого элемента вводится непосредственно. Каждый элемент может быть также математическим выражением, которое может включать предопределенные переменные, числа и функции. Часто элементы вектора строки – это серия чисел с постоянным шагом. В таких случаях вектор может быть создан командами MATLAB. Вектор может быть также создан в результате математических операций, как показано в главе 3.

Создание вектора из известного списка чисел

Вектор создается вводом элементов (чисел) в квадратных скобках [].

имя_переменой = [элементы вектора]

Вектор-строка: для создания *вектора-строки* достаточно ввести элементы с пробелом или запятой между каждыми двумя элементами в квадратных скобках.

Вектор-столбец: Для создания *вектора-столбца* нужно ввести левую квадратную скобку [и затем ввести элементы с точкой с запятой между ними, или нажимать клавишу **Enter** после каждого элемента (тогда точка с запятой не требуется). В конце нужно ввести правую квадратную скобку] после последнего элемента.

Учебная программа 2.1 показывает, как используются данные из таблицы 2.1 и координата точки A для создания векторов-строк и векторов-столбцов.

Учебная программа 2.1. Создание векторов имеющихся данных

```
>> yr=[1984 1986 1988 1990 1992 1994 1996]
    Список лет присвоен вектор-строке с именем yr.
yr =
    1984      1986      1988      1990      1992      1994      1996
>> pop=[127; 130; 136; 145; 158; 178; 211]
    Данные о популяции присвоены
    вектор-столбцу с именем pop.
pop =
    127
    130
    136
```



```

145
158
178
211
>> pntAH=[2, 4, 5]
pntAH =
    2      4      5
>> pntAV=[2
4
5]
pntAV =
    2
    4
    5
>>

```

Координаты точки А присвоены вектор-строке с именем pntAH.

Координаты точки А присвоены вектор-столбцу с именем pntAV.
(здесь нажимается клавиша **Enter** после введения очередного элемента.)

Создание вектора с постоянным шагом, определяя первый член, шаг и последний член

В векторе с постоянным шагом разность между элементами одна и та же. Например, в векторе $v = 2 \ 4 \ 6 \ 8 \ 10$, интервал между элементами равен 2. Вектор, в котором первый элемент – m , шаг – q и последний элемент – n , создается следующим образом:

имя_переменой = [m:q:n] или имя_переменой = m:q:n

Скобки возможны, но не обязательны. Некоторые примеры:

```

>> x=[1:2:13]          Первый элемент 1, шаг 2 и последний элемент 13.
x =
    1    3    5    7    9    11   13
>> y=[1.5:0.1:2.1]    Первый элемент 1.5, шаг 0.1 и последний элемент 2.1.
y =
    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000    2.1000

>> z=[-3:7]           Первый элемент -3, последний элемент 7.
                           Если шаг опущен, значение по умолчанию = 1.
z =
    -3    -2    -1    0    1    2    3    4    5    6    7
>> xa=[21:-3:6]       Первый элемент 21, шаг -3 и последний элемент 6.
xa =
    21    18    15    12    9    6
>>

```

- Если числа m , q и n такие, что значение n не может быть получено прибавлением чисел q к m , то (для положительного n) последний элемент в векторе будет числом, которое не превышает n .

- Если введены только два числа (первый и последний члены, шаг опущен), тогда используется значение шага по умолчанию = 1.

Создание вектора с линейным (равномерно) распределением через определение первого и последнего членов и числа членов

Вектор с n элементами, которые линейно (равномерно) распределены на промежутке с первым элементом – x_i и последним элементом – x_f , может быть создан командой `linspace` (MATLAB сам определяет необходимый шаг):

имя_переменой = `linspace(xi, xf, n)`



↑ ↑ ↑
Первый Последний Число
элемент элемент элементов

Когда число элементов опущено, принимается значение по умолчанию 100. Некоторые примеры:

```
>> va=linspace(0, 8, 6)      6 элементов, первый элемент 0, последний – 8.  
va =  
0 1.6000 3.2000 4.8000 6.4000 8.0000  
>> vb=linspace(30,10,11)      11 элементов, первый и последний – 30 и 11.  
vb =  
30 28 26 24 22 20 18 16 14 12 10  
>> u=linspace(49.5,0.5)      Первый элемент 49.5, последний – 0.5.  
                                Когда число элементов опущено,  
u =                                значение по умолчанию = 100.  
Columns 1 through 10  
49.5000 49.0051 48.5101 48.0152 47.5202 47.0253  
46.5303 46.0354 45.5404 45.0455  
.....  
                                100 элементов выводятся на экран.  
Columns 91 through 100  
4.9545 4.4596 3.9646 3.4697 2.9747 2.4798  
1.9848 1.4899 0.9949 0.5000  
>>
```

2.2. Создание двумерных массивов (матриц)

Двумерный массив, называемый также матрицей, имеет числа в строках и столбцах. Матрицы могут использоваться для хранения информации в виде таблицы. Матрицы играют важную роль в линейной алгебре и используются в науке и инженерии для описания многих физических величин.

В квадратной матрице число строк и число столбцов равны. Например, матрица

7	4	9
3	8	1
6	5	3

3×3-матрица

является квадратной с тремя строками и тремя столбцами. Вообще, число строк и столбцов может отличаться. Например, матрица:

31	26	14	18	5	30
3	51	20	11	43	65
28	6	15	61	34	22
14	58	6	36	93	7

4×6-матрица

имеет четыре строки и шесть столбцов. Матрица $m \times n$ имеет m строк и n столбцов, и m -на- n называют размером матрицы.

Матрица создается присвоением переменной элементов матрицы. Это делается вводом элементов строка за строкой в квадратных скобках []. Сначала вводится левая скобка [, затем вводится первая строка с разделением элементов пробелами или запятыми. Чтобы ввести следующую строку надо ввести точку с запятой (символ конца строки и перехода к следующей), или нажать **Enter**. В конце последней строки вводится правая скобка].

```
имя_матрицы = [элементы 1-й строки; элементы 2-й строки;
... ; элементы последней строки]
```

Элементы, которые вводятся, могут быть числами или математическими выражениями, которые могут включать числа, предопределенные переменные и функции. У всех строк должно быть одно и то же число элементов. Если элемент – нуль, он должен быть введен как таковой. MATLAB выводит на экран сообщение об ошибке, если предпринята попытка задания неполной матрицы. Примеры матриц, определенных по-разному, показаны в Учебной программе 2.2.

Учебная программа 2.2. Создание матриц

```
>> a=[5 35 43; 4 76 81; 21 32 40]
a =
      5      35      43
      4      76      81
     21      32      40
      ↑
      Точка с запятой вводится перед
      введением новой строки.

>> b = [7 2 76 33 8
1 98 6 25 6
5 54 68 9 0]
b =
      7      2      76      33      8
      1     98      6      25      6
      5     54     68      9      0
      ↑
      Клавиша Enter нажимается перед введением
      новой строки.

>> cd=6; e=3; h=4;
      →
      Определены три переменные
```

```
>> Mat=[e, cd*h, cos(pi/3); h^2, sqrt(h*h/cd), 14]
Mat =
    3.0000    24.0000    0.5000
   16.0000    1.6330   14.0000
>>
```

↗ Элементы определены
математическими выражениями.

Строки матрицы могут также быть введены как векторы, используя процедуру создания векторов с постоянным шагом или команду `linspace`. Например:

```
>> A=[1:2:11; 0:5:25; linspace(10,60,6); 67 2 43 68 4 13]
A =
    1      3      5      7      9      11
    0      5     10     15     20     25
   10     20     30     40     50     60
   67      2     43     68      4     13
>>
```

В этом примере первые две строки введены как векторы чисел с постоянным шагом, третья строка вводится с использованием команды `linspace`, а в последней строке элементы вводятся индивидуально.

2.2.1. Команды `zeros`, `ones` и `eye`

Команды `zeros(m,n)`, `ones(m,n)` и `eye(n)` могут использоваться для создания матриц, у которых элементы имеют специальные значения. Команды `zeros(m,n)` и `ones(m,n)` создают матрицу с *m* строками и *n* столбцами, у которой все элементы = 0 и 1, соответственно. Команда `eye(n)` создает квадратную матрицу с *n* строками и *n* столбцами, у которой диагональные элементы равны 1, а остальные элементы = 0. Этую матрицу вызывают единичной матрицей. Примеры:

```
>> zr=zeros(3,4)
zr =
    0      0      0      0
    0      0      0      0
    0      0      0      0
>> ne=ones(4,3)
ne =
    1      1      1
    1      1      1
    1      1      1
    1      1      1
>> idn=eye(5)
idn =
    1      0      0      0      0
    0      1      0      0      0
```



```
>> 0     0     1     0     0  
      0     0     0     1     0  
      0     0     0     0     1
```

Матрицы могут быть также созданы в результате математических операций с векторами и матрицами. Эта тема обсуждается в главе 3.

2.3. Замечания о переменных в MATLAB

- Все переменные в MATLAB – массивы. Скаляр – это массив с одним элементом, вектор – это массив с одной строкой или одним столбцом элементов, а матрица – это массив с элементами в строках и столбцах.
- Переменная (скаляр, вектор или матрица) определяется вводом при присвоении переменной. Нет никакой необходимости определять размер массива перед присвоением элементов (один элемент для скаляра, строку или столбец для элементов вектора, или двумерный массив для элементов матрицы).
- После задания переменной – как скаляра, вектора, или матрицы – она может быть изменена на любой другой размер, или тип переменной. Например, скаляр может быть изменен на вектор или матрицу; вектор может быть изменен на скаляр, на вектор другой длины, или матрицу; а матрица может быть изменена в размерах, или может быть приведена к вектору или скаляру. Эти изменения производятся добавлением или удалением элементов. Об этом рассказывается в разделах 2.7 и 2.8.

2.4. Оператор транспонирования

Оператор *транспонирования* примененный к вектору преобразует вектор строку (столбец) в вектор столбец (строку). Если он применяется к матрице, он переводит строки (столбцы) в столбцы (строки). Оператор транспонирования применяется введением одинарной кавычки ' после переменной, которая будет преобразовываться. Примеры:

```
>> aa=[3 8 1]  
aa =  
      3     8     1  
>> bb=aa'  
bb =  
      3  
      8
```

Определение вектора строки aa.

Определение вектора столбца bb,
как транспонированного к вектору aa.



```

1
>> C=[2 55 14 8; 21 5 32 11; 41 64 9 1]      Определение матрицы С
C =                                                 с 3 строками и 4 столбцами.

   2      55      14      8
   21      5      32      11
   41      64      9      1
>> D=C'                                         Определение матрицы D как
D =                                                 транспонирование матрицы С.
                                                 (D имеет 4 строки и 3 столбца.)
   2      21      41
   55      5      64
   14      32      9
   8      11      1
>>
  
```

2.5. Адресация (индексация) массива

К элементам массива (вектор или матрица) можно обратиться индивидуально или к подгруппе. Это полезно, когда есть необходимость переопределить только некоторые из элементов, когда определенные элементы должны использоваться в вычислениях, или когда используется подгруппа элементов для определения новой переменной.

2.5.1. Вектор

Адрес (индекс) элемента в векторе – это его позиция в строке (или столбце). Для вектора с именем ve , $ve(k)$ отсылает к элементу в позиции k . Первая позиция имеет номер 1. Например, если вектор ve имеет девять элементов:

$$ve = 35 \ 46 \ 78 \ 23 \ 5 \ 14 \ 81 \ 3 \ 55$$

тогда

$$ve(4) = 23, \ ve(7) = 81 \text{ и } ve(1) = 35.$$

Отдельный элемент вектора $v(k)$ может использоваться в качестве переменной. Например, можно изменить значение только одного элемента вектора, присваивая новое значение по определенному адресу. Это можно сделать вводя: $v(k) = \text{значение}$. Отдельный элемент может также использоваться в качестве переменной в математическом выражении. Примеры:

```

>> VCT=[35 46 78 23 5 14 81 3 55]      Определение вектора.
VCT =
   35      46      78      23      5      14      81      3      55
>> VCT(4)                                     Вывод 4-го элемента.
ans =
   23
  
```



```
>> VCT(6)=273
```

Присвоение нового значения 6-му элементу.

```
VCT =
    35      46      78      23
```

Вывод всего вектора.

```
>> VCT(2)+VCT(8)
```

5 273 81 3 55

```
ans =
```



49

Использование элементов вектора
в математических выражениях.

```
>> VCT(5)^VCT(8)+sqrt(VCT(7))
```



```
ans =
```

134

```
>>
```

2.5.2. Матрица

Адрес элемента в матрице – это его позиция, определенная его номером строки и номером столбца. Для матрицы, присвоенной переменной *ma*, *ma(k,p)* соответствует элементу строки *k* и столбец *p*.

Например, если матрица: $ma = \begin{bmatrix} 3 & 11 & 6 & 5 \\ 4 & 7 & 10 & 2 \\ 13 & 9 & 0 & 8 \end{bmatrix}$,

тогда $ma(1,1) = 3$ и $ma(2,3) = 10$.

Как и с векторами, можно изменить значение элемента матрицы, присваивая новое значение этому элементу. Кроме того, отдельные элементы матрицы могут использоваться как переменные в математических выражениях и функциях. Некоторые примеры:

```
>> MAT=[3 11 6 5; 4 7 10 2; 13 9 0 8]
```

Создание матрицы 3×4 .

```
MAT =
```

3	11	6	5
4	7	10	2
13	9	0	8

```
>> MAT(3,1)=20
```

Присвоение нового значения элементу (3,1).

```
MAT =
```

3	11	6	5
4	7	10	2
20	9	0	8

```
>> MAT(2,4)-MAT(1,2)
```

Использование элементов
в математическом выражении.

```
ans =
```

-9

2.6. Использование двоеточия : в адресации массивов

Двоеточие может использоваться для адресации диапазона элементов в векторе или матрице.

Для вектора:

- $va(:)$ – обращается ко всем элементам вектора va (как для вектора строки, так и для вектора столбца).
- $va(m:n)$ – обращается к элементам вектора va в позиции от m до n .

Пример:

```
>> v=[4 15 8 12 34 2 50 23 11]          Создание вектора v
v =
    4      15      8      12      34      2      50      23      11
>> u=v(3:7)      Вектор u создается из элементов от 3-го до 7-го вектора v.
u =
    8      12      34      2      50
>>
```

Для матриц:

- $A(:, n)$ – обращается к элементам во всех строках n -го столбца матрицы A .
- $A(n, :)$ – обращается к элементам во всех столбцах n -ой строки n -ой матрицы A .
- $A(:, m:n)$ – обращается к элементам во всех строках между столбцами m и n матрицы A .
- $A(m:n, :)$ – обращается к элементам во всех столбцах между строками m и n матрицы A .
- $A(m:n, p:q)$ обращается к элементам в строках от m до n и в столбцах от p до q матрицы A .

Использование символа двоеточия в адресации элементов матриц демонстрируется в учебной программе 2.3.

Учебная программа 2.3. Использование двоеточие в адресации массивов

```
>> A=[1 3 5 7 9 11; 2 4 6 8 10 12; 3 6 9 12 15 18;
4 8 12 16 20 24; 5 10 15 20 25 30]      Определение матрицы A
A =                                              с 5-ю строками и 6-ю столбцами.
     1      3      5      7      9      11
     2      4      6      8      10      12
     3      6      9      12      15      18
     4      8      12      16      20      24
     5     10      15      20      25      30
```

```

>> B=A(:,3)
B =
    5
    6
    9
   12
   15
>> C=A(2,:)
C =
    2     4     6     8
>> E=A(2:4,:)
E =
    2     4     6     8     10    12
    3     6     9    12     15    18
    4     8    12    16     20    24
>> F=A(1:3,2:4)
F =
    3     5     7
    4     6     8
    6     9    12
>>

```

Определение вектора столбца B из всех элементов 3-го столбца матрицы A.

Определение вектора строки C из всех элементов 2-ой строки матрицы A.

Определение матрицы E из всех элементов строк со 2-ой до 4-ой матрицы A.

Определение матрицы F из всех элементов в строках с 1-ой до 3-ой и в столбцах с 2-го до 4-го матрицы A.

В руководстве к учебной программе 2.3 новые векторы и матрицы создаются из существующих при использовании диапазона элементов, или диапазона строк и столбцов (использование :). Однако можно выбрать только определенные элементы, или определенные строки и столбцы существующих переменных, чтобы создать новые переменные. Это делается введением выбранных элементов или строк, или столбцов в скобки, как показано ниже:

```

>> v=4:3:34
v =
    4     7    10    13    16    19    22    25    28    31    34
>> u=v([3, 5, 7:10])
u =
    10    16    22    25    28    31
>> A=[10:-1:4; ones(1,7); 2:2:14; zeros(1,7)]
A =
    10     9     8     7     6     5     4
     1     1     1     1     1     1     1
     2     4     6     8    10    12    14
     0     0     0     0     0     0     0
>> B = A([1,3],[1,3,5:7])
B =
    10     8     6     5     4
     2     6    10    12    14
>>

```

Создание вектора v из 11 элементов

Создание вектора u из 3-го и 5-го элементов и элементов с 7-го по 10-ый вектора v.

Создание матрицы A размеров 4×7.

Создание матрицы B из элементов 1-ой и 3-ей строк и столбцов с номерами 1-ый, 3-ий и с 5-го до 7-го матрицы A.



2.7. Добавление элементов к существующим переменным

Переменная, которая существует как вектор, или матрица, может быть изменена добавлением к ней элементов (напомним, что скаляр – это вектор с одним элементом). Вектор (матрица с единственной строкой или столбцом) может быть изменен добавлением дополнительных элементов, или он может быть изменен в двумерную матрицу. Строки и/или столбцы могут быть также добавлены к существующей матрице, чтобы получить матрицу другого размера. Добавление элементов может быть сделано простым присваиванием значений дополнительным элементам, или добавляя существующие переменные.

Добавление элементов к вектору

К существующему вектору новые элементы могут быть добавлены просто присваивая значения новым элементам. Например, если у вектора есть 4 элемента, вектор может быть сделан более длинным, присваивая значения элементам с индексами 5, 6, и так далее. Если вектор имеет n элементов и новое значение присваивается элементу с адресом $n + 2$ или больше, тогда MATLAB присваивает нулевые значения элементам, которые расположены между последним исходным элементом и новым элементом. Примеры:

```
>> DF=1:4                                Задание вектора DF с 4 элементами.
DF =
    1   2   3   4
>> DF(5:10)=10:5:35                  Добавление 6 элементов начиная с 5-го.
DF =
    1   2   3   4   10   15   20   25   30   35
>> AD=[5 7 2]                            Задание вектора AD с 3 элементами.
AD =
    5     7     2
>> AD(8)=4                                Присвоение значения 8-му элементу.
AD =
    5     7     2     0     0     0     0     4
>> AR(5)=24                               Присвоение значения 5-му элементу нового вектора.
AR =
    0     0     0     0    24
>>
```

Элементы могут также быть добавлены к вектору при помощи добавления существующих векторов. Два примера:

```
>> RE=[3 8 1 24];                      Задание вектора RE с 4 элементами.
>> GT=4:3:16;                          Задание вектора GT с 4 элементами.
>> KNH=[RE GT]                         Определение нового вектора KNH добавлением GT к RE.
```



```
>> KNH=[RE GT]      Определение нового вектора KNH добавлением GT к RE.
KNH =
    3      8      1      24      4      7      10      13      16
>> KNV=[RE'; GT']
KNV =
    3
    8
    1
    24
    4
    7
    10
    13
    16
>>
```

Создание нового вектора столбца KNV
добавлением к RE' вектора GT'

Добавление элементов к матрице

Строки и/или столбцы могут быть добавлены к существующей матрице присвоением значений новым строкам или столбцам. Это может быть сделано присвоением новых значений, или добавляя существующие переменные. Это должно быть сделано аккуратно, так как размер добавленных строк или столбцов должен соответствовать существующей матрице. Примеры:

```
>> E=[1 2 3 4; 5 6 7 8]      Создание матрицы E размера 2×4.
E =
    1      2      3      4
    5      6      7      8
>> E(3,:)=[10:4:22]          Добавление вектора [10 14 18 22] как третьей
E =                           строки матрицы E.
    1      2      3      4
    5      6      7      8
   10     14     18     22
>> K=eye(3)                  Создание матрицы K размера 3×3.
K =
    1      0      0
    0      1      0
    0      0      1
>> G=[E K]                  Добавление матрицы K к матрице E. Число
G =                           строк в E и K должно быть одинаковым.
    1      2      3      4      1      0      0
    5      6      7      8      0      1      0
   10     14     18     22      0      0      1
>>
```

Если матрица имеет размер $m \times n$ и присваивается новое значение элементу с адресом вне размеров матрицы, то MATLAB автоматически увеличивает размер

матрицы для включения нового элемента. Другим элементам, которые добавляются при этом, присваиваются нули. Примеры:

```
>> AW=[3 6 9; 8 5 11]          Создание матрицы 2x3.
AW =
    3      6      9
    8      5     11
>> AW(4,5)=17                Присвоение значения элементу (4,5).
AW =
    3      6      9      0      0      MATLAB создает матрицу 4x5
    8      5     11      0      0      и присваивает нули новым
    0      0      0      0      0      элементам.
    0      0      0      0      17
>> BG(3,4)=15                Присвоение значения элементу (3,4)
BG =                           новой (ранее не существующей) матрице.
    0      0      0      0      MATLAB создает матрицу 3x4
    0      0      0      0      и присваивает нули остальным
    0      0      0      15      элементам кроме BG(3,4).
>>
```

2.8. Удаление элементов

Элемент или диапазон элементов существующей переменной может быть *удален* путем повторного присвоения «ничего» (пустого множества) этим элементам. Это делается с помощью двойных квадратных скобок [], когда между ними ничего нет. Удаляя элементы, вектор может быть сделан короче, и матрица может быть сделана меньше. Примеры:

```
>> kt=[2 8 40 65 3 55 23 15 75 80]      Задание вектора из 10 элементов.
kt =
    2      8      40      65      3      55      23      15      75      80
>> kt(6)=[]
kt =                                         Удаление 6-го элемента.
                                         Получается вектор из 9 элементов.
    2      8      40      65      3      23      15      75      80
>> kt(3:6)=[]
kt =                                         Удаление элементов с 3-го до 6-го.
                                         Получается вектор из 5 элементов.
    2      8      15      75      80
>> mtr=[5 78 4 24 9; 4 0 36 60 12; 56 13 5 89 3]   Создание матрицы 3x5.
mtr =
    5      78      4      24      9
    4      0      36      60      12
    56     13      5      89      3
>> mtr(:,2:4)=[]
mtr =                                         Удаление всех столбцов с 2-го до 4-го.
                                         Получается вектор из 5 элементов.
    5      9
    4     12
    56     3
>>
```

2.9. Встроенные функции для управления массивами

В MATLAB есть много *встроенных функций* для управления и обработки массивов. Некоторые из них приведены ниже:

Таблица 2.2. Встроенные функции для управления массивами

Функция	Описание	Пример
<code>length(A)</code>	Возвращает число элементов вектора A.	<pre>>> A=[5 9 2 4]; >> length(A) ans = 4</pre>
<code>size(A)</code>	Возвращает строку [m, n], где m и n – размеры m × n матрицы A.	<pre>>>> A=[6 1 4 0 12; 5 19 6 8 2] A = 6 1 4 0 12 5 19 6 8 2 >> size(A) ans = 2 5</pre>
<code>reshape(A, m, n)</code>	Создание m-на-n матрицы из элементов матрицы A. Элементы берутся по столбцам – столбец за столбцом и расставляются по столбцам. Матрица A должна иметь m × n элементов.	<pre>>> A=[5 1 6; 8 0 2] A = 5 1 6 8 0 2 >> B = reshape(A, 3, 2) B = 5 0 8 6 1 2</pre>
<code>diag(v)</code>	Для вектора v создает квадратную матрицу с элементами v на диагонали.	<pre>>> v=[7 4 2]; >> A=diag(v) A = 7 0 0 0 4 0 0 0 2</pre>
<code>diag(A)</code>	Для матрицы A создает вектор из элементов диагонали матрицы A.	<pre>>> A=[1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 >> vec=diag(A) vec = 1 5 9</pre>

Дополнительные встроенные функции для управления массивами описаны в окне справки. В этом окне, выберите **MATLAB**, затем в **Contents Functions** и затем **By Category**.

Пример задачи 2.1. Создание матриц

Используя команды `ones` и `zeros`, создать 4×5 матрицу, в которой первые две строки – нули, а следующие две строки – единицы.

Решение

Задача решается следующими командами в командном окне.

```
>> A(1:2, :)=zeros(2,5)                                         Создание 4×5 матрицы из нулей.
A =
    0     0     0     0     0
    0     0     0     0     0
>> A(3:4, :)=ones(2,5)                                         Добавление 3 и 4 строк из единиц.
A =
    0     0     0     0     0
    0     0     0     0     0
    1     1     1     1     1
    1     1     1     1     1
```

Другое решение этой задачи:

```
A=[zeros(2,5);ones(2,5)]                                         Создание 4×5 матрицы из двух 2×5 матриц.
A =
    0     0     0     0     0
    0     0     0     0     0
    1     1     1     1     1
    1     1     1     1     1
```

Пример задачи 2.2. Создание матриц

Создать 6×6 матрицу, в которой средние две строки и средние два столбца – единицы, а остальная часть заполнена нулями.

Решение

```
>> AR=zeros(6,6)                                                 Создание 6×6 матрицы из нулей.
AR =
    0     0     0     0     0     0
    0     0     0     0     0     0
    0     0     0     0     0     0
    0     0     0     0     0     0
    0     0     0     0     0     0
    0     0     0     0     0     0
```

```
>> AR(3:4, :)=ones(2, 6)
```

```
AR =
```

0	0	0	0	0	0
0	0	0	0	0	0
1	1	1	1	1	1
1	1	1	1	1	1
0	0	0	0	0	0
0	0	0	0	0	0

Переприсвоение числа 1 3-ей и 4-ой строкам.

```
>> AR(:, 3:4)=ones(6, 2)
```

```
AR =
```

0	0	1	1	0	0
0	0	1	1	0	0
1	1	1	1	1	1
1	1	1	1	1	1
0	0	1	1	0	0
0	0	1	1	0	0

Переприсвоение числа 1 3-му и 4-му столбцам.

Пример задачи 2.3. Манипуляции с матрицами

Дана 5×6 матрица A , 3×6 матрица B и вектор v с 9 элементами.

$$A = \begin{bmatrix} 2 & 5 & 8 & 11 & 14 & 17 \\ 3 & 6 & 9 & 12 & 15 & 18 \\ 4 & 7 & 10 & 13 & 16 & 19 \\ 5 & 8 & 11 & 14 & 17 & 20 \\ 6 & 9 & 12 & 15 & 18 & 21 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 10 & 15 & 20 & 25 & 30 \\ 30 & 35 & 40 & 45 & 50 & 55 \\ 55 & 60 & 65 & 70 & 75 & 80 \end{bmatrix},$$

$$v = [99 \ 98 \ 97 \ 96 \ 95 \ 94 \ 93 \ 92 \ 91]$$

Создать эти три массива в командном окне и затем одной командой заменить:

- последние четыре столбца первой и третьей строк матрицы A первыми четырьмя столбцами первых двух строк матрицы B ,
- последние четыре элемента четвертой строки заменить на элементы с 5-го по 8-ой вектора v и
- последние четыре элемента пятой строки – элементами с 2-го по 5-ый из третьей строки матрицы B .

Решение

```
>> A=[2:3:17; 3:3:18; 4:3:19; 5:3:20; 6:3:21]
```

```
A =
```

2	5	8	11	14	17
3	6	9	12	15	18
4	7	10	13	16	19
5	8	11	14	17	20
6	9	12	15	18	21



```

>> B=[5:5:30; 30:5:55; 55:5:80]
B =
    5     10     15     20     25     30
   30     35     40     45     50     55
   55     60     65     70     75     80
>> v=[99:-1:91]
v =
    99     98     97     96     95     94     93     92     91
>> A([1 3 4 5],3:6) = [B([1 2],1:4); v(5:8); B(3,2:5)]
 4x4 матрица из      4x4 матрица. Первые две строки — это
элементов столбцов 3 - 6      элементы столбцов 1 - 4 из строк 1 и 2 матрицы B.
и строк 1, 3, 4, и 5.      Третья строка состоит из элементов 5 - 8 вектора v.
                                Четвертая строка состоит из элементов 2 - 5
                                из строки 3 матрицы B.
A =
    2     5     5     10     15     20
    3     6     9     12     15     18
    4     7    30     35     40     45
    5     8    95     94     93     92
    6     9    60     65     70     75

```

2.10. Строки символов и строки как переменные

Строка – это массив *символов*. Она создается введением символов между одинарными кавычками.

- Строки могут включать буквы, цифры, другие символы и пробелы.
- Примеры строк: 'ad ef', '3%fr2', '{edcba:21!', 'MATLAB'.
- Стока, которая содержит одинарную кавычку, создается введением двух одинарных кавычек в пределах строки (т.е. в пределах внешних одинарных кавычек).
- Когда вводится строка, после введения первой одинарной кавычки, цвет текста на экране изменяется на коричневый. Когда вводится одинарная кавычка в конце строки, цвет строки изменяется на фиолетовый.

MATLAB использует строки несколькими способами. Строки используются при выводе на экран текстовых сообщений о результатах команд (глава 4), в форматировании команд графиков (глава 5), и как входные аргументы некоторых функций (глава 7). В этих главах представлены детали использования строк в указанных целях.

- Когда строки используются в форматировании графиков (обозначения к осям, заголовки и текстовым примечаниям), символы в пределах строки могут быть отформатированы, чтобы иметь указанный шрифт, размер, позицию (верхний регистр, нижний регистр), цвет, и т. д. См. главу 5 для деталей.

Строки могут также быть присвоены переменным, просто вводя строку на правой стороне оператора присвоения, как показано в примерах ниже:

```
>> a='FRty 8'  
a =  
FRty 8  
>> B='My name is John Smith'  
B =  
My name is John Smith  
>>
```

Когда переменная определена как строка, символы строки сохраняются в массиве, как числа. Каждый символ, включая пробел, является элементом в массиве. Это означает, что односторонняя строка – это вектор-строка, в которой число элементов равно числу символов. Элементы векторов адресуются позицией. Например, в векторе B, который был определен выше, 4-й элемент – это символ n, 12-й элемент – символ J, и так далее.

```
>> B(4)  
ans =  
n  
>> B(12)  
ans =  
J  
>>
```

Как и с векторами, которые содержат числа, можно изменить определенные элементы, обращаясь к ним непосредственно. Например, в векторе B выше имя Джон может быть изменено на имя Билл:

```
>> B(12:15)='Bill'      Использование двоеточие для присвоения  
B =                      новых символов элементам от 12-го до 15-го в векторе B.  
My name is Bill Smith  
>>
```

Строки могут также быть помещены в матрицу. Как и с числами, это делается при помощи точки с запятой ; (или нажатием клавиши **Enter**) в конце каждой строки. Каждая строка (матрицы) должна быть введена как строка (символьная), что означает, что она должна быть вложена в одинарные кавычки. Кроме того, как и с числовой матрицей, у всех символьных строк матрицы должно быть то одно и то же число элементов (символов). Это требование может вызвать некоторые проблемы при создании строк с различными вариантами текста. Строки могут быть сделаны одинаковой длины (иметь одно и то же число элементов) добавлением пробелов.

У MATLAB есть встроенная функция по имени `char`, которая создает массив со строками, имеющими одно и то же число символов при вводе строк не одинаковой длины. MATLAB делает длину всех строк равной длине самой длинной строки (символов), прибавляя пробелы в конце коротких строк. У функции `char` строки вводятся как строки (символов, strings), разделенные запятыми (а не точками с запятой), согласно следующему формату:

```
variable_name= char('string 1','string 2','string 3')
```

Например:

```
>> Info=char('Student Name:','John Smith','Grade:','A+')
Info =
Student Name:
John Smith
Grade:
A+
>>
```

↑ Переменной с именем `Info` присвоено
четыре строки строк символов, каждая разной длины.

Функция `char` создает массив с четырьмя строками
одной длины, равной самой длинной строке, добавляя
пробелы к более коротким линиям.

Переменная может быть определена или как число или как строка, составленная из тех же цифр. Например, как показано ниже, переменная `x` определена как число 536, а `y` определена как строка символов, составленная из цифр 536.

```
>> x=536
x =
536
>> y='536'
y =
536
>>
```

Эти две переменные не то же самое даже притом, что они кажутся идентичными на экране. Отметим, что символы 536 на строке ниже `x=` расположены с отступом, в то время как символы 536 на строке ниже `y=` – без отступа. Переменная `x` может использоваться в математических выражениях, тогда как переменная `y` не может.

2.11. Задачи

1. Создать вектор строку, которая имеет следующие элементы: 8, $10/4$, $12 \cdot 1.4$, 51, $\tan 85^\circ$, $\sqrt{26}$ 0.15.
2. Создать вектор строку, которая имеет следующие элементы: $\sqrt{15} \cdot 10^3$, $\frac{25}{10 - 6^2}$, $\ln 35/0.43$, $\frac{\sin 65^\circ}{\cos 80^\circ}$ 129 и $\cos^2(\pi/20)$.

3. Создать вектор столбец, который имеет следующие элементы: $25.5, \frac{14 \tan 58^\circ}{2.1^2 + 11}, 6!, 2.7^4, 0.00552$ и $\pi/5$.
4. Создать вектор столбец, который имеет следующие элементы: $\frac{32}{3.2^2}, \sin^2 35^\circ, 6.1, \ln 29^2, 0.00552, \ln^2 29$ и 133.
5. Определить переменные $x = 0.85, y = 112.5$ и использовать их для создания вектора столбца с элементами: $y, y^x, \ln(y/x), x \times y$ и $x + y$.
6. Определить переменные $a = 3.5, b = -6.4$ и использовать их для создания вектора строки с элементами: $a, a^2, a/b, a \cdot b$ и \sqrt{a} .
7. Создать вектор строку, у которого первый элемент – 1, последний элемент – 43, с инкрементом 6 между элементами (1, 7, 13, ..., 43).
Создать вектор строку, которая имеет следующие элементы: $8, 10/4, 12 \times 1.4, 51, \tan 85^\circ, \sqrt{26}, 0.15$.
8. Создать вектор строку с 11 равномерно (линейно) распределенными элементами, у которых первый элемент 96, а последний элемент 2.
9. Создать вектор столбец, у которого первый элемент 26, элементы уменьшаются с инкрементом -3.6 , а последний элемент –10. (Вектор столбец может быть создан транспонированным вектора строки.)
10. Создать вектор столбец с 9 равномерно (линейно) распределенными элементами, в котором первый элемент –34, а последний элемент –7. (Вектор столбец может быть создан транспонированным вектора строки.)
11. Используя символ двоеточия, создать вектор строку (присвоить ему имя `Fives`) с пятью элементами, все равными 5.
12. Используя команду `linspace`, создать вектор строку (присвоить ему имя `Nines`) с девятью элементами, все равными 9.
13. Использовать одну единственную команду для создания вектора строки (с именем `a`) с 6 элементами так, что последний элемент 4.7, а остальные элемент – нули. Не вводить явно элементы вектора.
14. Использовать одну единственную команду, для создания вектора строки (с именем `b`) с 8 элементами так, что последние три элемента 3.8, а остальные элементы – нули. Не вводить явно элементы вектора.
15. Использовать одну единственную команду, для создания вектора строки (с именем `b`) с 11 элементами так, что

b =

0	2	4	6	8	10	12	9	6	3	0
---	---	---	---	---	----	----	---	---	---	---

16. Создать два вектора строки: $a=2:3:17$ и $b=3:4:15$. Затем, используя только имена векторов (a и b), создать вектор строку c, которая состоит из элементов a и следующих за ними элементов b.
17. Создать два вектора столбца: $a=[2:3:17]'$ и $b=[3:4:15]'$. Затем, используя только имена векторов (a и b), создать вектор столбец c, который состоит из элементов a и следующих за ними элементов b.
18. Создать вектор (с именем vta) из 10 элементов, из которых первый 8, инкремент 7, а последний элемент 71. Затем присвоить элементы vta новому вектору (с именем vtB) из 7 элементов. Первые 4 элемента – это первые 4 элемента вектора vta, а последние 3 – это последние 3 элемента вектора vta. Не вводить явно элементы вектора vta.
19. Создать вектор (с именем vtc) из 12 элементов, из которых первое 5, инкремент 4 и последний элемент 49. Затем, присваивая элементы vtc к новым векторам, создать следующие два вектора:
- Вектор (с именем Codd), который содержит все элементы vtc с нечетным индексами; то есть, $Codd = 5 \ 13 \ 21 \dots \ 45$.
 - Вектор (с именем Ceven), который содержит все элементы vtc с четным индексами; то есть, $Ceven = 9 \ 17 \ 25 \dots \ 49$.
- В обеих случаях использовать векторы четных и нечетных чисел для индексов Codd и Ceven, соответственно. Не вводить явно элементы векторов.
20. Создать вектор (по имени vctD) из 9 элементов, из которых первый 0, инкремент 3 и последний элемент 27. Затем создать вектор (назвать его vctDop), который состоит из элементов vctD в обратном порядке. Сделать это, присваивая элементы vctD к vctDop. (Не вводить явно элементы вектора vctDop.)
21. Создать следующую матрицу с использованием векторной записи для создания векторов с постоянным шагом и/или командой linspace. Не вводите явно отдельные элементы.

$$A = \begin{bmatrix} 130 & 110 & 90 & 70 & 50 & 30 & 10 \\ 1 & 2.8333 & 4.6667 & 6.5 & 8.3333 & 10.667 & 12 \\ 12 & 22 & 32 & 42 & 52 & 62 & 72 \end{bmatrix}.$$

22. Создать следующую матрицу с использованием векторных обозначений для создания векторов командой linspace. Не вводите явно отдельные элементы.

$$B = \begin{bmatrix} 5 & 2 & 3 \\ 5 & 2 & 3 \\ 5 & 2 & 3 \\ 5 & 2 & 3 \end{bmatrix}.$$

23. Создать следующую матрицу, вводя только одну команду. Не вводите явно отдельные элементы.

$$C = \begin{bmatrix} 7 & 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 & 7 \end{bmatrix}.$$

24. Создать следующую матрицу, вводя только одну команду. Не вводить явно отдельные элементы.

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 6 \end{bmatrix}.$$

25. Создать следующую матрицу, вводя только одну команду. Не вводить явно отдельные элементы.

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 4 & 3 \\ 0 & 0 & 2 & 1 & 0 \end{bmatrix}.$$

26. Создать следующую матрицу, вводя только одну команду. Не вводить явно отдельные элементы.

$$F = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 10 & 20 \\ 0 & 0 & 2 & 8 & 26 \\ 0 & 0 & 3 & 6 & 32 \end{bmatrix}.$$

27. Создать три вектора строки:

$$a = [3 \ -1 \ 5 \ 11 \ -4 \ 2], \quad b = [7 \ -9 \ 2 \ 13 \ 1 \ -2], \quad c = [-2 \ 4 \ -7 \ 8 \ 0 \ 9].$$

- (a) Использовать эти три вектора в команде MATLAB создания матрицы, в которой строки – векторы a , b и c .
- (b) Использовать эти три вектора в команде MATLAB создания матрицы, в которой столбцы – векторы a , b и c .

28. Создать три вектора строки:

$$a = [3 \ -1 \ 5 \ 11 \ -4 \ 2], \quad b = [7 \ -9 \ 2 \ 13 \ 1 \ -2], \quad c = [-2 \ 4 \ -7 \ 8 \ 0 \ 9].$$



- (a) Использовать эти три вектора в команде MATLAB, для создания матрицы, у которой первая, вторая и третья строки состоят из последних четырех элементов векторов a , b и c , соответственно.
- (b) Использовать эти три вектора в команде MATLAB, для создания матрицы, у которой первый, второй и третий столбцы состоят из первых трех элементов векторов a , b и c , соответственно.

29. Создать два вектора строки:

$$a = [3 \ 9 \ -0.5 \ 3.6 \ 1.5 \ -0.8 \ 4], \quad b = [12 \ -0.8 \ 6 \ 2 \ 5 \ 3 \ -7.4].$$

- (a) Использовать эти два вектора в команде MATLAB для создания матрицы такой, что первая строка состоит из элементов 3–6 вектора a , вторая строка состоит из элементов 4–7 вектора a , а третья строка состоит из элементов 2–5 вектора b .
- (b) Использовать эти два вектора в команде MATLAB для создания матрицы такой, что первый столбец состоит из элементов 2–7 вектора a , а второй столбец состоит из элементов 1–3 и 5–7 из вектора b .
- 30.** Вручную (карандаш и бумага) напишите то, что будет выведено на экран, если следующие команды MATLAB будут выполнены. Проверьте свои ответы, выполняя команды с MATLAB. (Части (b), (c), (d) и (e) используют вектор, который был определен в пункте (a).)

$$(a) a=1:4:17 \quad (b) b=[a(1:3) \ a] \quad (c) c=[a;a]' \\ (d) d=[a' \ a']' \quad (e) e=[[a; \ a; \ a; \ a; \ a] \ a']$$

31. В MATLAB определен следующий вектор:

$$v = [6 \ 11 \ -4 \ 5 \ 8 \ 1 \ -0.2 \ -7 \ 19 \ 5].$$

Вручную (карандаш и бумага) напишите то, что будет выведено на экран, если следующие команды MATLAB будут выполнены. Проверьте свои ответы, выполняя команды с MATLAB.

$$(a) a=v(3:8) \quad (b) b=v([1,3,2:7,4,6]) \quad (c) c=v([9,1,5,4])'$$

32. В MATLAB определен следующий вектор:

$$v = [6 \ 11 \ -4 \ 5 \ 8 \ 1 \ -0.2 \ -7 \ 19 \ 5].$$

Вручную (карандаш и бумага) напишите то, что будет выведено на экран, если следующие команды MATLAB будут выполнены. Проверьте свои ответы, выполняя команды с MATLAB.

$$(a) a=[v([1:3 \ 7:-1:5]);v([10,1,4:6,2])] \\ (b) b=[v([9,2:4,1])' \ v([5 \ 3 \ 10 \ 2 \ 7])' \ v([10:-2:4,10])']$$

33. Создать следующую матрицу A .

$$A = \begin{bmatrix} 36 & 34 & 32 & 30 & 28 & 26 \\ 24 & 22 & 20 & 18 & 16 & 14 \\ 12 & 10 & 8 & 6 & 4 & 2 \end{bmatrix}.$$

Используйте только одну команду и оператор двоеточия для обращения к диапазону элементов (не вводя явно отдельные элементы) для того, чтобы:

- (a) Создать вектор строку с шестью элементами с именем `ha`, которая содержит элементы второй строки матрицы A .
- (b) Создать вектор столбец с тремя элементами с именем `hb`, который содержит элементы шестого столбца матрицы A .
- (c) Создать вектор строку с пятью элементами с именем `hc`, которая содержит первые два элемента третьей строки матрицы A и последние три элемента первой строки A .

34. Создать следующий вектор A :

$$A = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18].$$

Затем используя встроенную функция MATLAB `reshape`, создать следующую матрицу B из вектора A :

$$B = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 \\ 2 & 5 & 8 & 11 & 14 & 17 \\ 3 & 6 & 9 & 12 & 15 & 18 \end{bmatrix}.$$

Используйте только одну команду и оператор двоеточия для обращения к диапазону элементов (не вводя явно отдельные элементы) для того, чтобы:

- (a) Создать вектор столбец с девятью элементами с именем `va`, который содержит элементы первого, третьего и пятого столбцов матрицы B .
- (b) Создать вектор строку с семью элементами с именем `vб`, которая содержит элементы 2–5 из второй строки матрицы B и элементы третьего столбца B .
- (c) Создать вектор строку с шестью элементами по имени `vc`, которая содержит элементы 3–5 из первой строки, и элементы 2–4 из третьей строки B .

35. Создать следующий вектор A :

$$C = [1.5 2 2.5 3 3.5 4 4.5 5 9.6 9.1 8.9 8.1 7.6 6.6 6.1].$$

Затем, используя встроенную функция MATLAB `reshape` и оператор транспонирования, создать следующую матрицу D из вектора C :

$$D = \begin{bmatrix} 1.5 & 2 & 2.5 & 3 \\ 3.5 & 3 & 4.5 & 5 \\ 9.6 & 9.1 & 8.6 & 8.1 \\ 7.6 & 7.1 & 6.6 & 6.1 \end{bmatrix}.$$

Используйте только одну команду и оператор двоеточия для обращения к диапазону элементов (не вводя явно отдельные элементы) для того, чтобы:

- (a) Создать вектор столбец с восемью элементами по имени `Da`, который содержит элементы первых и третьих строк матрицы D .
- (b) Создать вектор строку с восемью элементами под названием `Db`, которая содержит элементы второго и четвертого столбцов матрицы D .
- (c) Создать вектор строку с восемью элементами по имени `Dc`, которая содержит первые два элемента первой строки, последние три элемента второго столбца и первые три элемента четвертой строки матрицы D .

36. Создать следующую матрицу E :

$$E = \begin{bmatrix} 0 & 5 & 5 & 5 & 5 & 5 \\ 0.1 & 0.3 & 0.5 & 0.7 & 0.7 & 0.9 \\ 12 & 9 & 6 & 3 & 0 & -3 \\ 6 & 7 & 8 & 9 & 10 & 11 \end{bmatrix}.$$

- (a) Создать матрицу F из элементов второй и третьей строк и столбцов с третьего до пятого матрицы E .
- (b) Создать матрицу G из элементов всех строк и столбцов с третьего до шестого матрицы E .

37. Создать следующую матрицу H :

$$H = \begin{bmatrix} 1.25 & 1.5 & 1.75 & 2 & 2.25 & 2.5 & 2.75 \\ 1 & 2 & 3 & 1 & 2 & 3 & 4 \\ 45 & 40 & 35 & 30 & 25 & 20 & 15 \end{bmatrix}.$$

- (a) Создать матрицу G такую, что ее первая строка включает первые три элемента и последние два элемента первой строки H , а вторая строка G включает последние пять элементов третьей строки H .
- (b) Создать матрицу K такую, что первая, вторая, третья и четвертая строки – это второй, третий, пятый и седьмой столбцы матрицы H .

38. В MATLAB определена следующая матрица:

$$M = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 \\ 2 & 5 & 8 & 11 & 14 & 17 \\ 3 & 6 & 9 & 12 & 15 & 18 \end{bmatrix}.$$

Вручную (карандаш и бумага) написать то, что будет выведено на экран, если следующие команды MATLAB будут выполнены. Проверьте свои ответы, выполняя команды с MATLAB.

$$(a) A=M([1,3],[1,5,6])$$

$$(b) B=M(:,[4,4:6])$$

$$(c) C=M([1,2],:)$$

$$(d) D=M([2,3],[2,3])$$

39. В MATLAB определена следующая матрица:

$$M = \begin{bmatrix} 2 & 10 & 18 & 29 & 41 \\ 4 & 12 & 20 & 32 & 44 \\ 6 & 14 & 23 & 35 & 47 \\ 8 & 16 & 26 & 38 & 50 \end{bmatrix}.$$

Вручную (карандаш и бумага) написать то, что будет выведено на экран, если следующие команды MATLAB будут выполнены. Проверьте свои ответы, выполняя команды с MATLAB.

$$(a) A=[N(1,1:4)', N(2,2:5)']$$

$$(b) B=[N(:,3)' N(3,:)]$$

$$(c) C(3:4,5:6)=N(2:3,4:5)$$

40. Вручную (карандаш и бумага) написать то, что будет выведено на экран, если следующие команды MATLAB будут выполнены. Проверьте свои ответы, выполняя команды с MATLAB.

```
v=1:2:23
M=reshape(v,3,4)
M(2,:)=[]
M(:,3)=[]
N=ones(size(M))
```

41. Используя команды zeros, ones и eye создать одной командой следующие массивы:

$$(a) \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}; (b) \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}; (c) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

42. Используя команды zeros, ones и eye создать одной командой следующие массивы:

$$(a) \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}; \quad (b) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \quad (c) \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

43. Используйте команды zeros, ones и eye для создания следующих массивов:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Используя переменные A, B, и C написать команду, которая создает следующую матрицу D:

$$D = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

44. Создать 2×3 матрицу A, у которой все элементы 1. Затем повторным присвоением A (несколько раз) сделать так, чтобы матрица A стала иметь вид:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$



ГЛАВА 3.

Математические операции с массивами

После создания, переменные в MATLAB могут использоваться в большом числе разнообразных математических операций. В главе 1 все переменные, которые использовались в математических операциях, были определены как скаляры. Тем не менее, все они были массивами (с одной строкой и одним столбцом и с одним элементом) и математические операции выполнялись с отдельными числами. Однако массивы могут быть одномерными (массивы с одной строкой, или с одним столбцом), двумерными (массивы со многими строками и столбцами) и даже более высоких размерностей. В этих случаях математические операции являются более сложными. MATLAB, как указывает его имя, проектировался для выполнения расширенных операций с массивами, имеющих много применений в науке и инженерии. Эта глава представляет основные наиболее распространенные математические операции, которые выполняет MATLAB используя массивы.

Сложение и вычитание – это относительно простые операции и изложены в первую очередь, в разделе 3.1. Другие основные операции – умножение, деление и возведение в степень – могут выполняться в MATLAB двумя различными способами. Один способ, который использует стандартные символы (*, / и ^), следует правилам линейной алгебры и представлен в разделах 3.2 и 3.3. Второй способ, который называют поэлементными операциями, рассматривается в разделе 3.4. Эти операции используют символы .*, ./ и .^ (перед стандартным символом операции ставится точка). Кроме того, в обоих типах вычислений, MATLAB имеет операторы деления (. \ или \), которые также обсуждаются в разделах 3.3 и 3.4.

Замечание новым пользователям MATLAB

Хотя матричные операции представлены вначале, а поэлементные операции потом, порядок может быть обращен, так как они независимы друг от друга. Предполагается, что почти каждый пользователь MATLAB имеет некоторые знания матричных операций и линейной алгебры и, таким образом, будет в состоянии без труда понять материал, представленный в разделах 3.2 и 3.3. Некоторые читатели, однако, могут предпочесть прочитать сначала раздел 3.4. MATLAB может использоваться с поэлементными операциями в многочисленных приложениях, которые не требуют операции умножения линейной алгебры (или деления).

3.1. Сложение и вычитание

Операции + (сложение) и – (вычитание) могут использоваться для сложения (вычитания) массивов одинакового размера (одно и то же число строк и столбцов) и прибавления (вычитания) скаляра к массиву. Когда два массива участвуют в сложении или вычитании, элементы массива результата получаются сложением, или вычитанием их соответствующих элементы.

Вообще, если A и B – два массива (например, 2×3 матрицы),

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \quad \text{и} \quad B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \end{bmatrix},$$

тогда матрица, которая получена сложением A и B есть:

$$A + B = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} & A_{13} + B_{13} \\ A_{21} + B_{21} & A_{22} + B_{22} & A_{23} + B_{23} \end{bmatrix}.$$

Примеры:

```
>> VectA=[8 5 4]; VectB=[10 2 7];      Задание двух векторов.
>> VectC=VectA+VectB
VectC =
    18          7          11
>> A=[5 -3 8; 9 2 10]                  Задание двух матриц А и В.
A =
    5          -3          8
    9           2          10
>> B=[10 7 4; -11 15 1]
B =
    10          7          4
   -11         15          1
>> A-B                                Вычитание матрицы В из матрицы А.
ans =
    -5         -10          4
    20         -13          9
>> C=A+B                                Сложение матриц А и В.
C =
    15          4          12
   -2         17          11
>> VectA+A                                Попытка сложения массивов разных размеров.
??? Error using ==> plus
Matrix dimensions must agree. Сообщение об ошибке.
>>
```

Когда скаляр (число) прибавляется к (или вычитается из) массиву, то скаляр прибавляется ко (или вычитается из) всем элементом массива. Примеры:

```

>> VectA=[1 5 8 -10 2]                                Задание вектора VectA.
VectA =
    1         5         8        -10         2
>> VectA+4                                         Прибавление скаляра 4 к VectA
ans =                                                 Число 4 прибавляется к каждому
    5         9        12        -6
>> A=[6 21 -15; 0 -4 8]                            Задание 2×3 матрицы A
A =
    6         21        -15
    0        -4          8
>> A-5                                         Вычитание скаляра 5 из A
ans =                                                 Число 5 вычитается из каждого элемента A
    1         16        -20
   -5        -9          3
>>

```

3.2. Умножение массивов

Операция умножения * выполняется MATLAB согласно правилам линейной алгебры. Это означает, что, если A и B – две матрицы, операция A^*B может быть выполнена, только если число столбцов у матрицы A равно числу строк у матрицы B . Результат – матрица, у которой число строк как у A и число столбцов – как у B . Например, если A – это 4×3 матрицы, и B – 3×2 матрицы:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{bmatrix} \quad \text{и} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{bmatrix},$$

тогда матрица, которая получена умножением A^*B имеет размеры 4×2 с элементами:

$$A^*B = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31} & A_{11}B_{12} + A_{12}B_{22} + A_{13}B_{32} \\ A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31} & A_{21}B_{12} + A_{22}B_{22} + A_{23}B_{32} \\ A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31} & A_{31}B_{12} + A_{32}B_{22} + A_{33}B_{32} \\ A_{41}B_{11} + A_{42}B_{21} + A_{43}B_{31} & A_{41}B_{12} + A_{42}B_{22} + A_{43}B_{32} \end{bmatrix}.$$

Числовой пример:

$$\begin{bmatrix} 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} 5 & 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 5 + 4 \cdot 1 + 3 \cdot 2 & 1 \cdot 4 + 4 \cdot 3 + 3 \cdot 6 \end{bmatrix} = \begin{bmatrix} 15 & 34 \end{bmatrix}.$$

$$\begin{bmatrix} 2 & 6 & 1 \end{bmatrix} \begin{bmatrix} 1 & 3 \end{bmatrix} = \begin{bmatrix} 2 \cdot 5 + 6 \cdot 1 + 1 \cdot 2 & 2 \cdot 4 + 6 \cdot 3 + 1 \cdot 6 \end{bmatrix} = \begin{bmatrix} 18 & 32 \end{bmatrix}.$$

$$\begin{bmatrix} 5 & 2 & 8 \end{bmatrix} \begin{bmatrix} 2 & 6 \end{bmatrix} = \begin{bmatrix} 5 \cdot 5 + 2 \cdot 1 + 8 \cdot 2 & 5 \cdot 4 + 2 \cdot 3 + 8 \cdot 6 \end{bmatrix} = \begin{bmatrix} 43 & 74 \end{bmatrix}.$$

Результат умножения двух квадратных матриц (они должны иметь один и тот же размер) является квадратной матрицей того же самого размера. Однако, умножение матриц не является коммутативным. Это означает это, если A и B – $n \times n$ матрицы, то вообще говоря, $A^*B \neq B^*A$. Кроме того, операция возведения в степень может быть выполнена только с квадратной матрицей (так как произведение A^*A может быть выполнено, только если число столбцов в первой матрице равно числу строк во второй матрице).

Два вектора могут быть перемножены, если они имеют одно и то же число элементов и один из них – вектор строки, а другой – вектор столбец. Умножение вектор-строки на вектор-столбец дает 1×1 матрицу, которая является скаляром. Это – скалярное произведение двух векторов. В MATLAB также есть встроенная функция, `dot(a, b)`, которая вычисляет скалярное произведение двух векторов. При использовании функции `dot` векторы `a` и `b` могут каждый быть вектором строкой или вектором столбцом независимо друг от друга (см. табл. 3.1). Умножение вектора столбца на вектор строку (каждый из n элементов) дает $n \times n$ матрицу. Умножение массива демонстрируется в учебной программе 3.1

Учебная программа 3.1. Умножение массивов

```
>> A=[1 4 2; 5 7 3; 9 1 6; 4 2 8]
A =
    1      4      2
    5      7      3
    9      1      6
    4      2      8
>> B=[6 1; 2 5; 7 3]
B =
    6      1
    2      5
    7      3
>> C=A*B
C =
    28      27
    65      49
    98      32
    84      38
>> D=B*A
??? Error using ==> *
Inner matrix dimensions must agree.
>> F=[1 3; 5 7]
F =
    1      3
    5      7
>> G=[4 2; 1 6]
G =
    4      2
    1      6
```

Задание 4×3 матрицы A.

Задание 3×2 матрицы B.

Умножение матрицы A на матрицу B.
и присвоение результата матрице C.

Попытка умножения B на A вызывает
ошибку, поскольку число 2 столбцов B
не совпадает с числом 4 строк A.

Задание 2×2 матриц F и G.

Учебная программа 3.1. Умножение массивов (окончание)

```

>> F*G
ans =
    7      20
   27      52
>> G*F
ans =
    14      26
   31      45
>> AV=[2 5 1]
AV =
    2      5      1
>> BV=[3; 1; 4]
BV =
    3
    1
    4
>> AV*BV
ans =
    15
>> BV*AV
ans =
    6      15      3
    2      5      1
    8      20      4
>>

```

Умножение матриц $F*G$.

Умножение матриц $G*F$.

Отметим, что $F*G$ не равно $G*F$.

Задание трехэлементного вектора AV .

Задание трехэлементного вектора столбца BV .

Умножение AV на BV . Ответ скаляр. Это скалярное (dot) произведение векторов.

Умножение BV на AV . Ответ – 3×3 матрица.

Когда массив умножается на число (фактически, число есть массив), то каждый элемент в массиве умножен на это число. Например:

```

>> A=[2 5 7 0; 10 1 3 4; 6 2 11 5]
A =
    2      5      7      0
   10     1      3      4
    6      2     11      5
>> b=3
b =
    3
>> b*A
ans =
    6      15     21      0
   30      3      9     12
   18      6     33     15
>> C=A*5
C =
   10     25     35      0
   50      5     15     20
   30     10     55     25
>>

```

Задание 3×4 матрицы A .

Присвоение переменной b значения 3.

Умножение матрицы A на b .
Это можно сделать либо так $b*A$, либо $A*b$.

Умножение матрицы A на 5 и
присвоение результата матрице C .
(Можно было и так: $5*A$).

Правила линейной алгебры умножения массивов дают удобный способ записи системы линейных уравнений. Например, систему трех уравнений с тремя неизвестными

$$A_{11}x_1 + A_{12}x_2 + A_{13}x_3 = B_1$$

$$A_{21}x_1 + A_{22}x_2 + A_{23}x_3 = B_2$$

$$A_{31}x_1 + A_{32}x_2 + A_{33}x_3 = B_3$$

можно записать в матричном виде:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}.$$

и в матричных обозначениях как:

$$A \cdot X = B,$$

где

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{и} \quad B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}.$$

3.3. Деление массивов

Операция *деления* также ассоциируется с правилами линейной алгебры. Эта операция является более сложной и ниже дано только ее краткое объяснение. Полное изложение может быть найдено в книгах по линейной алгебре. Операция деления может быть объяснена с помощью единичной матрицы и обратной операции.

Единичная матрица

Единичная матрица – это квадратная матрица, у которой диагональные элементы – единицы, а остальная часть элементов – нули. Как было показано в разделе 2.2.1, единичная матрица может быть создана в MATLAB командой `eye`. Когда единичная матрица умножается на другую матрицу (или вектор), то матрица (или вектор) не меняются (при умножении по правилам линейной алгебры). Это эквивалентно умножению скаляра на 1. Например:

$$\begin{bmatrix} 7 & 3 & 8 \\ 4 & 11 & 5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 3 & 8 \\ 4 & 11 & 5 \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{или} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 2 \\ 15 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \\ 15 \end{bmatrix}, \quad \text{или}$$

$$\begin{bmatrix} 6 & 2 & 9 \\ 1 & 8 & 3 \\ 7 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 2 & 9 \\ 1 & 8 & 3 \\ 7 & 4 & 5 \end{bmatrix}.$$

Если матрица A квадратная, то она может быть умножена на единичную матрицу и слева, и справа:

$$A \cdot I = I \cdot A = A.$$

Обратная матрица

Матрица B является *обратной* для матрицы A , если при их перемножении получается единичная матрица. Обе матрицы должны быть квадратными и порядок умножения может быть любой, $B \cdot A$, или $A \cdot B$.

$$B \cdot A = A \cdot B = I.$$

Очевидно, что если B – обратная для A , то A – обратная для B . Например:

$$\begin{bmatrix} 2 & 1 & 4 \\ 4 & 1 & 8 \\ 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 5.5 & -3.5 & 2 \\ 2 & -1 & 0 \\ -3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 5.5 & -3.5 & 2 \\ 2 & -1 & 0 \\ -3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 4 \\ 4 & 1 & 8 \\ 2 & -1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Обратная матрица для A обычно записывается в виде A^{-1} . В MATLAB обратная матрица может быть получена либо возведением в степень -1 , A^{-1} , или функцией `inv(A)`. Далее показано умножение в MATLAB вышеприведенных матриц.

```
>> A=[2 1 4; 4 1 8; 2 -1 3]
A =
    2      1      4
    4      1      8
    2     -1      3
>> B=inv(A)
B =
    5.5000    -3.5000     2.0000
    2.0000    -1.0000      0
   -3.0000     2.0000   -1.0000
>> A*B
ans =
    1      0      0
    0      1      0
    0      0      1
>> A*A^-1
ans =
    1      0      0
    0      1      0
    0      0      1
>>
```

Задание 3×3 матрицы A .

Использование функции `inv(A)` для нахождения обратной B для A .

Умножение A и B дает единичную матрицу.

Использование степени -1 для нахождения обратной к A . Ее перемножение с A дает единичную матрицу.

Не у каждой матрицы есть обратная. Матрица имеет обратную, только в том случае, когда она квадратная и ее определитель не равен нулю.

Определители

Определитель – это функция, соответствующая квадратным матрицам. Ниже дан краткий обзор по определителям. Более подробное изложение см. в книгах по линейной алгебре.

Определитель – это функция, которая каждой квадратной матрицей ставит в соответствие число, называемое определителем матрицы. Определитель обычно обозначается либо $\det(A)$, либо $|A|$. Определитель вычисляется согласно определенным правилам. Для матрицы второго порядка это правило следующее:

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}, \text{ например, } \begin{vmatrix} 6 & 5 \\ 3 & 9 \end{vmatrix} = 6 \cdot 9 - 5 \cdot 3.$$

Определитель квадратной матрицы может быть вычислен командой `det` (см. табл. 3.1).

Деление массивов

MATLAB имеет два типа деления массивов, правое деление и левое деление.

Левое деление (\backslash)

Левое деление используется для решения матричного уравнения $A \cdot X = B$. В этом уравнении X и B векторы столбцы. Это уравнение может быть решено, умножением слева обеих сторон на обратную к матрице A :

$$A^{-1} \cdot A \cdot X = A^{-1} \cdot B.$$

Левая часть этого уравнения есть X , поскольку

$$A^{-1} \cdot A \cdot X = I \cdot X = X.$$

Поэтому решение уравнения $A \cdot X = B$ есть

$$X = A^{-1} \cdot B.$$

В MATLAB последнее равенство может быть записано с использованием символа левого деления:

$$X = A \backslash B.$$

Нужно заметить, что хотя две последние операции, кажется, дают тот же самый результат, но метод, которым MATLAB вычисляет X , различный. В первом случае MATLAB вычисляет обратную матрицу A^{-1} и затем использует ее для умножения на B . Во втором случае (левое деление), решение X получается численно на основе метода исключения Гаусса. Левое деление рекомендуется для решения системы

линейных уравнений, потому что вычисление обратной матрицы может быть менее точным чем метод исключения Гаусса при использовании больших матриц.

Правое деление (/)

Правое деление используется для решения матричного уравнения $X \cdot C = D$. В этом уравнении X и D – векторы строки. Это уравнение может быть решено умножением справа, обеих стороны уравнения на обратную к C :

$$X \cdot C \cdot C^{-1} = D \cdot C^{-1},$$

которое дает

$$X = D \cdot C^{-1}.$$

В MATLAB последнее уравнение может быть записано с использованием символа правого деления:

$$X = D/C.$$

Следующий пример демонстрирует использование левого и правого деления, и функцию `inv` для решения системы линейных уравнений.

Пример задачи 3.1. Решение трех линейных уравнений (деление массивов)

Используя матричные операции решить следующую систему линейных уравнений.

$$4x - 2y + 6z = 8$$

$$2x + 8y + 2z = 4$$

$$6x + 10y + 3z = 0$$

Решение

Используя упомянутые выше правила линейной алгебры, данная система уравнений может быть записана в матричной форме $A \cdot X = B$ или в форме $X \cdot C = D$:

$$\begin{bmatrix} 4 & -2 & 6 \\ 2 & 8 & 2 \\ 6 & 10 & 3 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix}, \text{ или } \begin{bmatrix} x & y & z \end{bmatrix} \cdot \begin{bmatrix} 4 & 2 & 6 \\ -2 & 8 & 10 \\ 6 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 8 & 4 & 0 \end{bmatrix}.$$

Решения для обеих форм показаны ниже:

```
>> A=[4 -2 6; 2 8 2; 6 10 3];
>> B=[8; 4; 0];
>> X=A\B
X =
    -1.8049
    0.2927
```

Решение для вида: $AX = B$.

Решение с использованием левого деления: $X = A \setminus B$.



```

2.6341
>> Xb=inv(A) *B
Xb =
-1.8049
0.2927
2.6341
>>

```

Решение с обратной для А матрицей: $X = A^{-1}B$.

```

>> C=[4 2 6; -2 8 10; 6 2 3];
>> D=[8 4 0];
>> Xc=D/C
Xc =
-1.8049      0.2927      2.6341
>> Xd=D*inv(C)
Xd =
-1.8049      0.2927      2.6341
>>

```

Решение для вида: $X C = D$.

Решение с использованием правого деления: $X = D/C$.

Решение с обратной для С матрицей: $X = DC^{-1}$.

3.4. Поэлементные операции

В разделах 3.2 и 3.3 было показано, что когда используются регулярные символы ($*$ и $/$) для умножения и деления массивов, то математические операции следуют правилам линейной алгебры. Однако есть много ситуаций, которые требуют поэлементных операций. Эти операции выполняются отдельно на каждом из элементов массива (или массивов). Сложение и вычитание по определению уже поэлементно операции, когда два массива складываются (или вычтываются), операция выполняется с элементами, которые находятся в массивах в одной и той же позиции. Поэлементно операции могут быть сделаны только с массивами одного и того же размера.

Поэлементное умножение, деление, или возвведение в степень двух векторов или матриц используется в MATLAB введением точки перед арифметическим оператором.

Символ	Описание	Символ	Описание
$\cdot *$	Умножение	$\cdot /$	Правое деление
$\cdot ^$	Возвведение в степень	$\cdot \backslash$	Левое деление

Если имеются два вектора a и b , $a = [a_1, a_2, a_3, a_4]$ и $b = [b_1, b_2, b_3, b_4]$, то поэлементно умножение, деление, и возведение в степень этих двух векторов дают:

$$a \cdot * b = [a_1 b_1, a_2 b_2, a_3 b_3, a_4 b_4],$$

$$a./b = [a_1/b_1, a_2/b_2, a_3/b_3, a_4/b_4],$$

$$a.^{\wedge} b = [(a_1)^{b_1}, (a_2)^{b_2}, (a_3)^{b_3}, (a_4)^{b_4}].$$

Если даны матрицы A и B

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \quad \text{и} \quad B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix},$$

то поэлементное умножение и деление этих двух матриц выглядит так:

$$A.*B = \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} & A_{13}B_{13} \\ A_{21}B_{21} & A_{22}B_{22} & A_{23}B_{23} \\ A_{31}B_{31} & A_{32}B_{32} & A_{33}B_{33} \end{bmatrix}, \quad A./B = \begin{bmatrix} A_{11}/B_{11} & A_{12}/B_{12} & A_{13}/B_{13} \\ A_{21}/B_{21} & A_{22}/B_{22} & A_{23}/B_{23} \\ A_{31}/B_{31} & A_{32}/B_{32} & A_{33}/B_{33} \end{bmatrix}.$$

Поэлементное возвведение в степень матрицы A :

$$A.^n = \begin{bmatrix} (A_{11})^n & (A_{12})^n & (A_{13})^n \\ (A_{21})^n & (A_{22})^n & (A_{23})^n \\ (A_{31})^n & (A_{32})^n & (A_{33})^n \end{bmatrix}.$$

Поэлементное умножение, деление, и возвведение в степень демонстрируются в учебной программе 3.2.

Учебная программа 3.2. Поэлементные операции

```
>> A=[2 6 3; 5 8 4]                                Задание 2×3 массива A.
A =
    2       6       3
    5       8       4
>> B=[1 4 10; 3 2 7]                               Задание 2×3 массива B.
B =
    1       4      10
    3       2       7
>> A.*B                                         Поэлементное умножение массива A и B.
ans =
    2       24      30
    15      16      28
>> C=A./B                                         Поэлементное деление массива A и B.
C =
    2.0000      1.5000      0.3000
    1.6667      4.0000      0.5714
>> B.^3                                         Поэлементное возведение в степень массива B.
ans =                                              Каждый элемент возводится в степень 3. ➔
```

Учебная программа 3.2. Поэлементные операции (окончание)

```

1      64      1000
27      8      343
>> A*B
??? Error using ==> *
Inner matrix dimensions must agree.
>>

```

Попытка матричного произведения $A \cdot B$
вызывает ошибку, поскольку число
столбцов A не равно числу строк B .

Поэлементные вычисления очень полезны для вычисления значений функции на многих значениях ее аргумента. Сначала создаем вектор, который содержит значения независимой переменной, а затем используем этот вектор в поэлементных вычислениях для создания вектора, у которого каждый элемент есть соответствующее значение функции. Один пример:

```

>> x=[1:8]                                Создание вектора x с 8 элементами.
x =
    1      2      3      4      5      6      7      8
>> y=x.^2-4*x
y=
    -3      -4      -3      0      5     12     21     32
>>

```

В этом примере $y = x^2 - 4x$. Поэлементно операция необходима при возведении x в квадрат. Каждый элемент в векторе y является значением, которое получается, когда значение соответствующего элемента вектора x подставляется в уравнение. Другой пример:

```

>> z=[1:2:11]                                Создание вектора z с 7 элементами.
z =
    1      3      5      7      9      11
>> y=(z.^3 + 5*z) ./ (4*z.^2 - 10)        Вектор z используется в поэлементном
y =                                         вычислении элементов вектора y.
    -1.0000    1.6154    1.6667    2.0323    2.4650    2.9241
>>

```

В последнем примере $y = \frac{z^3 + 5z}{4z^2 - 10}$. Поэлементные операции используются в этом примере три раза: при вычислении z^3 и z^2 и при делении числителя на знаменатель.

3.5. Использование массивов во встроенных математических функциях MATLAB

Встроенные функции в MATLAB записаны так, что, когда аргумент (ввод) является массивом, то операция, которая определена функцией, выполняется на каждом

элементе массива. (Можно считать эту операцию как поэлементное применение функции.) Результатом (выводом) такой операции является массив, каждый элемент которого вычисляется введением соответствующего элемента массива аргумента в функцию. Например, если вектор x с семью элементами подставляют в функцию $\cos(x)$, то результат – это вектор с семью элементами, в котором каждый элемент – косинус соответствующего элемента в x . Это показано ниже.

```
>> x=[0:pi/6:pi]
x =
    0    0.5236   1.0472   1.5708   2.0944   2.6180   3.1416
>>y=cos(x)
y =
    1.0000    0.8660    0.5000    0.0000   -0.5000   -0.8660   -1.0000
>>
```

Пример, в котором переменная в аргументе – матрица:

```
>> d=[1 4 9; 16 25 36; 49 64 81]                                Создание 3×3 массива.
d =
    1      4      9
    16     25     36
    49     64     81
>> h=sqrt(d)                                                 h есть 3×3 массив, в котором каждый элемент
h =                                                       есть корень квадратный соответствующего
    1      2      3                                         элемента массива d.
    4      5      6
    7      8      9
>>
```

Это свойство MATLAB, когда массивы могут использоваться в качестве аргументов в функциях, называется векторизацией.

3.6. Встроенные функции для анализа массивов

У MATLAB есть много встроенных функций для анализа массивов. В табл. 3.1 приведен список некоторых из этих функций.

Таблица 3.1. Функции массивов

Функция	Описание	Пример
mean (A)	Если A вектор, возвращает среднее значение элементов вектора.	<pre>>> A=[5 9 2 4]; >> mean (A) ans = 5</pre>

Функция	Описание	Пример
C=max (A)	Если A – вектор, то C – самый большой элемент в A. Если A – матрица, то C –вектор строки из самых больших элементов каждого столбца A.	>> A=[5 9 2 4 11 6 11 1]; >> C=max (A) C = 11
[d, n]=max (A)	Если A – вектор, то d – самый большой элемент в A и n – позиция элемента (первая, если есть несколько максимальных значений).	>> [d, n]=max (A) d = 11 n = 5
min (A)	То же самое, что и max (A) , но для наименьшего элемента.	>> A=[5 9 2 4]; >> min (A) ans = 2
[d, n]=min (A)	То же самое, что и [d, n]=max (A) , но для наименьшего элемента.	
sum (A)	Если A вектор, возвращает сумму элементов вектора.	>> A=[5 9 2 4]; >> sum (A) ans = 20
sort (A)	Если A – вектор, располагает элементы вектора в порядке возрастания.	>> A=[5 9 2 4]; >> sort (A) ans = 2 4 5 9
median (A)	Если A вектор, возвращает медианное значение элементов вектора.	>> A=[5 9 2 4]; >> median (A) ans = 4.5000
std (A)	Если A вектор, возвращает стандартное отклонение элементов вектора.	>> A=[5 9 2 4]; >> std (A) ans = 2.9439
det (A)	Возвращает определитель матрицы A.	>> A=[2 4; 3 5]; >> det (A) ans = -2
dot (a,b)	Вычисляет скалярное (dot) произведение двух векторов a и b. Каждые вектор может быть строкой или столбцом.	>> A=[5 9 2 4]; >> min (A) ans = 2

Функция	Описание	Пример
cross (a, b)	Вычисляет векторное произведение двух векторов a и b, ($a \times b$). Каждый из этих векторов должен иметь три элемента.	>> a=[1 3 2]; >> b=[2 4 1]; >> cross(a,b) ans = -5 3 -2
inv (A)	Возвращает обратную для квадратной матрицы A.	>> A = [2 -2 1; 3 2 -1; 2 -3 2]; >> inv (A) ans = 0.2000 0.2000 0 -1.6000 0.4000 1.0000 -2.6000 0.4000 2.0000

3.7. Генерация случайных чисел

Моделирование многих физических процессов и технических приложений часто требует использования чисел (или ряда чисел) со *случайными* значениями. MATLAB имеет есть три команды – `rand`, `randn` и `randi` – которые могут использоваться для присвоения случайных чисел переменным.

Команда `rand`

Команда `rand` генерирует равномерно распределенные случайные числа со значениями между 0 и 1. Команда может использоваться для присвоения этих чисел скаляру, вектору, или матрице, как показано в табл. 3.2.

Таблица 3.2. Команда `rand`

Команда	Описание	Пример
<code>rand</code>	Генерирует одно случайное число между 0 и 1.	>> rand ans = 0.2311
<code>rand(1, n)</code>	Генерирует n-элементный вектор-строку случайных чисел между 0 и 1.	>> a=rand(1, 4) a = 0.6068 0.4860 0.8913 0.7621
<code>rand(n)</code>	Генерирует n×n матрицу случайных чисел между 0 и 1.	>> b=rand(3) b = 0.4565 0.4447 0.9218 0.0185 0.6154 0.7382 0.8214 0.7919 0.1763

Команда	Описание	Пример
<code>rand(m, n)</code>	Генерирует $m \times n$ матрицу случайных чисел между 0 и 1.	<pre>>> c=rand(2, 4) c = 0.4057 0.9169 0.8936 0.3529 0.9355 0.4103 0.0579 0.8132</pre>
<code>randperm(n)</code>	Генерирует вектор-строку из n элементов, которые образуют случайную перестановку целых чисел от 1 до n .	<pre>>> randperm(8) ans = 8 2 7 4 3 6 5 1</pre>

Иногда есть потребность в случайных числах, которые распределены на интервале отличном от $(0,1)$, или для чисел, которые являются только целыми. Это может быть сделано, используя математические операции с функцией `rand`. Случайные числа, которые распределены на интервале (a, b) можно получить умножая `rand` на $(b - a)$ и прибавляя произведение к a :

$$(b - a) * \text{rand} + a$$

Например, вектор из 10 элементов со случайными значениями между -5 и 10 может быть создан так ($a = -5$, $b = 10$):

```
>> v=15*rand(1,10)-5
v =
    -1.8640    0.6973    6.7499    5.2127    1.9164    3.5174
    6.9132   -4.1123    4.0430   -4.2460
>>
```

Команда `randi`

Команда `randi` генерирует равномерно распределенные случайные целые числа. Команда может использоваться для присвоения этих чисел скаляру, вектору, или матрице, как показано в табл. 3.3.

Таблица 3.3. Команда `randi`

Команда	Описание	Пример
<code>randi(imax)</code> (imax is an integer)	Генерирует одно случайное число между 0 и imax .	<pre>>> a=randi(15) a = 9</pre>
<code>randi(imax, n)</code>	Генерирует $n \times n$ матрицу случайных чисел между 0 и imax .	<pre>>> b=randi(15, 3) b = 4 8 11 14 3 8 1 15 8</pre>

Команда	Описание	Пример
<code>randi(imax, m, n)</code>	Генерирует $m \times n$ матрицу случайных чисел между 0 и $imax$.	<pre>>> c=randi(15, 2, 4) c = 1 1 8 13 11 2 2 13</pre>

Диапазон случайных целых чисел может быть установлен между любыми двумя целыми числами, для этого нужно ввести `[imin imax]` вместо `imax`. Например, матрица 3×4 со случайными целыми числами между 50 и 90 создается так:

```
>> d=randi([50 90], 3, 4)
d =
    57      82      71      75
    66      52      67      61
    84      66      76      67
```

Команда `randn`

Команда `randn` обычно генерирует нормально распределенные случайные числа со средним значением 0 и стандартным отклонением 1. Команда может использоваться для создания одного числа, вектора, или матрицы таким же образом как команда `rand`. Например, создание матрицы:

```
>> d=randn(3, 4)
d =
    -0.4326      0.2877      1.1892      0.1746
    -1.6656     -1.1465     -0.0376     -0.1867
     0.1253      1.1909      0.3273      0.7258
```

Среднее значение и стандартное отклонение чисел могут быть изменены математическими операциями для того, чтобы иметь любые значения. Это делается умножением числа, сгенерированного функцией `randn` на требуемое стандартное отклонение и прибавляя требуемое среднее значение. Например, вектор шести чисел со средним значением 50 и стандартным отклонением 6 генерируется так:

```
>> v=4*randn(1, 6)+50
v =
    42.7785    57.4344    47.5819    50.4134    52.2527    50.4544
```

Целые нормально распределенные случайные числа могут быть получены при использовании функции округления.

```
>> w=round(4*randn(1, 6)+50)
w =
    51      49      46      49      50      44
```

3.8. Примеры приложений MATLAB

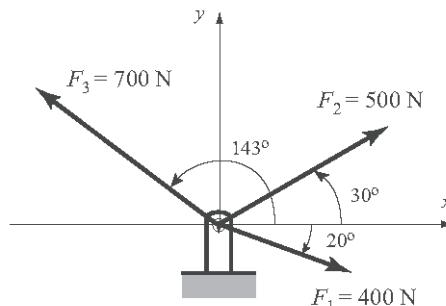
Пример задачи 3.2. Эквивалентная система сил (сложение векторов)

Три силы приложены к скобе как показано на рисунке.

Найти полную (результатирующую) силу приложенную к скобе.

Решение

Сила – это вектор (физическая величина, у которой есть величина и направление). В декартовой системе координат двумерный вектор \mathbf{F} может быть записан как:



$$\mathbf{F} = F_x \mathbf{i} + F_y \mathbf{j} = F \cos(\theta) \mathbf{i} + F \sin(\theta) \mathbf{j} = F (\cos(\theta) \mathbf{i} + \sin(\theta) \mathbf{j})$$

где F – это величина силы, θ – ее угол относительно оси x , F_x и F_y – это компоненты силы \mathbf{F} в направлении осей x и y , соответственно, и \mathbf{i}, \mathbf{j} – единичные векторы в этих направлениях. Если F_x и F_y известны, то F и θ могут быть определены по формулам:

$$F = \sqrt{F_x^2 + F_y^2} \quad \text{и} \quad \tan \theta = \frac{F_y}{F_x}.$$

Результирующая (эквивалентная) сила, примененная к скобе, получается сложением сил, которые действуют на скобу. Решение задачи с MATLAB проводится в три шага:

- Записать каждую силу как вектор с двумя элементами, где первый элемент – это x -компоненты вектора, а второй элемент – y -компоненты.
 - Определить векторную форму эквивалентной силы, складывая векторы.
 - Определить величину и направление эквивалентной силы.

Задача решена в следующем файле сценария.

% Sample Problem 3-2 solution (script file)

clear

F1M=400; F2M=500; F3M=700;

Tb1=-30 : Tb2=30 : Tb3=143 :

E1-E1M⁺ [sind (Tb1), sind (Tb1)]

$$E1 = E1M^* [\cos(\theta_1) \quad \sin(\theta_1)]$$

$$E2 = E2M^* [\cos(\theta_2) \quad \sin(\theta_2)]$$

$F2-F2M^+ [\cos(\text{fHz}) \quad \sin(\text{fHz})]$

$$F_S = F_{SM} \times [\cos\alpha]$$

Задание переменный – величин каждого вектора.

Задание углов направлений каждого вектора.

Определение трех векторов-

Вычисление результирующего вектора

```
FtotM=sqrt(Ftot(1)^2+Ftot(2)^2)
```

Вычисление величины

результатирующего вектора.

```
Th=atand(Ftot(2)/Ftot(1))
```

Вычисление угла направления

результатирующего вектора.

После выполнения программы на экран в командном окне выводится следующее:

```
F1 =
375.8770 -136.8081
F2 =
433.0127 250.0000
F3 =
-559.0449 421.2705
Ftot =
249.8449 534.4625
FtotM =
589.9768
Th =
64.9453
```

Компоненты вектора F_1 .

Компоненты вектора F_2 .

Компоненты вектора F_3 .

Компоненты результирующего вектора.

Величина результирующего вектора.

Направление результирующего вектора.

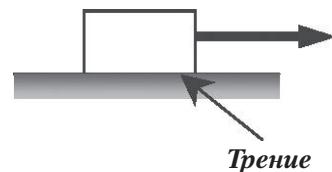
Результирующая сила имеет величину 589.98 N и направлена под углом 54.95° (против часовой стрелки) относительно оси x. В векторном виде сила есть $F = 249.84 \mathbf{i} + 534.46 \mathbf{j}$ N.

Пример задачи 3.3. Эксперимент с трением (поэлементные вычисления)

Коэффициент трения μ , может быть определен экспериментально измерением силы F необходимой для перемещения массы m . Когда F измерена и m известна, коэффициент трения может быть вычислен по формуле:

$$\mu = F/(m g), \quad (g = 9.81 \text{ m/s}^2)$$

Результаты измерения F в шести испытаниях приведены в таблице ниже. Определить коэффициент трения в каждом испытании и среднее значение при всех испытаниях.



Испытание	1	2	3	4	5	6
Масса m (кг)	2	4	5	10	20	50
Сила F (N)	12.5	23.5	30	61	117	294

Решение

Решение в командном окне MATLAB показано ниже.

```
>> m=[2 4 5 10 20 50];           Вводим значения вектора m
>> F=[12.5 23.5 30 61 117 294];   Вводим значения вектора F
>> mu=F./ (m^9.81)             Значение mu вычисляется для каждого испытания,
                                используя поэлементные вычисления.

mu =
0.6371    0.5989    0.6116    0.6218    0.5963    0.5994

>> mu_ave=mean(mu)            Среднее значение элементов вектора mu
mu_ave =                      определяется с использованием функции mean.
0.6109
```

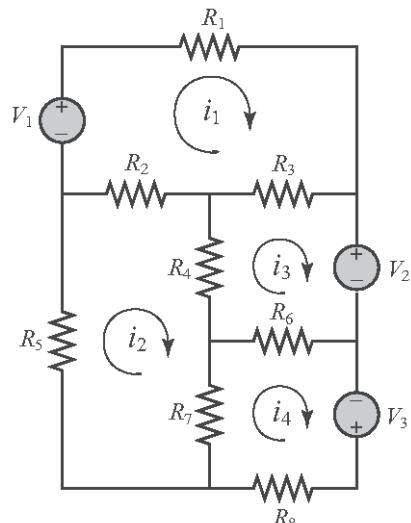
Пример задачи 3.4: Анализ электрической резистивной сети (решение систем линейных уравнений)

Показанная на рисунке электрическая цепь состоит из резисторов и источников напряжения. Определить ток в каждом резисторе, используя метод контурного тока, который основан на законе напряжений Кирхгофа.

$$\begin{aligned} V_1 &= 20 \text{ В}, \quad V_2 = 12 \text{ В}, \quad V_3 = 40 \text{ В} \\ R_1 &= 18 \Omega, \quad R_2 = 10 \Omega, \quad R_3 = 16 \Omega, \\ R_4 &= 6 \Omega, \quad R_5 = 15 \Omega, \quad R_6 = 8 \Omega, \\ R_7 &= 12 \Omega, \quad R_8 = 14 \Omega. \end{aligned}$$

Решение

Закон напряжений Кирхгофа утверждает, что сумма напряжений по замкнутой цепи равна нулю. В методе контурного тока ток сначала определяется для каждой петли (i_1, i_2, i_3, i_4 на рисунке). Затем закон напряжений Кирхгофа применяется для каждой петли. Это приводит к системе линейных уравнений для токов (в данном случае четыре уравнения). Решение дает значения контурных токов. Ток в резисторе, который принадлежит двум петлям, является суммой токов в соответствующих петлях. Удобно предположить, что все токи проходят в одном направлении (по часовой стрелке в данном случае). В уравнении для каждой петли источник напряжения положителен, если электрический ток течет к полюсу «+» и напряжение резистора отрицательно для тока в направлении тока контура.



Уравнения для четырех контуров в задаче тока:

$$\begin{aligned} V_1 - R_1 i_1 - R_3(i_1 - i_3) - R_2(i_1 - i_2) &= 0 \\ -R_5 i_2 - R_2(i_2 - i_1) - R_4(i_2 - i_3) - R_7(i_2 - i_4) &= 0 \\ -V_2 - R_6(i_3 - i_4) - R_4(i_3 - i_2) - R_3(i_3 - i_1) &= 0 \\ V_3 - R_8 i_4 - R_7(i_4 - i_2) - R_6(i_4 - i_3) &= 0 \end{aligned}$$

Эти четыре уравнения могут быть переписаны в матричной форме $[A][x] = [B]$:

$$\left[\begin{array}{ccc|c} -(R_1 + R_2 + R_3) & R_2 & R_3 & 0 \\ R_2 & -(R_2 + R_4 + R_5 + R_7) & R_4 & R_7 \\ R_3 & R_4 & -(R_3 + R_4 + R_6) & R_6 \\ 0 & R_7 & R_6 & -(R_6 + R_7 + R_8) \end{array} \right] \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} -V_1 \\ 0 \\ V_2 \\ -V_3 \end{bmatrix}$$

Задача решена в следующей программе, записанной в файле сценария:

```
V1=20; V2=12; V3=40;          Определение переменных со значениями Vs и Rs.
R1=18; R2=10; R3=16; R4=6;
R5=15; R6=8; R7=12; R8=14;
A=[-(R1+R2+R3) R2 R3 0      Создание матрицы A.
R2 -(R2+R4+R5+R7) R4 R7
R3 R4 -(R3+R4+R6) R6
0 R7 R6 -(R6+R7+R8) ]        Создание вектора B.
>> B=[-V1; 0; V2; -V3]       Решение для токов с использованием левого деления.
```

После выполнения файл сценария на экран в командном окне выводится следующее:

A =	Числовые значения элементов матрицы A.			
- 44 10 16 0				
10 - 43 6 12				
16 6 - 30 8				
0 12 8 - 34				
B =	Числовые значения элементов матрицы B.			
-20				
0				
12				
-40				
I =				
0.8411	\leftarrow	$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}$		
0.7206	\leftarrow	$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}$		
0.6127	\leftarrow	$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}$		
1.5750	\leftarrow	$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}$		
>>	Решение.			

Последний вектор столбец дает ток в каждом контуре. Токи в резисторах R_1 , R_5 и R_8 есть $i_1 = 0.8411$ А, $i_2 = 0.7206$ А, и $i_4 = 1.5750$ А, соответственно. Другие резисторы принадлежат двум петлям, и их токи – это суммы токов в контурах.

Ток в резисторе R_2 есть является $i_1 - i_2 = 0.1205$ А.

Ток в резисторе R_3 есть является $i_1 - i_3 = 0.2284$ А.

Ток в резисторе R_4 есть является $i_2 - i_3 = 0.1079$ А.

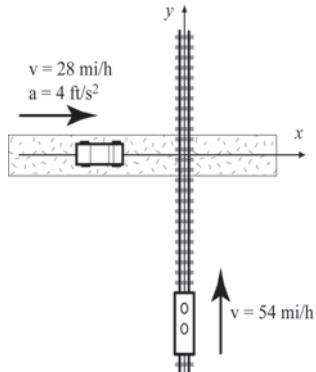
Ток в резисторе R_6 есть является $i_4 - i_3 = 0.9623$ А.

Ток в резисторе R_7 есть является $i_4 - i_2 = 0.8544$ А.

Пример задачи 3.5. Движение двух материальных тел

Поезд и автомобиль приближаются к дорожному пересечению. В данное время поезд находится в 400 футах к югу от пересечения и движется на север с постоянной скоростью 54 миль/час. В то же самое время автомобиль находится в 200 футах к западу от пересечения и движется на восток со скоростью 28 миль/час с ускорением в 4 фута/сек². Определить положение поезда и автомобиля, расстояние между ними, и скорость поезда относительно автомобиля каждую секунду в течение следующих 10 секунд.

Чтобы показать результаты, создать матрицу, в которой в первом столбце стоит время, а в следующих пяти столбцах – положение поезда, положение автомобиля, расстояние между поездом и автомобилем, скорость автомобиля и скорость поезда относительно автомобиля, соответственно.



Решение

Положение объекта, который движется по прямой с постоянном ускорением задается уравнением $s = s_0 + v_0 t + \frac{1}{2} a t^2$, где s_0 и v_0 – положение и скорость при $t = 0$ и a – ускорение. Применение этого уравнения к поезду и автомобилю дает:

$$y = -400 + v_{0\text{поезд}} t \quad (\text{поезд}),$$

$$y = -200 + v_{0\text{авто}} t + \frac{1}{2} a_{\text{авто}} t^2 \quad (\text{автомобиль}).$$

Расстояние между автомобилем и поездом есть: $d = \sqrt{x^2 + y^2}$. Скорость поезд является постоянной и в векторной системе обозначений: $\mathbf{v}_{\text{поезд}} = \mathbf{v}_{0\text{поезд}}$. Автомобиль ускоряется и его скорость задается вектором: $\mathbf{v}_{\text{авто}} = (v_{0\text{авто}} + a_{\text{авто}} t) \mathbf{i}$. Скорость поезда относительно автомобиля, $\mathbf{v}_{n/a}$, есть $\mathbf{v}_{n/a} = \mathbf{v}_{\text{поезд}} - \mathbf{v}_{\text{авто}} = -(v_{0\text{авто}} + a_{\text{авто}} t) \mathbf{i} + \mathbf{v}_{0\text{поезд}}$. Величина этой скорости – это длина данного вектора.

Задача решена в следующей программе, записанной в файле сценария. Сначала создается вектор t с 11 элементами – моментами времени от 0 до 10 сек, потом для каждого момента вычисляются положения поезда и автомобиля, расстояния между ними и скорость поезда относительно автомобиля.

```

v0train=54*5280/3600; v0car=28*5280/3600; acar=4;
    Создание переменных для скоростей и ускорения.
t=0:10;                                Создание вектора t.
y=-400+v0train*t;                      Вычислите положения поезда и автомобиля.
x=-200+v0car*t+0.5*acar*t.^2;          Вычисление расстояния между поездом и автомобилем.
d=sqrt(x.^2+y.^2);                      Вычисление скорости автомобиля.
vcar=v0car+acar*t;                      Вычисление скорости поезда относительно автомобиля.
speed_trainRcar=sqrt(vcar.^2+v0train.^2);
table=[t' y' x' d' vcar' speed_trainRcar'];
>>                                         Вычисление таблицы (см. ниже).

```

Замечание: В командах выше `table` – это имя переменной, которая является матрицей, содержащей данные, которые будут выведены на экран.

После выполнения файла сценария, на экран в командном окне выводится следующее:

	table =				
0	-400.0000 -200.0000 447.2136 41.0667 89.2139				
1.0000	-320.8000 -156.9333 357.1284 45.0667 91.1243				
2.0000	-241.6000 -109.8667 265.4077 49.0667 93.1675				
3.0000	-162.4000 -58.8000 172.7171 53.0667 95.3347				
4.0000	-83.2000 -3.7333 83.2837 57.0667 97.6178				
5.0000	-4.0000 55.3333 55.4777 61.0667 100.0089				
6.0000	75.2000 118.4000 140.2626 65.0667 102.5003				
7.0000	154.4000 185.4667 241.3239 69.0667 105.0849				
8.0000	233.6000 256.5333 346.9558 73.0667 107.7561				
9.0000	312.8000 331.6000 455.8535 77.0667 110.5075				
10.0000	392.0000 410.6667 567.7245 81.0667 113.3333				
Время (сек)	Положение поезда (фут)	Положение автомобиля (фут)	Расстояние поезд – автомобиль (фут)	Скорость автомобиля (фут/сек)	Скорость поезда относительно автомобиля (фут/сек)
>>					

В этой задаче результаты (числа) выведены на экран MATLAB без какого-либо текста. В главе 4 даются рекомендации, как добавить текст к результату, полученному MATLAB.

3.9. Задачи

Замечание: В конце главы 4 представлены дополнительные задачи для того, чтобы попрактиковаться с математическими операциями с массивами.

1. Для функции $y = x^3 - e^{0.5x} + x$ вычислить значение y для следующих значений x используя поэлементно операции: $-3, -2, -1, 0, 1, 2, 3$.

2. Для функции $y = \frac{(x+5)^3}{x^2}$ вычислить значение y для следующих значений x используя поэлементно операции: $1, 2, 3, 4, 5, 6$.

3. Для функции $y = \frac{(x+7)^4}{(x+1)\sqrt{x}}$ вычислить значение y для следующих значений x используя поэлементно операции: $1.5, 2.5, 3.5, 4.5, 5.5, 6.5$.

4. Для функции $y = \frac{2 \sin x + \cos^2 x}{\sin^2 x}$ вычислить значение y для следующих значений x используя поэлементно операции: $20^\circ, 30^\circ, 40^\circ, 50^\circ, 60^\circ, 70^\circ$.

5. Радиус r сферы может быть вычислен от ее площади s по формуле: $r = \frac{\sqrt{s/\pi}}{2}$.
Объем V шара радиуса r вычисляется по формуле: $V = \frac{4\pi r^3}{3}$.

Определить объемы сфер с площадями поверхности 50, 100, 150, 200, 250 и 300 фут². Вывести на экран результаты в таблице с двумя столбцами со значениями s и V в первом и втором столбцах, соответственно.

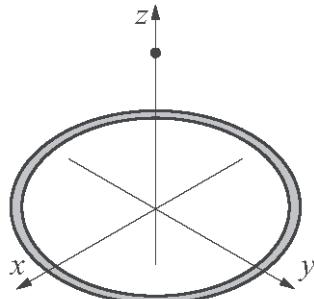
6. Интенсивность $E(z)$ электрического поля, определенного кольцом радиуса R в любой точке z вдоль оси кольца задается формулой:

$$E(z) = \frac{\lambda}{2\epsilon_0} \frac{Rz}{(z^2 + R^2)^{3/2}},$$

где λ – плотность заряда, $\epsilon_0 = 8.85 \times 10^{-12}$ – электрическая постоянная и R – радиус кольца.
Рассмотреть случай, когда $\lambda = 1.7 \times 10^{-7}$ Кл/м и $R = 10$ см.

(a) Определить $E(z)$ при $z = 0, 2, 4, 6, 8$ и 10 см.

(b) Определите расстояние z , где E максимальна. Выполнить это созданием вектора z с элементами в пределах от 2 см до 6 см с шагом 0.01 см. Вычислить E для каждого значения z и затем найти максимум E и соответствующее значение z с использованием встроенной функции MATLAB `max`.

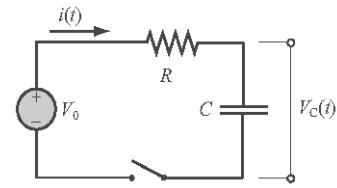


7. Напряжение $V_C(t)$ (в вольтах) и ток $i(t)$ (в амперах) спустя t секунд после замыкания переключателя на показанной справа цепи задается формулами:

$$V_C(t) = V_0(1 - e^{-t/\tau_0}) \quad \text{и} \quad i(t) = \frac{V_0}{R} e^{-t/\tau_0},$$

где $\tau_0 = RC$ – постоянная (по времени).

Рассмотреть случай где $V_0 = 24 \text{ В}$, $R = 3800 \Omega$ и $C = 4000 \times 10^{-6} \Phi$. Определить напряжение и ток в течение первых 20 сек после замыкания переключателя. Создать вектор со значениями моментов времени от 0 до 20 сек шагом 2 сек и использовать его для вычисления $V_c(t)$ и $i(t)$. Вывести на экран результаты в таблице с тремя столбцами со значениями времени, напряжения и тока в первом, втором и третьем столбцах, соответственно.



8. Длина $|\mathbf{u}|$ (величина) вектора $\mathbf{u} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ задается формулой $|\mathbf{u}| = \sqrt{x^2 + y^2 + z^2}$.

Для данного вектора $\mathbf{u} = 23.5\mathbf{i} - 17\mathbf{j} + 6\mathbf{k}$, определить длину следующими двумя способами:

- Определите вектор в MATLAB и затем записать математическое выражение длины, которое использует компоненты вектора.
- Определите вектор в MATLAB, затем определите длину одной командой, которая использует поэлементные операции и встроенные функции MATLAB `sum` и `sqrt`.

9. Вектор \mathbf{w}_L длины L в направлении вектора $\mathbf{u} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ может быть определен формулой $\mathbf{w}_L = L\mathbf{u}_n$ (умножение единичного вектора в направлении \mathbf{u} на L). Единичный вектор \mathbf{u}_n в направлении вектора \mathbf{u} задается формулой

$$\mathbf{u}_n = \frac{x\mathbf{i} + y\mathbf{j} + z\mathbf{k}}{\sqrt{x^2 + y^2 + z^2}}. \text{ Одной командой MATLAB, определить вектор длины 18}$$

в направлении вектора $\mathbf{u} = 7\mathbf{i} - 4\mathbf{j} - 11\mathbf{k}$.

10. В MATLAB определены следующие два вектора: $v = [15, 8, -6]$, $u = [15, 8, -6]$.

Вручную (карандаш и бумага) написать то, что будет выведено на экран, если следующие команды MATLAB будут выполняться. Проверить свои ответы, выполняя команды с MATLAB.

$$(a) v ./ u \quad (b) u' * v \quad (c) u * v'$$

11. Даны два вектора: $\mathbf{u} = 5\mathbf{i} - 6\mathbf{j} + 9\mathbf{k}$ и $\mathbf{v} = 11\mathbf{i} + 7\mathbf{j} - 4\mathbf{k}$.

Используйте MATLAB для вычисления скалярного произведения $\mathbf{u} \cdot \mathbf{v}$ векторов тремя способами:

- Записать выражение, использующее поэлементно вычисление и встроенную функцию MATLAB `sum`.
- Определить \mathbf{u} как вектор строку и \mathbf{v} как вектор столбец, а затем использовать умножение матриц.
- Использовать встроенную функцию MATLAB `dot`.

12. Определите вектор $v = [2\ 3\ 4\ 5\ 6]$. Затем используйте этот вектор в математическом выражении, чтобы создать следующие векторы:

$$(a) \ a = [4\ 6\ 8\ 10\ 12] \quad (c) \ c = [2^2\ 3^3\ 4^4\ 5^5\ 6^6]$$

$$(b) \ b = [8\ 27\ 64\ 125\ 216] \quad (d) \ d = [1\ 1.5\ 2\ 2.5\ 3].$$

13. Определите вектор $v = [8\ 6\ 4\ 2]$. Затем используйте этот вектор в математическом выражении, чтобы создать следующие векторы:

$$(a) \ a = [1\ 1\ 1\ 1]$$

$$(b) \ b = [3\ 1\ -1\ -3]$$

$$(c) \ c = \left[\frac{1}{\sqrt{8}} \frac{1}{\sqrt{6}} \frac{1}{\sqrt{4}} \frac{1}{\sqrt{2}} \right]$$

$$(d) \ d = \left[\frac{1}{8^2} \frac{1}{6^2} \frac{1}{4^2} \frac{1}{2^2} \right].$$

14. Определите x и y как векторы $x = [1, 2, 3, 4, 5]$ и $y = [2, 4, 6, 8, 10]$. Затем используйте их в следующих выражениях для вычисления z используя поэлементные вычисления.

$$(a) \ z = \frac{(x+y)^2}{x-y}$$

$$(b) \ z = x \ln(x^2 + y^2) + \sqrt{\frac{y^3}{(y-x)^2}}.$$

15. Определите r и s как скаляры $r = 1.6 \times 10^3$ и $s = 14.2$ и t , x и y как векторы $t = [1, 2, 3, 4, 5]$, $x = [0, 4, 6, 8, 10]$ и $y = [3, 6, 9, 12, 15]$. Затем используйте эти переменные для вычисления следующих выражений используя поэлементные вычисления для векторов.

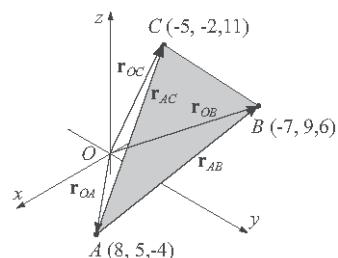
$$(a) \ G = xt + \frac{r}{s^2}(y^2 - x)t$$

$$(b) \ R = \frac{r(-xt + yt^2)}{15} - s^2(y - 0.5x^2)t.$$

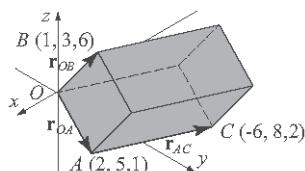
16. Площадь треугольника ABC может быть вычислена по формуле $|\mathbf{r}_{AB} \times \mathbf{r}_{AC}|/2$, где и векторы \mathbf{r}_{AB} и \mathbf{r}_{AC} соединяют вершины A и B и A и C , соответственно.

Определить площадь треугольника показанного на рисунке. Использовать следующую схему файла сценария для вычисления площади. Во-первых, определить векторы \mathbf{r}_{OA} , \mathbf{r}_{OB} и \mathbf{r}_{OC} по известным координатам точек A , B и C .

Затем определить векторы \mathbf{r}_{AB} и \mathbf{r}_{AC} из \mathbf{r}_{OA} , \mathbf{r}_{OB} и \mathbf{r}_{OC} . Наконец, определить площадь с использованием встроенных функций MATLAB `cross`, `sum` и `sqrt`.



17. Объем параллелепипеда, показанного на рисунке может быть вычислен по формуле $\mathbf{r}_{OB} \cdot (\mathbf{r}_{OA} \times \mathbf{r}_{AC})$. Использовать следующую схему файла сценария для вычисления объема. Определить векторы \mathbf{r}_{OA} , \mathbf{r}_{OB} и \mathbf{r}_{AC} по известным положениям точек A , B и C .



Затем определить векторы \mathbf{r}_{AB} и \mathbf{r}_{AC} из \mathbf{r}_{OA} , \mathbf{r}_{OB} и \mathbf{r}_{OC} . Определить объем с использованием встроенных функций MATLAB `dot` и `cross`.

18. Определите векторы: $\mathbf{u} = 5\mathbf{i} - 2\mathbf{j} + 4\mathbf{k}$, $\mathbf{v} = -2\mathbf{i} + 7\mathbf{j} + 3\mathbf{k}$ и $\mathbf{w} = 8\mathbf{i} + 1\mathbf{j} - 3\mathbf{k}$.

Использовать эти векторы для проверки тождества:

$$(\mathbf{u} + \mathbf{v})[(\mathbf{v} + \mathbf{w}) \times (\mathbf{w} + \mathbf{u})] = 2\mathbf{u} \cdot (\mathbf{v} \times \mathbf{w}).$$

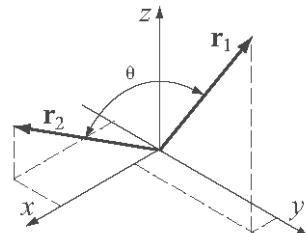
Использовать встроенные функции MATLAB `dot` и `cross` для вычисления значений левой и правой сторон тождества.

19. Скалярное произведение может использоваться для определения угла между двумя векторами:

$$\theta = \cos^{-1}\left(\frac{\mathbf{r}_1 \cdot \mathbf{r}_2}{|\mathbf{r}_1| \cdot |\mathbf{r}_2|}\right).$$

Использовать встроенные функции MATLAB `acosd`, `sqrt` и `dot` для нахождения угла (в радианах) между векторами $\mathbf{r}_1 = 6\mathbf{i} - 3\mathbf{j} + 2\mathbf{k}$ и $\mathbf{r}_2 = 2\mathbf{i} + 9\mathbf{j} + 10\mathbf{k}$.

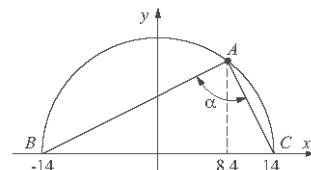
Напомним, что $|\mathbf{r}| = \sqrt{\mathbf{r} \cdot \mathbf{r}}$.



20. Использовать MATLAB для того, чтобы показать, что угол, вписанный в полукруг является прямым углом. Использовать следующие этапы в файле сценария для вычисления угла.

(a) Определить переменную со значением x координаты точки A .

(b) Определить y координату точки A из уравнения $x^2 + y^2 = R^2$. Определить векторы, которые соответствуют положениям точек A , B и C и использовать их для определения векторов \mathbf{r}_{AB} и \mathbf{r}_{AC} . Вычислить угол α двумя способами.



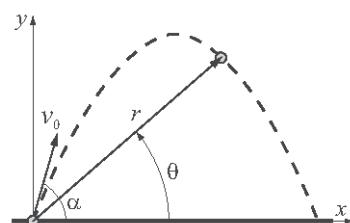
Сначала с использованием равенства $\alpha = \cos^{-1}\left(\frac{\mathbf{r}_{AB} \cdot \mathbf{r}_{AC}}{|\mathbf{r}_{AB}| \cdot |\mathbf{r}_{AC}|}\right)$, а затем с

использованием равенства $\alpha = \sin^{-1}\left(\frac{|\mathbf{r}_{AB} \times \mathbf{r}_{AC}|}{|\mathbf{r}_{AB}| \cdot |\mathbf{r}_{AC}|}\right)$. Оба способа должны дать 90° .

21. Положение $(x(t), y(t))$ снаряда, запущенного со скоростью v_0 и под углом α задаются следующими функциями времени,

$$x(t) = v_0 \cos \alpha \cdot t, \quad y(t) = v_0 \sin \alpha \cdot t - gt^2 / 2,$$

где $g = 9.81 \text{ м/с}^2$. Полярные координаты снаряда в момент времени t есть $(r(t), \theta(t))$ где



$$r(t) = \sqrt{x(t)^2 + y(t)^2} \quad \text{и} \quad \tan \theta = \frac{x(t)}{y(t)}.$$

Рассмотреть случай когда $v_0 = 162$ м/с и $\alpha = 70^\circ$. Определить $r(t)$ и $\theta(t)$ для $t = 1, 6, 11, \dots, 31$ сек.

22. Использовать MATLAB для того, чтобы показать, что сумма ряда $\sum_{n=0}^{\infty} \frac{2^n}{n!}$ сходится к e^2 . Для этого вычислить сумму для:

$$(a) n = 5, \quad (b) n = 10, \quad (c) n = 50.$$

Для каждого случая создать вектор n , в котором первый элемент 0, шаг – 1 и последний элемент 5, 10 или 50. Затем использовать поэлементные вычисления для создания вектора, элементы которого есть $\frac{2^n}{n!}$. Наконец, использовать встроенную функцию MATLAB `sum` для суммирования ряда. Сравнить полученное значение с e^2 (использовать `format long` для вывода чисел на экран).

23. Использовать MATLAB для того, чтобы показать, что сумма ряда $\sum_{n=1}^{\infty} \frac{(9/10)^n}{n}$ сходится к $\ln 10$. Для этого вычислить сумму для:

$$(a) n = 10, \quad (b) n = 50, \quad (c) n = 10.$$

Для каждого случая создать вектор n , в котором первый элемент 1, шаг – 1 и последний элемент 10, 50 или 100. Затем использовать поэлементные вычисления для создания вектора, элементы которого есть $\frac{(9/10)^n}{n}$. Наконец, использовать встроенную функцию MATLAB `sum` для суммирования ряда. Сравнить полученное значение с $\ln 10$ (использовать `format long` для вывода чисел на экран).

24. Согласно парадоксу Зенона любой объект в движении должен сначала пройти половину пути. Достигнув половины пути, он должен затем пройти половину оставшейся половины и так далее к бесконечности. Так как невозможно завершить этот бесконечный процесс, Зенон пришел к заключению, что все движение должно быть иллюзией. Считая длину пути единицей, парадокс Зенона может быть записан как бесконечная сумма $\sum_{n=1}^{\infty} \frac{1}{2^n}$. Чтобы увидеть, как быстро этот ряд сходится к 1, вычислите сумму для:

$$(a) n = 5, \quad (b) n = 10, \quad (c) n = 40.$$

Для каждого случая создать вектор n , в котором первый элемент 1, шаг – 1 и последний элемент 5, 10 или 40. Затем использовать поэлементные вычисления для создания вектора, элементы которого есть $\frac{1}{2^n}$. Наконец, использовать встроенную функцию MATLAB `sum` для суммирования ряда. Сравнить полученное значение с 1.

25. Показать, что $\lim_{x \rightarrow 0} \frac{\cos(2x) - 1}{\cos(x) - 1} = 4$. Для этого создайте вектор x , элементы которого есть 1.0, 0.5, 0.1, 0.01, 0.001, 0.00001 и 0.0000001. Затем, создайте новый вектор y , в котором каждый элемент определен элементом x по формуле $y = \frac{\cos(2x) - 1}{\cos(x) - 1}$. Сравните элементы вектора y со значением 4 (используйте

`format long` для вывода чисел на экран).

26. Показать, что $\lim_{x \rightarrow 1} \frac{x^{1/3} - 1}{x^{1/4} - 1} = \frac{4}{3}$. Для этого создайте вектор x , элементы которого есть 2.0, 1.5, 1.1, 1.01, 1.001, 1.00001 и 1.0000001. Затем, создайте новый вектор y , в котором каждый элемент определен элементом x по формуле $y = \frac{x^{1/3} - 1}{x^{1/4} - 1}$. Сравните элементы вектора y со значением 3/4 (используйте `format long` для вывода чисел на экран).

27. Потребность в воде во время пожара часто является наиболее важным фактором в проектировании резервуаров для хранения и насосов. Для общин с населением менее 200 000, потребность Q (галлонах/мин) может быть рассчитана по формуле :

$$Q = 1020\sqrt{P}(1 - 0.01\sqrt{P}),$$

где P – численность населения в тысячах. Создать вектор P , который начинается с 10 и с шагом 10 до 200. Использовать поэлементные вычисления для определения спроса Q для каждой группы населения в векторе P .

28. Уравнение идеального газа утверждает, что $P = \frac{nRT}{V}$, где P – давление, V – объем, T – температура, $R = 0.08206$ (L atm)/(mol K) – газовая постоянная и n – число молей. Реальные газы, особенно при высоком давлении, отклоняются от этого поведения. Их можно моделировать уравнением Ван-дер-Ваальса

$$P = \frac{nRT}{V - nb} - \frac{n^2a}{V^2},$$

где a и b являются материальными константами. Рассмотрите 1 моль ($n = 1$) газа азота при $T = 300$ К. (Для газа азота $a = 1.39$ – (Л²·атм)/моль² и $b = 0.0391$ Л/моль.) Создайте вектор со значениями V для $0.1 \leq V \leq 1$ Л, используя шаг 0.02 Л. Используйте этот вектор для вычисления P дважды для каждого значения V , один раз с использованием уравнения идеального газа, а другой – с уравнением Ван-дер-Ваальса. Используя два множества

значений P , вычислите процент ошибки $\frac{P_{ideal} - P_{wqqls}}{P_{wqqls}} \cdot 100\%$ для каждого

значения V . Наконец, используйте встроенную функцию MATLAB `max` для определения максимальной ошибки и соответствующего объема.

29. Создать следующие три матрицы:

$$A = \begin{bmatrix} 1 & -3 & 5 \\ 2 & 2 & 4 \\ -2 & 0 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & -2 & 1 \\ 5 & 1 & -6 \\ 2 & 7 & -1 \end{bmatrix}, \quad C = \begin{bmatrix} -3 & 4 & -1 \\ 0 & 8 & 2 \\ -3 & 5 & 3 \end{bmatrix}$$

- (a) Вычислить $A + B$ и $B + A$ чтобы показать, что сложение матриц коммутативно.
- (b) Вычислить $A + (B + C)$ и $(A + B) + C$ чтобы показать, что сложение матриц ассоциативно.
- (c) Вычислить $3(A + B)$ и $3A + 3B$ чтобы показать, что, когда матрицы умножаются на скаляр, умножение дистрибутивно.
- (d) Вычислить $A^*(B + C)$ и $A^*B + A^*C$ чтобы показать, что умножение матриц дистрибутивно.

30. Используя матрицы A , B и C из предыдущей задачи ответить на следующие вопросы:

- (a) Когда $A^*B = B^*A$?
- (b) Когда $A^*(B^*C) = (A^*B)^*C$?
- (c) Когда $(A^*B)^t = A^{t*}B^t$? (т означает транспонирование)
- (d) Когда $(A + B)^t = A^t + B^t$?

31. Создайте 4×4 матрицу A имеющую целочисленные случайные значения между 1 и 10. Вызовите матрицу A и используя MATLAB, выполните следующие операции. Для каждой части пояснить операцию.

- | | | |
|--------------|---------------|---------------------|
| (a) A^*A | (b) $A.^*A$ | (c) $A \setminus A$ |
| (d) $A ./ A$ | (e) $\det(A)$ | (f) $\text{inv}(A)$ |

32. Магический квадрат – это размещение чисел в виде квадратной матрицы таким образом, что сумма чисел в каждой строке, в каждом столбце и в каждой диагонали – одна и та же. В MATLAB есть встроенное функция `magic(n)`, которая возвращает магический квадрат. В файле сценарии создайте (6×6) магический квадрат и затем проверьте свойства получающейся матрицы, находя сумму элементов в каждой строке, в каждом столбце и в обеих диагоналях. В каждом случае, используйте встроенную функцию MATLAB `sum`. (Другие функции, которые могут быть полезными – это `diag` и `fliplr`.)

33. Решить следующую систему трех линейных уравнений:

$$\begin{aligned} -4x + 3y + z &= -18.2 \\ 5x + 6y - 2z &= -48.8 \\ 2x - 5y + 4.5z &= 92.5 \end{aligned}$$

34. Решить следующую систему пяти линейных уравнений:

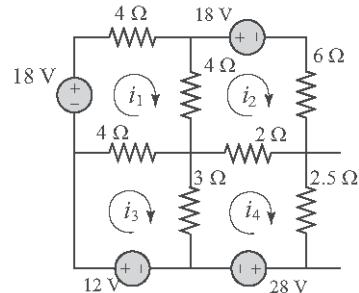
$$\begin{aligned} 2.5a - b + 3c + 1.5d - 2e &= 57.1 \\ 3a + 4b - 2c + 2.5d - e &= 27.6 \\ -4a + 3b + c - 6d + 2e &= -81.2 \\ 2a + 3b + c - 2.5d + 4e &= -22.2 \\ a + 2b + 5c - 3d + 4e &= -12.2 \end{aligned}$$

35. Продовольственная компания производит пять типов пакетов по 8 унций смесей сухофруктов с использованием различных смесей арахиса, миндаля, грецких орехов, изюма и M&M. Смеси содержат следующие компоненты:

Смеси	Арахис (унций)	Миндаль (унций)	Грецкие орехи (унций)	Изюм (унций)	M&M (унций)
Смесь 1	3	1	1	2	1
Смесь 2	1	2	1	3	1
Смесь 3	1	1	0	3	3
Смесь 4	2	0	3	1	2
Смесь 5	1	2	3	0	2

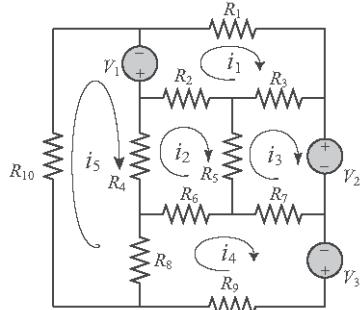
Сколько пакетов каждой смеси может быть произведено, если имеется 128 фунтов арахисов, 118 фунтов миндаля, 112 фунтов грецких орехов, 112 фунтов изюма и 104 фунта M&M? Запишите систему линейных уравнений и решите (1 фунт = 16 унций).

36. Показанная на рисунке электрическая цепь состоит из резисторов и источников напряжения. Определить для каждой петли i_1, i_2, i_3 и i_4 используя метод контурного тока, который основан на законе напряжений Кирхгофа (см. пример задачи 3.4).



37. Показанная на рисунке электрическая цепь состоит из резисторов и источников напряжения. Определить для каждой петли i_1, i_2, i_3, i_4 и i_5 используя метод контурного тока, который основан на законе напряжений Кирхгофа (см. пример задачи 3.4).

$$\begin{aligned} V_1 &= 40 \text{ В}, \quad V_2 = 39 \text{ В}, \quad V_3 = 36 \text{ В} \\ R_1 &= 16 \Omega, \quad R_2 = 20 \Omega, \quad R_3 = 10 \Omega \\ R_4 &= 14 \Omega, \quad R_5 = 8 \Omega, \quad R_6 = 16 \Omega, \\ R_7 &= 10 \Omega, \quad R_8 = 15, \quad R_9 = 6 \Omega, \quad R_{10} = 4 \Omega. \end{aligned}$$





ГЛАВА 4.

Использование файлов сценариев и управление данными

Файл сценария, или скрипт-файл (см. раздел 1.8), является списком команд MATLAB, называемых программой, который сохранен в отдельном файле. Когда выполняется файл сценария, MATLAB выполняет его команды. Раздел 1.8 описывает, как создать, сохранить, и выполнить простой файл сценария, в котором команды выполняются в порядке, в котором они записаны, и в котором все переменные определены в пределах этого файла. Эта глава дает дополнительные сведения о входных данных файла сценария, о том, как хранятся данные в MATLAB, о различных способах вывода на экран и сохранения данных, которые создаются в процессе работы файла сценария и как обмениваться данными между MATLAB и другими приложениями. (В главе 6 обсуждается как написать более усовершенствованные программы, в которых команды не обязательно выполняются в простом порядке.)

Вообще, переменные могут быть определены (созданы) несколькими способами. Как показано в главе 2, переменные могут быть определены неявно присвоением значения имени переменной. Переменным могут быть также присвоены значения результатов выполнения функций. Кроме того, переменные могут быть определены с помощью данных, которые импортированы из файлов вне MATLAB. После определения переменных (или в командном окне или в результате выполнения файла сценария) они сохраняются в Рабочем пространстве MATLAB.

Переменные, которые находятся в рабочем пространстве, могут быть выведены на экран различными способами, сохранены, или экспортированы во внешние приложения. Точно так же данные от внешних файлов по отношению к MATLAB могут быть импортированы в рабочее пространство и затем использоваться в MATLAB.

В разделе 4.1 объясняется, как MATLAB хранит данные в рабочем пространстве и как пользователь может увидеть эти данные. Раздел 4.2 показывает как переменные, которые используются в файлах сценария могут быть определены в командном окне и/или в файлах сценария. В разделе 4.3 показано, как вывести данные, созданные в результате работы файла сценария. В разделе 4.4 объясняется, как переменные могут быть сохранены в рабочем пространстве и затем использованы. Раздел 4.5 показывает как импортировать и экспортировать данные от и к внешним приложениям.

4.1. Рабочее пространство MATLAB и окно рабочего пространства

Рабочее пространство MATLAB состоит из множества переменных (называемых массивами), которые определены и сохранены во время сеанса MATLAB. Это переменные, которые были определены в командном окне и переменные, определенные во время выполнения файла сценария. Это означает, что командное окно и файлы сценария используют одну и ту же область памяти в компьютере. Следовательно, как только переменная появилась в рабочем пространстве, она распознаваема и может использоваться, ей могут быть присвоены новые значения как в командном окне, так и в файлах сценария. Как будет объяснено в главе 7 (раздел 7.3), в MATLAB есть другой тип файлов, называемых файлами функциями, где переменные могут также быть определены. Однако эти переменные обычно не являются общими для использования другими частями программы, так как они используют отдельное рабочее пространство.

Напомним, что команда `who` показывает список переменных, находящихся в настоящий момент в рабочем пространстве. Команда `whos` выводит на экран список переменных имеющихся в настоящий момент в рабочем пространстве вместе с информацией об их размере, байтах и классе. Ниже показаны примеры.

<code>>> 'Variables in memory'</code>	Ввод строки.																														
<code>ans =</code>	Переменной <code>ans</code> присвоена строка.																														
<code>Variables in memory</code>																															
<code>>> a = 7;</code>	Создание переменных <code>a</code> , <code>E</code> , <code>d</code> и <code>g</code>																														
<code>>> E = 3;</code>																															
<code>>> d = [5, a+E, 4, E^2]</code>																															
<code>d =</code>																															
5 10 4 9																															
<code>>> g = [a, a^2, 13; a*E, 1, a^E]</code>																															
<code>g =</code>																															
7 49 13																															
21 1 343																															
<code>>> who</code>	Команда <code>who</code> выводит на экран список текущих																														
<code>Your variables are:</code>	переменных в рабочем пространстве.																														
<code>E a ans d</code>																															
<code>>> whos</code>																															
<table border="1"> <thead> <tr> <th>Name</th> <th>Size</th> <th>Bytes</th> <th>Class</th> <th>Attributes</th> </tr> </thead> <tbody> <tr> <td>E</td> <td>1x1</td> <td>8</td> <td>double</td> <td>Команда <code>who</code> выводит на экран</td> </tr> <tr> <td>a</td> <td>1x1</td> <td>8</td> <td>double</td> <td>список переменных в рабочем</td> </tr> <tr> <td>ans</td> <td>1x19</td> <td>38</td> <td>char</td> <td>пространстве вместе</td> </tr> <tr> <td>d</td> <td>1x4</td> <td>32</td> <td>double</td> <td>с информацией об их размере,</td> </tr> <tr> <td>g</td> <td>2x3</td> <td>48</td> <td>double</td> <td>байтах и классе.</td> </tr> </tbody> </table>	Name	Size	Bytes	Class	Attributes	E	1x1	8	double	Команда <code>who</code> выводит на экран	a	1x1	8	double	список переменных в рабочем	ans	1x19	38	char	пространстве вместе	d	1x4	32	double	с информацией об их размере,	g	2x3	48	double	байтах и классе.	
Name	Size	Bytes	Class	Attributes																											
E	1x1	8	double	Команда <code>who</code> выводит на экран																											
a	1x1	8	double	список переменных в рабочем																											
ans	1x19	38	char	пространстве вместе																											
d	1x4	32	double	с информацией об их размере,																											
g	2x3	48	double	байтах и классе.																											
<code>>></code>																															

Переменные, находящиеся в текущий момент в памяти могут быть также просмотрены в **Окне рабочего пространства**. Это окно может быть открыто, выбирая **Workspace** в меню **Desktop**. (В версии MTLAB R2014 в разделе **Variable** ленты инструментов есть все возможности для просмотра и редактирования переменных рабочего пространства). Рис. 4.1 показывает окно рабочего пространства с определенными выше переменными.

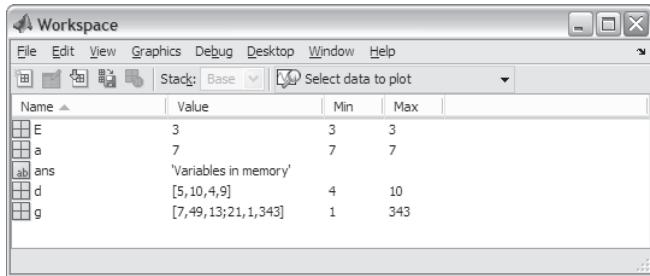


Рис. 4.1. Окно рабочего пространства

Переменные, которые выведены на экран в окне рабочего пространства, могут также быть отредактированы (изменены). Двойной щелчок по переменной открывает окно редактора переменных, где содержание переменной выводится на экран в виде таблицы. Например, рис. 4.2 показывает окно редактора переменных, которое открывается, после двойного щелчка по переменной *g* на рис. 4.1.

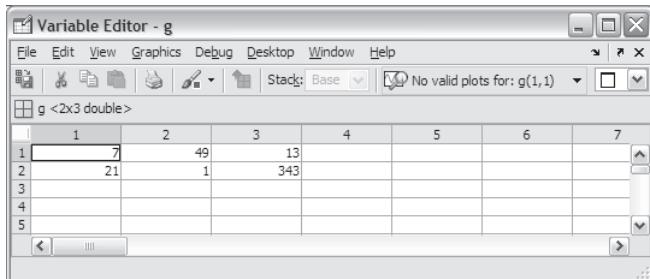


Рис. 4.2. Окно редактора переменных

Элементы в окне редактора переменных могут быть отредактированы. Переменные в окне рабочего пространства могут быть удалены. Для этого их нужно выбрать и затем либо нажать клавишу **Delete** на клавиатуре, либо выбрать **Delete** из меню **Edit**. Тот же самый эффект можно получить исполнением команды `clear variable_name` в командном окне.

4.2. Входные параметры файла сценария

Когда выполняется файл сценария, всем переменным, которые используются в вычислениях в пределах файла, должны быть присвоены значения. Другими сло-

вами, переменные должны быть в рабочем пространстве. Присвоение значения переменной может быть сделано тремя способами, в зависимости от того, где и как эта переменная определяется.

1. Переменная определена и ей присвоено значение в файле сценария

В этом случае присвоение значения переменной – это часть файла сценария. Если пользователь желает присвоить переменной новое значение, файл должен быть отредактирован для присвоения переменной измененного значения. После сохранения файла он может быть выполнен снова.

Ниже идет пример такого случая. Файл сценария (сохраненный как Chapter4Example2) вычисляет средние очки, набранные в трех играх.

```
% Этот скрипт-файл вычисляет среднее очков, набранных в трех играх.  
% Присвоение значений очков - это часть скрипта-файла.  
game1=75;  
game2=93;  
game3=68;  
ave_points=(game1+game2+game3) / 3
```

**Переменным присвоены значения
в пределах файла сценария.**

После выполнения файла сценария в командном окне следующий вывод:

```
>> Chapter4Example2  
ave_points =  
78.6667  
>>
```

**Файл сценария выполняется вводом имени файла.
Переменная ave_points с ее значением
выводится на экран в командном окне.**

2. Переменная определена и ей присвоено значение в командном окне

В этом случае присвоение значения переменной сделано в командном окне. (Напомним, что переменная указана в файле сценария.) Если пользователь хочет выполнить файл сценария с другим значением переменной, то новое значение присваивается в командном окне и файл выполняется снова.

Для предыдущего примера, когда у файла сценария есть программа, которая вычисляет среднее число очков, набранное в трех играх, файл сценария в этом случае (сохраненный как Chapter4Example3) имеет вид:

```
% Этот скрипт-файл вычисляет среднее очков, набранных в трех играх.  
% Присвоение значений очков переменным game1, game2 и game3  
% выполнено в командном окне.
```

```
ave_points=(game1+game2+game3) / 3
```



Для работы этого файла в командном окне должно быть следующее:

```
>> game1 = 67;  
>> game2 = 90;  
>> game3 = 81;  
  
>> Chapter4Example3  
  
ave_points =  
79.3333  
>> game1 = 87;  
>> game2 = 70;  
>> game3 = 50;  
  
>> Chapter4Example3  
  
ave_points =  
69  
>>
```

Присвоение значений переменным в командном окне.

Команда выполнения скрипта-файла.

Результат выполнения скрипта-файла выводится на экран в командном окне. Переменным присваиваются новые значения.

Скрипт-файл выполняется снова.

Результат выполнения скрипта-файла выводится на экран в командном окне.

3. Переменная определена в файле сценария, но конкретное значение вводится в командном окне при выполнении файла сценария

В этом случае переменная определена в файле сценария и при выполнении файла у пользователя запрашивается присвоить ей значение в командном окне. Это делается с использованием команды `input` для создания переменной. Вид команды `input`:

```
variable_name = input('строка с сообщением, которое появится  
в командном окне')
```

Когда выполняется команда `input` при работе файла сценария, строка выводится на экран в командном окне. Стока – это сообщение, запрашивающее пользователя ввести значение, которое должно быть присвоено переменной. Пользователь вводит значение и нажимает клавишу **Enter**. Это действие присваивает значение переменной. Как и любая другая переменная, эта переменная и ее присвоенное значение будут выведены на экран в командном окне, если точка с запятой не будет введена в самом конце команды `input`. Ниже приведен файл сценария, который использует команду `input` для введения очков, набранных в каждой игре для программы, которая вычисляет среднее значение очков.

```
% Этот скрипт-файл вычисляет среднее очков, набранных в трех играх.  
% Очки от каждой игры присвоены переменным с использованием  
% команды input.
```



```
game1=input('Введите очки, набранные в первой игре ');
game2=input('Введите очки, набранные во второй игре ');
game3=input('Введите очки, набранные в третьей игре ');
ave_points=(game1+game2+game3) / 3
```

После выполнения этого файла сценария (сохраненного как Chapter4Example4) командное окно будет иметь следующий вид:

```
>> Chapter4Example4
Введите очки, набранные в первой игре 67
Введите очки, набранные во второй игре 91
Введите очки, набранные в третьей игре 70
ave_points =
76
>>
```

Компьютер выводит сообщение. Затем пользователь вводит значение очков и нажимает Enter.

В этом примере переменным присваиваются скаляры. Однако могут также быть присвоены векторы и массивы. Это делается введением значений массива так же, как обычно при присвоении переменной (левая скобка, затем ввод строки за строкой и правая скобка).

Команда `input` может также использоваться для присвоения строки переменной. Это может быть сделано одним из двух способов. Первый способ состоит в том, чтобы использовать команду `input` в той же самой форме как показано выше. Когда появляется подсказка, нужная строка вводится между двумя одинарными кавычками таким же образом, что и при присвоении строки переменной без команды `input`. Второй способ состоит в том, чтобы использовать опцию в команде `input`, которая определяет символы, которые вводятся после подсказки как строку (`string`). Форм команды:

```
variable_name = input('подсказка', 's')
```

где '`s`' в команде определяет символы, которые будут вводиться как строка. В этом случае после появления подсказки текст вводится без одинарных кавычек, но он присваивается переменной как строка (`string`). Пример, где команда `input` используется именно с этой опцией, имеется в примере задачи 6.4.

4.3. Команды вывода

Как обсуждалось выше, MATLAB автоматически генерирует вывод на экран при выполнении некоторых команд. Например, когда переменной присвоено значение, или введено имя ранее присвоенной переменной и нажата клавиша **Enter**, MATLAB выводит на экран переменную и ее значение. Если в конце команды поставлена точка с запятой, то этот тип результата не будет выведен на экран. В дополнение к этому автоматическому выводу на экран у MATLAB есть несколько

команд, которые могут использоваться для генерирования вывода на экран дисплея. Выводится на экран могут сообщениями, которые предоставляют информацию, числовые данные и графики. Есть две команды, которые часто используются для вывода результата, это `disp` и `fprintf`. Команда `disp` выводит результат на экран, а команда `fprintf` может использоваться как для вывода на экран результата, так и для сохранения результата в файле. Эти команды могут использоваться в командном окне, в файле сценария, и, как будет показано позже, в файле функции. Когда эти команды используются в файле сценария, результат который они генерируют, выводится на экран в командном окне.

4.3.1. Команда *disp*

Команда `disp` используется для того, чтобы вывести на экран элементы переменной не выводя на экран имя переменной и для того, чтобы вывести на экран текст. Формат команды `disp`:

`disp(имя переменной)` или `disp('текст как строка (string)')`

- Каждый раз, когда выполняется команда `disp`, вывод, который она генерирует, появляется в новой строке. Один пример:

Следующий пример показывает использование команды `disp` в файле сценария, который вычисляет среднее значение очков, набранных в трех играх.

После выполнения этого файла сценария (сохраненного как Chapter4Example5) командное окно будет иметь следующий вид:

```
>> Chapter4Example5
Введите очки, набранные в первой игре 89
Введите очки, набранные во второй игре 60
Введите очки, набранные в третьей игре 82
77
    Выведено значение переменной ave_points.
```

Выведена пустая строка.
Выведен текст.
Выведена пустая строка.

- Только одна переменная может быть выведена на экран командой `disp`. Если элементы двух переменных должны быть выведены на экран вместе, новая переменная (которая содержит элементы, предполагаемые к выводу на экран), должна быть сначала определена и затем выведена на экран.

Во многих ситуациях желательно вывести результаты (числа) на экран в таблице. Это может быть сделано сначала определением переменной как массива чисел и затем использованием команды `disp` для вывода на экран массива. Заголовки к столбцам могут также быть созданы командой `disp`. Поскольку в команде `disp` пользователь не может управлять форматом (ширины столбцов и расстоянием между столбцами) вывода массива на экран, то положение заголовков должно быть выровнено со столбцами добавлением пробелов. Для примера следующий файл сценария показывает, как вывести на экран в таблице данные о населении из главы 2.

```
yr=[1984 1986 1988 1990 1992 1994 1996];
pop=[127 130 136 145 158 178 211];
tableYP(:,1)=yr';           Данные о населении введены
                           в два вектора строки.
tableYP(:,2)=pop';          yr вводится как первый столбец в массиве tableYP.
                           pop вводится как первый столбец в массиве tableYP.
disp('      ГОД          НАСЕЛЕНИЕ')   Вывод заголовка (первая строка).
disp('                  (МИЛЛИОНОВ) ')     Вывод заголовка (первая линия).
disp(' ')
disp(tableYP)               Вывод пустой строки.
                           Вывод массива tableYP.
```

После выполнения этого файла сценария (сохраненного как PopTable) вид командного окна будет следующий:

```
>> PopTable
      ГОД          НАСЕЛЕНИЕ      Выведены заголовки.
                           (МИЛЛИОНОВ)
                           1984          127
                           1986          130
    Выведена пустая строка.
```

1988	136
1990	145
1992	158
1994	178
1996	211

Выведена таблица tableYR.

4.3.2. Команда *fprintf*

Команда *fprintf* может использоваться для вывода на экран результата (текст и данные) или для сохранения результата в файле. Эта команда (в отличие от *disp*) позволяет форматирование результата. Например, текст и числовые значения переменных могут быть смешанно выведены на экран на одной той же строке. Кроме того, форматом вывода чисел можно управлять.

Со многими доступными параметрами эта команда *fprintf* может быть длинной и сложной. Чтобы избежать путаницы, эта команда представлена постепенно. Сначала в этом разделе показано, как использовать эту команду для вывода на экран текстовых сообщений, затем как смешанно вывести числовые данные и текст, а затем как отформатировать вид выводимых чисел, и наконец как сохранить результат в файле.

Использование команды *fprintf* для вывода на экран текста

Для вывода на экран текста у команды *fprintf* есть следующая форма:

```
fprintf('выводимый текст как строка string')
```

Например:

```
fprintf ('Введенная задача не имеет решения. Пожалуйста, проверьте  
входные данные.')
```

Если эта строка – часть файла сценария, то после выполнения этой строки на экран в командном окне будет выведено следующее:

```
Введенная задача не имеет решения. Пожалуйста, проверьте входные данные.
```

Команда *fprintf* позволяет перейти на новую строку вывода в середине текстовой строки. Это делается вставкой *\n* перед символом, с которого начнется новая строка. Например, вставка *\n* после первого предложения в предыдущем примере дает:

```
fprintf ('Введенная задача не имеет решения. \nPожалуйста, проверьте  
входные данные.')
```

Когда эта команда будет выполнена в командном окне появится следующее:

Введенная задача не имеет решения.
Пожалуйста, проверьте входные данные.

Эту команду \n называют символом перехода на новую строку. Она используется для управления выводом на экран. Другие символы управления, которые могут быть вставлены в пределах строки:

\b возврат назад на одну позицию.

\t горизонтальная табуляция.

Когда у программы есть больше одной команды `fprintf`, то создаваемый вывод на экран не прерывается (команда `fprintf` автоматически не запускает новую строку). Это верно, даже если есть другие команды между командами `fprintf`. Пример – в следующем файле сценария:

```
fprintf ('Введенная задача не имеет решения. Пожалуйста, проверьте
входные данные.')
x = 6; d = 19 + 5*x;
fprintf('Попытайтесь выполнить программу позже.')
y = d + x;
fprintf('Используйте другие входные значения.')
```

Когда этот файл будет выполнен, в командном окне появится следующее:

Введенная задача не имеет решения. Пожалуйста, проверьте входные данные.
Попытайтесь выполнить программу позже. Используйте другие входные значения.

Чтобы запустить новую строку с командой `fprintf`, в начале этой строки должен быть введен символ \n.

Использование команды `fprintf` для вывода на экран смешанно текста и числовых данных

Для вывода на экран смешанно текста и чисел (значений переменных) у команды `fprintf` есть следующая форма:

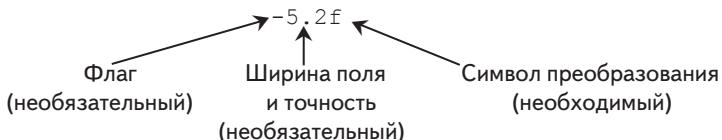
```
fprintf('текст как string %-5.2f дополнительный текст', variable_name')
```

Знак % отмечает место, где вставлено число в пределах текста.

Форматирующие элементы (определяют формат числа).

Имя переменной, значение которой выводится на экран.

Три элемента форматирования имеют следующий смысл:



Флаг, который является необязательным, может быть одним из следующих трех символов:

Символ используемый для флага	Описание
- (знак «минус»)	Левое выравнивание числа в поле.
+ (знак «плюс»)	Печатает символ знака (+ или –) перед числом.
0 (нуль)	Добавляет нули, если число короче чем поле.

Ширина поля и точность (5.2 в предыдущем примере) являются необязательными. Первое число (5 в примере) является шириной поля, которая определяет минимальное число цифр на дисплее. Если число, которое будет выведено на экран, короче ширины поля, то перед этим числом добавляются пробелы или нули. Точность – это второе число (2 в примере). Оно определяет число цифр, которые будут выведены на экран после десятичной точки.

Последний элемент в элементах форматирования (f в примере) обязателен, он является символом преобразования числа в систему обозначений, в которой будет выведено число на экран.

Некоторые из распространенных систем обозначений:

- e – экспоненциальное представление чисел, используя нижний регистр e (например, 1.709098e+001).
- E – экспоненциальное представление чисел, используя верхний регистр E (например, 1.709098E+001).
- f – с фиксированной точкой (например, 17.090980).
- g – более компактное чем e или f. Незначащие нули не печатаются.
- G – более компактное чем E или f. Незначащие нули не печатаются.
- i – целое.

Информация о дополнительных способах записи числа доступна в меню Help MATLAB. В качестве примера использования команды `fprintf` со смешиванием текста и чисел, приведем файл сценария, который вычисляет средние очки, набранные в трех играх.

% Этот скрипт-файл вычисляет среднее очков, набранных в трех играх.

% Очки от каждой игры присвоены переменным с использованием команды `input`.

% Для вывода на экран результата используется команда `fprintf`. ➔

```
game1=input('Введите очки, набранные в первой игре ');
game2=input('Введите очки, набранные во второй игре ');
game3=input('Введите очки, набранные в третьей игре ');
ave_points= mean(game);
fprintf(' Среднее значение %f очков было набрано в этих трех играх.',ave points)
```

Текст

% показывает
положение числа

Дополнительный текст

Имя переменной
которая будет
выводится

Заметим, что помимо использования команды `fprintf`, этот файл отличается от приведенных ранее в этой главе, значения очков записываются в первых трех элементах вектора под именем `game` и среднее значение очков вычисляется с использованием функции `mean`. Ниже показано командное окно после выполнения этого файла сценария (сохраненного как `Chapter4Example6`).

```
>> Chapter4Example6
Введите очки, набранные в первой игре 75
Введите очки, набранные во второй игре 60
Введите очки, набранные в третьей игре 81
Среднее значение 72.000000 очков было набрано в этих трех играх.
>>                                         Вывод, сгенерированный командой fprintf
                                         в комбинации текста и числа (значения переменной).
```

Команда `fprintf` позволяет вставить в пределах текста более одного числа (значения переменной). Это делается вводом `%g` (или `%` и далее любой элемент форматирования) в местах в тексте, где должны быть вставлены числа. Затем, после строки аргумента команды (после запятой), нужно вставить имена переменных в порядке, в котором они будут вставлены в текст. Вообще команда имеет вид:

```
fprintf('...text...%g...%g...%f...',variable1,variable2,variable3)
```

Пример показан в следующем файле сценария:

```
% Эта программа вычисляет расстояние, на которое летит снаряд,
% учитывая его начальную скорость и угол к горизонту при выстреле.
% используется команда fprintf для вывода на экран текста и чисел.
```

```
v=1584; % Initial velocity (km/h)
theta=30; % Angle (degrees)
vms=v*1000/3600;
t=vms*sind(30)/9.81;
d=vms*cosd(30)*2*t/1000;
```

Изменение модуля скорости км/с.
Вычисление времени для самой высокой точки.
Вычисление максимального расстояния.

```
fprintf('Снаряд выпущенный под углом %3.2f градусов со скоростью
%4.2f km/h пролетит расстояние %g km.\n',theta,v,d)
```

Когда этот файл сценария (сохраненный как Chapter4Example7) будет выполнен, в командном окне отобразится следующее:

```
>> Chapter4Example7
```

Снаряд выпущенный под углом 30.00 градусов со скоростью 1584.00 km/h пролетит расстояние 17.091 km.
>>

Дополнительные комментарии о команде *fprintf*

- Чтобы вывести на экран в тексте одинарную кавычку, нужно ввести две одинарные кавычки в строке внутри команды.
- Команда *fprintf* является векторизованной. Это означает, что если включенная в команду переменная является вектором или матрицей, то команда повторяется до тех пор, пока все элементы не будут выведены на экран. Если переменная является матрицей, то данные используются столбец за столбцом.

Например, приведенный ниже файл сценария создает 2×5 матрицу *T*, в которой первая строка содержит номера от 1 до 5, а вторая строка показывает соответствующие квадратные корни

```
x=1:5;                  Создание вектора x
y=sqrt(x);              Создание вектора y
T=[x; y]                Создание 2x5 матрицы T, с первой строкой x, а второй y
fprintf('Если число есть: %i, его квадратный корень есть: %f\n',T)
Команда fprintf выводит на экран два числа из T в каждой строке.
```

После выполнения этого скрипта-файла в командном окне будет следующее:

```
T =
1.0000 2.0000 3.0000 4.0000 5.0000
1.0000 1.4142 1.7321 2.0000 2.2361
Если число есть: 1, его квадратный корень есть: 1.000000
Команда fprintf
Если число есть: 2, его квадратный корень есть: 1.414214
повторяется пять раз,
Если число есть: 3, его квадратный корень есть: 1.732051
используя числа из
Если число есть: 4, его квадратный корень есть: 2.000000
матрицы T столбец
Если число есть: 5, его квадратный корень есть: 2.236068
за столбцом.
```

Использование команды *fprintf* для сохранения результата в файле

В дополнение к отображению результата в командном окне команда *fprintf* может использоваться для записи результат в файл, когда необходимо сохранить

этот результат. Данные, которые сохраняются могут быть впоследствии выведены на экран или использоваться в MATLAB и в других приложениях.

Для записи результата в файлу нужно пройти три шага:

1. Открыть файл, используя команду `fopen`.
2. Записать результат в открытом файле, используя команду `fprintf`.
3. Закрыть файл, используя команду `fclose`.

Шаг 1

Прежде, чем данные могут быть записаны в файл, этот файл должен быть открыт. Это делается командой `fopen`, которая создает новый файл или открывает существующий. Команда `fopen` имеет следующую форму:

```
fid = fopen('имя_файла', 'разрешение')
```

`fid` – это переменная, называемая идентификатором файла. Когда `fopen` выполняется, переменной `fid` присваивается скалярная величина,. Имя файла записано (включая его расширение) внутри одинарных кавычек как строка. Разрешение – это код (также записанный как строка) который говорит, как файл открыт. Некоторые из более общих кодов разрешения:

- '`r`' – файл открыт для чтения (значение по умолчанию).
- '`w`' – файл открыт для записи. Если файл уже существует, его содержание удаляется. Если файл не существует, создается новый файл.
- '`a`' – то же самое, что и '`w`', за исключением того, что, если файл существует, то записываемые данные добавляются в конец файла.
- '`r+`' – файл открыт (не создан) для чтения и записи.
- '`w+`' – файл открыт для чтения и записи. Если файл уже существует, его содержание удаляется. Если файл не существует, создается новый файл.
- '`a+`' – то же самое, что и '`w+`', за исключением того, что, если файл существует, то записываемые данные добавляются в конец файла.

Если код разрешения не включен в команду, файл открывается с кодом '`r`' значения по умолчанию. Дополнительные коды разрешения описаны в справочной системе MATLAB.

Шаг 2

Как только файл открыт, команда `fprintf` может использоваться для записи результата в файл. Команда `fprintf` используется точно таким же образом, как она использовалась для вывода результатов на экран в командном окне, за исключением того, что в команде вставлена переменная `fid`. Команда `fprintf` имеет следующую форму:

```
fprintf(fid, 'text %-5.2f additional text', variable_name)
```

Добавлен идентификатор `fid`.

Шаг 3

После выполнения записи данных в файл, этот файл закрывается, используя команду `fclose`. Команда `fclose` имеет следующую форму:

```
fclose (fid)
```

Дополнительные примечания по использованию команды `fprintf` для сохранения результатов в файл

- Создаваемый файл сохраняется в текущем каталоге.
- Можно использовать команду `fprintf` для записи в несколько различных файлов. Это делается сначала открытием файлов и присвоением различных `fid` каждому (например, `fid1`, `fid2`, `fid3` и т. д.), а затем использованием `fid` определенного файла в команде `fprintf` для записи в этот файл.

Пример использования команды `fprintf` для сохранения результатов в двух файлах показан в следующем файле сценария. Эта программа в файле генерирует две таблицы преобразования модулей. Одна таблица преобразовывает модули скоростей из миль в час к километрам в час, а другая таблица преобразовывает модули силы из фунтов в Ньютоны. Каждая таблица преобразования сохраняется в отдельном текстовом файле (с расширением .txt).

```
% Скрипт-файл, в котором используется fprintf для записи результата в файлы.
% Созданы две таблицы пересчета и сохранены в двух различных файлах.
% Одна преобразует миль/час в км/ч, а другая преобразует фунты в Ньютоны.
clear all
Vmph=10:10:100;                                     Создание вектора скоростей в мил/ч.
Vkmh=Vmph.*1.609;                                   Преобразование мили в час в км/ч.
TBL1=[Vmph; Vkmh];                                 Создание таблицы (матрицы) с двумя строками.
Flb=200:200:2000;                                  Создание вектора сил в фунтах.
FN=Flb.*4.448;                                    Преобразование фунтов в ньютоны.
TBL2=[Flb; FN];                                   Создание таблицы (матрицы) с двумя строками.
fid1=fopen ('VmphToVkm.txt','w');                 Открытие.txt файла с именем VmptoVkm.
fid2=fopen ('FlbtoFN.txt','w');                    Открытие.txt файла с именем FlbtoFN.
fprintf(fid1,'Таблица преобразования скоростей \n \n');
                                                Запись заголовка и пустой строки в файле fid1.
fprintf(fid1,'    миль/час      км/ч \n');
                                                Запись двух заголовков столбцов в файле fid1.
printf(fid1,'    %8.2f      %8.2f\n',TBL1);
                                                Запись данных из переменной TBL1 в файл fid1.
fprintf(fid2,'Таблица преобразования сил\n \n');   Запись таблицы преобразо-
fprintf(fid2,'    Фунты      Ньютоны \n');        вания силы в файл fid1.
fprintf(fid2,'    %8.2f      %8.2f\n',TBL2);       (переменной TBL2) в файл fid2
fclose(fid1);                                     Закрытие файлов fid1 и fid2.
fclose(fid2);
```

После выполнения этого файла сценария в текущем каталоге создаются и сохраняются два новых .txt файла с именами VmphtoVkm и FlbtoFN. Эти файлы могут быть открыты в любом приложении, которое может читать .txt файлы. Рис. 4.3 и 4.4 показывают, как эти два файла выглядят, когда они открыты с Microsoft Word.

mi/h	km/h
10.00	16.09
20.00	32.18
30.00	48.27
40.00	64.36
50.00	80.45
60.00	96.54
70.00	112.63
80.00	128.72
90.00	144.81
100.00	160.90

Рис. 4.3. Файл VmptphoVkm.txt открылся в Word

Pounds	Newtons
200.00	896.00
400.00	1792.00
600.00	2688.00
800.00	3584.00
1000.00	4480.00
1200.00	5376.00
1400.00	6224.00
1600.00	7112.00
1800.00	8000.00
2000.00	8896.00

Рис. 4.4. Файл FlbtoFN.txt открылся в Word

4.4. Команды save и load

Команды save и load сохранения и загрузки являются самыми полезными для сохранения и получения данных для их использования в MATLAB. Команда save может использоваться для того, чтобы сохранить переменные, которые находятся

в настоящий момент в рабочем пространстве, а команда загрузки `load` используется для восстановления в рабочем пространстве переменных, которые были ранее сохранены. Рабочее пространство может быть сохранено, когда MATLAB используется на одной платформе (например, PC) и восстановлено для использования в MATLAB на другой платформе (например, Mac). Команды сохранения и загрузки могут также использоваться для обмена данными с внешними приложениями MATLAB. Дополнительные команды, которые могут использоваться с этой целью, представлены в разделе 4.5.

4.4.1. Команда `save`

Команда `save` используется для того, чтобы сохранить переменные на диске (все или некоторые из них), которые находятся в рабочем пространстве. Две самых простых формы команды `save`:

```
save file_name и save('file_name')
```

Когда выполняется любая из этих команд, все переменные находящиеся в настоящий момент в рабочем пространстве сохраняются в файле с именем `file_name.mat`, который создается в текущем каталоге. В `.mat` файлах, которые записаны в бинарном формате, каждая переменная сохраняет свое имя, тип, размер и значение. Эти файлы не могут быть считаны другими приложениями. Команды `save` может также использоваться для сохранения только некоторых из переменных из рабочего пространства. Например, чтобы сохранить две переменные с именами `var1` и `var2`, эта команда имеет вид:

```
save file_name var1 var2 или save('file_name','var1','var2')
```

Команды `save` может также использоваться для сохранения в формате ASCII, который может быть считан внешними приложениями MATLAB. Сохранение в формате ASCII делается добавлением в команде аргумента `-ascii` (например, `save file_name -ascii`). В формате ASCII имя переменной, тип и размер не сохраняются. Данные сохраняются как символы, разделенные пробелами, но без имен переменных. Например, ниже показано, как две переменные (вектор и матрица) определены в командном окне и затем сохраняются формате ASCII в файле под именем `DatSavAsci`:

```
>> V=[3 16 -4 7.3];
>> A=[6 -2.1 15.5; -6.1 8 11];
>> save -ascii DatSavAsci
```

Создание 1×4 вектора V
Создание 2×3 матрицы A
Сохранение переменных в файле с именем DatSavAsci.

После сохранения файл может быть открыт любым приложением, которое может читать файлы ASCII. Например, рис. 4.5 показывает данные, когда файл открыт в блокноте Notepad.

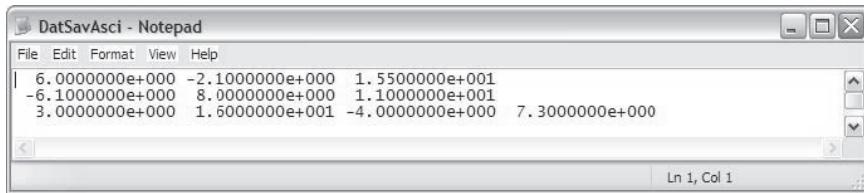


Рис. 4.5. Данные сохранены в формате ASCII

Отметим, что файл не включает имена переменных, а только список численных значений переменных (сначала A, затем V).

4.4.2. Команда *load*

Команда загрузки *load* может использоваться для того, чтобы получить обратно в рабочее пространство переменные, которые были сохранены командой *save* и для того, чтобы импортировать данные, которые создавались другими приложениями в формате ASCII, или в текстовых (.txt) файлах. Переменные, которые были сохранены командой *save* в .mat файлах, могут быть получены следующей командой:

```
load имя_файла или load('имя_файла')
```

Когда выполняется команда, все переменные в файле (с именем, типом, размером и значениями, как они были сохранены) добавляются (загружаются обратно) в рабочее пространство. Если у рабочего пространства уже есть переменная с тем же самым именем что и загружаемая переменная, то эта загружаемая переменная заменяет существующую переменную. Команда загрузки *load* может также использоваться для получения только некоторые из переменных, которые находятся в сохраненном .mat файле. Например, чтобы получить две переменные с именами *var1* и *var2*, используется команда:

```
load имя_файла var1 var2 или load('имя_файла', 'var1', 'var2')
```

Команда *load* может также использоваться для импортирования в рабочее пространство данных, которые сохранены в ASCII или текстовом (.txt) файле. Это возможно, однако, только если данные в файле находятся в форме переменных MATLAB. Это означает, что у файла может быть одно число (скаляр), строка или столбец чисел (вектор), или набор строк с одинаковым числом чисел в каждом (матрица). Например, данные, показанные в рис. 4.5, не могут быть загружены командой *load* (даже при том, что это был формат ASCII, в котором сохранены данные командой *save*), потому что число элементов не одно и то же во всех строках. (Напомним, что этот файл создавался ранее при сохранении двух различных переменных.)

Когда данные загружаются в рабочее пространство из ASCII или текстового файла, им должно быть присвоено имя переменной. Данные в формате ASCII могут быть загружены любой из следующих двух форм команды загрузки *load*:

```
load имя_файла или VarName=load('имя_файла')
```

Если данные находятся в текстовом файле, расширение .txt должно быть добавлено к имени файла. Тогда вид команды загрузки:

```
load имя_файла.txt или VarName=load('имя_файла.txt')
```

В первой форме команды данные присваиваются переменной, имя которой совпадает с именем файла. Во второй форме данные присвоены переменной под именем VarName.

Например, данные на рис. 4.6 (3×2 матрица) введены в **Блокнот**, а затем сохранены как DataFromText.txt.



Рис. 4.6. Данные, сохраненные в виде .txt файла

Затем используются вторая формы команды загрузки для импорта данных из текстового файла в рабочее пространство MATLAB. В первой команде данные присвоены переменной с именем DfT. Во второй команде данным автоматически присваивается переменная с именем DataFromText, которое является именем текстового файла, в котором были сохранены эти данные.

```
>> DfT=load('DataFromText.txt')
DfT =
    56.0000    -4.2000
    3.0000     7.5000
   -1.6000    198.0000
>> load DataFromText.txt
>> DataFromText
DataFromText =
    56.0000    -4.2000
    3.0000     7.5000
   -1.6000    198.0000
```

Загрузка файла DataFromText и присвоение загруженных данных переменной DfT.

Использование команды load с файлом DataFromText.

Данные присвоены переменной с именем DataFromText.

Импорт данных из (или экспорт в) других приложений может быть также сделан командами MATLAB, которые представлены в следующем разделе.

4.5. Импорт и экспорт данных

MATLAB часто используется для того, чтобы проанализировать данные, которые были записаны в экспериментах или созданы другими компьютерными программами. Это может быть сделано первым импортом данных в MATLAB. Точно

так же данные, которые созданы MATLAB иногда должны передаваться другим компьютерным приложениям. Есть различные типы данных (числовые, текст, аудио, графика и изображения). Этот раздел описывает как импортировать и экспорттировать только числовые данные, которые являются, вероятно, наиболее распространенным типом данных при передаче новыми пользователями MATLAB. О передаче других типов данных, см. в пункте **File I/O** окна справки команды.

Импорт данных может быть сделан или с использованием команд или с использованием Мастера импорта. Команды полезны, когда известен формат импортируемых данных. MATLAB имеет несколько команд, которые могут использоватьсь для импорта различных типов данных. Команды импорта могут быть также включены в файл сценария так, чтобы данные импортировались при выполнении скрипта-файла. Мастер импорта полезен, когда формат данных (или команда, которая применима для импорта данных) не известна. Мастер импорта определяет формат данных и автоматически их импортирует.

4.5.1. Команды для импорта и экспорта данных

Этот раздел описывает в деталях как передать данные «в» и «из» электронных таблиц Excel. Microsoft Excel обычно используется для хранения данных и Excel совместим со многими устройствами записи данных и компьютерными приложениями. Кроме того, много людей умеют импортировать и экспорттировать различные форматы данных «в» и «из» Excel. MATLAB имеет также команды для передачи данных непосредственно «в» и «из» форматов, таких как csv и ASCII, а также электронных таблиц программы Lotus 123. Подробности этого и многие другие команды могут быть найдены в **Окне справки** в пункте **File I/O**.

Импорт и экспорт данных «в» и «из» Excel

Импорт данных «из» Excel делается командой `xlsread`. При выполнении команды данным из электронной таблицы присваиваются значения переменной как массива. Самая простая форма команды `xlsread`:

```
variable_name = xlsread('filename')
```

- 'filename' (вводится как строка (string)) является именем файла Excel. Каталог файла Excel должен быть или текущим каталогом или перечисленным в путях поиска.
- Если у файла Excel будет более одного листа, то данные будут импортированы из первого листа.

Когда у файла Excel есть несколько листов, команда `xlsread` может использоваться для импорта данных из указанного листа. Форма такой команды:

```
variable_name = xlsread('filename', 'sheet_name')
```

- Имя листа вводится как строка.

Еще одна опция позволяет импортировать только часть данных, которые находятся в электронной таблице. Это делается введением дополнительного аргумента в команде:

```
variable_name = xlsread('filename', 'sheet_name', 'range')
```

- Здесь диапазон 'range' (вводится как строка) является прямоугольной областью электронной таблицы, определенной адресами (в системе обозначений Excel) ячеек в противоположных углах области. Например, 'C2:E5' – это область строк 2, 3, 4, и 5 и столбцов C, D, и E.

Экспорт данных из MATLAB «в» электронную таблицу Excel делается с использованием команды xlswrite. Самая простая форма команды:

```
xlswrite('filename', variable_name)
```

- 'filename' (вводится как строка (string)) является именем файла Excel в который экспортируются данные. Этот файл должен быть в текущем каталоге. Если файл не существует, то будет создан новый файл Excel с указанным именем.
- variable_name – это имя переменной в MATLAB с присвоенными данными, которые экспортируются.
- Аргументы 'sheet_name' и 'range' могут быть добавлены к команде xlswrite для экспорта в указанный лист и в указанный диапазон ячеек, соответственно.

Например, на рис. 4.7 показаны данные из электронной таблицы Excel, которые импортируются в MATLAB с использованием команды xlswrite.

The screenshot shows a Microsoft Excel window titled "Microsoft Excel - TestData1.xls". The spreadsheet contains four rows of data (1-4) and eight columns (A-K). The data is as follows:

	A	B	C	D	E	F	G	H	I	J	K
1	11	2	34	14	-6	0	8				
2	15	6	-20	8	0.56	33	5				
3	0.9	10	3	12	-25	-0.1	4				
4	55	9	1	-0.555	17	6	-30				

The status bar at the bottom left shows "Ready". The formula bar above the grid displays the current cell address "M8".

Рис. 4.7. Электронная таблица Excel с данными

Электронная таблица хранится на диске A в файле по имени TestData1. После изменения текущего каталога на диск A, данные импортируются в MATLAB присвоением им переменной DATA:

```
>> DATA = xlsread('TestData1')
DATA =
11.0000 2.0000 34.0000 14.0000 -6.0000 0 8.0000
15.0000 6.0000 -20.0000 8.0000 0.5600 33.0000 5.0000
0.9000 10.0000 3.0000 12.0000 -25.0000 -0.1000 4.0000
55.0000 9.0000 1.0000 -0.5550 17.0000 6.0000 -30.0000
```

4.5.2. Использование Мастера импорта

Использование **Мастера импорта** – это, вероятно, самый легкий способ импортировать данные в MATLAB, так как пользователь не должен знать, или определять, формат данных. Мастер импорта активируется выбором **Import Data** в меню **File** командного окна. (Он может также быть запущен командой `uiimport`.) При запуске Мастера импорта открывается обычное диалоговое окно выбора файла, в котором видны все файлы с данными текущего каталога, опознанные Мастером. Пользователь выбирает файл, который содержит данные и которые будут импортированы, и щелкает **Open**. Мастер импорта открывает файл и выводит на экран часть данных в поле предварительного просмотра так, чтобы пользователь мог проверить, что данные выбраны верно. Мастер импорта пытается обработать данные, и если это удается, он выводит на экран предварительного просмотра переменные, которые он создает для части данных. Мастер импорта позволяет выбрать диапазон импортируемых данных и тип данных, в котором они будут в MATLAB: столбцы векторов (каждый столбец – отдельная переменная); матрица; массив ячеек; таблица. Пользователь может выбрать тип переменной, а также имя переменной (или имена векторов столбцов данных) загрузки этот формат и затем нажать кнопку **Import Selection** для импорта выбранных данных в MATLAB. (Когда все данные являются числовыми, переменная в MATLAB имеет то же самое имя, что и файл, из которого были импортированы данные.)

В качестве примера используем Мастер импорта для импорта числовых данных ASCII сохраненных в .txt файле. Эти данные сохранены в файле с именем *TestData2* и показаны на рис. 4.8.

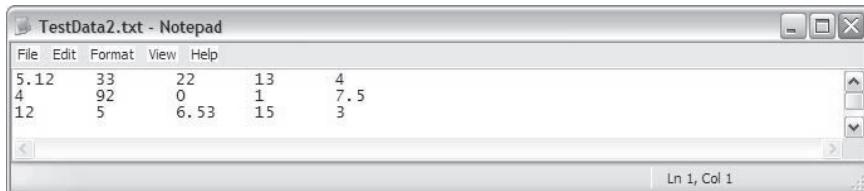
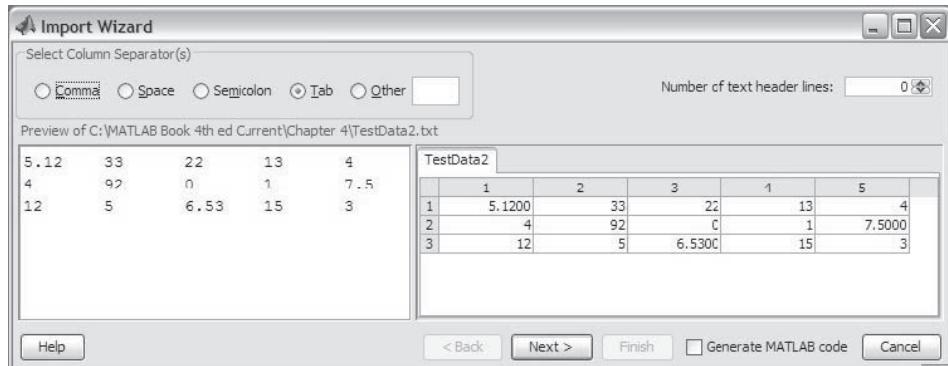
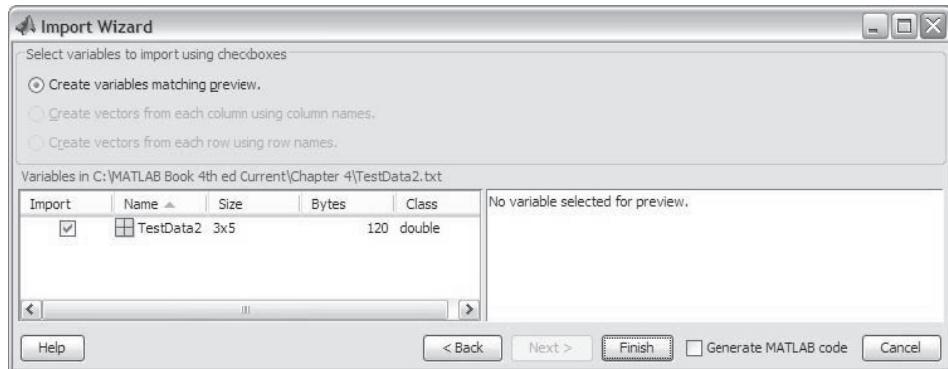


Рис. 4.8. Электронная таблица Excel с данными

Дисплей Мастера импорта во время процесса импорта для файла *TestData2* показан на рис. 4.9 и 4.10. Рис. 4.10 показывает, что имя переменной в MATLAB есть *TestData2*, его размер 3×5 .

**Рис. 4.9.** Мастер импорта, начальный вывод на экран**Рис. 4.10.** Мастер импорта, второй вывод

В командном окне MATLAB импортированные данные могут быть выведены на экран вводом имени переменной.

```

>> TestData2
TestData2 =
    5.1200    33.0000    22.0000    13.0000    4.0000
    4.0000    92.0000         0    1.0000    7.5000
   12.0000     5.0000    6.5300    15.0000    3.0000

```

4.6. Примеры приложений MATLAB

Пример задачи 4.1. Высота и площадь поверхности бункера

Цилиндрический бункер радиуса r имеет крышу в виде сферического сегмента радиуса R .

Высота цилиндрической части – H . Написать программу в виде скрипта-файла, которая находит высоту H для заданных значений r , R и объема V .

Дополнительно программа вычисляет площадь поверхности бункера.

Используйте эту программу для вычисления высоты и площади поверхности бункера с $r = 30$ фут, $R = 45$ фут и объемом 200 000 фут³. Присвойте значения r , R и V в командном окне.

Решение

Суммарный объем бункера получается сложением объема цилиндрической части и объема сферического сегмента. Объем цилиндра определяется по формуле

$$V_{cyl} = \pi r^2 H,$$

а объем сферического сегмента:

$$V_{cap} = \frac{1}{3} \pi h^2 (3R - h),$$

где $h = R - R \cos \theta = R(1 - \cos \theta)$ и θ вычисляется из соотношения $\sin \theta = \frac{r}{R}$.

Используя указанные уравнения, высота H цилиндрической части может быть выражена формулой

$$H = \frac{V - V_{cap}}{\pi r^2}.$$

Площадь поверхности бункера получается сложением площадей поверхностей цилиндрической части и сферического сегмента.

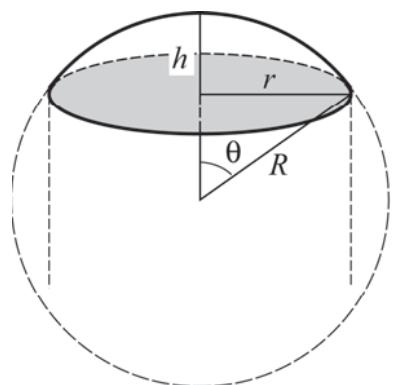
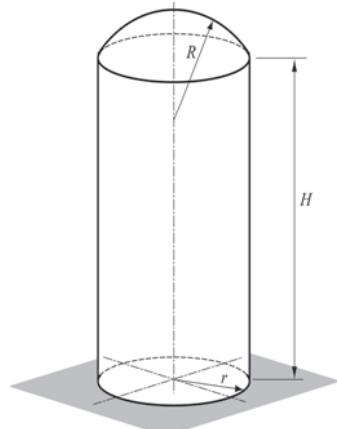
$$S = S_{cyl} + S_{cap} = 2\pi r H + 2\pi R h.$$

Ниже представлена программа в файле сценария, которая решает эту задачу:

```

theta=asin(r/R);
h=R*(1-cos(theta));
Vcap=pi*h^2*(3*R-h)/3;
H=(V-Vcap)/(pi*r^2);
S=2*pi*(r*H + R*h);
fprintf('The height H is: %f ft.',H)
fprintf('\nThe surface area of the silo is: %f square ft.',S)

```



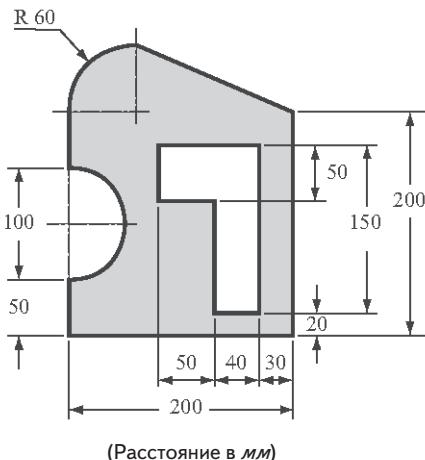
Командное окно при выполнении этого файла сценария с именем `silo`:

```
>> r=30; R=45; V=200000;
>> silo
The height H is: 64.727400 ft.
The surface area of the silo is: 15440.777753 square ft.
```

Присвоение значений r , R и V .
Запуск скрипта-файла `silo`.

Пример задачи 4.2. Центр тяжести сложной области

Записать программу в файле сценария, которая вычисляет координаты центра тяжести сложной области. (Составная область может легко быть разделена на разделы, центры тяжести которых известны.) Пользователь должен разделить область на части и узнать координаты центра тяжести (два числа) и площадь каждой части (одно число). При выполнении файла сценария он предлагает пользователю ввести эти три числа как строку в матрице. Пользователь вводит столько строк, сколько имеется частей. Часть, которая представляет прорезь, берется как имеющая отрицательную площадь. В качестве результата программы выводит на экран координаты центра тяжести сложной области. Используйте программу, чтобы вычислить центр тяжести области, показанной на рисунке.



(Расстояние в мм)

Решение

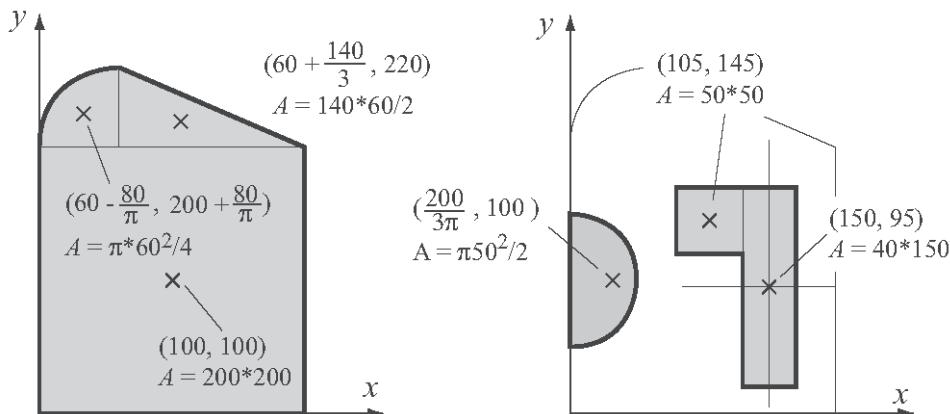
Область делится на шесть частей как показано на следующем рисунке. Общая площадь вычисляется сложением трех частей слева и вычитанием трех частей справа. Расположение и координаты центра тяжести, а также площади каждой части указаны на рисунке.

Координаты \bar{X} и \bar{Y} центра тяжести общей области задаются формулами

$$\bar{X} = \frac{\sum A\bar{x}}{\sum A} \quad \text{и} \quad \bar{Y} = \frac{\sum A\bar{y}}{\sum A},$$

где \bar{x} , \bar{y} и A – это координаты центра тяжести и площадь каждой части, соответственно.

Файл сценария с программой для вычисления координат центра тяжести сложной области представлен ниже.



Примечание: координаты в мм, площадь в мм²

```
% Эта программа вычисляет координаты центра тяжести
% составной области.
clear C xs ys As
C=input ('Введите матрицу, у которой каждая строка имеет три
элемента. \n В каждую строку введите координаты x и y центра
тяжести и площадь каждой части. \n');
xs=C(:,1)';
ys=C(:,2)';
As=C(:,3)';
A=sum(As);
x=sum(As.*xs)/A;
y=sum(As.*ys)/A;
fprintf('Координаты центра тяжести есть: (%f, %f)\n',x,y)
```

Создание вектора строки для x координаты каждой части (первый столбец С).

Создание вектора строки для y координаты каждой части (второй столбец С).

Создание вектора строки для площади каждой части (третий столбец С).

Вычисление полной площади.

Вычисление координат центра тяжести составной области.

Файл сценария был сохранен с именем Centroid. Ниже показано командное окно при выполнении этого скрипта-файла.

```
>> Centroid
Введите матрицу, у которой каждая строка имеет три элемента.
В каждую строку введите координаты x и y центра тяжести и площадь
каждой части.
[100 100 200*200
60-80/pi 200+80/pi pi*60^2/4
60+140/3 220 140*60/2
200/(3*pi) 100 -pi*50^2/2
105 145 -50*50
150 95 -40*150]
Координаты центра тяжести есть: (85.387547, 131.211809)
```

Ввод данных для матрицы С.

Каждая строка имеет три элемента:
 x , y и A части.

Пример задачи 4.3. Распределитель напряжения

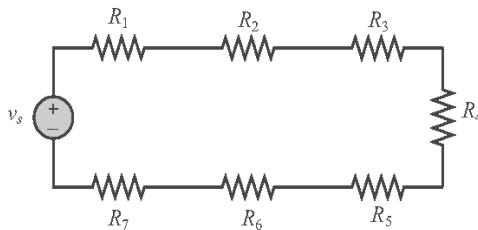
Когда несколько резисторов соединены в электрической цепи последовательно, напряжение через каждого из них определяется правилом распределения напряжения:

$$v_n = \frac{R_n}{R_{eq}} v_s ,$$

где v_n и R_n – это напряжение на резисторе n и его сопротивление, соответственно, $R_{eq} = \sum R_n$ – эквивалентное сопротивление и v_s – исходное напряжение. Рассеиваемая мощность в каждом резисторе определяется формулой:

$$P_n = \frac{R_n}{R_{eq}^2} v_s^2 .$$

На рисунке ниже показана цепь с семью резисторами, соединенными последовательно.



Написать программу в файле сценария, которая вычисляет напряжение на каждом резисторе и мощность, рассеиваемую каждым резистором в цепи соединенных последовательно резисторов. При выполнении скрипта-файла он должен запрашивать у пользователя сначала ввода исходного напряжения, а затем ввода сопротивления резисторов в виде вектора. Программа выводит на экран таблицу с сопротивлениями, указанными в первом столбце, напряжениями на резисторах – во втором столбце и мощностями, рассеянные на резисторах – в третьем столбце. После таблицы программа выводит на экран ток в цепи и полную мощность.

Выполните файл и введите следующие данные для v_s и R_n .

$v_s = 24$ В;

$R_1 = 20$ Ом, $R_2 = 14$ Ом, $R_3 = 12$ Ом, $R_4 = 18$ Ом, $R_5 = 8$ Ом, $R_6 = 150$ Ом, $R_7 = 10$ Ом.

Решение

Ниже показан скрипт-файл, который решает эту задачу.

```
% Программа вычисляет напряжение на каждом резисторе цепи,
% в которой резисторы соединены последовательно.
vs=input ('Пожалуйста, введите исходное напряжение');
Rn=input ('Введите значения сопротивлений как элементов вектор-
строки \n');
```

```

Req=sum (Rn);
vn=Rn*vs/Req;
Pn=Rn*vs^2/Req^2;
i = vs/Req;
Ptotal = vs*i;
Table = [Rn', vn', Pn'];
disp(' ')
disp('Сопротивление    Напряжение    Мощность')
disp('   (Ом)          (Вольт)       (Ватт)')
disp(' ')
disp(Table)
disp(' ')
fprintf ('Ток в цепи есть %f Ампер.', i)
fprintf ('\nПолная мощность, рассеянная в цепи есть %f
Ватт.', Ptotal)

```

Ниже показано командное окно при выполнении этого скрипта-файла.

>> VoltageDivider	Имя скрипта-файла.	
Пожалуйста, введите исходное напряжение 24		Напряжение введенное
Введите значения сопротивлений как элементов вектор-строки		пользователем.
[20 14 12 18 8 15 10]	Значения сопротивлений, введенные как вектор.	
Сопротивление	Напряжение	Мощность
(Ом)	(Вольт)	(Ватт)
20.0000	4.9485	1.2244
14.0000	3.4639	0.8571
12.0000	2.9691	0.7346
18.0000	4.4536	1.1019
8.0000	1.9794	0.4897
15.0000	3.7113	0.9183
10.0000	2.4742	0.6122

Ток в цепи есть 0.247423 Ампер.
Полная мощность, рассеянная в цепи есть 5.938144 Ватт.

4.7. Задачи

Решить следующие задачи написанием программы в файле сценария и затем выполнением программы.

1. Индекс тепла Hl , вычисленный из температуры воздуха и относительной влажности, является кажущейся температурой, которую чувствует тело. Уравнение, используемое Национальной метеорологической службой для вычисления Hl , дает:

$$HI = -42.379 + 2.04901523T + 10.14333127R - 0.22475541R - 6.83783 \times 10^{-3} T^2 - 5.481717 \times 10^{-2} R^2 + 1.22874 \times 10^{-3} T^2 R + 8.5282 \times 10^{-4} T R^2 - 1.99 \times 10^{-6} T^2 R^2$$

где T – температура в градусах по Фаренгейту и R – относительная влажность в целочисленных процентах. Написать программу MATLAB в файле сценария, которая вычисляет HI . При этом, программа требует ввода значений T и R . В качестве результата программа выводит на экран сообщение: «Температура индекса тепла: XX», где XX – значение индекса тепла, округленное к самому близкому целому числу. Исполнить программу, вводя $T = 90$ °F и $R = 90\%$.

2. Ежемесячный вклад P на депозитный счет, который должен зачисляться на счет с годовой процентной ставкой r для накопления общей суммы F за N лет, может быть вычислен по формуле:

$$P = \frac{F(r/12)}{(1+r/12)^{12N} - 1}.$$

Вычислить размер ежемесячного вклада для накопления 100 000 \$ за 5, 6, 7, 8, 9 и 10 лет, если годовая процентная ставка составляет 4.35%. Вывести на экран результаты в таблице с двумя столбцами, где первый столбец – число лет, а второй столбец – ежемесячный депозит.

3. Рост некоторых популяций бактерий может быть описан формулой

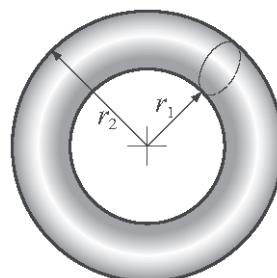
$$N = N_0 e^{kt},$$

где N – число индивидуумов в момент времени t , N_0 – их число в момент времени $t = 0$ и k – некоторая постоянная. Считая, что число бактерий удваивается каждые 40 минут, определить число бактерий через каждые два часа в течение 24 часов, начиная с единственной бактерии.

4. Объем V и площадь поверхности S трубы с водой торической формы задается формулами:

$$V = \frac{1}{4} \pi^2 (r_1 + r_2)(r_2 - r_1)^2 \quad \text{и} \quad S = \pi^2 (r_2^2 - r_1^2).$$

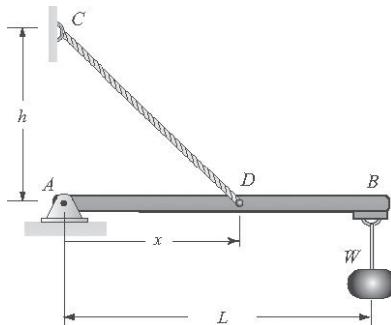
Если $r_1 = 0.7 r_2$, определить V и S для $r_2 = 12, 16, 20, 24$ и 28 дюймов. Вывести на экран результаты в таблице с четырьмя столбцами, где первый столбец есть r_2 , второй – r_1 , третий – V и четвертый – S .



5. Балка длины L присоединена к стене с кабелем как показано на рисунке. К балке приложена нагрузка $W = 500$ фунтов. Сила натяжения T в кабеле задается формулой:

$$T = \frac{WL\sqrt{h^2 + x^2}}{hx}.$$

Для балки с $L = 120$ дюймов и $h = 50$ дюймов вычислить T для $x = 10, 30, 50, 70, 90$ и 110 дюймов.



6. Написать программу MATLAB в файле сценария, которая вычисляют среднее значение, стандартное отклонение и медиану списка оценок, а также число оценок в списке. При работе программа запрашивает пользователя (команда `input`) ввести оценки как элементы вектора. Затем программа вычисляет необходимые величины используя встроенные функции `length`, `mean`, `std` и `median` MATLAB. Результаты выводятся на экран в командном окне в следующем формате:

«Имеется XX оценок.», где XX – числовое значение.

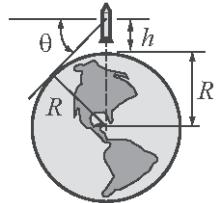
«Средняя оценка XX.», где XX – числовое значение.

«Стандартное отклонение XX.», где XX – числовое значение.

«Средняя оценка XX.», где XX – числовое значение.

Выполните программу и введите следующие оценки: 92, 74, 53, 61, 100, 42, 80, 66, 71, 78, 91, 85, 79 и 68.

7. Ракета, летящая прямо, измеряет угол с горизонтом на различных высотах h . Напишите MATLAB программу в файле сценария, которая вычисляет радиус Земли R (предполагается, что земля – круглая сфера) по указанным ниже данным и затем определяет среднее всех значений.



h (км)	4	8	12	16	20	24	28	32	36	40
θ (град.)	2.0	2.9	3.5	4.1	4.5	5.0	5.4	5.7	6.1	6.4

8. Распад радиоактивных материалов может быть смоделировано уравнением $A = A_0 e^{kt}$, где A – количество в момент времени t , A_0 – количество при $t = 0$, и k является постоянной затухания ($k \leq 0$). Йод-132 является радиоизотопом, который используется в исследовании функции щитовидной железы. Его полу-период существования составляет 13.3 часов. Вычислить относительное количество йода-132 (A/A_0) в теле пациента спустя 48 часов после получения дозы. После определения значения k задать вектор $t = 0, 4, 8, \dots, 48$ и вычислить соответствующие значения.
9. Ежемесячная оплата P ипотеки в сумме L длительностью N лет с ежегодной процентной ставкой r задается формулой:

$$P = L \frac{\frac{r}{100 \cdot 12} \left(1 + \frac{r}{100 \cdot 12}\right)^{12N}}{\left(1 + \frac{r}{100 \cdot 12}\right)^{12N} - 1},$$

где r учитывается в % (например 7.5% вводится как 7.5). Написать программу MATLAB в файле сценария, которая вычисляет P . При выполнении программы должна запросить у пользователя введения суммы ипотеки, числа лет и процентной ставку. Результат выводится на экран в следующем формате: «Ежемесячная оплата XX летней ипотеки в сумме XXXXXX.XX с процентной ставкой XX.XX равна \$XXXX.XX», где XXX – это подстановка соответствующих величин. Использовать программу для определения ежемесячной платы по ипотеке в сумме 250 000 \$ в течение 30 лет и годовой процентной ставкой 4.5%.

- 10.** Состояние счета ссуды B после n ежемесячных платежей задается формулой:

$$B = A \left(1 + \frac{r}{1200}\right)^n - \frac{P}{r/1200} \left[\left(1 + \frac{r}{1200}\right)^n - 1 \right],$$

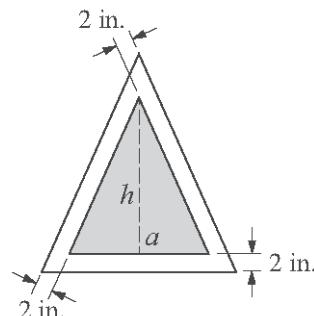
где A – величина ссуды P – размер ежемесячной оплаты и r – годовая процентная ставка, вводимая в % (например, 7.5% вводится как 7.5). Рассмотрим 5-летний автокредит в сумме 20 000 \$ с годовой процентной ставкой 6.5% и ежемесячной платой в размере 391,32 \$. Вычислить сумму задолженности (баланс) ссуды после каждого из 6 месяцев гашения (то есть, для $n = 6, 12, 18, 24, \dots, 54, 60$). Для каждого n вычислить процент ссуды, которая уже оплачена. Вывести на экран результаты в таблице с тремя столбцами, где первый столбец выводит на экран число месяцев, а второй и третий столбцы выводят на экран соответствующие значения B и процента ссуды, которая уже оплачена, соответственно

- 11.** Раньше проводники альпинистов часто оценивали высоту измеряя температуру кипящей воды. Используйте следующие два уравнения для создания таблицы, которую современные путешественники могли бы использовать в тех же целях.

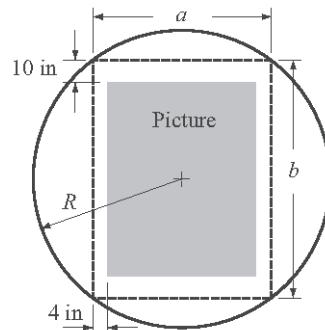
$$p = 29.921 (1 - 6.8753 \times 10^{-6} h), \quad T_b = 49.161 \ln p + 44.932,$$

где p – атмосферное давление в дюймах ртутного столба, T_b – температура кипения в °F и h – высота в футах. У таблицы должно быть два столбца, первый – высота и второй – температура кипения. Высота должна располагаться между –500 футов и 10 000 футов с шагом 500 футов.

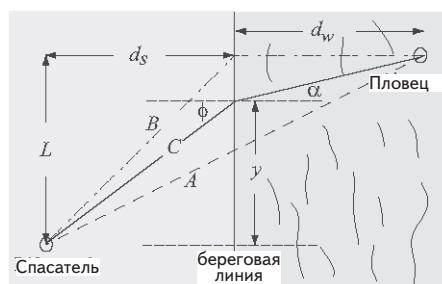
12. Знак равнобедренного треугольника проектируется так, чтобы иметь a треугольной печатной области 600 дюймов² (заштрихованная область с длиной основания a и высотой h на рисунке). Как показано в рисунке, промежуток между сторонами треугольников есть 2 дюйма. Напишите программу MATLAB, которая определяет размеры a и h так, что полная площадь знака будет как можно меньше. В программе определить вектор a со значениями в пределах от 10 до 120 дюймов с шагом приращения 0.1. Используйте этот вектор для того, чтобы вычислить соответствующие значения h и полную площадь знака. Затем используйте встроенную функцию `min` MATLAB, чтобы найти размеры наименьшего знака.



13. Круглый рекламный щит радиуса $R = 55$ дюймов проектируется так, чтобы в нем поместить прямоугольное изображение со сторонами a и b . Поля между прямоугольником и изображением 10 дюймов сверху и 4 дюйма внизу с каждой стороны. Напишите программу MATLAB, которая определяет размеры a и b так, чтобы полная область изображения была бы наибольшей. В программе определить вектор a со значениями в пределах от 5 до 100 с шагом приращения 0.25. Используйте этот вектор для того, чтобы вычислить соответствующие значения b и полной площади изображения. Затем используйте встроенную функцию MATLAB `max` чтобы найти размеры самого большого прямоугольника.



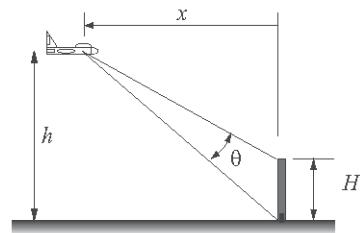
14. Студент работает летом спасателем на пляже. После того, как он заметил проблемы одного из пловцов, он стал пытаться найти путь, по которому он может достигнуть пловца за самое короткое время. Путь кратчайшего расстояния (путь A) очевидно, не лучший вариант, поскольку в этом случае увеличивается время, потраченное на плавание (он может бежать быстрее, чем плавать). Путь B минимизирует время, затраченное на плавание, но является, вероятно, не лучшим вариантом, поскольку это самый длинный (разумный) путь. Ясно, что оптимальный путь находится где-нибудь между путями A и B .



Рассмотрите промежуточный путь C и определите время, требуемое для достижения пловца, если скорость бега $v_{run} = 3$ м/с, а скорость его плавания $v_{swim} = 1$ м/с; расстояния $L = 48$ м, $d_s = 30$ м, и $d_w = 42$ м; и боковое расстояние y , где спасатель входит в воду. Создайте вектор y , который располагается между путем A и путем B ($y = 20, 21, \dots, 48$ м) и вычислите время t для каждого значения y . Используйте встроенную функциональную \min MATLAB, чтобы найти минимальное время t_{\min} и соответствующее место входа y . Определите углы, которые соответствуют расчетному значению y и выясните, удовлетворяет ли Ваш результат закону преломления Снеллиуса:

$$\frac{\sin \phi}{\sin \alpha} = \frac{v_{run}}{v_{swim}}.$$

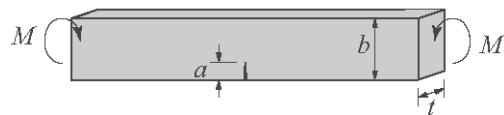
- 15.** Самолет летит на высоте $h = 900$ футов наблюдая цель высотой 70 футов ($H = 70$ футов), как показано на рисунке. Лучшее представление о цели будет тогда, когда угол θ максимален. Напишите программу MATLAB, которая определяет расстояние x при котором угол θ максимален. Определить вектор x с элементами в пределах от 50 до 1500 с интервалом 0.5. Используйте этот вектор для вычисления соответствующих значений θ . Затем используйте встроенную функцию \max MATLAB для нахождения значения x , которое соответствует самому большому значению θ .



- 16.** Коэффициент интенсивности напряжения K в трещине балки, в результате чистого изгиба M задается формулой:

$$K = C \sigma \sqrt{\pi a},$$

где $\sigma = \frac{6M}{tb^2}$, a – длина трещины, b – ширина, t – толщина и C – параметр,



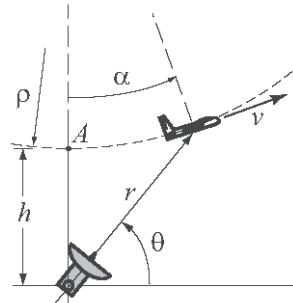
который зависит от геометрии экземпляра и трещины. Для случая чистого изгиба,

$$C = \sqrt{\frac{\tan \beta}{\beta} \left[\frac{0.923 + 0.199(1 - \sin \beta)^2}{\cos \beta} \right]}, \quad \text{где } \alpha = a/b \quad \text{и} \quad \beta = (\pi \alpha)/2.$$

Напишите программу в файле сценария, которая вычисляет коэффициент интенсивности напряжения K . Программа должна считать значения M , b , t и a из ASCII текстового файла, используя команду загрузки `load`. Результат должен быть в форме абзаца в комбинации текста и чисел, то есть, примерно так: «Коэффициент интенсивности напряжения для балки шириной 0.25 м и толщиной

0.01 м с граничной трещиной 0.05 м и приложенным моментом 20 Н·м, есть XX Pa·sqrt(м).», где XX – место для значения K . Использовать программу для вычисления K , когда $M = 20$ Н·м, $b = 0.25$ м, $t = 0.01$ м и $a = 0.25$ м.

- 17.** Показанный на рисунке самолет летит с постоянной скоростью $v = 50$ м/с по окружности радиуса $\rho = 2000$ м и отслеживается радаром, расположенным на расстоянии $h = 500$ м ниже нижней части плоскости траектории (точка A). Самолет находится в точке A при $t = 0$ и угол α как функция времени задается (в радианах) формулой $\alpha = vt/\rho$. Напишите программу MATLAB для вычисления θ и r как функций времени. Эта программа должна сначала определить время, когда $\alpha = 90^\circ$. Затем создайте вектор t имеющий 15 элементов в интервале $0 \leq t \leq t_{90^\circ}$ и вычислите θ и r для каждого момента времени. Программа должна напечатать заданные значения ρ , h и v , а затем результаты вычислений в таблице размерами 15×3 , где первый столбец – время t , второй – угол в градусах и третий – соответствующее значение r .



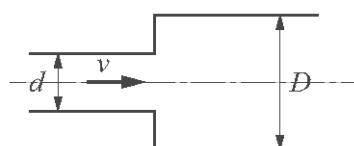
- 18.** Внутренняя электрическая удельная проводимость σ полупроводника может быть аппроксимирована формулой:

$$\sigma = e^{\left(c - \frac{E_g}{2kT}\right)},$$

где σ измерена в ($\text{Ом}^{-1} \cdot \text{м}^{-1}$), E_g является шириной запрещенной зоны, k является постоянной Больцмана ($\text{эВ}/\text{К}$) и T – температура в кельвинах. Для Германия, $C = 13.83$ и $E_g = 0.67$ эВ. Напишите программу в файле сценария, которая вычисляет внутреннюю электрическую удельную проводимость для Германия для различных температур. Значения температуры должны быть считаны из электронной таблицы xls, используя команду xlsread. Результат должен быть представлен таблицей, где первый столбец – температура, а второй столбец – внутренняя электрическая удельная проводимость. Используйте следующие значения для температуры: 400, 435, 475, 500, 520 и 545 К.

- 19.** Понижение давления Δp (Па) для течения жидкости в канале с внезапным увеличением диаметра задается формулой:

$$\Delta p = \frac{1}{2} \left[1 - \left(\frac{d}{D} \right)^2 \right]^2 \rho v^2,$$



где ρ – плотность жидкости, v – скорости потока, а d и D определены на рисунке. Запишите программу в файле сценария, которая вычисляет падение

давления Δp . При выполнении скрипта-файл запрашивает пользователя ввести плотность в кг/м³, скорость в м/с и значения безразмерного отношения d/D в виде вектора. Программа выводит на экран введенные значения ρ и v и затем таблицу со значениями d/D в первом столбце и соответствующими значениями Δp во втором столбце.

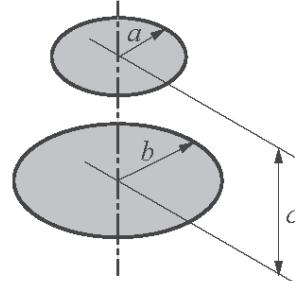
Выполните программу в предположении потока бензина ($\rho = 737$ кг/м³) при $v = 5$ м/с и следующих отношениях диаметров $d/D = 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2$.

- 20.** Поток тепла при излучении от пластины 1 с радиусом b к пластиине 2 с радиусом a , которые находятся на расстоянии c , задается формулой:

$$q = \sigma\pi b^2 F_{1-2} (T_1^4 - T_2^4),$$

где T_1 и T_2 – абсолютные температуры пластин, $\sigma = 5.669 \times 10^{-8}$ Вт·м⁻²·К⁻⁴ – это постоянная Стефана-Больцмана и F_{1-2} – коэффициент конфигурации, который, для размещения показанном на рисунке, задается формулой:

$$F_{1-2} = \frac{1}{2} \left[Z - \sqrt{Z^2 - 4X^2 Y^2} \right],$$



где $X = a/c$, $Y = c/b$ и $Z = 1 + (1 + X^2)Y^2$. Напишите скрипт-файл, который вычисляет теплообмен q . Программа просит ввода значений T_1 , T_2 , a , b и c . При выводе программа печатает параметры конфигурации F_{1-2} , X , Y и Z и температуры, а затем печатает значение q . Используйте скрипт для вычисления результатов для $T_1 = 400$ К, $T_2 = 600$ К, $a = 1$ м., $b = 2$ м и $c = 0.1, 1$ и 10 м.

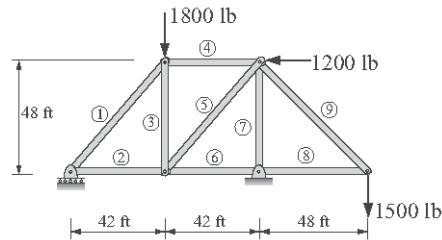
- 21.** Для данных координат трех точек (x_1, y_1) , (x_2, y_2) и (x_3, y_3) можно найти координаты центра круга (C_x, C_y) , который проходит через эти три точки, решением следующей системы уравнений:

$$2 \begin{bmatrix} (x_1 - x_2) & (y_1 - y_2) \\ (x_2 - x_3) & (y_2 - y_3) \end{bmatrix} \begin{bmatrix} C_x \\ C_y \end{bmatrix} = \begin{bmatrix} (x_1^2 + y_1^2) - (x_2^2 + y_2^2) \\ (x_2^2 + y_2^2) - (x_3^2 + y_3^2) \end{bmatrix}.$$

Напишите программу в файле сценария, которая вычисляет координаты центра и радиус круга, который проходит через три данные точки. При выполнении программы предлагает пользователю ввести координаты трех точек. Эта программа вычисляет центр и радиус и выводит на экран результаты в следующем формате: «Координаты центра есть (xx.x, xx.x) и радиус есть xx.x.», где xx.x обозначает расчетные величины, округленные до «десятых». Выполните программу, вводя следующие три точки: (10.5, 4), (2, 8.6) и (-4, -7).

- 22.** Ферма – это структура, сделанная из элементов, соединенных в их концах. Для фермы, показанной в рисунке, силы в этих девяти элементах определяются решением следующей системы девяти уравнений:

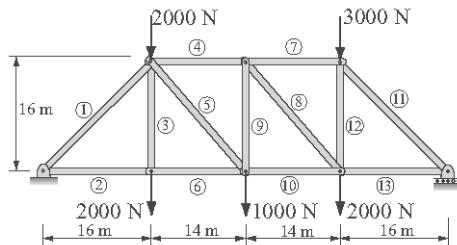
$$\begin{aligned}
 F_2 + \cos(48.81^\circ) F_1 &= 0, \\
 F_6 + \cos(48.81^\circ) F_5 - F_2 &= 0, \\
 \sin(48.81^\circ) F_5 + F_3 &= 0, \\
 -\cos(48.81^\circ) F_1 + F_4 &= 0, \\
 -\sin(48.81^\circ) F_1 + F_3 &= 1800, \\
 -F_4 - \cos(48.81^\circ) F_5 &= 1200, \\
 -F_7 - \sin(48.81^\circ) F_5 - \sin(45^\circ) F_9 &= 0, \\
 \sin(45^\circ) F_9 &= 1500, \\
 -\cos(45^\circ) F_9 - F_8 &= 0.
 \end{aligned}$$



Запишите уравнения в матричной форме и используйте MATLAB для определения силы в элементах. Положительная сила означает силу растяжения, а отрицательная сила означает силу сжатия. Выведите на экран результаты в таблице, где первый столбец – число элемента, а второй столбец – соответствующая сила.

- 23.** Ферма – это структура, сделанная из элементов, соединенных в их концах. Для фермы, показанной в рисунке, силы в этих 13 элементах определяются решением следующей системы 13 уравнений:

$$\begin{aligned}
 F_2 + 0.7071 F_1 &= 0, \\
 -F_2 + F_6 &= 0, \\
 F_3 - 2000 &= 0, \\
 F_4 + 0.5685 F_5 - 0.7071 F_1 &= 0, \\
 0.7071 F_1 + F_3 + 0.7526 F_5 + 2000 &= 0, \\
 F_7 + 0.6585 F_8 - F_4 &= 0, \\
 0.7526 F_8 + F_9 &= 0, \\
 F_{10} - 0.6585 F_5 - F_6 &= 0, \\
 F_9 + 0.7526 F_5 - 1000 &= 0, \\
 0.7071 F_{11} - F_7 &= 0, \\
 0.7071 F_{11} + F_{12} + 3000 &= 0, \\
 F_{12} + 0.7526 F_8 - 2000 &= 0, \\
 F_{13} + 0.7071 F_{11} &= 0.
 \end{aligned}$$

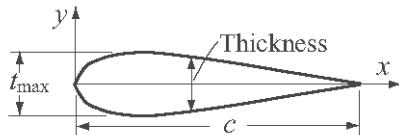


Запишите уравнения в матричной форме и используйте MATLAB для определения силы в элементах. Положительная сила означает силу растяжения, а отрицательная сила означает силу сжатия. Выведите на экран результаты в таблице, где первый столбец – число элемента, а второй столбец – соответствующая сила.

- 24.** График функции $f(x) = ax^3 + bx^2 + cx + d$ проходит через точки $(-2.6, -68)$, $(0.5, 5.7)$, $(1.5, 4.9)$ и $(3.5, 88)$. Определить константы a , b , c и d . (Записать систему четырех уравнений с четырьмя неизвестными и используйте MATLAB для решения уравнений.)

25. Поверхность многих крыльев может быть описана уравнением вида

$$y = \mp \frac{tc}{0.2} \left[a_0 \sqrt{x/c} + a_1 (x/c) + a_2 (x/c)^2 + a_3 (x/c)^3 + a_4 (x/c)^4 \right]$$



где t – максимальная толщина как функция длины хорды c (например, $t_{\max} = ct$).

При $c = 1$ м и $t = 0.2$ м были измерены следующие значения y для некоторого крыла:

x (м)	0.15	0.35	0.5	0.7	0.85
y (м)	0.08909	0.09914	0.08823	0.06107	0.03421

Определить константы a_0 , a_1 , a_2 , a_3 и a_4 . (Записать систему пяти уравнений с пятью неизвестными и использовать MATLAB для ее решения.)

26. Во время матча по гольфу определенное число очков присуждается за каждого орла и другое число – за каждую «птичку» (берди). Очки не присуждаются за пары и определенное число очков вычитается за каждого богги и другое число очков вычитается за каждого двойного богги (или хуже). Газетный отчет важного матча забыл упомянуть, каковы были эти значения очков, но предоставил следующую таблицу результатов:

Игрок	Орлы	Птички	Пары	Богги	Дубли	Очки
A	1	2	10	1	1	5
B	2	3	11	0	1	12
C	1	4	10	1	10	11
D	1	3	10	12	0	8

Исходя из информации в таблице написать четыре уравнения с четырьмя неизвестными. Решить эти уравнения для неизвестных величин очков, присуждаемых за орлов и пташек и очков, вычитаемых за богги и двойные богги.

27. Растворение медного сульфида в водной азотной кислоте описывается следующим химическим уравнением:



где коэффициенты a , b , c , d , e , f и g являются числами различных молекул, участвующих в реакции и они являются неизвестными. Неизвестные коэффициенты определяются балансом каждого атома слева и справа и последующим балансом ионного заряда. Получающиеся уравнения:

$$a = d, \quad a = e, \quad b = f, \quad 3b = 4e + f + g, \quad c = 2g, \quad -b + c = 2d - 2e.$$

Имеется семь неизвестных и только шесть уравнений. Однако решение может быть получено, используя дополнительно тот факт, что все коэффициенты должны быть положительными целыми числами. Добавьте седьмое уравнение, предполагая $a = 1$ и решите систему уравнений. Решение является подходящим, если все коэффициенты – положительные целые числа. Если дело обстоит не так, возьмите $a = 2$ и повторите решение. Продолжайте процесс, пока все коэффициенты в решении не будут положительными целыми числами.

- 28.** Температура холода ветра, T_{wc} , является температурой воздуха, которую чувствуют кожей подставленной ветру. В американских общепринятых единицах она вычисляется по формуле:

$$T_{wc} = 35.74 + 0.6215 T - 35.75 v^{0.16} + 0.4275 T v^{0.16},$$

где T – температура в градусах по Фаренгейту и v – скорость ветра в миль/час.

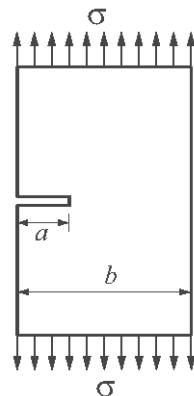
Напишите программу MATLAB в файле сценария, которая выводит на экран в командном окне следующую диаграмму температуры холода ветра для данной температуры (в строке) и скорости ветра (в первом столбце):

Temperature (F)									
	40	30	20	10	0	-10	-20	-30	-40
Speed (mi/h)									
10	34	21	9	-4	-16	-28	-41	-53	-66
20	30	17	4	-9	-22	-35	-48	-61	-74
30	28	15	1	-12	-26	-39	-53	-67	-80
40	27	13	-1	-15	-29	-43	-57	-71	-84
50	26	12	-3	-17	-31	-45	-60	-74	-88
60	25	10	-4	-19	-33	-48	-62	-76	-91

- 29.** Коэффициент интенсивности напряжения K в трещине задается формулой $K = C\sigma\sqrt{\pi a}$, где σ – напряжение на бесконечности, a – длина трещины и C – параметр, который зависит от геометрии экземпляра и трещины. Для случая граничной трещины показанной на рисунке, C задается:

$$C = 0.256 \left(1 - \frac{a}{b}\right) + \frac{0.857 + 0.265 a/b}{(1 - a/b)^{3/2}}.$$

Напишите скрипт-файл, который печатает таблицу значений отношений a/b в первом столбце и соответствующий параметр C – во втором столбце. Пусть a/b меняется в диапазоне между 0 и 0.95 с инкрементом 0.05.





ГЛАВА 5.

Двумерные графики

Графики – это очень полезный инструмент для представления информации. Это справедливо в любой области, но особенно в науке и разработке, где MATLAB главным образом используется. У MATLAB есть много команд, которые могут использоваться для создания различных типов графических изображений графиков. Они включают стандартные графики с линейными осями, графики с логарифмическими и полулогарифмическими осями, бары и ступенчатые графики, в полярных координатах, трехмерные контурные поверхности и сеточные графики и еще много других. Графики могут быть отформатированы, чтобы иметь требуемый вид. Может быть указан тип линии (сплошная, пунктириная линия и т. д.), цвет и толщина, могут быть добавлены маркеры линии и линии сетки, а также текстовые комментарии и заголовки. Несколько графиков могут быть созданы в одном и том же графическом окне и несколько окон могут быть помещены на одной странице. Когда график содержит несколько диаграмм и/или точек данных, может быть добавлена также легенда к графику.

Эта глава описывает, как может использоваться MATLAB для создания и форматировать многих типов двумерных графиков. Трехмерные графики будут рассмотрены отдельно в главе 9. Пример простого двумерного графика, который создан с MATLAB, показан на рис. 5.1. Этот рисунок содержит две кривые, которые показывают изменение интенсивности света с расстоянием. Одна кривая создана из точек экспериментальных данных, а другая кривая показывает изменение света как предсказано теоретической моделью. Обе оси на рисунке линейны и для кривых используются различные типы линий (одна сплошная, а другая – пунктирная линия). Теоретическая кривая показана сплошной линией, а экспериментальные точки соединены с пунктирной линией. Каждая точка данных отмечена круговым маркером. Пунктирная линия, которая соединяет экспериментальные точки, является фактически красной при выводе на экран в окне графиков Figure. Как видно на рис. 5.1, график отформатирован так, что имеет заголовок, заголовки осей, легенду, маркеры и текстовое поле с поясняющим текстом.

5.1. Команда `plot`

Команда `plot` используется для создания графического изображения двумерных графиков. Самая простая форма команды:

`plot(x, y)`

Вектор Вектор

Каждый из аргументов x и y – вектор (одномерный массив). У этих двух векторов должно быть одно и то же число элементов. При выполнении команды `plot` рисунок создается в окне **Figure** графического изображения графиков. Если окно графиков **Figure** еще открыто, то оно открывается автоматически при выполнении команды. Рисунок имеет единственную кривую со значениями x на оси абсцисс (горизонтальная ось) и значениями y на оси ординат (вертикальная ось). Кривая создана из прямолинейных сегментов, которые соединяют точки, координаты которых определены элементами векторов x и y . У каждого из векторов, конечно, может быть любое имя. Вектор, который введен первым в команде `plot`, используется для горизонтальной оси, а вектор, который введен вторым, используется для вертикальной оси. Если в команде `plot` вводится только один вектор как входной аргумент (например, `plot(y)`), тогда отображается график значений элементов вектора ($y(1), y(2), y(3), \dots$), которые соответствуют значениям аргумента x , равным $(1, 2, 3, \dots)$.

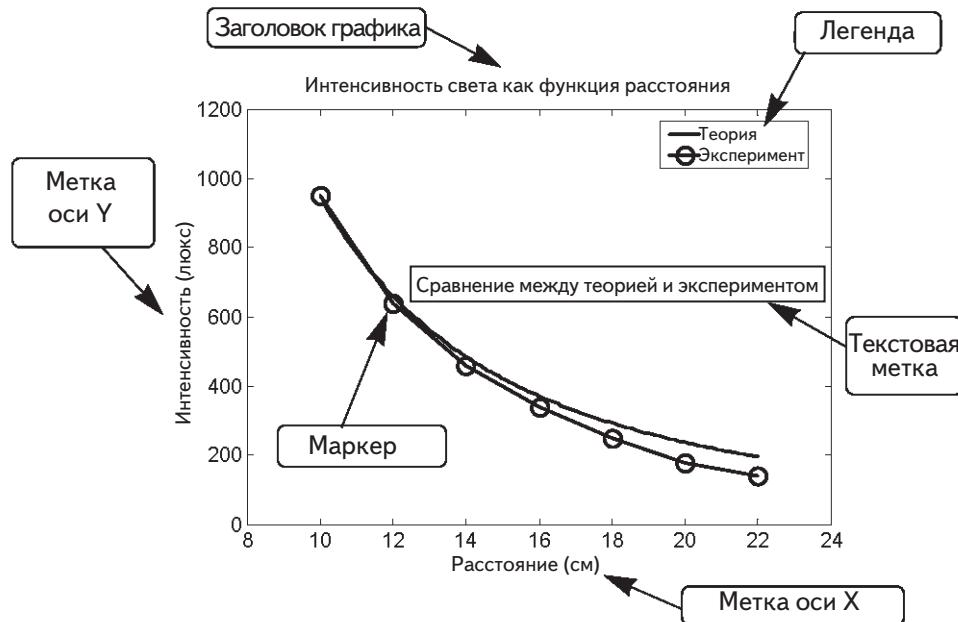


Рис. 5.1. Пример отформатированного двумерного графика

У рисунка, который создается, имеются оси с диапазонами значений по умолчанию и линейной шкалой. Например, если вектор x имеет элементы 1, 2, 3, 5, 7, 7.5, 8, 10, а вектор y имеет элементы 2, 6.5, 7, 7, 5.5, 4, 6, 8, тогда простой график y как функции от x может быть создан следующим образом в командном окне:

```
>> x=[1.1 1.8 3.2 5.5 7 7.5 8 10];
>> y=[2 6.5 7 7 5.5 4 6 8];
>> plot (x,y)
```

При исполнении команды `plot` открывается графическое окно `Figure` и график выводится на экран, как показано на рис. 5.2.

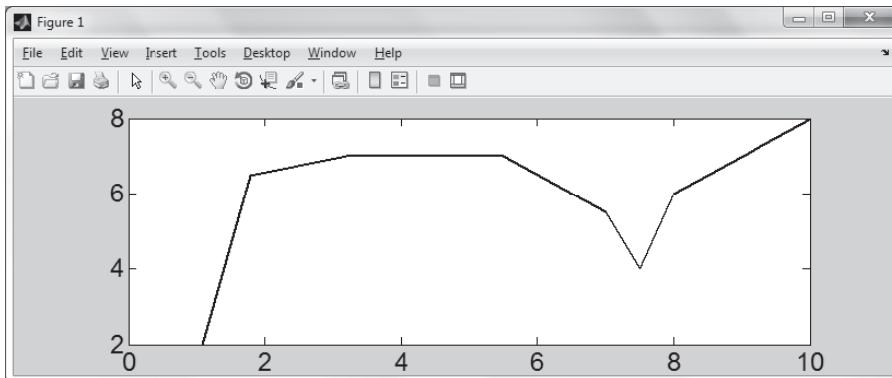


Рис. 5.2. Графическое окно `Figure` с простым графиком

График появляется на экране в синем цвете, который является значением по умолчанию цвета линии.

У команды `plot` есть дополнительные необязательные аргументы, которые могут использоваться для определения цвета и стиля линии, цвета и типа маркеров, если это требуется. Форма команды с этими опциями:

```
plot(x,y,'line specifiers','PropertyName',PropertyValue)
          ↑           ↑           ↑           ↑           ↑
          Вектор     Вектор   (Необязательные) Специфи- (Необязательные) Свойства со
          тип и цвет линии и маркеров   каторы, которые определяют значениями, которые могут ис-
          толщины линии и размера мар- пользоваться для определения
          кера, границы и цвета заливки
```

Спецификаторы линии

Спецификаторы линии являются дополнительными и могут использоваться для определения стиля и цвета линии и типа маркеров (если маркеры требуются). Спецификаторы стиля линии:

Стиль линии	Спецификатор
Сплошная (по умолчанию)	-
Штрихованый	--

Стиль линии	Спецификатор
Пунктирный	:
Штрихпунктир	-.

Спецификаторы цвета линии:

Цвет линии	Спецификатор
красный (red)	r
зеленый (green)	g
голубой (blue)	b
синий (cyan)	c

Цвет линии	Спецификатор
пурпурный (magenta)	m
желтый (yellow)	y
черный (black)	k
белый (white)	w

Спецификаторы типа маркера:

Тип маркера	Спецификатор
знак «плюс»	+
кружок	o
звездочка	*
точка	.
крестик	x
треугольник (острием вверх)	^
треугольник (острием вниз)	v

Тип маркера	Спецификатор
квадрат	s
ромб	d
пятиконечная звезда	p
шестиконечная звезда	h
треугольник (острием налево)	<
треугольник (острием направо)	>

Замечания об использовании спецификаторов

- Спецификаторы вводятся в команде `plot` как строки (string).
- В пределах строки команды спецификаторы могут быть введены в любом порядке.
- Спецификаторы являются дополнительными. Это означает, что в команду может быть не включено ни одного, или включен один, два, или все три типа.

Некоторые примеры:

- `plot(x, y)` – синяя сплошная линия соединяет точки без маркеров (по умолчанию).
- `plot(x, y, 'r')` – красная сплошная линия соединяет точки.
- `plot(x, y, '--y')` – желтая пунктирная линия соединяет точки.
- `plot(x, y, '*')` – точки отмечены как * (между точками нет никакой линии).
- `plot(x, y, 'g:d')` – зеленая пунктирная линия соединяет точки, которые отмечены ромбовидными маркерами.

Имя свойства (*Property Name*) и значение свойства (*Property Value*)

Свойства являются дополнительными и могут использоваться для заданиятолщины (ширины) линии, размера маркера, цвета граничной линии маркера и заливки. Имя свойства (*Property Name*) вводится как строка, сопровождаемая запятой и значением для свойства – все внутри команды `plot`.

Четыре свойства и их возможные значения:

Имя свойства	Описание	Возможные значения свойства
<code>LineWidth</code> (или <code>linewidth</code>)	Определяет толщину (ширину) линии.	Число в единицах пунктов (значение по умолчанию 0.5).
<code>MarkerSize</code> (или <code>markersize</code>)	Определяет размер маркера.	Число в единицах пунктов.
<code>MarkerEdgeColor</code> (или <code>markeredgecolor</code>)	Определяет цвет маркера, или цвет граничной линии для заполненных маркеров.	Спецификаторы цвета из таблицы выше, введенныекак строка.
<code>MarkerFaceColor</code> (или <code>markerfacecolor</code>)	Определяет цвет заливки для заполненных маркеров.	Спецификаторы цвета из таблицы выше, введенныекак строка.

Например, команда

```
plot(x, y, '-mo', 'LineWidth', 2, 'markersize', 12,
      'MarkerEdgeColor', 'g', 'markerfacecolor', 'y')
```

создает график, который соединяет точки пурпурной сплошной линией и кругами в качестве маркеров в точках. Толщина линии – 2 пункта, а размер маркеров круга – 12 пунктов. У маркеров зеленая граничная линия и желтое заполнение.

Замечание о спецификаторах линии и их свойствах

Три спецификатора линии, которые указывают на стиль, цвет линии и тип маркера, могут также быть присвоены аргументу `PropertyName`, с последующим аргументом `PropertyValue`. Имена свойств (*Property Names*) спецификаторов линии:

Спецификатор	Имя свойства	Возможные значения свойства
Стиль линии	<code>linestyle</code> (или <code>LineStyle</code>)	Спецификатор стили линии из таблицы выше, введенный как строка (<code>string</code>).
Цвет линии	<code>color</code> (или <code>Color</code>)	Спецификатор цвета из таблицы выше, введенный как строка.
Маркер	<code>marker</code> (или <code>Marker</code>)	Спецификатор маркера из таблицы выше, введенный как строка.

Как и любая другая, команда `plot` может быть введена в командном окне, или может быть включена в файл сценария. Она может также использоваться в файле функции (это объяснено в главе 7). Нужно также помнить о том, что перед выполнением команды `plot` векторам `x` и `y` должны быть присвоены элементы. Это может быть сделано так, как было объяснено в главе 2, либо непосредственно введением значений, либо использованием команд, или как результат математических операций. Следующие два подраздела показывают примеры создания простых графиков.

5.1.1. График определенных данных

В этом случае используются уже определенные данные для создания векторов, которые будут использоваться в команде `plot`. Следующая таблица содержит данные о сбыте компании с 1988 до 1994.

Год	1988	1989	1990	1991	1992	1993	1994
Продажи (в миллионах)	8	12	20	22	18	24	27

Чтобы графически изобразить этих данных, присваиваем список лет одному вектору (с именем `yr`), а соответствующие данные о сбыте присвоим второму вектору (с именем `sle`). Ниже показано Командное окно, где создаются векторы и используется команда `plot`:

```
>> yr=[1988:1:1994];
>> sle=[8 12 20 22 18 24 27];
>> plot(yr,sle,'--r*','linewidth',2,'markersize',12)
    Спецификаторы линии:  
штриховая красная линия  
и маркер звездочки.           Имя свойства и значение свойства:  
шрина линии — 2 точки и  
размер маркера — 12 пунктов.
```

После исполнения команды `plot` открывается графическое окно Figure с графиком, как показано на рис. 5.3. График появляется на экране в красном.

5.1.2. График функции

В многих ситуациях требуется изобразить график заданной функции. Это может быть сделано в MATLAB с использованием команд `plot` или `fplot`. Использование команды `plot` объяснено ниже. Команда `fplot` подробно рассматривается в следующем разделе.

Чтобы построить график функции $y = f(x)$ командой `plot`, пользователь должен сначала создать вектор значений переменной `x` для области, над которой будет графически изображена функция. Затем создается вектор `y` с соответствующими значениями функции с использованием поэлементных вычислений (см. главу 3). Как только эти два вектора определены, они могут использоваться в команде `plot`.

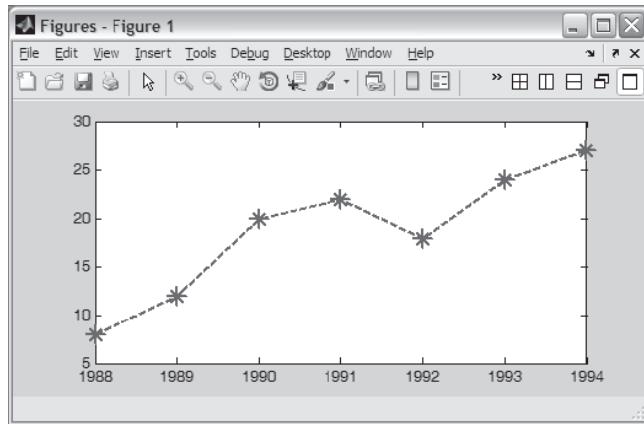


Рис. 5.3. Графическое окно **Figure** с графиком данных о событии

В качестве примера, используем команду `plot` используется для построения графика функции $y = 3.5^{-0.5x} \cos(6x)$, где $-2 \leq x \leq 4$. Программа, которая графически изображает эту функцию, показана в следующем скрипте-файле.

```
% A script file that creates a plot of
% the function: 3.5.^(-0.5*x).*cos(6x)
x=[-2:0.01:4];           Создание вектора x области определения функции.
y=3.5.^(-0.5*x).*cos(6*x);   Создание вектора y значений функции
plot(x,y)                 График y как функции от x.
```

После выполнения скрипта-файла создается график в графическом окне Figure, как показано в рис. 5.4.

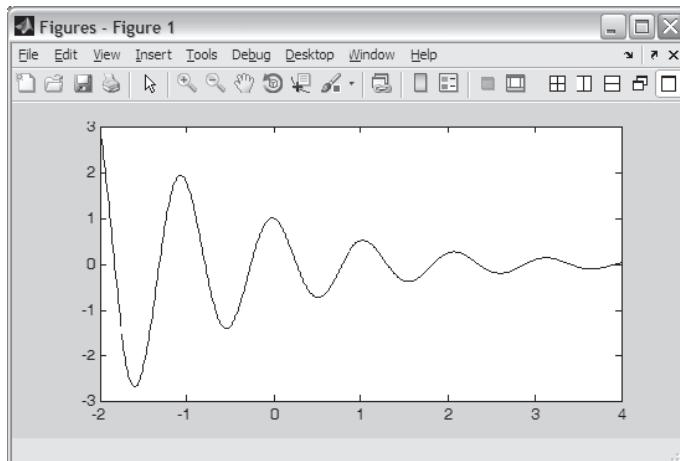


Рис. 5.4. Графическое окно **Figure** с графиком функции $y = 3.5^{-0.5x} \cos(6x)$.

Так как график составлен из сегментов прямых, которые соединяют точки, то для получения более точного графика функции, интервал между элементами вектора x должен быть соответствующим. Для функции, которая изменяется быстро, необходим меньший интервал. В последнем примере небольшой интервал 0.01 между значениями аргумента x произвел график, который показан на рис. 5.4. Однако, если та же самая функция в той же области $-2 \leq x \leq 4$ графически изображается с намного большим интервалом между значениями аргумента, например, 0.3, то получающийся график, который показан на рис. 5.5, дает искаженное изображение функции.

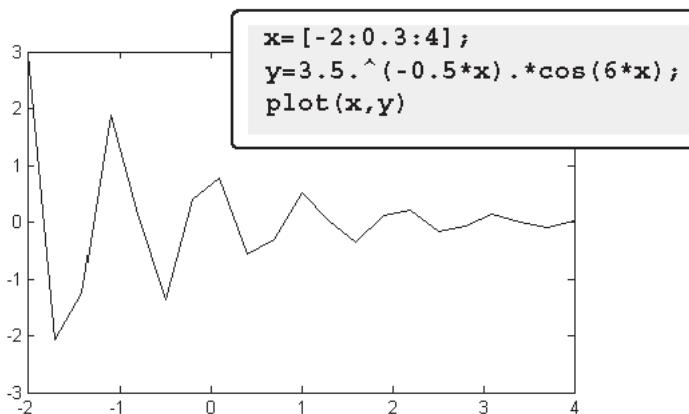


Рис. 5.5. График функции $y = 3.5^{-0.5x} \cos(6x)$ с большим интервалом

Отметим также, что на рис. 5.4 график показан в окне графиков Figure, в то время как на рис. 5.5 показан только график. Этот график может быть скопирован из окна графиков Figure (в меню **Edit**, выберите **Copy Figure**) и затем вставлен в другие приложения.

5.2. Команда **fplot**

Команда **fplot** изображает график функции $y = f(x)$ с указанными пределами изменения аргумента. Команды имеет вид:

```
plot('function',limits,'line specifiers')
```

↑ ↑ ↑
 Функция Область изменения Спецификаторы, которые
 для графика x и, необязательно,
 пределы по оси y . определяют тип и цвет
 линии и маркеров.

'**function**': Функция может быть введена в команде непосредственно как строка. Например, если графически изображается функция $y = 8x^2 + 5\cos(x)$, тогда она

вводится как ' $8*x^2+5*cos(x)$ '. Функции могут быть встроенными функциями MATLAB и функциями, которые создаются пользователем (об этом см. главу 6).

- При построении графика аргумент у функции может быть обозначен любым символом. Например, функция в предыдущем абзаце может быть введена как ' $8*z^2+5*cos(z)$ ' или ' $8*t^2+5*cos(t)$ '.
- Стока 'function' не может включать ранее определенные переменные. Например, в функции выше этого невозможно присвоить значение 8 переменной, а потом использовать эту переменную в строке ввода функции в команде `fplot`.

limits: Пределы изменения аргумента – это вектор с двумя элементами $[x_{\min}, x_{\max}]$, которые определяют область определения x , или вектор с четырьмя элементами $[x_{\min}, x_{\max}, y_{\min}, y_{\max}]$, который определяет область определения x и пределы по оси y .

'line specifiers': Спецификаторы линии – такие же, как и команде `plot`.

Например, график функции $y = x^2 + 4\sin(2x) - 1$ для $-3 \leq x \leq 3$ может быть создан командой `fplot`, следующей строкой в командном окне:

```
>> fplot('x^2+4*sin(2*x)-1', [-3 3]).
```

Изображение, которое получается в окне графиков Figure, показано на рис. 5.6.

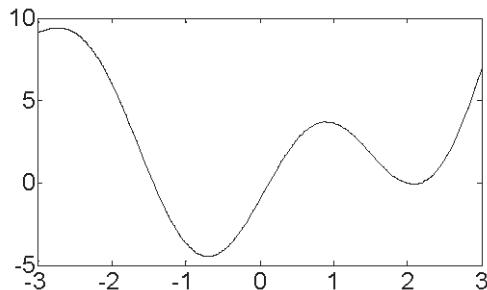


Рис. 5.6. График функции $y = x^2 + 4\sin(2x) - 1$

5.3. Графическое изображение нескольких графиков в одном окне

Во многих ситуациях требуется создать несколько графиков в одном и том же окне. Это показывает, например, рис. 5.1, где изображены два графика. Есть три метода для изображения нескольких графиков в одном окне. Один из них состоит в использовании команды `plot`, второй – использует команды `hold on` и `hold off`, а третий – использованием команды `line`.

5.3.1. Использование команды *plot*

Два или более графиков могут быть созданы в одном окне, если ввести пары векторов в команде *plot*. Например, команда

```
plot(x,y,u,v,t,h)
```

создает три графика: y как функция от x , v от u и h от t – все в одном и том же окне. Векторы каждой пары должны иметь одну и ту же длину. MATLAB автоматически изображает графики в разных цветах так, чтобы они могли быть легче идентифицированы. Также возможно добавить спецификаторы линии после каждой пары. Например команда

```
plot(x,y,'-b',u,v,'--r',t,h,'g:')
```

изображает y от x сплошной синей линией, v от u – штрихованной красной линией и h от t – пунктирной зеленой линией.

Пример задачи 5.1. График функции и ее производных

Построить график функции $y = 3x^3 - 26x + 10$ и графики ее первой и второй производных для $-2 \leq x \leq 4$, все в одном окне.

Решение

Первая производная функции: $y' = 9x^2 - 26$.

Вторая производная функции: $y'' = 18x$.

Скрипт-файл, который создает вектор x и вычисляет значения y , y' и y'' :

<code>x=[-2:0.01:4];</code>	Создание вектора значений аргумента x .
<code>y=3*x.^3-26*x+6;</code>	Создание вектора y значений функции.
<code>yd=9*x.^2-26;</code>	Создание вектора yd значений первой производной.
<code>ydd=18*x;</code>	Создание вектора ydd значений второй производной.
<code>plot(x,y,'-b',x,yd,'--r',x,ydd,:k')</code>	Создание трех графиков y от x , yd от x и ydd от x в одном и том же окне.

Созданное изображение показано на рис. 5.7.

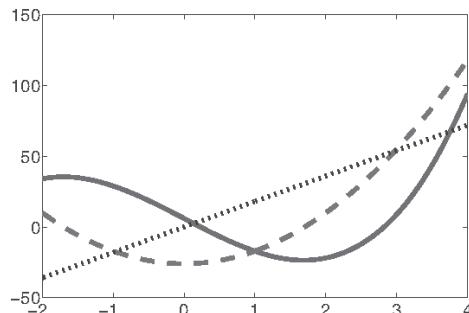


Рис. 5.7. Графики функции $y = 3x^3 - 26x + 10$ и ее первой и второй производных

5.3.2. Использование команд *hold on* и *hold off*

Чтобы изобразить несколько графиков в одном окне, используя команды удержания *hold on* и *hold off*, сначала нужно командой *plot* построить один график. Затем ввести команду *hold on*. Она сохраняет открытым окно графиков Figure с первым графиком, включая свойства осей и форматирующие свойства (см. раздел 5.4), если они были сделан. Дополнительные графики могут быть добавлены последующими командами *plot*. Каждая команда *plot* создает график, который добавляется к тому же рисунку. Команда *hold off* останавливает этот процесс. Она возвращает MATLAB в режим по умолчанию, при котором команда *plot* стирает предыдущий график и вновь устанавливает свойства оси.

В качестве примера использования команд *hold on* и *hold off*, в следующем скрипте-файле показано решение примера задачи 5.1

```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'-b')
hold on
plot(x,yd,'--r')
plot(x,ydd,:k')
hold off
```

Создание первого графика.

Добавлены еще два графика.

5.3.3. Использование команды *line*

Командой *line* могут быть добавлены дополнительные графики (линии) к уже существующему графику. Вид команды линии:

```
line(x,y,'PropertyName',PropertyValue)
```

(Опционально). Свойства со значениями, которые могут использоваться для определения стиля линии, цвета и ширины, типа маркера, размера и цвета его границы и заливки.

Формат команды *line* – почти такой же, как у команды *plot* (см. раздел 5.1). У команды *line* нет спецификаторов линии, но стиль линии, цвет и маркер могут быть определены при помощи свойств *PropertyName* и *PropertyValue*. Эти свойства являются дополнительными и если ни один из них не вводится, MATLAB использует свойства и значения по умолчанию. Например, команда:

```
line(x,y,'linestyle','--','color','r','marker','o')
```

добавляет штрихованную красную линию с круговыми маркерами к уже существующему графику.

Существенное различие между командами `plot` и `line` в том, что команда `plot` запускает новый график каждый раз, когда она выполняется, в то время как команда `line` добавляет новые линии к уже существующему графику. Для того, чтобы сделать рисунок с несколькими графиками, сначала вводится команда `plot`, а затем вводятся команды `line` для дополнительных графиков. (Если команда `line` вводится перед командой `plot`, на экран выводится сообщение об ошибке.)

Решение задачи 5.1, которое изображено на рис. 5.7, может быть получено при использовании команд `plot` и `line`, как показано в следующем файле сценария:

```
x=[-2:0.01:4];
y=3*x.^3-26*x+6;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'LineStyle','-', 'color','b')
line(x,yd,'LineStyle','--', 'color','r')
line(x,ydd,'linestyle',':', 'color','k')
```

5.4. Форматирование окна графика

Команды `plot` и `fplot` создают пустые графики. Однако обычно рисунок, который содержит график, должен быть отформатирован для определенного вида и вывода на экран информации в дополнение непосредственно к графику. Это может включать задание обозначений осей, заголовка графика, легендры, сетки, диапазона пользовательской оси и текстовых меток.

Окна с графиками могут быть отформатированы при использовании команд MATLAB, которые следуют за командами `plot` и `fplot`, или в интерактивном режиме с использованием редактора графика в графическом окне `Figure`. Первый метод полезен, когда команда `plot` – это часть компьютерной программы (файла сценария). Если команды форматирования включены в программу, то отформатированный рисунок создается каждый раз, когда выполняется программа. С другой стороны, форматирование после построения графика, которое выполнено в графическом окне `Figure` с использованием редактора графика, сохраняется только в данном определенном окне, а в дальнейшем не сохраняется и должно быть повторено в следующий раз, когда график создается снова.

5.4.1. Использование команд форматирования окна графика

Команды форматирования вводятся после команды `plot` или `fplot`. Различные команды форматирования:

Команды `xlabel` и `ylabel`

Обозначения осей могут быть помещены рядом с осями командами `xlabel` и `ylabel`, которые имеют вид:

```
xlabel('text as string')
ylabel('text as string')
```

Команда **title**

Заголовок может быть добавлен к графику командой:

```
title('text as string')
```

Текст помещается вверху окна графика как заголовок.

Команда **text**

Текстовая метка может быть помещена на окно графика командами `text` или `gtext`:

```
text(x,y,'text as string')
gtext('text as string')
```

Команда `text` помещает указанный текст в рисунок так, что, первый символ расположен в точке с координатами `x`, `y` (относительно осей рисунка). Команда `gtext` помещает текст в позицию, указанную пользователем. Когда выполняется эта команда, открывается окно графиков `Figure` и пользователь определяет позицию мышкой.

Команда **legend**

Команда `legend` помещает в окно графика легенду. Легенда показывает список типов линий изображенных графиков и помещает обозначение, определенное пользователем, около каждой линии. Вид команды:

```
legend('string1','string2', .... ,pos)
```

Строки `string` – это обозначения, которые помещаются рядом с соответствующей линией. Их порядок соответствует порядку, в котором создавались графики. Параметр `pos` есть дополнительное число, которое определяет, куда должна быть помещена легенда на рисунке. Возможны следующие варианты:

`pos = -1` – Помещает легенду вне границ осей на правой стороне.

`pos = 0` – Помещает легенду в границах осей в положение, которое наименее мешает графикам.

`pos = 1` – Помещает легенду в верхний правый угол окна графика (значение по умолчанию).

`pos = 2` – Помещает легенду в верхний левый угол окна графика.

`pos = 3` – Помещает легенду в нижний левый угол окна графика.

`pos = 4` – Помещает легенду в нижний правый угол окна графика.

Форматирование текста в рамках команд ***xlabel***, ***ylabel***, ***title***, ***text*** и ***legend***

Текст в строке, которая включена в команду и будет выведена на экран при выполнении команды, может быть отформатирован. Форматирование может использоваться для определения шрифта, размера, положения (верхний индекс, нижний индекс), стиля (курсив, полужирный, и т. д.), цвета символов, цвета фона

и много других деталей. Ниже описаны некоторые из наиболее распространенных возможностей форматирования. Полный перечень всех функций форматирования может быть найден в окне **Help**, в разделах **Text** и **Text Properties**. Форматирование может быть выполнено либо добавлением модификатора внутри строки, либо добавлением к команде дополнительных аргументов `PropertyName` и `PropertyValue` после строки.

Модификаторы – это символы, которые вставляются в пределах строки. Некоторые из модификаторов, которые могут быть использованы:

Модификатор	Действие	Модификатор	Действие
<code>\bf</code>	полужирный шрифт	<code>\fontname{fontname}</code>	Использовать указанный шрифт
<code>\it</code>	курсив	<code>\fontsize{fontsize}</code>	Использовать указанный размер шрифта
<code>\rm</code>	нормальный		

Эти модификаторы действуют на текст от места, в котором они вставлены и до конца строки. Можно также применить модификаторы только для части строки, если поместить в фигурные скобки {} модификатор и текст, на который он будет влиять.

Нижний и верхний индексы (*subscript* и *superscript*)

Одиночный символ может быть выведен на экран как нижний или верхний индекс, если ввести `_` (символ подчеркивания) или `^` перед символом-индексом, соответственно. Несколько последовательных символов могут быть выведены на экран как нижний или верхний индекс, если эти символы поместить в фигурных скобках {} после `_` или `^`.

Греческие символы

Греческие символы могут быть включены в текст, вводя команду символа `\name` в пределах строки. Чтобы вывести на экран строчную греческую букву, имя символа должно быть введено во всех строчных английских символах. Чтобы вывести на экран греческую букву прописной буквы, имя символа должно начинаться с прописной буквы. Некоторые примеры:

Символы в строке	Греческая буква	Символы в строке	Греческая буква
<code>\alpha</code>	α	<code>\Phi</code>	Φ
<code>\beta</code>	β	<code>\Delta</code>	Δ
<code>\gamma</code>	γ	<code>\Gamma</code>	Γ
<code>\theta</code>	θ	<code>\Lambda</code>	Λ
<code>\pi</code>	π	<code>\Omega</code>	Ω
<code>\sigma</code>	σ	<code>\Sigma</code>	Σ

Форматирование текста, который выводится на экран командами `xlabel`, `ylabel`, `title` и `text` может также быть сделано использованием дополнительных аргументов `PropertyName` и `PropertyValue` после строки в команде. С этой опцией у текста команды, например, имеет вид:

```
text(x,y,'text as string',PropertyName,PropertyValue)
```

В трех других командах аргументы `PropertyName` и `PropertyValue` добавляются таким же образом. `PropertyName` вводится как строка, а `PropertyValue` – как число, если значение свойства – число, и как строка, если значение свойства – слово или буквенный символ. Ниже указаны некоторые из имен свойств и соответствующих возможных значений свойств:

Имя свойства	Описание	Возможные значения свойства
<code>Rotation</code>	Определяет ориентацию текста.	Скаляр (градусы) По умолчанию: 0
<code>FontAngle</code>	Определяет стиль символов курсив или нормальный.	<code>normal</code> , <code>italic</code> По умолчанию: <code>normal</code>
<code>FontName</code>	Определяет шрифт для текста.	Имя шрифта доступное в системе.
<code>FontSize</code>	Определяет размер шрифта.	Скаляр (пункты) По умолчанию: 10
<code>FontWeight</code>	Определяет полноту символов.	<code>light</code> , <code>normal</code> , <code>bold</code> По умолчанию: <code>normal</code>
<code>Color</code>	Определяет цвет текста.	Спецификаторы цвета (см. раздел 5.1).
<code>Background-Color</code>	Определяет цвет фона (прямоугольная область).	Спецификаторы цвета (см. раздел 5.1).
<code>EdgeColor</code>	Определяет цвет края прямоугольника вокруг текста.	Спецификаторы цвета (см. раздел 5.1). По умолчанию: нет (<code>none</code>).
<code>LineWidth</code>	Определяет ширину края прямоугольника вокруг текста.	Скаляр (points) По умолчанию: 0.5

Команда `axis`

Когда выполняется команда `plot(x, y)`, MATLAB создает оси с пределами, которые основаны на минимальных и максимальных значениях элементов `x` и `y`. Команда `axis` может использоваться для изменения диапазона и отображения осей. Во многих ситуациях график выглядит лучше, если диапазон осей расширяется вне диапазона данных. Ниже – некоторые из возможных форм команды `axis`:

`axis([xmin, xmax, ymin, ymax])` – Устанавливает пределы обеих осей `x` и `y`
(`xmin`, `xmax`, `ymin` и `ymax` – это числа).

`axis equal` – Устанавливает одинаковый масштаб для обеих осей.

`axis square` – Устанавливает квадратной область осей.

`axis tight` – Устанавливает пределы осей по диапазонам данных.

Команда *grid*

grid on – Добавляет линии сетки в окно графика.

grid off – Удаляет линии сетки из окна графика.

Пример форматирования графика с использованием команд дан в следующем файле сценария, он использовался для создания отформатированного графика на рис. 5.1.

```

x=[10:0.1:22];
y=95000./x.^2;
xd=[10:2:22];
yd=[950 640 460 340 250 180 140];
plot(x,y,'-', 'LineWidth',1.0)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
title('\fontname{Arial}Light Intensity as a Function of Distance',
'FontSize',14)
axis([8 24 0 1200])
text(14,700,'Comparison between theory and experiment.', 'EdgeColor',
'r', 'LineWidth',2)
hold on
plot(xd,yd,'ro--','linewidth',1.0, 'markersize',10)
legend('Theory','Experiment',0)
hold off

```

5.4.2. Форматирование графика используя редактор графиков

График может быть отформатирован в интерактивном режиме в окне графиков Figure, щелкнув по окну графика и/или используя меню. Рис. 5.8 показывает графическое окно Figure с графиком рис. 5.1. Редактор графика может использоваться для введения новых элементов форматирования или изменения форматирования, которое было первоначально выполнено командами форматирования.

5.5. Графики с логарифмическими осями

В многих научных и технических приложениях требуются графики, в которых одна или обе оси имеют логарифмический (log) масштаб. Это удобное средство для представления данных с широким диапазоном значений. Логарифмический масштаб также обеспечивает средства идентификации характеристик данных и возможных форм математических соотношений, которые могут быть подходящими для моделирования данных (см. раздел 8.2.2).

Нажмите кнопку **стрелки**, чтобы запустить режим редактирования графика. Затем щелкните по элементу графического окна. Открывается окно с инструментами для форматирования элемента.

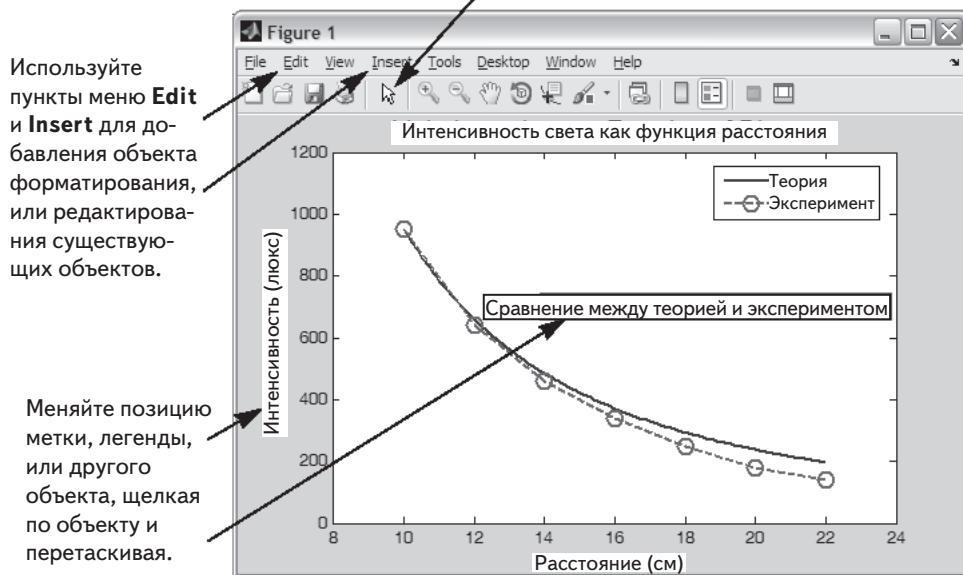


Рис. 5.8. Форматирование графика используя редактор графиков

Команды MATLAB для создания графиков с логарифмическим масштабом осей:

`semilogy(x, y)` – График y от x с логарифмическим (база 10) масштабом для оси y и линейным масштабом по оси x .

`semilogx(x, y)` – График y от x с логарифмическим (база 10) масштабом для оси x и линейным масштабом по оси y .

`semilog (x, y)` – График y от x с логарифмическим (база 10) масштабом по обеим осям.

Так же, как в команде `plot` могут быть добавлены (дополнительно) к командам спецификаторы линии и аргументы свойств имени и значений (*Property Name* и *Property Value*). Как пример, рис. 5.9 показывает график функции $y = 2^{(-0.2x + 10)}$ для $0.1 \leq x \leq 60$. Рисунок показывает четыре графика одной и той же функции: один с линейными осями, один с логарифмическим масштабом по оси y , один с логарифмическим масштабом по оси x и один с логарифмическим масштабом на обеих осях.

Примечания для графиков с логарифмическими осями

- Число нуль не может быть графически изображено в логарифмическом масштабе (так как логарифм нуля не определен).

- Отрицательные числа также не могут быть графически изображены (так как логарифм отрицательного числа не определен).

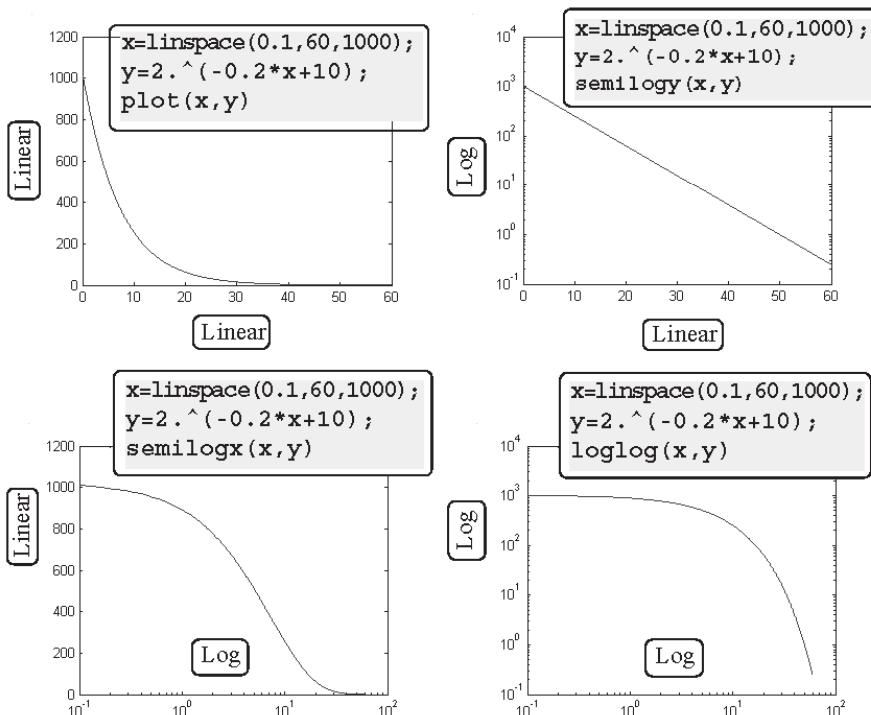


Рис. 5.9. Графики с линейным, полу-log и log-log масштабами

5.6. Графики с планками погрешностей

Экспериментальные данные, которые измерены и затем выведены на экран в виде графиков часто содержат ошибки и разброс значений. Даже данные, которые сгенерированы вычислительными моделями, включают ошибки или неточности, которые зависят от точности входных параметров и предположений в используемых математических моделях. Один из методов графического изображения данных с визуализацией ошибки, или неопределенности, заключается в использовании планок погрешностей (error bars). Значение погрешности – это обычно короткая вертикальная линия, которая присоединена к точке данных в графике. Она показывает величину ошибки, соответствующей значению, выведенной на экран точки данных. Например, рис. 5.10 показывает график со значением погрешности для экспериментальных данных из рис. 5.1.

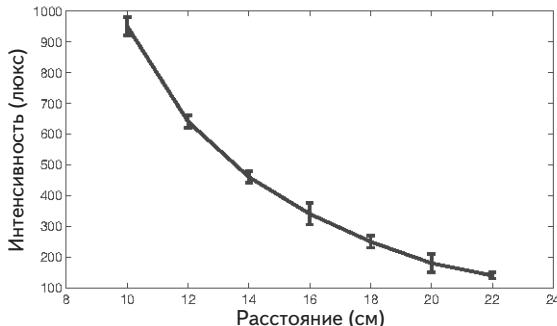


Рис. 5.10. График со значениями погрешностей

Графики со значениями погрешностей могут быть созданы в MATLAB командой `errorbar`. Имеются две формы команды: одна для того, чтобы создать график с симметричными значениями погрешностей (относительно значения точки данных) и другая – для несимметричных значений погрешностей в каждой точке. В случае симметричной ошибки планка погрешности распространяется на одну и ту же длину выше и ниже точки данных, и команда имеет вид

`errorbar(x, y, e)`

Векторы с горизонтальными и вертикальными координатами каждой точки.

Вектор значений ошибок в каждой точке.

- Длины этих трех векторов `x`, `y` и `e` должны быть одинаковы.
- Длина бара погрешности – удвоенное значение `e`. В каждой точке бар погрешности распространяется от $y(i) - e(i)$ до $y(i) + e(i)$.

График на рис. 5.10, который имеет симметричные значения погрешностей, создан выполнением следующего кода:

```
xd=[10:2:22];
yd=[950 640 460 340 250 180 140];
ydErr=[30 20 18 35 20 30 10];
errorbar(xd,yd,ydErr)
xlabel('DISTANCE (cm)')
ylabel('INTENSITY (lux)')
```

Команда для создания графика с несимметричными значениями погрешностей:

`errorbar(x, y, d, u)`

Векторы с координатами каждой точки.

Вектор с верхними границами ошибок.

Вектор с нижними границами ошибок.

- Длины этих четырех векторов x , y , d и u должны быть одинаковы.
- В каждой точке бар погрешности распространяется от $y(i)-d(i)$ до $y(i)+u(i)$.

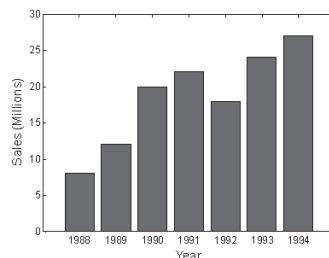
5.7. Графики специального вида

Все графики, которые до сих пор обсуждались в этой главе, являются графиками-линиями, в которых точки данных соединены линиями. Во многих ситуациях эффективнее представить данные на рисунках с другой графикой или геометрией. У MATLAB есть много опций для создания большого разнообразия графиков. Они включают бары (bar), ступеньки, стебли (stem), круговые диаграммы и еще много других. Ниже представлены некоторые из специальных графиков, которые могут быть созданы с MATLAB. Полный список функций графического изображения, которыми располагает MATLAB и информация о том, как их использовать, может быть найдена в окне справки **Help MATLAB**. В этом окне сначала нужно выбрать **Functions by Category**, затем **Graphics** и потом выбрать **Basic Plots and Graphs** или **Specialized Plotting**.

Бары (вертикальные и горизонтальные), лестница и графики-стебли представлены на следующих диаграммах, используя данные о сбыте из раздела 5.1.1.

График вертикальными барами

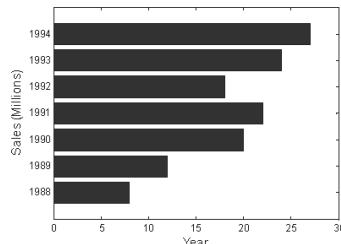
Формат функции:
bar(x,y)



```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
bar(yr,sle,'r') Красные бары
xlabel('Year')
ylabel('Sales (Millions) ')
```

График горизонтальными барами

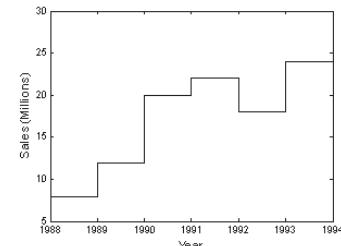
Формат функции:
barh(x,y)



```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
barh(yr,sle)
xlabel('Sales (Millions)')
ylabel('Year')
```

Ступеньки

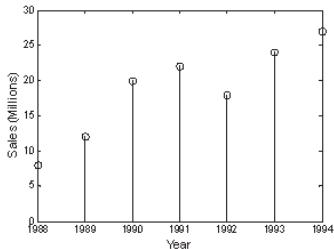
Формат функции:
stairs(x,y)



```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
stairs(yr,sle)
```

Стебли

Формат функции:
stem(x,y)



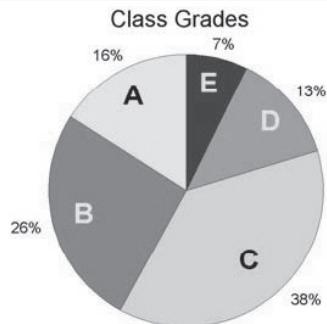
```
yr=[1988:1994];
sle=[8 12 20 22 18 24 27];
stem(yr,sle)
```

Круговые диаграммы (секторы) полезны для визуализации относительных размеров различных но связанных величин. Например, таблица ниже показывает оценки, которые были выставлены классу на контрольной работе. Эти данные используются для создания следующей круговой диаграммы.

Оценки	A	B	C	D	E
Число студентов	11	18	26	9	5

Секторы

Формат функции:
pie(x)



```
grd=[11 18 26 9 5];
pie(grd)
title('Class Grades')
```

MATLAB раскрашивает части в разные цвета. Буквы и % добавлены редактором графиков.

5.8. Гистограммы

Гистограммы – это графики, которые показывают распределение данных. Весь диапазон значений множества точек данных делится на поддиапазоны (корзины) и гистограмма показывает, сколько точек данных находится в каждом корзине. Гистограмма – это вертикальный бар-график, в котором ширина каждого бара равна диапазону соответствующей корзины, а высота бара соответствует числу точек данных в корзине. Гистограммы создаются в MATLAB командой `hist`. Самая простая форма команды следующая:

```
hist(y)
```

`y` – это вектор с точками данных. В этой команде MATLAB делит диапазон точек данных на 10 равных поддиапазонов (корзин) и затем графически изображает числа точек данных в каждой корзине.

Пусть, например, следующие точки данных представляют ежедневную максимальную температуру (в °F) в Вашингтоне, округ Колумбия, в течение месяца апреля 2002: 58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69 (данные от американского Национального управления океанических и атмосферных исследований). Гистограмма этих данных получена командами:

```
>> y=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89  
91 80 59 69 56 64 63 66 64 74 63 69];  
>> hist(y)
```

Созданная диаграмма показана на рис. 5.11 (были добавлены заголовки осей с использование редактора окна графика). Наименьшее значение в наборе данных 48, а самое большое 93, что означает, что длина диапазона 45, а ширина каждой корзины 4.5. Диапазон первой корзины от 48 до 52.5 и содержит две точки. Диапазон второй корзины от 52.5 до 57 и содержит три точки, и так далее. Две корзины (от 75 до 79.5 и от 84 до 88.5) не содержат точек.

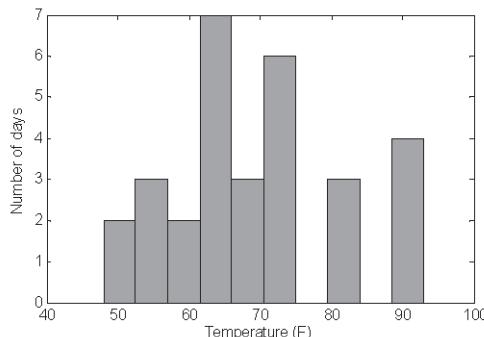


Рис. 5.11. Гистограмма температурных данных

Поскольку деление диапазона данных на 10 равных корзин может оказаться не таким делением, которое желал бы пользователь, то число корзин может быть определено отличным от 10. Это может быть сделано или определением числа корзин, или определением центральной точки каждой корзины, как показано в следующих двух формах команды `hist`:

`hist(y,nbins)` или `hist(y,x)`

`nbins` – скаляр, который определяет число корзин. В этом случае MATLAB делит диапазон на равномерно распределенные поддиапазоны.

`x` – вектор, который определяет положение центра каждой корзины (расстояние между центрами не должно одинаковым для всех корзин). Края корзин – в серединах между центрами.

В примере выше пользователь мог бы предпочесть делить температуру диапазон на три корзины. Это может быть сделано командой:

```
>> hist(y, 3)
```

Как показано на верхнем рисунке, созданная гистограмма имеет три корзины равной величины.

Число и ширина корзин может быть также определена вектором x , элементы которого определяют центры корзин. Например, диаграмма показанная ниже – гистограмма, которая выводит на экран температурные данные в шесть корзин с равной шириной в 10 градусов. Элементы вектора x для этого графика: 45, 55, 65, 75, 85 и 95. График был получен следующими командами:

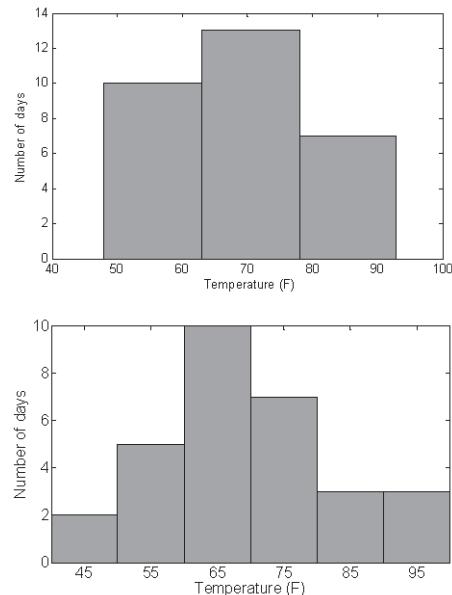
```
>> x=[45:10:95]
x =
45 55 65 75 85 95
>> hist(y,x)
```

Команда `hist` может использоваться с опциями, которые обеспечивают числовой результат в дополнение к графическому изображению гистограммы. Число точек данных в каждой корзине может быть получено одной из следующих команд:

```
n=hist(y)          n=hist(y,nbins)        n=hist(y,x)
```

Результат n является вектором. Число элементов в n равно числу корзин, а значение каждого элемента вектора n есть число точек данных (частота попадания) в соответствующей корзине. Например, гистограмма на рис. 5.11 может также быть создана со следующей командой:

```
>> n = hist(y)
n =
2 3 2 7 3 6 0 3 0 4
```



Вектор n показывает, сколько элементов находится в каждой корзине.

Вектор n показывает, что в первой корзине есть две точки данных, во второй корзине – три точки данных и так далее.



Еще один дополнительный необязательный числовoy результат – это расположение корзин. Данный результат может быть получен одной из следующих команд:

```
[n xout]=hist(y)
```

```
[n xout]=hist(y,nbins)
```

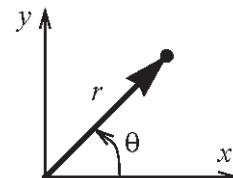
`xout` – это вектор, у которого значение каждого элемента есть расположение центра соответствующей корзины. Например, для гистограммы на рис. 5.11:

```
>> [n xout]=hist(y)
n =
    2   3   2   7   3   6   0   3   0   4
xout =
    50.2500  54.7500  59.2500  63.7500  68.2500  72.7500
77.2500  81.7500  86.2500  90.7500
```

Вектор `xout` показывает, что центр первой корзины в точке 50.25, центр второй корзины – в 54.75 и так далее.

5.9. Графики в полярных координатах

Полярные координаты, в которых положение точки на плоскости определяется углом θ и радиусом (расстоянием) до точки, часто используются в решении научно-технических задач. Команда `polar` используется для графического изображения функций в полярных координатах. Команда имеет вид:



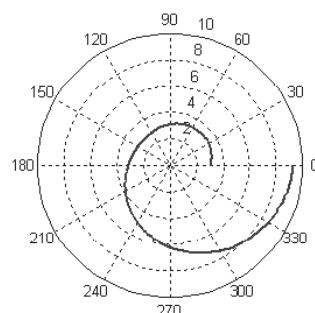
```
polar(theta, radius, 'line specifiers')
```

Вектор

(Дополнительные).
Спецификаторы
определяющие тип и цвет
линий и маркеров.

где `theta` и `radius` – это векторы, элементы которых определяют координаты точек, которые будут графически изображены. Команда `polar` графически изображает точки и рисует полярную сетку. Спецификаторы линии – те же самые, что и в команде `plot`. Чтобы графически изобразить функцию $r = f(\theta)$ в определенной области, сначала создается вектор значений θ , а затем – вектор r с соответствующими значениями $f(\theta)$, используя поэлементные вычисления. Эти два вектора затем используются в команде `polar`.

Например, на рисунке показан график функции $r = 3\cos^2(0,5\theta) + \theta$ для $0 \leq \theta \leq 2\pi$.



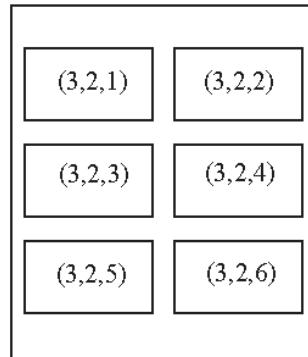
```
t=linspace(0,2*pi,200);
r=3*cos(0.5*t).^2+t;
polar(t,r)
```

5.10. Расположение нескольких окон графиков на одной странице

Несколько графиков могут быть созданы на одной и той же странице Figure при помощи команды `subplot`, вид которой следующий:

```
subplot(m,n,p)
```

Эта команда делит все окно графиков Figure (и страницу при печати) на $m \times n$ прямоугольных подокон графиков. Эти подграфики расположены как элементы в матрице, где каждый элемент – подграфик. Подграфики пронумерованы от 1 до $m \cdot n$. Верхний левый подграфик имеет номер 1, а нижний правый подграфик – номер $m \cdot n$. Номер увеличивается слева направо по строкам от первой строки до последней. Команда `subplot(m,n,p)` создает текущее p -ое подокно. Это означает, что следующая за ней команда `plot` (и любые команды форматирования) создаст график (с соответствующим форматом) в этом подокне. Например, команда `subplot(3,2,1)` создает шесть областей, расположенных в трех строках и двух столбцах как показано на рисунке и делает текущим верхнее левое подокно. Пример использования команды `subplot` показан при решении примера задачи 5.2.



5.11. Несколько окон графиков Figure

Когда выполняется `plot` или любая другая команда создающая график, открывается окно графиков Figure (если оно еще не открыто) и выводит на экран график. MATLAB помечает окно графиков как Figure 1 (см. верхний левый угол окна графиков Figure, которое выведено на экран на рис. 5.4). Если окно графиков Figure уже открыто, когда выполняется `plot` или любая другая команда создающая график, то новый график будет выведен на экран в этом же окне графиков Figure (заменив существующий график). Команды, которые форматируют графики, применяются к этому графику в открытом окне графиков Figure.

Однако можно открыть дополнительные окна графиков Figure и иметь одновременно несколько из них открытыми (с графиками). Это делается введением команды `figure`. Каждый раз, когда вводится команда `figure`, MATLAB открывает

ет новое окно графиков Figure. Если команда, которая создает график, вводится после команды `figure`, MATLAB создает и выводит на экран новый график в последнем окне графиков Figure, которое было открыто и которое называют активным или текущим окном. MATLAB помечает новые окна графиков Figure последовательно: то есть, Figure 2, Figure 3 и так далее. Например, после выполнения следующих трех команд на экран выводятся два окна графиков Figure, которые показаны на рис. 5.12.,

```
>> fplot('x*cos(x)', [0, 10])
>> figure
>> fplot('exp(-0.2*x)*cos(x)', [0, 10])
```

График выводится в окне Figure 1.

Открывается окно Figure 2.

График выводится в окне Figure 2.

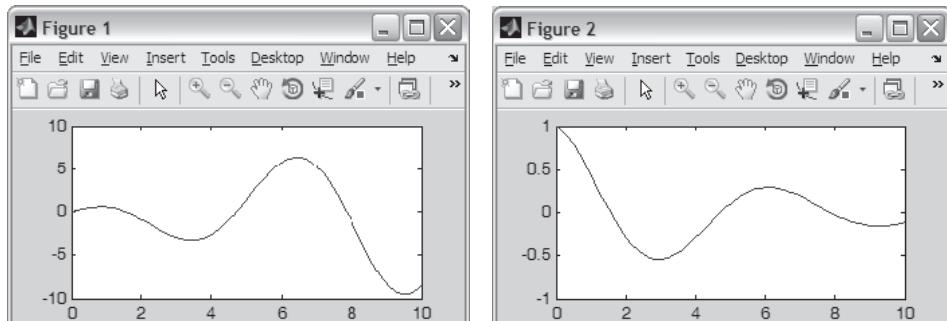


Рис. 5.12. Два открытых окна Figure

У команды `figure` может также быть входной аргумент, который является номером (integer) в виде `figure(n)`. Номер соответствует номеру соответствующего окна графиков Figure. Когда эта команда выполняется, окно с номером `n` становится активным окном графиков Figure (если окно графиков Figure с этим номером не существует, открывается новое окно с этим номером). При выполнении команд, создающих новые графики, эти графики выводятся на экран в активном окне графиков Figure. Таким же образом, команды, которые форматируют графики, применяются к графику в активном окне. Команда `figure(n)` позволяет программно в скрипте открыть и создать графики в нескольких определенных окнах графиков Figure. (Если в программе вместо этого будут использоваться несколько команд `figure`, то новые окна графиков Figure будут открываться каждый раз, когда выполняется файл сценария.)

Окна графиков Figure могут быть закрыты командой `close`. Несколько форм команды:

`close` – закрывает активное окно графиков Figure;

`close(n)` – закрывает окно графиков Figure с номером `n`;

`close all` – закрывает все открытые окна графиков Figure.

5.12. Построение графиков с использованием ленты инструментов PLOTS

Графики могут также быть созданы в командном окне в интерактивном режиме при использовании ленты инструментов PLOTS. Лента инструментов PLOTS, как показано на рис. 5.13, выводится на экран, когда выбирается вкладка PLOTS. Чтобы сделать двумерный график, векторы с точками данных, которые будут использоваться для графика, должны быть уже присвоены и выведены на экран в окне рабочего пространства (см. раздел 4.1). Чтобы создать график, выберите переменную в окне рабочего пространства и затем, удерживая клавишу **Ctrl**, выберите любые дополнительные необходимые переменные. Как только выбор переменных сделан, лента инструментов показывает значки с изображениями типов графика, которые могут быть созданы с выбранными переменными (например, график-линия, диаграмма рассеяния, график в виде столбцов, круговая диаграмма, и т. д.). Щелчок по значку открывает окно графиков Figure с соответствующим графиком. Кроме того, команда MATLAB, которая создала этот график, выводится на экран в командном окне. Пользователь может после этого скопировать команду и вставить ее в файл сценария с тем, чтобы в будущем, создавался бы тот же самый рисунок, когда будет выполняться файл сценария. Справа на ленте инструментов пользователь может выбрать для просмотра различные типы графиков в том же самом окне графиков Figure (повторное использование Figure), или просмотреть новый рисунок в новом окне графиков Figure сравнивая рядом оба типа графиков.

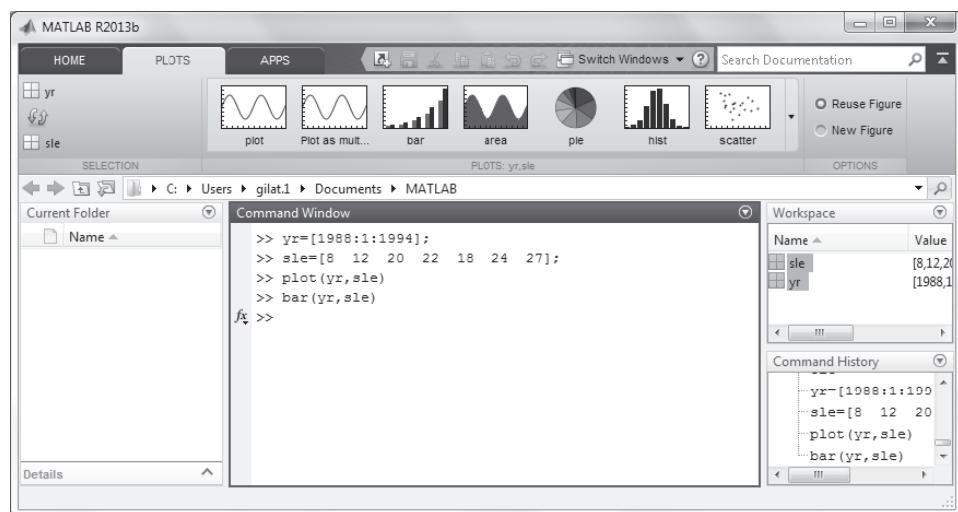


Рис. 5.13. Использование ленты инструментов PLOTS

Использование ленты инструментов графиков полезно, когда пользователь хочет посмотреть различные опции построения графика на определенных данных.

Например, рис. 5.13 показывает выведенную на экран ленту инструментов PLOTS с расположением элементов по умолчанию MATLAB. В командном окне данные о сбите из раздела 5.1.1 присвоены двум векторам `yr` и `sle`. Векторы также выведены на экран (и выбраны) в окне рабочего пространства. Значки с изображениями различных типов графиков, которые могут быть созданы, выведены на экран на ленте инструментов PLOTS наверху. Дополнительные типы графиков могут быть выведены на экран, щелкнув по стрелке вниз справа.

В качестве примера, созданы два различных графика, один с графиком в виде линии, а другой с графиком в виде баров, используя эти два вектора `yr` и `sle`. Оба рисунка показаны на рис. 5.14, а команды, которые создали эти графики в командном окне, показаны на рис. 5.13.

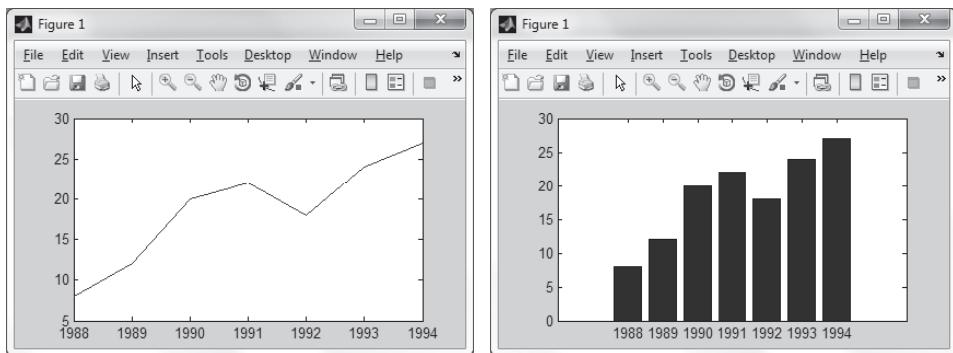
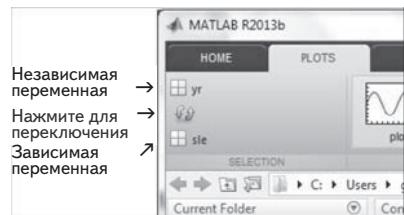


Рис. 5.14. Использование ленты инструментов PLOTS

Дополнительные замечания

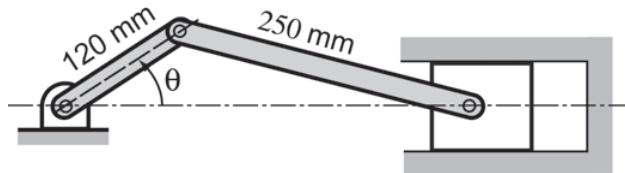
- При выборе переменных для графика (в окне рабочего пространства) первой выбирается независимая переменная (горизонтальная ось), а второй будет зависимая переменная (вертикальная ось). После выбора переменные могут быть переключены, щелкнув по значку **Switch**.
- Если будет выбрана только одна переменная (вектор), то будут графически изображены значения элементов вектора как функции от номера элемента.



5.13. Примеры приложений MATLAB

Пример задачи 5.2. Кривошипно-шатунный механизм

Кривошипно-шатунный механизм используется во многих технических приложениях. В механизме, показанном в следующем рисунке, кривошип вращается с постоянной скоростью 500 об/мин.



Вычислите и графически изобразите положение, скорость и ускорение поршня для одного оборота колена. Сделайте три окна графиков на одной странице. Считайте, что $\theta = 0^\circ$ при $t = 0$.

Решение

Кривошип вращается с постоянной угловой скоростью $\dot{\theta}$. Это означает, что если мы устанавливаем $\theta = 0$ при $t = 0$, тогда в момент времени t угол поворота θ задается формулой $\theta = \dot{\theta}t$ и $\ddot{\theta} = 0$ всегда.

Расстояния d_1 и h задаются формулами:

$$d_1 = r \cos \theta \quad \text{и} \quad h = r \sin \theta.$$

Когда h найдено, расстояние d_2 может быть вычислено по теореме Пифагора:

$$d_2 = (c^2 - h^2)^{1/2} = (c^2 - r^2 \sin^2 \theta)^{1/2}.$$

Тогда положение x поршня находится по формуле

$$x = d_1 + d_2 = r \cos \theta + (c^2 - r^2 \sin^2 \theta)^{1/2}.$$

Производная x по времени дает скорость поршня:

$$\dot{x} = -r \dot{\theta} \sin \theta - \frac{r^2 \dot{\theta} \sin 2\theta}{2(c^2 - r^2 \sin^2 \theta)^{3/2}}.$$

Вторая производная x по времени дает ускорение поршня:

$$\ddot{x} = -r \dot{\theta}^2 \cos \theta - \frac{4r^2 \dot{\theta} \cos 2\theta (c^2 - r^2 \sin^2 \theta) + (r^2 \dot{\theta} \sin 2\theta)^2}{2(c^2 - r^2 \sin^2 \theta)^{3/2}}.$$

В уравнении выше, учитывалось, что $\ddot{\theta} = 0$.

Ниже приведена программа MATLAB (файл сценария), которая вычисляет и графически изображает положение, скорость и ускорение поршня для одного оборота колена.

```
THDrpm=500; r=0.12; c=0.25;
```

Определение $\dot{\theta}$, r и c .

```
THD=THDrpm*2*pi/60;
```

Измените единицы измерения $\dot{\theta}$ от об/мин. в rad/s.

```
tf=2*pi/THD;
```

Вычисление времени одного поворота плеча.

```
t=linspace(0,tf,200);
```

Создание вектора времени с 200 элементами.

```
TH=THD*t;
```

Вычисление θ для каждого t .

→

```

d2s=c^2-r^2*sin(TH).^2;
x=r*cos(TH)+sqrt(d2s);          Вычисление квадрата  $d_2$  для каждого  $\theta$ .
                                                                               Вычисление  $x$  для каждого  $\theta$ .
Вычисление  $\dot{x}$  и  $\ddot{x}$  для каждого  $\theta$ :
xd=-r*THD*sin(TH)-(r^2*THD*sin(2*TH))./(2*sqrt(d2s));
xdd=-r*THD^2*cos(TH)-(4*r^2*THD^2*cos(2*TH).*d2s+           График  $x$  как функции от  $t$ .
(r^2*sin(2*TH)*THD).^2)./(4*d2s.^ (3/2));                  Форматирование окна первого графика.
subplot(3,1,1)
plot(t,x)
grid
xlabel('Time (s)')
ylabel('Position (m)')
subplot(3,1,2)
plot(t,xd)
grid
xlabel('Time (s)')
ylabel('Velocity (m/s)')
subplot(3,1,3)
plot(t,xdd)
grid
xlabel('Time (s)')
ylabel('Acceleration (m/s^2)')                                     График  $\dot{x}$  как функции от  $t$ .
                                                               Форматирование окна второго графика.
                                                               График  $\ddot{x}$  как функции от  $t$ .
                                                               Форматирование окна третьего графика.

```

При выполнении этого скрипта создаются три графика на одной самой странице Figure как показано на рис. 5.13. Данные ясно показывают, что скорость поршня равна нулю в конечных точках диапазона перемещения, где поршень изменяет направление движения. Ускорение максимально (и направлено налево), когда поршень находится в правом конце.

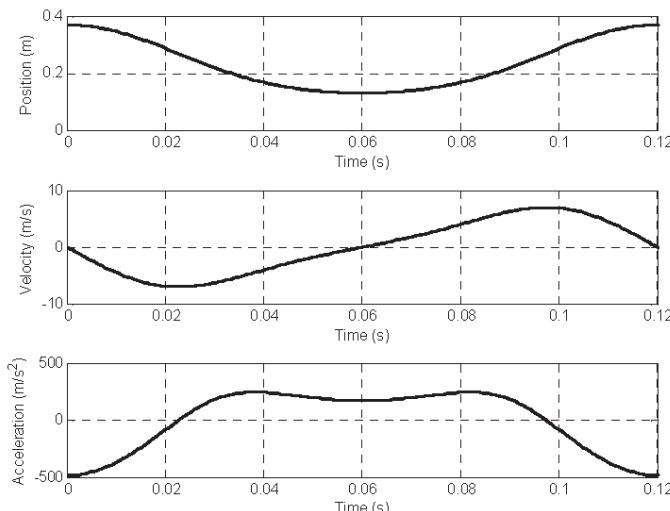


Рис. 5.15. Положение, скорость и ускорение поршня в зависимости от времени

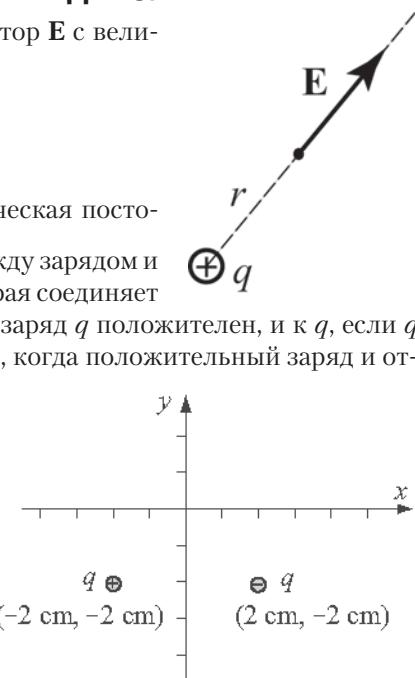
Пример задачи 5.3. Электрический диполь

Электрическое поле заряда в точке есть вектор \mathbf{E} с величиной E , заданной законом Кулона:

$$E = \frac{1}{4\pi\epsilon_0} \frac{q}{r^2}.$$

где $\epsilon_0 = 8.8541878 \times 10^{-12} \frac{C^2}{N \cdot m^2}$ – диэлектрическая постоянная, q – величина заряда и r – расстояние между зарядом и точкой. Вектор \mathbf{E} направлен вдоль линии, которая соединяет заряд с точкой. Вектор \mathbf{E} направлен от q , если заряд q положителен, и к q , если q отрицателен. Электрический диполь создается, когда положительный заряд и отрицательный заряд равной величины помещены отдельно на некотором расстоянии. Электрическое поле \mathbf{E} в каждой точке получается наложением электрического поля каждого заряда.

Электрический диполь с $q = 12 \times 10^{-9}$ Кл создан, как показано на рисунке. Определите и графически изобразите величины электрического поля вдоль оси x от $x = -5$ см до $x = 5$ см.



Решение

Электрическое поле \mathbf{E} в каждой точке $(x, 0)$ вдоль оси x получается сложением векторов электрического поля от каждого из зарядов:

$$\mathbf{E} = \mathbf{E}_- + \mathbf{E}_+.$$

Величина электрического поля – это длина вектора \mathbf{E} .

Для решения задачи нужно выполнить следующие действия:

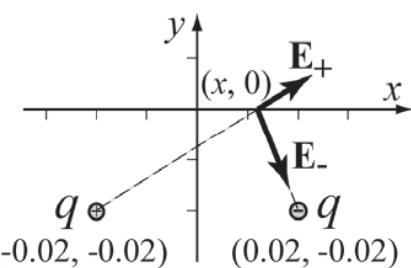
Шаг 1. Создание вектора x для точек вдоль оси x .

Шаг 2. Вычисление расстояние (и расстояния в квадрате) от каждого заряда до точек на оси x :

$$r_{\minus} = \sqrt{(0.02 - x)^2 + 0.02^2}, \quad r_{\plus} = \sqrt{(x + 0.02)^2 + 0.02^2}.$$

Шаг 3. Запись единичных векторов в направлении от каждого заряда до точек на оси x :

$$\mathbf{E}_{\minus\ UV} = \frac{1}{r_{\minus}} ((0.02 - x)\mathbf{i} - 0.02\mathbf{j}), \quad \mathbf{E}_{\plus\ UV} = \frac{1}{r_{\plus}} ((x + 0.02)\mathbf{i} + 0.02\mathbf{j}).$$



Шаг 4. Вычисление величин векторов \mathbf{E}_- и \mathbf{E}_+ в каждой точке используя закон Кулона:

$$E_{\text{minusMAG}} = \frac{1}{4\pi\epsilon_0} \frac{q}{r_{\text{minus}}^2}, \quad E_{\text{plusMAG}} = \frac{1}{4\pi\epsilon_0} \frac{q}{r_{\text{plus}}^2}.$$

Шаг 5. Создание векторов \mathbf{E}_- и \mathbf{E}_+ умножением единичных векторов на величины.

Шаг 6. Создание вектора \mathbf{E} сложением векторов \mathbf{E}_- и \mathbf{E}_+ .

Шаг 7. Вычисление E , величины (длины) вектора \mathbf{E} .

Шаг 8. График E как функции от x .

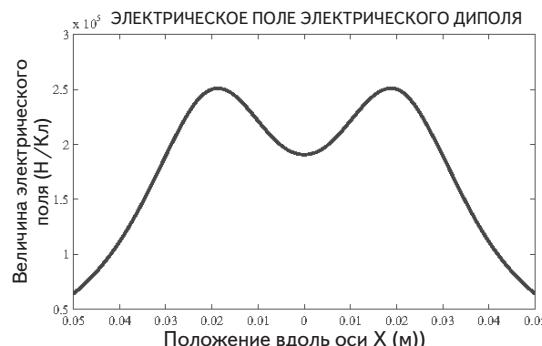
Программа в файле сценария, которая решает эту задачу:

```

q=12e-9;
epsilon0=8.8541878e-12;
x=[-0.05:0.001:0.05]';           Шаг 1. Создание вектора столбца x.
rminusS=(0.02-x).^2+0.02^2;
rminus=sqrt(rminusS);           Шаг 2. Каждая переменная есть вектор столбец.
rplusS=(x+0.02).^2+0.02^2;      Шаги 3 и 4. Каждая переменная –
rplus=sqrt(rplusS);           это матрица из двух
EminusUV=[((0.02-x)./rminus), (-0.02./rminus)];   столбцов.
EplusUV=[((x+0.02)./rplus), (0.02./rplus)];       Каждая строка – это вектор
EminusMAG=(q/(4*pi*epsilon0))./rminusS;           для соответствующего x.
EplusMAG=(q/(4*pi*epsilon0))./rplusS;           Шаг 5.
Eminus=[EminusMAG.*EminusUV(:,1), EminusMAG.*EminusUV(:,2)];   Шаг 6.
Eplus=[EplusMAG.*EplusUV(:,1), EplusMAG.*EplusUV(:,2)];       Шаг 7.
E=Eminus+Eplus;
EMAG=sqrt(E(:,1).^2+E(:,2).^2);
plot(x,EMAG,'k','LineWidth',1)
xlabel('Положение вдоль оси X (м)', 'FontSize',12)
ylabel('Величина электрического поля (Н/Кл)', 'FontSize',12)
title('ЭЛЕКТРИЧЕСКОЕ ПОЛЕ ЭЛЕКТРИЧЕСКОГО ДИПОЛЯ', 'FontSize',12)

```

После выполнения этого файла сценария в командном окне, создается следующий рисунок в окне графиков Figure:



5.14. Задачи

1. Построить график функции $f(x) = \frac{x^2 - 3x + 7}{\sqrt{2x + 5}}$ для $-1 \leq x \leq 5$.
2. Построить график функции $f(x) = (3 \cos x - \sin x)e^{-0.2x}$ для $-4 \leq x \leq 9$.
3. Построить график функции $f(x) = \frac{x^2}{2 + \sin x + x^4}$ для $-4 \leq x \leq 4$.
4. Построить график функции $f(x) = x^3 - 2x^2 - 10 \sin^2 x - e^{0.9x}$ и ее производной для $-2 \leq x \leq 4$ в одном окне. График функции изобразить сплошной линией, а производной – пунктирной линией. Добавить легенду и метки осей.
5. Создать два отдельных графика функции $f(x) = -3x^4 + 10x^2 - 3$, один график для $-4 \leq x \leq 3$, а второй для $-4 \leq x \leq 4$
6. Используя команду `fpplot` создать график функции $f(x) = (\sin 2x + \cos^2 5x)e^{-0.2x}$ для $-6 \leq x \leq 6$.
7. Построить график функции $f(x) = \sin^2(x) \cos(2x)$ и ее производной для $0 \leq x \leq 2$ в одном окне. График функции изобразить сплошной линией, а производной – пунктирной линией. Добавить легенду и метки осей.
8. Создать график окружности с центром в $(4.2, 2.7)$ и радиусом 7.5 .
9. Функция задана параметрически уравнениями:

$$x = \sin(t)\cos(t), \quad y = 1.5\cos(t).$$

Изобразить график функции для $-\pi \leq t \leq \pi$. Отформатировать график так, что, обе оси имели диапазоны от -2 до 2 .
10. Две функции заданы параметрически уравнениями:

$$x = \cos^3(t), \quad y = \sin^3(t),$$

$$u = \sin(t), \quad v = \cos(t).$$

В одном окне изобразить графики функций y от x и v от u для $0 \leq t \leq 2$. Отформатировать графики так, что, обе оси имели диапазоны от -2 до 2 .
11. Построить график функции $f(x) = \frac{x^2 - 5x - 12}{x^2 - x - 6}$ в области $-1 \leq x \leq 7$. Отметим, что у функции есть вертикальная асимптота в точке $x = 3$. Построить график функции, создавая два вектора для области изменения x . Первый вектор (назовем его $x1$) включает элементы от -1 до 2.9 , а второй вектор (назовем его $x2$), включает элементы от 3.1 до 7 . Для каждого x -вектора создать вектор значений

y (назовем их y_1 и y_2) с соответствующими значениями y согласно функции. Чтобы графически изобразить функцию создайте две кривые в одном окне (график y_1 от x_1 и график y_2 от x_2). Отформатировать график так, что диапазоны оси y был от -20 до 20 .

12. Построить график функции $f(x) = \frac{x^2 + 3x - 5}{x^2 - 3x - 10}$ в области $-4 \leq x \leq 9$. Отметим, что у функции есть две вертикальных асимптоты в точке $x = 3$. Построить график функции, разбивая область изменения переменой x на три части: первая от -4 до левой асимптоты, вторая часть между этими двумя асимптотами и третья – от правой асимптоты до 9 . Установить диапазон оси y от -20 до 20 .

13. Функция задана параметрически уравнениями:

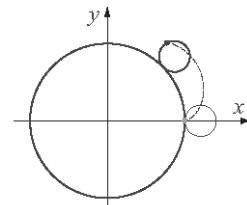
$$x = \frac{3t}{1+t^3}, \quad y = \frac{3t^2}{1+t^3}.$$

(Отметим, что знаменатель стремится к 0 , когда t стремится к -1 .) Изобразить график функции (он вызывается декартовым листом), строя графики двух кривых в одном окне. Первая кривая для значений параметра $-30 \leq t \leq -1.6$, а другая для $-0.6 \leq t \leq 40$.

14. Эпициклоида – это кривая (показанная частично на рисунке) полученная, как траектория точки на окружности, которая катится по другой фиксированной окружности. Параметрическим уравнением циклоиды задаются формулами:

$$\begin{aligned} x &= 13\cos(t) - 2\cos(6.5t), \\ y &= 13\sin(t) - 2\sin(6.5t). \end{aligned}$$

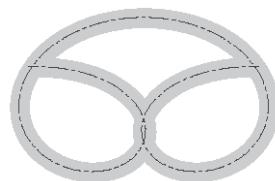
Построить график циклоиды для $0 \leq t \leq 4\pi$.



15. Форма показанного на рисунке кренделя с солью задается следующими параметрическими уравнениями:

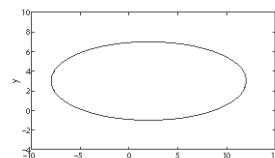
$$x = (3.3 - 0.4 t^2) \sin(t), \quad y = (2.5 - 0.3 t^2) \cos(t),$$

где $-4 \leq t \leq 3$. Построить график кренделя.



16. Создать график функции, заданной в полярных координатах $r = 2\sin(3\theta)\sin(\theta)$ для $0 \leq \theta \leq 2\pi$.

17. Графически изобразить эллипс с главными осями $a = 10$ и $b = 4$ и центром в точке $x = 2$ и $y = 3$.



18. Следующие данные дают приблизительное количество населения Земли в течение выбранных лет с 1850 до 2000.

Год	1850	1910	1950	1980	2000	2010
Население (миллиардов)	1.3	1.75	3	4.4	6	6.8

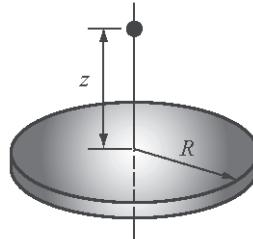
Население P с 1900 может быть смоделировано логистической функцией:

$$P = \frac{11.55}{1 + 18.7 e^{-0.0193t}},$$

где P есть население в миллиардах и t – годы с 1850. Создайте график народонаселения как функцию от лет. Рисунок должен показать информацию из приведенной выше таблицы в виде точек и население, моделируемое уравнением в виде сплошной линии. Установите диапазон горизонтальной оси от 1800 до 2200. Добавьте легенду и метки осей.

19. Сила F (в Н) действующая между частицей с зарядом q и круглым диском радиуса R и заряда Q задается формулой:

$$F = \frac{Qqz}{2\epsilon_0} \left(1 - \frac{z}{\sqrt{z^2 + R^2}}\right),$$

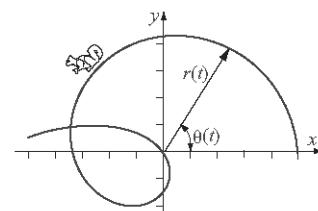


где $\epsilon_0 = 0.885 \times 10^{-12}$ Кл²/Н·м² – диэлектрическая постоянная и z – расстояние до частицы. Рассмотрите случай, когда $Q = 9.4 \times 10^{-6}$ Кл, $q = 2.4 \times 10^{-5}$ Кл и $R = 0.1$ м. Создать график F как функцию от z для $0 \leq z \leq 0.3$ м. Использовать встроенную функцию MATLAB `max` для нахождения максимального значения F и соответствующего расстояния z .

20. Зависимость от времени положения белки, бегущей на поле, задается в полярных координатах следующими соотношениями:

$$\begin{aligned} r(t) &= 25 + 30[1 - e^{\sin(0.07t)}] \text{ м,} \\ \theta(t) &= 2\pi(1 - e^{-0.2t}). \end{aligned}$$

Графически изобразить траекторию (положение) белки для $0 \leq t \leq 20$ сек.

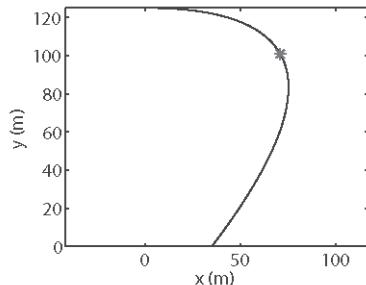


21. Рассмотрим движение белки в предыдущей задаче. Компоненты вектора скорости белки задаются формулами $v_r = \frac{dr}{dt}$ и $v_\theta = r \frac{d\theta}{dt}$. Скорость белки есть $v = \sqrt{v_r^2 + v_\theta^2}$. Построить график скорости белки как функции времени для $0 \leq t \leq 20$ сек.

22. Криволинейное движение частицы определено следующими параметрическими уравнениями:

$$x = 52t - 9t^2 \text{ м} \quad \text{и} \quad y = 125 - 5t^2 \text{ м.}$$

Скорость частицы есть $v = \sqrt{v_x^2 + v_y^2}$, где $v_x = \frac{dx}{dt}$ и $v_y = \frac{dy}{dt}$. Для $0 \leq t \leq 5$ сек построить один график, который показывает положение частицы (y от x) и второй график (на той же самой странице) скорости частицы как функции времени. Кроме того, с использованием функции `min` MATLAB, определить время, когда скорость является наименьшей и соответствующее положение частицы. Используя маркер звездочки, показать это положение частицы на первом графике. Для времени использовать вектор с шагом 0.1 сек.



23. Спрос на воду во время пожара часто является наиболее важным фактором при проектировании распределения резервуаров для хранения и насосов. Для поселений численностью меньше чем 200 000 требование Q (в галлонах/минуту) может быть вычислено по формуле:

$$Q = 1020\sqrt{P}(1 - 0.01\sqrt{P}),$$

где P – это население в тысячах. Графически изобразить необходимое количество Q воды как функцию от количества населения P (в тысячах) для $0 \leq P \leq 200$. Обозначить оси и создайте заголовок для графика.

24. Положение x частицы как функции времени, которая проходит по прямой линии, задается формулой:

$$x(t) = (-3 + 4t)e^{-0.4t} \text{ футов.}$$

Скорость $v(t)$ частицы определена производной $x(t)$ относительно t , а ускорение $a(t)$ определяется как производная от скорости $v(t)$ по t . Получить выражения для скорости и ускорения частицы и создать графики положения частицы, скорости и ускорения как функций времени для $0 \leq t \leq 20$ сек. Использовать команду `subplot` для создания трех графиков на одной странице Figure с графиком положения вверху, скорости – в середине и ускорении – в нижней части. Обозначить оси соответственно корректными единицами.

25. Площадь аортального клапана, A_V в см^2 , может быть оценена уравнением (Формула Хакки):

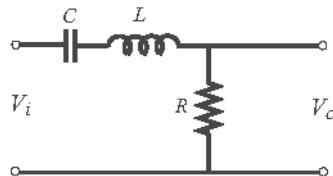
$$A_V = \frac{Q}{\sqrt{PG}},$$

где Q – минутный сердечный выброс (в $\text{Л}/\text{мин}$) и PG – разность между системическим давлением левого желудочка и аортальным систолическим дав-

лением (в миллиметрах ртутного столба). Сделайте один график с двумя кривыми величины A_v как функции от PG , для $2 \leq PG \leq 60$ (мм рт. ст.) – одна кривая для $Q = 4$ Л/мин и другая – для $Q = 5$ Л/мин. Используйте легенду и обозначения осей.

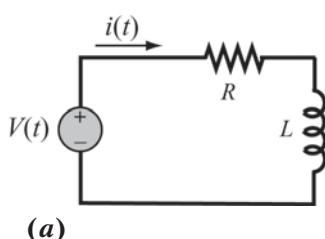
- 26.** Полоснопропускающий фильтр пропускает сигналы с частотами, которые лежат в пределах определенного диапазона. В фильтре указанном на рисунке отношение величин напряжений задается формулой

$$RV = \left| \frac{V_o}{V_i} \right| = \left| \frac{\omega RC}{\sqrt{(1 - \omega^2 LC)^2 + (\omega RC)^2}} \right|,$$

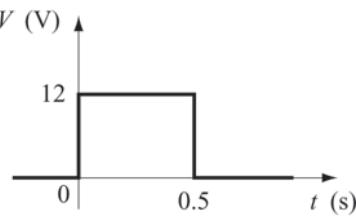


где ω – частота входного сигнала. Для данных $R = 200$ Ом, $L = 8$ мГн и $C = 5$ мкФ создать два графика RV как функция от ω для $10 \leq \omega \leq 500\ 000$. В первом графике использовать линейную шкалу для осей, а во втором графике использовать логарифмическую шкалу для горизонтальной оси (ω) и линейную шкалу для вертикальной оси. Какой график лучше иллюстрирует этот фильтр?

- 27.** Резистор $R = 4$ Ом и индуктор $L = 1.3$ Гн соединены в цепи с источником напряжения как показано на рисунке (a) (цепь RL).



(a)



(b)

Когда источник напряжения подает прямоугольный импульс напряжения с амплитудой $V = 12$ В и продолжительностью 0.5 сек, как показано на рисунке (b), ток $i(t)$ в цепи, как функция времени задается формулами:

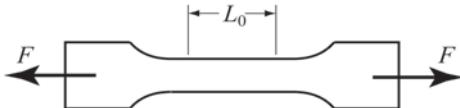
$$i(t) = \frac{V}{R} (1 - e^{(-Rt)/L}), \text{ для } 0 \leq t \leq 0.5 \text{ сек}$$

$$i(t) = e^{(-Rt)/L} \frac{V}{R} (e^{(0.5R)/L} - 1), \text{ для } 0.5 \leq t \leq 2 \text{ сек}$$

Создать график тока как функции времени для $0 \leq t \leq 2$ сек.

- 28.** В типичном испытании на растяжение экземпляр формы собачьей кости вытягивают в машине. Во время испытания сила F вытягивает экземпляр, а дли-

на L некоторой части – измеряется. Эти данные используются для того, чтобы графически изобразить диаграмму растяжений материала. Для напряжения и деформации имеются два определения: техническое и истинное.



Технические напряжение σ_e и деформация ε_e определены формулами: $\sigma_e = \frac{F}{A_0}$ и $\varepsilon_e = \frac{L - L_0}{L_0}$, где L_0 и A_0 – это начальная длина и начальная площадь поперечного сечения экземпляра, соответственно. Истинные напряжение σ_t и деформация ε_t определены формулами: $\sigma_t = \frac{F}{A_0} \frac{L}{L_0}$ и $\varepsilon_t = \ln \frac{L}{L_0}$.

Ниже представлены результаты измерений силы и длину при испытании на растяжение алюминиевого экземпляра. Этот экземпляр имеет круглое сечение радиуса 6.4 mm (перед тестом). Начальная длина $L_0 = 25.4$ mm. Используйте эти данные для вычисления и создания технических и истинных кривых напряжения-деформации, обоих графиков в одном окне. Обозначьте оси и используйте легенду для идентификации кривых.

Единицы измерения: когда сила измеряется в Ньютонах (Н), а площадь вычислена в м², тогда единицы напряжения – Паскали (Па).

F (Па)	0	13,031	21,485	31,963	34,727	37,119	37,960	39,550
L (мм)	25.4	25.474	25.515	25.575	25.615	25.693	25.752	25.978
F (Н)	40,758	40,986	41,076	41,255	41,481	41,564		
L (мм)	26.419	26.502	26.600	26.728	27.130	27.441		

29. Согласно специальной теории относительности, стержень длины L перемещающийся со скоростью v становится короче на величину:

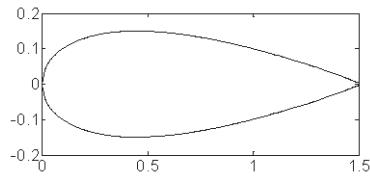
$$\delta = L \left(1 - \sqrt{1 - \frac{v^2}{c^2}} \right),$$

где c – скорость света (примерно 300×10^6 м/с). Для стержня длины 2 м создайте три графика δ как функция v для $0 \leq v \leq 300 \times 10^6$ м/с. В первом графике используйте линейную шкалу для обеих осей. Во втором – логарифмическую шкалу для v и линейную шкалу для δ , а в третьем графике используйте логарифмическую шкалу и для v и для δ . Какой из графиков является самым информативным?

30. Форма симметричного четырехразрядного крыла NACA описывается уравнением

$$y = \pm \frac{tc}{0.2} \left[0.2969 \sqrt{\frac{x}{c}} - 0.1260 \frac{x}{c} - 0.3516 \left(\frac{x}{c} \right)^2 + 0.2843 \left(\frac{x}{c} \right)^3 - 0.1015 \left(\frac{x}{c} \right)^4 \right],$$

где c - длина корда и t – максимальная толщина как функция длины корда (максимальная толщина). Симметричные четырехразрядные крылья NACA обозначаются NACA 00XX, где XX есть $100t$ (то есть, NACA 0012 имеет $t = 0.12$). Графически изобразите форму крыльев NACA 0020 с длиной корда 1.5 м.



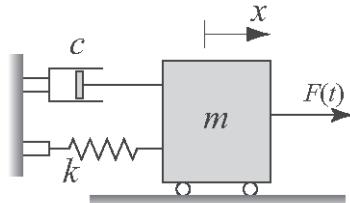
- 31.** Закон идеального газа связывает давление P , объем V и температуру T идеального газа:

$$PV = nRT,$$

где n – число молей и $R = 8.3245$ Дж/(К·моль). Графики давления от объема при постоянной температуре называют изотермами. Изобразить графически изотермы одного моля идеального газа для объема в пределах от 1 до 10 м^3 , при температурах $T = 100, 200, 300$ и 400 К (четыре кривые в одном окне графиков). Обозначьте оси и выведите на экран легенду. Единицы измерения давления – Паскали (Па).

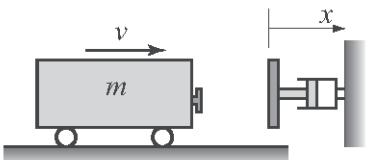
- 32.** Колебания корпуса вертолета из-за периодической силы, создаваемой вращением ротора, можно моделировать лишенной трения системой пружина-масса-демпфер, когда она подвергается действию внешней периодической силы. Положение массы задается уравнением:

$$x(t) = \frac{2f_0}{\omega_n^2 - \omega^2} \sin\left(\frac{\omega_n - \omega}{2} t\right) \sin\left(\frac{\omega_n + \omega}{2} t\right),$$



где $F(t) = F_0 \sin(\omega t)$ и $f_0 = F_0/m$, ω – частота приложенной силы, и ω_n является естественной частотой вертолета. Когда значение ω близко к значению ω_n , вибрация состоит из быстрых колебаний с медленно изменяющейся амплитудой, называемых биением. Использовать $F_0/m = 12$ Н/кг, $\omega_n = 10$ рад/с и $\omega = 12$ рад/с для построения графика $x(t)$ как функции от t для $0 \leq t \leq 10$ сек.

- 33.** Железнодорожный бампер проектируется для замедления быстро движущегося железнодорожного вагона. После того, как 20 000-килограммовый железнодорожный вагон, двигающийся со скоростью 20 м/с,



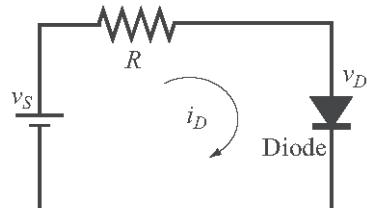
касается бампера, его смещение x (в метрах) и скорость v (в м/с) как функции времени t (в секундах) задаются:

$$x(t) = 4.219(e^{-1.58t} - e^{-6.32t}) \text{ и } v(t) = 26.67e^{-6.32t} - 6.67e^{-1.58t}.$$

Графически изобразите смещение и скорость как функции времени для $0 \leq t \leq 4$ сек. Сделайте два графика на одной странице.

- 34.** Рассмотрим диодную цепь показанную на рисунке. Ток i_D и напряжение v_D могут быть определены из решения следующей системы уравнений:

$$i_D = I_0 \left(e^{\frac{qv_D}{kT}} - 1 \right), \quad i_D = \frac{v_S - v_D}{R}.$$

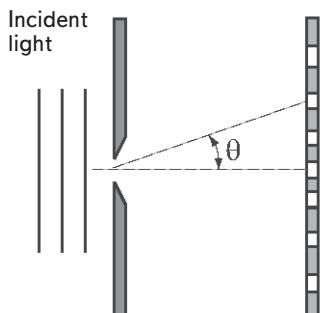


Эта система может быть решена численно или графически. Графическое решение находится построением графиков i_D как функций от v_D из обоих уравнений. Тогда решение – пересечение двух кривых. Создайте графики и оцените решение для случая когда $I_0 = 10^{-14}$ А, $v_S = 1.5$ В, $R = 1200$ Ом и $kT/q = 30$ мВ.

- 35.** Когда монохроматический свет проходит через узкий разрез, он производит на экране дифракционный образ, состоящий из ярких и темных краев. Интенсивность ярких краев I , как функция θ может быть вычислена по формуле,

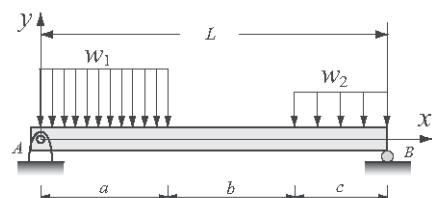
$$I = I_{\max} \left(\frac{\sin \alpha}{\alpha} \right), \text{ где } \alpha = \frac{\pi a}{\lambda} \sin \theta,$$

где λ – длина световой волны и a ширина разреза. Графически изобразите относительную интенсивность I/I_{\max} как функцию θ для $-20^\circ \leq \theta \leq 20^\circ$. Сделайте один рисунок, который содержит три графика для случаев $a = 10\lambda$, $a = 5\lambda$ и $a = \lambda$. Обозначьте оси и выведите на экран легенду.



- 36.** Просто поддерживаемая балка подвергнута распределенным нагрузкам w_1 и w_2 как показано на рисунке. Изгибающий момент как функция x задан следующими уравнениями:

$$M(x) = R_A x = \frac{w_1 x^2}{2}, \text{ для } 0 \leq x \leq a,$$



$$M(x) = R_A x = \frac{w_1 x^2}{2} (2x - a), \text{ для } 0 \leq x \leq (a + b),$$

$$M(x) = R_B(L-x) - \frac{w_2(L-x)^2}{2}, \text{ для } (a+b) \leq x \leq L,$$

где $R_A = [w_1a(2L-a) + w_2c^2]/(2L)$ и $R_B = [w_2c(2L-a) + w_1a^2]/(2L)$ реакции при поддерживании. Сделайте график изгибающего момента M как функция x (один график, который показывает момент для $0 \leq x \leq L$). Возьмите $L = 16$ футов, $a = b = 6$ футов, $w_1 = 400$ фунт/фут и $w_2 = 200$ фунт/фут.

- 37.** Биологическое потребность в кислороде (BOD) является мерой эффекта относительного кислородного источения ненужного загрязнителя и широко используется для оценки количества загрязнения в водном источнике. BOD в сточных водах (L_c в мг/Л) горного фильтра без рециркуляции задается формулой:

$$L_c = \frac{L_0}{1 + \frac{2.5D^{2/3}}{\sqrt{Q}}},$$

где L_0 – впадающий BOD (мг/Л), D – глубина фильтра (м) и Q – расход потока жидкости (Л/(м²·день)). Принимая $Q = 300$ Л/(м²·день) графически изобразить BOD в сточных водах как функцию глубины фильтра ($100 \leq D \leq 2000$ м) для $L_0 = 5, 10$ и 20 мг/Л. Сделайте три графика в одном рисунке и оцените глубину фильтра, требуемого для каждого из этих случаев для получения питьевой воды. Обозначьте оси и выведите на экран легенду.

- 38.** Температурная зависимость коэффициента диффузии D (см²/с) дана уравнением типа Архениуса:

$$D = D_0 e^{\left(-\frac{E_a}{RT}\right)},$$

где D_0 (см²/с) – предэкспоненциальная постоянная, E_a (Дж/моль) – энергия активации для диффузии, $R = 8.31$ (Дж/моль·К) – газовая постоянная и T – температура в Кельвинах. Для диффузии углерода в нержавеющую сталь $D_0 = 6.18$ см²/с и $E_a = 187$ К·Дж/кмоль. Создать два графика D от T для $200 \leq T \leq 800$ К. В первом графике использовать линейную шкалу для обеих осей, а во втором графике использовать линейную шкалу для T и логарифмическую шкалу для D . Оценить, какой график более полезен?

- 39.** Резонансная частота f (в Гц) для показанной на рисунке цепи задается формулой:

$$f = \frac{1}{2\pi} \sqrt{LC \frac{R_1^2 C - L}{R_2^2 C - L}}.$$

Для заданных значений $L = 0.2 \text{ Гн}$, $C = 2 \times 10^{-6} \Phi$ создать следующие графики:

(a) функции f от R_2 для $500 \leq R_2 \leq 2000$ при заданном $R_1 = 1500 \text{ Ом}$.

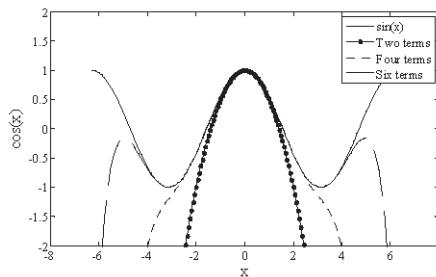
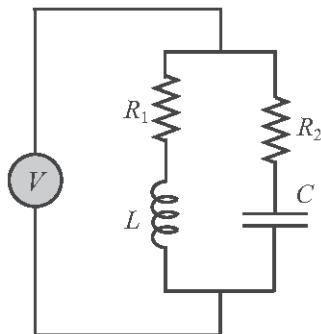
(b) функции f от R_1 для $500 \leq R_1 \leq 2000$ при заданном $R_2 = 1500 \text{ Ом}$.

Графически изобразите оба графика на одном листе Figure (расположение графиков – в столбец).

40. Ряд Тейлора функции $\cos(x)$ есть:

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} + \dots$$

Создайте рисунок справа, который показывает для $-2\pi \leq x \leq 2\pi$ график функции $\cos(x)$ и графики частичных сумм разложения Тейлора $\cos(x)$ с двумя, четырьмя и шестью членами. Сделайте метки осей и выведите на экран легенду.





ГЛАВА 6.

Программирование в MATLAB

Компьютерная программа – это последовательность компьютерных команд. В простой программе команды выполняются одна за другой в том порядке, как они введены. Например, в этой книге все программы, которые до сих пор представлялись в скрипт-файлах, были простыми программами. Однако во многих ситуациях требуются более сложные программы, в которых команды не обязательно выполняются в порядке введения или, когда программа работает с различными входными переменными, то выполняются соответствующие данным различные команды (или группы команд). Например, компьютерная программа, которая вычисляет стоимость отправки пакета по почте, использует различные математические выражения, чтобы вычислить стоимость в зависимости от веса и размера пакета, содержание (книги менее дорогие по почте) и тип службы (авиапочта, основа, и т.д.). В других ситуациях возможно потребуется повторить последовательность команд несколько раз в пределах программы. Например, программы, которые численно решают уравнения, повторяют последовательность вычислений, пока ошибка в ответе не становится меньше чем некоторая мера.

MATLAB обеспечивает ряд инструментов, которые могут использоваться при управлении потоком программы. Условные утверждения (раздел 6.2) и переключатели (раздел 6.3) позволяют пропустить команды или выполнить определенные группы команд в различных ситуациях. Циклы и циклы с условием продолжения (раздел 6.4) позволяют повторить последовательность команд несколько раз.

Очевидно, что изменение потока команд программы требует некоторого процесса принятия решений в пределах программы. Компьютер должен решить, выполнить ли следующую команду или пропускать одну или более команд и продолжать выполнение на другой строке программы. Программа принимает эти решения, сравнивая значения переменных. Это делается с использованием операторов отношения и логических операторов, которые обсуждаются в разделе 6.1.

Нужно также отметить, что в программировании могут использоваться определяемые пользователем функции (представленные в главе 7). Определяемая пользователем функция может использоваться в качестве подпрограммы. Когда главная программа достигает командной строки, которая имеет определяемую пользователем функцию, она обеспечивает входные аргументы для этой функции и «ожидает» ее результатов. Определяемая пользователем функция выполняет

вычисления и возвращает результаты основной программе, которая после этого продолжает выполнение следующей команды.

6.1. Операторы сравнения и логические операторы

Оператор сравнения сравнивает два числа, определяя, является ли утверждение сравнения (например, $5 < 8$) истиной или ложью. Если утверждение – истина (true), он присваивает значение 1. Если утверждение – ложь (false), он присваивает значение 0. Логический оператор проверяет утверждения на истину/ложь и производит результат истина (true, 1) или ложь (false, 0) в соответствии с данным утверждением. Например, логический оператор AND дает 1, только если оба утверждения истины. Операторы сравнения и логические операторы могут использоваться в математических выражениях и, как будет показан в этой главе, часто используются в комбинации с другими командами, чтобы принять решения для управления потоком компьютерной программы.

Операторы сравнения

Операторы сравнения в MATLAB:

Оператор сравнения	Описание
<	Меньше чем
>	Больше чем
<=	Меньше или равно
>=	Больше или равно
==	Равно
~=	Не равно

Отметим, что оператор сравнения «равно» состоит из двух знаков = (без пробела между ними), поскольку один знак = это оператор присвоения. В других операторах отношения, которые состоят из двух символов, также нет пробела между символами (<=, >=, ~=).

- Операторы сравнения используются в качестве арифметических операторов в пределах математического выражения. Результат может использоваться в других математических операциях, в адресации (в индексах) массивов и вместе с другими командами MATLAB (например, if) для управления потоком программы.
- Когда сравниваются два числа, то результат есть 1 (логическая истина, true), если сравнение в операторе отношения является истинным и 0 (логическая ложь, false), если сравнение является ложным.

- Если сравниваются два скаляра, то результат есть скаляр 1 или 0. Если сравниваются два массива (одного и того же размера), то сравнение выполняется поэлементно и результатом является логический массив того же размера из единиц и нулей, согласно результатам сравнения в каждом адресе.
- Если скаляр сравнивается с массивом, то этот скаляр сравнивается с каждым элементом массива и результатом является логический массив из единиц и нулей, согласно результатам сравнения в каждом адресе.

Некоторые примеры:

```
>> 5>8                                Проверка 5 больше чем 8.  
ans =                                     Так как это не верно (5 не больше чем 8), то ответ 0.  
      0  
>> a=5<10                            Проверка, что 5 меньше 10 и присвоение ответа переменой а.  
a =                                         Так как это верно (5 меньше чем 10), число 1 присвоено  
      1                                         переменой а.  
>> y=(6<10)+(7>8)+(5*3==60/4)    Использование операторов отношения  
                                         в математическом выражении.  
                                         Равно 1 поскольку 6 меньше чем 10.  
                                         Равный 0 поскольку 7 не больше 8.  
                                         Равно 1 поскольку 5*3 равно 60/4.  
  
y =  
    2  
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];   Задание векторов b и с.  
>> d=c>=b    Проверка, какие элементы из с больше или равны элементам из b.  
d =           Присваивается 1, когда элемент из с больше или равен элемента из b.  
      0       1       1       0       1       1       0  
>> b == c    Проверка, какие элементы из с равны элементам из b.  
ans =          0       0       1       0       0       1       0  
>> b~=c     Проверка, какие элементы из с не равны элементам из b.  
ans =          1       1       0       1       1       0       1  
>> f=b-c>0  Вычитание с из b и проверка, какие элементы получаются больше нуля.  
f =            1       0       0       1       0       0       1  
>> A=[2 9 4; -3 5 2; 6 7 -1]        Задание 3×3 матрицы A.  
A =  
    2       9       4  
   -3      5       2  
    6       7      -1  
>> B=A<=2    Проверки, какие элементы из A меньше или равны 2.  
B =           1       0       0  
             1       0       1  
             0       0       1
```

- Результаты операций сравнения являются векторами из нулей и единиц, они называются логическими векторами и могут использоваться для адресации (индексации) векторов. Когда логический вектор используется в индексации другого вектора, он извлекает из того вектора элементы в тех позициях, где логический вектор имеет значение 1. Например:

```
>> r = [8 12 9 4 23 19 10]      Задание вектора r.
r =
    8     12     9     4     23     19     10
>> s=r<=10      Проверка, какие элементы r меньше или равны 10.
s =                  Логический вектор s с единицами в позициях, где элементы r
    1     0     1     1     0     0     1 меньше или равны 10.
>> t=r(s)      Использование s в индексах вектора r, для создания вектора t.
t =                  Вектор t состоит из элементов r в позициях, где s имеет значение 1.
    8     9     4     10
>> w=r(r<=10)      Та же самая процедура может быть сделана за один шаг.
w =
    8     9     4     10
```

- Числовые векторы и массивы с элементами из 0 и 1 – это не то же самое как логические векторы и массивы из 0 и 1. Числовые векторы и массивы не могут использоваться для адресации. В то же время, логические векторы и массивы могут использоваться в арифметических операциях. Перед использованием логического вектора или массива в арифметических операциях он изменяется на числовой вектор или массив.
- Порядок очередности: в математическом выражении, которое включает операции сравнения и арифметические операции, арифметические операции (+, -, *, /, \) имеют приоритет перед операциями сравнения. Сами операторы сравнения имеют равный приоритет и выполняются слева направо. Для изменения порядка очередности могут использоваться круглые скобки. Примеры:

```
>> 3+4<16/2      + и / выполняются первыми.
ans =
    1
Ответ 1, так как 7 < 8.
>> 3+(4<16)/2   Сравнение 4 < 16 выполняется сначала и равно 1, так как это истина.
ans =
    3.5000
Получается 3.5 как 3 + 1/2.
```

Логические операторы

Логические операторы в MATLAB:

- Логические операторы имеют числа в качестве operandов. Ненулевое число – истина, а нулевое число – ложь.

Логический оператор	Имя	Описание
& Пример: A&B	AND (И)	Действует на двух операндах (A и B). Если оба они истинные, то результат – истина (1); иначе – результат ложь (0).
 Пример: A B	OR (ИЛИ)	Действует на двух операндах (A и B). Если или один, или оба истинные, то результат – истина (1); иначе (оба – ложь) – результат ложь (0).
~ Пример: ~A	NOT (НЕ)	Действует на одном операнде (A). Дает отрицание (противоположность) операнда; результат есть истина (1), если operand – ложь, и результат есть ложь (0), если operand – истина.

- Логические операторы (как и операторы сравнения) используются в качестве арифметических операторов в пределах математического выражения. Результат может использоваться в других математических операциях, в адресации массивов и вместе с другими командами MATLAB (например, `if`) для управления потоком команд программы.
 - Логические операторы (как и операторы сравнения) могут использоваться со скалярами и массивами.
 - У логических операций AND и OR оба операнда могут быть как скаляры, так и массивы, или один массив, а другой – скаляр. Если оба операнда – скаляры, то результат – скаляр 0 или 1. Если оба массивы, то они должны быть одинакового размера и логическая операция выполняется поэлементно. Результат – массив того же размера с элементами 1 или 0, в соответствии с результатами операции в каждой позиции. Если один operand – скаляр, а другой – массив, то логическая операция выполняется между скаляром и каждым из элементов в массиве, а результат – массив того же размера из 1 и 0.
 - Логическая операция NOT имеет один operand. Когда она используется со скаляром, то результат есть скаляр 0 или 1. Когда она используется с массивом, результат – массив того же самого размера с нулями в позициях, где у массива есть ненулевые числа и с единицами в позициях, где у массива есть 0.

Приведем несколько примеров:

```
>> 3&7          3 AND 7  
ans =           3 и 7 оба истина (ненулевые), поэтому результат есть 1.  
      1  
>> a=5|0        5 OR 0 (присваивается переменной a).  
a =            Присвоено значение 1, поскольку по крайней мере одно  
      1           из чисел – истина (ненулевое).
```

```

>> ~25          NOT 25.
ans =           Результат 0, так как 25 истина (ненулевое)
    0           и противоположность — ложь (0).
>> t=25*( (12&0)+(~0)+(0|5) )      Использование логических операторов
t =             в математическом выражении.
    50
>> x=[9 3 0 11 0 15]; y=[2 0 13 -11 0 4]; Задание двух векторов x и y.
>> x&y        Результат есть вектор с 1 в каждой позиции где и x и y —
ans =           истина (ненулевые элементы) и 0 — в противном случае.
    1     0     0     1     0     1
>> z=x|y       Результат есть вектор с 1 в каждой позиции где
z =             или x или y — истина (ненулевые) и 0 — в противном случае.
    1     1     1     1     0     1
>> ~ (x+y)    Результат есть вектор с 0 в каждой позиции где
ans =           вектор x+y является истиной (ненулевым элементом) и 1 в
                  каждой позиции, где x+y является ложью (нулевой элемент).
    0     0     0     1     1     0

```

Порядок очередности

Арифметические, сравнения и логические операторы можно комбинировать в математических выражениях. Когда выражение имеет такую комбинацию, то результат зависит от порядка, в котором выполняются операции. Ниже представлен порядок, используемый MATLAB:

Приоритет	Операция
1 (наивысший)	Круглая скобка (если вложенные круглые скобки существуют, внутренние, имеют приоритет)
2	Возведение в степень
3	Логическое NOT (~)
4	Умножение, деление
5	Сложение, вычитание
6	Операторы сравнения (>, <, >=, <=, ==, ~=)
7	Логическое AND (&)
8 (самый низкий)	Логическое OR ()

Если две или более операций имеют один и тот же приоритет, выражение выполняется в порядке слева направо.

Отметим, что порядок, указанный выше, используется начиная с MATLAB 6. Предыдущие версии MATLAB использовали немного отличающийся порядок (& не имел приоритета над |), поэтому пользователь должен проявить осторожность. Проблем совместимости между различными версиями MATLAB можно избежать при использовании круглых скобок, даже когда они не требуются.

Ниже представлены примеры выражений, которые содержат арифметические, сравнения и логические операторы:

```

>> x=-2; y=5;
>> -5<x<-1
ans =
      0
>> -5<x & x<-1
ans =
      1
>> ~(y<7)
ans =
      0
>> ~y<7
ans =
      1
>> ~((y>=8) | (x<-1))
ans =
      0
>> ~((y>=8) | (x<-1))
ans =
      1
    
```

Задание переменных x и y .
Это неравенство корректно математически. Ответ, однако, ложь, так как MATLAB выполняет операции слева направо. $-5 < x$ истина ($=1$), а затем $1 < -1$ ложно (0).
Математически корректное утверждение получается при использовании логического оператора $\&$. Неравенства выполняются первыми. Так как оба они истинны (1), ответ 1 .
Сначала выполняется $y < 7$, это истина (1) и тогда ~ 1 есть 0 .
Сначала выполняется $\sim y$, y есть истина (1) (так как y ненулевое), ~ 1 есть 0 , а $0 < 7$ — истинно (1).
Сначала: $y \geq 8$ (ложь) и $x < -1$ (истина).
OR ($|$) выполняется после них и есть (истина).
 \sim выполняется последним и дает ложь (0).
Сначала выполняются $y \geq 8$ (ложь) и $x < -1$ (истина).
Затем выполняется NOT для ($y \geq 8$) и это есть истина (1).
OR выполняется последним и дает истину (1).

Встроенные логические функции

MATLAB имеет встроенные функции, которые эквивалентны логическим операторам. Это следующие функции:

and (A, B)	эквивалентна $A \& B$
or (A, B)	эквивалентна $A B$
not (A)	эквивалентна $\sim A$

Кроме того, MATLAB имеет другие логические встроенные функции, некоторые из которых описаны в следующей таблице:

Функция	Описание	Пример
xor (a, b)	Исключающее ИЛИ. Возвращает истину (1), если один операнд есть истина, а другой есть ложь.	<pre> >> xor(7, 0) ans = 1 >> xor(7, -5) ans = 0 </pre>
all (A)	Возвращает 1 (true) если все элементы в векторе A ненулевые (true). Возвращает 0 (false), если один или более элементов — ложь (0). Если A — матрица, то обрабатывает столбцы как векторы и возвращает вектор из единиц и нулей.	<pre> >> A=[6 2 15 9 7 11]; >> all(A) ans = 1 >> B=[6 2 15 9 0 11]; >> all(B) ans = 0 </pre>

Функция	Описание	Пример
any (A)	Возвращает 1 (true) если любой элемент в векторе A – ненулевой (true). Возвращает 0 (false), если все элементы – false (0). Если A – матрица, то обрабатывает столбцы как векторы и возвращает вектор из единиц и нулей.	<pre>>> A=[6 0 15 0 0 11]; >> any (A) ans = 1 >> B = [0 0 0 0 0 0]; >> any(B) ans = 0</pre>
find (A)	Если A – вектор, возвращает индексы ненулевых элементов.	<pre>>> A=[0 9 4 3 7 0 0 1 8]; >> find(A) ans = 2 3 4 5 8 9</pre>
find (A>d)	Если A – вектор, возвращает индексы элементов, которые больше чем d (может использоваться любой оператор сравнения).	<pre>>> find(A>4) ans = 2 5 9</pre>

Операции этих четырех логических операторов and, or, xor и not могут быть подытожены в таблице истинности:

Вход		Выход				
A	B	AND A&B	OR A B	XOR (A,B)	NOT ~A	NOT ~B
false	false	false	false	false	true	true
false	true	false	true	true	true	false
true	false	false	true	true	false	true
true	true	true	true	false	false	false

Пример задачи 6.1. Анализ температурных данных

Имеются значения ежедневных максимальных температур (в °F) в Вашингтоне, округ Колумбия, в течение апреля месяца 2002: 58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69 (данные от американского Национального управления океанических и атмосферных исследований). Используйте операции сравнения и логические операции, чтобы определить следующее:

- (a) Число дней, когда температура была выше 75°.
- (b) Число дней, когда температура была между 65° и 80°.
- (c) Дни месяца, когда температура была между 50° и 60°.

Решение

В следующем файле сценария значения температур вводятся как вектор. Затем используются выражения сравнения и логические выражения для анализа данных.

```

T=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 ...
    91 93 89 91 80 59 69 56 64 63 66 64 74 63 69];
Tabove75=T>=75;                                     Вектор с единицами в адресах, где T>=75.
NdaysTabove75=sum(Tabove75)                         Сложение всех единиц вектора Tabove75.
Tbetween65and80=(T>=65) & (T<=80);                 Вектор с единицами в адресах
                                                       где T>=65 и T<=80.
NdaysTbetween65and80=sum(Tbetween65and80)           Сложение всех единиц вектора Tbetween65and80.
datesTbetween50and60=find( (T>=50) & (T<=60) )      Функция find возвращает адреса элементов в T,
                                                       значения которых находятся между 50 и 60.

>> Exp6_1
NdaysTabove75 =                                         Для 7 дней температура была выше 75.
7
NdaysTbetween65and80 =                               Для 12 дней температура была между 65 и 80.
12
datesTbetween50and60 =                                Даты месяца с температурой между 50 и 60.
1          4          5          7          21         23
  
```

6.2. Условные операторы

Условный оператор (утверждение) – это команда, которая позволяет MATLAB принимать решение: выполнить ли группу команд, которые следуют за условным оператором, или пропустить эти команды. В условном операторе стоит условное выражение. Если это выражение – истина, то выполняется группа команд, которые следуют за утверждением. Если выражение – ложь, то компьютер пропускает группу. Общая форма условного оператора:

```
if условное выражение, состоящее из операторов сравнения
и/или логических операторов
```

Примеры:

```

if a < b
if c >= 5
if a == b
if a ~= 0
if (d<h) & (x>7)
if (x~=13) | (y<0)
```

Все переменные должны
иметь присвоенные значения.

- Условные операторы могут быть частью программы, записанной в файле сценария или определяемой пользователем функции (глава 7).
- Как показано ниже, для каждого оператора `if` имеется оператор `end`.

Оператор `if` обычно используется в трех структурах, `if-end`, `if-else-end` и `if-elseif-else-end`, которые описаны ниже.

6.2.1. Структура if-end

Структура условного оператора `if-end` схематично показана на рис. 6.1. Этот рисунок показывает, как вводятся команды в программе и блок-схему, которая символически показывает поток, или последовательность выполнения команд. Когда программа выполняется, она достигает оператора `if`. Если условное выражение в операторе `if` истинно (1), программа продолжает выполнять команды, которые следуют за оператором `if` полностью вниз до оператора `end`. Если условное выражение ложно (0), программа пропускает группу команд между `if` и `end` и продолжается с команд, которые следуют за `end`.

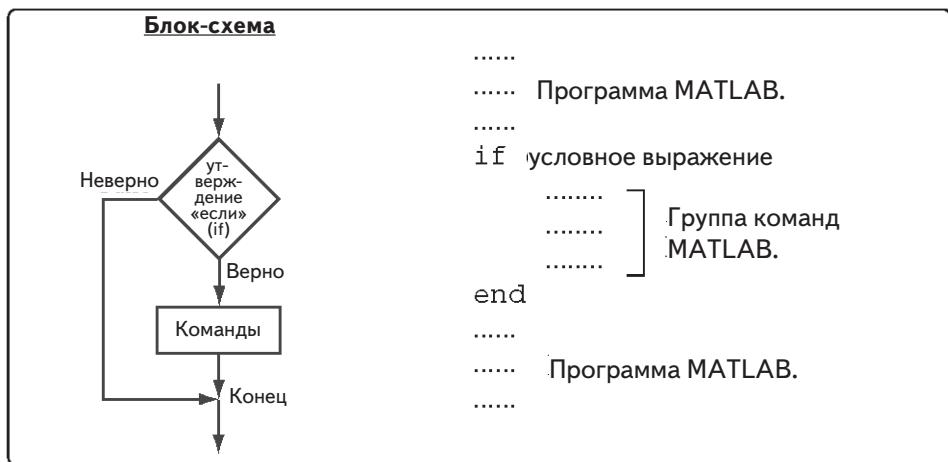


Рис. 6.1. Структура условного оператора `if-end`

Слова `if` и `end` появляются на экране в синем цвете и команды между утверждением `if` и утверждением `end` автоматически располагаются с отступом, что делает программу легче читаемой. Пример, где оператор `if-end` используется в файле сценария, показан в примере задачи 6.2.

Пример задачи 6.2. Вычисление зарплаты рабочего

Рабочему оплачивают согласно его почасовой заработной плате до 40 часов, а за сверхурочное время на 50% больше почасовой зарплаты. Напишите программу в файле сценария, которая вычисляет зарплату рабочему. Эта программа требует, чтобы пользователь ввел число часов и ставку почасовой зарплаты. Программа выводит на экран зарплату.

Решение

Программа в скрипт-файле показана ниже. Сначала программа вычисляет плату, умножая число часов на ставку почасовой зарплаты. Затем утверждение `if` проверяет, больше ли число часов чем 40. Если так, то выполняется следую-

щая строка и добавляется дополнительная плата за отработанные часы свыше 40. В противном случае программа пропускает все это до end.

```
t=input('Пожалуйста, введите число отработанных часов ');
h=input('Пожалуйста, введите почасовую зарплату в $ ');
Pay=t*h;
if t>40
    Pay=Pay+(t-40)*0.5*h;
end
fprintf('Зарплата рабочего есть $ %5.2f',Pay)
```

Ниже показано использование программы (в командном окне) для двух случаев (скрипт-файл был сохранен как Workerpay.m):

```
>> Workerpay
Пожалуйста, введите число отработанных часов 35
Пожалуйста, введите почасовую зарплату в $ 8
Зарплата рабочего есть $ 280.00
>> Workerpay
Пожалуйста, введите число отработанных часов 50
Пожалуйста, введите почасовую зарплату в $ 10
Зарплата рабочего есть $ 550.00
```

6.2.2. Структура if-else-end

Структура условного оператора if-else-end служит для того, чтобы выбрать для выполнения одну группу команд из возможных двух. Структура if-else-end показана на рис. 6.2. Он показывает как в программе вводятся команды и включает блок-схему, которая иллюстрирует поток, или последовательность выполнения команд. Первая строка if – это утверждение с выражением условия. Если это условное выражение – истина (true), то программа выполняет группу 1 из команд между операторами if и else и затем пропускает все команды между else и end. Если условное выражение – ложь (false), то программа пропускает команды до else и затем выполняет группу 2 команд между else и end.

6.2.3. Структура if-elseif-else-end

Структура условного оператора if-elseif-else-end показана на рис. 6.3. Он показывает как в программе вводятся команды и включает блок-схему, которая иллюстрирует поток, или последовательность выполнения команд. Эта структура включает два условных утверждения (if и elseif), которые позволяют выбрать для выполнения одну из трех групп команд. Первая строка – есть оператор if с условным выражением. Если условное выражение – истина (true), то программа выполняет первую группу команд между утверждениями if и elseif и затем пропускает все до end. Если условное выражение в операторе if есть ложь (false), то программа, пропускает все команды до утверждения elseif. Если условное вы-

ражение в утверждении `elseif` есть истина (`true`), программа выполняет группу 2 из команд между `elseif` и `else` и затем пропускает все остальные команды до `end`. Если условное выражение в операторе `elseif` есть ложь (`false`), то программа пропускает команды до оператора `else` и затем выполняет группу 3 из команд между `else` и `end`.

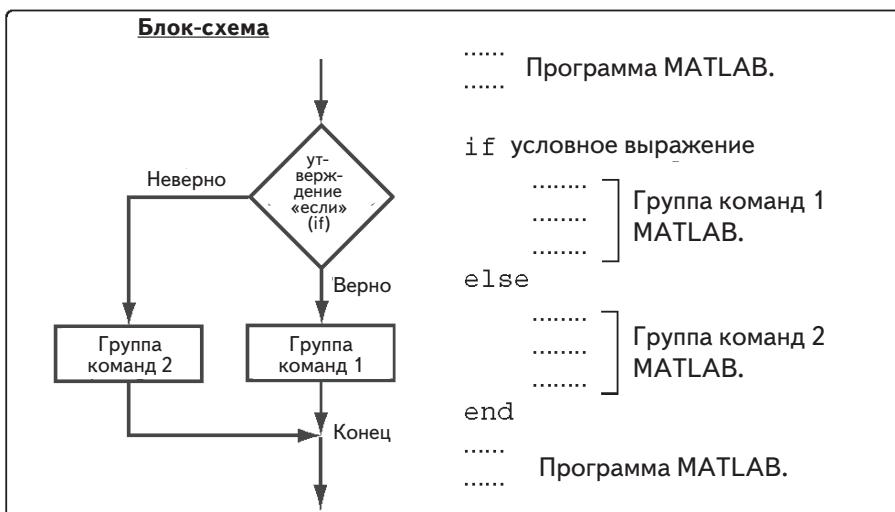


Рис. 6.2. Структура условного оператора if-else-end

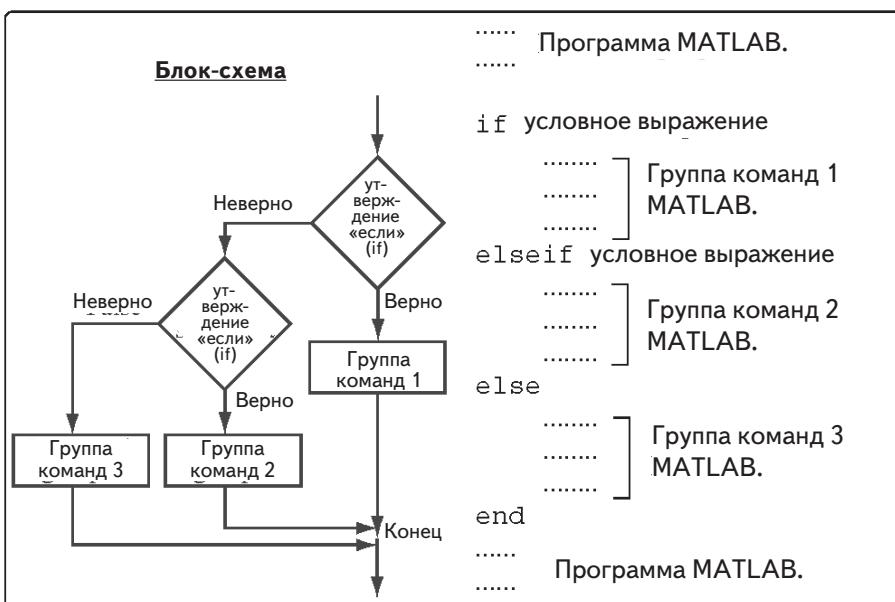


Рис. 6.3. Структура условного оператора if-elseif-else-end

Отметим, что можно добавить несколько утверждений `elseif` и соответствующих групп команд. Таким образом больше условий может быть включено. Кроме того, оператор `else` является необязательным. Это означает, что в случае нескольких операторов `elseif` и без оператора `else`, если какое-либо из условных утверждений `elseif` – истина, то выполняются соответствующие команды, в противном случае – ничего не выполняется из этой структуры.

Следующий пример использует в программе структуру `if-elseif-else-end`.

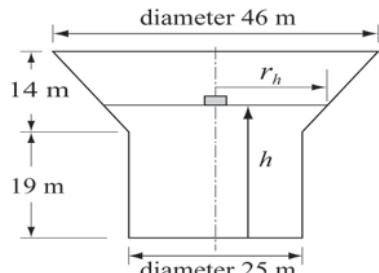
Пример задачи 6.3. Уровень воды в водонапорной башне

Корпус водонапорной башни имеет геометрию, показанную на рисунке (нижняя часть – цилиндр, а верхняя часть – перевернутый усеченный конус). В корпусе есть поплавок, который показывает уровень воды. Требуется написать программу MATLAB, которая определяет объем воды в корпусе по положению (высоте h) поплавка. Эта программа запрашивает пользователя введения значения высоты h в м, а в качестве результата выводит на экран объем воды в м^3 .

Решение

Для $0 \leq h \leq 19$ м объем воды определяется как объем цилиндра с высотой h : $V = \pi 12.5^2 h$.

Для $19 < h \leq 33$ м объем воды определяется как объем цилиндра высоты $h = 19$ м и объем воды в конусе:



$$V = \pi 12.5^2 \cdot 19 + \frac{1}{3} \pi (h-19)(12.5^2 + 12.5 \cdot r_h + r_h^2), \text{ где } r_h = 12.5 + \frac{10.5}{14}(h-19).$$

Вот программа:

```
% Программа вычисления объема воды в водонапорной башне.
h=input('Введите высоту поплавка в метрах ');
if h > 33
    disp('ОШИБКА. Высота не может быть больше 33 м.')
elseif h < 0
    disp('ОШИБКА. Высота не может быть отрицательной.')
elseif h <= 19
    v = pi*12.5^2*h;
    fprintf('Объем воды равен %7.3f кубических метров.\n',v)
else
    rh=12.5+10.5*(h-19)/14;
    v=pi*12.5^2*19+pi*(h-19)*(12.5^2+12.5*rh+rh^2)/3;
    fprintf('Объем воды равен %7.3f кубических метров.\n',v)
end
```

Ниже показано командное окно, когда программа использовалась с тремя различными значениями высоты воды.

```
Введите высоту поплавка в метрах 8
Объем воды равен 3926.991 кубических метров.
```

```
Введите высоту поплавка в метрах 25.7
Объем воды равен 14114.742 кубических метров.
```

```
Введите высоту поплавка в метрах 35
ОШИБКА. Высота не может быть больше 33 м.
```

6.3. Оператор переключения switch-case

Оператор переключения `switch-case` – это другой метод, который может использоваться для управления потоком команд программы. Он обеспечивает способ выбора одной группы команд для выполнения из нескольких возможных групп. Структура оператора показана на рис. 6.4.

```
..... Программа MATLAB.
.....  
  
switch выражение переключателя
    case значение1
        ..... ] Группа команд 1.
    case значение2
        ..... ] Группа команд 2.
    case значение3
        ..... ] Группа команд 3.
    otherwise
        ..... ] Группа команд 4.
end
..... Программа MATLAB.
```

Рис. 6.4. Структура оператора `switch-case`

- Первая строка – это команда переключателя, которая имеет вид:

```
switch выражение переключателя
```

Это выражение переключателя может быть скаляром или строкой. Обычно это переменная, которой присвоен скаляр или строка. Однако это может быть и математическое выражение, которое включает заданные переменные и может быть вычислено.

- После команды `switch` следуют одна или несколько команд `case`. Рядом с каждой такой командой стоит значение (может быть скаляр или строка: `value1`, `value2`, и т. д.) и ниже – соответствующая группа команд.
- После последней команды `case` может стоять необязательная команда `otherwise`, сопровождаемая своей группой команд.
- Последняя строка должна быть оператором `end`.

Как работает оператор переключения switch-case?

Значение выражения переключателя в команде `switch` сравнивается со значениями, которые стоят рядом с каждым выражением `case`. Если совпадение найдено, то выполняется соответствующая группа команд, которые следуют за выражением `case`. (Выполняется только одна группа команд – та, которая между совпадающим значением `case` и любой следующей командой `case`, `otherwise` или `end`.)

- Если имеется более одного совпадения, то выполняется только первый случай совпадения.
- Если не найдено ни одного совпадения и присутствует оператор `otherwise` (который является дополнительным), то выполняется группа команд между `otherwise` и `end`.
- Если не найдено ни одного совпадения и нет также оператора `otherwise`, то ни одна из групп команд не выполняется.
- Оператор `case` может иметь более одного значения. Это делается вводом значений в виде: `{value1, value2, value3...}`. (Этот вид данных, который не затронут в этой книге, называется массивом ячеек.) Тогда `case` выполняется, если по крайней мере одно из значений соответствует значению выражения переключателя.

Замечание: В MATLAB выполняется только первый случай совпадения. После группы команд, соответствующих первому случаю совпадения `case`, программа пропускает все остальное до оператора `end`. Это отличается от языка С, где требуются операторы выхода из цикла.

Пример задачи 6.4. Преобразование единиц измерения энергии

Написать программу в виде скрипта-файла, которая преобразовывает количество энергии (работы), данное в джоулях (Дж), футо-фунтах (фут-фунт), калориях (кал), или электрон-вольтах (эВ) к эквивалентному количеству в других единицах.

ницах, определенных пользователем. (Футофунт – это работа по поднятию одного фунта на один фут). Программа просит пользователя ввести количество энергии, ее текущие единицы измерения и требуемые новые единицы. Результат – количество энергии в новых единицах.

Коэффициенты преобразования:

$$1 \text{ Дж} = 0.738 \text{ фут-фунт} = 0.239 \text{ кал} = 6.24 \times 10^{18} \text{ эВ.}$$

Использовать программу для:

- (a) Преобразования 150 Дж в фут-фунт.
- (b) Преобразования 2,800 кал в Дж.
- (c) Преобразования 2.7 эВ в кал.

Решение

Программа включает два множества операторов `switch-case` и один оператор `if-else-end`. Первый оператор `switch-case` используется для преобразования входной величины в исходных единицах измерения – в джоули. Второй оператор `switch-case` используется для преобразования количества энергии в джоулях в указанные новые единицы. Оператор `if-else-end` используется для создания сообщения об ошибке, если единицы вводятся неправильно.

```

Ein=input('Введите значение энергии (работы) для преобразования: ');
EinUnits=input('Введите текущие единицы (J, ft-lb, cal или eV): ','s');
EoutUnits=input('Введите новые единицы (J, ft-lb, cal или eV): ','s');
error=0;
switch EinUnits
    case 'J'
        EJ=Ein;
    case 'ft-lb'
        EJ=Ein/0.738;          Каждое из четырех case имеет значение (строка), которое
                               соответствует одной из начальных единиц и команду,
                               которая преобразовывает Ein в единицы J.
    case 'cal'
        EJ=Ein/0.239;         (Присваивается значение переменной EJ.)
    case 'eV'
        EJ=Ein/6.24e18;
    otherwise               Присвоение 1 переменной error, если не найдено совпадений.
        error=1;              Возможно, если начальные единицы были введены неправильно.
end
switch EoutUnits
    case 'J'
        Eout=EJ;
    case 'ft-lb'
        Eout=EJ*0.738;       Каждое из четырех case имеет значение (строка), которое
                               соответствует одной из новых единиц и команду,
                               которая преобразовывает EJ в новые единицы.
    case 'cal'
        Eout=EJ*0.239;       (Присваивается значение Eout.)
    case 'eV'
        Eout=EJ*6.24e18;
    otherwise               Присвоение 1 переменной error, если не найдено совпадений.
        error=1;

```



```

error=1;      Возможно, если новые единицы были введены неправильно.
end
if error      Конструкция if-else-end. Если error – истина (ненулевая),
               на экран выводится сообщение об ошибке.
    disp('ОШИБКА: текущие или новые единицы введены неправильно.')
else          В противном случае, если error – ложь (нуль),
               на экран выводится преобразованная энергия.
    fprintf('E = %g %s',Eout,EoutUnits)
end
  
```

Ниже показан пример использования скрипта-файла (сохраненного как EnergyConversion) в командном окне для преобразования, указанного в пункте (b) этой задачи.

```

>> EnergyConversion
Введите значение энергии (работы) для преобразования: 2800
Введите текущие единицы (J, ft-lb, cal или eV): cal
Введите новые единицы (J, ft-lb, cal или eV): J
E = 11715.5 J
  
```

6.4. Циклы

Цикл – это другой метод для изменения потока компьютерной программы. В цикле, выполнение команды или группы команд, повторяется несколько раз последовательно. Каждый цикл выполнения называют проходом. При каждом проходе по крайней мере одна переменная, но обычно более одной, или даже все переменные, которые определены в пределах цикла, получает новые значения. В MATLAB'е есть два вида циклов. В циклах `for-end` (раздел 6.4.1) определено число проходов в начале цикла. В циклах `while-end` (раздел 6.4.2) заранее неизвестно число проходов, и циклический процесс продолжается до тех пор, пока не будет выполнено указанное условие. Оба вида циклов могут быть завершены в любое время командой прерывания `break` (см. раздел 6.6).

6.4.1. Циклы `for-end`

В циклах `for-end` выполнение команды, или группы команд повторяется predeterminedное число раз. Вид цикла показан на рис. 6.5.

- Переменный индекс цикла может иметь любое имя (обычно используются `i`, `j`, `k`, `m` и `n`, но `i` и `j` не должны использоваться, если MATLAB использует в программе комплексные числа).
- При первом проходе `k = f` и компьютер выполняет команды между `for` и `end`. Затем, программа возвращается к команде `for` для второго прохода. `k` получает новое значение, равное `k = f + s` и затем выполняются коман-

ды между `for` и `end` с новым значением k . Процесс повторяется до последнего прохода, когда $k = t$. После этого программа уже не возвращается к `for`, а продолжает выполнение команд, которые следуют за оператором окончания цикла `end`. Например, если $k = 1:2:9$, то будет пять циклов и соответствующие значения k следующие: 1, 3, 5, 7 и 9.

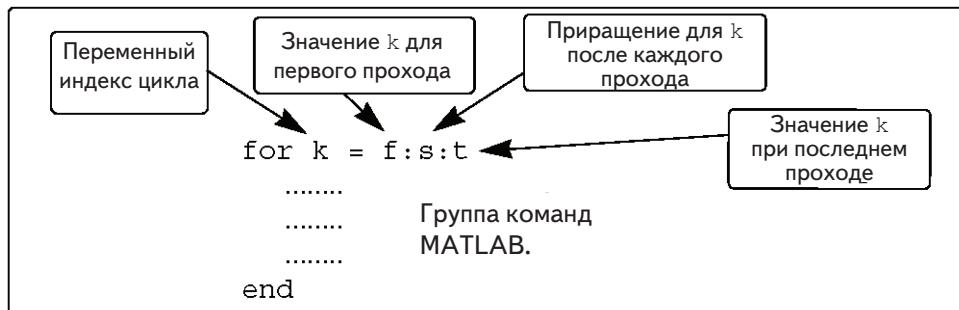


Рис. 6.5. Структура цикла `for-end`

- Инкремент s может быть отрицательным (то есть, например: $k = 25:-5:10$ производит четыре прохода с $k = 25, 20, 15, 10$).
- Если значение инкремента s опущено, то его значение равно 1 (значение по умолчанию) (то есть, например: $k = 3:7$ производит пять проходов с $k = 3, 4, 5, 6, 7$).
- Если $f = t$, цикл выполняется один раз.
- Если $f > t$ и $s > 0$, или если $f < t$ и $s < 0$, то цикл не выполняется.
- Если значения k , s и t – такие, что k не может быть равным t , то, если s положительно, последний проход – это тот, когда у k имеет наибольшее значение, которое меньше чем t (то есть, например, $k = 8:10:50$ производит пять проходов с $k = 8, 18, 28, 38, 48$). Если же s отрицательно, последний проход – тот, где k имеет наименьшее значение, которое больше чем t .
- В команде `for` счетчику k может быть также присвоено определенное значение (введенное как вектор). Пример: `for k = [7 9 -1 3 3 5]`.
- Значение k не должно переопределяться в пределах цикла.
- Для каждой команды `for` в программе должна быть команда `end`.
- Значение переменной индекса цикла (k) не выводится на экран автоматически. Можно вывести на экран это значение при каждом проходе (что иногда полезно для отладки), вводя k как одну из команд в цикле.
- Когда цикл заканчивается, переменная индекса цикла (k) имеет последнее присвоенное ей значение.

Простой пример `for-end` цикла (в файле сценария):

```

for k=1:3:10
    x = k^2
end

```

При работе этой программы цикл выполняется четыре раза. Значения счетчика k в четырех проходах: $k = 1, 4, 7$ и 10 , поэтому значения, которые присвоены переменной x в этих проходах – это следующие: $x = 1, 16, 49$ и 100 , соответственно. Поскольку точка с запятой не введена в конце второй строки, эти значения x выводятся на экран в командном окне при каждом проходе. При выполнении этого файла сценария в командном окне отображается следующее:

```
>> x =
    1
x =
    16
x =
    49
x =
    100
```

Пример задачи 6.5. Сумма ряда

(a) Использовать цикл `for-end` в файле сценария для вычисления суммы первых n членов ряда $\sum_{k=1}^n \frac{(-1)^k k}{2^k}$. Выполнить файл сценария для $n = 4$ и $n = 20$.

(b) Функция $\sin(x)$ может быть записана в виде ряда Тейлора:

$$\sin(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)}.$$

Написать определенную пользователем функцию, который вычисляет $\sin(x)$ с использованием ряда Тейлора. Использовать имя этой функции и аргументы как $y = \text{Tsin}(x, n)$. Входные аргументы – это угол x в градусах и число n членов в ряде. Использовать функцию для вычисления $\sin(150^\circ)$ используя три и семь членов.

Решение

(a) Файл сценария, который вычисляет сумму первых n членов ряда, показан ниже.

Суммирование выполнено с циклом. При каждом проходе вычисляется один член ряда (в первом проходе первый член, во втором проходе второй член, и так далее) и добавляется к сумме предыдущих элементов.

```
n=input('Введите число членов ' );
S=0;                                Начальное значение суммы – нуль.
for k=1:n                            Цикл for-end.
    S=S+(-1)^k*k/2^k;                При каждом проходе вычисляется один член ряда
end                                    и добавляется к сумме предыдущих элементов.
fprintf('Сумма ряда равна: %f',S)
```

Файл сохранен как `Expr6_5a` и затем выполнен дважды в командном окне:

```
>> Expr6_5a
Введите число членов 4
Сумма ряда равна: -0.125000
>> Expr7_5a
Введите число членов 20
Сумма ряда равна: -0.222216
```

(b) Файл определенной пользователем функции для вычисления $\sin(x)$ сложением n членов ряда Тейлора, показан ниже.

```
function y = Tsin(x,n)
% Tsin вычисляет синус, используя формулу Тейлора.
% Входные аргументы:
% x угол в градусах, n число членов.
xr=x*pi/180;           Преобразование угла из градусов в радианы.
y=0;
for k=0:n-1            Цикл for-end.
    y=y+(-1)^k*xr^(2*k+1)/factorial(2*k+1);
end
```

Первый элемент соответствует $k = 0$, это означает что для сложения n членов ряда нужно в последнем цикле иметь $k = n - 1$. Использование функции в командном окне для вычисления $\sin(150^\circ)$ с использованием трех и семи членов:

<pre>>> Tsin(150, 3) ans = 0.6523</pre>	Вычисления $\sin(150^\circ)$ с тремя членами ряда Тейлора.
<pre>>> Tsin(150, 7) ans = 0.5000</pre>	Вычисления $\sin(150^\circ)$ с семью членами ряда Тейлора.

Замечание о циклах `for-end` и поэлементных операциях

В некоторых ситуациях тот же самый конечный результат может быть получен либо с использованием циклов `for-end`, либо с использованием поэлементных операций. Пример задачи 6.5 иллюстрирует, как работает цикл `for-end`, но задача может быть также решена с использованием поэлементных операций (см. задачи 7 и 8 в разделе 3.9). Поэлементные операции с массивами – это одно из главных преимуществ MATLAB, которое обеспечивает средства для вычисления величин в ситуациях, когда обычно требуются циклы. Вообще, поэлементные операции быстрее чем циклы и они рекомендуются, если может использоваться любой метод.

Пример задачи 6.6. Изменение элементов вектора

Дан вектор $V = [5, 17, -3, 8, 0, -7, 12, 15, 20, -6, 6, 4, -7, 16]$. Написать программу как скрипт-файл, которая удваивает положительные элементы, которые делятся на 3 или 5, и возводит в степень 3 элементы, которые отрицательны, но больше чем -5 .

Решение

Задача решается с использованием цикла `for-end`, у которого внутри есть условная конструкция `if-elseif-end`. Число проходов равно числу элементов у вектора. При каждом проходе один очередной элемент проверяется условным оператором. Этот элемент изменяется, если он удовлетворяет условиям задачи. Скрипт-файл Программы, которая выполняет необходимые операции:

```

V=[5, 17, -3, 8, 0, -7, 12, 15, 20 -6, 6, 4, -2, 16];
n=length(V);                                     Полагаем n равным числу элементов V.
for k=1:n                                       Цикл for-end.
    if V(k)>0 & (rem(V(k),3) == 0 | rem(V(k),5) == 0)
        V(k)=2*V(k);
    elseif V(k) < 0 & V(k) > -5
        V(k)=V(k)^3;
    end
end
V
  
```

Файл сохранен как `Exp7_6` и затем выполнен в командном окне:

```

>> Exp7_6
V =
    10   17   -27   8   0   -7   24   30   40   -6   12   4   -8   16
  
```

6.4.2. Циклы `while-end`

Циклы `while-end` используются в ситуациях, когда образование цикла необходимо, но число проходов заранее не известно. В циклах `while-end` не определено число проходов, когда запускается циклический процесс. Вместо этого циклический процесс продолжается до тех пор, пока не будет удовлетворено установленное условие. Структуру цикла `while-end` показана на рис. 6.6.

Первая строка – это утверждение `while`, которое включает условное выражение. Когда программа достигает этой строки, проверяется условное выражение. Если оно ложно (0), то MATLAB пропускает все до утверждения `end` и продолжает программу. Если условное выражение истинно (1), MATLAB выполняет группу команд, которые следуют между операторами `while` и `end`. Затем MATLAB пере-

ходит назад к оператору `while` и снова проверяет условное выражение. Этот циклический процесс продолжается до тех пор, пока условное выражение не станет ложным.

```

while условное выражение
    .....
    .....
    .....
end
```

Рис. 6.6. Структура цикла `while-end`

Для цикла `while-end` имеют место свойства:

- Условное выражение в команде `while` должно включать по крайней мере одну переменную.
- Переменным в условном выражении должны быть присвоены значения, когда MATLAB выполняет команду `while` впервые.
- По крайней мере одной из переменных в условном выражении должно быть присвоено новое значение при выполнении команд, которые стоят между `while` и `end`. Иначе при запуске цикла он никогда не остановится, так как условное выражение всегда останется истинным.

Пример простого цикла `while-end` показан в следующей программе. В этой программе переменная `x` с начальным значением 1 удваивается при каждом проходе, пока ее значение меньше или равно 15.

```
x=1
while x<=15
    x=2*x
end
```

Начальное значение `x` есть 1.
Следующая команда выполняется только если `x<=15`.
При каждом проходе `x` удваивается.

При выполнении этой программы в командном окне отображается следующее:

```
x =
1
x =
2
x =
4
x =
8
x =
16
```

Начальное значение `x`.
При каждом проходе `x` удваивается.
Когда `x = 16`, условное выражение
в команде `while` – ложь и процесс останавливается.

Важное примечание

При использовании `while-end` цикла программист должен убедиться, что переменной (или переменным), которые находятся в условном выражении и которым присваиваются новые значения во время циклического процесса, будут в конечном счете присвоены такие значения, которые делают условное выражение в команде `while` ложным. Иначе циклы будут продолжаться неопределенно долго (неопределенный цикл). В примере выше, если условное выражение изменить на $x \geq 0.5$, то образование циклов будет продолжаться неопределенно долго. Такой ситуации можно избежать, если считать проходы и останавливать образование цикла, когда число проходов превышает некоторое большое значение. Это может быть сделано добавлением максимального количества проходов в условное выражение, или с использованием команды прерывания работы (раздел 6.6).

Так как никто не застрахован от ошибок, ситуация неопределенного цикла может случиться несмотря на осторожное программирование. Если это произошло, пользователь может остановить выполнение неопределенного цикла, нажимая клавиши **Ctrl+C** или **Ctrl+Break**.

Пример задачи 6.7. Представление функции рядом Тейлора

Функция $f(x) = e^x$ может быть представлена рядом Тейлора $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$.

Написать программу в файле сценария, которая определяет e^x с использованием представления в виде ряда Тейлора. Программа вычисляет e^x складывая члены ряда и останавливается, когда абсолютное значение последнего добавленного члена будет меньше чем 0.0001. Использовать цикл `while-end`, но ограничить число проходов числом 30. Если при 30-ом проходе значение последнего добавленного члена не меньше чем 0.0001, программа останавливается и выводит на экран сообщение, что необходимо более 30 членов.

Использовать программу для вычисления e^2 , e^{-4} и e^{21} .

Решение

Несколько первых членов ряда Тейлора:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

Программа, которая использует ряд для вычисления функции, показана ниже. Эта программа просит, чтобы пользователь ввел значение x . Затем первому члену a_0 присвоено значение 1 и сумме s присвоено a_0 . Далее, начиная со второго члена ($n = 1$), программа использует цикл `while` для вычисления n -го члена ряда и прибавления его к сумме. Программа также считает число членов n . Условное выражение в команде `while` является истинным пока абсолютное значение n -го члена ряда больше чем 0.0001 и число проходов n меньше чем 30. Это означает это, если 30-й член не меньше чем 0.0001, цикл останавливается.

```

x=input('Введите x ');
n=1; an=1; S=an;
while abs(an) >= 0.0001 & n <= 30
    an=x^n/factorial(n);
    S=S+an;
    n=n+1;
end
if n >= 30
    disp('Необходимо более 30 членов')
else
fprintf('exp (%f) = %f',x,S)
fprintf('\n Число использованных членов: %i',n)
end

```

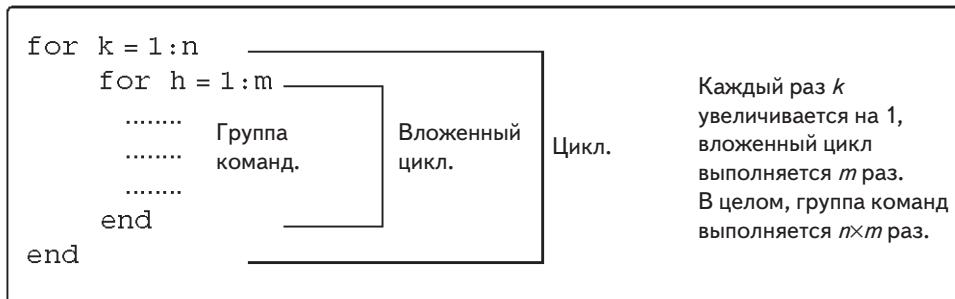
Запуск цикла while.
Вычисление n -го члена.
Добавление n -го члена к сумме.
Подсчет числа проходов.
Конец цикла while.
Конструкция if-else-end.

Эта программа использует конструкцию if-else-end для вывода на экран результатов. Если образование цикла остановилось в случае, когда 30-й член не меньше чем 0.0001, на экран выводится сообщение, указывающее на это. Если значение функции вычислено успешно, на экран выводится значение функции и число использованных членов. При выполнении программы число проходов зависит от значения x . Программа (сохраненная как `expx`) используется для вычисления e^2 , e^{-4} и e^{21} :

>> expx	
Введите x 2	Вычисление exp(2).
exp(2.000000) = 7.389046	
Число использованных членов: 12	Потребовалось 12 членов.
>> expx	
Введите x -4	Вычисление exp(-4).
exp(-4.000000) = 0.018307	
Число использованных членов: 18	Потребовалось 18 членов.
>> expx	
Введите x 21	Вычисление exp(21).
Необходимо более 30 членов	

6.5. Вложенные циклы и вложенные условные операторы

Циклы и условные операторы могут быть вложены в другие циклы или условные конструкции. Это означает, что цикл и/или условный оператор могут начаться (и закончиться) в пределах другого цикла или условного оператора. Нет ограничений на число вложенных циклов и условных утверждений. Нужно помнить, однако, что каждому утверждению if, case, for и while должно соответствовать утверждение end. Рис. 6.7 показывает структуру вложенного for-end цикла пределах другого цикла for-end.

**Рис. 6.7.** Структура вложенных циклов

В циклах, показанных на этом рисунке, если например, $n = 3$ и $m = 4$, то сначала $k = 1$ и вложенный цикл выполняется четыре раза для $h = 1, 2, 3, 4$. Затем $k = 2$ и вложенный цикл выполняется снова четыре раза для $h = 1, 2, 3, 4$. Наконец $k = 3$ и вложенный цикл выполняется снова четыре раза. Каждый раз, когда вводится вложенный цикл, MATLAB автоматически располагает новый цикл с отступом относительно внешнего цикла. Вложенные циклы и условные утверждения демонстрируются в следующей типовой задаче.

Пример задачи 6.8. Создание матрицы при помощи цикла

Написать программу в файле сценария, которая создает $n \times m$ матрицу с элементами, которые имеют следующие значения. Значение каждого элемента в первой строке есть номер столбца. Значение каждого элемента в первом столбце есть номер строки.

Каждый из остальных элементов имеет значение, равное сумме значений элемента выше и элемента слева. При выполнении программы просит, чтобы пользователь ввел значения n и m .

Решение

У приведенной ниже программы, есть два цикла (один вложенный) и вложенная if-elseif-else-end структура. Элементы в матрице присваиваются строка за строкой. Переменная индекса первого цикла, k , является номером строки, а переменная индекса второго цикла, h , является номером столбца.

<code>n=input ('Введите число строк');</code>	
<code>m=input ('Введите число столбцов');</code>	
<code>A=[];</code>	Задание пустой матрицы.
<code>for k=1:n</code>	Начало первого for-end цикла.
<code>for h=1:m</code>	Начало второго for-end цикла.
<code>if k==1</code>	Начало условной конструкции.
<code>A(k,h)=h;</code>	Присвоение значений элементам первой строки. ➔

```

elseif h==1
    A(k,h)=k;      Присвоение значений элементам первого столбца.
else
    A(k,h)=A(k,h-1)+A(k-1,h);  Присвоение значений другим элементам.
end          Конец (end), утверждения if.
end          Конец вложенного for-end цикла.
end          Конец первого for-end цикла.
A

```

Эта программа выполняется в командном окне для создания 4×5 матрицы.

```

>> Chap6_exp8
Введите число строк      4
Введите число столбцов     5
A =
 1   2   3   4   5
 2   4   7  11  16
 3   7  14  25  41
 4  11  25  50  91

```

6.6. Команды **break** и **continue**

Команда прерывания работы **break**

Если она в цикле (`for` или `while`), команда `break` завершает выполнение цикла (всего цикла, не только остатка прохода). Когда в цикле встречается команда `break`, MATLAB переходит к команде `end` цикла и продолжает выполнение со следующей команды (не возвращаясь к команде `for` этого цикла).

- Если команда `break` во вложенном цикле, то завершается только вложенный цикл.
- Когда команда `break` появляется вне цикла в скрипте или файле функции, она завершает выполнение файла.
- Команда `break` обычно используется в пределах условного утверждения. В циклах она обеспечивает метод завершения циклического процесса, когда выполняется некоторое условие – например, если число циклов превышает заданное значение, или ошибка в некоторой числовой процедуре меньше чем заданное значение. Когда эта команда вводится вне цикла, она дает возможность завершить выполнение файла, например, когда данные, переданные в файл функции противоречат тому, что ожидается.

Команда продолжения **continue**

Команда `continue` может использоваться в цикле (`for` или `while`), чтобы остановить данный проход и запустить следующий проход циклического процесса.

- Обычно команда `continue` обычно – это часть условного утверждения. Когда MATLAB достигает команды `continue`, он не выполняет оставшиеся команды в цикле, а пропускает все до команды `end` цикла и затем запускает новый проход.

6.7. Примеры приложений MATLAB

Пример задачи 6.9. Снятие средств с пенсионного счета

Пенсионер вносит 300 000 \$ на счет для сохранения под 5% годовых. Он планирует снимать деньги со счета один раз в год. Вначале он снимает 25 000 \$ после первого года, а в следующие годы он увеличивает снимаемую сумму в соответствии с ростом инфляции. Например, если инфляция в год составляет 3%, он снимает 25 750 \$ после второго года. Вычислить число лет, в течение которых на счете будут оставаться деньги, принимая постоянный ежегодный рост инфляции 2%. Создать график, который показывает ежегодные выводы средств со счета за эти годы.

Решение

Задача решается с использованием цикла (`while` цикл, так как число проходов не известно до начала цикла). В каждом проходе вычисляется сумма, которая будет сниматься и баланс счета. Цикл продолжается, пока баланс счета больше чем или равен суммы, которое будет снята. Ниже представлена программа в виде скрипта-файла, которая решает задачу. В этой программе `year` – это вектор, каждый элемент которого есть число лет, `W` – это вектор с суммами, снимаемыми каждый год и `AB` – это вектор с балансом счета каждый год.

```

rate=0.05; inf=0.02;
clear W AB year
year(1)=0;
W(1)=0;
AB(1)=300000;
Wnext=25000;
ABnext=300000*(1 + rate);
n=2;
while ABnext >= Wnext
    year(n)=n-1;
    W(n)=Wnext;
    AB(n)=ABnext-W(n);
    ABnext=AB(n)*(1+rate);
    Wnext=W(n)*(1+inf);
    n=n+1;
end
fprintf('Деньги на счете будут %f лет',year(n-1))
bar(year,[AB' W'],2.0)

```

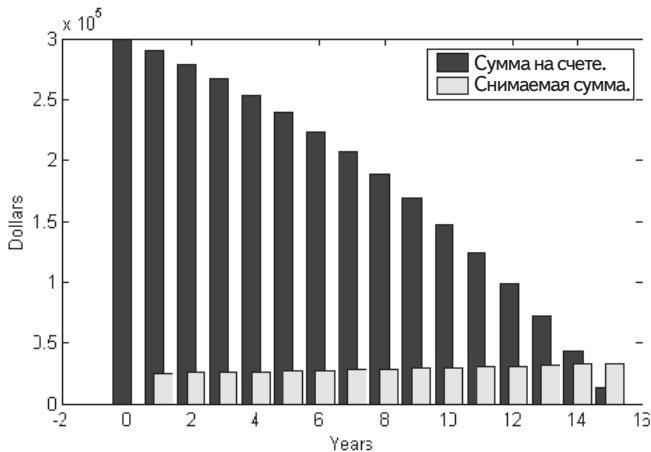
Первый элемент `year` есть 0.
 Начальная сумма снятия.
 Начальная сумма на счете.
 Сумма, которое будет снята после 1 года.
 Баланс счета после первого года.

`while` проверяет, больше ли следующий баланс, чем следующая снимаемая сумма.
 Сумма, снятая в $n-1$ году.
 Баланс счета в $n-1$ году после снятия.
 Баланс счета через год.
 Снимаемая сумма через год.

Программа выполняется следующим образом в командном окне:

```
>> Chap6_exp9
Деньги на счете будут 15 лет.
```

Программа также генерирует следующий рисунок (метки осей и легенда были добавлены к графику с использованием редактора графика).



Пример задачи 6.10. Создание случайного списка

Шесть певцов: Джон, Мэри, Трейси, Майк, Кейти и Дэвид – выступают на конкурсе. Написать программу MATLAB, которая создает список выступающих певцов в произвольном порядке.

Решение

Каждому имени присвоим целое число (1 – 6) (1 – Джону, 2 – Мэри, 3 – Трейси, 4 – Майку, 5 – Кейти и 6 – Дэвиду). Приведенная ниже программа сначала создает список целых чисел от 1 до 6 в произвольном порядке. Эти целые числа являются составляющими вектор с шестью элементами. Это делается с использованием встроенной функции MATLAB `randi` (см. раздел 3.7) для присвоения целых чисел элементам вектора. Чтобы удостовериться, что все целые числа элементов отличаются друг от друга, целые числа присваиваются один за другим. Каждое целое число, которое предложено функцией `randi` сравнивается со всеми целыми числами, которые были присвоены предыдущим элементам. Если найдено совпадение, то целое число не присваивается и `randi` используется снова для того, чтобы предложить новое целое число. Так как каждое имя певца ассоциировано с целым числом, после составления списка целых чисел используется переключатель `switch-case` для того, чтобы создать соответствующий список имени.

```

L(1)=randi(n);
for p=2:n
    L(p)=randi(n);
    r=0;
    while r==0
        r=1;
        for k=1:p-1
            if L(k)==L(p)
                L(p)=randi(n);
                r=0;
            end
        end
    end
    for i=1:n
        switch L(i)
            case 1
                disp('Джон')
            case 2
                disp('Мэри')
            case 3
                disp('Трейси')
            case 4
                disp('Майк')
            case 5
                disp('Кейти')
            case 6
                disp('Дэвид')
        end
    end
end

```

Присвоение первого целого числа $L(1)$.

Присвоение следующего целого числа $L(p)$.

Обнуление r .

См. пояснение ниже.

Полагаем $r=1$.

for цикл сравнивает целое число, присвоенное $L(p)$ с целыми числами, которые были уже присвоены предыдущим элементам.

if совпадение найдено, новое $L(p)=randi(n)$; целое число присваивается $L(p)$ и r обнуляется.

break Выход из вложенного цикла for. Возврат к началу цикла while. Поскольку $r=0$, снова запускается вложенный цикл проверяет равно ли новое целое число $L(p)$, целым числам, которые были уже присвоены.

Переключатель switch-case составляет список имен в соответствии значениями элементов вектора L .

Цикл `while` проверяет, что каждое новое целое число (элемент), которое должно быть добавлено к вектору L , не равно любому из целых чисел в элементах имеющихся уже в векторе L . Если найдено совпадение, он продолжает генерировать новые целые числа, пока новое целое число не будет отличаться от всех целых чисел, которые уже находятся в L .

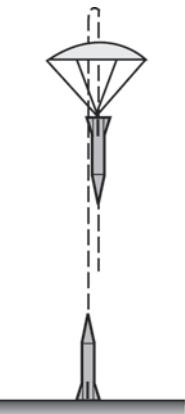
Когда программа выполняется, на экран в командном окне выводится следующее. Очевидно, что при каждом запуске программы список будет выводиться в другом порядке.

Порядок выступлений:

Кейти
Трейси
Дэвид
Мэри
Джон
Майк

Пример задачи 6.11. Полет модели ракеты

Полет модели ракеты может быть смоделирован следующим образом. Во время первых 0.15 сек ракету движет вверх ракетный двигатель силой 16 Н. Затем ракета летит по инерции, замедляясь под действием силы тяжести. После того, как она достигает верхней точки, ракета начинает падать вниз. Когда скорость падения достигает 20 м/с, открывается парашют (его открытие предполагается моментальным) и ракета продолжает падение с постоянной скорости 20 м/с, пока не достигнет земли. Написать программу, которая вычисляет и графически изображает скорость и высоту ракеты как функции времени во время полета.



Решение

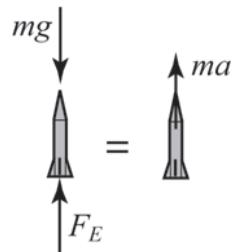
Предполагается, что ракета является материальной частицей, которая движется по прямой в вертикальной плоскости. Для движения с постоянным ускорением вдоль прямой, скорость и положение как функция времени определяются формулами:

$$v(t) = v_0 + at \quad \text{и} \quad s(t) = s_0 + v_0 t + \frac{1}{2} a t^2,$$

где v_0 и s_0 – начальная скорость и положение, соответственно. В компьютерной программе полет ракеты разбит на три участка. Каждый участок движения вычисляется в некотором цикле. При каждом проходе время увеличивается на некоторое приращение.

Участок 1: Первые 0.15 сек, когда работает ракетный двигатель. В течение этого периода ракета движется вверх с постоянным ускорением. Это ускорение находится из движения свободного тела под действием силы и ускорения силы тяжести (рисунок показан справа). Из второго закона Ньютона, сумма сил в вертикальном направлении равна массе, умноженной на ускорение (уравнение равновесия):

$$+\uparrow \sum F = F_E - mg = ma .$$



Решение этого уравнения относительно ускорения дает:

$$a = \frac{F_E - mg}{m} .$$

Скорость и высота как функции времени:

$$v(t) = 0 + at \quad \text{и} \quad h(t) = 0 + 0 + \frac{1}{2} a t^2 ,$$

где начальная скорость и начальное положение – нулевые. В компьютерной программе этот участок движения начинается в $t = 0$ и цикл продолжается пока $t < 0.15$ с. Время, скорость и высота в конце этого сегмента есть t_1 , v_1 и h_1 .



Участок 2: Движение от остановки двигателя до момента открытия парашюта. На этом участке ракета движется с постоянным замедлением g . Скорость и высота ракеты как функции времени определяются формулами:

$$v(t) = v_1 - g(t - t_1) \quad \text{и} \quad h(t) = h_1 + v_1(t - t_1) - \frac{1}{2}g(t - t_1)^2.$$

На этом участке цикл продолжается до тех пор, пока скорость ракеты не составит -20 м/с (отрицательна, так как ракета движется вниз). Время и высота в конце этого участка есть t_2 и h_2 .

Участок 3: Движение от момента открытия парашюта, до момента, когда ракета достигает земли. На этом участке ракета движется с постоянной скоростью (нулевое ускорение). Высота как функция времени задается формулой $h(t) = h_2 - v_{chute}(t - t_2)$, где v_{chute} – это постоянная скорость после открытия парашюта. На этом участке цикл продолжается до тех пор, пока высота больше нуля. Программа в файле сценария, которая выполняет вычисления, показана ниже.

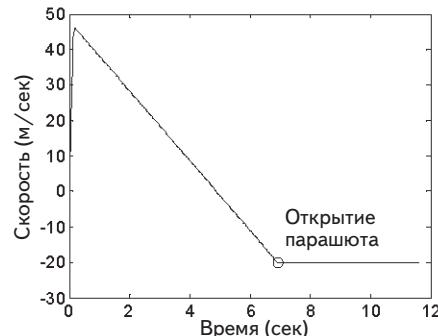
```
m=0.05; g=9.81; tEngine=0.15; Force=16; vChute=-20; Dt=0.01;
clear t v h
n=1;
t(n)=0; v(n)=0; h(n)=0;
% Участок 1
a1=(Force-m*g)/m;
while t(n) < tEngine & n < 50000
    n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=a1*t(n);
    h(n)=0.5*a1*t(n)^2;
end
v1=v(n); h1=h(n); t1=t(n);
% Участок 2
while v(n) >= vChute & n < 50000
    n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=v1-g*(t(n)-t1);
    h(n)=h1+v1*(t(n)-t1)-0.5*g*(t(n)-t1)^2;
end
v2=v(n); h2=h(n); t2=t(n);
% Участок 3
while h(n) > 0 & n < 50000
    n=n+1;
    t(n)=t(n-1)+Dt;
    v(n)=vChute;
    h(n)=h2+vChute*(t(n)-t2);
end
subplot(1,2,1)
plot(t,h,t2,h2,'o')
subplot(1,2,2)
plot(t,v,t2,v2,'o')
```

Первый цикл while.

Второй цикл while.

Третий цикл while.

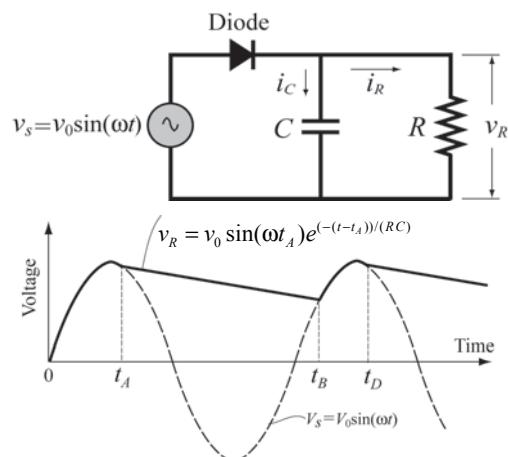
Точность результатов зависит от величины приращения времени Δt . Приращение 0.01 сек, кажется, дает хорошие результаты. Условное выражение в команде `while` также включает условие для n (если n больше чем 50 000, то цикл останавливается). Это предусмотрено для того, чтобы избежать бесконечного цикла в случае, если есть ошибка в операторах внутри цикла. Графики, сгенерированные программой, показаны ниже (метки осей и текст были добавлены с использование редактора графика).



Замечание: Задача может быть решена и запрограммирована разными способами. Здесь показан один вариант. Например, вместо того, чтобы использовать, циклы `while`, можно вычислить сначала время, когда открывается парашют и когда ракета достигает земли, а затем использоваться циклы `for-end` вместо циклов `while`. Если сначала определены эти моменты времени, можно также использовать поэлементно вычисления вместо циклов.

Пример задачи 6.12. Преобразователь переменного тока в постоянный

Полуволновой диодный выпрямитель – это электрическая цепь, которая преобразует напряжение переменного тока в напряжение постоянного тока. Цепь выпрямителя, которая состоит из источника напряжения переменного тока, диода, конденсатора и нагрузки (резистор), показана на рисунке. Напряжение источника есть $v_s = v_0 \sin(\omega t)$, где $\omega = 2\pi f$, где f – частота. Работа цепи показана на нижнем рисунке, где пунктирная линия показывает исходное напряжение, а сплошная линия показывают напря-



жение через резистор. В первом цикле от $t = 0$ до $t = t_A$ диод включен (проводит ток). В этот момент времени диод выключается и питание к резистору подается разряжающимся конденсатором. В момент $t = t_B$ диоде включается снова и продолжает проводить ток до момента $t = t_D$. Этот цикл продолжается, пока есть источник напряжения. В этом упрощенном анализе цепи диод предполагается идеальным, а конденсатор первоначально (при $t = 0$) не заряжен. Когда диод включен, напряжением резистора и ток определяются формулами:

$$v_R = v_0 \sin(\omega t) \quad \text{и} \quad i_R = v_0 \sin(\omega t)/R.$$

Ток в конденсаторе:

$$i_C = \omega C v_0 \cos(\omega t)/R.$$

Когда диод выключен, напряжение через резистор определяется формулой:

$$v_R = v_0 \sin(\omega t_A) e^{(-(t-t_A))/(RC)}.$$

Моменты времени, когда диод выключается (t_A, t_D , и так далее) вычисляются из условия. Диод включается снова, когда напряжение от источника достигает напряжения через резистор (время t_B на рисунке).

Написать программу MATLAB, которая графически изображает напряжение v_R через резистор и напряжение источника v_s как функции времени для $0 \leq t \leq 70$ мсек. Сопротивление нагрузки равно 1,800 Ом, напряжение источник $v_0 = 12$ В и $f = 60$ Гц. Чтобы исследовать эффект емкости конденсатора на напряжении через загрузку, выполните программу дважды, один – при $C = 45$ мкФ, а другой – при $C = 10$ мкФ.

Решение

Программа, которая решает эту задачу, представлена ниже. Она состоит из двух частей – первая вычисляет напряжение, когда диод включен, а вторая – когда диод выключен. Для переключения между этими двумя частями используется команда переключателя `switch`. Вычисления начинаются когда диод включен на (переменная `state='on'`), а когда $i_R - i_C \leq 0$, значение `state` меняется на '`off`', и программа переключается на команды, которые вычисляют v_R для этого состояния. Эти вычисления продолжаются до тех пока не будет достигнуто условие $v_s \geq v_R$, когда программа переключается назад к уравнениям, которые справедливы когда диод в состоянии '`on`'.

```
V0=12; C=45e-6; R=1800; f=60;
Tf=70e-3; w=2*pi*f;
clear t VR Vs
t=0:0.05e-3:Tf;
n=length(t);
state='on'          Присвоение 'on' переменной state.
for i=1:n
    Vs(i)=V0*sin(w*t(i));   Вычисление напряжения источника в момент t.
    switch state
        case 'on'
            iR=v0*sin(w*t(i))/R;
            if iR <= 0
                state='off';
            else
                VR=v0*sin(w*t(i));
            end
        case 'off'
            VR=v0*sin(w*t(i)) * exp(-(t(i)-tA)/(RC));
    end
end
```



```

case 'on'
VR(i)=Vs(i);
iR=Vs(i)/R;
iC=w*C*V0*cos(w*t(i));
sumI=iR+iC;
if sumI <= 0
    state='off ';
    tA=t(i);
end
case 'off '
VR(i)=V0*sin(w*tA)*exp(-(t(i)-tA)/(R*C));
if Vs(i) >= VR(i)
    state='on';
end
plot(t,Vs,':',t,VR,'k','linewidth',1)
xlabel('Время (сек)'); ylabel('\Напряжение (В)')

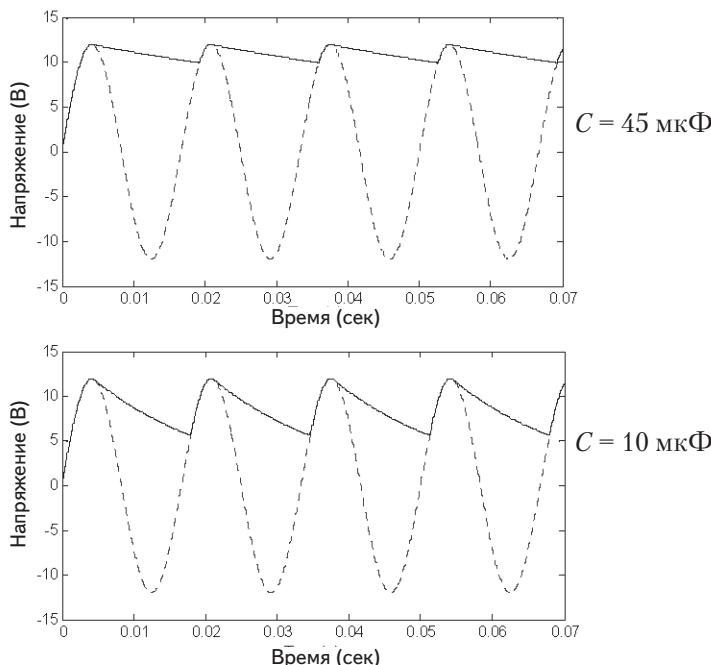
```

Диод в состоянии on.

Проверка $i_R - i_C \leq 0$.
Если истина, присвоение 'off' для state.
Присвоение значение для t_A .

Диод в состоянии off.
Проверка $v_S \geq v_R$.
Если истина, присвоение 'on' для state.

Ниже показаны два графика, созданные программой. Один график показывает результат при $C = 45 \text{ мкФ}$, а другой – при $C = 10 \text{ мкФ}$. Можно заметить, что с более мощным конденсатором напряжение постоянного тока является более гладким (меньшая пульсация в волне).



6.8. Задачи

1. Оцените следующие выражения не используя MATLAB. Проверьте ответы с MATLAB.
- (a) $12 - 4 < 5 \times 3$ (b) $y = 8/4 > 6 \times 3 - 4^2 > -3$
 (c) $y = -3 < (8 - 12) + 2 \times (5 > 18/6 - 4)^2$ (d) $(\sim 5 + \sim 0) \times 6 == 3 + 3 \times \sim 0$
2. Дано: $a = -2, b = 3, c = 5$. Оцените следующие выражения не используя MATLAB. Проверьте ответы с MATLAB.
- (a) $y = a - b > a - c < b$ (b) $y = -4 < a < 0$
 (c) $y = a - c \leq b > a + c$ (d) $y = 3 \times (c + a \sim= a / b - b) == (a + c) \sim= b$
3. Дано: $v = [4 -1 2 3 1 -2 5 0]$ и $u = [5 -1 0 3 -3 2 1 5]$ Оцените следующие выражения не используя MATLAB. Проверьте ответы с MATLAB.
- (a) $\sim \sim u$ (b) $v == \sim u$
 (c) $u == \text{abs}(v)$ (d) $v \geq u + v$
4. Используйте векторы v и u задачи 3. Используйте операторы сравнения для создания вектора w , который составлен из элементов u , которые меньше чем или равны элементов v .
5. Оцените следующие выражения не используя MATLAB. Проверьте ответы с MATLAB.
- (a) $-3 \& 3$ (b) $\sim 5 < 4 \& \sim 0 > -3$
 (c) $-2 \& 2 > 3 | 8 / 3$ (d) $-3 < -1 \sim 0 | 5 < 4 < 3$
6. Используйте циклы для создания 3×5 матрицы, в которой значение каждого элемента есть номер его строки в степени номера столбца, деленное на сумму его номера строки и номера столбца. Например, значение элемента (2,3) есть $2^3 / (2+3)$.
7. Симметричная (5×5) матрица Паскаля выведена на экран справа. Напишите программу MATLAB, которая создает симметричную $n \times n$ матрицу Паскаля. Используйте эту программу для создания 4×4 и 7×7 матриц Паскаля.
- | | | | | |
|---|---|----|----|----|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 2 | 3 | 4 | 5 |
| 1 | 3 | 6 | 10 | 15 |
| 1 | 4 | 10 | 20 | 35 |
| 1 | 5 | 15 | 35 | 70 |
8. Среднемесячные осадки (в дюймах) для Бостона и Сиэтла в течение 2012 представлены векторами ниже (данные американского Национального управления океанических и атмосферных исследований).
- BOS = [2.67 1.00 1.21 3.09 3.43 4.71 3.88 3.08 4.10 2.62 1.01 5.93]
 SEA = [6.83 3.63 7.20 2.68 2.05 2.96 1.04 0.00 0.03 6.71 8.28 6.85]
 где элементы в векторах стоят в порядке месяцев (январь, февраль, и т. д.) Напишите программу в файле сценария, чтобы ответить на следующее:

- (a) Вычислите полное количество осадков в течение года и ежемесячные средние осадки в каждом городе.
- (b) Сколько месяцев в году осадки были выше среднего значения в каждом городе?
- (c) Сколько месяцев и в какие месяцы в Бостоне были осадки ниже, чем в Сиэтле?
- 9.** Напишите программу в файле сценария, которая находит наименьшее четное целое число, которое делится на 13 и на 16 и квадратный корень которого больше чем 120. Используйте цикл в программе. Цикл должен запуститься с 1 и остановиться, когда это число будет найдено. Программа печатает сообщение «Необходимое число есть:» и затем печатает число.
- 10.** Числа Фибоначчи – это числа последовательности, в которой первые два элемента есть 0 и 1, а значение каждого последующего элемента равно сумме двух предыдущих элементов: 0, 1, 1, 2, 3, 5, 8, 13,
Напишите программу MATLAB в файле сценария, которая определяет и выводит на экран первые 20 чисел Фибоначчи.
- 11.** Обратная постоянная Фибоначчи ψ определена бесконечной суммой:
- $$\psi = \sum_{n=1}^{\infty} \frac{1}{F_n},$$
- где F_n – числа Фибоначчи 1, 1, 2, 3, 5, 8, 13, Каждый элемент в этой последовательности есть сумма предыдущих двух. Устанавливая первые два элемента равным 1, для остальных имеем $F_n = F_{n-1} + F_{n-2}$. Напишите программу MATLAB в файле сценария, которая вычисляет ψ для данного числа слагаемых n . Выполните программу для $n = 10, 50$ и 100 .
- 12.** Напишите программу в файле сценария, которая определяет действительные корни квадратного уравнения $ax^2 + bx + c = 0$. Назовите файл `quadroots`. При своей работе файл должен предложить пользователю ввести значения констант a , b и c . Для вычисления корней уравнения программа вычисляет дискриминант D по формуле:

$$D = b^2 - 4ac.$$

Если $D > 0$, то программа выводит сообщения «Уравнение имеет два корня», и корни выводятся на экран в следующей строке.

Если $D = 0$, то программа выводит сообщения «Уравнение имеет один корень», и корень выводится на экран в следующей строке.

Если $D < 0$, то программа выводит сообщения «Уравнение не имеет корней».

Выполните это файл сценария в командном окне три раза, чтобы получить решения следующих трех уравнений:

$$(a) 3x^2 + 6x + 3 = 0; \quad (b) -3x^2 + 4x - 6 = 0; \quad (c) -3x^2 + 7x + 5 = 0.$$

13. Значение π можно оценить выражением:

$$\sqrt{6 \sum_{n=1}^{\infty} \frac{1}{n^2}}.$$

Напишите программу (используя цикл), которая вычисляет это выражение (через сумму первых n слагаемых). Выполните программу для $n = 100$, $n = 10\,000$ и $n = 1\,000\,000$. Сравните результат с π (используйте `format long`).

14. Значение π можно оценить из выражения:

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2 + \sqrt{2}}}{2} \cdot \frac{\sqrt{2 + \sqrt{2 + \sqrt{2}}}}{2} \dots$$

Напишите программу MATLAB в файле сценария, которая определяет π для любого числа членов. Программа просит, чтобы пользователь ввел число членов, а затем вычисляет соответствующее значение π . Выполните программу для 5, 10 и 40 членов. Сравните результат с π (используйте `format long`).

15. Напишите программу, которая генерирует вектор с 20 случайными элементами между -10 и 10 и затем находит сумму положительных элементов.

16. Напишите программу, которая: (a) генерирует вектор с 20 случайными целыми числами между -10 и 10 ; (b) заменяет все отрицательные целые числа на случайные целые числа между -10 и 10 ; (c) повторяет (b) до тех пор, пока все элементы не станут положительными. Программа должна также посчитать, сколько раз будет повторен пункт (b) для того, чтобы все элементы стали положительными. При завершении программа выводит на экран вектор и предложение, которое указывает, сколько итераций потребовалось для генерирования этого вектора.

17. Напишите программу, которая запрашивает пользователя ввести вектор целых чисел произвольной длины. Эта программа находит число элементов, число положительных элементов и число отрицательных элементов, которые делятся на 3. Программа выводит на экран вектор, который вводился и результаты в форме предложения, то есть в виде: «Вектор имеет XX элементов. XX элементов положительны, и XX элементов отрицательны и делятся на 3», где XX – это для соответствующего числа элементов. Выполните программу и когда программа просит пользователя ввести вектор, введите `randi([-20 20], 1, 16)`. Это создает вектор с 16 элементами со случайными целыми числами между -20 и 20 .

18. Дан вектор $x = [4.5 \ 5 \ -16.12 \ 21.8 \ 10.1 \ 10 \ -16.11 \ 5 \ 14 \ -3 \ 3 \ 2]$. Используя условные утверждения и циклы, напишите программу, которая перестраивает элементы x в порядке от наименьшего до самого большого. Не используйте встроенную функцию `sort` MATLAB.

19. Теорема Пифагора утверждает, что $a^2 + b^2 = c^2$. Напишите программу MATLAB, которая находит все комбинации троек a , b и c положительных целых чисел меньше или равных 50, которые удовлетворяют теореме Пифагора. Выведите на экран результаты в таблице с тремя столбцами, в которой каждая строка соответствует одной тройке. Первые три строки таблицы:

3	4	5
5	12	13
6	8	10

20. Парные простые – это пара простых чисел таких, что разность между ними равна 2 (например, 17 и 19). Напишите компьютерную программу, которая находит все парные простые между 10 и 500. Программа выводит на экран результаты в матрице с двумя столбцами, в которой каждая строка – парные простые. Не используйте встроенную функцию MATLAB `isprime`.
21. Изолированное простое – это простое число p такое, что ни $p - 2$, ни $p + 2$ не являются простыми. Например, 47 – изолированное простое, так как 45 и 49 оба не простые. Напишите компьютерную программу, которая находит все изолированные простые между 50 и 100. Не используйте встроенную функцию MATLAB `isprime`.
22. Список 30 оценок экзамена: 31, 70, 92, 5, 47, 88, 81, 73, 51, 76, 80, 90, 55, 23, 43, 98, 36, 87, 22, 61, 19, 69, 26, 82, 89, 99, 71, 59, 49, 64.

Напишите компьютерную программу, которая определяет, сколько оценок между 0 и 19, между 20 и 39, между 40 и 59, между 60 и 79 и между 80 и 100. Результаты выведите на экран в следующей форме:

Оценки от 0 до 19	2 студента
Оценки от 20 до 39	4 студента
Оценки от 40 до 59	6 студентов
и т. д.	

(Подсказка: используйте команду `fprintf` для вывода на экран результатов.)

23. Ряд Тейлора функции $\cos(x)$ есть:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n},$$

где x в радианах. Напишите программу MATLAB, которая определяет $\cos(x)$ используя это разложение в ряд Тейлора. Программа запрашивает пользователя ввести значение угла в градусах. Затем программа использует цикл для добавления членов ряда Тейлора. Если a_n есть n -ый член ряда, то сумма S_n первых n членов есть $S_n = S_{n-1} + a_n$. При каждом проходе вычисляется оценка ошибки E по формуле $E = \left| \frac{S_n - S_{n-1}}{S_{n-1}} \right|$. Остановка сложения членов выполняется если

$E \leq 0.000001$. Программа выводит на экран значение $\cos(x)$. Используйте программу для того, чтобы вычислить:

$$(a) \cos(35^\circ); \quad (b) \cos(125^\circ).$$

Сравните значения с полученными на калькуляторе.

24. Напишите программу MATLAB, которая находит положительное целое число n такое, что сумма всех целых чисел $1 + 2 + 3 + \dots + n$ есть число между 100 и 1000, у которого три цифры одинаковы. В качестве результата программа выводит на экран такое целое число n и соответствующую сумму.

25. Ниже приведены формулы для вычисления тренировочного сердечного ритма (*THR*) для мужчин и женщин:

Для мужчин (формула Карвонена):

$$THR = [(220 - AGE) - RHR] \times INTEN + RHR.$$

Для женщин:

$$THR = [(206 - 0.88 \times AGE) - RHR] \times INTEN + RHR,$$

где AGE – возраст человека, RHR – сердечный ритм в состоянии покоя и $INTEN$ – уровень натренированности (0.55 для низкой, 0.65 для средней и 0.8 для высокой натренированности). Напишите программу, которая определяет THR . Программа просит пользователя ввести пол (мужчина или женщина), возраст (число), сердечный ритм в состоянии покоя (число) и уровень натренированности (низкий, средний или высокий). В результате программа выводит на экран тренировочный сердечный ритм. Используйте программу для определения тренировочного сердечного ритма для следующих двух индивидуумов:

- (a) 21-летний мужчина, сердечный ритм в состоянии покоя 62 и низкий уровень натренированности.
- (b) 19-летняя женщина, сердечный ритм в состоянии покоя 67 и высокий уровень натренированности.

26. Индекс массы тела (*BMI*) является мерой тучности. В стандартных единицах он вычисляется по формуле

$$BMI = 703 \frac{W}{H^2},$$

где W – вес в фунтах, и H – высота в дюймах. Классификация тучности:

BMI	Классификация
ниже 18.5	Недостаточный вес
от 18.5 до 24.9	Нормальный
от 25 до 29.9	Избыточный вес
30 и выше	Тучный

Напишите программу, которая вычисляет *BMI* человека. Программа предлагает человеку ввести его или её вес (фунты) и высоту (в дюймах). Программа выводит на экран результат в предложении, которое читается так: «Ваше значение *BMI* есть XXX, что относит Вас к классу SSSS», где XXX – значение *BMI*, округленное к самой близкой десятой части, и SSSS – соответствующая классификация. Используйте программу для определения тучности следующих двух людей:

- (a) Человек роста 6 футов 2 дюйма и веса 180 фунтов.
- (b) Человек роста 5 футов 1 дюйм и веса 150 фунтов.

27. Напишите программу, которая вычисляет стоимость доставки пакета согласно следующему ценовому расписанию:

Тип службы	Вес		
	0 – 0.5 фунтов	0.5 – 5 фунтов	Более 5 фунтов
Базовый (5–7 дней)	\$0.70 + \$0.06/ унция	\$1.18 + \$0.42 за каждый дополнительный 0.5 фунт (или часть).	\$4.96 + \$0.72 за каждый дополнительный фунт (или часть).
Экспресс (3–3 дня)	\$2.40 + \$0.25/ унция	\$4.40 + \$1.20 за каждый дополнительный 0.5 фунт (или часть).	\$15.20 + \$1.80 за каждый дополнительный фунт (или часть).
Ночной (5–7 дней)	\$12.20 + \$0.80/ унция	\$18.60 + \$4.80 за каждый дополнительный 0.5 фунт (или часть).	\$61.80 + \$6.40 за каждый дополнительный фунт (или часть).

Программа просит пользователя ввести тип службы (Базовый, Экспресс или Ночной) и вес пакета (два числа. Первое для числа фунтов и второе для числа унций.) Программа выводит на экран стоимость отправки. Выполните программу три раза для следующих случаев:

- (a) Базовый 2 фунта 7 унций.
- (b) Экспресс 0 фунтов 7 унций.
- (c) Ночной 5 фунтов 10 унций.

28. Напишите программу, которая определяет сдачу клиенту в кассе самообслуживания супермаркета для покупок до 50 \$. Программа генерирует случайное число между 0.01 и 50.00 и выводит на экран это число как сумму оплаты покупки. Затем программа просит, чтобы пользователь ввел оплату одной купюрой 1 \$, 5 \$, 10 \$, 20 \$, или 50 \$. Если этот платеж меньше чем сумма оплаты, то на экран выводится сообщение об ошибке. Если платеж достаточен, программа вычисляет сдачу как список купюр и/или монет для сдачи, который должен быть составлен из наименьшего количества числа купюр и монет. Например, если сумма покупки составляет 2,33 \$, и купюра 10 \$ вводится как платеж, то

сдача – одна купюра 5 \$, две купюры по 1 \$, две четверти (по 25 центов), десять центов, один никель (5 центов) и два пенса. Выполните программу три раза.

29. Концентрация C_p препарата в теле моделируется формулой:

$$C_p = \frac{D_G}{V_d} \frac{k_a}{k_a - k_e} \left(e^{-k_e t} - e^{-k_a t} \right),$$

где D_G – введенная доза (мг), V_d – объем распределения (Л, литров), k_a – постоянная степени поглощения (час⁻¹), k_e – постоянная степени удаления из организма (час⁻¹) и t – время (в часах) с момента, когда лекарство было применено. Для некоторого препарата определены следующие величины: $D_G = 150$ мг, $V_d = 50$ Л, $k_a = 1.6$ час⁻¹ и $k_e = 0.4$ час⁻¹.

- (a) Введена одна доза при $t = 0$. Вычислите и графически изобразите C_p в зависимости от t в течение 10 часов.
- (b) Первая доза введена при $t = 0$ и впоследствии введены еще четыре дозы с промежутками в 4 часа (то есть, при $t = 4, 8, 12, 16$). Вычислите и графически изобразите C_p в зависимости от t в течение 24 часов.

30. Один из численных методов для вычисления кубического корня из числа $\sqrt[3]{P}$ является итерационным. Процесс запускается с выбора значения x_1 в качестве первой оценки решения. Используя это значение, второе, более точное значение x_2 может быть вычислено по формуле $x_2 = (P/x_1^2 + 2x_1)/3$, которая затем используется для вычисления третьего, еще более точного значения x_3 , и так далее. Общее выражение для вычисления значения x_{i+1} через значение x_i задается формулой $x_{i+1} = (P/x_i^2 + 2x_i)/3$. Напишите программу MATLAB, которая вычисляет кубический корень из числа. В программе используйте $x_1 = P$ для первой оценки решения. Затем, используя в цикле общее выражение, вычислите новые, более точные значения. Цикл останавливается, когда относительная ошибка $E = \left| \frac{x_{i+1} - x_i}{x_i} \right|$ становится меньше чем 0.00001. Используйте программу для вычисления:

$$(a) \sqrt[3]{100}; \quad (b) \sqrt[3]{53701}; \quad (c) \sqrt[3]{19.35}.$$

31. Напишите программу, которая преобразовывает меру давления в единицах Па, пси, атм или торр в эквивалентное количество в других единицах, определенных пользователем. Программа предлагает пользователю ввести величину давления в его текущих единицах и требуемые новые единицы. Результат есть давление в новых единицах. Используйте программу для:

- (a) Преобразования 70 пси в Па.
- (b) Преобразования 120 торр в атм.
- (c) Преобразования 8000 Па в пси.

32. В одномерном случайному блуждании положение x движущейся частицы вычисляется по формуле:

$$x_j = x_{j-1} + s,$$

где s – случайное число. Напишите программу, которая вычисляет число шагов, требуемых частице для достижения границы $x = \pm B$. Используйте встроенную функцию MATLAB `randn(1, 1)` для нахождения s . Выполните программу 100 раз (с использованием цикла) и вычислите среднее число шагов когда $B = 10$.

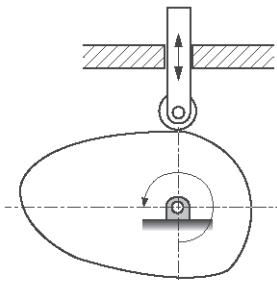
33. Треугольник Серпинского может быть реализован в MATLAB, графически многократно изображая точки, полученные в соответствии с одним из следующих трех правил, которые выбираются в произвольном порядке с равной вероятностью.

$$\begin{array}{ll} \text{Правило 1: } x_{n+1} = 0.5x_n & y_{n+1} = 0.5y_n \\ \text{Правило 2: } x_{n+1} = 0.5x_n + 0.25, & y_{n+1} = 0.5y_n + (\sqrt{3})/4 \\ \text{Правило 3: } x_{n+1} = 0.5x_n + 0.5, & y_{n+1} = 0.5y_n \end{array}$$

Напишите программу, которая вычисляет векторы x и y и затем графически изображает y как функцию от x , как индивидуальные точки (используйте `plot(x, y, '^')`). Начните с $x_1 = 0$ и $y_1 = 0$. Выполните программу четыре раза с 10, 100, 1000 и 10000 итераций.

34. Бегунок – это механическое устройство, которое преобразовывает движение вращения в линейное движение. Форма диска проектируется так, чтобы выполнить указанный профиль смещения. Профиль смещения – это график смещения повторителя как функции угла вращения бегунка. Движение некоторого бегунка задано следующими формулами:

$$\begin{aligned} y &= 6 [20 - 0.5 \sin \theta] / \pi && \text{для } 0 \leq \theta \leq \pi/2, \\ y &= 6 && \text{для } \pi/2 \leq \theta \leq 2\pi/3, \\ y &= 6 - 3 [1 - 0.5 \cos(3(\theta - 2\pi/3))] && \text{для } 2\pi/3 \leq \theta \leq 4\pi/3, \\ y &= 3 && \text{для } 4\pi/3 \leq \theta \leq 3\pi/2, \\ y &= 3 - 1.5 [1 - (\theta - 3\pi/2)/(\pi/4)]^2 && \text{для } 3\pi/2 \leq \theta \leq 7\pi/4, \\ y &= 0.75 - 0.75 [1 - (\theta - 7\pi/4)/(\pi/4)]^2 && \text{для } 7\pi/4 \leq \theta \leq 2\pi. \end{aligned}$$



Напишите программу MATLAB, которая графически изображает профиль смещения для одного полного оборота. Сначала создайте 100-элементный векторов для θ , а затем с использованием цикла и условного оператора вычислите значения y для соответствующих значений θ . Когда y и θ найдены, постройте график y как функции от θ .

35. Полная оценка по курсу определяется из оценок 6 контрольных опросов, 3 экзаменов в течение семестра и итогового экзамена, используя следующую схему:

Контрольные опросы: Они оцениваются по шкале от 0 до 10. Самая низкая оценка опускается, а среднее значение остальных (5-ти) более высоких оценок составляет 30% оценки за курс.

Экзамены в середине семестра и итоговый экзамен: Они оцениваются по шкале от 0 до 100. Если среднее значение баллов экзаменов в течение семестра выше чем оценка на итоговом экзамене, то среднее число баллов промежуточных экзаменов в семестре составляет 50% оценки за курс, а оценка итогового экзамена составляет 20% оценки за курс. Если итоговая оценка выше чем среднее значение баллов промежуточных экзаменов в семестре, то это среднее значение баллов экзаменов в середине семестра составляет 20% оценки за курс, а оценка итогового экзамена составляет 50% оценки за курс.

Напишите компьютерную программу, которая определяет оценку за курс для студента. Программа запрашивает пользователя ввести шесть оценок контрольных опросов (как вектор), три оценки промежуточных экзаменов (как вектор) и оценку итогового экзамена. Тогда программа вычисляет числовую оценку за курс (число между 0 и 100). В заключение программа присваивает буквенную оценку согласно следующему правилу:

- A – когда оценка ≥ 90 ;
- B – когда $80 \leq$ оценка < 90 ;
- C – когда $70 \leq$ оценка < 80 ;
- D – когда $60 \leq$ оценка < 70 ;
- E – когда оценка меньше 60.

Выполните программу для следующих случаев:

- Оценки контрольных опросов: 6, 10, 6, 8, 7, 8. Оценки промежуточных экзаменов: 82, 95, 89. Итоговый экзамен: 81.
- Оценки контрольных опросов: 9, 5, 8, 8, 7, 6. Оценки промежуточных экзаменов: 78, 82, 75. Итоговый экзамен: 81.

36. Дифференциал гандикапа (*HCD*) для раунда в гольф вычисляется по формуле:

$$HCD = \frac{(Очки - Рейтинг)}{Наклон} \times 113.$$

Параметры *Рейтинг* и *Наклон* – это характеристики сложности определенной площадки для гольфа. Гандикап игроков в гольф вычисляется из определенного числа *N* их лучших (нижних) очков гандикапа согласно следующей таблице.

# Раундов	N	# Раундов	N
5–6	1	15–16	6
7–8	2	17	7
9–10	3	18	8
11–12	4	19	9
13–14	5	20	10

Например, если сыграно 13 раундов, то используются только лучшие пять гандикапов. Гандикап не может быть вычислен менее чем из пяти раундов. Если игрались более 20 раундов, то используются только 20 новых результатов. Как только определены N самых низких дифференциалов гандикапов, они усредняются и затем округляются в меньшую сторону до самой близкой десятой части. Результат – гандикап игроков.

Напишите программу, которая вычисляет гандикап участников. Программа просит пользователя ввести записи игроков в гольф в матрице из трех столбцов, где первый столбец – это рейтинг площадки для гольфа, второй – наклон площадки для гольфа и третий – очки игрока. Каждая строка соответствует одному раунду. Программа выводит на экран гандикап участников. Выполните программу для игр со следующими записями.

(a)

Рейтинг	Наклон	Очки
71.6	122	85
72.8	118	87
69.7	103	83
70.3	115	81
70.9	116	79
72.3	117	91
71.6	122	89
70.3	115	83
72.8	118	92
70.9	109	80
73.1	132	94
68.2	115	78
74.2	135	103
71.9	121	84

(b)

Рейтинг	Наклон	Очки
72.2	119	71
71.6	122	73
74.0	139	78
68.2	125	69
70.2	130	74
69.6	109	69
66.6	111	74



ГЛАВА 7.

Определенные пользователем функции и файлы функций

Обычная функция $f(x)$ в математике каждому значению x ставит в соответствие единственное число y . Функция может быть выражена в виде $y = f(x)$, где $f(x)$ – обычно некоторое математическое выражение содержащее x . Значение y (результат) получается, когда в это выражение подставляют значение x (вход). В MATLAB запрограммировано много функций (встроенные функции) и они могут использоваться в математических выражениях просто если ввести их имя вместе с аргументом (см. раздел 1.5); например, $\sin(x)$, $\cos(x)$, \sqrt{x} и $\exp(x)$. Часто в компьютерных программах есть необходимость вычислить значения функций, которые не являются встроенными. Когда выражение функции простое и должно быть вычислено только однажды, оно может быть введено как часть программы. Однако, когда функция должна быть вычислена много раз для разных значений аргументов, удобно создать «определенную пользователем» функцию. Когда такая пользовательская функция создана (и сохранена), она может использоваться точно так же как встроенные функции.

Определяемая пользователем функция – это программа MATLAB, которая создается пользователем, сохраняется как файл функции, а затем используется так же, как встроенная функция. Такая функция может быть как одним простым математическим выражением, так и сложным и содержать ряд вычислений. Во многих случаях это фактически подпрограмма в пределах компьютерной программы. Главная особенность файла функции – это то, что у нее есть ввод и вывод. Это означает, что вычисления в файле функции выполняются используя входные данные, а результаты вычислений передаются из файла функции как результат. Вход и результат могут быть одной или несколькими переменными и могут быть скаляром, вектором или массивом любого размера. Схематично файл функции может быть представлен так:



Очень простой пример пользовательской функции – это функция, которая вычисляет максимальную высоту, которой достигает брошенный вверх шар с определенной скоростью. Для скорости v_0 максимум высоты h_{max} определяется формулой

$$h_{max} = \frac{v_0^2}{2g},$$
 где g – это гравитационное ускорение. В виде функции это может быть записано так: $h_{max}(v_0) = \frac{v_0^2}{2g}$. В этом случае вход функции – это скорость (число) и

результат – максимальная высота (число). Например, в единицах SI ($g = 9.81 \text{ м/с}^2$), если вход составляет 15 м/с, результат будет 11.47 м.



Определяемые пользователем функции могут использоваться и как математические функции, и в качестве подпрограмм в больших программах. Таким образом, большие компьютерные программы могут быть составлены из меньших «стандартных блоков», которые могут быть проверены независимо. Файлы функций подобны подпрограммам в Бейсике и ФОРТРАН'е, процедурам в Паскале и функциям в С.

Основные принципы определяемых пользователем функций изложены в разделах 7.1 до 7.7. В дополнение к пользовательским функциям, которые сохранены в отдельных файлах функциях и вызываются для использования в компьютерной программе, MATLAB предоставляет возможность определять и использовать пользовательскую математическую функцию в пределах компьютерной программы (не в отдельном файле). Это может быть сделано при использовании анонимной функции, которая представлена в разделе 7.8. Есть встроенные и определяемые пользователем функции, которые при их вызове принимают на входе другие функции. Эти функции, которые в MATLAB называются функциями функций, представлены в разделе 7.9. Последние два раздела покрывают подфункции и вложенные функции. Это методы для включения двух или более пользовательских функций в один файл функцию.

7.1. Создание файла функции

Файлы функции создаются и редактируются, как и скрипт-файлы, в окне редактора/отладчика. Это окно открывается из командного окна. На ленте инструментов выбрать **New**, затем выбрать **Function**. Когда открывается окно редактора/отладчика, оно похоже на показанное на рис. 7.1. При открытии редактор уже содержит несколько введенных строк, которые обрисовывают в общих чертах структуру файла функции. Первая строка – это строка определения функции, которая сопровождается комментариями для описания функции. Затем идет программа (пустые строки 4 и 5 в рис. 7.1), и последняя строка – это утверждение `end` конца,

которое является опциональным. Структура файла функции описана подробно в следующем разделе.

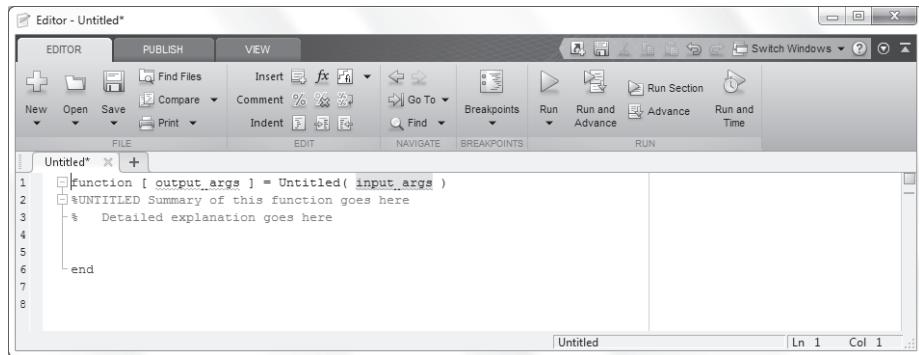


Рис. 7.1. Окно редактора-отладчика

Замечание: Окно редактора/отладчика может также быть открыто (как было описано в главе 1), щелчком по значку **New Script**, или по **New** в ленте инструментов, и последующим выбором **Script** из открывшегося меню. Тогда окно, которое открывается, является пустым, без любых предварительно введенных строк. Вообще, окно редактора/отладчика может использоваться для записи и файла сценария и файла функции.

7.2. Структура файла функции

Структура типичного полного файла функции показана на рис. 7.2. Эта определенная функция вычисляет ежемесячную оплату и полную оплату ссуды. Входные аргументы функции – это количество ссуды, годовая процентная ставка и продолжительность ссуды (число лет). Результат функции – ежемесячная оплата и полная оплата.

Различные части файла функции описаны подробно в следующих разделах.

7.2.1. Страна определения функции

Первая исполнимая строка в файле функции должна быть строкой определения функции. Иначе файл является файлом сценария. Страна определения функции:

- Определяет файл как файл функции.
- Определяет имя функции.
- Определяет число и порядок аргументов входа и выхода.

Вид строки определения функции:

```
function [output arguments] = function_name(input arguments)
```



Слово «*function*» должно быть первым и начинаться со строчной буквы.



Список выходных аргументов в квадратных скобках.



Имя функции



Список входных аргументов в круглых скобках.

```

Editor - C:\MATLAB Book 5th Edition\Chapter7\loan.m
FILE EDIT NAVIGATE BREAKPOINTS RUN
loan.m + [loan.m]
1 function [mpay,tpay] = loan(amount,rate,years)
2 %loan calculates monthly and total payment of loan. ← Стока определения функции.
3 %Input arguments: ← Это H1 строка.
4 %amount=loan amount in $.
5 %rate=annual interest rate in percent.
6 %years=number of years.
7 %Output arguments:
8 -%mpay=monthly payment, tpay=total payment.
9
10 - format bank
11 - ratem=rate*0.01/12;
12 - a=1+ratem;
13 - b=(a^(years*12)-1)/ratem;
14 - mpay=amount*a^(years*12)/(a*b);
15 - tpay=mpay*years*12; ← Присвоение значений выходным переменным.

%loan
Ln 15 Col 20 ...

```

Строка определения функции.
Это H1 строка.
Текст справки.
Тело функции (компьютерная программа).

Рис. 7.2. Структура типичного файла функции

Слово «function» вводится строчными буквами, оно должно быть первым словом на строке определения функции. На экране слово «function» появляется в синем цвете. Имя функции вводится после знака «равно». Имя может быть составлено из символов, цифр и символов подчеркивания (имя не может включать пробелы, кириллица также не допускается). Правила для ввода имени функции – те же самые, что и для имен переменных, описанные в разделе 1.6.2. Желательно избегать имен встроенных функций и имен переменных, уже определенных пользователем или предопределенных MATLAB.

7.2.2. Входные и выходные аргументы

Входные и выходные аргументы используются для передачи данных «в» и «из» функции. Входные аргументы перечисляются в круглых скобках после имени функции. Обычно у функции, есть по крайней мере один входной аргумент, хотя бывают функции, у которой нет входных аргументов. Если имеется более одного аргумента, то входные аргументы разделяются запятыми. Машинный код, который выполняет вычисления в пределах файла функция, записан в терминах входных аргументов и предполагается, что аргументам присвоены числовые значения. Это означает, что математические выражения в файле функция должны быть записаны согласно размерностям аргументов, так как аргументы могут быть скалярами, векторами, или массивами. В примере, показанном на рис. 7.2 есть три входных аргумента (`amount, rate, years`) и в математических выражениях они предполагаются скалярами. Фактические значения входных аргументов присва-

иваются когда функция используется (вызывается). Точно так же, если входные аргументы – векторы или массивы, то математические выражения в теле функции должны быть записаны в соответствии с правилами линейной алгебры или поэлементных вычислений.

Выходные аргументы, которые перечислены в квадратных скобках слева от знака «равно» в строке определения функции, передают результат из файла функции. У файлов функций может быть нуль, один, или несколько выходных аргументов. Если есть более одного, выходные аргументы разделяются запятыми. Если есть только один выходной аргумент, то он может быть введен без скобок. Для корректной работы файла функции выходным аргументам должны быть присвоены значения в компьютерной программе – теле функции. В примере на рис. 7.2 есть два выходных аргумента, `mpay` и `tplay`. Когда у функции нет выходных аргументов, оператор присвоения (знак «равно») в строке определения функции может быть опущен. Функция без выходного аргумента может, например, генерировать график или записать данные в файл.

Также можно передать строки в файл функции. Это делается вводом строки как часть входных переменных (текст в одинарных кавычках). Строки могут использоваться для передачи имен других функций в файл функции.

Обычно все входы в, и выходы от файла функции передаются через аргументы входа и выхода. Кроме того, однако, допустимы все входные и выходные параметры файлов сценариев и могут быть использованы в файлах функции. Это означает, что любая переменная, которой присваивается значение в коде файла функции, будет выведена на экран, если не будет введена точка с запятой в конце команды. Кроме того, команда `input` может использоваться для ввода данных в интерактивном режиме, а команды `disp`, `fprintf` и `plot` могут использоваться для вывода на экран информации, записи в файл, или построения графиков так же, как в скрипте-файле. Ниже представлены примеры строк определения функций с различными комбинациями входных и выходных аргументов.

Строка определения функции	Комментарии
<code>function [mpay,tplay] = loan(amount,rate,years)</code>	Три входных аргумента, два выходных аргумента.
<code>function [A] = RectArea(a,b)</code>	Два входных аргумента, один выходной аргумент.
<code>function A = RectArea(a,b)</code>	То же самое как выше: один выходной аргумент может быть введен без скобок.
<code>function [V, S] = SphereVolArea(r)</code>	Одна входная переменная, две переменные результата.
<code>function trajectory(v,h,g)</code>	Три входных аргумента без выходных аргументов.

7.2.3. Стока H1 и текстовые строки справки

Строка H1 и текстовые строки справки – это строки комментария (строки которые начинаются с знака процента %) после строки определения функции. Они являются необязательными, но часто используются для того, чтобы предоставить информацию о функции. Стока H1 – это первая строка комментария и обычно содержит имя и короткое определение функции. Когда пользователь вводит (в командном окне) `lookfor a_word`, MATLAB ищет `a_word` в строках H1 всех функций, и если совпадение найдено, соответствующая строка H1 выводится на экран.

Текстовые строки справки – это строки комментария, которые следуют за линией H1. Эти строки содержат поясняющую информацию о функции и любые инструкции, связанные с аргументами входа и выхода. Строки комментария, которые введены между строкой определения функции и первой линией некомментария выводятся на экран (это линия H1 и текст справки), когда пользователь вызывает `help function_name` в командном окне. Это справедливо для встроенных функций MATLAB, а также для определяемых пользователем функций. Например, для функции `loan` на рис. 7.2, если ввести `help loan` в командном окне (если, конечно, текущий каталог или путь поиска включают каталог, где сохранен этот файл), то на экране отобразиться следующее:

```
>> help loan
loan calculates monthly and total payment of loan.
Input arguments:
amount=loan amount in $.
rate=annual interest rate in percent.
years=number of years.
Output arguments:
mpay=monthly payment, tpay=total payment.
```

Файл функции может включать дополнительные строки комментария в теле функции. Эти строки игнорируются командой справки `help`.

7.2.4. Тело функции

Тело функции содержит компьютерную программу (код), который фактически выполняет вычисления. Код может использовать все возможности программирования в MATLAB. Это включает вычисления, присвоения, любые встроенные или определяемые пользователем функции, управление потоками (условные утверждения и циклы) как объяснено в главе 6, комментарии, пустые строки и интерактивный ввод и вывод.

7.3. Локальные и глобальные переменные

Все переменные в файле функции локальны (входные и выходные аргументы и любые переменные, которым присвоены значения в пределах файла функции). Это означает, что эти переменные определены и опознаются только в файле функции. Когда выполняется файл функции, MATLAB использует область памяти, отдельную от рабочего пространства (область памяти для командного окна и файлов сценария). В файле функции входным переменным присваиваются значения каждый раз, когда вызывается функция. Эти переменные используются в вычислениях в пределах файла функции. Когда файл функции заканчивает свое выполнение, значения выходных аргументов передаются тем переменным, которые использовались при вызове функции. Все это означает, что у файла функции могут быть переменные с теми же именами, что и у переменных в командном окне или в скриптах-файлах. Файл функции не опознает переменные с теми же самыми именами и с присвоенными значениями вне функции. Присвоение значений таким переменным в файле функции не будет изменять присвоенные им значения в другом месте.

У каждого файла функции есть свои собственные локальные переменные, которые не являются общими с другими функциями или с рабочим пространством командного окна и файлов сценария. Можно, однако, сделать переменную общей (опознаваемой) для нескольких различных файлов функций, и также даже для рабочего пространства. Это делается объявлением этой переменной глобальной командой `global`, которая имеет вид:

```
global variable_name
```

Несколько переменных могут быть объявлены глобальными, перечисляя их через пробелы в команде `global`. Например:

```
global GRAVITY_CONST FrictionCoefficient
```

- Переменная должна быть объявлена глобальной в каждом файле функции, в котором пользователь хочет, чтобы она была опознана. Переменная тогда является общей только для таких файлов.
- Команда `global` должна стоять раньше использования этой переменной. Рекомендуется вводить команду `global` в начале файла.
- Команда `global` должна быть введена в командном окне, или в файле сценария, для того, чтобы она была опознана в рабочем пространстве.
- Переменной может быть присвоено, или повторно присвоено, значение в любом из расположений, где она объявлена общей.
- Для глобальных переменных рекомендуется использование длинных описательных имен (или всех прописных букв), чтобы отличить их от обычных переменных.

7.4. Сохранение файла функции

Файл функции должен быть сохранен до его использования. Это делается так же, как и со скриптом-файлом, выбором **Save as ...** из меню **File**, затем выбором местоположения (многие студенты сохраняют на флеш-карте) и вводом имени файла. Строго рекомендуется сохранять этот файл под именем, которое идентично имени функции в строке определения функции. Тогда функция вызывается (используется) при использовании имени функции. (Если файл функции сохранен под другим именем, то имя, под которым он сохранен, должно использоваться при вызове функции.) Файлы функции сохраняются с расширением .m. Примеры:

Строка определения функции	Имя файла
function [mpay,tpay] = loan(amount,rate,years)	loan.m
function [A] = RectArea(a,b)	RectArea.m
function [V, S] = SphereVolArea(r)	SphereVolArea.m
function trajectory(v,h,g)	trajectory.m

7.5. Использование пользовательских функций

Определяемая пользователем функция используется таким же образом, как истроенная функция. Эта функция может быть вызвана из командного окна, из скрипта-файла, или из другой функции. Чтобы использовать файл функции, папка, где он сохранен, должна или быть текущей папкой или быть в путях поиска (см. разделы 1.8.3 и 1.8.4).

Функция может использоваться для присвоения ее результат переменной (или переменным), как часть математического выражения, как аргумент в другой функции, или просто вводом ее имени в командном окне или в скрипте-файле. Во всех случаях пользователь должен знать точно, каковы аргументы входа и выхода. Входной аргумент может быть числом, вычислимым выражением, или переменной, у которой есть присвоенное значение. Аргументы присваиваются в соответствии с их позицией в списке входных и выходных аргументов в строке определения функции.

Два из способов использования функции проиллюстрированы ниже с пользовательской функцией *loan* из рис. 7.2, которая вычисляет ежемесячные и полные платежи (два выходных аргумента) ссуды. Входные аргументы – это величина ссуды, годовая процентная ставка и срок (число лет) ссуды. В первой иллюстрации функция *loan* используется с числами в качестве входных аргументов:

```
>> [month total]=loan(25000, 7.5, 4)
month =
    600.72.
total =
    28834.47
```

Первый аргумент — величина ссуды, второй —процентная ставка и третий — число лет.

На второй иллюстрации функция `loan` используется с двумя заданными переменными и числом в качестве входных аргументов:

```
>> a=70000; b=6.5;
>> [x y]=loan(a,b,30)
x =
    440.06
y =
    158423.02
```

Определение переменных `a` и `b`.
Использование `a`, `b` и числа 30
для входных аргументов и переменных `x`
(ежемесячная плата) и `y` (полная плата) для
выходных аргументов.

7.6. Примеры простых пользовательских функций

Пример задачи 7.1. Пользовательская функция для математической функции

Написать файл функции (с именем `chp7one`) для функции $f(x) = \frac{x^4 \sqrt{3x+5}}{(x^2 + 1)^2}$.

Входной аргумент функции — x , а выходной — $f(x)$. Написать эту функцию так, что x может быть вектором. Используйте функцию для вычисления:

- (a) для $x = 6$.
- (b) для $x = 1, 3, 5, 7, 9$ и 11 .

Решение

Файл функции для этой функции $f(x)$ есть:

```
function y=chp7one(x)
y=(x.^4.*sqrt(3*x+5))./(x.^2+1).^2;
```

Строка определения функции.
Присвоение выходного значения.

Отметим, что математическое выражение в файле функции записано для поэлементных вычислений. Таким образом, если x будет вектором, то y также будет вектором. Сохраняем функцию и затем модифицируем путь поиска для включения каталога, где это файл был сохранен. Ниже показано, как функция используется в командном окне.

(a) Вычисление функции для $x = 6$ может быть сделано, вводом строки `chp7one(6)` в командном окне, или присваивая значение функции новой переменной:

```
>> chp7one(6)
ans =
    4.5401
>> F=chp7one(6)
F =
    4.5401
```

(b) Чтобы вычислить функцию для нескольких значений x , создается вектор со значениями x и затем он используется как аргумент функции.

```
>> x=1:2:11
x =
    1      3      5      7      9      11
>> chp7one(x)
ans =
    0.7071    3.0307    4.1347    4.8971    5.5197    6.0638
```

Другой способ заключается в том, чтобы ввести вектор x непосредственно в аргументе функции.

```
>> H=chp7one([1:2:11])
H =
    0.7071    3.0307    4.1347    4.8971    5.5197    6.0638
```

Пример задачи 7.2. Преобразование единиц температуры

Написать определяемую пользователем функцию (с именем `FtoC`), которая преобразовывает температуру в градусах по Фаренгейту в температуру в градусах по Цельсию. Использовать функцию для решения следующей задачи. Изменением длины объекта ΔL вследствие изменения температуры ΔT определяется формулой: $\Delta L = \alpha \Delta T$, где α – это коэффициент теплового расширения. Определить изменение площади прямоугольного (4.5 м на 2.25 м) листа алюминия ($\alpha = 23 \times 10^{-6} 1/^\circ\text{C}$), температура меняется от 40 °F до 92 °F.

<pre>function C=FtoC(F) %FtoC преобразовывает градусы по Фаренгейту в градусы по Цельсию C=5*(F-32)./9;</pre>	Строка определения функции. Присвоение выходного значения.
---	---

Скрипт-файл (названный `Chapter7Example2`), который вычисляет изменение площади листа вследствие изменения температуры:

```
a1=4.5; b1=2.25; T1=40; T2=92; alpha=23e-6;
deltaT=FtоС(T2)-FtоС(T1);           Использование функции FtоС для вычисления
                                         перепада температур в градусах по Цельсию.
a2=a1+alpha*a1*deltaT;                Вычисление новой длины.
b2=b1+alpha*b1*deltaT;                Вычисление новой ширины.
AreaChange=a2*b2-a1*b1;                Вычисление изменения площади.
fprintf ('Изменение площади равно %6.5f квадратных метров.', 
AreaChange)
```

Выполнение скрипта в командном окне дает следующее решение:

```
>> Chapter7Example2
Изменение площади равно 0.01346 квадратных метров.
```

7.7. Сравнение файлов функций и скрипт-файлов

Студенты, которые изучают MATLAB впервые, иногда испытывают затруднения, связанные с точным пониманием различий между скриптами и файлами функции, поскольку для многих задач, требующих для их решения использования MATLAB, может использоваться любой тип файла. Сходство и различия между скриптами и файлами функции представлены ниже.

- И скрипты и файлы функции сохраняются с расширением .m (именно поэтому их иногда вызывают М-файлами).
- Первая исполняемая строка в файле функции (должна быть) – строка определения функции.
- Переменные в файле функции локальны. Переменные в скрипте опознаются в командном окне.
- Скрипты-файлы могут использовать переменные, которые были определены в рабочем пространстве.
- Скрипты-файлы содержат последовательность команд MATLAB (утверждений).
- Файлы функций могут принять данные через входные аргументы и могут возвратить данные через выходные аргументы.
- При сохранении файла функции имя файла должно совпадать с именем функции.
- Определяемая пользователем функция используется таким же образом, что и встроенная функция. Она может использоваться (вызываться) в командном окне, в скрипте или в другой функции.

7.8. Анонимные функции

Определяемые пользователем функции, записанные в файлах функции, могут использоваться как для простых математических функций, так и для больших и сложных математических функций, которые требуют обширного программирования, а также как подпрограммы в больших компьютерных программах. В случаях, когда значение относительно простого математического выражения должно быть вычислено много раз в пределах программы, MATLAB предоставляет возможность использования анонимных функций. Анонимная функция – это пользовательская функция, которая определена и записана в пределах машинного кода (не в отдельном файле функции) и затем используется в коде. Анонимные функции могут быть определены в любой части MATLAB (в командном окне, в скриптах и в регулярных определяемых пользователем функциях).

Анонимная функция – это простая (короткая) определяемая пользователем функция, которая определена без создания отдельный файл функции (m-файла). Анонимные функции могут быть созданы в командном окне, в пределах скрипта-файла, или в регулярной определяемой пользователем функции.

Анонимная функция создается вводом следующей команды:



Простой пример – `cube = @(x) x^3`, который вычисляет куб входного аргумента.

- Эта команда создает анонимную функцию и присваивает дескриптор функции к имени переменной слева от `= sign`. (Дескрипторы функций обеспечивают средства для того, чтобы они использовали функцию и передали ее к другим функциям; см. раздел 7.9.1.)
- `expr` состоит из простого допустимого математического выражения MATLAB.
- Математическое выражение может иметь одну или несколько независимых переменных. Независимая переменная(ые) вводятся в `(arglist)`. Несколько независимых переменных разделяются запятыми. Пример анонимной функции, у которой есть две независимые переменные: `circle = @(x,y) 16*x^2+9*y^2`
- Математическое выражение может включать любые встроенные или определяемые пользователем функции.
- Это выражение должно быть записано в соответствии с размерностями аргументов (поэлементных или вычислений по правилам линейной алгебры).
- Выражение может включать переменные, которые уже определены, когда записывается анонимная функция. Например, если три переменные

a , b и c определены (им присвоены числовые значения), то они могут использоваться в выражении анонимной функции $\text{parabola} = @(\text{x}) \text{a} * \text{x}^2 + \text{b} * \text{x} + \text{c}$.

Важное замечание: Когда анонимная функция определяется, MATLAB берет значения предопределенных переменных. Это означает, что, если новые значения впоследствии присвоены этим переменным, то анонимная функция не изменяется. Анонимная функция должна быть переопределена для новых значений переменных, которые будут использоваться в выражении.

Использование анонимной функции:

- Когда анонимная функция определена, она может использоваться, вводом ее имени и значений аргумента (или аргументов) в круглых скобках (см. примеры ниже).
- Анонимные функции могут также использоваться в качестве аргументов в других функциях (см. раздел 7.9.1).

Пример анонимной функции с одной независимой переменной

Функция $f(x) = \frac{e^{x^2}}{\sqrt{x^2 + 5}}$ может быть определена (в командном окне) как анонимная функция для x как скаляра следующим образом:

```
>> FA = @(x) exp(x^2) / sqrt(x^2+5)
FA =
    @(x) exp(x^2) / sqrt(x^2+5)
```

Если точка с запятой не введена в конце, MATLAB отвечает, выводит на экран функцию. Функция может тогда использоваться для различных значений x , как показано ниже.

```
>> FA(2)
ans =
    18.1994
>> z = FA(3)
z =
    2.1656e+003
```

Если x , возможно, будет массивом и вычисления должны производиться для каждого элемента, тогда функция должна быть изменена для поэлементно вычислений.

```
>> FA = @(x) exp(x.^2)./sqrt(x.^2+5)
FA =
@(x)exp(x.^2)./sqrt(x.^2+5)
>> FA([1 0.5 2])           Использование вектора во входном аргументе.
ans =
    1.1097      0.5604      18.1994
```

Пример анонимной функции с несколькими независимыми переменными

Функция $f(x,y) = 2x^2 - 4xy + y^2$ может быть определена как анонимная функция следующим образом:

```
>> HA = @(x,y) 2*x^2 - 4*x*y + y^2
HA =
@(x,y) 2*x^2-4*x*y+y^2
```

Теперь анонимная функция может использоваться для различных значений x и y . Например, ввод $HA(2,3)$ дает:

```
>> HA(2,3)
ans =
-7
```

Другой пример использования анонимной функции с несколькими аргументами показан в примере типовой задаче 6.3.

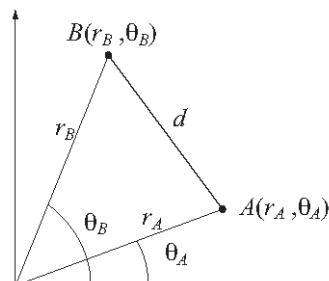
Пример задачи 7.3. Расстояние между точками в полярных координатах

Написать анонимную функцию которая вычисляет расстояние между двумя точками на плоскости, когда положения точек заданы в полярных координатах. Использовать анонимную функцию для вычисления расстояния между точкой $A(2, \pi/6)$ и точкой $B(5, 3\pi/4)$.

Решение

Расстояние между двумя точками в полярных координатах может быть вычислено по теореме косинусов:

$$d = \sqrt{r_A^2 + r_B^2 - 2r_A r_B \cos(\theta_A - \theta_B)}.$$



Эта формула для расстояния вводится как анонимная функция с четырьмя входными аргументами $(r_A, \theta_A, r_B, \theta_B)$. Затем она используется для вычисления расстояния между точками A и B .

```
>> d = @(rA,thetaA,rB,thetB) sqrt(rA^2+rB^2-2*rA*rB*cos(thetaB -thetaA))
d =
@(rA,thetaA,rB,thetB)sqrt(rA^2+rB^2-2*rA*rB*cos(thetaB-thetaA))
>> DistAtob = d(2,pi/6,5,3*pi/4)
DistAtob =
5.8461
```

Список входных аргументов
Аргументы в том порядке, как они определены в функции.

7.9. Функции от функций

Есть много ситуаций, когда функция (Функция A) работает с (использованием) другой функции (Функция B). Это означает, что, когда Функция A выполняется, ей нужно предоставить Функцию B. Функция, которая принимает другую функцию, называется в MATLAB функцией от функции. Например, в MATLAB есть встроенная функция, называемая `fzero` (Функция A), которая находит нуль математической функции $f(x)$ (Функция B), то есть, значение x где $f(x) = 0$. Программа в функции `fzero` написана так, что, она может найти нуль любой функции $f(x)$. Когда вызывается функция `fzero`, то ей передают определенную функцию для решения $f(x)$ и тогда `fzero` находит ее нуль. (Функция `fzero` описана подробно в главе 9.)

Функция от функции, которая принимает другую функцию (импортируемую функцию), включает в свои входные аргументы имя, которое представляет импортируемую функцию. Имя импортируемой функции используется для операций в программе (коде) функции от функции. Когда используется (вызывается) функция от функции, определенная импортируемая функция перечисляется в ее входных аргументах. Таким образом различные функции могут быть импортированы (переданы) в функцию от функции. Есть два способа для того, чтобы перечислить имя импортированной функции в списке параметров функции от функции. Первый заключается в использовании дескриптора функции (раздел 7.9.1), а другой – вводом имени передаваемой функции как строкового выражения (раздел 7.9.2). Конкретный метод, который используется, связан с тем, каким образом записаны операции в функции от функции (это объяснено более подробно в следующих двух разделах). Использование дескрипторов функций легче и более эффективно, и поэтому является предпочтительным методом.

7.9.1. Использование дескрипторов функций для передачи функции в функцию от функции

Дескрипторы функции используются для того, чтобы передать (импортировать) пользовательские функции, встроенные функции и анонимные функции в функции от функций, которые могут их принять. В этом разделе сначала объясняется, что такое дескриптор функции, затем показывается, как записать определяемую пользователем функцию от функции, которая принимает дескрипторы функций и, наконец, показывается, как использовать дескрипторы функций для передачи функции в функцию от функции.

Дескриптор функции

Дескриптор функции – это некоторое значение MATLAB, которое ассоциируется с функцией. Это тип данных MATLAB и он может быть передан как аргумент в другую функцию. После передачи дескриптор функции обеспечивает средства для вызова (использования) соответствующей функции. Дескрипторы функции могут использоваться с любым видом функции MATLAB, включая встроенные функции, определяемые пользователем функции (записанный в файлах функции) и анонимные функции.

- Для встроенных и определяемых пользователем функций дескриптор функции задается, вводом символа @ перед именем функции. Например, @cos – дескриптор встроенной функции cos, а @FtоС – дескриптор пользовательской функции FtоС, которая создавалась в примере типовой задаче 7.2.
- Дескриптор функции может также быть присвоен переменной с некоторым именем. Например, cosHandle=@cos присваивает дескриптор @cos переменной cosHandle. Затем имя cosHandle может использоваться для передачи дескриптора.
- Для анонимных функций (см. раздел 7.8.1) их имя уже – дескриптор функции.

Запись функции от функции, которая принимает дескриптор функции как входной аргумент

Как уже упоминалось, входные аргументы функции от функции (которая принимает другую функцию) включают имя (фиктивное, временное имя функции), которое представляет импортированную функцию. Эта фиктивная функция (включая список входных аргументов, приложенных в круглых скобках), используется для операций программы в функции от функции.

- Функция, которая фактически импортируется, должна быть в таком виде, чтобы не противоречить тому, как фиктивная функция используется в программе. Это означает, что она должна иметь то же самое число и тип аргументов входа и выхода.

Ниже идет пример пользовательской функции от функции, названной funplot, который создает график функции (любой функции $f(x)$, которая будет импортирована) между точками и $x = a$ и $x = b$. Входные аргументы есть (Fun, a, b), где Fun – это фиктивное имя, которое представляет импортируемую функцию и a и b – конечные точки области. Функция funplot имеет также есть числовой результат xout, который является 3×2 матрицей со значениями x и $f(x)$ в трех точках $x = a$ и $x = (a+b)/2$ и $x = b$. Отметим, что в этой программе у фиктивной функциональной Fun есть один входной аргумент (x) и один выходной аргумент y, которые оба являются векторами.

```

function xyout=funplot(Fun,a,b) Имя для функции, которая будет передаваться.
% funplot создает график функции Fun, которую передают
% в funplot при вызове в области [a, b].
% Входные аргументы:
% Fun: Дескриптор функции, для которой строится график.
% a: Начальная точка области.
% b: Конечная точка области.
% Выходные аргументы:
% xyout: это значения x и y в x = a, x = (a+b)/2, и x = b
% представленные в 3-на-2 матрице.

x=linspace(a,b,100);
y=Fun(x); Использование импортированной функции вычисления f(x) в 100 точках.
xyout(1,1)=a; xyout(2,1)=(a+b)/2; xyout(3,1)=b;
xyout(1,2)=y(1);
xyout(2,2)=Fun((a+b)/2); Использование импортированной функции
xyout(3,2)=y(100); для вычисления f(x) в средней точке.
plot(x,y)
xlabel('x'), ylabel('y')

```

В качестве примера, передадим функцию $f(x) = e^{-0.17x} x^3 - 2x^2 + 0.8x - 3$ на промежутке $[0.5, 4]$ в пользовательскую функцию `funplot`. Это можно сделать двумя способами: первый – с записью пользовательской функции $f(x)$, а второй с записью $f(x)$ как анонимная функция.

Передача пользовательской функции в функцию от функции

Сначала пишется пользовательская функция для $f(x)$. Функция, названная `Fdemo`, вычисляет $f(x)$ для данного значения x и пишется с использованием по-элементных операций.

```

function y=Fdemo(x)
y=exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3;

```

Затем, функция `Fdemo` передается в определяемую пользователем функцию от функции `funplot`, которая вызывается в командном окне. Отметим, что вместо входного аргумента `Fun` в пользовательской функции `funplot` вводится дескриптор пользовательской функции `Fdemo` (дескриптор – это `@Fdemo`).

```

>> ydemo=funplot(@Fdemo,0.5,4)
ydemo =
    0.5000      -2.9852
    2.2500      -3.5548
    4.0000       0.6235

```

Вводится дескриптор пользовательской функции Fdemo.

В дополнение к отображению числового результата при выполнении команды создается график, показанный на рис. 7.3, выведенный на экран в окне графиков Figure.

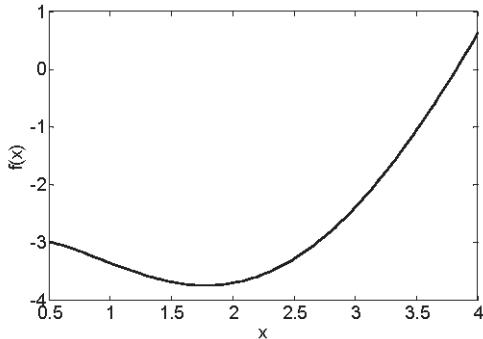


Рис. 7.3. График функции $f(x) = e^{-0.17x}x^3 - 2x^2 + 0.8x - 3$

Передача анонимной функции в функцию от функции

Чтобы использовать анонимную функцию $f(x) = e^{-0.17x}x^3 - 2x^2 + 0.8x - 3$, эта функция сначала должна быть записана как анонимная функция, и затем передана в определяемую пользователем функцию funplot. Ниже показано, как оба из этих шагов выполнить в командном окне. Отметим, что в пользовательской функции funplot имя анонимной функции FdemoAnonyis, вводится вместо входного аргумента Fun без знака @ (так как это имя уже дескриптор анонимной функции).

```
>> FdemoAnony=@(x) exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3
FdemoAnony =
    Создание анонимной функции для f(x).
    @(x) exp(-0.17*x).*x.^3-2*x.^2+0.8*x-3
>> ydemo=funplot(FdemoAnony,0.5,4)
ydemo =
    Вводится имя анонимной функции (FdemoAnony).
    0.5000      -2.9852
    2.2500      -3.5548
    4.0000       0.6235
```

В дополнение к отображению числового результата при выполнении команды создается график, показанный на рис. 7.3, выведенный на экран в окне графиков Figure.

7.9.2. Использование имени функции для передачи функции в функцию от функции

Второй метод для передачи функции в функцию от функции заключается во вводе имени импортируемой функции в виде строки во входном аргументе функции от функции. Этот метод, который использовался перед введением дескрипторов функции, может использоваться для импорта определяемых пользователем функций. Как отмечалось ранее, дескрипторы функции использовать легче и более

эффективно, поэтому это предпочтительный метод. Импорт пользовательских функций с использованием их имени изложен в данной книге для читателей, которым нужно понять программы, написанные до выхода MATLAB 7. Новые программы должны использовать дескрипторы функции.

Когда пользовательская функция импортируется с использованием ее имени, значение импортируемой функции в функции от функции должно быть вычислено при помощи команды `feval`. Это отличается от случая, когда используется дескриптор функции, т. е. есть различие в способе, которым записан код в функции от функции, который зависит от того, как передают импортируемую функцию.

Команда *feval*

Команда `fEval` (сокр. от «function evaluate» – оценивание функции) оценивает (вычисляет) значение функции для данного значения (или значений) аргумента функции (или аргументов). Формат команды:

variable = feval('имя функции', значение аргумента)

Значение, которое определяется командой `feval`, может быть присвоено переменной, или если команда введена без присвоения, MATLAB выводит на экран `ans =` и значение функции.

- Имя функции вводится в виде строки (`string`).
 - Функция может быть встроенной или пользовательской.
 - Если входных аргументов больше одного, они разделяются запятыми.
 - Если выходных аргументов больше одного, то переменные слева от оператора присвоения вводятся в квадратных скобках и разделяются запятыми.

Ниже приведены два примера использования команды `feval` со встроенными функциями.

```
>> feval('sqrt',64)
ans =
    8
>> x=feval('sin',pi/6)
x =
    0.5000
```

Следующий пример показывает использование команды `feval` с определяемой пользователем функцией `loan`, которая создавалась ранее в главе (рис. 7.2). У этой функции есть три входных аргумента и два выходных аргумента.

Запись функции от функции, которая принимает другую функцию вводом ее имени как входного аргумента

Как уже упоминалось, когда пользовательская функция импортируется с использованием ее имени, значение этой функции в функции от функции должно вычисляться командой `feval`. Это демонстрируется в следующей пользовательской функции от функции `funplots`. Эта функция та же, что и функция `funplot` из раздела 7.9.1, за исключением того, что используется команда `feval` для вычислений с импортированной функцией.

```
function xyout=funplots(Fun,a,b)      Имя для функции, которая предаётся.
% funplots создает график функции Fun, которую передают
% в funplots при вызове в области [a, b].
% Входные аргументы:
% Fun: Функция для которой строится график.
% Ее имя вводится как строковое (string) выражение.
% a: Начальная точка области.
% b: Конечная точка области.
% Выходные аргументы:
% xyout: это значения x и y в x = a, x = (a+b)/2, и x = b
% представленные в 3-на-2 матрице.

x=linspace(a,b,100);
y=feval(Fun,x);  Использование импортированной функции для вычисления f(x)
xyout(1,1)=a; xyout(2,1)=(a+b)/2; xyout(3,1)=b;           в 100 точках.
xyout(1,2)=y(1);
xyout(2,2)=feval(Fun,(a+b)/2);          Использование импортированной
xyout(3,2)=y(100);                      функции для вычисления f(x)
plot(x,y)                                в средней точке.
xlabel('x'), ylabel('y')
```

Передача пользовательской функции в другую функцию в виде строки

Ниже показано, как передать пользовательскую функцию в функцию от функции, вводя имя импортируемой функции как строку во входном аргументе. Эта функция $f(x) = e^{-0.17x} x^3 - 2x^2 + 0.8x - 3$ из раздела 7.9.1, созданная как пользовательская под именем `Fdemo`, передается в определяемую пользователем функцию `funplots`. Отметим, что имя `Fdemo` введено в виде строки на место входного аргумента `Fun` в определяемой пользователем функции `funplots`.

```
>> ydemoS=funplots('Fdemo',0.5,4)
ydemo =                               Имя импортируемой функции введенное как строка.
    0.5000          -2.9852
    2.2500          -3.5548
    4.0000          0.6235
```



В дополнение к выводу числовых результатов в командном окне, на экран в окне графиков Figure выводится график, показанный в рис. 7.3.

7.10. Подфункции

Файл функции может содержать более одной определяемой пользователем функции. Эти функции вводятся одна за другой. Каждая функция начинается со строки определения функции. Первая функция называется первичной (главной) функцией, а остальные функции вызываются подфункциями. Подфункции могут быть введены в любом порядке. Имя файла функции должно соответствовать имени первичной функции. Каждая из функций в файле может вызывать любую из других функций в этом файле. Из внешних функций, или программ (скриптов-файлов) можно вызвать только первичную функцию. У каждой из функций в файле есть свое собственное рабочее пространство, это означает, что в каждой функции переменные локальны. Другими словами, главная функция и подфункции не могут получить доступ к переменным друг друга (если эти переменные не объявлены как глобальные).

Подфункции могут помочь в организации написания определяемых пользователем функций. Программа первичной (главной) функции может быть разделена на меньшие задачи, каждая из которых может быть выполнена в виде подфункции. Это демонстрируется в примере типовой задачи 7.4.

Пример задачи 7.4. Среднее и стандартное отклонение

Написать пользовательскую функцию, которая вычисляет среднее число и стандартное отклонение списка чисел. Использовать эту функцию для вычисления среднего числа и стандартного отклонения следующего списка оценок:

80 75 91 60 79 89 65 80 95 50 81

Решение

Среднее число x_{ave} (среднее значение) данного множества из n чисел x_1, x_2, \dots, x_n определяется формулой:

$$x_{ave} = (x_1 + x_2 + \dots + x_n)/n.$$

Стандартное отклонение задается формулой:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - x_{ave})^2}{n-1}}.$$

Пользовательская функция, названная `stat`, написана для решения этой задачи. Чтобы продемонстрировать использование подфункций, файл функции включает `stat` в качестве первичной функции и две подфункции под названием `AVG` и `StandDiv`. Функция `AVG` вычисляет x_{ave} , а функция `StandDiv` вычисляет σ . Обе

подфункции вызываются первичной функцией. Следующий листинг сохранен как один файл функции с именем `stat`.

```
function [me SD] = stat(v)                                Первичная (главная) функция.
n=length(v);
me=AVG(v,n);
SD=StandDiv(v,me,n);

function av=AVG(x,num)                                  Подфункция.
av=sum(x)/num;

function Sdiv=StandDiv(x,xAve,num)                      Подфункция.
xdif=x-xAve;
xdif2=xdif.^2;
Sdiv= sqrt(sum(xdif2)/(num-1));
```

Теперь эта пользователем функция `stat` используется в командном окне для вычисления среднего числа и стандартного отклонения оценок:

```
>> Grades=[80 75 91 60 79 89 65 80 95 50 81];
>> [AveGrade StanDeviation] = stat(Grades)
AveGrade =
    76.8182
StanDeviation =
    13.6661
```

7.11. Вложенные функции

Вложенная функция – это определяемая пользователем функция, которая записана в другой определяемой пользователем функции. Часть кода, который соответствует вложенной функции, запускается со строки определения функции и заканчивается утверждением конца `end`. Утверждение `end` должно быть также введено и в конце функции, которая содержит вложенную функцию. (Обычно пользовательская функция не требует завершающегося утверждения `end`. Однако, это утверждение `end` требуется, если функция содержит одну или более вложенных функций.) Вложенные функции могут также содержать вложенные функции. Очевидно, что наличие многих уровней вложенных функций может ввести в заблуждение. В этом разделе мы рассматриваем только два уровня вложенных функций.

Одна вложенная функция

Формат пользовательской функции `a` (первичной функции), которая содержит одну вложенную функцию `b`:

```
function y=A(a1,a2)
.....
function z=B(b1,b2)
.....
end
.....
end
```

- Отметим два утверждения `end` в конце функций `В` и `А`.
- Вложенная функция `В` может получить доступ к рабочему пространству первичной функции `А`, и первичная функция `А` может получить доступ к рабочему пространству функции `В`. Это означает, что переменная, определенная в первичной функции `А`, может быть считана и переопределена во вложенной функции `В` и наоборот.
- Функция `А` может вызвать функцию `В`, а функция `В` может вызвать функцию `А`.

Два (или более) вложенные функции на одном уровне

Формат пользовательской функции `А` (первичной функции), которая содержит две вложенных функции `В` и `С` на одном уровне:

```
function y=A(a1,a2)
.....
function z=B(b1,b2)
.....
end
.....
function w=C(c1,c2)
.....
end
.....
end
```

- Три функции могут получить доступ к рабочему пространству друг друга.
- Три функции могут вызвать друг друга.

Как пример, следующая пользовательская функция (названная `statNest`) с двумя вложенными функциями на одном уровне, решает пример типовой задачи 7.4. Отметим, что вложенные функции используют переменные (`n` и `me`), которые определены в первичной функции.

```
function [me SD]=statNest (v)
n=length(v);
me=AVG(v);
```

Первичная (главная) функция.



```
function av=AVG(x)
av=sum(x)/n;
end
```

Вложенная функция.

```
function Sdiv=StandDiv(x)
xdif=x-me;
xdif2=xdif.^2;
Sdiv= sqrt(sum(xdif2)/(n-1));
end
```

Вложенная функция.

```
SD=StandDiv(v);
end
```

Использование пользовательской функции statNest в командном окне для вычисления среднего и стандартного отклонения оценок дает следующее:

```
>> Grades=[80 75 91 60 79 89 65 80 95 50 81];
>> [AveGrade StanDeviation] = statNest(Grades)
AveGrade =
    76.8182
StanDeviation =
    13.6661
```

Два уровня вложенных функций

Два уровня вложенных функций создаются в том случае, когда вложенные функции пишутся внутри вложенных функций. Ниже показан пример формата пользовательской функции с четырьмя вложенными функциями в двух уровнях.

```
function y=A(a1,a2)                                (Первичная функция A.)
.....
function z=B(b1,b2)                                (B вложенная функция в A.)
.....
function w=C(c1,c2)                                (C вложенная функция в B.)
.....
end
end
function u=D(d1,d2)                                (D вложенная функция в A.)
.....
function h=E(e1,e2)                                (E вложенная функция в D.)
.....
end
end
.....
end
```

Следующие правила применяются к вложенным функциям:

- Вложенная функция может быть вызвана из уровня выше ее на один. (В предыдущем примере функция A может вызвать B или D, но не C или E.)
- Вложенная функция может быть вызвана из вложенной функции на том же самом уровне в пределах первичной функции. (В предыдущем примере функция B может вызвать D, и D может вызвать B.)
- Вложенная функция может быть вызвана из вложенной функции на любом более низком уровне.
- Переменная, определенная в первичной функции, опознается и может быть переопределена вложенной функцией на любом уровне в пределах первичной функции.
- Переменная, определенная во вложенной функции, опознается и может быть переопределена любой из функций, которые содержат эту вложенную функцию.

7.12. Примеры приложений MATLAB

Пример задачи 7.5. Экспоненциальный рост и затухание

Модель для экспоненциального роста или затухания некоторой величины задается формулой

$$A(t) = A_0 e^{kt},$$

где $A(t)$ и A_0 есть величина в моменты времени t и 0, соответственно, k является постоянной, характерной для определенного приложения. Написать пользовательскую функцию, которая использует эту модель для предсказания количества $A(t)$ в момент времени t , зная A_0 и количество $A(t_1)$ в некоторый другой момент времени t_1 . Для имени функции и аргументов использовать At = expGD(A0, At1, t1, t), где выходной аргумент At соответствует $A(t)$, а для входных аргументов, использовать A0, At1, t1, t, соответствующих A_0 , $A(t_1)$, t_1 и t .

Использовать это файл функции в командном окне для следующих двух случаев:

- (a) Количество населения Мексики в 1980 году было 67 миллионов, а в 1986 – 79 миллионов. Оценить количество населения в 2000.
- (b) Период полураспада радиоактивного материала составляет 5.8 лет. Сколько останется через 30 лет от 7-граммовой выборки?

Решение

Для использования модели экспоненциального роста сначала должно быть определено значение постоянной k через данные A_0 , $A(t_1)$ и t_1 :

$$k = \frac{1}{t_1} \ln \frac{A(t_1)}{A_0}.$$

Когда k найден, можно использовать модель для оценки населения в любой момент времени.

Пользовательская функция, которая решает эту задачу:

```
function At=expGD(A0,At1,t1,t)
    % expGD вычисляет экспоненциальный рост и затухание
    % Входные аргументы:
    % A0: Количество в момент времени 0.
    % At1: Количество в момент времени t1.
    % t1: время t1.
    % t: время t.
    % Выходной аргумент:
    % B: Количество в момент времени t.
    k=log(At1/A0)/t1; % Определение k.
    At=A0*exp(k*t); % Определение A(t). (Присвоение значения
                      % выходному аргументу.)
```

После сохранения функции, она используется в командном окне для решения указанных выше двух задач. Для случая (a) $A_0 = 67$, $A(t_1) = 79$, $t_1 = 6$ и $t = 20$:

```
>> expGD(67, 79, 6, 20)
ans =
    116.03
```

Оценка населения в 2000 году.

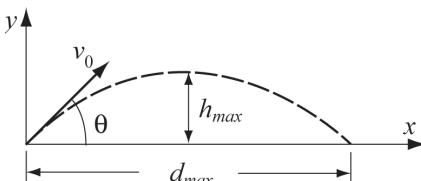
Для случая (b) $A_0 = 7$, $A(t_1) = 3.5$ (поскольку соответствует полураспаду, времени, в течение которого материал распадается до половины (3.5) его начального количества), $t_1 = 5.8$ и $t = 30$:

```
>> expGD(7, 3.5, 5.8, 30)
ans =
    0.19
```

Количество материала через 30 лет.

Пример задачи 7.6. Движение снаряда

Создать файл функции, которая вычисляет траекторию снаряда. Входные аргументы – начальная скорость и угол, под которым запущен снаряд. Результаты функции – максимальная высота и расстояние. Кроме того, функция генерирует график траектории. Использовать эту функцию для вычисления траектории снаряда, который выпущен со скоростью 230 м/с под углом 39°.



Решение

Движение снаряда можно проанализировать, рассмотрев горизонтальные и вертикальные составляющие его движения. Начальная скорость v_0 может быть разложена на горизонтальную и вертикальную составляющие

$$v_{0x} = v_0 \cos(\theta) \quad \text{и} \quad v_{0y} = v_0 \sin(\theta).$$

В вертикальном направлении скорость и положение снаряда задаются формулами:

$$v_y = v_{0y} - gt \quad \text{и} \quad y = v_{0y}t - gt^2 / 2.$$

Время, необходимое снаряду для достижения самой высокой точки ($v_y = 0$) и соответствующая высота задаются формулами:

$$t_{h\max} = \frac{v_{0y}}{g} \quad \text{и} \quad h_{\max} = \frac{v_{0y}^2}{2g}.$$

Полное время полета равно удвоенному времени, необходимому снаряду для достижения самой высокой точки $t_{tot} = 2t_{h\max}$. В горизонтальном направлении скорость является постоянной и положение снаряда задается формулой:

$$x = v_{0x} t.$$

В системе MATLAB имя функции и аргументы вводятся как `[hmax, dmax] = trajectory(v0, theta)`. Файл функции:

```
function [hmax, dmax]=trajectory(v0,theta)
% Страна определения функции.
% Вычисляет траекторию, максимальную высоту и расстояние
% снаряда и создает график траектории.
% Входные аргументы:
% v0: начальная скорость в (м/с).
% theta: угол в градусах.
% Выходные аргументы:
% hmax: максимальная высота в (м).
% dmax: максимальная дальность в (м).
% Функция также создает график траектории.
g=9.81;
v0x=v0*cos(theta*pi/180);
v0y=v0*sin(theta*pi/180);
thmax=v0y/g;
hmax=v0y^2/(2*g);
ttot=2*thmax;
dmax=v0x*ttot;
% Создание графика траектории
tplot=linspace(0,ttot,200); % Создание вектора времени, 200 элементов.
x=v0x*tplot; % Вычисление координат x и y снаряда каждый момент времени.
y=v0y*tplot-0.5*g*tplot.^2; % Отметим поэлементное умножение.
plot(x,y)
xlabel('Расстояние (м)')
ylabel('Высота (м)')
title('ТРАЕКТОРИЯ СНАРЯДА')
```

После сохранения функция она используется в командном окне для снаряда, который выпущен со скоростью 230 м/с и под углом 39°.

```
>> [h d]=trajectory(230, 39)
h =
    1.0678e+003
d =
    5.2746e+003
```

Кроме того, создается следующий рисунок в окне графиков Figure:



7.13. Задачи

1. Напишите пользовательскую функцию MATLAB для следующей математической функции:

$$y(x) = (-0.2x^3 + 7x^2)e^{-0.17x}.$$

Входной аргумент функции – x , а результат – y . Напишите функцию так, чтобы x мог быть вектором (используя поэлементные операции).

- (a) Используйте функцию для вычисления $y(-1.5)$ и $y(5)$.
- (b) Используйте функцию для создания графика функции $y(x)$ для $-2 \leq x \leq 6$.

2. Напишите пользовательскую функцию MATLAB для следующей математической функции:

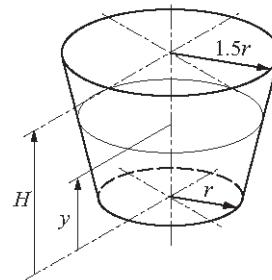
$$r(\theta) = 4\cos(4\sin \theta).$$

Входной аргумент функции – θ , а результат – r . Напишите функцию так, чтобы θ мог быть вектором (используя поэлементные операции).

- (a) Используйте эту функцию для вычисления $r(\pi/6)$ и $r(5\pi/6)$.
- (b) Используйте функцию для создания графика (в полярных координатах) функции $r(\theta)$ для $0 \leq \theta \leq 2\pi$.

- 3.** Потребление топлива самолета измеряется в гал/миль (галлонов на милю) или в Л/км (литров на километр). Напишите определяемую пользователем функцию MATLAB, которая преобразовывает удельное потребление топлива из гал/миль в Л/км. Для имени функции и аргументов, используйте следующие обозначения `Lkm=LkmToGalmi(gmi)`. Входной аргумент `gmi` является потреблением в `gal/mi`, а выходной аргумент, `Lkm` – потребление в `Л/км`. Используйте эту функцию в командном окне для:
- Определения потребления топлива Boeing 747 в Л/км, если его потребление топлива приблизительно равно 5 гал/миль.
 - Определения потребления топлива Concorde в Л/км, если его потребление топлива приблизительно равно 5.8 гал/миль.
- 4.** Таблицы свойств материалов приводят плотность в единицах $\text{кг}/\text{м}^3$, если используется международная система единиц (SI), и удельный вес в единицах фунт/дюйм³, когда используется американская общепринятая система единиц. Напишите пользовательскую функцию MATLAB, которая преобразует плотность в удельный вес. Для имени функции и аргументов, используйте следующее: `use[sw] = DenToSw(den)`. Входной аргумент `den` – это плотность материала в $\text{кг}/\text{м}^3$, а выходной аргумент `sw` является удельным весом в фунт/дюйм³. Используйте эту функцию в командном окне для:
- Определения удельного веса стали, плотность которой составляет $7860 \text{ кг}/\text{м}^3$.
 - Определения удельного веса титана, плотность которого составляет $4730 \text{ кг}/\text{м}^3$.
- 5.** Напишите пользовательскую функцию MATLAB, которая преобразует скорость, данную в узлах (один узел составляет одну морскую милю в час) в скорость в футах в секунду. Для имени функции и аргументов, используйте `fps = ktsTOfps(kts)`. Входной аргумент `kts` является скоростью в узлах, а выходной аргумент `fps` – скорость в фут/сек. Используйте функцию для преобразования 400 `kts` в единицы фут/сек.
- 6.** Площадь поверхности тела (*BSA*) (в м^2) человека (используемая для определения дозы лекарства) может быть вычислена по формуле (формула Дю Буа (*Du Bois*)):
- $$BSA = 0.007184 W^{0.425} H^{0.75},$$
- в которой W – масса тела в кг и H – рост в см.
- Напишите пользовательскую функцию MATLAB, которая вычисляет площадь поверхности тела. Для имени функции и аргументов используйте `BSA = BodySurfA(w, h)`. Входные аргументы w и h – это масса и рост, соответственно. Выходной аргумент `BSA` есть значение `BSA`. Используйте функцию для вычисления площади поверхности тела:
- человека веса 95 кг и роста 1.87 м.
 - человека веса 61 кг и роста 1.58 м.

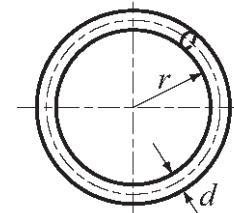
7. Топливный бак, показанный на рисунке справа, имеет форму усеченного двумя параллельными плоскостями конуса с параметрами $r = 20$ дюймов и $H = 2r$. Напишите пользовательскую функцию (для имени функции и аргументов, используйте $v = \text{VolFuel}(y)$), которая дает объем топлива в баке (в галлонах) как функцию высоты y (измеренную от нижней части). Используйте эту функцию для построения графика объема как функции от y для $0 \leq y \leq 40$ дюйм.



8. Площадь поверхности S кольца в форме тора с внутренним радиусом r и диаметром d задается формулой:

$$S = \pi^2 (2r + d) d.$$

Кольцо должно быть покрыто металлом тонким слоем. Вес покрытия W может быть вычислен приблизительно как $W = \gamma St$, где γ – определенный вес материала покрытия и t – его толщина. Напишите анонимную функцию, которая вычисляет вес покрытия. Функция должна иметь четыре входных аргумента, r , d , t и γ . Используйте анонимную функцию для вычисления веса золотого покрытия ($\gamma = 0.696$ фунт/дюйм 3) кольца с параметрами: $r = 0.35$ дюйм, $d = 0.12$ дюйм, и $t = 0.002$ дюйм.



9. Температура (коэффициент) T_{wc} резкости погоды (windchill temperature) – это воспринимаемая температура воздуха с учетом ветра, температура, которую ощущают открытые участки кожи за счет потока воздуха. Для температур ниже 50 °F и скорости ветра выше чем 3 мили в час, она вычисляется по формуле:

$$T_{wc} = C_1 + C_2 T_A + C_3 V^{0.16} + C_4 T_A V^{0.16},$$

где T_A – температура воздуха в градусах по Фаренгейту (°F), V – скорость ветра в милях в час (мвч), $C_1 = 35.74$, $C_2 = 0.6215$, $C_3 = -35.75$ и $C_4 = 0.4275$.

Напишите пользовательскую функцию для вычисления T_{wc} для данных T_A и V . Для имени функции и аргументов используйте $\text{Twc}=\text{WindChill}(t, v)$. Входные аргументы: t – температура воздуха в °F и v – скорость ветра в мвч, соответственно. Выходной аргумент – Twc , температура/коэффициент резкости погоды в °F (округленный к самому близкому целому числу). Используйте эту функцию для определения температуры/коэффициента резкости погоды для следующих условий:

- (a) $T = 35$ °F, $V = 26$ мвч.
 (b) $T = 10$ °F, $V = 50$ мвч.

10. Напишите пользовательскую функцию, которая вычисляет средний балл (GPA) в масштабе от 0 до 4, где $A = 4$, $A- = 3.7$, $B+ = 3.3$, $B = 3$, $B- = 2.7$, $C+ = 2.3$,

$C = 2$, $C- = 1.7$, $D+ = 1.3$, $D = 1$ и $E = 0$. Для имени функции и аргументов используйте $\text{av} = \text{GPA}(g, h)$. Входной аргумент g – это вектор, элементы которого есть числовые значения оценок по предметам. Входной аргумент h – это вектор с соответствующими зачетными часами по предметам. Выходной аргумент av является вычисленным GPA ($\text{GPA} = (\text{Сумма оценок, умноженных на зачетные часы}) / (\text{Сумма зачетных часов})$). Используйте функцию, чтобы вычислить GPA для студента со следующими результатами:

Оценки	$A-$	B	$B+$	C	E	A	$D+$	A
Зачетные часы	4	3	3	2	3	4	3	3

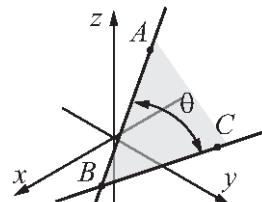
11. Факториал $n!$ положительного (целого) числа определен формулой

$$n! = n \cdot (n - 1) \cdot (n - 2) \cdots 3 \cdot 2 \cdot 1, \text{ где } 0! = 1.$$

Напишите пользовательскую функцию, которая вычисляет факториал $n!$ из числа. Для имени функции и аргументов используйте $y=\text{fact}(x)$, где входной аргумент x является числом, факториал которого должен быть вычислен, и выходной аргумент y – значение $x!$. Функция выводит на экран сообщение об ошибке, если при вызове функции введено отрицательное число или нецелое. Не используйте встроенную функцию факториал MATLAB. Используйте `fact` для вычисления:

$$(a) 9!; \quad (b) 8.5!; \quad (c) 0!; \quad (d) -5!.$$

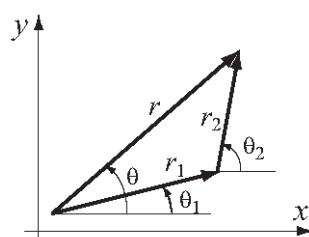
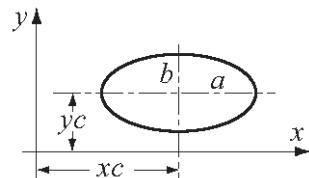
12. Напишите пользовательскую функцию MATLAB, которая определяет угол, который образует пересечением двух линий. Для имени функции и аргументов используйте $\text{th}=\text{anglines}(A, B, C)$. Входные аргументы функции – векторы с координатами точек A , B и C , как показано на рисунке, которые могут быть двух- или трехмерными. Результат th – это угол в градусах. Используйте функцию `anglines` для определения угла для следующих случаев:
- $A(-5, -1, 6)$, $B(2.5, 1.5, -3.5)$, $C(-2.3, 8, 1)$.
 - $A(-5.5, 0)$, $B(3.5, -6.5)$, $C(0, 7)$.



13. Напишите пользовательскую функцию MATLAB, которая определяет единичный вектор в направлении линии, которая соединяет две точки (A и B) в пространстве. Для имени функции и аргументов используйте $\text{n} = \text{unitvec}(A, B)$. Ввод функции – два вектора A и B , каждый с декартовыми координатами соответствующей точки. Результат n – вектор с компонентами единичного вектора в направлении от A к B . Если у точек A и B две координаты каждый (они находятся в плоскости $x-y$), то n – вектор с двумя элементами. Если у точек A и B три координаты каждый (точки в пространстве), то n – вектор с тремя элементами. Используйте функцию для определения следующих единичных векторов:

- (a) В направлении от точки $(1.2, 3.5)$ к точке $(12, 15)$.
 (b) В направлении от точки $(-10, -4, 2.5)$ к точке $(-13, 6, -5)$.
- 14.** Напишите пользовательскую функцию MATLAB, которая определяет векторное произведение двух векторов. Для имени функции и аргументов используйте $w=crosspro(u, v)$. Входные аргументы функции – два вектора, которые могут быть двух- или трехмерными. Выходные аргумент (вектор) w является результатом их произведения. Используйте функцию $crosspro$ для нахождения векторного произведения:
- (a) Векторов $a = 3\mathbf{i} + 11\mathbf{j}$ и $b = 14\mathbf{i} - 7.3\mathbf{j}$.
 (b) Векторов $c = -6\mathbf{i} + 14.2\mathbf{j} + 3\mathbf{k}$ и $d = 6.3\mathbf{i} - 8\mathbf{j} - 5.6\mathbf{k}$.
- 15.** Площадь треугольника ABC может быть вычислена по формуле:
- $$S = \frac{1}{2} |AB \times AC|,$$
- где AB – вектор от точки A до точки B и AC – вектор от точки A до точки C . Напишите пользовательскую функцию MATLAB, которая определяет площадь треугольника, заданного координатами ее вершин. Для имени функции и аргументов используйте $[Area] = TriArea(A, B, C)$. Входные аргументы A , B и C являются векторами, каждый из координат соответствующей вершины. Напишите код $TriArea$ так, что в нем есть две подфункции – одна определяет векторы AB и AC , а другая выполняет векторное произведение. (При возможности, используйте пользовательские функции задачи 14). Функция должна работать для треугольников в xy -плоскости (каждая вершина определяется двумя координатами), или для треугольников в пространстве (каждая вершина определена тремя координатами). Используйте функцию для определения площади треугольников со следующими вершинами:
- (a) $A = (1, 2)$, $A = (10, 3)$, $C = (6, 11)$.
 (b) $A = (-1.5, -4.2, -3)$, $A = (-5.1, 5.3, 2)$, $C = (12.1, 0, -0.5)$.
- 16.** Напишите пользовательскую функцию MATLAB, которая определяет длину описанной окружности треугольника, когда даны координаты его вершин. Для имени функции и аргументов используйте $[cr] = cirtriangle(A, B, C)$. Входные аргументы A , B , C являются векторами с координатами вершин, а результат cr является длиной описанной окружности. Функция должна работать для треугольников в xy -плоскости (каждая вершина определяется двумя координатами), или для треугольников в пространстве (каждая вершина определена тремя координатами). Используйте функцию для определения длины описанной окружности треугольника со следующими вершинами:
- (a) $A = (1, 2)$, $A = (10, 3)$, $C = (6, 11)$.
 (b) $A = (-1.5, -4.2, -3)$, $A = (-5.1, 5.3, 2)$, $C = (12.1, 0, -0.5)$.

- 17.** Напишите пользовательскую функцию, которая графически изображает окружность, заданную координатами центра и точки на окружности. Для имени функции и аргументов используйте `circlePC(c, p)`. Входной аргумент c является двухэлементным вектором с x и y координатами центра, а входной аргумент p – вектор с x и y координатами точки на окружности. Эта функция не имеет выходных аргументов. Используйте функцию для графического изображения следующих двух окружностей (обе в одном окне Figure):
- Центр в точке $x = 7.2$, $y = -2.9$, точки на окружности: $x = -1.8$, $y = 0.5$.
 - Центр в точке $x = -0.9$, $y = -3.3$, точки на окружности: $x = 0$, $y = 10$.
- 18.** Напишите пользовательскую функцию MATLAB, которая преобразует целые десятичные числа в двоичную форму. Имя функции `b=Bina(d)`, где входной аргумент d является целым числом, которое будет преобразовано, а выходной аргумент b – вектор из единиц и нулей, который представляет число в двоичной форме. Наибольшее число, которое может быть преобразовано этой функцией, должно быть двоичным числом с 16 единицами. Если вводится большее число в качестве d , функция должна вывести на экран сообщение об ошибке. Используйте функцию для преобразования следующих чисел:
- 100;
 - 1002;
 - 52,601;
 - 200,090.
- 19.** Напишите пользовательскую функцию, которая графически изображает треугольник и описанную окружность, используя координаты его вершин. Для имени функции и аргументов используйте `TriCirc(A, b, c)`. Входные аргументы – это векторы с x и y координатами вершин, соответственно. Функция не имеет выходных аргументов. Используйте функцию с точками $(1.5, 3)$, $(9, 10.5)$ и $(6, -3.8)$.
- 20.** Напишите пользовательскую функцию, которая графически изображает эллипс с осями, которые параллельны осям x и y , по координатам его центра и длинам осей. Для имени функции и аргументов используйте `ellipseplot(xc, yc, a, b)`. Входные аргументы xc и yc – это координаты центра, а a и b – половины длин горизонтальной и вертикальной осей эллипса (см. рисунок), соответственно. Функция не имеет выходных аргументов. Используйте функцию, чтобы графически изобразить следующие эллипсы:
- $xc = 3.5$, $yc = 2.0$, $a = 8.5$, $b = 3$.
 - $xc = -5$, $yc = 1.5$, $a = 4$, $b = 8$.
- 21.** В полярных координатах двумерный вектор задается его радиусом и углом (r, θ) . Напишите пользовательскую функцию MATLAB, которая



складывает два вектора, которые заданы в полярных координатах. Для имени функции и аргументов используйте $[r \text{ th}] = \text{AddVecPol}(r_1, \theta_1, r_2, \theta_2)$, где входные аргументы (r_1, θ_1) и (r_2, θ_2) , а выходные аргументы – радиус и угол результата. Используйте функцию для выполнения следующие сложения:

$$(a) (r_1, \theta_1) = (5, 23^\circ), (r_2, \theta_2) = (12, 40^\circ); \quad (b) (r_1, \theta_1) = (6, 80^\circ), (r_2, \theta_2) = (15, 125^\circ).$$

- 22.** Напишите пользовательскую функцию, которая находит все простые числа между двумя числами m и n . Имя функции $\text{pr}=\text{prime}(m, n)$, где входные аргументы m и n – положительные целые числа, а выходной аргумент pr , является вектором из простых чисел. Если вводится $m > n$ при вызове функции, то на экран выводится сообщение об ошибке «Значение n должно быть больше, чем значение m ». Если при вызове функции вводится отрицательное число или число, которое не является целым, то на экран выводится сообщение об ошибке «Входной аргумент должен быть положительным целым числом». Не используйте встроенные функции MATLAB `primes` и `isprime`. Используйте функцию для случаев:

$$(a) \text{prime}(12, 80); \quad (b) \text{prime}(21, 63.5); \\ (c) \text{prime}(100, 200); \quad (d) \text{prime}(90, 50).$$

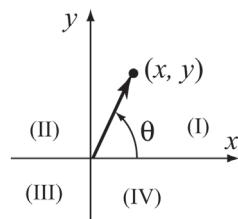
- 23.** Среднее геометрическое GM набора из n положительные числа определяется формулой:

$$GM = (x_1 \cdot x_2 \cdot \dots \cdot x_n)^{1/n}.$$

Напишите пользовательскую функцию, которая вычисляет геометрическое среднее значение набора чисел. Для имени функции и аргументов используйте $GM=\text{Geomean}(x)$, где входной аргумент x является вектором чисел (любой длины), и GM – выходной аргумент – их геометрическое среднее значение. Геометрическое среднее полезно для вычисления среднего числа показателей. Следующая таблица дает рост инфляции в Соединенных Штатах с 1978 до 1987 (инфляция на 7.6% обозначена в таблице как 1.076). Используйте пользовательскую функцию `Geomean` для вычисления средней инфляции в течение десятилетнего периода.

Год	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987
Рост инфляции	1.076	1.113	1.135	1.103	1.062	1.032	1.043	1.036	1.019	1.036

- 24.** Напишите пользовательскую функцию, которая определяет полярные координаты точки из ее декартовых координат в двумерной плоскости. Для имени функции и аргументов используйте $[\text{th } \text{rad}] = \text{CartToPolar}(x, y)$. Входные аргументы – координаты x и y точки, а выходные аргументы – угол θ и радиальное расстояние rad до точки. Угол θ находится в градусах и измеряется от положительного направления оси x в противоположную полуплоскость.



- жительной оси X так, что его значение – положительное число в квадрантах I и II и отрицательное число в квадранте III и IV. Используйте функцию для определения полярных координат точек $(14, 9)$, $(-11, -20)$, $(-15, 4)$, и $(13.5, -23.5)$.
- 25.** Напишите пользовательскую функцию, которая определяет моду (значение из множества, которое случается чаще всего) ряда данных. Для имени функции и аргументов используйте `m=mostfrq(x)`. Входной аргумент – вектор значений x любой длины, а результат m – вектор с двумя элементами, в котором первый элемент есть значение из x , которое случается чаще всего (мода), а второй элемент – соответствующая частота. В случае равных частот у нескольких значений в качестве моды взять наименьшее. Если получается два или более значений для моды, выдается сообщение: «Более одного значения для моды». Не используйте встроенную функцию `mode` MATLAB. Проверьте функцию три раза. Для ввода создайте вектор с 20 элементами, используя следующую команду: `x=randi (10,1,20)`.
- 26.** Напишите пользовательскую функцию, которая сортирует элементы вектора от самого большого до наименьшего. Для имени функции и аргументов используйте `y=downsort(x)`. Ввод функции – вектор x любой длины, а результат y является вектором, в котором элементы x расположены в порядке по убыванию. Не используйте встроенные функции MATLAB типа `sort`, `max` или `min`. Протестируйте Вашу функцию на векторе с 14 числами (целыми) распределенные в произвольном порядке между -30 и 30 . Используйте функцию MATLAB `randi` для создания начальной вектора.
- 27.** Напишите пользовательскую функцию, которая сортирует элементы матрицы. Для имени функции и аргументов используйте `B = matrixsort(A)`, где A – любая матрица размера $m \times n$, и B – матрица того же размера перестроенная в порядке убывания элементов строка за строкой с самым большим элементом $(1,1)$ и наименьшим элементом (m,n) . При наличии, используйте пользовательскую функцию `downsort` из задачи 26 как подфункцию в функции `matrixsort`. Проверьте свою функцию на матрице 4×7 с элементами (целыми) распределенными в произвольном порядке между -30 и 30 . Используйте функцию MATLAB `randi` для создания начальной матрицы.
- 28.** Напишите пользовательскую функцию, которая находит самый большой элемент матрицы. Для имени функции и аргументов используйте `[Em, rc] = matrixmax(A)`, где A – матрица любого размера. Выходной аргумент Em является значением самого большого элемента, а rc – вектор с двумя элементами, индексами самого большого элемента (номер строки и номер столбца). Если получается два, или более элементов с максимальным значением, выходной аргумент rc есть матрица с двумя столбцами, где строки перечисляют адреса элементов. Проверьте функцию три раза. Для ввода создайте матрицу 4×6 следующей командой: `x=randi([-20 100], 4, 6)`.

29. Напишите пользовательскую функцию MATLAB, которая вычисляет определитель матрицы 3×3 по формуле:

$$\det = A_{11} \begin{vmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{vmatrix} - A_{12} \begin{vmatrix} A_{21} & A_{23} \\ A_{31} & A_{33} \end{vmatrix} + A_{13} \begin{vmatrix} A_{21} & A_{22} \\ A_{31} & A_{32} \end{vmatrix}.$$

Для имени функции и аргументов используйте `d3 = det3by3(A)`, где входной аргумент `A` является матрицей 3×3 , а выходной аргумент `d3` – значение определителя. Напишите код `det3by3` так, чтобы у него была подфункция, которая вычисляет определитель. Запишите код `det3by3such`, что у этого есть подфункция, которая вычисляет определитель 2×2 . Используйте `det3by3` для вычисления определителей:

$$(a) \begin{bmatrix} 1 & 3 & 2 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{bmatrix}; \quad (b) \begin{bmatrix} -2.5 & 7 & 1 \\ 5 & -3 & -2.6 \\ 4 & 2 & -1 \end{bmatrix}.$$

30. Состояние напряжения в направлении в точке двумерного нагруженного материала в системе xy -координат, определяется тремя компонентами напряжения σ_{xx} , σ_{yy} и τ_{xy} . Компоненты напряжения в точке в системе $x'y'$ -координат, вычисляются по формулам преобразования напряжения:

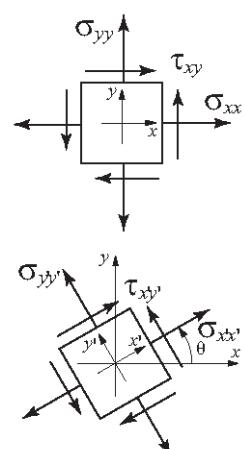
$$\sigma_{x'x'} = \frac{\sigma_{xx} + \sigma_{yy}}{2} + \frac{\sigma_{xx} - \sigma_{yy}}{2} \cos 2\theta + \tau_{xy} \sin 2\theta$$

$$\tau_{x'y'} = -\frac{\sigma_{xx} - \sigma_{yy}}{2} \sin 2\theta + \tau_{xy} \cos 2\theta$$

$$\sigma_{y'y'} = \sigma_{xx} + \sigma_{yy} - \sigma_{x'x'}$$

где θ – угол, показанный на рисунке. Напишите пользовательскую функцию MATLAB, которая определяет напряжения $\sigma_{x'x'}$, $\sigma_{y'y'}$ и $\tau_{x'y'}$ по заданным напряжениям σ_{xx} , σ_{yy} и τ_{xy} и углу θ . Для имени функции и аргументов используйте `[Stran]=StressTrans(S, th)`. Входной аргумент `S` является вектором со значениями трех компонентов напряжения σ_{xx} , σ_{yy} и τ_{xy} и входной аргумент `th` является скаляром со значением угла θ . Выходной аргумент `Stran` является вектором со значениями трех компонентов напряжения $\sigma_{x'x'}$, $\sigma_{y'y'}$ и $\tau_{x'y'}$. Используйте функцию для определения преобразования напряжений для следующих случаев:

- (a) $\sigma_{xx} = 160$ мПа, $\sigma_{yy} = -40$ мПа, $\tau_{xy} = 60$ мПа, $\theta = 20^\circ$.
 (b) $\sigma_{xx} = 18$ кси, $\sigma_{yy} = 10$ кси, $\tau_{xy} = -8$ кси, $\theta = 65^\circ$.



- 31.** Точка росы T_d и относительная влажность RH могут быть вычислены (приблизительно) из температуры T сухого термометра и температуры T_w влажного термометра (<http://www.wikipedia.org>) по формулам:

$$e_s = 6.112 \exp\left(\frac{17.67 T}{T + 243.5}\right), \quad e_w = 6.112 \exp\left(\frac{17.67 T_w}{T_w + 243.5}\right),$$

$$e = e_w - p_{sta} (T - T_w) 0.00066 (1 + 0.00115 T_w),$$

$$RH = 100 \frac{e}{e_s}, \quad T_d = \frac{243.5 \ln(e / 6.112)}{17.67 - \ln(e / 6.112)},$$

где температуры находятся в градусах Цельсия ($^{\circ}\text{C}$), RH находится в %, и p_{sta} является атмосферным давлением в миллибарах.

Напишите пользовательскую функцию MATLAB, которая вычисляет точку росы и относительную влажность для данных температуры сухого и влажного термометра в градусах по Фаренгейту ($^{\circ}\text{F}$) и атмосферного давления в дюймах ртутного столба (дюйм рт. ст.). Для имени функции и аргументов используйте `[Td, RH] = DewptRhum(T, Tw, BP)`, где входные аргументы T, Tw, BP – температуры сухого влажного термометра в $^{\circ}\text{F}$ и BP – атмосферное давление в дюйм рт. ст., соответственно. Выходные аргументы Td, RH – точка росы в $^{\circ}\text{F}$ и относительная влажность в %. Значения выходных аргументов должны быть округлены к самой близкой десятой части. Используйте анонимную функцию или подфункции в `DewptRhum` для преобразования единиц. Используйте пользовательскую функцию `DewptRhum` для вычисления точки росы и относительной влажности в следующих случаях:

- (a) $T = 78 ^{\circ}\text{F}$, $T_w = 66 ^{\circ}\text{F}$, $p_{sta} = 29.09$ дюйм рт. ст.
- (b) $T = 97 ^{\circ}\text{F}$, $T_w = 88 ^{\circ}\text{F}$, $p_{sta} = 30.12$ дюйм рт. ст.

- 32.** В лотерее игрок должен выбрать несколько чисел из списка. Напишите пользовательскую функцию, которая генерирует список n случайных целых чисел, которые равномерно распределены между числами a и b . Все выбранные числа в списке должны отличаться. Для имени функции и аргументов, используйте `x = lotto(a, b, n)`, где входные аргументы – числа a, b и n , соответственно. Выходной аргумент x является вектором с выбранными числами.

- (a) Используйте эту функцию для создания списка семи чисел от 1 до 59.
- (b) Используйте эту функцию для создания списка восьми чисел от 50 до 65.
- (c) Используйте эту функцию для создания списка девяти чисел от -25 до -2.

- 33.** Ряд Тейлора функции $\cos(x)$ в окрестности нуля задается формулой:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n},$$

где x в радианах. Напишите пользовательскую функцию, которая определяет $\cos(x)$ используя это разложение в ряд Тейлора. Для имени функции и аргументов используйте $y=\cosTay(x)$, где входной аргумент x является углом в градусах, а выходной аргумент y – значение $\cos(x)$. В пользовательской функции используйте цикл для добавления членов ряда Тейлора. Если a_n есть энний член ряда, то сумма S_n первых n членов есть $S_n = S_{n-1} + a_n$. При каждом проходе вычисляется оценка ошибки E по формуле $E = \left| \frac{S_n - S_{n-1}}{S_{n-1}} \right|$. Остановка сложения членов выполняется, если $E \leq 0.000001$. Поскольку $\cos(\theta) = \cos(\theta \pm 360n)$, напишите пользовательскую функцию так, что если угол будет больше 360° , или меньше -360° , то ряд Тейлора будет вычисляться с наименьшим числом членов (используя значение для аргумента x , которое является самым близким к 0).

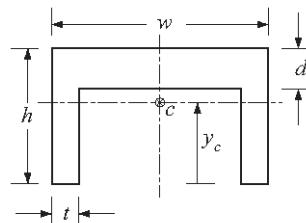
Используйте \cosTay для вычисления:

- (a) $\cos(67^\circ)$; (b) $\cos(200^\circ)$; (c) $\cos(-80^\circ)$;
 (d) $\cos(794^\circ)$; (e) $\cos(20\,000^\circ)$; (f) $\cos(-738^\circ)$.

Сравнитесь значения, вычисленные функцией \cosTay со значениями, полученными при использовании встроенной функции \cosd MATLAB.

- 34.** Напишите пользовательскую функцию, которая определяет координату y_c центра тяжести U-образной пластинки, показанной на рисунке. Для имени функции и аргументов используйте $y_c = \text{centroidU}(w, h, t, d)$, где входные аргументы w, h, t и d являются размерами, показанными на рисунке, а выходной аргумент y_c – координаты.

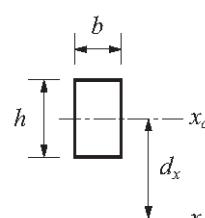
Используйте эту функцию для нахождения y_c для области с размерами: $w = 10$ дюйм, $h = 7$ дюйм, $d = 1.75$ дюйм и $t = 0.5$ дюйм.



- 35.** Момент инерции I_{x_o} прямоугольной области относительно оси x_o , проходящей через ее центр тяжести находится по

формуле $I_{x_o} = \frac{1}{12}bh^3$. Момент инерции относительно оси x параллельной x_o , находится по формуле $I_x = I_{x_o} + Ad_x^2$,

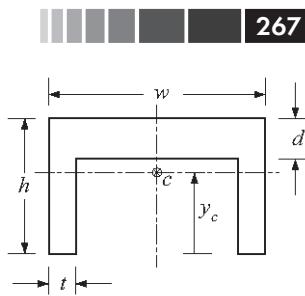
где A – площадь прямоугольника, и d_x есть расстоянием между этими двумя осями.



Напишите пользовательскую функцию MATLAB, которая определяет плоский момент инерции I_{x_o} U-образной балки относительно оси, проходящей через ее центр тяжести (см. рисунок). Для имени функции и аргументов используйте $Ixc=IxcUBeam(w, h, t, d)$, где входные аргументы w, h, t и d есть

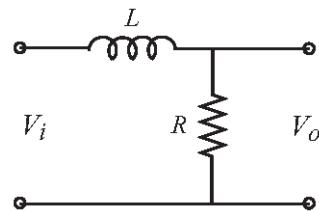
размеры, показанные на рисунке, а выходной аргумент `Ixc` есть I_{x_0} . Для нахождения координат y_c центра тяжести, используйте пользовательскую функцию `centroidU` из задачи 34 как подфункцию в `IxcUBeam`. (Момент инерции составной области получается разбиением области на части и сложением моментов инерции частей.)

Используйте эту функцию для нахождения момента инерции U-образной балки с размерами: $w = 12$ дюйм, $h = 8$ дюйм, $d = 2$ дюйм и $t = 0.75$ дюйм.



- 36.** В низкочастотном RL фильтре (фильтр, который пропускает сигналы с низкими частотами), отношение величин напряжений задается формулой:

$$RV = \left| \frac{V_o}{V_i} \right| = \frac{1}{\sqrt{1 + \left(\frac{\omega L}{R} \right)^2}},$$



где ω – частота входного сигнала.

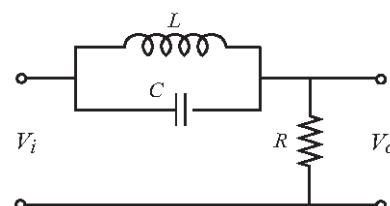
Напишите пользовательскую функцию MATLAB, которая вычисляет это отношение величин. Для имени функции и аргументов используйте `RV = LRFilt(R, L, w)`. Входные аргументы: R – размер резистора в Ом; L – индуктивность обмотки в Гн (Генри) и w – частота входного сигнала в рад/с. Напишите функцию так, чтобы частота w могла быть вектором.

Напишите программу в скрипт-файле, который использует функцию `LRFilt` для создания графика RV как функции ω для $10 \leq \omega \leq 10^6$ рад/с. У графика должна быть логарифмическая шкала по горизонтальной оси (ω). При выполнении скрипта-файла, он просит пользователя ввести значения R и L . Создайте метки осей графика.

Выполните скрипт-файл с $R = 600$ Ом и $L = 0.14$ Н.

- 37.** Цепь, которая фильтрует определенную частоту, показана на рисунке. В этом фильтре отношение величин напряжений задается формулой:

$$RV = \left| \frac{V_o}{V_i} \right| = \frac{|R(1 - \omega^2 LC)|}{\sqrt{(R - R\omega^2 LC)^2 + (\omega L)^2}},$$



где ω – частота входного сигнала. Напишите пользовательскую функцию MATLAB, которая вычисляет это отношение величин. Для имени функции и аргументов используйте `RV=filtfreq(R, C, L, w)`. Входные аргументы: R –

размер резистора (Ω); C – Фарад; L – индуктивность обмотки (Генри) и w – частота входного сигнала в рад/с. Напишите функцию так, чтобы частота w могла быть вектором.

Напишите программу в скрипт-файле, который использует функцию `filterfreq` для создания двух графиков RV как функции ω для $10 \leq \omega \leq 110^4$ рад/с. В одном из графиков $C = 160 \text{ мкФ}$, $L = 45 \text{ мГн}$ и $R = 200 \Omega$. Для второго графика C и L те же самые, а $R = 50 \Omega$. У графика есть логарифмическая шкала на горизонтальной оси (ω). Создайте метки осей графика и легенду.

- 38.** Первая производная $\frac{df(x)}{dx}$ функции $f(x)$ в точке $x = x_0$ может быть аппроксимирована двуточечной центральной разностной формулой:

$$\frac{df(x_0)}{dx} = \frac{f(x_0 + h) - f(x_0 - h)}{2h},$$

где h – небольшое число относительно x_0 . Напишите пользовательскую функцию от функции (см. раздел 7.9), которая вычисляет производную математической функции с использованием двуточечной центральной разностной формулы. Для имени функции и аргументов используйте `dfdx=Funder(Fun, x0)`, где `Fun` – имя для функции, которую передают в `Funder` и x_0 – есть точка, где вычисляется производная. Используйте в двуточечной центральной разностной формуле значение $h = x_0/100$. Используйте пользовательскую функцию `Funder` для вычисления:

- (a) Производной $f(x) = x^3 e^{2x}$ в точке $x_0 = 0.6$.
 (b) Производной $f(x) = 3^x/x^2$ в точке $x_0 = 2.5$.

В обоих случаях сравните ответ полученный от `Funder` с решением полученным аналитически (используйте формат `long`).

- 39.** Новые координаты (X_r, Y_r) точки в xy -плоскости, когда система координат поворачивается вокруг оси z на угол θ (положительный – по часовой стрелке) задаются формулами:

$$\begin{aligned} X_r &= X_0 \cos \theta - Y_0 \sin \theta \\ Y_r &= X_0 \sin \theta + Y_0 \cos \theta, \end{aligned}$$

где (X_0, Y_0) – координаты точки до вращения системы координат. Напишите пользовательскую функцию, которая вычисляет (X_r, Y_r) по заданным (X_0, Y_0) и θ . Для имени функции и аргументов используйте `[xr, yr]=rotation(x, y, q)`, где входные аргументы – это начальные координаты и угол вращения в градусах, а выходные аргументы – новые координаты точки.

- (a) Используйте `rotation` для определения новых координат точки с первоначальными координатами $(6.5, 2.1)$ после поворота на угол 25° вокруг оси Z .

- (b) Рассмотрите функцию $f(x) = (x - 7)^2 + 1.5$ для $5 \leq x \leq 9$. Напишите программу в скрипт-файле, которая создает график этой функции. Затем используйте функцию `rotation` для поворота всех точек этого графика и создайте график повернутой функции. Сделайте оба графика в одном окне Figure и установите диапазон обеих осей от 0 до 10.
40. В лотерее игрок должен угадать правильно r чисел из первых n натуральных чисел. Вероятность P угадывания r чисел из n чисел может быть вычислена по формуле:

$$P = \frac{C_{r,m} C_{(n-r)(r-m)}}{C_{n,r}},$$

где $C_{x,y} = \frac{x!}{y!(x-y)!}$. Напишите пользовательскую функцию MATLAB

для вычисления P . Для имени функции и аргументов используйте `P=ProbLottery(m, r, n)`. Входные аргументы: m – число правильных угадываний; r – число чисел, которые должны быть угаданы и n – число всех возможных натуральных чисел. Для вычисления $C_{x,y}$ используйте подфункцию в `ProbLottery`.

- (a) Используйте `ProbLottery` для вычисления вероятности правильного выбора 3 из 6 чисел, которые выбираются в лотерейной игре из 49 чисел.
- (b) Рассмотрите лотерейную игру, в которой 6 чисел выбираются из 49 чисел. Запишите программу в скрипте-файле, которая выводит на экран таблицу с семью строками и двумя столбцами. Первый столбец имеет номера 0, 1, 2, 3, 4, 5 и 6, которые являются количеством правильно угадываемых чисел. Второй столбец показывает соответствующую вероятность угадывания предположения.



ГЛАВА 8.

Многочлены, подбор кривых и интерполяция

Многочлены – это математические выражения, которые часто используются для решения задач и моделирования в науке и разработке. Во многих случаях уравнение, которое выписывается в процессе решения задачи, является полиномиальным и решение задачи есть нуль многочлена. MATLAB имеет широкий набор функций, которые специально предназначены для работы с многочленами. Как использовать многочлены в MATLAB, описано в разделе 8.1.

Подбор, вычерчивание кривой по точкам – это процесс нахождения функции, которая может быть использована в моделировании данных. Эта функция не обязательно проходит через каждую из точек, но моделирует данные с наименьшей ошибкой. Нет никаких ограничений на типы уравнений, которые могут использоваться для подбора кривой. Однако часто используются многочлены, показательные и степенные функции. В MATLAB подбор кривой может выполняться созданием специальной программы или в интерактивном режиме анализируя данные, которые выводятся на экран в окне графиков Figure. Раздел 8.2 описывает, как использовать программирование в MATLAB для подбора кривой с помощью многочленов и других функций. Раздел 8.4 описывает основной интерфейс, который используется для интерактивного подбора кривой и интерполяции.

Интерполяция – это процесс оценки значений между точками данных. Самый простой вид интерполяции делается при вычерчивании прямой между соседними точками. В более сложной интерполяции используются данные от дополнительных точек. Как интерполировать с MATLAB обсуждается в разделах 8.3 и 8.4.

8.1. Многочлены

Многочлены – это функции, которые имеют вид:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 ,$$

Коэффициенты $a_n, a_{n-1}, \dots, a_1, a_0$ – вещественные числа, а n , которое является неотрицательным целым числом, есть степень, или порядок многочлена.

Примеры многочленов:

$$f(x) = 5x^5 + 6x^2 + 7x + 3$$

многочлен степени 5.

$$f(x) = 2x^2 - 4x + 10$$

многочлен степени 2.

$$f(x) = 11x - 5$$

многочлен степени 1.

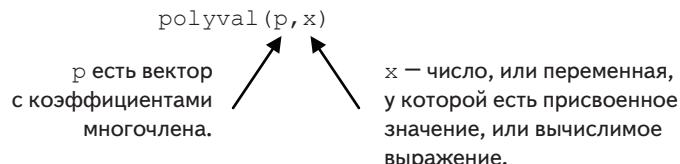
Постоянная (например, $f(x) = 6$) является многочленом степени 0.

В MATLAB многочлены представлены вектором строкой, элементы которой – это коэффициенты многочлена $a_n, a_{n-1}, \dots, a_1, a_0$. Первый элемент – это коэффициент при старшей степени x . Вектор должен включать все коэффициенты, даже те, которые равны 0. Например:

Многочлен	Представление в MATLAB
$8x + 5$	$p = [8 5]$
$2x^2 - 4x + 10$	$d = [2 -4 10]$
$6x^2 - 150$	$h = [6 0 -150]$
Вид в MATLAB: $6x^2 + 0x - 150$	
$5x^5 + 6x^2 - 7x$	$c = [5 0 0 6 0 -7 0]$
Вид в MATLAB: $5x^5 + 0x^4 + 0x^3 + 6x^2 - 7x + 0$	

8.1.1. Значение многочлена

Значение многочлена в точке x может быть вычислено функцией `polyval`, которая имеет вид:



х может быть также вектором или матрицей. В этом случае многочлен вычисляется для каждого элемента (поэлементно) и ответ – вектор, или матрица с соответствующими значениями многочлена.

Пример задачи 8.1. Вычисление многочленов с MATLAB

Для многочлена $f(x) = x^5 - 12.1x^4 + 40.59x^3 - 17.015x^2 - 71.95x + 35.88$:

(a) Вычислить $f(9)$.

(b) Создать график для промежутка $-1.5 \leq x \leq 6.7$.

Решение

Задачу решаем в командном окне.

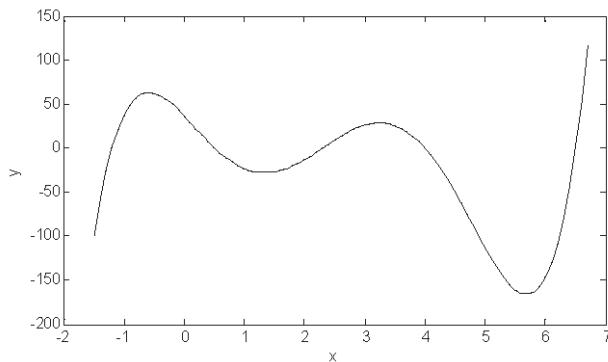
(a) Коэффициенты многочленов присваиваем вектору `p`. Затем используем функцию `polyval` для вычисления значения в точке $x = 9$.

```
>> p = [1 -12.1 40.59 -17.015 -71.95 35.88];
>> polyval(p, 9)
ans =
    7.2611e+003
```

(b) Чтобы графически изобразить многочлен, сначала определяем вектор x с элементами в пределах от -1.5 до 6.7 . Затем создается вектор y со значениями многочлена для каждого элемента x . Наконец, строим график y как функции от x .

```
>> x=-1.5:0.1:6.7;
>> y=polyval(p,x);          Вычисление значения многочлена для каждого
>> plot(x,y)                элемента вектора x.
```

Ниже представлен график, создаваемый MATLAB, (метки осей были добавлены с редактором графиков).



8.1.2. Корни многочлена

Корни многочлена – это значения аргумента, при которых значение многочлена равно нулю. Например, корни многочлена $f(x) = x^2 - 2x - 3$ – это такие значения x , для которых $x^2 - 2x - 3 = 0$ и это два значения $x = -1$ и $x = 3$.

В MATLAB есть функция, называемая `roots`, которая определяет корень, или корни, многочлена. Вид функции:

$r = \text{roots}(p)$

r есть вектор столбец корней многочлена.

p есть вектор строки с коэффициентами многочлена.

Например, корни многочлена в примере типовой задачи 8.1 могут быть найдены следующим образом:

```
>> p= 1 -12.1 40.59 -17.015 -71.95 35.88];
>> r=roots (p)
r =
    6.5000
    4.0000
    2.3000
   -1.2000
    0.5000
```

Когда корни известны, этот многочлен может быть записан так:

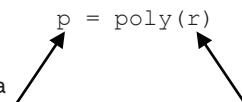
$$f(x) = (x + 1.2)(x - 0.5)(x - 2.3)(x - 4)(x - 6.5)$$

Команда `roots` очень полезна для нахождения корней квадратного уравнения. Например, для нахождения корней $f(x) = 4x^2 + 10x - 8$ достаточно ввести:

```
>> roots([4 10 -8])
ans =
    -3.1375
     0.6375
```

Если известны корни многочлена, то коэффициенты этого многочлена (когда коэффициент при старшем члене = 1) могут быть определены командой `poly`. Форма команды `poly`:

$p = \text{poly}(r)$



р есть вектор строка с коэффициентами многочлена.

r есть вектор столбец корней многочлена.

Например, коэффициенты многочлена в примере типовой задачи 8.1 могут быть получены из корней многочлена (см. выше):

```
>> r=[6.5 4 2.3 -1.2 0.5];
>> p=poly(r)
p =
    1.0000   -12.1000   40.5900  -17.0150  -71.9500  35.8800
```

8.1.3. Сложение, умножение и деление многочленов

Сложение

Два многочлена могут быть сложены (или вычтены) сложением (вычитанием) векторов коэффициентов. Когда порядки многочленов не совпадают (это означает, что векторы коэффициентов имеют разную длину), более короткий вектор должен быть дополнен нулями впереди, чтобы иметь ту же самую длину, как у более

длинного вектора (это называется дополнением, padding). Например, многочлены $f_1(x) = 3x^6 + 15x^5 - 10x^3 - 3x^2 + 15x - 40$ и $f_2(x) = 3x^3 - 2x - 6$ могут быть сложены так:

```
>> p1=[3 15 0 -10 -3 15 -40];
>> p2=[3 0 -2 -6];
>> p=p1+[0 0 0 p2]
p =
    3      15      0      -7      -3      13      -46
```

Три нуля добавлены перед $p2$, так как
порядок $p1$ равен 6, а порядок $p2$ равен 3.

Умножение

Два многочлена могут быть перемножены, используя встроенную функцию MATLAB `conv`, формат которой следующий:

$c = \text{conv}(a, b)$

c есть вектор коэффициентов
произведения многочленов.

a, b – это векторы коэффициентов
перемножаемых многочленов.

- Эти два многочлена не обязаны иметь одинаковый порядок.
- Умножение трех или более многочленов делается многократным применением функции `conv`.

Например, умножение многочленов $f_1(x)$ и $f_2(x)$ упомянутых выше, дает:

```
>> pm=conv(p1,p2)
pm =
    9      45      -6      -78      -99      65      -54      -12      -10      240
```

это означает, что ответ следующий:

$$9x^9 + 45x^8 - 6x^7 - 78x^6 - 99x^5 + 65x^4 - 54x^3 - 12x^2 - 10x + 240.$$

Деление

Многочлен может быть разделен на другой многочлен встроенной функцией MATLAB `deconv`, формат которой следующий:

$[q, r] = \text{deconv}(u, v)$

q – вектор коэффициентов
многочлена частного от деления.

r – вектор коэффициентов
многочлена остатка.

u – вектор коэффициентов
многочлена числителя.

v – вектор коэффициентов
многочлена знаменателя.

Например, деление $2x^3 + 9x^2 + 7x + 240$ на $x + 3$ делается так:

```
>> u=[2 9 7 -6];
>> v=[1 3];
>> [a b]=deconv(u, v)
a =
    2      3      -2          Ответ есть  $2x^2 + 3x - 2$ 
b =
    0      0      0      0          Остаток есть нуль (деление без остатка).
```

Пример деления с остатком, $2x^6 - 13x^5 + 75x^3 + 2x^2 - 60$ делится на $x^2 - 5$:

```
>> w=[2 -13 0 75 2 0 -60];
>> z=[1 0 -5];
>> [g h]=deconv(w, z)
g =
    2   -13   10   10   52          Частное есть  $2x^4 - 13x^3 + 10x^2 + 10x + 52$ 
h =
    0     0     0     0     0     50   200          Остаток есть  $50x + 200$ 
```

Ответ в этом примере: $2x^4 - 13x^3 + 10x^2 + 10x + 52 + \frac{50x + 200}{x^2 - 5}$.

8.1.4. Производные многочленов

Встроенная функция `polyder` может использоваться для вычисления производной одного многочлена, произведения двух многочленов, или частного двух многочленов, как показано в следующих трех командах.

`k = polyder(p)`

Производная одного многочлена. `p` – вектор с коэффициентами многочлена. `k` – вектор с коэффициентами многочлена, который является производной.

`k = polyder(a,b)`

Производная произведения двух многочленов. `a` и `b` – векторы с коэффициентами многочленов сомножителей. `k` – вектор с коэффициентами многочлена, который является производной произведения.

`[n d] = polyder(u,v)`

Производная частного двух многочленов. `u` и `v` – векторы с коэффициентами многочленов числителя и знаменателя. `n` и `d` – векторы с коэффициентами многочленов числителя и знаменателя производной.

Единственная разница между последними двумя командами – это число выходных аргументов. С двумя выходными аргументами MATLAB вычисляет производную частного двух многочленов. С одним выходным аргументом вычисляется производная произведения.

Например, если, $f_1(x) = 3x^2 - 2x + 4$ и $f_2(x) = x^2 + 5$, то производные выражений $3x^2 - 2x + 4$, $(3x^2 - 2x + 4)(x^2 + 5)$ и $(3x^2 - 2x + 4)/(x^2 + 5)$ могут быть определены следующим образом:

```
>> f1= 3 -2 4];
>> f2=[1 0 5];
>> k=polyder(f1)
k =
       6      -2
>> d=polyder(f1, f2)
d =
      12      -6      38
>> [n d]=polyder(f1, f2)
n =
      2      22      -10
d =
      1      0      10      0      25
```

Создание векторов коэффициентов f_1 и f_2 .

Производная f_1 , это $6x - 2$.

Производная $f_1 * f_2$, это $12x^3 - 6x^2 + 38x - 10$.

Производная f_1/f_2 , это $(2x^2 + 22x - 10)/(x^4 + 10x^2 + 25)$.

8.2. Подбор кривой

Подбор кривой., также называемый регрессионным анализом, является процессом подбора функции по ряду точек данных. Тогда эта функция может использоваться в качестве математической модели данных. Поскольку есть много типов функций (линейные, многочлены, степенные, показательные функции и т. д.), подбор кривой может быть сложным процессом. Часто имеется некоторая идея относительно типа функции, которая могла бы соответствовать определенным данным и тогда нужно только определить коэффициенты функции. В других ситуациях, где неизвестно о данных, можно делать разные типы графиков, которые дают информацию о возможных видах функций, которые могли бы хорошо соответствовать данным. Этот раздел описывает некоторые из основных методов для подбора кривой и инструменты, которые имеет MATLAB для этих целей.

8.2.1. Подбор кривой многочленами; функция *polyfit*

Многочлены могут использоваться для подбора к точкам данных двумя способами. В одном случае многочлен проходит через все точки данных, а в другом – многочлен не обязательно проходит через каждую из точек, но везде дает хорошее приближение данных. Эти два способа описаны ниже.

Многочлены, которые проходят через все точки

Когда даны n точек (x_i, y_i) , можно написать многочлен степени $n - 1$, который проходит через все эти точки. Например, если даны две точки, можно написать линейное уравнение вида $y = mx + b$, которое проходит через эти две точки. Для трех точек уравнение имеет вид $y = ax^2 + bx + c$. Для n точек полиномиальное уравнение имеет вид $y = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0$. Коэффициенты этого многочлена (их всего n) определяются как решение системы n уравнений, которая получается подстановкой координат (x_i, y_i) каждой из n точек в данное полиномиальное уравнение. Как будет показан ниже в этом разделе, многочлены высокой степени могут давать большую ошибку, когда они используются для оценки значений между точками данных.

Многочлены, которые не обязательно проходят через каждую точку

Когда даны n точек, можно написать многочлен степени меньше чем $n - 1$ не обязательно проходящий через каждую из точек, но который равномерно всюду аппроксимирует данные. Наиболее распространенный метод нахождения наилучшей подгонки к точкам данных – это метод наименьших квадратов. В этом методе коэффициенты многочлена определяются минимизацией суммы квадратов невязок (разностей) во всех точках данных. Невязка R_i в каждой точке x_i определяется как разность между значением многочлена $f(x_i)$ и значением данных y_i . Например, рассмотрим случай нахождения прямой, которая лучше всего соответствует четырем точкам данных как показано в рис. 8.1.

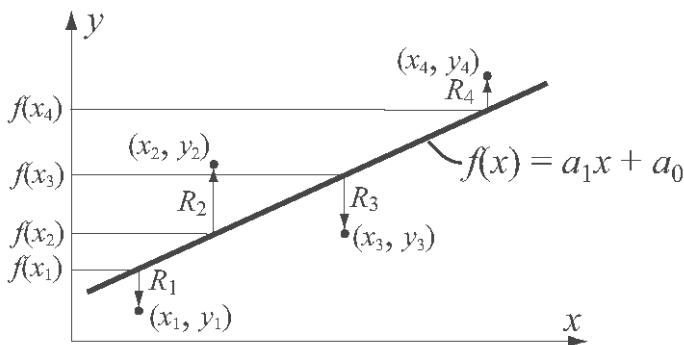


Рис. 8.1. Подбор многочлена первой степени к четырем точкам методом наименьших квадратов

Эти точки есть (x_1, y_1) , (x_2, y_2) , (x_3, y_3) и (x_4, y_4) , а многочлен первой степени может быть записан как $f(x) = a_1x + a_0$. Невязка R_i в каждой точке x_i есть разность между значением функции в точке x_i и y_i , $R_i = f(x_i) - y_i$. Сумма квадратов разностей во всех точках задается выражением:

$$R = [f(x_1) - y_1]^2 + [f(x_2) - y_2]^2 + [f(x_3) - y_3]^2 + [f(x_4) - y_4]^2.$$

или, после замены f многочленом $a_1x + a_0$ в каждой точке:

$$R = [a_1x_1 + a_0 - y_1]^2 + [a_1x_2 + a_0 - y_2]^2 + [a_1x_3 + a_0 - y_3]^2 + [a_1x_4 + a_0 - y_4]^2.$$

На данном этапе R есть функция a_1 и a_0 (многочлен второй степени). Минимум R может быть определен приравниванием к нулю частных производных R относительно a_1 и a_0 (два уравнения):

$$\frac{\partial R}{\partial a_1} = 0 \quad \text{и} \quad \frac{\partial R}{\partial a_0} = 0.$$

Это приводит к системе двух уравнений с двумя неизвестными a_1 и a_0 . Решение этих уравнений дает значения коэффициентов многочлена который наилучшим образом приближает данные. Та же самая процедура может проведена с большим количеством точек и с многочленом более высокого порядка. Подробности о методе наименьших квадратов можно найти в книгах по численному анализу.

Подбор кривой многочленами выполняется в MATLAB функцией `polyfit`, которая использует метод наименьших квадратов. Основной формат функции `polyfit`:

`p = polyfit(x, y, n)`

`p` — вектор коэффициентов многочлена, который приближает данные.

`x` — вектор с горизонтальными координатами точек данных (независимая переменная).

`y` — вектор с вертикальными координатами точек данных (зависимая переменная).

`n` — степень многочлена.

Функция `polyfit` может использоваться на одном и том же множестве из m точек, для подбора многочлена любого порядка до $m - 1$. Если $n = 1$, то многочлен — прямая, если $n = 2$, то многочлен представляет параболу и так далее. Многочлен проходит через все точки если $n = m - 1$ (порядок многочлена на единицу меньше чем число точек). Необходимо отметить, что многочлен, который проходит через все точки, или многочлены высокого порядка, не обязательно дают лучшую подгонку всюду. Многочлены высокого порядка могут значительно отклоняться между точками данных.

Рис. 8.2 показывает, как многочлены разных степеней соответствуют одному и тому же множеству точек данных. Множество данных задано семью точками $(0.9, 0.9), (1.5, 1.5), (3, 2.5), (4, 5.1), (6, 4.5), (8, 4.9)$ и $(9.5, 6.3)$. Эти точки приближаются многочленами степеней 1 – 6 с использованием функции `polyfit`. Каждый график рис. 8.2 показывает одни и те же точки данных, отмеченные кругами и подобранную кривую линию, которая соответствует многочлену указанной степени. Можно заметить, что многочлен с $n = 1$ является прямой, а для $n = 2$ – слегка кривая линия. При увеличении степени многочлена, у линии появляется больше из-

гibов для того, чтобы проходить ближе к большему количеству точек. Когда $n = 6$, что на единицу меньше числа точек, линия проходит через все точки. Однако, между некоторыми из точек, линия значительно отклоняется от тренда данных.

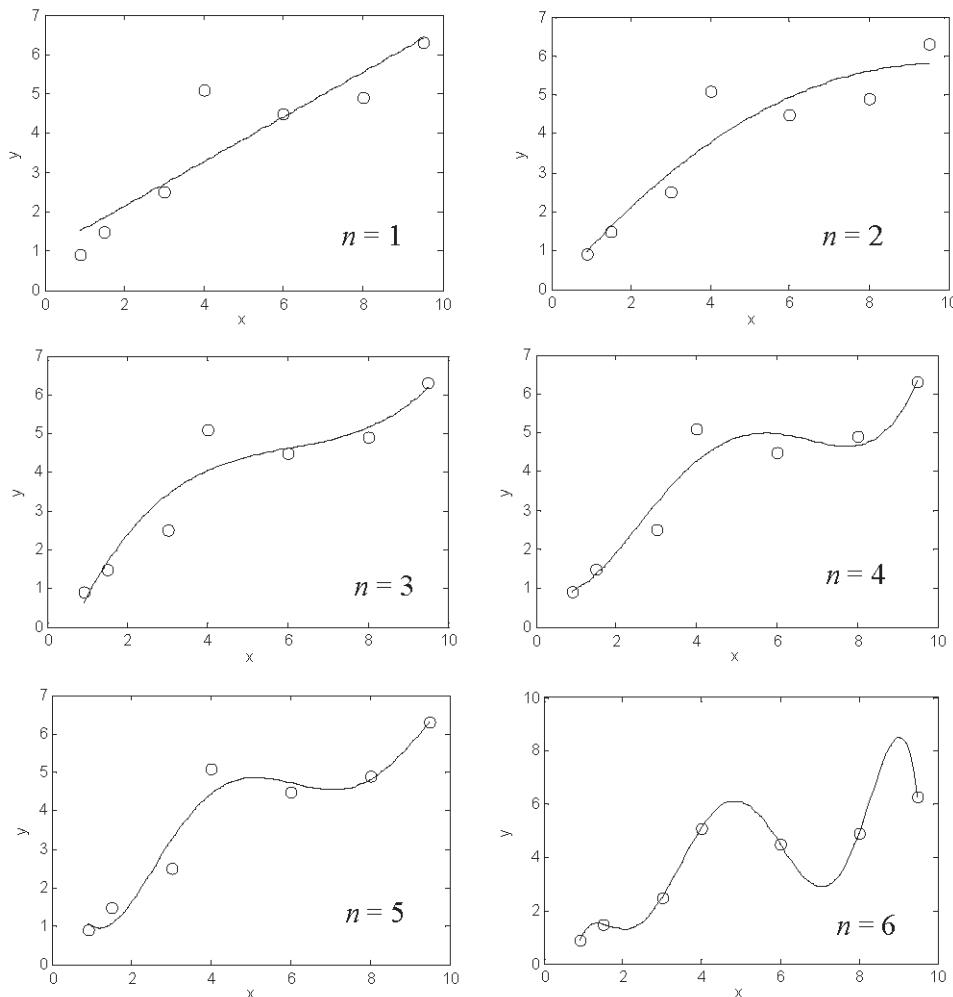


Рис. 8.2. Подгонка данных многочленами разного порядка

Ниже показан обычный скрипт-файл, используемый для создания одного из графиков на рис. 8.2 (многочлен с $n = 3$). Отметим, что для создания графика многочлена (линии) создается новый вектор x_p с малым шагом. Затем этот вектор используется функцией `polyval` для создания вектора y_p значений многочлена для каждого элемента x_p .

```

x=[0.9 1.5 3 4 6 8 9.5];           Создание векторов x и у с
y=[0.9 1.5 2.5 5.1 4.5 4.9 6.3];   координатами точек данных.
p=polyfit(x,y,3)                  Создание векторов p функцией polyfit.
xp=0.9:0.1:9.5;                   Создание вектора xp для построения графика многочлена.
yr=polyval(p,xp);                Создание вектора yr значений многочлена в точках xp.
plot(x,y,'o',xp,yr)              График семи точек и многочлена.
xlabel('x'); ylabel('y')

```

При выполнении скрипта-файла на экран в командном окне выводится следующий вектор p.

```

p =
    0.0220      -0.4005      2.6138      -1.4158

```

Это означает, что многочлен третьей степени на рис. 8.2 имеет вид

$$0.022x^3 - 0.4005x^2 + 2.6138x - 1.4158.$$

8.2.2. Подбор кривой другими функциями

Во многих случаях в науке и разработке для подбора к определенным данным требуются функции, которые не являются многочленами. Теоретически, любая функция может использоваться для моделирования данных в пределах некоторого диапазона. Однако для заданного конкретного набора данных некоторые функции обеспечивают лучшую подгонку чем другие. Кроме того, определение коэффициентов оптимальной подгонки может быть для некоторых функций более трудным чем для других. В этом разделе рассматривается подбор кривой степенными, экспоненциальными, логарифмическими и обратными (дробными) функциями, которые обычно используются. Вид этих функций:

$y = b x^m$	(степенная функция);
$y = b e^{mx}$ или $y = b 10^{mx}$	(показательная функция);
$y = m \ln(x) + b$ или $y = m \log(x) + b$	(логарифмическая функция);
$y = \frac{1}{mx + b}$	(обратная дробная функция).

Все эти функции могут быть легко подобраны к определенным данным с помощью функции `polyfit`. Для этого нужно записать функцию в виде линейного многочлена ($n = 1$) типа

$$y = m x + b.$$

Логарифмическая функция уже находится в этом виде, а степенная, показательная функции и обратное выражение могут быть переписаны как:

$$\ln(y) = m \ln(x) + \ln(b) \quad (\text{степенная функция});$$

$\ln(y) = mx + \ln(b)$ или $\log(y) = mx + \log(b)$ (показательная функция);

$$\frac{1}{y} = mx + b \quad (\text{обратная дробная функция}).$$

Эти уравнения описывают линейные соотношения между $\ln(y)$ и $\ln(x)$ для степенной функции, между $\ln(y)$ и x для показательной функции, между y и $\ln(x)$ или $\log(x)$ для логарифмической функции и между $1/y$ и x для обратной функции. Это означает, что при определении подходящих констант m и b для наилучшей подгонки может использоваться функция `polyfit(x, y, 1)`, если вместо x и y использовать следующие аргументы.,

Функция		Вид функции <code>polyfit</code>
Степенная	$y = b x^m$	<code>p=polyfit(log(x), log(y), 1)</code>
Показательная	$y = b e^{mx}$ или $y = b 10^{mx}$	<code>p=polyfit(x, log(y), 1)</code> или <code>p=polyfit(x, log10(y), 1)</code>
Логарифмическая	$y = m \ln(x) + b$ или $y = m \log(x) + b$	<code>p=polyfit(log(x), y, 1)</code> или <code>p=polyfit(log10(x), y, 1)</code>
Обратная дробь	$y = \frac{1}{mx + b}$	<code>p=polyfit(x, 1./y, 1)</code>

Результат функции `polyfit` есть вектор `p` с двумя элементами. Первый элемент, `p(1)`, является константой m , а второй элемент, `p(2)`, является b для логарифмических и обратных дробей, $\ln(b)$ или $\log(b)$ для показательной функции и $\ln(b)$ для степенной функции ($b = e^{p(2)}$ или $b = 10^{p(2)}$ для показательной функции, и $b = e^{p(2)}$ для степенной функции).

Для конкретных данных можно в какой-то степени оценить, у какой из функций есть потенциал для того, чтобы обеспечить хорошую подгонку. Это можно сделать графически изображением данных с использованием различные комбинации линейных и логарифмических осей. Если точки данных в одном из таких графиков лежат вдоль прямой, то соответствующая функция может обеспечить хорошую подгонку согласно списку ниже.

Ось x	Ось y	Функция
Линейная	Линейная	Линейная $y = mx + b$
Логарифмическая	Логарифмическая	Степенная $y = b x^m$
Линейная	Логарифмическая	Показательная $y = b e^{mx}$ или $y = b 10^{mx}$
Логарифмическая	Линейная	Логарифмическая $y = m \ln(x) + b$ или $y = m \log(x) + b$
Линейная	Линейная (график $1/y$)	Обратная дробная $y = \frac{1}{mx + b}$

Другие соображения в выборе функции:

- Показательные функции не могут проходить через начало координат.
- Показательные функции могут подгоняться только к данным, у которых все y положительны или все y отрицательны.
- Логарифмические функции не могут моделировать $x = 0$ или отрицательные значения x .
- Для степенной функции $y = 0$, когда $x = 0$.
- Обратная дробная не может моделировать $y = 0$.

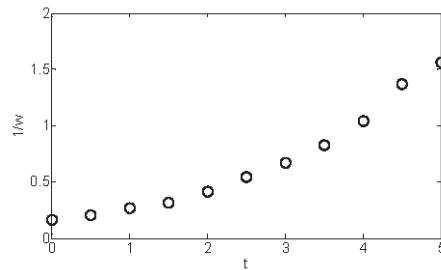
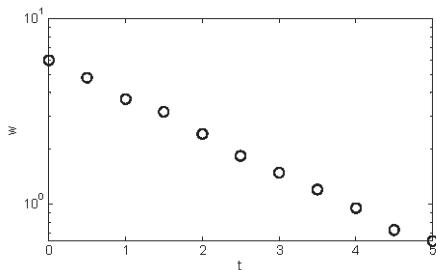
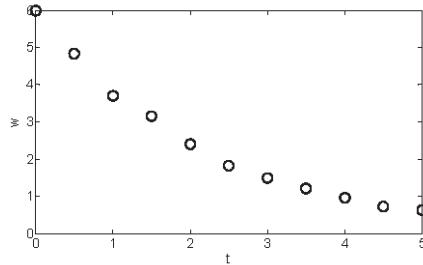
Пример задачи 8.2. Подбор уравнения к точкам данных

Даны следующие точки данных. Определить функцию $w = f(t)$ (t – независимая переменная, w – зависимая переменная) того типа, которые обсуждались в этом разделе, и которая лучше всего приближает данные.

t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
w	6.00	4.83	3.70	3.15	2.41	1.83	1.49	1.21	0.96	0.73	0.64

Решение

Изобразим сначала данные графически с линейными шкалами на обеих осях. Рисунок показывает, что линейная функция не будет давать лучшую подгонку, так как точки, кажется, не выстраиваются в линию вдоль прямой. Из других возможных функций исключаем логарифмическую функцию, поскольку для первой точки $t = 0$, и степенная функция тоже исключается, поскольку при $t = 0$, $w \neq 0$. Чтобы проверить, могут ли другие две функции (показательная функция и обратная величина) дать лучшую подгонку, сделаем два дополнительных графика, показанные ниже. У графика слева логарифмический масштаб на вертикальной оси и линейный – на горизонтальной. В графике справа у обеих осей линейные шкалы и на вертикальной оси графически изображается величина $1/w$.



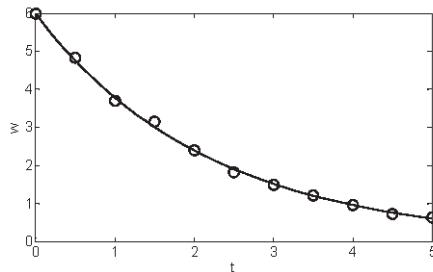
На левом рисунке видно, что точки данных выстраиваются вдоль прямой линии. Это показывает, что показательная функция типа $y = be^{mx}$ может дать хорошую подгонку к данным. Программа в скрипте-файле, которая определяет константы b и m и графически изображает точки данных и функцию, приведена ниже.

```
t=0:0.5:5;          Создание векторы t и w с координатами точек данных.
w=[6 4.83 3.7 3.15 2.41 1.83 1.49 1.21 0.96 0.73 0.64];
p=polyfit(t,log(w),1);      Использование функции polyfit c t и log(w).
m=p(1)
b=exp(p(2))          Определение коэффициента b.
tm=0:0.1:5;          Создание вектора tm для построения графика многочлена.
wm=b*exp(m*tm);     Вычисление значений функции в каждом элементе tm.
plot(t,w,'o',tm,wm)   Графики точек данных и функции.
```

При выполнении программы значения констант m и b выводятся на экран в командном окне.

```
m =
-0.4580
b =
5.9889
```

Ниже представлен график, созданный программой, который показывает точки данных и функцию (с метками осей, добавленными при помощи редактора графиков).



Отметим здесь, что в дополнение к степенным, экспоненциальным, логарифмическим и обратным дробным функциям, которые обсуждались в этом разделе, много других функций, могут быть записаны в форме, подходящей для подбора кривой с функцией `polyfit`. Один пример, когда функция вида $y = e^{(a_2x^2+a_1x+a_0)}$ подгоняется к точкам данных, используя функцию `polyfit` с многочленом третьего порядка, описан в примере типовой задачи 8.7.

8.3. Интерполяция

Интерполяция – это оценка значений между точками данных. MATLAB имеет функции интерполяции, основанные на многочленах, которые описаны в этом разделе, и на преобразовании Фурье, что выходит за рамки этой книги. В одномерной интерполяции каждая точки имеет одну независимую переменную (x) и одну зависимую переменную (y). В двумерной интерполяции каждая точки имеет две независимые переменные (x и y) и одну зависимую переменную (z).

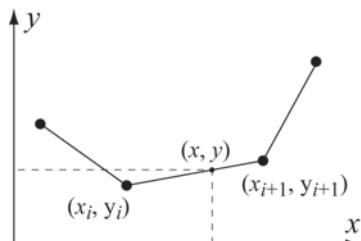
Одномерная интерполяция

Если имеются только две точки данных, эти точки могут быть соединены прямой и для оценки значений между точками может использоваться линейное уравнение (многочлен первого порядка). Как было отмечено в предыдущем разделе, если имеется три (или четыре) точки данных, то может быть определен многочлен второго (или третьего) порядка, который проходит через эти точки и тогда он может использоваться для оценки значений между точками. При увеличении числа точек для прохождения через эти точки требуются многочлены высокого порядка. Однако такие многочлены не обязательно дают хорошее приближение значений между точками. Это показано на рис. 8.2 при $n = 6$.

Более точная интерполяция может быть получена, когда вместо того, чтобы рассмотреть все точки в наборе данных (при использовании одного многочлена, который проходит через все точки), рассматривают только несколько точек данных в окрестности которых интерполяция необходима. В этом методе, названном сплайновой интерполяцией, используются много многочленов невысоких степеней, каждый из которых подходит только в небольшой области набора данных.

Самый простой метод сплайновой интерполяции называется линейной сплайновой интерполяцией. В этом методе, показанном справа, каждые две смежные точки соединены прямой (многочлен первой степени). Уравнение прямой, которая проходит через две смежные точки (x_i, y_i) и (x_{i+1}, y_{i+1}) и может использоваться для вычисления значения y для любого x между точками, записывается так:

$$y = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} x + \frac{y_i x_{i+1} - y_{i+1} x_i}{x_{i+1} - x_i}.$$



В линейной интерполяции линия между двумя точками данных имеет постоянный наклон и он может меняться в каждой точке. Более гладкая интерполяционная кривая может быть получена при использовании квадратных или кубических многочленов. В этих методах, называемых квадратичными сплайнами и кубическими сплайнами, для интерполяции между каждыми двумя точками используются многочлены второй, или третьей степени. Коэффициенты многочлена определяются с использованием точек данных, которые смежны с этими двумя

точками данных. Теоретическая основа для определения констант многочленов выходит за рамки этой книги и может быть найдена в книгах по численному анализу.

Одномерная интерполяция в MATLAB выполняется функция `interp1` (последний знак – цифра один), которая имеет следующий формат:

`yi = interp1(x, y, xi, 'method')`

yi
 \uparrow
 yi – интерполированное значение.

x
 \uparrow
 x – вектор с горизонтальными координатами точек входных данных (независимая переменная).

y
 \uparrow
 y – вектор с вертикальными координатами точек входных данных (зависимая переменная).

xi
 \uparrow
 xi – горизонтальная координата точки интерполяции (независимая переменная).

$'method'$
 \uparrow
 Метод интерполяции, введенный как строка (дополнительный).

- Вектор x должен быть монотонным (с элементами в порядке по возрастанию или по убыванию).
- xi может быть скаляром (интерполяция одной точки) или вектором (интерполяция многих точек). yi – скаляр или вектор с соответствующими интерполированными значениями.
- MATLAB может сделать интерполяцию, используя один из нескольких методов, который может быть указан. Эти методы включают:

<code>'nearest'</code> <code>'linear'</code> <code>'spline'</code> <code>'pchip'</code>	Возвращает значение точки данных, ближайшей к интерполируемой точке. Использует линейную сплайновую интерполяцию. Использует кубическую сплайновую интерполяцию. Использует кусочную кубическую эрмитову интерполяцию, также названную <code>'cubic'</code> .
--	--

- Когда используются методы `'nearest'` и `'linear'`, значение(я) xi должно быть в пределах области x . Если используются методы `'spline'` или `'pchip'`, xi может иметь значения вне области x и функция `interp1` выполняет экстраполяцию.
- Метод `'spline'` может давать большие ошибки, если точки входных данных расположены неравномерно так, что некоторые точки намного ближе друг к другу, чем другие.
- Спецификация метода `'method'` является дополнительной. Если метод не задан, то по умолчанию значение `'linear'`.

Пример задачи 8.3. Интерполяция

Даны следующие точки данных, которые являются значениями функции $f(x) = 1.5^x \cos(2x)$. Использовать линейный, сплайн и pchip методы интерполяции для вычисления значения y между точками. Создать рисунок для каждого из методов интерполяции. На рисунке показать точки, график функции и кривые, которые соответствуют методам интерполяции.

x	0	1	2	3	4	5
y	1.0	-0.6242	-1.4707	3.2406	-0.7366	-6.3717

Решение

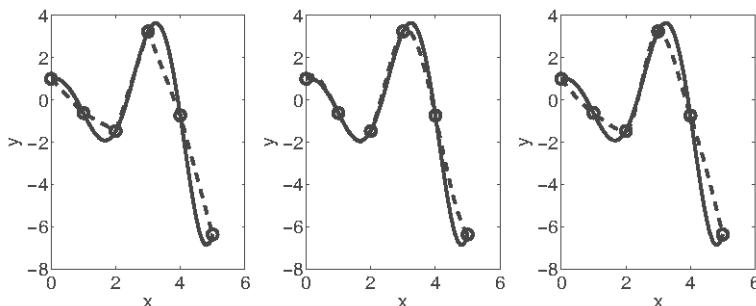
Следующая программа, записанная в скрипт-файле, решает эту задачу:

```

x=0:1.0:5;           Создание векторов x и y с координатами точек данных.
y=[1.0 -0.6242 -1.4707 3.2406 -0.7366 -6.3717];
xi=0:0.1:5;          Создание вектора xi с точками для интерполяции.
                      Вычисление у точек из:
yilin=interp1(x,y,xi,'linear');      линейной интерполяции.
yispl=interp1(x,y,xi,'spline');       сплайновой интерполяции.
yipch=interp1(x,y,xi,'pchip');        pchip интерполяции.
yfun=1.5.^xi.*cos(2*xi);            из функции.
subplot(1,3,1)
plot(x,y,'o',xi,yfun,xi,yilin,'--');
subplot(1,3,2)
plot(x,y,'o',xi,yfun,xi,yispl,'--');
subplot(1,3,3)
plot(x,y,'o',xi,yfun,xi,yipch,'--');

```

Ниже показаны три рисунка, созданные программой (метки осей были добавлены с редактором графиков). Точки данных отмечены с кругами, кривые интерполяции графически изображены с пунктирными линиями, а функция показана сплошной линией. Слева показана линейная интерполяция, в середине – сплайн, а справа показана pchip интерполяция.

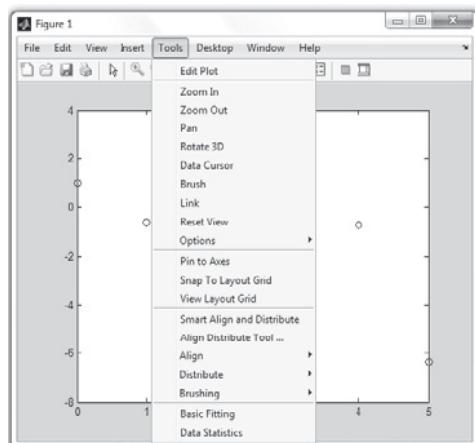


8.4. Базовый интерфейс для подбора Basic fitting

Базовый интерфейс для подбора (Basic fitting) – это инструмент, который может использоваться для подбора кривой и интерполяции в интерактивном режиме. При использовании этого интерфейса пользователь может:

- Выполнить подбор кривой для точек данных с многочленами различных степеней до 10, со сплайновыми и эрмитовыми методами интерполяции.
- Графически изобразить различные подборы кривых на одном графике так, чтобы они можно было сравнить.
- Графически изобразить невязки различных подборов многочленами и сравнить нормы невязок.
- Вычислить значения отдельных точек с разными подборами.
- Добавить уравнения многочленов к графику.

Чтобы активировать базовый интерфейс для подбора, пользователь сначала должен создать график точек данных. После этого активация интерфейса делается просто выбором **Basic Fitting** в меню **Tools**, как показано справа. Открывается окно базового интерфейса, показанное в рис. 8.3. При открытии сначала видна только одна панель (панель **Plot fits**). Окно может быть расширено, чтобы показать вторую панель (панель **Numerical results**), щелкнув по кнопке \rightarrow . Один щелчок прибавляет первую часть панели, а второй щелчок заставляет окно выглядеть как на рис. 8.3. Окно может быть свернуто назад, щелкнув по кнопке \leftarrow . Первые два элемента в окне базового интерфейса связаны с выбором точек данных:



Select data: Используется для выбора определенного множества точек данных для подбора кривой на рисунке, который имеет более одного множества точек данных. Для подбора кривой может быть использовано только одно множество точек данных, но несколько подборов могут быть выполнены одновременно для одного и того же множества.

Center and scale x data: Когда этот флагок, данные центрируются к нулевому среднему значению и масштабируются к единичному стандартному отклонению. Это может быть необходимо для улучшения точности числового вычисления.

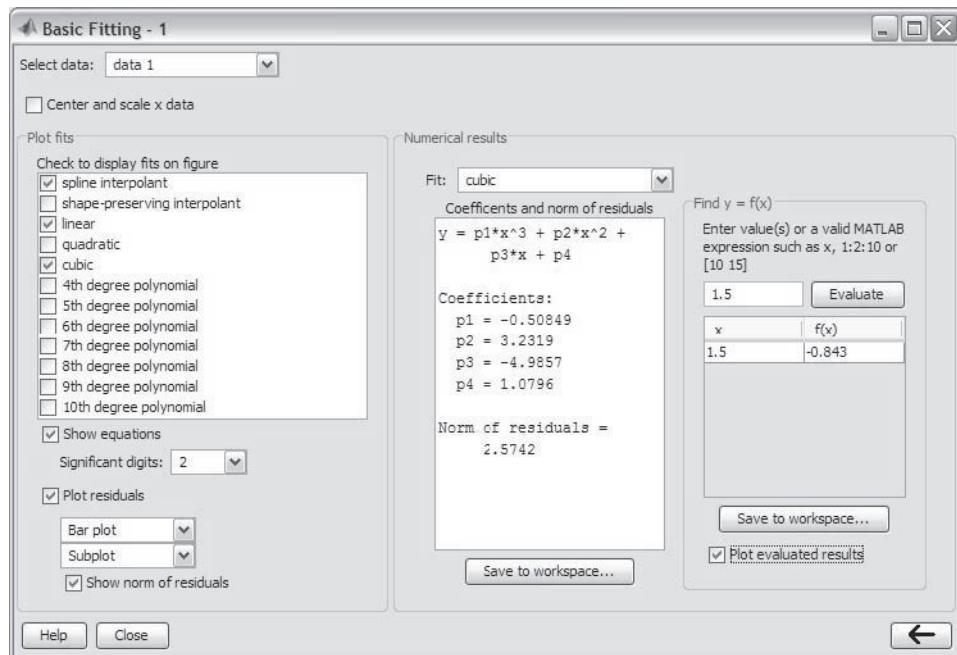


Рис. 8.3. Окно базового интерфейса для подгонки

Следующие четыре элемента находятся в панели **Plot fits** и связаны с выводом на экран кривой подгонки.

Check to display fits on figure: Пользователь выбирает типы подгонки, которые будут выведены на экран на рисунке. Можно выбрать следующие методы интерполяции: сплайновую интерполяцию, которая использует функцию `spline`, эрмитову интерполяцию, которая использует функцию `ppchip` и многочлены различных степеней, которые используют функцию `polyfit`. Можно одновременно выбрать несколько подборов кривых и вывести их на экран.

Show equations: Когда этот флагок установлен, на рисунке выводятся уравнения многочленов, которые были выбраны для подгонки. Уравнения выводятся на экран с числом значащих разрядов, выбранных в соседнем поле.

Plot residuals: Когда этот флагок установлен, создается график, который показывает невязки в каждой точке данных (невязки определены в разделе 8.2.1). Варианты в меню включают график баров, график рассеяния и график линии, который может быть выведен на экран как подграфик в том же самом окне графиков **Figure**, где график точек данных, или как отдельный график в отдельном окне графиков **Figure**.

Show norm of residuals: Когда этот флагок установлен, на экран выводится норма всех разностей (невязок) в графике разностей. Норма разности – это мера качества подгонки. Меньшая норма соответствует лучшей подгонке.

Следующие три элемента находятся на панели численных результатов **Numerical results**. Они предоставляют числовую информацию для одной подгонки, независимо от других подгонок, выведенных на экран:

Fit: Здесь пользователь выбирает подгонку, которая будет исследована в цифровой форме. Подгонка показывается на графике, только если она выбрана на панели **Plot fit**.

Coefficients and norm of residuals: Выводит на экран числовые результаты для подбора многочлена, который выбран в меню **Fit**. Это включает коэффициенты многочлена и норму разностей. Результаты могут быть сохранены, щелкнув по кнопке **Save to workspace**.

Find $y = f(x)$: Обеспечивает средство для того, чтобы оно получило интерполированный (или экстраполируемый) числовые значения для указанных значений независимой переменной. Введите значение независимой переменной в поле, и щелкните по кнопке **Evaluate**. То, когда **Plot evaluated results** проверено, точка выведена на экран на графике.

В качестве примера используем базовый интерфейс для подбора кривой к точкам данных в типовой задаче 8.3. Основное окно базового интерфейса для подбора показано на рис. 8.3, а соответствующее окно **Figure** показано на рис. 8.4.

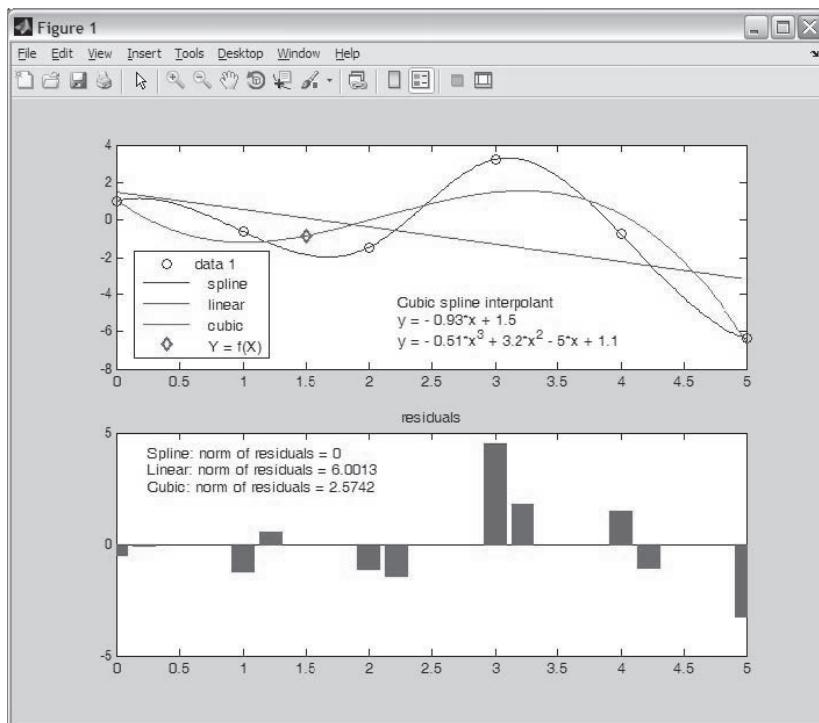


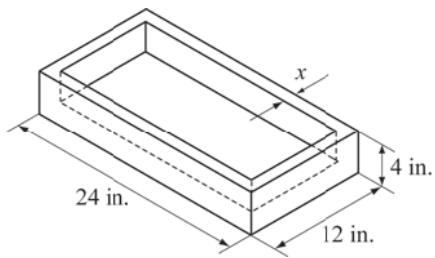
Рис. 8.4. Окно базового интерфейса для подгонки

Окно графиков **Figure** включает изображение точек, одну интерполяционную (сплайновую) подгонку, два подбора многочленами (линейный и кубический), показаны уравнения подобранных многочленов и отмечена точка $x = 1.5$, которая вводится в поле **Find $y = f(x)$** основного окна базового интерфейса. Окно графиков **Figure** включает также график разностей (невязок) для подборов многочленами и отображение их нормы.

8.5. Примеры приложений MATLAB

Пример задачи 8.4. Определение толщины стенок коробки

Внешние размерности прямоугольной коробки (нижняя часть и четыре стороны без крышки), сделанной из алюминия, есть 24-на-12-на-4 дюйма. Толщина стенок нижней части и сторон равна x . Вывести выражение, которое связывает вес коробки и толщину стенок x . Определить толщину x для коробки, которая весит 15 фунтов. Удельный вес алюминия равен 0.101 фунт/дюйм³.



Решение

Объем алюминия V_{al} вычисляется через вес W коробки по формуле:

$$V_{al} = \frac{W}{\gamma},$$

где γ – удельный вес. Объем алюминия, основанный на размерах коробки, находится по формуле:

$$V_{al} = 24 \cdot 12 \cdot 4 - (24 - 2x)(12 - 2x)(4 - x),$$

где внутренний объем коробки вычитается из внешнего объема. Это уравнение может быть переписано в виде уравнения

$$(24 - 2x)(12 - 2x)(4 - x) + V_{al} - (24 \cdot 12 \cdot 4) = 0,$$

левая часть которого является многочленом третьей степени. Корень этого многочлена – это требуемая толщина x . Программа в скрипте-файле, которая определяет многочлен и находит его корни:

```
W=15; gamma=0.101;
VAlum=W/gamma;
a=[-2 24];
b=[-2 12];
```

Присвоение W и gamma. Вычисление объема алюминия. Присвоение многочлена $24 - 2x$ переменной a. Присвоение многочлена $12 - 2x$ переменной b.	→
--	---

```
c=[-1 4];
Prисвоение многочлена  $4 - x$  переменной c.
Vin=conv(c, conv(a,b));
Перемножение этих многочленов.
polyeq=[0 0 0 (VAlum-24*12*4)]+Vin Сложение  $V_{al} - 24 \cdot 12 \cdot 4$  и Vin.
x=roots(polyeq)
```

Отметим, что на предпоследней строке для прибавления величины $V_{al} - (24 \cdot 12 \cdot 4)$ к многочлену Vin , эта величина должна быть записана как многочлен того же порядка, что и Vin (Vin – многочлен третьего порядка). При выполнении программы (сохраненной под именем Chap8SamPro4), коэффициенты многочлена и значение x выводятся на экран следующим образом:

```
>> Chap8SamPro4
polyeq =
    -4.0000 88.0000 -576.0000 148.5149
x =
    10.8656 + 4.4831i
    10.8656 - 4.4831i
    0.2687
Этот многочлен есть:
 $-4x^3 + 88x^2 - 57x + 148.5149.$ 
Многочлен имеет один действительный корень,  $x = 0.2687$  in,
который и является толщиной алюминиевой стенки.
```

Пример задачи 8.5. Высота плавания бакена

Алюминиевая тонкостенная сфера используется в качестве бакена. Сфера имеет радиус 60 см и толщину стенки 12 мм. Плотность алюминия $\rho_{Al} = 2690$ кг/м³. Бакен находится в океане, где плотность воды составляет 1030 кг/м³. Определите высоту h между верхушкой бакена и поверхностью океана.

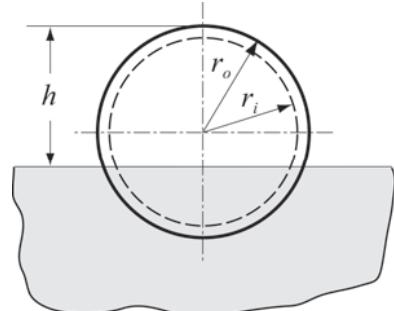
Решение

Согласно закону Архимеда, выталкивающая сила, действующая на тело, помещенное в жидкость, равна весу вытесненной жидкости. Соответственно, алюминиевая сфера будет погружена в воду настолько, что вес сферы равен весу вытесненной частью сферы жидкости. Вес сферы задается формулой

$$W_{sph} = \rho_{Al} V_{Al} g = \rho_{al} \frac{4}{3} \pi (r_0^3 - r_i^3) g,$$

где V_{al} – объем алюминия; r_0 и r_i – это внешний и внутренний радиусы сферы, соответственно; и g – гравитационное ускорение.

Вес воды, вытесненной частью сферы, определяется формулой:



$$W_{wtr} = \rho_{wtr} V_{wtr} g = \rho_{wtr} \frac{1}{3} \pi (2r_0 - h)^2 (r_0 + h) g .$$

Приравнивая эти два веса, получаем следующее уравнение:

$$h^3 - 3r_0 h^2 + 4r_0^3 - 4 \frac{\rho_{Al}}{\rho_{wtr}} (r_0^3 - r_i^3) = 0 .$$

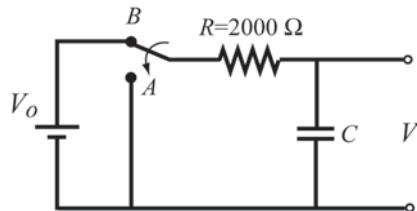
Последнее уравнение – это многочлен третьей степени относительно h . Корень этого многочлена и есть ответ.

Решение получается записью этого многочлена в MATLAB и затем использованием функции `roots` для определения значения h . Это выполнено в следующем скрипте-файле:

<code>rout=0.60; rin=0.588;</code>	Присвоение радиусов переменным.
<code>rhoalum=2690; rhowtr=1030;</code>	Присвоение плотностей переменным.
<code>a0=4*rout^3-4*rhoalum*(rout^3-rin^3)/rhowtr;</code>	Присвоение коэффициента a_0 – свободного члена.
<code>p = [1 -3*rout 0 a0];</code>	Создание вектора коэффициентов многочлена.
<code>h = roots(p)</code>	Вычисление корней из многочлена.

Пример задачи 8.6. Определение объема конденсатора

Электрический конденсатор имеет неизвестную емкость. Для определения этой емкости конденсатор включается в показанную на рисунке цепь. Вначале переключатель включен установлен в положение B и конденсатор заряжается. Затем переключатель устанавливается в положение A и конденсатор разряжается через резистор. При разрядке конденсатора измеряется напряжение от конденсатора в течение 10 сек с интервалом 1 сек. Результаты измерения приведены в таблице ниже. Изобразите графически напряжение как функцию времени и определите емкость конденсатора, подбирая экспоненту к этим точкам данных.



$t(c)$	1	2	3	4	5	6	7	8	9	10
$V(B)$	9.4	7.31	5.15	3.55	2.81	2.04	1.26	0.97	0.74	0.58

Решение

Когда конденсатор разряжается через резистор, напряжение конденсатора как функция времени задается формулой

$$V = V_0 e^{(-t)/(RC)},$$

где V_0 – начальное напряжение, R – сопротивление резистора и C – емкость конденсатора. Как было объяснено в разделе 8.2.2, показательная функция может быть записана как линейная относительно $\ln(V)$ и t в виде:

$$\ln(V) = \frac{-1}{RC}t + \ln(V_0).$$

Это уравнение вида $y = mx + b$ и оно может быть подогнано к точкам данных с использованием функции `polyfit(x, y, 1)`, где t – независимая переменная x и $\ln(V)$ – зависимая переменная y . Коэффициенты m и b , определяемые функцией `polyfit`, используются для нахождения C и V_0 по формуле:

$$C = \frac{-t}{Rm} \text{ и } V_0 = e^b.$$

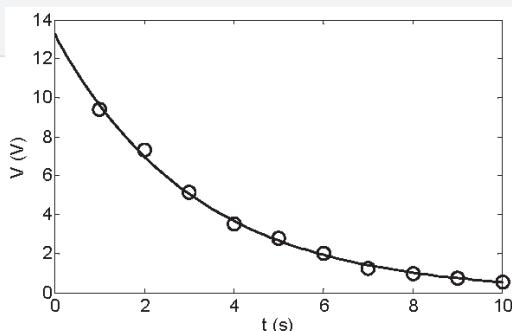
Следующая программа, записанная в скрипт-файле, определяет наилучшую показательную функцию, подобранную к точкам данных, определяет C и V_0 и графически изображает точки и подогнанную функцию.

```
R=2000;                                         Задание R.
t=1:10;                                         Присвоение точек данных векторам t и v.
v=[9.4 7.31 5.15 3.55 2.81 2.04 1.26 0.97 0.74 0.58];
p=polyfit(t,log(v),1);                         Использование polyfit c t и log(v).
C=-1/(R*p(1))                                  Вычисление C из p(1), которое есть m в уравнении.
V0=exp(p(2))                                    Вычисление V0 из p(2), которое есть b в уравнении.
tplot=0:0.1:10;                                 Создание векторов tplot времени и vplot
vplot=V0*exp(-tplot./(R*C));                  значений для построения графика функции.
plot(t,v,'o',tplot,vplot)
```

Когда выполняется скрипт-файл (сохраненный как `Chap8SamProb`), значения C и $V0$ выводятся на экран в командном окне как показано ниже:

```
>> Chap8SamProb
C =
    0.0016                                         Емкость конденсатора равна 1600 мкФ.
V0 =
    13.279
```

Эта программа создает следующий график (метки осей были добавлены к графику с использованием редактора графика):



Пример задачи 8.7. Зависимость вязкости от температуры

Вязкость μ – это свойство газов и жидкостей, которое характеризует их сопротивление движению. Для большинства материалов вязкость очень чувствительна к температуре. Ниже приведена таблица которая указывает вязкость нефти SAE 10W при различных температурах (данные из B. R. Munson, D. F. Young, and T. H. Okiishi, Fundamentals of Fluid Mechanics, 4th ed., John Wiley and Sons, 2002). Определите уравнение, которое может быть подобрано к этим данным.

T (°C)	-20	0	20	40	60	80	100	120
μ ((Н·сек/м²) ($\times 10^{-5}$))	4	0.38	0.095	0.032	0.015	0.0078	0.0045	0.0032

Решение

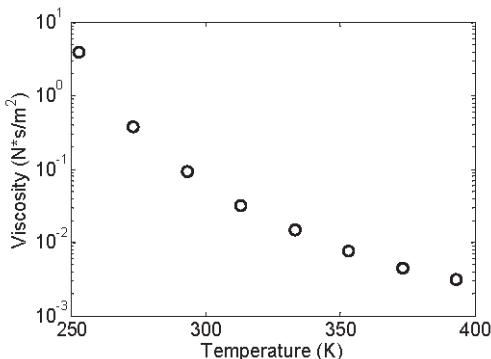
Чтобы определить, какое уравнение могло бы обеспечить хорошую подгонку к этим данным, изобразим графически μ как функцию T (абсолютная температура) с линейной шкалой для T и логарифмической шкалой для μ . График справа показывает, что точки данных, кажется, не выстраиваются в линию вдоль прямой. Это означает, что простая показательная функция вида $y = b e^{mx}$, которая моделирует прямую с такими осями, не будет обеспечивать лучшую подгонку. Так как точки в рисунке, кажется, простираются вдоль кривой линии, возможно, что хорошей подгонкой к данным будет функция следующего вида:

$$\ln(\mu) = a_2 T^2 + a_1 T + a_0.$$

Такая функция может быть подобрана к данным с использованием функции MATLAB `polyfit(x, y, 2)` (многочлен второй степени), где независимая переменная есть T , а зависимая переменная – $\ln(\mu)$. Указанное выше уравнение может быть решено относительно μ для получения вязкости как функции температуры:

$$\mu = e^{(a_2 T^2 + a_1 T + a_0)} = e^{a_0} e^{a_1 T} e^{a_2 T^2}.$$

Следующая программа определяет лучшую подгонку к функции и создает график, который выводит на экран точки данных и функцию.



```
T=[-20:20:120];
mu=[ 4 0.38 0.095 0.032 0.015 0.0078 0.0045 0.0032];
TK=T+273;
p=polyfit (TK, log (mu) , 2)
Tplot=273+[-20:120];
muplot = exp (p(1)*Tplot.^2 + p(2)*Tplot + p(3));
semilogy (TK,mu, 'o', Tplot,muplot)
```

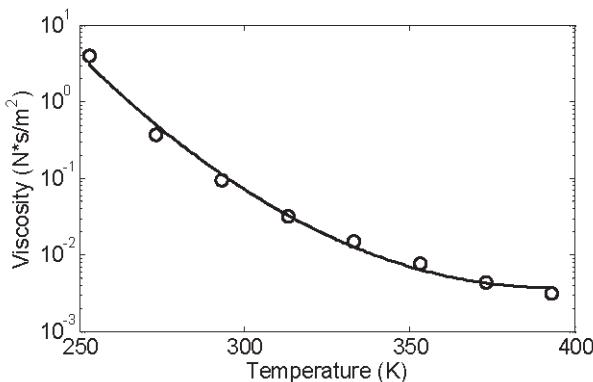
Когда эта программа выполняется (сохраненная как Chap8SamPro7), коэффициенты, которые определяются функцией `polyfit`, выводятся на экран в командном окне (показано ниже) как три элемента вектора `p`.

```
>> Chap8SamPro7
p =
    0.0003   -0.2685   47.1673
```

С этими коэффициентами вязкость нефти как функция температуры есть:

$$\mu = e^{(0.0003T^2 - 0.2685T + 47.1673)} = e^{47.1673} e^{-0.2685T} e^{0.0003T^2}.$$

Создаваемый график показывает, что это уравнение хорошо коррелирует к точкам данных (метки осей были добавлены редактором графика).



8.6. Задачи

- Построить график многочлена $y = 0.1x^5 - 0.2x^4 - x^3 + 5x^2 - 41.5x + 235$ на промежутке $-6 \leq x \leq 6$. Сначала создайте вектор для x , затем используйте функцию `polyval` для вычисления y и в конце используйте функцию `plot`.

2. Построить график многочлена $y = 0.008x^4 - 1.8x^2 - 5.4x + 54$ на промежутке $-14 \leq x \leq 16$. Сначала создайте вектор для x , затем используйте функцию `polyval` для вычисления y и в конце используйте функцию `plot`.

3. Используйте MATLAB для выполнения умножения следующих двух многочленов:

$$(-x^3 + 5x - 1)(x^4 + 2x^3 - 16x + 5).$$

4. Используйте MATLAB для выполнения умножения следующих двух многочленов:

$$x(x - 1.7)(x + 0.5)(x - 0.7)(x + 1.5).$$

Постройте график многочлена для $-1.6 \leq x \leq 1.8$.

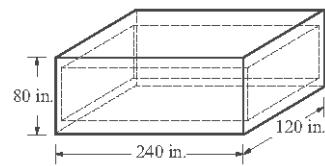
5. Разделите многочлен $-10x^6 - 20x^5 + 9x^4 + 10x^3 + 8x^2 + 11x - 3$ на многочлен $2x^2 + 4x - 1$.

6. Разделите многочлен $-0.24x^7 + 1.6x^6 + 1.5x^5 - 7.41x^4 - 1.8x^3 - 4x^2 - 75.2x - 91$ на многочлен $-0.8x^3 + 5x + 6.5$.

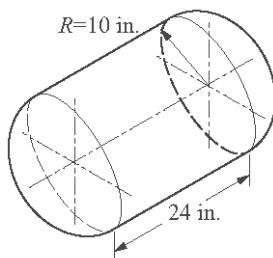
7. Произведение двух последовательных целых чисел равно 6 972. Используя встроенную функцию MATLAB для операций с многочленами, определите эти два целых числа.

8. Произведение трех целых чисел с интервалом 5 между ними (например, 9, 14, 19) равно 10 098. Используя встроенную функцию MATLAB для операций с многочленами, определите эти три целых числа.

9. Прямоугольный стальной контейнер имеет внешние размеры, показанные на рисунке (в дюймах). Толщина нижней и верхней граней равна t , а толщина боковых стенок равна $t/2$. Определите t , если вес контейнера составляет 12 212 фунтов. Удельный вес стали равен 0.284 фунт/дюйм³.

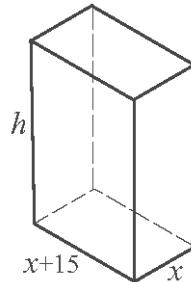


10. Алюминиевый топливный бак имеет цилиндрическую среднюю часть и полусферические боковые стороны. Внешний диаметр равен 10 дюймов, а длина цилиндрической части – 24 дюйма. Толщина стенок цилиндрической части равна t , а толщина стенок полусферических концов равна $1.5t$. Определите t , если вес корпуса составляет 42.27 фунтов. Удельный вес алюминия равен 0.101 фунт/дюйм³.



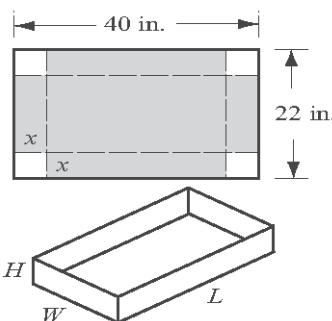
11. Прут длиной 20 футов разрезан на 12 частей, из которых сварен каркас прямоугольной коробки. Длина основания коробки на 15 дюймов длиннее его ширины.

- (a) Найдите полиномиальное выражение для объема V в терминах переменной x .
 (b) Создайте график V как функции от x .
 (c) Найдите значение x , которое максимизирует объем и определите этот объем.



12. Прямоугольный картонный лист длины 40 дюймов и ширины 22 дюйма используется для изготовления прямоугольной коробки (с открытым верхом), вырезанием угловых квадратов размером x -на- x и складыванием сторон.

- (a) Найдите полиномиальное выражение для объема V в терминах переменной x .
 (b) Создайте график V как функции от x .
 (c) Определите x , если объем коробки равен 1000 дюйм³.
 (d) Определите значение x , которое соответствует коробке самого большого объема, и определите этот объем.



13. Напишите пользовательскую функцию, которая складывает или вычитает два многочлена любого порядка. Имя этой функции `p=polyadd(p1,p2,operation)`. Первые два входных аргумента `p1` и `p2` являются векторами коэффициентов этих двух многочленов. (Если эти два многочлена не одного порядка, функция прибавляет необходимые нулевые элементы к более короткому вектору.) Третья входной аргумент `operation` есть строка, которая может быть или '`add`' или '`sub`', для сложения или вычитания многочленов, соответственно, а выходной аргумент – это получающийся многочлен.

Используйте эту функцию для сложения и вычитания следующих многочленов:

$$f_1(x) = 2x^6 - 3x^4 - 9x^3 + 11x^2 - 8x + 4 \quad \text{и} \quad f_2(x) = 5x^3 + 7x - 10.$$

14. Напишите пользовательскую функцию, которая перемножает два многочлена. Имя этой функции `p=polymult(p1,p2)`. Два входных аргумента `p1` и `p2` являются векторами коэффициентов этих двух многочленов. Выходной аргумент `p` – это получающийся многочлен.

Используйте эту функцию для умножения следующих многочленов:

$$f_1(x) = 2x^6 - 3x^4 - 9x^3 + 11x^2 - 8x + 4 \quad \text{и} \quad f_2(x) = 5x^3 + 7x - 10.$$

Проверьте ответ встроенной функцией MATLAB `conv`.

15. Напишите пользовательскую функцию, которая вычисляет максимум (или минимум) квадратного трехчлена вида:

$$f(x) = ax^2 + bx + c.$$

Для имени функции и аргументов используйте `[x, y, w] = maxormin(a, b, c)`. Входные аргументы – это коэффициенты a , b и c . Выходной аргумент x – координата максимума (или минимума); y – максимальное (или минимальное) значение; w принимает значение 1 в случае максимума и значение 2 – в случае минимума.

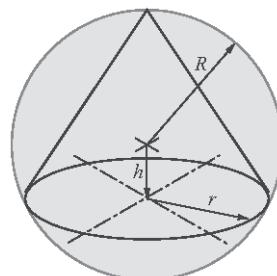
Используйте эту функцию для нахождения максимума или минимума следующих функций:

$$(a) f(x) = 3x^2 - 7x + 14;$$

$$(b) f(x) = -5x^2 - 11x + 15.$$

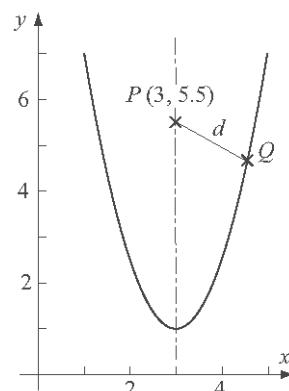
16. Конус с радиусом основания r вписан в сферу, как показано на рисунке. Радиус сферы $R = 9$ дюймов.

- (a) Создайте полиномиальное выражение для объема V конуса от переменной h .
 (b) Создайте график V как функции от h для $9 \leq h \leq 9$.
 (c) Используя команду `roots` определите h , если объем конуса равен 500 дюйм³.
 (d) Определите значение h , который соответствует конусу с наибольшим объемом и найдите этот объем.



17. Рассмотрим параболу $y = 1.5(x - 3)^2 + 1$ и точку $P(3, 5.5)$

- (a) Создайте полиномиальное выражение для расстояния d от точки P до произвольной точки Q на параболе.
 (b) Создайте график d как функции от x для $3 \leq x \leq 6$.
 (c) Найдите координаты точки Q если $d = 28$.
 (d) Определите координаты точек Q , которые соответствуют наименьшему d и вычислите соответствующее значение d .



18. Даны следующие данные:

x	2	5	6	8	9	13	15
y	7	8	10	11	12	14	15

- (a) Используйте линейную регрессию наименьших квадратов для определения коэффициентов m и b функции $y = mx + b$, которая наилучшим образом приближает данные.
 (b) Создайте график, который показывает функцию и точки данных.

- 19.** Температура кипения воды T_B на различных высотах h дана в следующей таблице. Определите линейное уравнение в виде $T_B = mh + b$, которое наилучшим образом приближает данные. Используйте это уравнение для того, чтобы вычислить температуру кипения на высоте 5 000 м. Создайте график точек и уравнения.

$h(\text{м})$	0	600	1500	2300	3000	6100	7900
$T(\text{°C})$	100	98.8	95.1	92.2	90	81.2	75.6

- 20.** Население США в указанные годы между 1815 и 1965 представлено в таблице ниже. Определите квадратичное выражение вида $P = a_2t^2 + a_1t + a_0$, где t – число лет после 1800, и P – население в миллионах, которое лучше всего соответствует данным. Используйте это уравнение для оценки населения в 1915 г. (тогда население составляло 98.8 миллионов). Создайте график населения как функции от года, который показывает точки данных и это уравнение.

Год	1815	1845	1875	1905	1935	1965
Население (миллионов)	8.3	19.7	44.4	83.2	127.1	190.9

- 21.** Число бактерий N_B , измеренное в разные моменты времени t , дано в следующей таблице. Определите показательную функцию вида $N_B = Ne^{at}$, которая наилучшим образом приближает данные. Используйте это уравнение, чтобы оценить число бактерий после 4.5 часов. Сделайте график точек и уравнения.

t (час)	1	2	3	4	5	6
N_B	2,000	4,500	7,500	15,000	31,000	64,000

- 22.** Данные роста подсолнечника на плантации представлены в следующей таблице:

Неделя	1	3	5	7	9	11	13
Высота (см)	22	51	127	202	227	248	252

Эти данные могут быть смоделированы функцией вида $H = C/(1 + Ae^{-Bt})$ (логистическое уравнение), где H – высота, C – максимальное значение для H , A и B – константы и t – число недель. Используя метод, описанный в разделе 8.2.2, и считая, что $C = 254$ см, определите константы A и B так, чтобы функция лучше всего соответствовала данным. Используйте эту функцию для оценки высоты на шестой неделе. На одном рисунке графически изобразите функцию и точки данных.

23. Используйте данные роста из задачи 22 для следующего:

(a) Подгонки данных многочленом третьего порядка. Используйте этот многочлен для оценки высоты на 6 неделе.

(b) Используйте для подгонки данных линейную и сплайновую интерполяцию и затем отдельно оцените высоту на 6 неделе.

Для каждого пункта создайте график точек данных (маркеры круга) и подогнанную кривую или интерполяционные кривые. Отметим, что в пункте (b) есть две кривые интерполяции.

24. Даны следующие данные:

x	1	2.2	3.7	6.4	9	11.5	14.2	17.8	20.5	23.2
y	12	9	6.6	5.5	7.2	9.2	9.6	8.5	6.5	2.2

(a) Приблизьте данные многочленом первого порядка. Создайте график точек и многочлена.

(b) Приблизьте данные многочленом второго порядка. Создайте график точек и многочлена.

(c) Приблизьте данные многочленом третьего порядка. Создайте график точек и многочлена.

(d) Приблизьте данные многочленом пятого порядка. Создайте график точек и многочлена.

25. Стандартная плотность воздуха, D (среднее из сделанных измерений), на различных высотах h от уровня моря до высоты 33 км представлена ниже.

h (км)	0	3	6	9	12	15
D (кг/м³)	1.2	0.91	0.66	0.47	0.31	0.19

h (км)	18	21	24	27	30	33
D (кг/м³)	0.12	0.075	0.046	0.029	0.018	0.011

(a) Сделайте следующие четыре графика точек данных (плотность как функция высоты): (1) обе оси с линейной шкалой; (2) логарифмическая ось h , линейная ось D ; (3) линейная ось h , логарифмическая ось D ; (4) обе оси логарифмические. Согласно графикам выберите функцию (линейный, степенный, показательную, или логарифмическую функцию), которая лучше подходит к точкам данных и определите коэффициенты функции.

(b) Графически изобразите функции и точки, используя линейные оси.

26. Напишите пользовательскую функцию, которая выполняет подгонку к точкам данных степенной функцией вида $y = bx^m$. Имя функции [`b, m`] = `powerfit`(`x, y`) ,

где входные аргументы x и y – это с координатами точек данных, а выходные аргументы b и m – константы подбираемого степенного выражения. Используйте эту функцию `powerfit` для подгонки к представленным ниже данным. Создайте график, который показывает точки данных и функцию.

x	0.5	2.4	3.2	4.9	6.5	7.8
y	0.8	9.3	37.9	68.2	155	198

- 27.** Вязкость – это свойство газов и жидкостей, которое характеризует их сопротивление потоку. Для большинства материалов вязкость очень чувствительна к температуре. Для газов изменение вязкости при изменении температуры часто моделируется уравнением вида

$$\mu = \frac{CT^{3/2}}{T + S},$$

где μ – вязкость, T – абсолютная температура, а C и S – эмпирические константы. Ниже приведена таблица, которая дает вязкость воздуха при различных температурах (данные из B. R. Munson, D. F. Young, and T. H. Okiishi, *Fundamentals of Fluid Mechanics*, 4th ed., John Wiley and Sons, 2002).

T (°C)	-20	0	40	100	200	300	400	500	1000
μ ($H \cdot c/m^2$) ($\times 10^{-5}$)	1.63	1.71	1.87	2.17	2.53	2.98	3.32	3.64	5.04

Определите константы C и S подбором уравнения кривой к точкам данных. Создайте график вязкости как функции температуры (в °C). На графике покажите точки данных маркерами, а эмпирическое уравнение кривой – сплошной линией.

Подбор кривой может быть выполнен, если переписать уравнение в виде

$$\frac{T^{3/2}}{\mu} = \frac{1}{C} T + \frac{S}{C}$$

и использовать многочлен первого порядка.

- 28.** Измерения топливной экономичности F_E автомобиля (в милях на галлон) на различных скоростях v приведены в таблице.

v (миль/ч)	5	15	25	35	45	55	65	75
F_E (миль/галлон)	11	22	28	29.5	30	30	27	23

- (a) Выполните подгонку данных многочленом второго порядка. Используйте многочлен для оценки топливной экономичности при 60 миль/ч. Создайте график точек и многочлена.

- (b) Выполните подгонку данных многочленом третьего порядка. Используйте многочлен для оценки топливной экономичности при 60 миль/час. Создайте график точек и многочлена.
- (c) Выполните подгонку данных линейной и сплайновой интерполяциями. Оцените топливную экономичность при 60 миль/час для линейной и сплайновой интерполяций. Создайте график, который показывает точки данных и интерполяционные кривые.

29. Известно соотношение между двумя переменными P и t :

$$P = \frac{mt}{b+t}.$$

Даны следующие частные значения

t	1	3	4	7	8	10
P	2.1	4.6	5.4	6.1	6.4	6.6

Определите константы m и b подбором уравнения кривой к данным точкам. Создайте график P как функции от t . На графике отметьте точки данных маркерами, а эмпирическое уравнение кривой – сплошной линией. (Подбор кривой может быть сделан если взять обратные величины переменных уравнения и использовать многочлен первого порядка.)

30. Когда резина растягивается, ее удлинение сначала пропорционально приложенной силе, но когда ее длина становится в 2 раза больше первоначальной длины, требуемая для дальнейшего растяжения сила увеличения быстро. Сила, как функция удлинения, которая потребовалась для растяжения резинового экземпляра первоначальной длины 3 дюйма, показана в следующей таблице.

- (a) Выполните подгонку данных многочленом четвертого порядка. Создайте график точек и многочлена. Используйте этот многочлен для оценки силы, необходимой для растяжения резинового экземпляра на 11.5 дюймов.
- (b) Выполните подгонку данных сплайновой интерполяцией (используйте встроенную функцию MATLAB `interp1`). Создайте график, который показывает точки данных и интерполяционную кривую. Используйте эту интерполяцию для оценки силы, необходимой для растяжения резинового экземпляра на 11.5 дюймов.

Сила (фунт)	0	0.6	0.9	1.16	1.18	1.19	1.24	1.48
Удлинение (дюйм)	0	1.2	2.4	3.6	4.8	6.0	7.2	8.4
Сила (фунт)	1.92	3.12	4.14	5.34	6.22	7.12	7.86	8.42
Удлинение (дюйм)	9.6	10.8	12.0	13.2	14.4	15.6	16.8	18

- 31.** Напряжение текучести, σ_y , многих металлов зависит от размера зернистости. Для этих металлов, соотношения между напряжением текучести и диаметром d средней зернистости может быть смоделировано уравнением Холла-Печа:

$$\sigma_y = \sigma_0 + kd^{-1/2}.$$

Ниже приведены результаты измерений среднего диаметра зернистости и напряжения текучести.

d (мм)	0.005	0.009	0.016	0.025	0.040	0.062	0.085	0.110
σ_y (мПа)	205	150	135	97	89	80	70	67

- (a) Используя подборку кривой, определите константы σ_0 и k в уравнении Холла-Печа для этого материала. Используя эти константы определите из уравнения напряжение текучести материала с размером зерен 0.05 мм. Создайте график, который показывает точки данных с маркерами круга и сплошной линией кривую, полученную из уравнения Холла-Печа.
- (b) Используйте линейную интерполяцию, чтобы определить напряжение текучести материала с размером зерен 0.05 мм. Создайте график, который показывает точки данных с маркерами круга и сплошной линией линейную интерполяцию.
- (c) Используйте кубическую интерполяцию для определения напряжение текучести материала с размером зерен 0.05 мм. Создайте график, который показывает точки данных с маркерами круга и сплошной линией кубическую интерполяцию.

- 32.** Передача света через прозрачное тело может быть описана уравнение:

$$I_T = I_0(1-R)^2 e^{-\beta L}.$$

где I_T – переданная интенсивность, I_0 – интенсивность падающего луча, β – коэффициент поглощения, L – толщина прозрачного тела и R – часть света, которая отражается от поверхности. Если свет ортогонален к поверхности и лучи передаются через воздух, то

$$R = \left(\frac{n-1}{n+1} \right)^2,$$

где n – показатель преломления прозрачного тела. Экспериментальные измерения интенсивности света, пропущенного через экземпляры прозрачного тела различной толщины, приведены в следующей таблице. Интенсивность падающего луча равна 5 ватт/м².

L (см)	0.5	1.2	1.7	2.2	4.5	6.0
I_T (ватт/м²)	4.2	4.0	3.8	3.6	2.9	2.5

кривой для определения коэффициента поглощения и показателя преломления тела.

33. Уравнение идеального газа связывает объем, давление, температуру и количество газа:

$$V = \frac{nRT}{P}.$$

где V – объем в литрах, P – давление в атмосферах (атм), T – температура в Кельвинах, n – число молей и R – газовая постоянная.

Для определения значения газовой постоянной R проводился эксперимент. В этом эксперименте 0.05 молей газа сжимались к различным объемам применением давления к газу. При каждом объеме регистрировалось давление и температура газа. Используя данные, представленные ниже, определите R построением графика V как функции от T/P и подгоняя точки данных линейным уравнением.

V (л)	0.75	0.65	0.55	0.45	0.35
T (°C)	25	37	45	56	65
P (атм)	1.63	1.96	2.37	3.00	3.96



ГЛАВА 9.

Приложения в численном анализе

Численные методы обычно используются для решения математических задач, которые возникают в науке и технике, где трудно или невозможно получить точные решения. MATLAB имеет большую библиотеку функций для численного решения разнообразных математических задач. Эта глава представляет ряд наиболее часто используемых таких функций. Отметим, что цель этой книги состоит в том, чтобы показать пользователям, как использовать MATLAB. Здесь дается общая информация о численных методах, но детали не включены – они могут быть найдены в книгах по численному анализу.

В этой главе представлены следующие темы: решение уравнения с одним неизвестным, нахождение минимума или максимума функции, численное интегрирование и решение дифференциального уравнения первого порядка.

9.1. Решение уравнения с одной переменной

Уравнение с одной переменной может быть написано в виде $f(x) = 0$. Решение уравнения (также называемое корнем) есть численное значение x , которое удовлетворяет уравнению этому. Графически, решение – это точка, где функция $f(x)$ пересекает или касается оси x . Точное решение – это значение x , для которого значение функции является точно нулевым. Если такое значение не существует, или его трудно определить, может быть найдено численное решение, это значение x , которое очень близко к точному решению. Это обычно делается итерационным процессом, где после каждой итерации компьютер находит значение x , которое ближе к решению. Итерации останавливаются, когда разность между двумя итерациями x меньше некоторой меры. Вообще, у функции может быть ноль, одно, несколько или бесконечное число решений.

В MATLAB нуль функции может быть определен командой (встроенная функция) `fzero` вида:



Встроенная функция `fzero` является функцией от функции MATLAB (см. раздел 7.9), что означает, что она принимает как входной аргумент другую функцию (функция, которая будет решаться).

Дополнительные детали об аргументах `fzero`

- x – это решение, являющееся скаляром.
- `function` – функция, которая будет решаться. Она может быть введена несколькими различными способами:
 1. Самый простой способ состоит в том, чтобы ввести математическое выражение как строку (string).
 2. Функция создается как пользовательская функция в файле функции и затем вводится дескриптор функции (см. раздел 7.9.1).
 3. Функция создается как анонимная функция (см. раздел 7.8.1) и затем вводится имя анонимной функции (которое является именем дескриптора) (см. раздел 7.9.1).

(Как объяснено в разделе 7.9.2, также можно передать пользовательскую функцию и действующую функцию в функцию от функции с использованием ее имени. Однако, дескрипторы функции более эффективны и легче использовать и должны быть предпочтительным методом.)

- Функция должна быть написана в стандартной форме. Например, если будет решаться функция $xe^{-x} = 0.2$, то она должна быть написана в виде $f(x) = xe^{-x} - 0.2 = 0$. Если эта функция вводится в команду `fzero` как строка, то печатается как: '`x*exp (-x) - 0.2`'.
- Когда функция вводится как выражение (строка string), она не может включать предопределенные переменные. Например, нельзя определить `b=0.2` и затем ввести '`x*exp (-x) - b`'.
- $x0$ может быть скаляром или вектором с двумя элементами. Если этот аргумент введен как скаляр, то он должен быть значением x около точки, где функция пересекает (или касается) ось x . Если $x0$ вводится как вектор, то эти два элемента должны быть точками с разных сторон от решения. Если пересечения ось X , то имеет различный знак чем. Если $x0$ введен как вектор, эти два элемента должны быть точками на противоположных сторонах решения. Если $f(x)$ пересекает ось x , то $f(x0(1))$ и $f(x0(2))$ имеют противоположные знаки. Когда у функции есть более одного решения, то каждое решение может быть определено отдельно с использованием функции `fzero` и вводом значения для $x0$ около каждого из решений.

- Хороший способ найти первое приближение x_0 к решению функции, стоит в том, чтобы создать график функции. Таким образом во многих приложениях в науке и технике может быть оценена область решения. Часто, когда у функции есть более одного решения, только одно из решений будет иметь физический смысл.

Пример задачи 9.1. Решение нелинейного уравнения

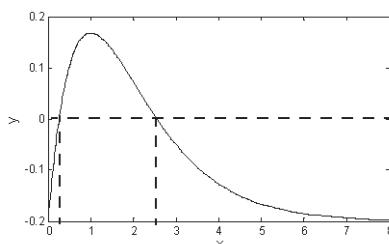
Определить решение уравнения $xe^{-x} = 0.2$.

Решение

Сначала записываем уравнение в виде функции: $f(x) = xe^{-x} - 0.2$. График функции, показанной справа, показывает, что у функции есть одно решение между 0 и 1 и другое решение – между 2 и 3. График получается вводом строки

```
>> fplot('x*exp(-x)-0.2', [0 8])
```

в командном окне. Решения функции находятся с использованием дважды команды `fzero`. Сначала уравнение вводится как строка и используется значение x_0 между 0 и 1 ($x_0 = 0.7$). Второй раз уравнение, которое будет решаться, записано как анонимная функция, которая потом используется в `fzero` со значением x_0 между 2 и 3 ($x_0 = 2.8$). Это показано ниже:



```
>> x1=fzero('x*exp(-x)-0.2', 0.7)          Функция вводится как строка
x1 =
0.2592                                         Первое решение есть 0.2592.
>> F=@(x)x*exp(-x)-0.2                      Создание анонимной функции.
F =
@(x)x*exp(-x)-0.2                           Использование имени анонимной функции в fzero.
>> fzero(F,2.8)
ans =
2.5426                                         Второе решение есть 2.5426.
```

Дополнительные комментарии

- Команда `fzero` обнаруживает нули функции только там, где функция пересекает ось x . Команда не обнаруживает нуль в точках, где функция касается, но не пересекает ось x .
- Если решение не может быть определено, переменной x присваивается значение `NaN`.
- У команды `fzero` есть дополнительные опции (см. окно справки). Две из наиболее важных опций:

`[x fval]=fzero(function, x0)` – присваивает переменной `fval` значение функции в точке `x`.

`x=fzero(function, x0, optimset('display','iter'))` – выводит на экран результат каждой итерации во время процесса нахождения решения.

- Если функция может быть записана в форме многочлена, решение, или корни, могут быть найдены командой `roots`, как объяснено в главе 8 (раздел 8.1.2).
- Команда `fzero` может также использоваться для нахождения значения x , где функции принимает определенное значение. Это делается переносом функции вверх или вниз. Например, в функции примера типовой задачи 9.1 первое значение x , где функция равна 0.1, может быть определено решением уравнения $xe^{-x} - 0.3 = 0$. Это показано ниже:

```
>> x=fzero('x*exp(-x)-0.3', 0.5)
x =
    0.4894
```

9.2. Нахождение минимума или максимума функции

Во многих приложениях есть необходимость определить локальный минимум или максимум функции вида $y = f(x)$. В математическом анализе значение x , которое соответствует локальному минимуму или максимуму, определяется нахождением нулей производной функции. Значение y определяется подстановкой этого x в функцию. В MATLAB значение x , где функция одной переменной имеет минимум на заданном интервале $x_1 \leq x \leq x_2$, может быть найдено командой `fminbnd`, формат которой следующий:

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

Значение x , где функция имеет минимум Функция Интервал переменной x

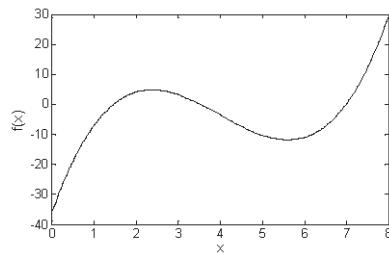
- Аргумент `function` может быть введен как строка, или как дескриптор функции, таким же образом как в команде `fzero`. См. раздел 9.1 для деталей.
- Значение функции в точке минимума может быть добавлено к результату при использовании опции

`[x fval]=fminbnd(function,x1,x2),`

где значение функции в точке `x` присвоено переменной `fval`.

- В пределах заданного интервала минимум функции может быть в одной из конечных точек интервала или в точке внутри интервала где нулевой наклон функции (локальный минимум). Когда выполняется команда `fminbnd`, MATLAB ищет локальный минимум. Когда найден локальный минимум, его значение сравнивается со значениями функции в конечных точках интервала. MATLAB возвращает точку с фактическим минимальным значением функции на промежутке.

Например, рассмотрим функцию $f(x) = x^3 - 12x^2 + 40.25x - 36.5$, график которой на интервале $0 \leq x \leq 8$ показан на рисунке справа. Можно заметить, что есть локальный минимум между 5 и 6, и что абсолютный минимум она имеет в точке $x = 0$. Используем команду `fminbnd` с интервалом $3 \leq x \leq 8$ для нахождения точки локального минимума и значения функции в этой точке:



```
>> [x fval]=fminbnd('x^3-12*x^2+40.25*x-36.5', 3, 8)
x =
      5.6073
fval =
     -11.8043
```

Локальный минимум в точке $x = 5.6073$.
Значение функции в этой точке = -11.8043 .

Заметим, что команда `fminbnd` дает локальный минимум. Если интервал изменить на $0 \leq x \leq 8$, то `fminbnd` дает:

```
>> [x fval]=fminbnd('x^3-12*x^2+40.25*x-36.5', 0, 8)
x =
      0
fval =
     -36.5000
```

Минимум в точке $x = 0$.
Значение функции в этой точке = -36.5000 .

Для этого интервала команда `fminbnd` дает абсолютный минимум, который достигается в конечной точке $x = 0$.

- Команда `fminbnd` может также использоваться для нахождения максимума функции. Это делается умножением функции на -1 и нахождением минимума. Например, максимум функции $f(x) = xe^{-x} - 0.2$ (из примера типовой задачи 9.1) в интервале $0 \leq x \leq 8$ может быть определен нахождением точки минимума функции $f(x) = -xe^{-x} + 0.2$, как показано ниже:

```
>> [x fval]=fminbnd('-x*exp(-x)+0.2', 0, 8)
x =
      1.0000
fval =
     -0.1679
```

Максимум в точке $x = 0$.
Значение функции в этой точке = 0.1679 .

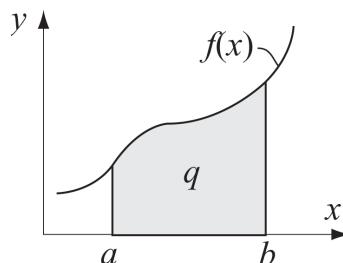
9.3. Численное интегрирование

Интегрирование – это распространенная математическая операция в науке и технике. Вычисление площади и объема, скорости из ускорения и работы из силы и перемещения – только несколько примеров, где используются интегралы. Интегрирование простых функций может быть сделано аналитически, но более сложные функции часто трудно или невозможно проинтегрировать аналитически. В курсах математического анализа подынтегральное выражение (величина, которая будет интегрироваться), обычно является функцией. В приложениях науки и техники подынтегральная функция может быть функцией или рядом точек данных. Например, для вычисления объема могут использоваться точки данных дискретных измерений скорости потока.

Мы предполагаем, что читатель знаком с основами интегралов и интегрирования. Определенный интеграл функции $f(x)$ от a до b записывается в виде:

$$q = \int_a^b f(x) dx.$$

Функция $f(x)$ называется подынтегральной, а числа a и b – пределы интегрирования. Графически, значение интеграла q является площадью между графиком функции, осью x , и пределами a и b (затененная область на рисунке). Когда определенный интеграл вычисляется аналитически, $f(x)$ всегда функция. Когда интеграл вычисляется численно, $f(x)$ может быть как функцией, так и рядом точек. При численном интегрировании общая площадь фигуры разбивается на небольшие части, затем вычисляется площадь каждой маленькой части и все они складываются. Для этой цели интегрирования были разработаны различные численные методы. Эти методы различаются способами, которыми область разбивается на кусочки и методами вычисления площади каждого кусочка. Книги по численному анализу содержат детали численных методов.



Следующее обсуждение показывает, как использовать три встроенные функции интегрирования MATLAB `quad`, `quadl` и `trapz`. Команды `quad` и `quadl` используются для интегрирования, когда $f(x)$ есть функция, а `trapz` используется, когда $f(x)$ задана точками данных.

Команда `quad`

Формат команды `quad`, которая использует для интегрирования адаптивный метод Симпсона:

`q = quad(function, a, b)`

Значение интеграла.
Функция для интегрирования.
Пределы интегрирования.

- Функция может быть введена как выражение строки или как дескриптор функции, таким же образом как в команде `fzero`. См. раздел 9.1 для деталей. Первые два метода демонстрируются в примере типовой задачи 9.2.
- Функция должна быть записана для аргумента x , который является вектором (используя поэлементные операции) так, что она вычисляет значение функции для каждого элемента x .
- Пользователь должен удостовериться, что у функции нет вертикальной асимптоты между a и b .
- `quad` вычисляет интеграл с абсолютной погрешностью, которая меньше чем $1.0e-6$. Это число может быть изменено добавление дополнительного аргумента `tol` в команде:

```
q = quad('function', a, b, tol)
```

tol – это число, которое определяет предельную ошибку. С большим tol интеграл вычисляется менее точно, но быстрее.

Команда `quadl`

Формат команды `quadl` (последний символ – это строчная L) точно такой же, что и к команды `quad`:

$q = \text{quadl}(\text{function}, a, b)$

Значение интеграла.
Функция для интегрирования.
Пределы интегрирования.

Все комментарии, которые перечислены для команды `quad`, справедливы для команды `quadl`. Отличие между этими двумя командами – это численный метод, используемый для интегрирования. Команда `quadl` использует аддитивный метод Лобатто, который может быть более эффективным для достижения высокой точности при интегрировании гладких функций.

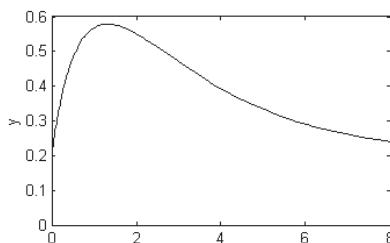
Пример задачи 9.2. Численное интегрирование функций

Использовать численное интегрирование для вычисления следующего интеграла:

$$\int_0^8 (xe^{-x^{0.8}} + 0.2) dx.$$

Решение

Для иллюстрации, справа показан график подынтегральной функции на интервале $0 \leq x \leq 8$. Решение использует команду `quad` и показывает, как ввести функцию в эту команду двумя способами. В первом случае она вводится непосредственно,



вводом выражения функции как аргумента. Во втором случае создается анонимная функция и ее имя затем вводится в команду.

Ниже показано использование команды quad в командном окне, когда подынтегральная функция вводится как строка. Заметим, что функция вводится с поэлементными операциями.

```
>> quad('x.*exp(-x.^0.8)+0.2', 0, 8)
ans =
    3.1604
```

Для второго метода нужно сначала создать пользовательскую функцию, которая вычисляет подынтегральную функцию. Файл функции (названный `y=Chap9Sam2 (x)`):

```
function y=Chap9Sam2(x)
y=x.*exp(-x.^0.8)+0.2;
```

Отметим снова, что функция записывается с поэлементными операциями так, что аргумент `x` может быть вектором. Теперь интегрирование выполняется в командном окне, вводом дескриптора `@Chap9Sam2` в качестве аргумента в команде quad, как показано ниже:

```
>> q=quad(@Chap9Sam2, 0, 8)
q =
    3.1604
```

Команда trapz

Команда `trapz` может использоваться для того, чтобы интегрировать функцию, которая задана точками данных. Она использует для численного интегрирования метод трапеций. Формат команды:

```
q = trapz(x, y)
```

где `x` и `y` – это векторы с `x` и `y` координатами точек, соответственно. Эти два вектора должны иметь одну и ту же длину.

9.4. Обыкновенные дифференциальные уравнения

Дифференциальные уравнения играют важную роль в науке и технике, так как они находятся в основе фактически каждого физического явления, которое используется в технических приложениях. Аналитически может быть решено только-

ко ограниченное число дифференциальных уравнений. С другой стороны, численные методы могут давать приближенные решения почти любого уравнения. Однако, получение численного решения может быть непростой задачей. Это потому, что не существует численного метода, который может решить любое уравнение. Вместо этого есть много методов, которые подходят для решения различных типов уравнений. В MATLAB есть большая библиотека инструментов, которые могут использоваться для решения дифференциальных уравнений. Однако для использования полной мощи MATLAB от пользователя требуется знание дифференциальных уравнений и различных численных методов, которые могут использоваться для их решения.

Этот раздел описывает подробно, как использовать MATLAB для решения обыкновенного дифференциального уравнения первого порядка. Возможные численные методы, которые могут использоваться для решения такого уравнения, описаны в общих чертах и не объяснены подробно с математической точки зрения. Этот раздел предоставляет информацию для решения простых, «непроблематичных» уравнения первого порядка. Такое решение дает основу для решения уравнения высокого порядка и системы уравнений.

Обыкновенное дифференциальное уравнение (ОДУ) является уравнением, которое содержит независимую переменную, зависимую переменную и производную зависимой переменной. Уравнения, которые рассматриваются здесь, имеют первый порядок вида

$$\frac{dy}{dx} = f(x, y).$$

где x и y – независимая и зависимая переменные, соответственно. Решение – это функция $y = f(x)$, которая удовлетворяет уравнению. Вообще говоря, много функций могут удовлетворить данному ОДУ и требуется дополнительная информация для определения решения определенной задачи. Эта дополнительная информация есть значение функции (зависимой переменной) при некотором значении независимой переменной.

Этапы решения простого ОДУ первого порядка

Далее в этом параграфе независимая переменная будет не x , а t (время). Это делается потому, что во многих приложениях независимая переменная – это время, а также еще и для того, чтобы не противоречить информации в меню помощи MATLAB.

Этап 1: Записать задачу в стандартной форме

Записываем уравнение в виде:

$$\frac{dy}{dx} = f(x, y) \text{ для } t_0 \leq t \leq t_f \text{ с условием } y = y_0 \text{ в точке } t = t_0.$$

Как указано выше, для решения ОДУ первого порядка, необходимы три части информации: само уравнение, которое дает выражение для производной y отно-

сительно t , интервал независимой переменной и начальное значение y . Решение – это значение y как функции от t между t_0 и t_f .

Пример задачи для решения:

$$\frac{dy}{dx} = \frac{t^3 - 2y}{t} \quad \text{для } 1 \leq t \leq 3 \text{ с условием } y = 4.2 \text{ в точке } t = 1.$$

Этап 2: Создание пользовательской функции (в файле функции) или анонимной функции

Дифференциальное уравнение, которое будет решаться, должно быть записано как пользовательская функция (в файле функции) или как анонимная функция. Обе они вычисляют $\frac{dy}{dx}$ для данных значений t и y . Для задачи приведенного выше примера, пользовательская функция (которая сохраняется как отдельный файл) есть:

```
function dydt=ODEexpr1(t,y)
dydt=(t^3-2*y)/t
```

окне, или быть в пределах скрипта-файла. Для задачи приведенного выше примера анонимная функция (названная `ode1`) есть:

```
>> ode1=@(t,y) (t^3-2*y)/t
ode1 =
    @(t,y) (t^3-2*y)/t
```

Этап 3: Выбор метода решения

Выбираем желаемый численный метод, который MATLAB будет использовать в решении. Для решения ОДУ первого порядка разработано много численных методов, из них несколько методов доступны как встроенные функции в MATLAB. В типичном численном методе временной интервал разбивается на небольшие промежутки. Решение начинается в известной точке y_0 , и затем, используя один из методов интегрирования, значение y вычисляется последовательно для каждого промежутка времени. Табл. 9.1 представляет список семи команд решателей ОДУ, которые являются встроенными функциями MATLAB, и которые могут использоваться для решения ОДУ первого порядка. Краткое описание каждого решателя включено в таблицу.

Таблица 9.1. Решатели ОДУ MATLAB

Имя решателя ОДУ	Описание
<code>ode45</code>	Для нежестких задач, одношаговый решатель, лучше всего подходит в качестве первой попытки применения для большинства задач. Основан на явном методе Рунге-Кutta.



Имя решателя ОДУ	Описание
ode23	Для нежестких задач, одношаговый решатель. Основан на явном методе Рунге-Кутта. Часто более быстрый, но менее точный чем <code>ode45</code> .
ode113	Для нежестких задач, многошаговый решатель.
ode15s	Для жестких задач, многошаговый решатель. Используется, если <code>ode45</code> не работает. Использует метод переменного порядка.
ode23s	Для жестких задач, одношаговый решатель. Может решить некоторые задачи, которые не может <code>ode15s</code> .
ode23t	Для умеренно жестких задач.
ode23tb	Для жестких задач. Часто более эффективный чем <code>ode15s</code> .

Вообще, решатели делятся на две группы в соответствии с их возможностями решать жесткие задачи и в соответствии с использованием одношаговых или многошаговых методов. Жесткие задачи – это такие, которые включают быстро и медленно изменяющиеся компоненты и требуют маленьких промежутков разбиения для их решения. Одношаговые решатели используют информацию от одной точки, чтобы получить решение на следующем шаге. Многошаговые решатели используют информацию от нескольких предыдущих точек, чтобы получить решение на следующем шаге. Детальная информация о различных методах выходит за рамки этой книги.

Невозможно знать заранее, какой решатель является самым подходящим для определенной задачи. Мы полагаем, что нужно сначала попробовать `ode45`, который дает хорошие результаты для многих задач. Если решение не получено, потому что задача жесткая, можно попробовать решатель `ode15s`.

Этап 4: Решение ОДУ

Формат команды, которая используется для решения задачи ОДУ с начальным условием, является одним и тем же для всех решателей и для всех уравнений, которые решаются. Вид команды:

```
[t,y] = solver_name(ODEfun,tspan,y0)
```

Дополнительная информация:

- | | |
|--------------------------|---|
| <code>solver_name</code> | Имя решателя (численного метода), который используется (например, <code>ode45</code> или <code>ode23s</code>). |
| <code>ODEfun</code> | Функция из этапа 2, который вычисляет dy/dt для данных значений t и y . Если она была записана как пользовательская функция, вводится дескриптор функции. Если она была запи- |

сана как анонимная функция, вводится имя анонимной функции. (См. пример, который следует далее.)

`tspan`

Вектор, который определяет промежуток решения. Вектор должен иметь по крайней мере два элемента, но может иметь и больше. Если этот вектор имеет только два элемента, они должны быть $[t_0 \ t_f]$, которые являются начальным и конечным значениями промежутка решения. У вектора `tspan` могут быть, однако, дополнительные точки между начальной и конечной точками. Число элементов в `tspan` влияет на результат этой команды. См. `[t, y]` ниже.

`y0`

Начальное значение y (значение y в первой точке интервала).

`[t, y]`

Результат, который является решением ОДУ. t и y – векторы столбцы. Первая и последняя точки – это начальная и конечная точки интервала. Шаг и число промежуточных точек зависят от входного вектора `tspan`. Если y `tspan` есть два элемента (начальная и конечная точки), векторы t и y содержат решение на каждом шаге интегрирования, вычисленное решателем.

Если `tspan` имеет более двух точек (дополнительные точки между первой и последней), тогда векторы t и y содержат решение только в этих точках. Число точек в `tspan` не влияет на шаги, используемые программой для решения.

Например, рассмотрим решение задачи поставленной на этапе 1:

$$\frac{dy}{dx} = \frac{t^3 - 2y}{t} \text{ для } 1 \leq t \leq 3 \text{ с условием } y = 4.2 \text{ в точке } t = 1.$$

Если функция ОДУ записана как пользовательская функция (см. этап 2), то решение со встроенной функцией MATLAB `ode45` получается так:

```
>> [t y]=ode45(@ODEexp1, [1:0.5:3], 4.2)
t =
    1.0000
    1.5000
    2.0000
    2.5000
    3.0000
y =
    4.2000
    2.4528
    2.6000
    3.7650
    5.8444
```

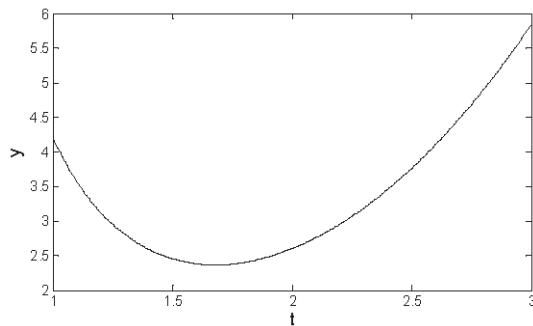
Начальное значение.

Вектор `tspan`.

Дескриптор пользовательской функции `ODEexp1`.

Решение получено решателем `ode45`. Имя пользовательской функции из этапа 2 есть `ODEexp1`. Решение начинается в точке $t = 1$ и заканчивается в $t = 3$ с шагом 0.5 (согласно вектору `tspan`). Чтобы показать решение графически, решаем задачу снова с использованием в `tspan` меньшего шага, и тогда решение графически изображается командой `plot`.

```
>> [t y]=ode45(@ODEexp1,[1:0.01:3],4.2);
>> plot(t,y)
>> xlabel('t'), ylabel('y')
```



Если функция ОДУ записана как анонимная функция под именем `ode1` (см. этап 2), то решение (то же самое как показано выше) получается вводом команды:

```
[t y]=ode45(ode1,[1:0.5:3],4.2).
```

9.5. Примеры приложений MATLAB

Пример задачи 9.3. Уравнение газа

Уравнение идеального газа связывает объем (V в Л), температуру (T в К), давление (P в атм) и количество газа (число молей n) следующим образом:

$$P = \frac{nRT}{V},$$

где $R = 0.08206$ (Л·атм)/(моль·К) есть газовая постоянная.

Уравнение Ван-дер-Ваальса дает соотношение между этими величинами для реального газа в виде:

$$\left(P + \frac{n^2 a}{V^2} \right) (V - nb) = nRT,$$

где a и b – определенные константы для каждого газа.

Использовать функцию `fzero` для вычисления объема 2 молей CO₂ при температуре 50 °C и давлении 6 атм. Для CO₂, $a = 3.59$ (Л²·атм)/моль² и $b = 0.0427$ Л/моль.

Решение

Решение, записанное в скрипт-файле, показано ниже.

```
global P T n a b R
R=0.08206;
P=6; T=323.2; n=2; a=3.59; b=0.047;
Vest=n*R*T/P;           Вычисление расчетного значения для V.
V=fzero (@Waals,Vest)   Используется дескриптор функции @waals для
                         вызова пользовательской функция waals в fzero.
```

Программа сначала вычисляет расчетное значение объема, используя уравнение идеального газа. Затем это значение используется в команде `fzero` для оценки решения. Уравнение Ван-дер-Ваальса записано как пользовательская функция под названием `Waals`, как показано ниже:

```
function fofx=Waals(x)
global P T n a b R
fofx=(P+n^2*a/x^2)*(x-n*b)-n*R*T;
```

Чтобы скрипт-файл и файл функции работали правильно, переменные P , T , n , a , b и R объявлены глобальными переменными. Когда скрипт-файл (сохраненный как `Chap9SamPro3`) выполняется в командном окне, значение V выводится на экран, как показано ниже:

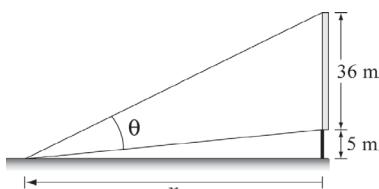
```
>> Chap9SamPro3
V =
8.6613
Объем газа равен 8.6613 Л.
```

Пример задачи 9.4. Максимальный угол обзора

Для лучшего просмотра фильма, человек должен находиться на расстоянии x от экрана так, чтобы угол обзора θ был максимальным. Определите расстояние x , при котором θ максимален для конфигурации, показанной на рисунке.

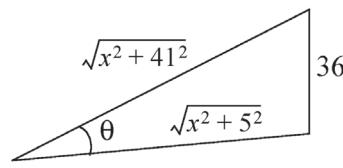
Решение

Задача может быть решена, если угол θ записать как функцию от x и затем найти значение x , для которого угол максимален. В треугольнике, который включает угол θ , одна сторона задана (высота экрана), а другие две стороны могут быть

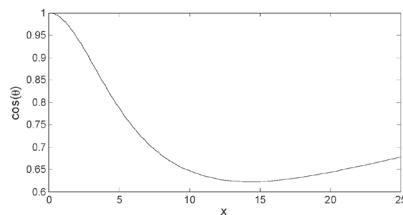


записаны через x , как показано на рисунке. Один из способов, когда θ может быть записан в виде функции от x , заключается в использовании теоремы косинусов:

$$\cos(\theta) = \frac{(x^2 + 5^2) + (x^2 + 41^2) - 36^2}{2\sqrt{x^2 + 5^2}\sqrt{x^2 + 41^2}}.$$



Естественно, что искомый угол θ будет между 0 и $\pi/2$. Поскольку косинус убывает с увеличением θ (в указанных пределах), то максимальный угол соответствует наименьшему значению $\cos(\theta)$. График $\cos(\theta)$ как функции от x показывает, что у функции есть минимум между 10 и 20 . Команды для построения графика:



```
>> fplot('((x^2+5^2)+(x^2+41^2)-36^2)/(2*sqrt(x^2+5^2)*sqrt(x^2+41^2))',[0 25])
>> xlabel('x'); ylabel('cos(\theta)')
```

Минимум может быть определен командой `fminbnd`:

```
>> [x anglecos]=fminbnd('((x^2+5^2)+(x^2+41^2)-36^2)/(2*sqrt(x^2+5^2)*sqrt(x^2+41^2))',10,20)

x =
    14.3178
anglecos =
    0.6225

>> angle=anglecos*180/pi
angle =
    35.6674
```

Минимум в точке $x = 14.3178$ м.
В этой точке $\cos(\theta) = 0.6225$.
В градусах угол равен 35.6674 .

Пример задачи 9.5. Поток воды в реке

Для оценки количества воды, которая течет в реке в течение года, часть реки сделана так, что имеет прямоугольное сечение как показано на рисунке. В начале каждого месяца (начиная с 1-го января) измеряется высота h воды и скорость v потока воды.

Первый день измерения берется как 1, а последний день – 1-го января следующего года – как день 366. При измерении были получены следующие данные:

День	1	32	60	91	121	152	182	213	244	274	305	335	366
h (м)	2.0	2.1	2.3	2.4	3.0	2.9	2.7	2.6	2.5	2.3	2.2	2.1	2.0
v (м/с)	2.0	2.2	2.5	2.7	5	4.7	4.1	3.8	3.7	2.8	2.5	2.3	2.0

Использовать эти данные, чтобы вычислить расход воды (количество жидкости, протекающей в единицу времени), и затем проинтегрировать этот расход воды для оценки общего количества воды, которая протекает в реке в течение года.

Решение

Расход воды Q (объем воды в секунду), в каждой точке данных получается умножением скорости воды на ширину и высоту поперечного сечения канала, в котором течет вода:

$$Q = v \cdot w \cdot h \quad (\text{м}^3/\text{с}).$$

Общий объем протекающей воды оценивается интегралом:

$$V = (60 \cdot 60 \cdot 24) \int_{t_1}^{t_2} Q dt.$$

Расход воды задан в кубических метрах в секунду, что означает, что время должно быть в секундах. Так как данные приведены в днях, интеграл умножается на $60 \cdot 60 \cdot 24$ сек/день (чтобы t_1 и t_2 были в днях).

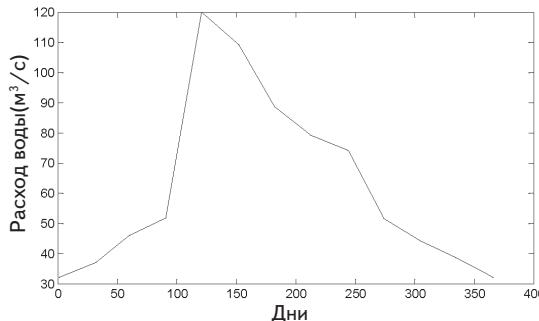
Ниже приведена программа, записанная в скрипт-файле, которая сначала вычисляет Q и затем выполняет интегрирование, используя команду `trapz`. Программа также создает график скорости потока как функции времени.

```
w=8;
d=[1 32 60 91 121 152 182 213 244 274 305 335 366];
h=[2 2.1 2.3 2.4 3.0 2.9 2.7 2.6 2.5 2.3 2.2 2.1 2.0];
speed=[2 2.2 2.5 2.7 5 4.7 4.1 3.8 3.7 2.8 2.5 2.3 2];
Q=speed.*w.*h;
Vol=60*60*24*trapz(d,Q);
fprintf('Оцененное количество воды, которая протекает в реке за год
есть %g кубических метров.',Vol)
plot(d,Q)
xlabel('дни'), ylabel('Расход воды (m^3/s)')
```

При выполнении файла (сохраненного как `Chap9SamPro5`) в командном окне вычисленное количество воды выводится на экран и создается график. Это все показано ниже:

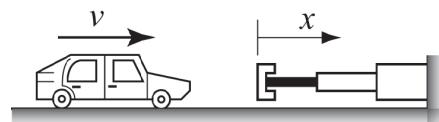
```
>> Chap9SamPro5
```

Оцененное количество воды, которая протекает в реке за год есть $2.03095e+009$ кубических метров.



Пример задачи 9.6. Удар автомобиля в бампер безопасности

Бампер безопасности размещается в конце рейстрека для остановки неконтролируемых автомобилей. Этот бампер проектирован так, что сила реакции бампера, действующая на автомобиль, есть функция скорости v и смещения x переднего края бампера согласно уравнению:



$$F = K v^3 (x + 1)^3,$$

где $K = 30$ (сек·кг)/м⁵ есть константа.

Автомобиль массы $m = 1500$ кг врезается в бампер со скоростью 90 км/ч. Определить и графически изобразить скорость автомобиля как функцию его положения для $0 \leq x \leq 3$ м.

Решение

Замедление автомобиля после удара о бампер может быть вычислено из второго закона движения Ньютона,

$$ma = -K v^3 (x + 1)^3,$$

которое может быть решено относительно ускорения как функция v и x :

$$a = \frac{-K v^3 (x + 1)^3}{m}.$$

Скорость как функция x может быть вычислена, подставляя ускорение в уравнение

$$v dv = a dx,$$

что дает

$$\frac{dv}{dx} = \frac{-Kv^2(x+1)^3}{m}.$$

Последнее уравнение – это ОДУ первого порядка, которое должно быть решено на интервале $0 \leq x \leq 3$ с начальным условием $v = 90$ км/ч в точке $x = 0$.

Численное решение дифференциального уравнения с MATLAB показано в следующей программе, которая записана в скрипт-файле:

```
global k m
k=30; m=1500; v0=90;
xspan=[0:0.2:3];           Вектор, который определяет интервал решения.
v0mps=v0*1000/3600;        Преобразования скорости в м/с.
[x v]=ode45(@bumper,xspan,v0mps)    Решение ОДУ.
plot(x,v)
xlabel('x (м)'); ylabel('Скорость (м/с)')
```

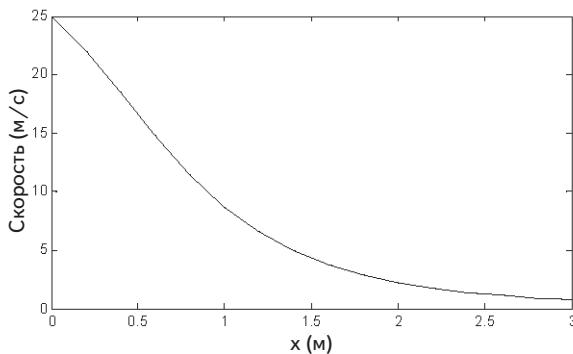
Отметим, что используется дескриптор функции `@bumper` для передачи в `ode45` пользовательской функции `bumper`. Листинг этой пользовательской функции для дифференциального уравнения, названной `bumper`:

```
function dvdx=bumper(x,v)
global k m
dvdx=-(k*v^2*(x+1)^3)/m;
```

При выполнении скрипта-файла (сохраненного как `Chap9SamPro6`) векторы x и v выведены на экран в командном окне (фактически, они выводятся на экран один за другим, но для экономии места, они показаны ниже рядом друг с другом).

```
>> Chap9SamPro6
x =          v =
      0          25.0000
    0.2000      22.0420
    0.4000      18.4478
    0.6000      14.7561
    0.8000      11.4302
    1.0000       8.6954
    1.2000       6.5733
    1.4000       4.9793
    1.6000       3.7960
    1.8000       2.9220
    2.0000       2.2737
    2.2000       1.7886
    2.4000       1.4226
    2.6000       1.1435
    2.8000       0.9283
    3.0000       0.7607
```

Созданный программой график скорости как функция расстояния:

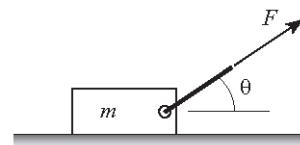


9.6. Задачи

1. Найти решение уравнения $e^{0.3x} - x^2 = -4$.
2. Найти решение уравнения $2\cos x - 0.5x = 1$.
3. Найти два корня уравнения $x^3 - 5x^{2.5} + e^{0.9x} + 4(x+1) = -2$.
4. Найти положительные корни уравнения $x^2 - 5x \sin(3x) + 3 = 0$.

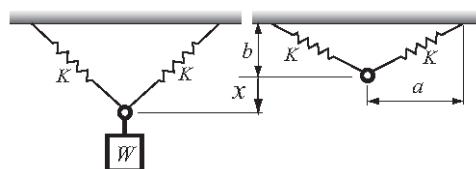
5. Ящик массой $m = 25$ кг тянет веревка. Сила, которая требуется для перемещения ящика, определяется формулой:

$$F = \frac{\mu mg}{\cos \theta + \mu \sin \theta},$$



где $\mu = 0.55$ – коэффициент трения и $g = 9.81$ м/с². Определите угол θ , если сила тяги F есть 150 N.

6. Весы состоят из двух пружин, как показано на рисунке. Пружины нелинейны так, что сила, которую они применяют, определяется по следующей формуле: $F_s = K_1 u + K_2 u^3$, где K – константы и $u = L - L_1$ является



ся удлинениями пружины ($L = \sqrt{a^2 + (b+x)^2}$ и $L_0 = \sqrt{a^2 + b^2}$ текущая и начальная длины пружин, соответственно). Изначально, пружины не растянуты. Если объект прикреплен к кольцу, пружины растягиваются и кольцо смещается вниз на расстояние x . Вес предмета может быть выражен через расстояние x по формуле:

$$W = 2F_S \frac{b+x}{L}.$$

Для данных весов $a = 0.22$ м, $b = 0.08$ м, и константы пружин $K_1 = 1600$ Гн/м и $K_2 = 100000$ Гн/м³. Графически изобразить W как функцию x для $0 \leq x \leq 0.25$. Определить расстояние x , когда к весам прикреплен объект 400 Гн.

7. Оценка минимальной скорости, необходимой для подскока круглого плоского камня при его ударении о воду дается формулой (Lyderic Bocquet, "The Physics of Stone Skipping," Am. J. Phys., vol. 71, no. 2, February 2003):

$$V = \frac{\sqrt{\frac{16Mg}{\pi C \rho_w d^2}}}{\sqrt{1 - \frac{8M \tan^2 \beta}{\pi d^3 C \rho_w \sin \theta}}},$$

где M и d – масса камня и его диаметр, ρ_w является плотностью воды, C – коэффициент, θ – угол наклона камня, β – угол падения и $g = 9.81$ м/с². Определите d если $V = 0.8$ м/с. (Предположите что $M = 0.1$ кг, $C = 1$, $\rho_w = 1000$ кг/м³ и $\beta = \theta = 10^\circ$.)

8. Диод показанной цепи представляет прямое смещение. Ток I текущий через диод определяется формулой:

$$I = I_s \left(e^{\frac{q v_D}{kT}} - 1 \right),$$

где v_D – падение напряжения через диод,

T – температура в кельвинах (К),

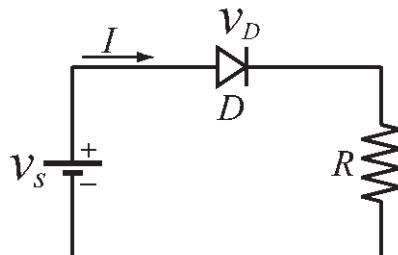
$I_s = 10^{-12}$ А – ток насыщения, $q = 1.6 \times 10^{-19}$ Кл – значение заряда электрона, и $k = 1.38 \times 10^{-23}$ Дж/К – постоянная Больцмана. Ток I текущий через цепь (тот же самый, что и ток в диоде) задается также формулой:

$$I = \frac{v_s - v_D}{R}.$$

Определите v_D , если $v_s = 2$ В, $T = 297$ К и $R = 1000$ Ом. (Подставьте I из одного уравнения в другое уравнение и решите полученное нелинейное уравнение.)

9. Определить минимум и максимум следующей функции:

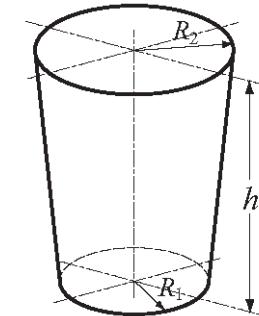
$$f(x) = \frac{3(x - 0.25)}{1 + 3.5(0.8x - 0.3)^2}.$$



10. Бумажный стаканчик в виде усеченного конуса с $R_2 = 2R_1$ проектируется так, чтобы иметь объем 250 см³. Определите R_1 , $2R_1$ и h так, чтобы для изготовления стаканчика использовать наименьшее количество бумаги. Объем и площадь поверхности бумажного стаканчика задаются формулами:

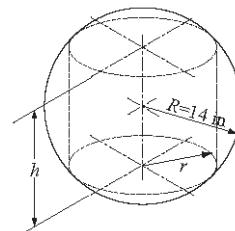
$$V = \frac{1}{3}\pi h(R_1^2 + R_1 R_2 + R_2^2),$$

$$S = \pi(R_1 + R_2)\sqrt{(R_1 - R_2)^2 + h^2} + \pi(R_1^2 + R_2^2).$$

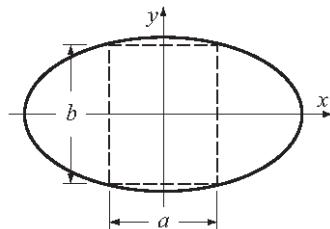


11. Рассмотрим снова блок из задачи 5. Определите угол θ , при котором сила, требуемая для движения ящика, когда его тянут, является наименьшей. Какова величина этой силы?

12. Определите размеры (радиус r и высоту h) и объем цилиндра наибольшего объема, который вписан в сферу радиуса $R = 14$ дюймов.



13. Рассмотрите эллипс $\frac{x^2}{19^2} + \frac{y^2}{5^2} = 1$. Определите стороны a и b прямоугольника наибольшей площади, вписанного в эллипс.



14. Закон излучения Планка задает спектральную яркость R как функция длины волны λ и температуры T (в кельвинах):

$$R = \frac{2\pi c^2 h}{\lambda^5} \frac{1}{e^{(hc)/(kT)} - 1},$$

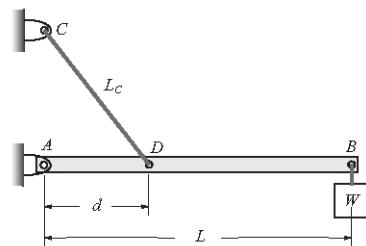
где $c = 3.0 \times 10^8$ м/с – скорость света, $h = 6.63 \times 10^{-34}$ Дж – постоянная Планка и $k = 1.38 \times 10^{-23}$ Дж/К – постоянная Больцмана.

Постройте график R как функции λ для $0.2 \times 10^{-6} \leq \lambda \leq 6.0 \times 10^{-6}$ м при $T = 1500$ К и определите длину волны, которая дает максимум R при этой температуре.

15. Длина балки AB , прикрепленной к стене в точке A – 108 дюймов и 68 дюймов – длина кабеля CD . К балке в точке B приложена нагрузка $W = 250$ фунтов. Растягивающая сила T кабеля задается формулой:

$$T = \frac{WLL_C}{d\sqrt{L_C^2 - d^2}},$$

где L и L_C – длины баки и кабеля, соответственно, и d – расстояние от точки A до точки D , где присоединен кабель. Создайте график график T как функции от d . Определите расстояние d , когда сила напряжения кабеля является наименьшей.



16. Используя MATLAB вычислите следующие интегралы:

$$(a) \int_2^{10} \frac{0.5x}{1+2\sqrt{x}} dx$$

$$(b) \int_0^9 \left(0.5 + \frac{\cos(1.2x)}{(x+2)^2} \right) dx$$

17. Используя MATLAB вычислите следующие интегралы:

$$(a) \int_1^8 \frac{e^x}{x^3} dx$$

$$(b) \int_0^{4\pi} \cos(x) e^{\sqrt{x}} dx$$

18. Скорость гоночного автомобиля в течение первых семи секунд гонки дана в таблице:

t (с)	0	1	2	3	4	5	6	7
v (миль/час)	0	14	39	69	95	114	129	139

Определите расстояние, пройденное автомобилем в течение первых шести секунд.

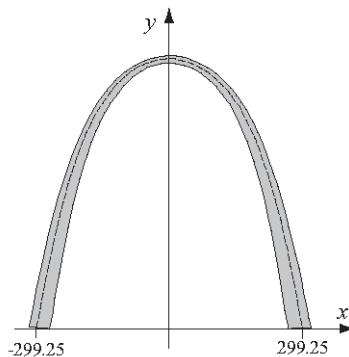
19. Форму линии центроида арки Gate-way в Сент-Луисе можно смоделировать приблизительно уравнением:

$$f(x) = 693.9 - 68.8 \cosh\left(\frac{x}{99.7}\right)$$

для $-299.25 \leq x \leq 299.25$ футов.

Используя данное уравнение определите длину этой дуги по формуле:

$$L = \int_a^b \sqrt{1 + (f'(x))^2} dx .$$

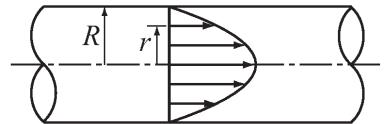


20. Расход воды Q (объем жидкости в секунду) потока в круглом канале может быть вычислен по формуле:

$$Q = \int_0^r 2\pi v r dr.$$

Для турбулентного потока профиль скорости может быть оценен по формуле:

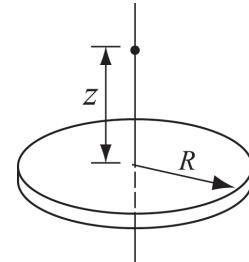
$$v = v_{\max} \left(1 - \frac{r}{R}\right)^{1/n}. \text{ Определите } Q \text{ для } R = 0.25 \text{ дюйм}, n = 7, v_{\max} = 80 \text{ дюйм/сек.}$$



- 21.** Электрическое поле E сообразуемое заряженным круговым диском в точке на расстоянии z вдоль оси диска задается формулой

$$E = \frac{\sigma z}{4\epsilon_0} \int_0^R (z^2 + r^2)^{-3/2} 2r dr,$$

где σ – плотность заряда, $\epsilon_0 = 8.85 \times 10^{-12} \text{ Кл}^2/(\text{Н} \cdot \text{м}^2)$ является диэлектрической постоянной и R – радиус диска. Определите электрическое поле в точке, расположенной в 5 см от диска с радиусом 6 см, с плотностью заряда $\sigma = 300 \text{ мККл}/\text{м}^2$.



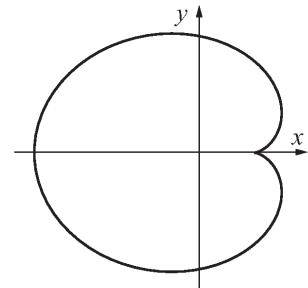
- 22.** Длина кривой, заданной параметрическим уравнением, определяется формулой:

$$\int_a^b \sqrt{[x'(t)]^2 + [y(t)]^2} dt.$$

Кардиоида, показанная на рисунке, задается уравнениями:

$$x = 2b\cos(t) - b\cos(2t), \quad y = 2b\sin(t) - b\sin(2t),$$

где $0 \leq t \leq 2\pi$.



Графически изобразите кардиоиду с параметром $b = 5$ и определите длину кривой.

- 23.** Изменение гравитационного ускорения g с высотой y определяется формулой

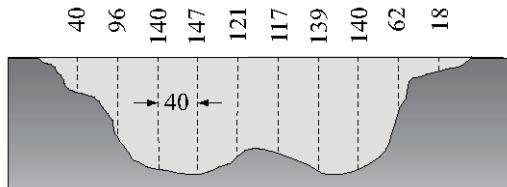
$$g = \frac{R^2}{(R+y)^2} g_0,$$

где $R = 6371 \text{ км}$ – радиус земли и $g_0 = 9.81 \text{ м}/\text{с}^2$ – гравитационное ускорение на уровне моря. Изменение гравитационной потенциальной энергии ΔU объекта, который поднят над землей на высоту h , задается формулой:

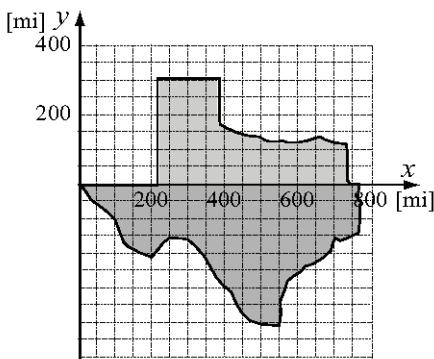
$$\Delta U = \int_0^h mg dy.$$

Определите изменение потенциальной энергии спутника массой 500 кг, который поднят от поверхности земли на высоту 800 км.

24. На рисунке показано сечение реки, полученное измерениями ее глубины с промежутками в 40 футов. Используйте численное интегрирование для оценки площади поперечного сечения реки.



25. Приблизительная карта Техаса показана на рисунке. Для определения площади штата, карта разделено на две части (одна выше, а другая ниже оси x). Определите площадь штата, численно интегрируя эти две области. Для каждой части сделайте таблицу значений координаты y границы как функции от x . Начните с $x = 0$ и используйте инкремент 50 миль так, что последняя точка $x = 750$. Сравните результат с фактической площадью Техаса, которая равна 261 797 квадратных миль.

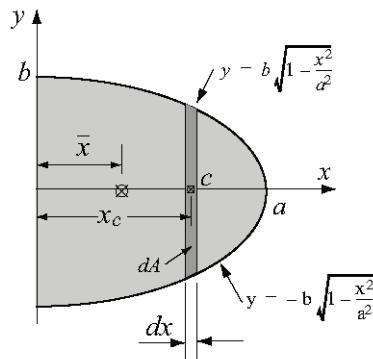


26. Поперечное сечение имеет геометрию половины эллипса, как показано на рисунке справа. Координата \bar{x} центра масс этой области может быть вычислена по формуле:

$$\bar{x} = \frac{M_y}{A},$$

где A – это площадь, заданная как $A = \pi ab/2$, и M_y – статический момент площади сечения относительно оси y , который вычисляется по формуле:

$$M_y = \int_A x_c dA = 2b \int_0^a x \sqrt{1 - \frac{x^2}{a^2}} dx.$$

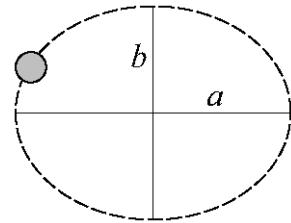


Определите \bar{x} , когда $a = 40$ мм и $b = 15$ мм.

27. Орбита Плутона имеет эллиптическую форму с $a = 5.9065 \times 10^9$ км и $b = 5.7208 \times 10^9$ км. Периметр эллипса может быть вычислен по формуле

$$P = 4a \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta} d\theta,$$

где $k = \frac{\sqrt{a^2 - b^2}}{2}$. Определите расстояние, которое проходит Плутон за один оборот. Вычислите среднюю скорость движения Плутон (в км/ч), если прохождение одного оборота занимает приблизительно 248 лет.



28. Интегралы Френеля имеют вид:

$$S(x) = \int_0^x \sin(t^2) dt \quad \text{и} \quad C(x) = \int_0^x \cos(t^2) dt.$$

Вычислите $S(x)$ и $C(x)$ для $0 \leq x \leq 4$ (используйте шаг 0.05). В одном окне Figure графически изобразите два графика – один $C(x)$ как функции x и другой $S(x)$ как функции $C(x)$.

29. Используйте встроенную функцию MATLAB для численного решения:

$$\frac{dy}{dx} = \frac{2x}{3y^2} \quad \text{для } 1 \leq x \leq 5 \quad \text{с условием } y(1) = 2.$$

В одном окне Figure графически изобразите численное решение сплошной линией и точное решение дискретными точками.

Точное решение: $y = \sqrt[3]{x^2 + 7}$.

30. Используйте встроенную функцию MATLAB для численного решения:

$$\frac{dy}{dx} = \frac{2x+1}{y+2} \quad \text{для } 0 \leq x \leq 8 \quad \text{с условием } y(0) = 2.$$

В одном окне Figure графически изобразите численное решение сплошной линией и точное решение дискретными точками.

Точное решение: $y = \sqrt{2x^2 + 2x + 16} - 2$.

31. Используйте встроенную функцию MATLAB для численного решения:

$$\frac{dy}{dt} = 80e^{-1.6t} \cos(4t) - 0.4y \quad \text{для } 0 \leq t \leq 4 \quad \text{с условием } y(0) = 0.$$

Постройте график решения.

32. Используйте встроенную функцию MATLAB для численного решения:

$$\frac{dy}{dx} = -x^2 + \frac{x^3 e^{-y}}{4} \quad \text{для } 1 \leq x \leq 5 \text{ с условием } y(1) = 1.$$

Постройте график решения.

- 33.** Рост рыбы часто моделируется моделью роста фон Берталанффи:

$$\frac{dw}{dt} = aw^{2/3} - bw,$$

где w – вес и a и b – константы. Решите уравнение относительно w для случая $a = 5$ фунт $^{1/3}$, $b = 2$ день $^{-1}$ и $w(0) = 0.5$ фунтов. Удостоверьтесь, что выбранный период времени только достаточно длинный с тем, чтобы приблизиться к максимальному весу. Каков будет максимальный вес для этого случая? Создайте график w как функции времени.

- 34.** Бак для воды имеет форму эллипсоида ($a = 1.5$ м, $b = 4$ м, $c = 3$ м). В нижней части бака имеется круглое отверстие, как показано на рисунке. Согласно закону Торричелли, скорость v вытекающей из отверстия воды определяется уравнением:

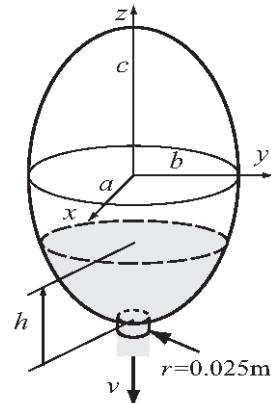
$$v = \sqrt{2gh}.$$

где h – высота воды и $g = 9/81$ м/сек 2 . Скорость изменения высоты h при истечении воды через отверстие описывается уравнением:

$$\frac{dh}{dt} = \frac{\sqrt{2gh}r^2}{ac\left(-1 + \frac{(h-c)^2}{c^2}\right)},$$

где r – радиус дыры.

Решите это дифференциальное уравнение относительно h . Начальная высота воды равна $h = 5.9$ м. Решите задачу определения h для разных периодов времени полного истечения и найдите оценку времени, когда $h = 0.1$ м. Создайте график h как функции времени.



- 35.** Внезапная вспышка роста численности насекомых может быть смоделирована уравнением:

$$\frac{dN}{dt} = RN\left(1 - \frac{N}{C}\right) - \frac{rN^2}{N_c^2 + N^2}.$$

Первый член относится к известной логистической модели прироста численности, где N – число насекомых, R – внутренний темп роста и C – кормовая

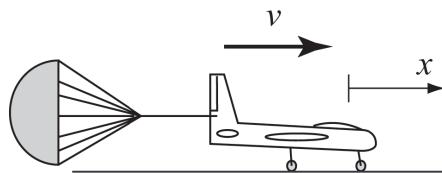
продуктивность окружающей среды. Второй член представляет эффект хищничества птиц. Его влияние становится существенным, когда популяция достигает некоторого критического размера $N_c r$ – это максимальное значение, которое второй член может достигнуть при больших значениях N .

Решите это дифференциальное уравнение в течение $0 \leq t \leq 50$ дней и для двух темпов роста $R = 0.55$ и $R = 0.58$ день $^{-1}$, и с $N(0) = 10000$. Другие параметры: $C = 10^4$, $N_c = 10^4$, $r = 10^4$ день $^{-1}$. Сделайте графики в одном окне для сравнения этих двух решений и обсудите, почему эту модель называют моделью «вспышки».

- 36.** Самолет использует парашют и другие средства торможения для торможения на взлетно-посадочной полосе после посадки. Его ускорение задается уравнением $a = -0.0035 v^3 - 3$ м/с 2 . Поскольку $a = \frac{dv}{dt}$, производная скорости задается так:

$$\frac{dv}{dt} = -0.0035v^2 - 3.$$

Рассмотрите самолет, который в момент $t = 0$ открывает парашют на скорости 300 км/ч и начинает уменьшать свою скорость.



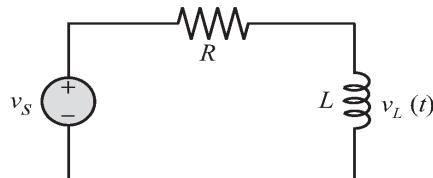
- (a) Решая дифференциальное уравнение, определите и графически изобразите скорость как функцию времени от $t = 0$ сек от и до остановок самолета.
(b) Используйте численное интегрирование для определения расстояния x пробега самолета как функции времени. Создайте график x как функции времени.

- 37.** Рост популяции некоторого вида с ограниченными возможностями роста может быть смоделирован уравнением:

$$\frac{dN}{dt} = kN(N_M - N),$$

где N – численность популяции, N_M – это предельная численность популяции и k – некоторая постоянная. Рассмотрите случай, когда $N_M = 5000$, $k = 0.000095$ 1/год и $N(0) = 100$. Определите N для $0 \leq t \leq 20$. Создайте график N как функции t .

- 38.** Цепь RL включает источник напряжения v_s , сопротивление $R = 1.8$ Ом и катушку индуктивности $L = 0.4$ Гн, как показано на рисунке. Дифференциальное уравнение, которое описывает реакцию цепи:



$$\frac{L}{R} \frac{di_L}{dt} + i_L = \frac{v_s}{R},$$

где i_L – ток в индукторе. Первоначально $i_L = 0$, а затем в момент $t = 0$ напряжение источника меняется. Определите реакцию цепи для следующих трех случаев:

- (a) $v_s = 10 \sin(30\pi t)$ В для $t \geq 0$.
- (b) $v_s = 10 e^{-t/0.06} \sin(30\pi t)$ В для $t \geq 0$.

Каждый случай соответствует разному дифференциальному уравнению. Решение – ток в индукторе как функция времени. Решите каждый случай для $0 \leq t \leq 0.4$ сек. Для каждого случая создайте графики v_s и i_L как функций времени (сделайте два отдельных окна графиков на одной странице).

39. Рост опухоли может быть смоделирован уравнением:

$$\frac{dA}{dt} = \alpha A \left[1 - \left(\frac{A}{k} \right)^v \right],$$

где $A(t)$ – площадь опухоли, α , k и v – константы. Решите это уравнение для $0 \leq t \leq 30$ дней и для параметров $\alpha = 0.8$, $k = 60$, $v = 0.25$ и $A(0) = 1$ мм². Создайте график A как функции времени.

40. Скорость объекта, который падает свободно вследствие земного притяжения, может быть смоделирована уравнением:

$$m \frac{dv}{dt} = -mg + kv^2,$$

где m – масса объекта, $g = 9.81$ м/с² и k – постоянная. Решите это уравнение относительно v для случая $m = 5$ кг, $k = 0.05$ кг/м, $0 \leq t \leq 15$ сек и $v(0) = 0$ м/с. Создайте график v как функции времени.



ГЛАВА 10.

Трехмерные графики

Трехмерные (3D) графики могут быть полезны для представления данных, которые составлены более чем из двух переменных. MATLAB обеспечивает различные варианты для вывода на экран трехмерных данных. Они включают линии, поверхности, графики-сетки и многие другие. Графики могут быть также отформатированы для определенного вида и спецэффектов. В этой главе описаны многие из функций трехмерной графики. Дополнительная информация может быть найдена в справочной системе MATLAB в разделе «Plotting and Data Visualization».

Эта глава является во многом продолжением главы 5, где были представлены двумерные графики. 3D графики представлены в отдельной главе, потому что их используют не все пользователи MATLAB. Кроме того, новые пользователи MATLAB, вероятно обнаружат что прежде, чем пытаться создавать 3D графики, легче изучить материал в главах 6–9 и попрактиковаться в 2D графиках. Поэтому в остальной части этой главы предполагается, что читатель знаком с построением 2D графиков в MATLAB.

10.1. Графики линий

Трехмерный график линии – это линия, которая получается соединением точек в трехмерном пространстве. Базовый трехмерный график создается командой `plot3`, которая очень похожа на команду `plot` и имеет формат:

```
plot3(x,y,z, 'line specifiers', 'PropertyName', property value)
```

х, у и z – это
векторы координат
точек.

(Дополнительные.)
Спецификаторы, кото-
рые определяют тип и
цвет линии и маркеров.

(Дополнительные.) Свойства со значени-
ями, которые могут использоваться для
определения ширины линии и размера
маркера, его края и цвета заливки.

- У этих трех векторов с координатами точек данных должно быть одно и то же число элементов.
- Спецификаторы линии, свойства и значения свойств – те же самые как в 2D графиках (см. раздел 5.1).

Например, если координаты x , y и z заданы как функция параметра t

$$\begin{aligned}x &= \sqrt{t} \sin(2t) \\y &= \sqrt{t} \cos(2t) \\z &= 0.5t\end{aligned},$$

то график точек для $0 \leq t \leq 6\pi$ может быть создан следующим скриптом-файлом:

```
t=0:0.1:6*pi;
x=sqrt(t).*sin(2*t);
y=sqrt(t).*cos(2*t);
z=0.5*t;
plot3(x,y,z,'k','linewidth',1)
grid on
xlabel('x'); ylabel('y'); zlabel('z')
```

При выполнении этого сценария создается график, показанный на рис. 10.1.

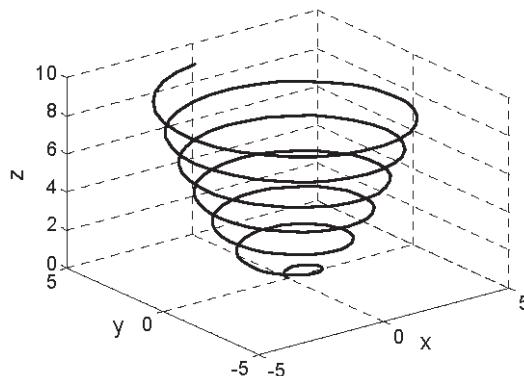


Рис. 10.1. График функции $x = \sqrt{t} \cdot \sin(2t)$, $y = \sqrt{t} \cdot \cos(2t)$, $z = 0.5t$ для $0 \leq t \leq 6\pi$

10.2. Сети и графики поверхностей

Сети и графики поверхностей – это трехмерные графики, используемые для графического изображения функций вида $z = f(x, y)$, где x и y – независимые переменные, а z – зависимая переменная. Это означает, что в пределах данной области значение z может быть вычислено для любой комбинации x и y . Сети и графики поверхностей создаются в три этапа. На первом этапе создается координатная сетка в плоскости xy , которая покрывает область определения функции. На втором этапе вычисляется значение z в каждой точке сетки. На третьем этапе создается график. Эти три этапа более подробно изложены ниже.

Создание координатной сетки в плоскости xy (прямоугольные координаты)

Координатная сетка (решетка) – это множество точек в плоскости xy в области определения функции. Плотность этой сетки (число точек, которые используются для определения области) определяется пользователем. Рис. 10.2 показывает сетку в области $-1 \leq x \leq 3$ и $1 \leq y \leq 4$.

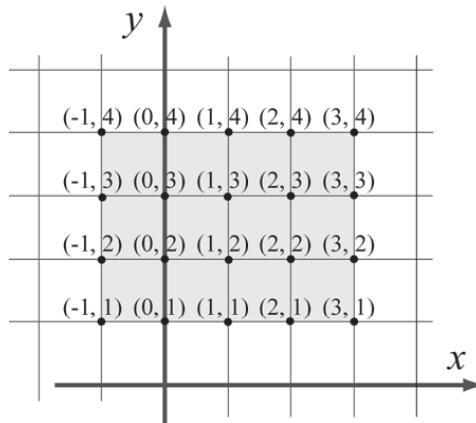


Рис. 10.2. Координатная сетка в области $-1 \leq x \leq 3$ и $1 \leq y \leq 4$ с шагом 1

расстояние между этими точками равно единице. Точки сетки могут быть определены двумя матрицами X и Y . Матрица X состоит из x -координат всех точек, а матрица Y – из y -координат всех точек:

$$X = \begin{bmatrix} -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \end{bmatrix} \quad \text{и} \quad Y = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Матрица X состоит из одинаковых строк, поскольку x -координаты точек будут повторяться в каждой строке сетки. Таким же образом матрица Y состоит из одинаковых столбцов, поскольку y -координаты точек будут повторяться в каждом столбце сетки.

MATLAB имеет встроенную функцию, называемую `meshgrid`, которая может использоваться для создания матриц X и Y . Вид функции `meshgrid`:

$[X, Y] = \text{meshgrid}(x, y)$

x – это матрица x -координат точек сетки.

y – это матрица y -координат точек сетки.

x – это вектор, который разбивает область изменения x .

y – это вектор, который разбивает область изменения y .

В векторах x и y первые и последние элементы – это соответствующие границы области определения. Плотность сетки определяется числом элементов в этих векторах. Например, сеточные матрицы X и Y , которые соответствуют координатной сетке на рис. 10.2, могут быть созданы командой `meshgrid` следующим образом:

```
>> x=-1:3;
>> y=1:4;
>> [X,Y]=meshgrid(x,y)
X =
-1     0     1     2     3
-1     0     1     2     3
-1     0     1     2     3
-1     0     1     2     3
Y =
1     1     1     1     1
2     2     2     2     2
3     3     3     3     3
4     4     4     4     4
```

Как только эти матрицы сетки созданы, они могут использоваться для вычисления значения z в каждом узле решетки.

Вычисление значения z в каждой точке координатной сетки

Значения переменной z в точках координатной сетки – это матрица, а переменные x и y – векторы. Чтобы использовать для z поэлементные вычисления, эти линейные переменные x и y были преобразованы на первом этапе в матрицы X и Y . Теперь значение z в каждой точке вычисляется через поэлементные вычисления так же, как и с векторами. Когда независимые переменные X и Y – матрицы (они должны иметь одинаковый размер), вычисляемая зависимая переменная – тоже матрица того же размера. Значение z в каждом адресе вычисляется по соответствующим значениям X и Y . Например, если z задано формулой

$$z = \frac{xy^2}{x^2 + y^2},$$

то значение z в каждой точке сетки выше вычисляется так:

```
>> Z = X.*Y.^2 ./ (X.^2 + Y.^2)
Z =
-0.5000      0      0.5000      0.4000      0.3000
-0.8000      0      0.8000      1.0000      0.9231
-0.9000      0      0.9000      1.3846      1.5000
-0.9412      0      0.9412      1.6000      1.9200
```

Когда созданы эти три матрицы X , Y и Z , они могут использоваться для графического изображения сети или графика поверхности.

Создание графика в виде сети и графика поверхности

Сеточный график (`mesh`, проволочный каркас) или график поверхности создаются командами `mesh` или `surf`, которые имеют вид:

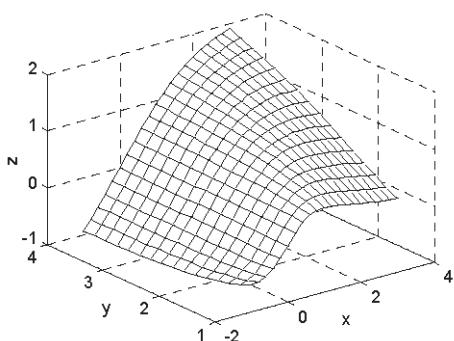
`mesh (X, Y, Z)` `surf (X, Y, Z),`

где X и Y – матрицы с координатами решетки, а Z – матрица со значениями z в узлах решетки. Сеточный график (сеть) получается из линий, которые соединяют точки. В графике поверхности внутренние области ячеек сети окрашены.

В качестве примера, следующий скрипт-файл содержит полную программу, которая создает координатную решетку (`grid`) и затем делает сеточный график (или график поверхности) функции $z = \frac{xy^2}{x^2 + y^2}$ в области $-1 \leq x \leq 3$ и $1 \leq y \leq 4$.

```
x=-1:0.1:3;
y=1:0.1:4;
[X, Y]=meshgrid(x, y);
Z=X.*Y.^2 ./ (X.^2+Y.^2);
mesh(X, Y, Z)           Для графика поверхности вводим surf(X, Y, Z).
xlabel('x'); ylabel('y'); zlabel('z')
```

Отметим, что в этой программе шаг векторов x и y намного меньше, чем было ранее в этом разделе. Меньший интервал создает более плотную координатную сетку. Рисунки, создаваемые программой:



Сеточный график (сеть)

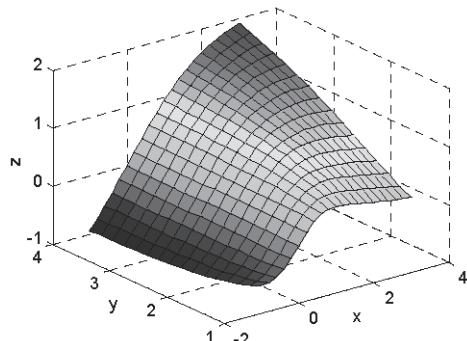


График поверхности

Дополнительные комментарии к команде `mesh`

- Линии создаваемых сеточных графиков имеют цвета, которые изменяются в соответствии с величиной z . Вариация в цвете добавляет трех-

мерность при визуализации графиков. Цвет может быть изменен на постоянный либо с использованием **Редактора графиков** в окне графиков **Figure** (выберите стрелку редактирования, щелкните по рисунку, чтобы открыть окно **Редактора свойств**, затем измените цвет в списке **Свойства сети**), или с использованием команды `colormap(c)`. В этой команде `c` есть вектор с тремя элементами, в котором первый, второй и третий элементы определяют интенсивность красного, зеленого и синего (RGB) цвета, соответственно. Каждый элемент может быть числом между 0 (минимальная интенсивность) и 1 (максимальная интенсивность). Некоторые типичные цвета:

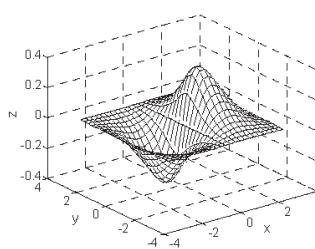
$$\begin{array}{lll} C = [0 \ 0 \ 0] \text{ черный} & C = [1 \ 0 \ 0] \text{ красный} & C = [0 \ 1 \ 0] \text{ зеленый} \\ C = [0 \ 0 \ 1] \text{ синий} & C = [1 \ 1 \ 0] \text{ желтый} & C = [1 \ 0 \ 1] \text{ пурпурный} \\ C = [0.5 \ 0.5 \ 0.5] \text{ серый} & & \end{array}$$

- Когда выполняется команда `mesh`, координатная сетка есть по умолчанию. Она может быть выключена командой `grid off`.
- Вокруг графика командой `box on` могут вычерчиваться ребра координатной коробки.
- Команды `mesh` и `surf` могут также использоваться в форме `mesh(z)` и `surf(z)`. В этом случае значения `z` графически изображаются в виде функции адресов (индексов) в матрице `z`. Номер строки соответствует оси `x`, а номер столбца соответствует оси `y`.

Есть несколько дополнительных команд графического изображения, которые подобны командам `mesh` и `surf`, и которые создают графики с различными особенностями. Табл. 10.1 показывает сводку `mesh` и `surf` команд графического изображения поверхности. Все примеры в таблице – графики функции

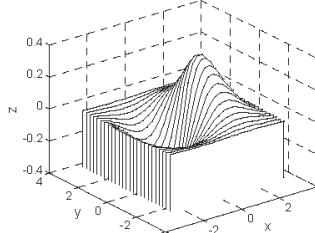
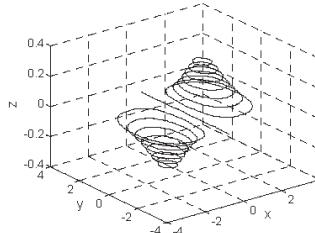
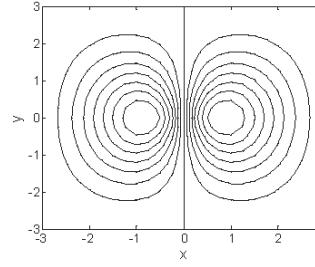
$$z = 1.8^{-1.5\sqrt{x^2+y^2}} \sin(x) \cos(0.5y) \text{ в области } -3 \leq x \leq 3 \text{ и } -3 \leq y \leq 3.$$

Таблица 10.1. Сеточные графики и графики поверхностей

Тип графика	Пример графика	Программа
Сеточный график		<pre> x=-3:0.25:3; y=-3:0.25:3; [X,Y]=meshgrid(x,y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); mesh(X,Y,Z) xlabel('x'); ylabel('y') zlabel('z') </pre>
Формат функции: <code>mesh(X, Y, Z)</code>		



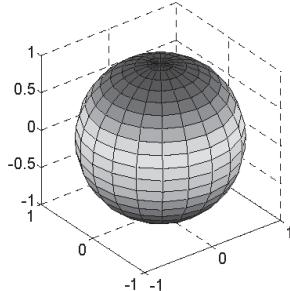
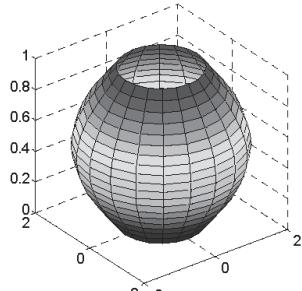
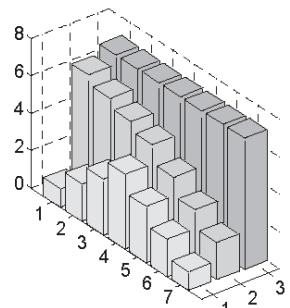
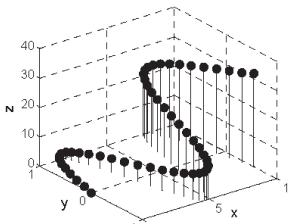
Тип графика	Пример графика	Программа
График поверхности Формат функции: <code>surf(X, Y, Z)</code>		<pre>x=-3:0.25:3; y=-3:0.25:3; [X, Y]=meshgrid(x, y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); surf(X, Y, Z) xlabel('x'); ylabel('y') zlabel('z')</pre>
Сеточный график со шторой (изображает штору вокруг сети) Формат функции: <code>meshz(X, Y, Z)</code>		<pre>x=-3:0.25:3; y=-3:0.25:3; [X, Y]=meshgrid(x, y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); meshz(X, Y, Z) xlabel('x'); ylabel('y') zlabel('z')</pre>
Сеточный и контурный график (изображает контурные линии ниже сети). Формат функции: <code>meshc(X, Y, Z)</code>		<pre>x=-3:0.25:3; y=-3:0.25:3; [X, Y]=meshgrid(x, y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); meshc(X, Y, Z) xlabel('x'); ylabel('y') zlabel('z')</pre>
График поверхности и контуров (изображает контурные линии ниже поверхности). Формат функции: <code>surfzc(X, Y, Z)</code>		<pre>x=-3:0.25:3; y=-3:0.25:3; [X, Y]=meshgrid(x, y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); surfzc(X, Y, Z) xlabel('x'); ylabel('y') zlabel('z')</pre>
График поверхности с подсветкой Формат функции: <code>surfl(X, Y, Z)</code>		<pre>x=-3:0.25:3; y=-3:0.25:3; [X, Y]=meshgrid(x, y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); surfl(X, Y, Z) xlabel('x'); ylabel('y') zlabel('z')</pre>

Тип графика	Пример графика	Программа
График водопада (изображает линии сети только в одном направлении), Формат функции: waterfall(X, Y, Z)		<pre>x=-3:0.25:3; y=-3:0.25:3; [X,Y] = meshgrid(x,y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); waterfall(X,Y,Z) xlabel('x'); ylabel('y') zlabel('z')</pre>
График 3D-контуров Формат функции: contour3(X, Y, Z, n) n – число уровней контуров (опционально)		<pre>x=-3:0.25:3; y=-3:0.25:3; [X,Y]=meshgrid(x,y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); contour3(X,Y,Z,15) xlabel('x'); ylabel('y') zlabel('z')</pre>
График 2D-контуров (проекции контуров разных уровней на плоскость xy) Формат функции: contour(X, Y, Z, n) n – число уровней контуров (опционально)		<pre>x=-3:0.25:3; y=-3:0.25:3; [X,Y]=meshgrid(x,y); Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X); contour(X,Y,Z,15) xlabel('x'); ylabel('y') zlabel('z')</pre>

10.3. Специальные графики

MATLAB имеет дополнительные функции для создания различных типов специальных трехмерных графиков. Полный список может быть найден в Окне справки MATLAB в разделах графики и визуализации данных. Несколько из этих 3D графиков представлены в табл. 10.2. Примеры в таблице не показывают всех вариантов, доступных каждому типу графиков. Подробности относительно каждого типа графиков могут быть найдены в окне **Справка**, или вводом `help command_name` в командном окне.

Таблица 10.2. Специализированные 3D графики

Тип графика	Пример графика	Программа
График сферы Формат функции: <code>sphere</code> Возвращает координаты x, y и z точек единичной сферы с 20×20 гранями. <code>sphere(n)</code> То же самое, но с $n \times n$ гранями.		<code>sphere</code> или <code>[X, Y, Z] = sphere(20);</code> <code>surf(X, Y, Z)</code>
График цилиндра Формат функции: <code>[X, Y, Z] = cylinder(r)</code> Возвращает координаты x, y и z точек цилиндра с профилем r .		<code>t=linspace(0, pi, 20);</code> <code>r=1+sin(t);</code> <code>[X, Y, Z] = cylinder(r);</code> <code>surf(X, Y, Z)</code> <code>axis square</code>
3D бар-график Формат функции: <code>bar3(Y)</code> Каждый элемент из Y есть один бар (брускок). Столбцы группируются.		<code>Y=[1 6.5 7; 2 6 7; 3 5.5 7; 4 5 7; 3 4 7; 2 3 7; 1 2 7];</code> <code>bar3(Y)</code>
3D стебли (изображает последовательные точки с маркерами и с верти- кальными линиями от xy-плоскости). Формат функции: <code>stem3(X, Y, Z)</code>		<code>t=0:0.2:10;</code> <code>x=t;</code> <code>y=sin(t);</code> <code>z=t.^1.5;</code> <code>stem3(x, y, z, 'fill')</code> <code>grid on</code> <code>xlabel('x');</code> <code>ylabel('y')</code> <code>zlabel('z')</code>

Тип графика	Пример графика	Программа				
3D график рассеяния Формат функции: scatter3(X, Y, Z)		t=0:0.4:10; x=t; y=sin(t); z=t.^1.5; scatter3(x,y,z, 'filled') grid on colormap([0.1 0.1 0.1]) xlabel('x'); ylabel('y'); zlabel('z')				
3D круговая диаграмма Формат функции: pie3(X, explode)	<table border="1"> <tr> <td>42%</td> </tr> <tr> <td>29%</td> </tr> <tr> <td>19%</td> </tr> <tr> <td>10%</td> </tr> </table>	42%	29%	19%	10%	X=[5 9 14 20]; explode=[0 0 1 0]; pie3(X=explode) explode – вектор (той же длины, что и X) из нулей и единиц. 1 смещает ломтик от центра.
42%						
29%						
19%						
10%						

Сетка полярных координат в xy-плоскости

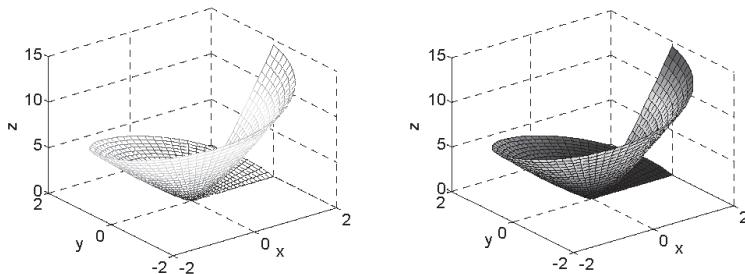
3D график функции, когда значение z задано в полярных координатах (например $z = r\theta$), может быть создан в результате следующих шагов:

- Создание сетки значений θ и r при помощи функции `meshgrid`.
- Вычисление значения z в каждой точке сетки.
- Преобразование сетки полярных координат в сетку прямоугольных координат. Это может быть сделано встроенной функцией MATLAB `pol2cart` (см. пример ниже).
- Создание 3D графика, используя значения z и прямоугольные координаты.

Например, следующий скрипт-файл создает график функции $z = r\theta$ в области $0 \leq \theta \leq 360^\circ$ и $0 \leq r \leq 2$.

```
[th,r]=meshgrid((0:5:360)*pi/180,0:.1:2);
Z=r.*th;
[X,Y] = pol2cart(th,r);
mesh(X,Y,Z)
Для графика поверхности вводим surf(X,Y,Z).
```

Рисунки, создаваемые программой:



10.4. Команда view

Команда `view` управляет направлением, с которого просматривается график. Это делается указанием направления через азимут и угол возвышения, как показано на рис. 10.3, или определением точки в пространстве из которой просматривается график. Для установки угла обзора графика, команды `view` имеет вид:

```
view(az,el) или view([az,el])
```

- `az` – это азимут, который является углом (в градусах) в xy -плоскости, измеряемый от отрицательного направления оси y против часовой стрелки.
- `el` – это угол возвышения (в градусах) от xy -плоскости. Положительное значение соответствует открытию угла в направлении оси z .
- Значения углов по умолчанию: $az = -37.5^\circ$ и $el = 30^\circ$.

В качестве примера, график поверхности из табл. 10.1 изображен снова с углами просмотра $az = 20^\circ$ и $el = 35^\circ$ (рис. 10.4).

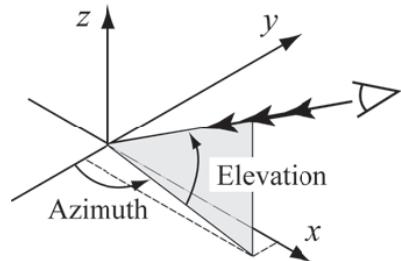


Рис. 10.3. Азимут и угол возвышения

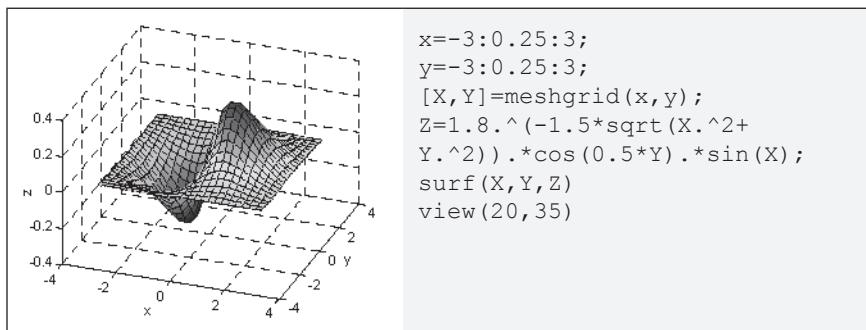


Рис. 10.4. График поверхности функции $z = 1.8^{-\sqrt{x^2+y^2}} \sin(x) \cos(0.5y)$ с углами просмотра $az = 20^\circ$ и $el = 35^\circ$

При выборе соответствующего азимута и угла возвышения, команда `view` может использоваться для графического изображения проекций 3D графиков на различные плоскости согласно следующей таблице:

Плоскость проекции	Значение <code>az</code>	Значение <code>el</code>
xy (вид сверху)	0	90
xz (вид сбоку)	0	0
yz (вид сбоку)	90	0

Ниже показаны примеры проекций 3D графиков. Рис. 10.5 показывает вид сверху функции, которая графически изображена на рис. 10.1. Далее, на рис. 10.6 и 10.7, соответственно, показаны примеры проекций на xz - и yz -плоскости. Эти рисунки показывают проекции сеточных графиков функции, графически изображенной в табл. 10.1.

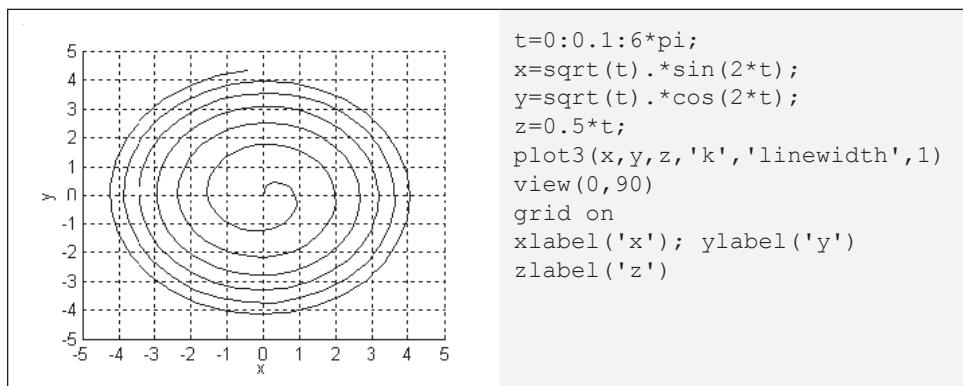


Рис. 10.5. Вид сверху функции $x = \sqrt{t} \cdot \sin(2t)$, $y = \sqrt{t} \cdot \cos(2t)$, $z = 0.5t$ для $0 \leq t \leq 6\pi$

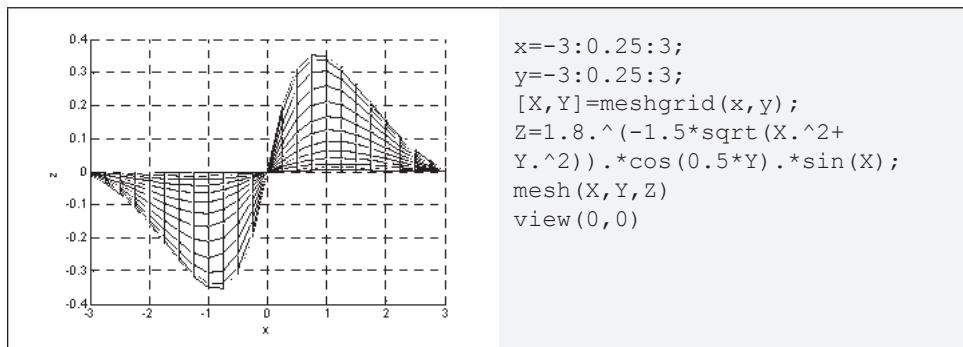
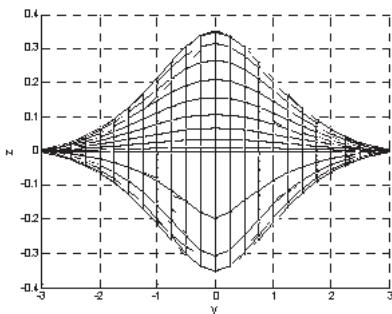


Рис. 10.6. Проекция на xz -плоскость функции $z = 1.8^{-1.5\sqrt{x^2+y^2}} \sin(x) \cos(0.5y)$



```
x=-3:0.25:3;
y=-3:0.25:3;
[X,Y]=meshgrid(x,y);
Z=1.8.^(-1.5*sqrt(X.^2+Y.^2)).*cos(0.5*Y).*sin(X);
surf(X,Y,Z)
view(90,0)
```

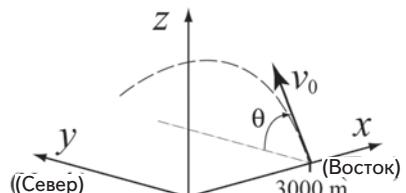
Рис. 10.7. Проекция на yz -плоскость функции $z = 1.8^{-1.5\sqrt{x^2+y^2}} \sin(x) \cos(0.5y)$

- Команда `view` может также установить значения просмотра по умолчанию:
 - `view(2)` устанавливает значение по умолчанию для вида сверху, что является проекцией на xy плоскость с азимутом $az = 0^\circ$ и $el = 90^\circ$.
 - `view(2)` устанавливает значение по умолчанию стандартного 3D просмотра с азимутом $az = 37.5^\circ$ и $el = 30^\circ$.
- Направление просмотра может также быть установлено выбором точки в пространстве из которой просматривается график. В этом случае команда `view` имеет формат `view([x, y, z])`, где x , y и z – координаты точки. Это направление определено направлением от указанной точки к началу системы координат и независимо от расстояния. Это означает, что вид из точки [6,6,6] точно такой же, как из точки [10,10,10]. Вид сверху может быть установлен из точки [0,0,1]. Вид сбоку на xz плоскость со стороны отрицательного направления оси y может быть установлен из точки [0,-1,0], и так далее.

10.5. Примеры приложений MATLAB

Пример задачи 10.1. 3D траектория снаряда

Снаряд выпущен с начальной скоростью 250 м/с под углом $\theta = 65^\circ$ относительно земли. Снаряд нацелен прямо на север. Из-за сильного ветра, дующего на запад, снаряд также перемещается в этом направлении с постоянной скоростью 30 м/с. Определить и графически изобразить траекторию снаряда до его падения на землю. Для сравнения создать также график (в том же самом окне Figure) траектории, которую имел бы снаряд, если бы не было ветра.



Решение

Как показано на рисунке, установлена система координат так, что оси x и y направлены на восток и север, соответственно. Тогда движение снаряда может быть проанализировано, рассматривая вертикальное направление z и эти две горизонтальных составляющие x и y . Так как снаряд выпущен прямо север, начальная скорость v_0 может быть разложена на горизонтальную y компоненту и вертикальную z компоненту:

$$v_{0y} = v_0 \cos(\theta) \quad \text{и} \quad v_{0z} = v_0 \sin(\theta).$$

Кроме того, вследствие ветра снаряд имеет постоянную скорость в отрицательном направлении оси x , $v_x = -30$ м/с.

Начальное положение снаряда (x_0, y_0, z_0) есть точка $(3000, 0, 0)$. В вертикальном направлении скорость и положение снаряда задаются формулами:

$$v_z = v_{0z} - gt \quad \text{и} \quad z = z_0 + v_{0z}t - \frac{1}{2}gt^2.$$

Время, которое требуется снаряду для достижения самой высокой точки ($v_z = 0$) есть $t_{h\max} = \frac{v_{0z}}{g}$. Полное время полета есть удвоенное это время, $t_{tot} = 2t_{h\max}$. В горизонтальном направлении скорость является постоянной (в обеих направлениях x и y), поэтому положение снаряда задается формулами:

$$x = x_0 + v_x t \quad \text{и} \quad y = y_0 + v_{oy} t.$$

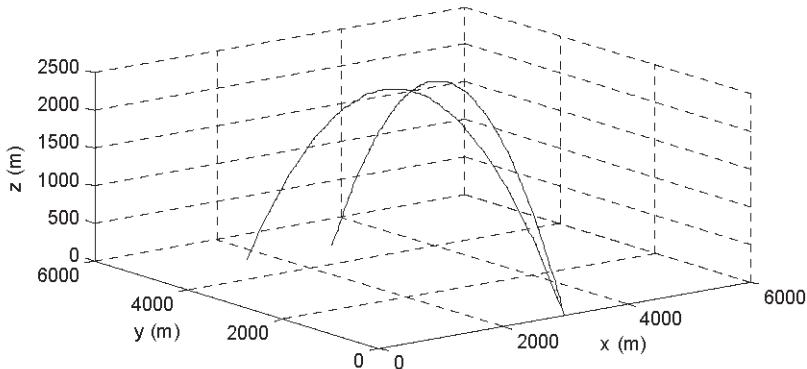
Следующая программа MATLAB, записанная в скрипт-файле, решает эту задачу по уравнениям, указанным выше.

```

v0=250; g=9.81; theta=65;
x0=3000; vx=-30;
v0z=v0*sin(theta*pi/180);
v0y=v0*cos(theta*pi/180);
t=2*v0z/g;
tplot=linspace(0,t,100);           Создание вектора времени из 100 элементов.
z=v0z*tplot-0.5*g*tplot.^2;        Вычисление координат x, y и z снаряда
y=v0y*tplot;                      в каждый момент времени.
x=x0+vx*tplot;                   Постоянная x координата когда нет ветра.
xnowind(1:length(y))=x0;          Постоянная x координата когда нет ветра.
plot3(x,y,z,'k-',xnowind,y,z,'k--')   Два 3D графика траекторий.
grid on
axis([0 6000 0 6000 0 2500])
xlabel('x (m)'); ylabel('y (m)'); zlabel('z (m)')

```

Ниже показан рисунок, сгенерированный программой.



Пример задачи 10.2. Электрический потенциал зарядов двух точек

Электрический потенциал V вокруг заряженной частицы задается формулой:

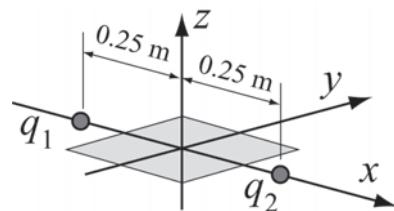
$$V = \frac{1}{4\pi\epsilon_0 r} q,$$

где $\epsilon_0 = 8.8541878 \times 10^{-12} \frac{C}{Nm^2}$ – диэлектрическая постоянная, q – величина заряда в кулонах и r – расстояние от точки до частицы в метрах. Электрическое поле двух или больше материальных частиц вычисляется с использованием наложения полей. Например, электрический потенциал в точке, определенный двумя частицами задается формулой

$$V = \frac{1}{4\pi\epsilon_0 r} \left(\frac{q_1}{r_1} + \frac{q_2}{r_2} \right),$$

где q_1 , q_2 , r_1 и r_2 – заряды материальных частиц и расстояния от данной точки до соответствующих частиц, соответственно.

Две частицы с зарядами $q_1 = 2 \times 10^{-10}$ Кл и $q_2 = 3 \times 10^{-10}$ Кл расположены в xy -плоскости в точках $(0.25, 0, 0)$ и $(-0.25, 0, 0)$, соответственно, как показано на рисунке. Вычислить и графически изобразить электрический потенциал в точках xy плоскости, определенный этими двумя частицами в области $-0.2 \leq x \leq 0.2$ и $-0.2 \leq y \leq 0.2$ (единицы измерения в xy -плоскости – метры). Сделайте график так, что, плоскость xy является плоскостью точек, а ось z – величина электрического потенциала.



Решение

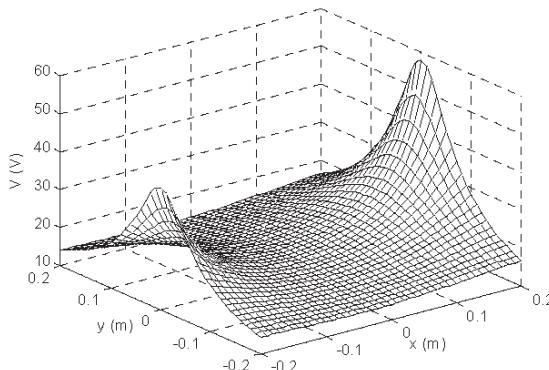
Задача решается в несколько этапов:

- Создание координатной сетки в плоскости xy в области $-0.2 \leq x \leq 0.2$ и $-0.2 \leq y \leq 0.2$.
- Вычисление расстояния от каждого узла сетки до каждого из зарядов.
- Вычисление электрического потенциала в каждой точке.
- Графическое изображение электрического потенциала.

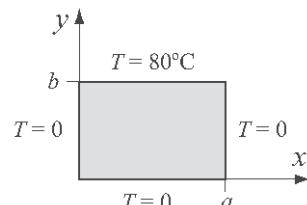
Далее представлена программа в скрипте-файле, которая решает эту задачу.

```
eps0=8.85e-12; q1=2e-10; q2=3e-10;
k=1/(4*pi*eps0);
x=-0.2:0.01:0.2;
y=-0.2:0.01:0.2;
[X,Y]=meshgrid(x,y);      Создание координатной сетки в плоскости xy.
r1=sqrt((X+0.25).^2+Y.^2);    Вычисление расстояний  $r_1$  и  $r_2$  для каждого
r2=sqrt((X-0.25).^2+Y.^2);    узла сетки до частиц.
V=k*(q1./r1+q2./r2);        Вычисление электрического потенциала  $V$ 
mesh(X,Y,V)                  в каждой точке.
xlabel('x (m)'); ylabel('y (m)'); zlabel('V (V)')
```

График, созданный программой в результате ее работы:

**Пример задачи 10.3. Теплопроводность квадратной пластины**

Три стороны прямоугольной пластины ($a = 5$ м, $b = 4$ м) сохраняют температуру 0°C , а одна сторона имеет температуру $T_1 = 80^\circ\text{C}$, как показано на рисунке. Определить и графически изобразить температурное распределения $T(x,y)$ в пластине.



Решение

Распределение температуры $T(x,y)$ в пластине может быть определено решением двумерного уравнения теплопроводности. Для заданных граничных условий функция $T(x,y)$ может быть выражена аналитически рядом Фурье (Erwin Kreyszig, Advanced Engineering Mathematics, John Wiley and Sons, 1993):

$$T(x,y) = \frac{4T_1}{\pi} \sum_{n=1}^{\infty} \frac{\sin\left[(2n-1)\frac{\pi x}{a}\right]}{2n-1} \frac{\sinh\left[(2n-1)\frac{\pi y}{a}\right]}{\sinh\left[(2n-1)\frac{\pi b}{a}\right]}.$$

Ниже приведена программа в скрипт-файле, которая решает эту задачу. Эта программа выполняется за несколько шагов:

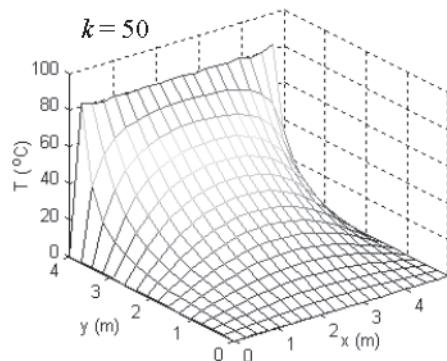
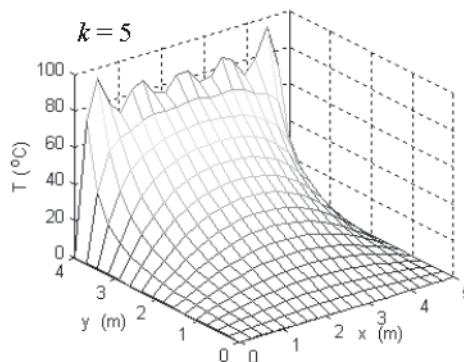
1. Создание X , Y координатной сетки в области $0 \leq x \leq a$ и $0 \leq y \leq b$. Длина пластины, a , делится на 20 сегментов, а ширина пластины, b , делится на 16 сегментов.
2. Вычисление температуры в каждой точке решетки. Вычисления выполняются точка за точкой, используя двойной цикл. В каждой точке определяется температура суммированием k членов ряда Фурье.
3. Создание графика поверхности T .

```

a=5; b=4; na=20; nb=16; k=5; T0=80;
clear T
x=linspace(0,a,na);
y=linspace(0,b,nb);
[X,Y]=meshgrid(x,y);
for i=1:nb
    for j=1:na
        T(i,j)=0;
        for n=1:k
            ns=2*n-1;             Создание координатной сетки в плоскости xy.
            T(i,j)=T(i,j)+sin(ns*pi*X(i,j)/a).*sinh(ns*pi*Y(i,j)/a)/
            (sinh(ns*pi*b/a)*ns);   Первый цикл. i – индекс строки сетки.
            end                     Второй цикл. j – индекс столбца сетки.
        end                     Третий цикл. n – n-ый член ряда Фурье, k – номер
        T(i,j)=T(i,j)*4*T0/pi;   члена ряда.
    end
end
mesh(X,Y,T)
xlabel('x (m)'); ylabel('y (m)'); zlabel('T ( ^oC )')

```

Программа выполнялась дважды, сначала используя пять членов ($k = 5$) ряда Фурье, для вычисления температуры в каждой точке, а затем с $k = 50$. Графики-сетки, создаваемые при каждом выполнении программы, показаны на рисунках ниже. Температура должна быть равномерно 80°C при $y = 4$ м. Отметим эффект числа членов (k) на точность при $y = 4$ м.



10.6. Задачи

1. Положение движущейся материальной точки как функции времени задано формулами:

$$x = \left[\frac{t-15}{100} + 1 \right] \sin(3t), \quad y = \left[\frac{t-15}{100} + 1 \right] \cos(0.8t), \quad z = 0.4t^{3/2}.$$

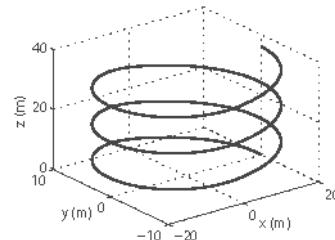
Изобразите положение точки для $0 \leq t \leq 30$.

2. Эллиптическая лестница, снижение которой может быть смоделировано параметрическими уравнениями

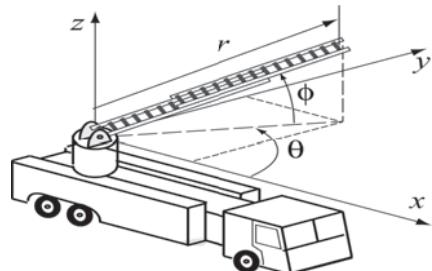
$$x = r \cos(t), \quad y = r \sin(t), \quad z = \frac{ht}{\pi n},$$

где $r = \frac{ab}{\sqrt{[b \cos(t)]^2 + [a \sin(t)]^2}}$, a и b – боль-

шая и полая полуоси эллипса, h – высота лестницы и n – число вращений, которые делает лестница. Создайте 3D график лестницы с $a = 20$ м, $b = 10$ м, $h = 18$ м и $n = 3$. (Создайте вектор t для области от 0 до $2\pi n$ и используйте команду `plot3`.)



3. Лестница пожарной машины может быть поднята (увеличение угла ϕ), повернута вокруг оси z (увеличение угла θ) и удлинена (увеличение r). Первоначальное положение лестницы – на пожарной машине ($\phi = 0$, $\theta = 0$ и $r = 8$ м). Затем лестница переводится в новое положение подня-



тием лестницы со скоростью 5 град/сек, вращением со скоростью 8 град/сек и удлинением лестницы со скоростью 0.6 м/с. Найдите координаты конечной точки лестницы в течение 10 секунд и постройте график.

4. Создайте 3D график поверхности функции $z = \frac{y^2}{4} - 2 \sin(1.5x)$ в области $-3 \leq x \leq 3$ и $-3 \leq y \leq 3$.
5. Создайте 3D график поверхности функции $z = 0.5x^2 + 0.5y^2$ в области $-2 \leq x \leq 2$ и $-2 \leq y \leq 2$.
6. Создайте 3D сеточный график функции $z = \frac{-\cos(2R)}{e^{0.2R}}$, где $R = \sqrt{x^2 + y^2}$ в области $-5 \leq x \leq 5$ и $-5 \leq y \leq 5$.
7. Создайте 3D график поверхности функции $z = \cos(x) \cos(\sqrt{x^2 + y^2}) e^{-|0.2x|}$ в области $-2\pi \leq x \leq 2\pi$ и $-\pi \leq y \leq \pi$.
8. Создайте график конуса мороженого, показанного на рисунке. Конус имеет 8 дюймов высоты и 4 дюйма диаметра основания. Сверху мороженое имеет форму поусфера диаметра 4 дюйма.

Параметрическое уравнение для конуса:

$$x = r \cos(\theta), \quad y = r \sin(\theta), \quad z = 4r,$$

где $0 \leq \theta \leq 2\pi$ и $0 \leq r \leq 2$.

Параметрическое уравнение сферы:

$$x = r \cos(\theta)\sin(\phi), \quad y = r \sin(\theta)\sin(\phi), \quad z = r \cos(\phi),$$

где $0 \leq \theta \leq 2\pi$ и $0 \leq \phi \leq \pi$.

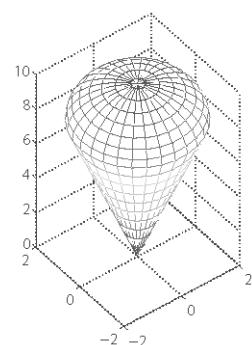
9. Уравнение Ван-дер-Ваальса дает соотношение между давлением P (атм), объемом V (Л) и температурой T (К) для реального газа:

$$P = \frac{nRT}{V - b} - \frac{n^2 a}{V^2},$$

где n – это число молей, $R = 0.08206$ (Л·атм)/(моль·К), – газовая постоянная, a ($\text{Л}^2 \cdot \text{атм}/\text{моль}^2$) и b (Л/моль) – материальные константы.

Рассмотрите 1.5 молей азота ($a = 1.39$ Л 2 атм/моль 2 , $b = 0.03913$ Л/моль). Создайте 3D график, который показывает изменение давления (зависимая переменная, ось z) от объема (независимая переменная, ось x) и температуры (независимая переменная, ось y). Области изменения объема и температуры: $0.3 \leq V \leq 1.2$ Л и $273 \leq T \leq 473$ К.

10. Молекулы газа в контейнере двигаются с различными скоростями. Закон распределения скорости Максвелла дает вероятностное распределение $P(v)$ как функции температуры и скорости:



$$P(v) = 4\pi \left(\frac{M}{2\pi RT} \right)^{3/2} v^2 e^{(-Mv^2)/(2RT)},$$

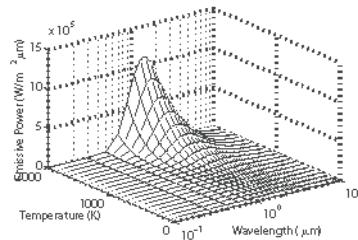
где M является молярной массой газа в кг/моль, $R = 8.31$ Дж/(моль·К) является газовой постоянной, T – температура в кельвинах и v – скорость молекулы в м/с.

Сделайте 3D график $P(v)$ как функции v и T для $0 \leq v \leq 1000$ м/с и $70 \leq T \leq 320$ К. для кислорода (молярная масса 0.032 кг/моль).

- 11.** Закон распределения Планка определяет излучательную способность абсолютно чёрного тела (количество испускаемой энергии излучения) как функцию температуры и длины волны:

$$E = \frac{C_1}{\lambda^5 [e^{C_2/(\lambda T)} - 1]} \left(\frac{W}{m^2 \mu m} \right),$$

где $C_1 = 3.742 \times 10^8$ Ватт·мкм⁴/м², $C_2 = 1.439 \times 10^4$ мкм·К, T – температура в градусах Кельвина (К) и λ – длина волны в мкм. Создайте 3D график (показанный на рисунке) E как функция λ ($0.1 \leq \lambda \leq 10$ мкм·К) и T для $100 \leq T \leq 2000$ К. Используйте логарифмическую шкалу для λ . Это может быть сделано командой: `set(gca, 'xscale', 'log')`.



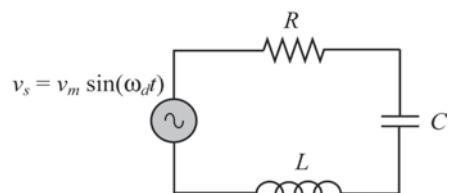
- 12.** Поток Q (м³/с) в прямоугольном канале задается уравнением Мэннинга:

$$Q = \frac{kdw}{n} \left(\frac{wd}{w+2d} \right)^{2/3} \sqrt{S},$$

где d – глубина воды (м), w – ширина канала (м), S – наклон канала (м/м), n – коэффициент шероховатости стен канала и k – коэффициент преобразования (равный 1, когда используются указанные выше единицы). Создайте 3D график Q (ось z) как функции w (ось x) для $0 \leq w \leq 8$ м и d (ось y) для $0 \leq d \leq 4$ м. Примите $n = 0.05$ и $S = 0.001$ м/м.

- 13.** Цепь RLC с источником переменного напряжения показана на рисунке. Исходное напряжение v_s задается формулой $v_s = v_m \sin(\omega_d t)$, где $\omega_d = 2\pi f_d$ и f_d – частота возбуждения. Амплитуда тока I , в этой цепи, описывается формулой:

$$I = \frac{v_m}{\sqrt{R^2 + (\omega_d L - 1/(\omega_d C))^2}},$$



где R и C – сопротивление резистора и емкость конденсатора, соответственно.

Для цепи на рисунке $C = 15 \times 10^{-6}$ Фарад, $L = 240 \times 10^{-3}$ Гн, и $v_m = 24$ В.

(a) Создайте 3D график I (ось z) как функции ω_d (ось x) для $60 \leq f_d \leq 110$ Гц и как функции R (ось y) для $10 \leq R \leq 40$ Ом.

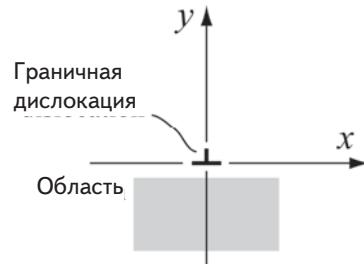
(b) Создайте график, который является проекцией на xz плоскость. Используя этот график оцените собственную я частота цепи (частота, при которой I максимальен). Сравните оценку с расчетным значением $1/(2\pi\sqrt{LC})$.

14. Дефект в кристаллической решетке, когда пропускается строка атомов, называется граничной дислокацией. Поле напряжений вокруг граничной дислокации задается формулой:

$$\sigma_{xx} = \frac{-Gb}{2\pi(1-\nu)} \frac{y(3x^2 + y^2)}{(x^2 + y^2)^2},$$

$$\sigma_{yy} = \frac{Gb}{2\pi(1-\nu)} \frac{y(x^2 - y^2)}{(x^2 + y^2)^2},$$

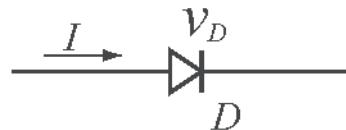
$$\tau_{xy} = \frac{Gb}{2\pi(1-\nu)} \frac{y(x^2 - y^2)}{(x^2 + y^2)^2},$$



где G – модуль сдвига, b – вектор Бюргерса и ν – коэффициент поперечной деформации Пуассона. Графически изобразите компоненты напряжений (каждую в отдельном рисунке Figure) возникающих вследствие граничной дислокации в алюминии, для которого $G = 27.7 \times 10^9$ Па, $b = 0.286 \times 10^{-9}$ м и $\nu = 0.334$. Графически изобразите напряжения в области $-5 \times 10^{-9} \leq x \leq 5 \times 10^{-9}$ м и $-5 \times 10^{-9} \leq y \leq -1 \times 10^{-9}$ м. Изобразите координаты x и y в горизонтальной плоскости и напряжения в вертикальном направлении.

15. Ток I текущий через полупроводниковый диод задается формулой

$$I = I_s \left(e^{\frac{qv_D}{kT}} - 1 \right),$$



где $I_s = 10^{-12}$ А – ток насыщения, $q = 1.6 \times 10^{-19}$ Кл – значение заряда электрона, $k = 1.38 \times 10^{-23}$ Дж/К – постоянная Больцмана, v_D – падение напряжения через диод и T – температура в кельвинах. Создайте 3D график I (ось z) как функции от v_D (ось x) для $0 \leq v_D \leq 0.4$ и от T (ось y) для $290 \leq T \leq 320$ К.

16. Уравнение для линий тока при равномерном обтекании цилиндра:

$$\psi(x, y) = y - \frac{y}{x^2 + y^2},$$

где ψ – функция тока. Например, если $\psi = 0$, то $y = 0$. Так как это уравнению выполняется для всех x , ось x есть нулевая ($\psi = 0$) линия тока. Заметим, что множество точек, где $x^2 + y^2 = 1$ также является линией тока. Таким образом, функция тока выше – для цилиндра радиуса 1. Создайте 2D контурный график линий тока вокруг цилиндра радиуса 1 дюйм. Область изменения x и y возьмите от -3 до 3 . Используйте 100 для числа контурных уровней. Добавьте к рисунку график окружности радиуса 1. Заметим, что MATLAB также графически изображает линии тока внутри цилиндра. Это математический артефакт.

- 17.** Отклонение w зафиксированной круговой мембранны радиуса r_d подвергнутой давлению P , задается формулой (теория малых деформаций):

$$w(r) = \frac{Pr_d^4}{64K} \left[1 - \left(\frac{r}{r_d} \right)^2 \right]^2,$$

где r – радиальная координата и $K = \frac{Et^3}{12(1-v^2)}$, где E , t и v являются модулем упругости, толщиной и коэффициентом Пуассона поперечной деформации мембранны, соответственно. Рассмотрите мембрану с параметрами $P = 15$ пси, $r_d = 15$ дюймов, $E = 18 \times 10^6$ пси, $t = 0.08$ дюймов, и $v = 0.3$. Создайте график поверхности мембранны.

- 18.** Модель Ферхольста, заданная следующим уравнением, описывает рост популяции, которая ограничена различными факторами, такими как перенаселённость и отсутствие ресурсов:

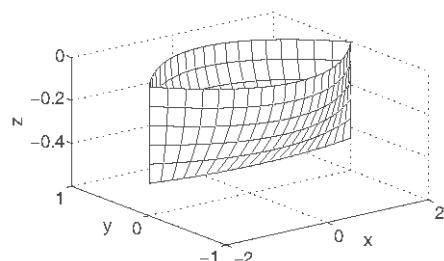
$$N(t) = \frac{N_\infty}{1 + \left(\frac{N_\infty}{N_0} \right) e^{-rt}},$$

где $N(t)$ – число индивидуумов в популяции, N_0 – начальная численность популяции, N_∞ – максимальная численность популяции вследствие возможных ограничивающих факторов сброс к различным ограничивающим факторам, и r – константа скорости процесса. Сделайте график поверхности $N(t)$ как функции от t и N_∞ , считая, что $r = 0.1$ сек $^{-1}$ и $N_0 = 10$. Пусть t меняется между 0 и 100, а N_∞ – между 100 и 1 000.

- 19.** Геометрия корпуса судна (корпус Уигли) может быть смоделирована уравнением

$$y = \mp \frac{B}{2} \left[1 - \left(\frac{2x}{L} \right)^2 \right] \left[1 - \left(\frac{2z}{T} \right)^2 \right],$$

где x , y и z – длина, ширина и высота, соответственно. Используйте MATLAB для создания 3D рисунка корпуса как показано на рисунке. Используйте $B = 1.2$, $L = 4$, $T = 0.5$, $-2 \leq x \leq 2$ и $-0.5 \leq z \leq 0$.

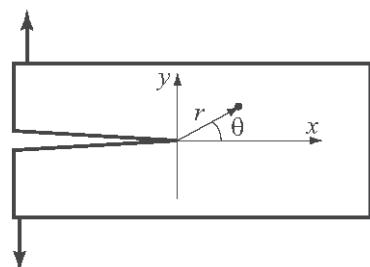


20. Поле напряжений вблизи вершины трещины в линейно упругом изотропном материале в режиме нагрузки I задается формулами:

$$\sigma_x = \frac{K_I}{\sqrt{2\pi r}} \cos\left(\frac{\theta}{2}\right) \left[1 - \sin\left(\frac{\theta}{2}\right) \sin\left(\frac{3\theta}{2}\right) \right],$$

$$\sigma_y = \frac{K_I}{\sqrt{2\pi r}} \cos\left(\frac{\theta}{2}\right) \left[1 + \sin\left(\frac{\theta}{2}\right) \sin\left(\frac{3\theta}{2}\right) \right],$$

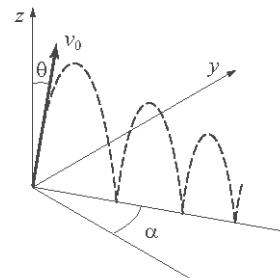
$$\tau_y = \frac{K_I}{\sqrt{2\pi r}} \cos\left(\frac{\theta}{2}\right) \sin\left(\frac{\theta}{2}\right) \cos\left(\frac{3\theta}{2}\right)$$



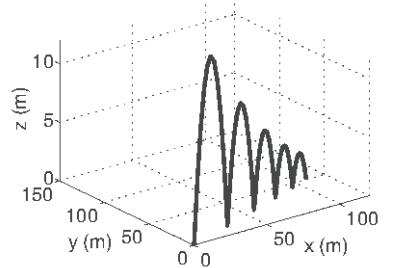
Для $K_I = 300$ кси·дюйм $^{1/2}$ графически изобразить напряжения (каждое в отдельном окне Figure) в области $0 \leq \theta \leq 90^\circ$ и $0.02 \leq r \leq 0.2$. Изобразите координаты x и y в горизонтальной плоскости, а напряжения – в вертикальном направлении.

21. Подброшенный шар отскакивает от пола и прыгает много раз. Для шара, подброшенного в направлении, показанном на первом рисунке, положение шара, как функция времени задается формулами:

$$x = v_x t, \quad y = v_y t, \quad z = v_z t - \frac{1}{2} g t^2.$$



Скорости в направлениях x и y постоянны во все время движения и задаются формулами: $v_x = v_0 \sin(\theta) \cos(\alpha)$, $v_y = v_0 \sin(\theta) \sin(\alpha)$. В вертикальном z -направлении начальная скорость $v_z = v_0 \cos(\theta)$, и когда шар отскакивает от пола, его скорость рикошета равна 0.8 от вертикальной скорости в начале предыдущего прыжка. Время между прыжками задается формулой $t_b = (2v_z)/g$, где v_z – вертикальная компонента скорости в начале прыжка. Создайте 3D график (как на втором рисунке), который показывает траекторию шара во время первых пяти прыжков. Возьмите $v_0 = 20$ м/с, $\theta = 30^\circ$, $\alpha = 25^\circ$ и $g = 9.81$ м/с 2 .





ГЛАВА 11.

Символьная математика

Все математические операции MATLAB, описанные в первых 10 главах, были численными. Эти операции выполнялись путем записи числовых выражений, которые могли содержать числа и переменные с присвоенными им численными значениями. Когда выполняется числовое выражение MATLAB, результат также является числовым (просто число или массив чисел). Число или числа являются или точными или имеют приближенные значения с плавающей запятой. Например, ввод $1/4$ дает точное значение 0.2500 , а ввод $1/3$ дает приближенное значение 0.3333 .

Многие приложения в математике, науке и технике требуют символьных операций, которые представляют собой математическими действиями с выражениями, содержащими символьные переменные (переменные, у которых нет определенных численных значений при выполнении операции). Результаты таких операций – также математические выражения в терминах символьных переменных. Рассмотрим например, решение алгебраического уравнения с несколькими переменными, когда требуется выразить одну из переменных через остальные. Если a , b и x – символьные переменные и $ax - b = 0$, тогда x может быть найдено через переменные a и b , а именно, $x = b/a$. Другие примеры символьных операций – это аналитическое дифференцирование или интегрирование математических выражений. Например, производная выражения $2t^3 + 5t - 8$ относительно t есть $6t^2 + 5$.

MATLAB имеет возможности выполнения многих типов символьных операций. Числовая часть символьных операций выполняется MATLAB точно, без приближенных численных значений. Например, результат сложения $\frac{x}{4}$ и $\frac{x}{3}$ есть $\frac{7}{12}x$, а не $0.5833x$.

Символьные операции могут выполняться в MATLAB только если установлен пакет расширения символьной математики Symbolic Math Toolbox. Этот пакет Symbolic Math Toolbox есть набор функций MATLAB, которые используются для выполнения символьных операций. Команды и функции для символьных операций имеют тот же самый стиль и синтаксис, что и для числовых операций. Сами символьные операции выполняются преимущественно в MuPad, которое является математическим программным обеспечением, созданным для этих целей. Программное обеспечение MuPad встроено в MATLAB и автоматически активи-

зируется при выполнении символьных функций MATLAB. MuPad может также использоваться в качестве отдельного независимого программного обеспечения. Тогда оно использует язык MuPAD, который имеет совершенно другую структуру и команды, чем MATLAB. Пакет символьной математики включен в студенческую версию MATLAB. В стандартной версии пакет покупается отдельно. Чтобы проверить, установлен ли Symbolic Math Toolbox на компьютере, пользователь может ввести команду `ver` в командном окне. В ответ MATLAB выводит на экран информацию об используемой версии, а также список установленных пакетов расширения.

Символьные операции начинаются с символьных объектов. Символьные объекты создаются из переменных и чисел, которые используются в математических выражениях, и указывают MATLAB выполнить выражение символьно. Как правило, пользователь сначала определяет (создает) необходимые символьные переменные (объекты), а затем использует их для создания символьных выражений, которые впоследствии используются в символьных операциях. Если необходимо, символьные выражения могут использоваться в численных операциях.

Первый раздел в этой главе описывает, как определить символьные объекты и как их использовать для создания символьных выражений. Второй раздел показывает, как изменить вид существующих выражений. Как только создано символьное выражение, оно может использоваться в математических действиях. MATLAB имеет большой набор функций для этих целей. Следующие четыре раздела (11.3–11.6) описывают, как использовать MATLAB для решения алгебраические уравнения, выполнения дифференцирования и интегрирования и для решения дифференциального уравнения. Раздел 11.7 покрывает графическое изображение символьных выражений. То, как использовать символьные выражения в последовательных численных расчетах, объяснено в следующем разделе 11.8.

11.1. Символьные объекты и символьные выражения

Символьный объект может быть переменной (без присвоенного числового значения), число, или выражение, созданное из символьных переменных и чисел. Символьное выражение – это математическое выражение, содержащее один или более символьных объектов. При вводе символьное выражение может быть похожим на стандартное числовое выражение. Однако, поскольку это выражение содержит символьные объекты, оно выполняется MATLAB символьно.

11.1.1. Создание символьных объектов

Символьные объекты могут быть переменными или числами. Они могут быть созданы командами `sym` и/или `syms`. Один символьный объект может быть создан командой `sym`:

```
object_name = sym('string')
```

где строка `string` является символьным объектом. Эта команда есть присвоение символьному объекту `string` имени `object_name`. Стока может быть:

- Одной буквой или комбинацией нескольких символов (без пробелов). Например: 'a', 'x', 'yad'.
- Комбинацией символов и цифр, начинающейся с символа и без пробелов. Например: 'xh12', 'r2d2'.
- Числом. Например: '15', '4'.

В первых двух случаях (где строка – одна буква, комбинация нескольких символов, или комбинация символов и цифр), символьный объект – это символьная переменная. В этом случае это удобно (но не обязательно дать символьному объекту то же самое имя, что и у строки).

Например, `a`, `bb` и `x` могут быть определены как символьные переменные следующим образом:

```
>> a=sym('a')          Создание символьного объекта a и присвоение ему имени a.
a =
a ←                                         При выводе на экран символьный объект расположен
>> bb=sym('bb')          без отступа.
bb =
bb ←
>> x=sym('x');           Символьная переменная x создана, но не выведена на экран,
>>                      так как в конце команды введена точка с запятой.
```

Имя символьного объекта может отличаться от имени переменной. Например:

```
>> g=sym('gamma')        Символьный объект есть gamma, а имя объекта – g.
g =
gamma
```

Как отмечено выше, символьные объекты могут быть также числами. Эти числа не должны вводиться как строки. Например, ниже используется команда `sym` для создания символьных объектов из чисел 5 и 7 и присвоения их переменным `c` и `d`, соответственно.

```
>> c=sym(5)            Создание символьного объекта числу 5 и присвоение ему имени c.
c =
5 ←
>> d=sym(7)            При выводе на экран символьный объект расположен
d =                         без отступа.
7 ←
```

Как показано выше, когда создается символьный объект и точка с запятой не введена в конце команды, то MATLAB выводит на экран имя объекта и непосредственно сам объект на следующих двух линиях. Отображение на экране символьных объектов начинается в начале командной строки и без отступа, как отображаются числовые переменные. Это отличие иллюстрируется ниже, где создается числовая переменная.

```
>> e=13      Число 13 присвоено переменной e (числовая переменная).
e =
13       Отображение значения числовой переменной a делается с отступом.
```

Несколько символьных переменных могут быть созданы одной командой `syms`, которая имеет вид:

```
syms variable_name variable_name variable_name
```

Эта команда создает символьные объекты, которые имеют те же самые имена, что и символьные переменные. Например, переменные y, z и d могут все быть созданы как символьные переменные в одной команде:

```
>> syms y z d
>> y
y =
```

Переменные, создаваемые командой `syms`, не выводятся на экран автоматически. Введение имени переменная показывает, что эта переменная создана.

Когда выполняется команда `syms`, переменные, которые она создает, не выводятся на экран автоматически – даже если в конце команды не введена точка с запятой.

11.1.2. Создание символьных выражений

Символьные выражения – это математические выражения, записанные с использованием символьных переменных. Как только созданы символьные переменные, они могут использоваться для создания символьных выражений. Символьное выражение – это символьный объект (выводится на экран без отступа). Формат создания символьных выражений:

```
Expression_name= Mathematical expression
```

Несколько примеров:

```
>> syms a b c x y
>> f=a*x^2+b*x + c
f =
a*x^2 + b*x + c
```

Определение a, b, c, x и y как символьных переменных.
Создание символьного выражения $ax^2 + bx + c$
и присвоение ему имени f.
Вывод на экран символьного выражения без отступа.

Когда вводится символьное выражение, включающее математические операции, которые могут быть выполнены (сложение, вычитание, умножение, и деление), MATLAB выполняет эти операции при создании выражение. Например:

```
>> g=2*a/3+4*a/7-6.5*x+x/3+4*5/3-1.5
    Введено  $\frac{2a}{3} + \frac{4a}{7} - 6.5x + \frac{x}{3} + 4 \cdot \frac{5}{3} - 1.5$ .
g =
(26*a)/21 - (37*x)/6 + 31/6
На экране отображается исполненное  $\frac{26a}{21} - \frac{37x}{6} + \frac{31}{6}$ .
```

Отметим, что все вычисления выполнены точно, без численного приближения.

В последнем примере $\frac{2a}{3}$ и $\frac{4a}{7}$ были сложены MATLAB и получилось $\frac{26a}{21}$, далее, $-6.5x + \frac{x}{3}$ были сложены и получилось $-\frac{37x}{6}$. Операции с членами в символьном выражении, которые содержат только числа, также выполняются точно. В последнем примере, $4 \cdot \frac{5}{3} - 1.5$ заменено на $\frac{31}{6}$.

Отличие между точными и приближенными вычислениями демонстрируется в следующем примере, где те же самые математические действия выполнены сначала с символьными переменными, а потом — с числовыми переменными.

```
>> a=sym(3); b=sym(5);
>> e=b/a+sqrt(2)
e =
 $2^{(1/2)} + 5/3$ 
>> c=3; d=5;
>> f=d/c+sqrt(2)
f =
3.0809
Определение a и b как символьных 3 и 5, соответственно.
Создание выражения, которое включает a и b.
На экран выведено точное значение e как
символьный объект (без отступа).
Определение c и d как чисел 3 и 5, соответственно.
Создание выражения, которое включает a и b.
На экран выведено приближенное значение f
как число (с отступом).
```

Создаваемое выражение может включать и символьные объекты и числовые переменные. Однако, если выражение будет включать символьный объект (или несколько), то все математические действия будут выполняться точно. Например, если в последнем выражении число с заменить на символьное a, то результат будет точным, как это было в первом примере.

```
>> g=d/a+sqrt(2)
g =
 $2^{(1/2)} + 5/3$ 
```

Дополнительные факты о символьных выражениях и символьных объектах

- Символьные выражения могут включать числовые переменные, которые были получены при выполнении числовых выражений. Когда эти переменные вставлены в символьные выражения, используется их точное значение, даже если переменная была ранее выведена на экран с приближенным значением. Например:

```
>> h=10/3          h определена как 10/3 (числовая переменная).
h =               На экран выведено приближенное значение h (число).
3.3333            На экран выведено приближенное значение h (число).
>> k=sym(5); m=sym(7);   Определение k и m как символьных 5 и 7, соответственно.
>> p=k/m+h      h, k и m используются в выражении.
p =               В определении p используется точное значение h.
85/21             На экран выведено точное значение p (символьный объект).
```

Для преобразования в числовую форму символьного выражения (объекта) *s*, которое записано в точной форме, может использоваться команда `double(s)`. (Название «`double`» происходит из того факта, что команда эта возвращает число с двойной точностью с плавающей запятой, представляющее значение *s*.) Приведем два примера. В первом *p* из последнего примера преобразуется в числовую форму. Во втором создается символьный объект и затем он преобразуется в числовую форму.

```
>> pN=double(p)      p преобразуется в числовую форму (с присвоением pN).
pN =
    4.0476
>> y=sym(10)*cos(5*pi/6)      Создание символьного выражения y.
y =
-5*3^(1/2)
>> yN=double(y)      На экран выведено точное значение y.
yN =
    -8.6603           y преобразовано в числовую форму (с присвоением yN).
```

- Создаваемый символьный объект может быть также символьным выражением, записанным с использованием переменных, которые первоначально не создавались как символьные объекты. Например, квадратный трехчлен может быть создан как символьный объект, названный *f*, использованием команды `sym` следующим образом:

```
>> f=sym('a*x^2+b*x+c')
f =
a*x^2 + b*x + c
```

Важно понимать, что в этом случае переменные a , b , c и x , включенные в объект, не существуют индивидуально как независимые символьные объекты (целое выражение есть один объект). Это означает, что невозможно выполнить символьные математические операции с индивидуальными переменными в объекте. Например, невозможно дифференцировать f по переменной x . Это отличается от способа, как квадратный трехчлен создавался в первом примере этого раздела, когда сначала создавались индивидуальные переменные как символьные объекты и затем они использовались в квадратном трехчлене.

- Существующие символьные выражения могут использоваться для создания новых символьных выражений. Это делается просто использованием имени существующего выражения в новом выражении. Например:

<pre>>> syms x y >> SA=x+y, SB=x-y SA = x+y SB = x-y >> F=SA^2/SB^3+x^2 F =</pre>	<p>Определение x и y как символьных переменных.</p> <p>Создание двух символьных выражений SA и SB.</p> <p>$SA = x + y$</p> <p>$SB = x - y$</p> <p>Создание нового символьного выражения F используя SA и SB.</p> $F = (SA^2) / (SB^3) + x^2 = \frac{(x + y)^2}{(x - y)^3} + x^2$
---	--

11.1.3. Команда *findsym* и значение символьной переменной по умолчанию

Команда *findsym* может использоваться для определения, какие символьные переменные присутствуют в данном символьном выражении. Формат команды:

findsym(S) или *findsym(S, n)*

Команда *findsym(S)* выводит на экран имена всех символьных переменных (разделенных запятыми), которые содержатся в выражении S в алфавитном порядке. Команда *findsym(S, n)* выводит на экран n символьных переменных, которые содержатся в выражении S в порядке по умолчанию. Для однобуквенных символьных переменных порядок по умолчанию начинается с x , и затем следуют символы в соответствии их близости к x . Если имеются два символа одинаково близких к x , то первым будет символ, который в алфавитном порядке стоит после x (т. е. y прежде w , и z прежде v). Символьная переменная по умолчанию в символьном выражении – это первая переменная в порядке по умолчанию. Символьная переменная по умолчанию в выражении S может быть определена командой *findsym(S, 1)*. Примеры:

```
>> syms x h w y d t      Определение x, h, w, y, d и t как символьных переменных.
>> S=h*x^2+d*y^2+t*w^2      Создание символьного выражения S.
S =
t*w^2 + h*x^2 + d*y^2
>> findsym(S)           Использование команды findsym(S).
ans =                   Символьные переменные выводятся на экран
d, h, t, w, x, y       в алфавитном порядке.
>> findsym(S,5)         Использование команды findsym(S,n), n=5.
ans =                   Символьные переменные выводятся на экран
x,y,w,t,h             в порядке по умолчанию.
> findsym(S,1)          Использование команды findsym(S,n), n=1.
ans =                   На экран выводится символьная переменная по умолчанию.
x
```

11.2. Изменение вида существующего символьного выражения

Символьные выражения создаются или пользователем, или MATLAB в результате символьных операций. Выражения, создаваемые MATLAB, могут получиться не в самом простом виде, или в не в том виде, который предпочитает пользователь. Вид существующего символьного выражения может быть изменен приведением подобных с одной степенью, раскрытием скобок в произведении, вынесением общих множителей за скобки, использованием математических и тригонометрических тождеств, и многими другими операциями. Следующие подразделы описывают некоторые из команд, которые могут использоваться для изменения вида существующего символьного выражения.

11.2.1. Команды *collect*, *expand* и *factor*

Команды *collect*, *expand* и *factor* могут использоваться для выполнения математических действий, соответствующих их названиям (приведение подобных, разложение и представление в виде множителей).

Команда *collect*

Команда *collect* собирает члены в выражении, в котором есть переменная одной степени. В новом выражении члены будут упорядочены в порядке убывания степеней. Команда имеет формат:

collect(S) или *collect(S, variable_name)*

где *S* – символьное выражение. Команда в виде *collect(S)* работает лучше, когда в выражении есть только одна символьная переменная. Если выражение имеет более одной символьной переменной, то MATLAB сначала соберет члены с одной переменной, затем второй переменной, и так далее. Порядок переменных опреде-



ляется MATLAB. Пользователь может сам указать первую переменную используя команду в виде `collect(S, variable name)`. Примеры:

```

>> syms x y
>> S=(x^2+x-exp(x))*(x+3)
S =
(x + 3)*(x - exp(x) + x^2)
>> F = collect(S)
F =
MATLAB возвращает выражение  $x^3 + 4x^2 + (3 - e^x)x - 3e^x$ .
x^3+4*x^2+(3-exp(x))*x-3*exp(x)
>> T=(2*x^2+y^2)*(x+y^2+3)
T =
Создание символьного выражения Т  $(2x^2 + y^2)(y^2 + x + 3)$ .
(2*x^2+y^2)*(y^2+x+3)
>> G=collect(T)
G =
MATLAB возвращает выражение  $2x^3 + (2y^2 + 6)x^2 + y^2x + y^2(y^2 + 3)$ .
2*x^3+(2*y^2+6)*x^2+y^2*x+y^2*(y^2+3)
>> H=collect(T,y)
H =
MATLAB возвращает выражение  $y^4 + (2x^2 + x + 3)y^2 + 2x^2(x + 3)$ .
y^4+(2*x^2+x+3)*y^2+2*x^2*(x+3)

```

Заметим, что когда используется `collect(T)`, преобразованное выражение записано в порядке уменьшения степеней переменной x , но когда используется `collect(T, y)`, преобразованное выражение записано в порядке уменьшения степеней переменной y .

Команда *expand*

Команда `expand` раскладывает выражения двумя способами. Она выполняет произведения членов выражения, которые содержат суммирование (используемое по крайней мере одним из членов) и она использует тригонометрические тождества и экспоненциальные и логарифмические законы для разложения соответствующих членов, которые включают суммирование. Формат команды:

`expand(S)`

где s – символьное выражение. Два примера:

>> syms a x y	Определение a, x и y как символьных переменных.
>> S=(x+5)*(x-a)*(x+4)	Создание символьного выражения $-(a - x)(x + 4)(x + 5)$ и присвоение его переменной S.
S = - (a-x) * (x+4) * (x+5)	
>> T=expand(S)	Использование команды expand(S).
T = 20*x-20*a-9*a*x-a*x^2+9*x^2+x^3	MATLAB возвращает выражение $20x - 20a - 9ax - ax^2 + 9x^2 + x^3$.
>> expand(sin(x-y))	Использование команды expand для разложения $\sin(x - y)$.
ans = cos(v)*sin(x)-cos(x)*sin(v)	MATLAB использует тригонометрическое тождество для разложения.



Команда **factor**

Команда **factor** преобразует выражение, которое является многочленом, в произведению многочленов более низкой степени (раскладывает на сомножители). Формат команды:

factor (S)

где S – символьное выражение. Пример:

```
>> syms x
>> S=x^3+4*x^2-11*x-30
S =
x^3+4*x^2-11*x-30
>> factor(S)
ans =
(x+5)*(x-3)*(x+2)
```

Определение x как символьной переменной.
Создание символьного выражения
 $x^3 + 4x^2 - 11x - 30$ и присвоение его переменной S.
Использование команды **factor (S)**.
MATLAB возвращает выражение $(x + 5)(x - 3)(x + 2)$.

11.2.2. Команды **simplify** и **simple**

Команды **simplify** и **simple** – это общие средства для упрощения вида выражения. Команда **simplify** использует встроенные правила упрощения для создания более простого вида выражения чем исходное. Команда **simple** запрограммирована так, чтобы создавать форму выражения с наименьшим количеством числа символов. Хотя нет никакой гарантии, что форма выражения с наименьшим количеством числа символов будет самой простой, в действительности это часто именно так.

Команда **simplify**

Команда **simplify** использует математические действия (сложение, умножение, правила сложения дробей, возведения в степень, логарифмирования и т. д.) и функциональные и тригонометрические тождества для создания более простого вида выражения. Формат команды **simplify**:

simplify (S)

где S является именем существующего выражения для упрощения, или вместо S может быть введено выражение для упрощения.

Два примера:

```
>> syms x y
>> S=(x^2+5*x+6) / (x+2)
S =
(x^2+5*x+6) / (x+2)
>> SA = simplify(S)
```

Определение x и y как символьных переменных.
Создание символьного выражения
 $(x^2 + 5x + 6)/(x + 2)$ и присвоение его переменной S.
Использование команды **simplify (S)**. ➔

```
SA =
x+3
>> simplify((x+y)/(1/x+1/y))
ans =
x*y
```

MATLAB упрощает выражения до $x + 3$.
 Упрощение выражения $(x + y) / \left(\frac{1}{x} + \frac{1}{y}\right)$.
 MATLAB упрощает выражения до xy .

Команда *simple*

Команда *simple* находит вид выражения с наименьшим количеством числа символов. Во многих случаях этот вид является также самым простым. Когда выполняется эта команда, MATLAB создает несколько форм выражения, применяя команды *collect*, *expand*, *factor* и *simplify*, и некоторые другие функции упрощения, которые здесь не рассматриваются. Затем MATLAB возвращает выражение самого короткого вида. У команды *simple* есть следующие три формы:

<code>F = simple(S)</code>	<code>simple(S)</code>	<code>[F how] = simple(S)</code>
----------------------------	------------------------	----------------------------------

Самая короткая форма выражения *S* присвоена переменной *F*.

Весь путь упрощения выведен на экран. Самое короткое выражение присвоено *ans*.

Самая короткая форма выражения *S* присвоена *F*. Имя (строка) использованного метода упрощения присвоено переменной *how*.

Разница между этими формами заключается в выводе. Ниже показано использование двух из этих форм.

```
>> syms x
>> S=(x^3-4*x^2+16*x)/(x^3+64)
S =
(x^3-4*x^2+16*x)/(x^3+64)
```

Определение *x* как символьной переменной.
 Создание символьного выражения

$$\frac{x^3 - 4x^2 + 16x}{x^3 + 64}$$
 и присвоение его переменной *S*.

```
>> F = simple(S)      Использование для упрощения команды F=simple(S).
F =                  Самый простой вид S есть  $x/(x + 4)$  и присвоен F.
x/(x+4)
```

Использование команды *[G how]=simple(S)*.
 Самый простой вид *S* есть $x/(x + 4)$ и присвоен *G*.

```
how =      Слово «simplify» присвоено how, это означает, что самая короткая
simplify   форма была получена использованием команды simplify.
```

Использование формы *simple(S)* команды не демонстрируется, потому что вывод результатов ее работы будет длинным. MATLAB выводит на экран 10 различных попыток и присваивает ответу *ans* самый короткий вид. Читателю рекомендуется попытаться выполнить эту команду и просмотреть вывод на экран результата.

11.2.3. Команда pretty

Команда `pretty` выводит на экран символьное выражение в виде, напоминающем математический формат, в котором обычно печатаются выражения. Команда имеет форму

$$\text{pretty}(S)$$

Пример:

```
>> syms a b c x          Определение a, b, c и x как символьных переменных.
>> S=sqrt(a*x^2 + b*x + c)    Создание символьного выражения
S =                                $\sqrt{ax^2 + bx + c}$  и присвоение его переменной S.
(a*x^2+b*x+c)^(1/2)
>> pretty(S)
           2                  1/2
           (a x + b x + c)      Команда pretty выводит на экран
                           выражение в математическом формате.
```

11.3. Решение алгебраических уравнений

Одно алгебраическое уравнение может быть решено относительно одной переменной, а система уравнений может быть решена относительно нескольких переменных функцией `solve`.

Решение одного уравнения

Алгебраическое уравнение может иметь одну или несколько символьных переменных. Если уравнения имеет одну переменную, оно решается числено. Если уравнение имеет несколько символьных переменных, то решение может быть получено относительно любой из переменных через другие переменные. Решение получается с использованием команды `solve`, которая имеет форму:

$$h = \text{solve}(eq) \quad \text{или} \quad h = \text{solve}(eq, var)$$

- Аргумент `eq` может быть именем ранее созданного символьного выражения, или введенным выражением. Когда `eq` есть ранее созданное символьное выражение `S`, или когда выражение, введенное на место `eq` не содержит знак `\approx` , то MATLAB решает уравнение `eq = 0`.
- Уравнение вида $f(x) = g(x)$ формы также может быть решено, если ввести это уравнение (включая знак `\approx`) как строку (`string`) на место `eq`.
- Если уравнение, которое будет решаться, имеется более одной переменной, команда `solve(eq)` решает его относительно символьной переменной по умолчанию (см. раздел 11.1.3). Решение относительно любой из переменных может быть получено командой `solve(eq, var)`, указывая имя переменной `var`.

- Если пользователь вводит `solve(eq)`, то решение присваивается переменной `ans`.
- Если уравнения имеет более одного решения, результат `h` есть символьный вектор-столбец с элементами решениями. Элементы этого вектора – символьные объекты. Когда массив символьных объектов выводится на экран, каждая строка располагается с отступом в один пробел (см. следующие примеры).

Следующие примеры поясняют использование команды `solve`.

```
>> syms a b x y z          Определение a, b, x, y и z как символьных переменных.
>> h=solve(exp(2*z)-5)    Использование команды solve для решения
h =                         уравнения  $e^{2z} - 5 = 0$ .
log(5)/2                     Решение присвоено переменной h.
log(5)/2
>> S=x^2-x-6              Создание символьного выражения  $x^2 - x - 6$  и присвоение его переменной S.
x^2-x-6
>> k=solve(S)             Использование команды solve для решения уравнения  $x^2 - x - 6 = 0$ .
k =                         Уравнение имеет два решения. Они присвоены
-2                           вектор-столбцу k символьных объектов.
 3
>> solve('cos(2*y)+3*sin(y)=2') Использование команды solve для решения
ans =                          $\cos(2y) + 3\sin(y) = 2$ . (Это уравнение введено
    pi/2                       в команду как строка.)
    pi/6
    (5*pi)/6                   Решение присвоено переменной ans.
>> T= a*x^2+5*b*x+20      Создание символьного выражения  $ax^2 + 5bx + 20$ 
T =                         и присвоение его переменной T.
a*x^2+5*b*x+20
>> solve(T)                Использование команды solve для решения T = 0.
ans =                         Уравнение T = 0 решается относительно x,
- (5*b+5^(1/2)*(5*b^2-16*a)^(1/2))/(2*a)   которая является
- (5*b-5^(1/2)*(5*b^2-16*a)^(1/2))/(2*a)   переменной по умолчанию.
>> M = solve(T,a)          Использование команды solve(eq,var) для решения T = 0.
M =                         Уравнение T = 0 решено относительно переменной a.
- (5*b*x+20)/x^2
```

- Также можно использовать команду `solve`, когда уравнение для решения вводится в виде строки с переменными, которые не определены заранее как символьные объекты. Однако, если решение содержит переменные (когда уравнение имеет более одной переменной), эти переменные не существуют как независимые символьные объекты. Например:

```
>> ts=solve('4*t*h^2+20*t-5*g')  Выражение  $4th^2 + 20t - 5g$  вписывается
                                     в команду solve. Переменные t, h и g не
                                     создавались ранее как символьные переменные.
ts =                               MATLAB решает уравнение  $4th^2 + 20t - 5g = 0$  относительно t.
(5*g)/(4*h^2+20)
```



Уравнение может также быть решено относительно другой переменной. Например, решение относительно g получается так:

```
>> gs=solve('4*t*h^2+20*t-5*g','g')
gs =
(4*t*h^2)/5 + 4*t
```

Решение системы уравнений

Команда `solve` может также использоваться для решения систем уравнений. Если число уравнений и число переменных – одно и то же, то решение численное. Если число переменных больше числа уравнений, тогда решение является символьным относительно требуемых переменных и выражает их через другие переменные. У системы уравнений (в зависимости от типа уравнений) может быть одно или несколько решений. Если у системы есть одно решение, то каждая из переменных, относительно которых решена система, имеет одно численное значение (или выражение). Если система имеет более одного решения, тогда у каждой из переменных может быть несколько значений.

Формат команды `solve` для решения системы из n уравнений:

```
output = solve(eq1,eq2,...,eqn)
```

или

```
output = solve(eq1,eq2,...,eqn,var1,var2,...,varn)
```

- Аргументы `eq1`, `eq2`, ..., `eqn` являются уравнениями, которые будут решаться. Каждый такой аргумент может быть именем ранее созданного символьного выражения, или выражения, которое введено в форме строки (`string`). В случае введенного ранее созданного символьного выражение `s`, решается уравнение `s = 0`. Если введена строка, которая не содержит знак `==>`, то решается уравнение `expression = 0`. Уравнение, которое содержит знак `==>`, должно быть введено как строка.
- В случае первого формата, если число уравнений n равно числу переменных в уравнениях, MATLAB дает численное решение для всех переменных. Если число переменных больше чем число уравнений n , MATLAB дает решение для n переменных через остальные переменные. Переменные, относительно которых будут получены решения, выбираются MATLAB в соответствии с порядком переменных по умолчанию (раздел 11.1.3).
- Когда число переменных больше чем число уравнений n , пользователь может выбрать переменные, относительно которых будет решаться система. Это делается с использованием второго формата команды `solve` путем введения имен этих искомых переменных `var1`, `var2`..., `varn`.

Результат команды `solve`, который является решением системы, может иметь две различных формы. Одна из форм есть массив ячеек, а другая имеет тип структуры. Массив ячеек – это массив, в котором каждый из элементов может быть мас-

сивом. Структура – это массив, в котором элементы (называемые полями) адресуются текстовыми указателями (именами полей). Поля структуры могут быть массивами различных размеров и типов. Массивы ячеек и структуры не представлены подробно в этой книге, однако ниже дано короткое объяснение так, чтобы читатель был в состоянии их использовать с командой `solve`.

Когда используется массив ячеек в выводе команды `solve`, эта команда должна иметь следующий вид (в случае системы трех уравнений):

```
[varA, varB, varC] = solve(eq1, eq2, eq3)
```

- После выполнения команды `solve` решение присваивается переменным, `varA`, `varB` и `varC`, и переменные выводятся на экран с присвоенным им решением. У каждой из переменных будут одно или несколько значений (в виде вектор-столбца) в зависимости от того, имеет ли система уравнений одно или несколько решений.
- Пользователь может выбрать любые имена для `varA`, `varB` и `varC`. MATLAB присваивает решения этим переменным в уравнениях в алфавитном порядке. Например, если переменные, относительно которых решены уравнения, есть x , u и t , то решение для переменной t присваивается `varA`, решение для u присваивается `varB`, и решение для x присваивается `varC`.

Следующие примеры показывают использование команды `solve` для случая, когда для результата используется массив ячеек:

```
>> syms x y t
>> S=10*x+12*y+16*t;
>> [xt yt]=solve(S, '5*x-y=13*t')
          Определение x, y и t как символьных переменных.
          Присвоение S выражения 10x + 12y + 16t.
          Использование команды solve
          для решения системы уравнений:
          10x + 12y + 16t = 0 и 5x - y = 13t.

xt =
2*t
yt =
-3*t
          Результат в массиве ячеек с двумя ячейками xt и yt.
          Решения для x и y присвоены xt и yt, соответственно.
```

Заметим, что в этом примере система двух уравнений решена MATLAB относительно x и y через t , так как x и y – это первые две переменные в порядке по умолчанию. Однако система может быть решена относительно других переменных. В качестве следующего примера система решается относительно y и t через x (используя вторую форму команды `solve`):

```
>> [tx yx]=solve(S, '5*x-y=13*t', y, t)
          Указаны переменные (y и t)
          относительно которых решается система.
tx =
x/2
yx =
-(3*x)/2
          Решения для указанных переменных присвоены в алфавитном порядке.
          Первая ячейка tx содержит решения для t, а вторая ячейка содержит
          решение для y.
```

Когда для вывода результата команды `solve` используется структура, эта команда имеет вид (в случае системы трех уравнений)

```
AN = solve(eq1, eq2, eq3)
```

- `AN` – имя структуры.
- При выполнении команды решение присваивается `AN`. MATLAB выводит на экран имя структуры и имена полей структуры, которые являются именами переменных, относительно которых решены уравнения. Размер и тип каждого поля выводится на экран рядом с именем поля. Содержание каждого поля, которое является решением для переменной, не выводится на экран.
- Для вывода на экран содержания поля (решение для переменной), пользователь должен ввести адрес поля. Общая форма для введения адреса: `structure_name.field_name` (см. пример ниже).

Для иллюстрации вывода результата в форме структуры решим снова систему уравнений последнего примера.

```
>> syms x y t
>> S=10*x+12*y+16*t;
>> AN=solve(S, '5*x-y=13*t')  Использование команды solve для решения
                                 системы уравнений: 10x + 12y + 16t = 0 и 5x - y = 13t.
AN =                               MATLAB выводит на экран имя структуры AN и имена
x: [1x1 sym]                     ее полей x и y (размер и тип), это имена тех переменных,
y: [1x1 sym]                     относительно которых решены уравнения.
>> AN.x                           Ввод адреса поля x.
ans =
2*t                                Содержание поля (решение для x) выводится на экран.
>> AN.y                           Ввод адреса поля y.
ans =
-3*t                                Содержание поля (решение для y) выводится на экран.
```

Типовая задача 11.1 показывает решение системы уравнений, у которой есть два решения.

Пример задачи 11.1. Пересечение круга и линии

Уравнение окружности в плоскости xy радиуса R и с центром в точке $(2, 4)$ имеет вид $(x - 2)^2 + (y - 4)^2 = R^2$. Уравнение линии на плоскости: $y = x/2 + 1$. Определить координаты точек (как функций R) пересечения линии и окружности.

Решение

Решение получается решением системы этих двух уравнений относительно x и y через переменную R . Чтобы показать отличие вывода результатов в виде массива ячеек и в форме структуры, система решается дважды. В первом случае решение выводится в массив ячеек:

```
>> syms x y R
>> [xc,yc]=solve ('(x-2)^2+(y-4)^2=R^2', 'y=x/2+1') Эти два уравнения
введены в команду solve.
xc = Результат в массиве ячеек.
((4*R^2)/5-64/25)^(1/2)+14/5 Результат в массиве ячеек с двумя ячейками
14/5-((4*R^2)/5-64/25)^(1/2) xc и yc. Каждая ячейка содержит два решения
yc = в виде символьного вектор-столбца.
((4*R^2)/5-64/25)^(1/2)/2+12/5
12/5-((4*R^2)/5-64/25)^(1/2)/2
```

Второе решение выводит результат в виде структуры:

```
>> COORD=solve ('(x-2)^2+(y-4)^2=R^2', 'y = x/2+1') Результат есть
структура с именем COORD.
COORD =
x: [2x1 sym] Результат в структуре COORD, имеет два поля x и y.
y: [2x1 sym] Каждое есть символьный вектор 2-на-1.
>> COORD.x Ввод адреса поля x.
ans = На экран выводится содержание поля (решение для x).
((4*R^2)/5-64/25)^(1/2)+14/5
14/5-((4*R^2)/5-64/25)^(1/2)
>> COORD.y Ввод адреса поля y.
ans = На экран выводится содержание поля (решение для y).
((4*R^2)/5-64/25)^(1/2)/2+12/5
12/5-((4*R^2)/5-64/25)^(1/2)/2
```

11.4. Дифференцирование

Символьное дифференцирование может быть выполнено при помощи команды `diff`. Формат этой команды:

`diff(S)` или `diff(S, var)`

- `S` может быть именем ранее созданного символьного выражения, или может быть введено непосредственно в команду.
- Если в команде `diff(S)` выражение содержит одну символьную переменную, то дифференцирование выполняется относительно этой переменной. Если выражение содержит более одной переменной, дифференцирование выполняется относительно переменной по умолчанию (раздел 11.1.3).
- В команде `diff(S, var)` (которая используется для дифференцирования выражений с несколькими символьными переменными) дифференцирование выполняется относительно переменной `var`.
- Вторая или более высокого порядка (n -го) производная может быть определена командами `diff(S, n)` или `diff(S, var, n)`, где n – положительное число. $n = 2$ для второй производной, $n = 3$ для третьей, и так далее.

Некоторые примеры:

```
>> syms x y t          Определение x, y и t как символьных переменных.
>> S=exp(x^4);         Присвоение S выражения  $e^{x^4}$ .
>> diff(S)             Использование команды diff для дифференцирования S.
ans =                  Выводится ответ:  $4x^3e^{x^4}$ .
4*x^3*exp(x^4)
>> diff((1-4*x)^3)   Использование команды diff для дифференцирования  $(1 - 4x)^3$ .
ans =                  Выводится ответ:  $-12(1 - 4x)^2$ .
-12*(1-4*x)^2
>> R=5*y^2*cos(3*t);  Присвоение R выражения  $5y^2\cos(3t)$ .
>> diff(R)             Использование команды diff для дифференцирования R.
ans =                  MATLAB дифференцирует R относительно y (символьная
10*y*cos(3*t)        переменная по умолчанию). Ответ  $10y\cos(3t)$  выводится на экран.
>> diff(R,t)           Использование команды diff для дифференцирования R
ans =                  относительно переменной t. Ответ  $-15y^2\sin(3t)$  выводится на экран.
-15*y^2*sin(3*t)
>> diff(S,2)            Использование команды diff для нахождения второй производной S.
ans =                  Ответ  $12x^2\exp(x^4) + 16x^6\exp(x^4)$  выводится на экран.
12*x^2*exp(x^4)+16*x^6*exp(x^4)
```

- Можно также использовать команду `diff`, когда выражение, которое будет дифференцироваться вводится прямо в команду как строка. При этом даже не требуется, что переменные в этом выражении заранее созданы как символьные объекты. Однако, отдельные переменные в этом выражении не существуют как независимые символьные объекты.

11.5. Интегрирование

Символьное интегрирование может быть выполнено с использованием команды `int`. Эта команда может использоваться для нахождения неопределенных интегралов (антипроизводные) и определенных интегралов. Для неопределенного интегрирования команда имеет вид:

`int(S) или int(S, var)`

- S может быть именем ранее созданного символьного выражения, или может быть введено непосредственно в команду.
- Если выражение S в команде `int(S)` содержит одну символьную переменную, то интегрирование выполняется относительно этой переменной. Если выражение содержит более одной переменной, интегрирование выполняется относительно символьной переменной по умолчанию (раздел 11.1.3).
- В команде `int(S, var)`, которая используется для интегрирования выражений с несколькими символьными переменными, интегрирование, выполняется относительно переменной var.

Некоторые примеры:

```
>> syms x y t          Определение x, y и t как символьных переменных.
>> S=2*cos(x)-6*x;    Присвоение S выражения  $2\cos(x) - 6x$ .
>> int(S)              Использование команды int(S) для интегрирования S.
ans =                   Выводится ответ:  $2\sin(x) - 3x^2$ .
2*sin(x)-3*x^2
>> int(x*sin(x))      Использование команды int(S) для интегрирования  $x\sin(x)$ .
ans =                   Ответ  $\sin(x) - x\cos(x)$  выводится на экран.
sin(x)-x*cos(x)
>>R=5*y^2*cos(4*t);   Присвоение R выражения  $5y^2\cos(4t)$ .
>> int(R)              Использование команды int(S) для интегрирования R. MATLAB
ans =                   интегрирует R относительно y (символьная переменная по умолчанию);
(5*y^3*cos(4*t))/3     Ответ  $(5y^3\cos(4t))/3$  выводится на экран.
>> int(R,t)            Использование команды diff для интегрирования R
ans =                   относительно переменной t.
(5*y^2*sin(4*t))/4     Ответ  $(5y^2\sin(4t))/4$  выводится на экран.
```

Для определенного интегрирования формат команды следующий:

`int(S,a,b)` или `int(S,var,a,b)`

- а и b – пределы интегрирования. Пределы могут быть как числами, так и символьными переменными.

Например, нахождение определенного интеграла $\int_0^\pi (\sin y - 5y^2) dy$ с MATLAB:

```
>> syms y
>> int(sin(y)-5*y^2,0,pi)
ans =
2 - (5*pi^3)/3
```

- Можно также использовать команду `int`, с вводом выражения которое будет интегрироваться, в виде строки (`string`), не задавая заранее переменные в выражении как символьные объекты. Однако, переменные в этом подынтегральном выражении не будут существовать как независимые символьные объекты.
- Интегрирование может быть иногда трудной задачей. Ответ в замкнутой форме (через известные функции), возможно не существует, или даже если он существует, MATLAB не всегда в состоянии найти его. Когда это случается MATLAB возвращает `int(S)` и сообщение: `Explicit integral could not be found`.



11.6. Решение обыкновенных дифференциальных уравнений

Обыкновенное дифференциальное уравнение (ОДУ) может быть решено символьно командой `dsolve`. Эта команда может использоваться для решения как одного уравнения, так и системы уравнений. Здесь рассматривается решение только одного уравнения. В главе 10 обсуждается использование MATLAB для решения ОДУ первого порядка численно. Предполагается знакомство читателя с общими понятиями дифференциальных уравнений. Цель этого раздела заключается в том, чтобы показать, как использовать MATLAB для решения таких уравнений.

ОДУ первого порядка – это уравнение, которое содержит производную искомой функции. Если t – независимая переменная, а y – зависимая переменная (искомая функция), то уравнение может быть записано в виде

$$\frac{dy}{dt} = f(t, y).$$

ОДУ второго порядка содержит вторую производную зависимой переменной (и может также содержать первую производную). Его общая форма:

$$\frac{d^2y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right).$$

Решение – это функция, которая удовлетворяет данному уравнению. Решение может быть общим или частным. Общее решение содержит неопределенные константы. В частном решении эти константы имеют определенные числовые значения такие, чтобы решение удовлетворяло заданным начальным или граничным условиям.

Команда `dsolve` может использоваться для нахождения общего решения или, когда начальные или граничные условия заданы, для того, чтобы получить частное решение.

Общее решение

Для нахождения общего решения команда `dsolve` применяется в виде:

```
dsolve('eq') или dsolve('eq', 'var')
```

- `eq` – это уравнение, которое будет решено. Оно должно быть введено как строка (даже если переменные – уже определены как символьные объекты).
- Переменные в уравнении не обязаны быть заранее созданными как символьные объекты. (Если они предварительно не были созданы, то в решении эти переменные не будут символьными объектами.)
- Для зависимой переменной может использоваться любой символ (строчного или верхнего регистра), кроме `D`.
- В команде `dsolve('eq')` MATLAB предполагает, что независимая переменная есть `t` (значение по умолчанию).

- В команде `dsolve('eq', 'var')` пользователь определяет независимую переменную, вводя ее как `var` (в виде строки).

При задании уравнения символ `D` обозначает дифференцирование. Если y – зависимая переменная, и t – независимая переменная, то `Dy` обозначает $\frac{dy}{dt}$. Например, уравнение $\frac{dy}{dt} + 3y = 100$ вводится как '`Dy + 3*y = 100`'.

Вторая производная вводится как `D2`, третья производная как `D3`, и так далее. Например, уравнение $\frac{d^2y}{dt^2} + 3\frac{dy}{dt} + 5y = \sin(t)$ вводится так:

$$'D2y + 3*Dy + 5*y = \sin(t)'.$$

- Переменные в уравнении ОДУ, которое вводится в команду `dsolve`, не должны быть заранее созданы как символьные переменные.
- При решении MATLAB использует символы `C1`, `C2`, `C3` и так далее, для непредeterminedных констант интегрирования.

Например, общее решение ОДУ первого порядка $\frac{dy}{dt} = 4t + 2y$ получается так:

```
>> dsolve('Dy=4*t+2*y')
ans =
C1*exp(2*t) - 2*t - 1
```

Ответ $y = C_1 \exp(2t) - 2t - 1$ выводится на экран.

Общее решение ОДУ второго порядка $\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + x = 0$ получается так:

```
>> dsolve('D2x+2*Dx+x=0')
ans =
C1/exp(t) + (C2*t)/exp(t)
```

Ответ $x = C_1 e^{-t} + C_2 t e^{-t}$ выводится на экран.

Следующие примеры поясняют решение дифференциальных уравнений, которые содержат символьные переменные в дополнение к независимым и зависимым переменным.

```
>> dsolve('Ds=a*x^2')
ans =
a*t*x^2 + C1
>> dsolve('Ds=a*x^2', 'x')
ans =
(a*x^3)/3 + C1
```

Независимая переменная – t (по умолчанию).

MATLAB решает уравнение $\frac{ds}{dt} = ax^2$.

Решение $s = ax^2 t + C_1$ выводится на экран.

Независимая переменная определена x .

MATLAB решает уравнение $\frac{ds}{dx} = ax^2$.

Решение $s = ax^3/3 + C_1$ выводится на экран.



```
>> dsolve('Ds=a*x^2', 'a')
ans =
(a^2*x^2)/2 + C2
```

Независимая переменная определена a .
MATLAB решает уравнение $\frac{ds}{da} = ax^2$.
Решение $s = a^2 x^2 / 2 + C_1$ выводится на экран.

Частное решение

Частное решение ОДУ может быть получено, когда определены граничные (или начальные) условия. Уравнение первого порядка требует одного условия, уравнение второго порядка требует двух условий, и так далее. Для нахождения частного решения команда `dsolve` имеет формат:

ОДУ первого порядка: `dsolve('eq', 'cond1', 'var')`

ОДУ высшего порядка: `dsolve('eq', 'cond1', 'cond2', ..., 'var')`

- Для решения уравнения высшего порядка в команду `dsolve` должны быть введены дополнительные граничные условия. Если число условий меньше чем порядок уравнения, то MATLAB возвращает решение, которое включает постоянные интегрирования (C_1, C_2, C_3 , и так далее).
- Граничные условия вводятся в виде строки, как показано ниже:

Математический вид	Формат MATLAB
$y(a) = A$	'y(a)=A'
$y'(a) = A$	'Dy(a)=A'
$y''a) = A$	'D2y(a)=A'

- Аргумент '`var`' является дополнительным и используется для определения независимой переменной в уравнении. Если он не введен, то принимается значение по умолчанию – t .

Например, ОДУ первого порядка $\frac{dy}{dt} + 4y = 60$ с начальным условием $y(0) = 5$ решается с MATLAB так:

```
>> dsolve('Dy+4*y=60', 'y(0)=5')
ans =
15 - 10/exp(4*t)
```

Ответ $y = 15 - (10/e^{4t})$ выводится на экран.

ОДУ второго порядка $\frac{d^2y}{dt^2} - 2\frac{dy}{dt} + 2y = 0$, с условиями $y(0) = 5$, $\left.\frac{dy}{dt}\right|_{t=0} = 0$ может быть решено с MATLAB таким образом:

```
>> dsolve('D2y-2*Dy+2*y=0','y(0)=1','Dy(0)=0')
ans =
    Ответ  $y = e^t \cos(t) - e^t \sin(t)$  выводится на экран.
exp(t)*cos(t)-exp(t)*sin(t)
>> factor(ans)
    Ответ может быть упрощен командой factor.
ans =
    Упрощенный ответ  $y = e^t(\cos(t) - \sin(t))$  выводится на экран.
exp(t)*(cos(t)-sin(t))
```

Дополнительные примеры решения дифференциальных уравнений показаны в типовой задаче 11.5.

Если MATLAB не может найти решение, он возвращает пустой символьный объект и сообщение `Warning: explicit solution could not be found.`

11.7. Графическое изображение символьных выражений

Во многих случаях, есть необходимость графически изобразить символьное выражение. Это может быть легко сделано командой `ezplot`. Для символьного выражения S , которое содержит одну переменную var , MATLAB считает, что выражение есть функция $S(var)$, и команда `ezplot` создает график $S(var)$. Для символьного выражения, которое содержит две символьных переменные $var1$ и $var2$, MATLAB полагает, что это выражение есть неявная функция в виде $S(var1, var2) = 0$, и команда создает график одной переменной, как функции другой.

Для графического изображения символьного выражения S , которое содержит одну или две переменные, команда `ezplot` имеет формат :

$\text{ezplot}(S)$ или $\text{ezplot}(S, [\text{min}, \text{max}])$ или $\text{ezplot}(S, [\text{xmin}, \text{xmax}, \text{ ymin}, \text{ ymax}])$	<p>Область изменения независимой переменной</p> <p>Область значений</p>
--	---

- S – это символьное выражение, график которого будет изображен. Это может быть имя ранее созданного символьного выражения, или это выражение может быть введено в команду.
- Можно также ввести выражение в виде строки, в котором переменные не определены заранее как символьные объекты.
- Если S имеет одну символьную переменную, то команда `ezplot` создает график $S(var)$ как функции от var , со значениями var (независимой переменной) на оси абсцисс (горизонтальная ось), и со значениями $S(var)$ на оси ординат (вертикальная ось).
- Если символьное выражение S имеет две символьных переменные, $var1$ и $var2$, то предполагается, что это выражение является функцией заданной

в виде $S(var1,var2) = 0$. MATLAB создает график одной переменной как функции другой. В качестве независимой берется та переменная, которая является первой в алфавитном порядке. Например, если в выражении S переменные – x и y , то x – независимая переменная со значениями на абсциссе, а y – зависимая переменная со значениями на ординате. Если в выражении S переменные – u и v , то u – независимая переменная, а v – зависимая переменная.

- В команде `ezplot(S)`, если S имеет одну переменную ($S(var)$), строится график в области $-2\pi < var < 2\pi$ (область определения по умолчанию), а область значений выбирает сам MATLAB. Если S имеет две переменные ($S(var1,var2)$), то $-2\pi < var1 < 2\pi$ и $-2\pi < var2 < 2\pi$.
- В команде `ezplot(S, [min, max])` область изменения независимой переменной определена значениями \min и \max : $\min < var < \max$, а диапазон значений выбирает MATLAB.
- В команде `ezplot(S, [xmin, xmax, ymin, ymax])` область значений независимой переменной была определена числами xmin и xmax , а область значений зависимой переменной определена через ymin и ymax .

Команда `ezplot` может также использоваться для графического изображения функции, которая задана в параметрическом виде. В этом случае в команду вводятся два символьных выражения S_1 и S_2 , где каждое выражение записано в терминах одной и той же символьной переменной (независимый параметр). Например, для графика y от x , где $x = x(t)$ и $y = y(t)$, форма команды `ezplot` будет такая:

$\text{ezplot}(S_1, S_2)$ **Область изменения независимого параметра**
 или $\text{ezplot}(S_1, S_2, [\min, \max])$

- S_1 и S_2 являются символьными выражениями, содержащими одну и ту же символьную переменную, которая является независимым параметром. S_1 и S_2 могут быть именами ранее созданных символьных выражений, или эти выражения могут быть введены в команду.
- Команда создает график $S_1(var)$ как функции от $S_2(var)$. Символьное выражение, которое введено в команде первым (S_1 в определении выше) используется для горизонтальной оси, и выражение, которое введено вторым (S_2 в определение выше) используется для вертикальной оси.
- В команде `ezplot(S1, S2)` область изменения независимой переменной есть $-0 < var < 2\pi$ (по умолчанию).
- В команде `ezplot(S1, S2, [min, max])` область изменения независимой переменной определена значениями \min и \max : $\min < var < \max$.

Дополнительные комментарии

После создания графика он может быть отформатирован таким же образом как и графики, создаваемые командами `plot` или `fplot`. Это может быть сделано дву-

мя способами: командами форматирования или с использованием Редактора графиков (см. раздел 5.4). При создании графика, выражение, которое графически изображается, автоматически выводится на экран сверху графика. MATLAB имеет дополнительные графические функции для изображения двумерных графиков в полярных координатах и для графического изображения трехмерных графиков. Для получения дополнительной информации мы рекомендуем читателю обратиться к меню **Help** данного пакета Symbolic Math Toolbox.

Несколько примеров использования команды `ezplot` показаны в табл. 11.1.

Таблица 11.1. Графики построенные командой `ezplot`

Программа	Пример графика
<pre>>> syms x >> S=(3*x+2) / (4*x-1) S = (3*x+2) / (4*x-1) >> ezplot(S)</pre>	
<pre>>> syms x y >> S=4*x^2-18*x+4*y^2+12*y-11 S = 4*x^2-18*x+4*y^2+12*y-11 >> ezplot(S)</pre>	
<pre>>> syms t >> x=cos(2*t) x = cos(2*t) >> y=sin(4*t) y = sin(4*t) >> ezplot(x,y)</pre>	

11.8. Численные расчеты с символьными выражениями

После создается символьного выражения или пользователем, или в результате символьных операций MATLAB, может потребоваться заменить числами некоторые символьные переменные и вычислить численное значение выражения. Это может быть сделано с использованием команды `subs`. Эта команда `subs` имеет несколько форм и может использоваться по-разному. Далее будут описаны несколько форм, которые являются удобными и подходят для большинства применений. В первой форме переменная (или переменные), вместо которой подставляется численное значение и само численное значение вводится в команду `subs`. В другой форме каждой переменной присваивается численное значение в отдельной команде, и затем переменная подставляется в выражение.

Сначала рассмотрим команду `subs`, в которой переменная и ее значение вводятся в эту команду. Представлены два случая – один для подстановки числового значения (или значений) одной символьной переменной, а другой – для замены числовыми значениями двух или больше символьных переменных.

Подстановка числового значения вместо одной символьной переменной

Числовое значением (или значения) можно подставить вместо одной символьной переменной, если, конечно, символьное выражение имеет одну или более символьных переменных. В этом случае команда `subs` имеет вид:

`R = subs(S,var,number)`



Имя символьного выражения. Переменная вместо которой подставляется числовое значение. Числовое значение (или значения), присваиваемое `var`.

- `number` может быть одним числом (скаляр), или массивом со многими элементами (вектор или матрица).
- Значение `S` вычисляется для каждого значения `number`, и результат присваивается `R`, который будет иметь тот же самый размер, что и `number` (скаляр, вектор или матрица).
- Если `S` имеет одну переменную, результат `R` является числовым. Если `S` имеет несколько переменных, а числовое значение подставляется только вместо одной из них, тогда результат `R` является символьным выражением.

Пример с выражением, которое включает одну символьную переменную:

```

>> syms x
>> S=0.8*x^3+4*exp(0.5*x)
S =
4*exp(x/2) + (4*x^3)/5
>> SD=diff(S)
SD =
2*exp(x/2)+(12*x^2)/5
>> subs(SD, x, 2)           Использование команды subs для подстановки x = 2 в SD.
ans =                               На экран выводится значение SD.
15.0366                           Использование команды subs для
>> SDU=subs(SD, x, [2:0.5:4])   подстановки вектора x = [2, 2.5, 3, 3.5, 4] в SD.
SDU =
15.0366      21.9807      30.5634      40.9092      53.1781
Значения SD (присвоенные SDU) для каждого значения x выводятся на экран как вектор.

```

Заметим, что в последнем примере, когда вычисляется числовое значение символьного выражения, ответ является числом (на экране расположен с отступом).

Пример замены числовыми значениями одной символьной переменной в выражении, у которого есть несколько символьных переменных:

```

>> syms a g t v      Определение a, g, t и v как символьных переменных.
>> Y=v^2*exp(a*t)/g    Создание символьного выражения  $v^2 e^{at}/g$ 
Y =                      и присвоение его переменной Y.
v^2*exp(a*t)/g
>> subs(Y,t,2)        Использование команды subs для подстановки t = 2 в Y.
ans =                   На экран выводится ответ  $v^2 e^{(2a)}/g$ .
v^2*exp(2*a)/g
>> Yt=subs(Y,t,[2:4])   Использование команды subs для подстановки
Yt =                     вектора t = [2, 3, 4] в Y.
[ v^2*exp(2*a)/g, v^2*exp(3*a)/g, v^2*exp(4*a)/g ]
Ответ есть вектор с элементами из символьных выражений для каждого значения t.

```

Замена числовыми значениями двух или более символьных переменных

Числовым значением (или значениями) можно заменить две или более символьных переменных, если у символьного выражения есть несколько символьных переменных. В этом случае команда `subs` имеет следующий вид (показан для двух переменных, но в той же самой форме может использоваться для большего числа переменных):

$$R = \text{subs}(S, \{var1, var2\}, \{number1, number2\})$$

Имя символьного выражения.

Переменные вместо которых подставляются числовые значения.

Числовые значения (или массивы), присваиваемые `var1` и `var2`.

- Переменные `var1` и `var2` являются переменными в выражении `S`, вместо которых подставляются числовые. Эти переменные вводятся как массив ячеек (в фигурных скобках `{ }`). Массив ячеек – массив из ячеек, где каждая ячейка может быть массивом чисел или текстом.
- Числа `number1`, `number2` подставляемые вместо переменных также вводятся как массив ячеек (в фигурных скобках `{ }`). Эти числа могут быть скалярами, векторами, или матрицами. Первой ячейкой в массиве ячеек чисел (`number1`) заменяют переменную, которая находится в первой ячейке массива переменных (`var1`), и так далее.
- Если все числа, которыми заменяют переменные, будут скалярами, то результатом будет одно число или одно выражение (если некоторые из переменных все еще будут символьными).
- Если по крайней мере для одной переменной подставляемые числа представляют собой массив, то математические операции выполняются поэлементно, и результат есть массив чисел или выражений. Нужно подчеркнуть, что вычисления выполняются поэлементно даже тогда, когда при введении выражения `S` не используются обозначения поэлементных операций. Это также означает, что все массивы, которыми заменяют различные переменные, должны иметь один и тот же размер.
- Можно одновременно подставить массивы (одного и того же размера) вместо некоторых из переменных и скаляры – вместо других переменных. В этом случае для выполнения поэлементных операций MATLAB расширяет скаляры до массивов (создает массив из одинаковых скаляров), чтобы получить в результате массив.

Подстановка числовых значений для двух или больше переменных демонстрируется на следующих примерах.

```
>> syms a b c e x      Определение a, b, c, e и x как символьных переменных.
>> S=a*x^e+b*x+c      Создайте символьного выражения  $ax^e + bx + c$ 
S =                      и присвоение его переменной S.
a*x^e+b*x+c
>> subs(S,{a,b,c,e,x},{5,4,-20,2,3})   Замена в S скаляров для всех
                                              символьных переменных.
                                              Массив ячеек      Массив ячеек
ans =
37                         Значение S выведено на экран.
>> T=subs(S,{a,b,c},{6,5,7})      Замена в S скалярами символьных
T =                           переменных a, b, c.
5*x+ 6*x^e+7                Результат есть выражение с переменными x и e.
>> R=subs(S,{b,c,e}, {[2 4 6],9,[1 3 5]})   Замена в S скаляром
                                              переменной c и векторами переменных b и e.
R =
Результат есть вектор символьных выражений.
[ 2*x+a*x+9, a*x^3+4*x+9, a*x^5+6*x+9]
>> W = subs(S,{a,b,c,e,x}, {[4 2 0],[2 4 6],[2 2 2],[1 3 5],[3 2 1] })
W =
Замена в S векторами всех переменных.
20      26      8      Результат есть вектор числовых значений.
```

Второй метод для замены символьных переменных числовыми значениями в символьном выражении заключается в том, что сначала переменным присваиваются числовые значения, а затем используется команда `subs`. В этом методе в существующем символьном выражении (в котором намеченные переменные являются символьными) переменным присваиваются числовые значения. Здесь команда `subs` используется в виде:

$$R = \text{subs}(S),$$

где S – имя символьного выражения. Как только символьные переменные определены как числовые переменные, они больше не могут использоваться в качестве символьных. Этот метод демонстрируется в следующих примерах.

```
>> syms A c m x y
>> S=A*cos(m*x)+c*y
S =
c*y+A*cos(m*x)
>> A=10; m=0.5; c=3;
>> subs(S)
ans =
3*y + 10*cos(x/2)
>> x=linspace(0,2*pi,4);
>> T = subs(S)
T =
[ 3*y+10, 3*y+5, 3*y-5, 3*y-10]
```

Определение A , c , m , x и y как символьных переменных.
Создайте символьного выражения $A\cos(mx)+cy$
и присвоение его переменной S .

Присвоение числовых значений для A , m и c .
Использование команды `subs` для выражения S .

Числовые значения переменных A , m и c подставлены в S .
Присвоение числовых значений (вектора) переменной x .

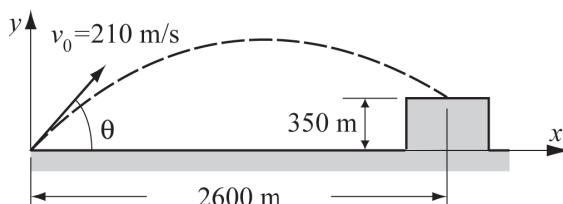
Использование команды `subs` для выражения S .

Подставлены числовые значения переменных A , c , m и x .

11.9. Примеры приложений MATLAB

Пример задачи 11.2. Угол стрельбы снаряда

Снаряд выпущен со скоростью 210 м/с под углом θ . Намеченная цель снаряда находится на расстоянии 2600 м и на 350 м выше точки выстрела.



- (a) Выведите уравнение для определения угла θ такого, что снаряд достигнет намеченной цели.

- (b) Используйте MATLAB для решения уравнения, полученного в пункте (a).
(c) Для угла, найденного в пункте (b), используйте команду `ezplot` для построения графика траектории снаряда.

Решение

- (a) Движение снаряда может быть разложено на горизонтальную и вертикальную составляющие. Начальная скорость имеет следующие горизонтальные и вертикальные составляющие:

$$v_{0x} = v_0 \cos(\theta) \quad \text{и} \quad v_{0y} = v_0 \sin(\theta).$$

В горизонтальном направлении скорость является постоянной, и положение снаряда как функция времени задается формулой:

$$x = v_{0x} t.$$

Подставляя $x = 2600$ м для горизонтального расстояния, которое пролетает снаряд для достижения цели и $210\cos(\theta)$ для v_{0x} , и решая относительно t , получаем:

$$t = \frac{2600}{210 \cos(\theta)}.$$

В вертикальном направлении положение снаряда дается формулой:

$$y = v_{0y} t - \frac{1}{2} g t^2.$$

Подставляя $y = 350$ м в качестве вертикальной координаты цели, $210\sin(\theta)$ вместо v_{0y} , $g = 9.81$ и значение t , получаем:

$$350 = 210\sin(\theta) \frac{2600}{210\cos(\theta)} - \frac{1}{2} 9.81 \left(\frac{2600}{210\cos(\theta)} \right)^2$$

или

$$350 = \frac{2600\sqrt{1-\cos^2(\theta)}}{\cos(\theta)} - \frac{1}{2} 9.81 \left(\frac{2600}{210\cos(\theta)} \right)^2.$$

Решение этого уравнения дает угол θ выстрела снаряда.

(b) Решение уравнения, выведенного в пункте (a), получается с использованием команды `solve` (в командном окне) и представлено следующим листингом:

```
>> syms th
Angle = solve('2600*sqrt(1-cos(th)^2)/cos(th)-0.5*9.81*(2600/
(210*cos(th)))^2 = 350')
Angle =
1.245354497237416168313813580656
0.4592528070320712127786452037279
-0.4592528070320712127786452037279
-1.245354497237416168313813580656
MATLAB выводит на экран четыре решения.
Два положительных имеют
отношение к задаче.

>> Angle1 = Angle(1)*180/pi
Angle1 =
224.16380950273491029648644451808/pi
MATLAB выводит на экран ответ как
символьный объект через π.

>> Angle1=double(Angle1)
Angle1 =
71.3536
Использование команды double для
получения числового значения Angle1.

>> Angle2=Angle(2)*180/pi
Angle2 =
Preобразование второго решения в Angle
из радиан в градусы. →
```

```
82.665505265772818300015613667102/pi
>> Angle2=double(Angle2)
Angle2 =
26.3132
```

MATLAB выводит на экран ответ как символьный объект через π .
Использование команды `double` для получения числового значения `Angle2`.

(c) Решение пункта (b) показывает, что есть два возможных угла и таким образом две траектории. Чтобы сделать график траектории выразим координаты x и y снаряда через t (параметрическая форма):

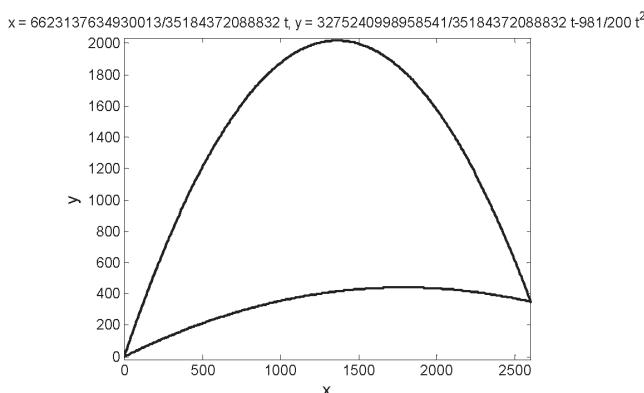
$$x = v_0 \cos(\theta)t \quad \text{и} \quad y = v_0 \sin(\theta)t - gt^2/2.$$

$$\text{Область изменения переменной } t: \text{ от } t = 0 \text{ до } t = \frac{2600}{210 \cos(\theta)}.$$

Данные уравнения могут использоваться в команде `ezplot` для создания графиков. Это показано в следующей программе записанной в скрипте-файле.

```
xmax=2600; v0=210; g=9.81;
theta1=1.24535; theta2=.45925;
t1=xmax/(v0*cos(theta1));
t2=xmax/(v0*cos(theta2));
syms t
X1=v0*cos(theta1)*t;
X2=v0*cos(theta2)*t;
Y1=v0*sin(theta1)*t-0.5*g*t^2;
Y2=v0*sin(theta2)*t-0.5*g*t^2;
ezplot(X1,Y1,[0,t1])           Присвоение двух решений из пункта (b)
hold on                         переменным theta1 и theta2.
ezplot(X2,Y2,[0,t2])           График одной траектории.
hold off                         График второй траектории.
```

В результате выполнения этой программы создается следующий график в окне графиков Figure:



Пример задачи 11.3. Сопротивление изгибу балки

Сопротивление изгибу прямоугольной балки ширины b и высоты h пропорционально моменту инерции балки I , определенного формулой $I = \frac{1}{12}bh^3$. Прямоугольная балка вырезана из цилиндрического бревна радиуса R . Определить b и h (как функции R) такие, чтобы балка имела максимум I .

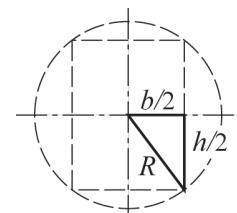
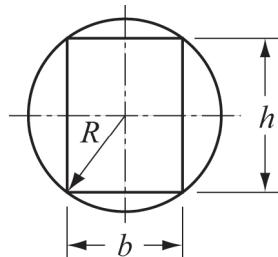
Решение

Задача решается в несколько этапов:

1. Записать уравнение, связывающее R , h и b .
2. Вывести выражение для I через h .
3. Вычислить производную I относительно h .
4. Приравнять производную к нулю и решать относительно h .
5. Определить соответствующее b .

Первый шаг выполняется по рисунку треугольника справа. Соотношение между R , h и b определяется теоремой Пифагора: $(b/2)^2 + (h/2)^2 = R^2$. Решение этого уравнения относительно b : $b = \sqrt{4R^2 - h^2}$.

Остальная часть шагов делается с использованием MATLAB:



```
>> syms b h R
>> b=sqrt(4*R^2-h^2);
>> I=b*h^3/12
I =
(h^3*(4*R^2-h^2)^(1/2))/12
>> ID=diff(I,h)
ID =
(h^2*(4*R^2-h^2)^(1/2))/4-h^4/(12*(4*R^2-h^2)^(1/2))
>> hs=solve(ID,h)
hs =
0
3^(1/2)*R
-3^(1/2)*R
>> bs=subs(b,hs(2))
bs =
(R^2)^(1/2)
```

Этап 1. Создание символьного выражения для b .
 Этап 2. Создание символьного выражения для I .
 MATLAB подставляет b в I .

Этап 3: Использование команды $diff(I,h)$ для дифференцирования I относительно h .
 Производная от I выводится на экран.

Этап 4: Использование команды $solve$ для решения уравнения $ID = 0$ относительно h .
 Присвоение ответа переменной hs .
 MATLAB выводит на экран три решения.
 Положительное ненулевое решение относится к задаче.

Этап 5: Используйте команды $subs$ для определения b подстановкой решения для h в выражении для b .
 Ответ для b выводится на экран. (Ответ есть R , но MATLAB выводит $(R^2)^{1/2}$).

Пример задачи 10.4. Уровень топлива в цистерне

Горизонтальная цилиндрическая цистерна, показанная на рисунке, используется для хранения топлива. Корпус имеет диаметр 6 м, а длина составляет 8 м. Количество топлива в корпусе может быть оценено по уровню топлива, что можно наблюдать через узкую вертикальную стеклянную щель на передней стенке корпуса. Шкала, метки которой находятся рядом со смотровой щелью, показывает уровни топлива, соответствующие 40, 60, 80, 120 и 160 тысяч литров. Определить вертикальное положение (измеренное от нижнего основания) меток шкалы.

Решение

Соотношение между уровнем топлива и его объемом может быть записано в виде определенного интеграла. После выполнения интегрирования получается уравнение для объема через высоту топлива. Тогда высота, соответствующая некоторому объему, может быть определена как решение этого уравнения относительно высоты.

Объем топлива V может быть определен, умножая площадь A сечения топлива (затененная область на рисунке) на длину L корпуса. Площадь поперечного сечения может быть вычислена интегрированием.

$$V = A \cdot L = L \int_0^h w dy.$$

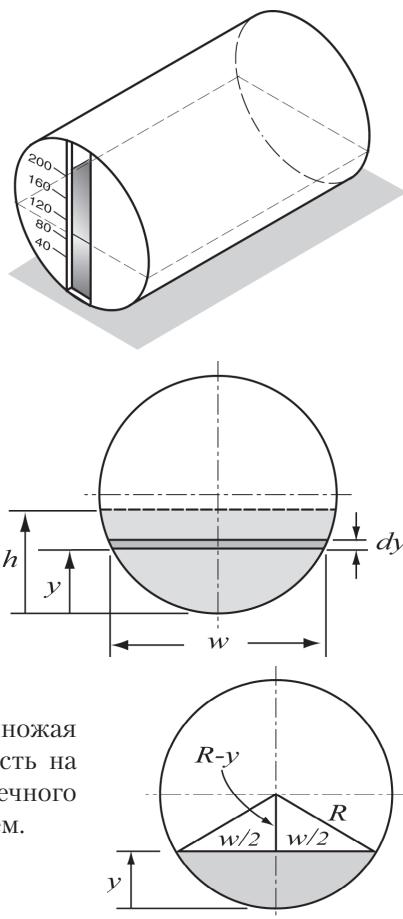
Ширина w верхней поверхности топлива для высоты y может быть записана как функция от y . В прямоугольном треугольнике на рисунке справа переменные y , w и R связаны соотношением:

$$\left(\frac{w}{2}\right)^2 + (R - y)^2 = R^2.$$

Решая это уравнение относительно w , получаем:

$$w = 2\sqrt{R^2 - (R - y)^2}.$$

Объем топлива на высоте h может быть теперь вычислен, если подставить это w в интеграл в уравнении для объема и выполнить интегрирование. В результате получается уравнение, которое дает объем V как функцию h . Значение h для данного V находится решением этого уравнения относительно h . В данной задаче зна-



чения h должны быть определены для объемов: 40, 60, 80, 120 и 160 тысяч литров. Решение дается в следующей программе MATLAB (скрипт-файл):

```
R=3; L=8;
syms w y h
w=2*sqrt(R^2-(R-y)^2)      Создание символьного выражения для w.
S = L*w                     Создание подынтегрального выражения.
V = int(S,y,0,h)            Использование команды int для интегрирования
                            S от 0 до h. Результат V получается как функция h.
Vsclae=[40:40:200]          Создание вектора со значениями V на шкале.
for i=1:5                  Каждый проход цикла находит h для одного значения V.
    Veq=V-Vscale(i);        Создание уравнения ( $V = V_{scale}$ ), которое должно
    h_ans(i)=solve(veq);    быть решено относительно h.
end                         Использование команды solve для решения относительно h.
                            ←
                            h_ans – это вектор (символьный с числами) со значениями h, которые
                            соответствуют значениям V в векторе Vscale.
h_scale=double(h_ans)       Использование команды double для получения
                            числовых значений элементов вектора h_ans.
```

При выполнении этого скрипта-файла на экран выводятся результаты команд, у которых нет точки с запятой в конце. Вид в командном окне:

```
>> w =                      Символьное выражение для w выведено на экран.
2*(9-(y-3)^2)^(1/2)
S =                         S – это выражение, которое будет интегрироваться.
16*(9-(y-3)^2)^(1/2)
V =                          Результат интегрирования; V как функция h.
36*pi+72*asin(h/3-1)+8*(9-(h-3)^2)^(1/2)*(h-3)
Vsclae =                     На экран выведены значения объемов V для шкалы.
40    80    120   160    200
h_scale =                    На экран выведены положения линий меток для шкалы.
1.3972   2.3042   3.1439   3.9957   4.9608
```

Единицы измерения: единица длины при решении – метр, это соответствует м^3 для объема ($1 \text{ м}^3 = 1000 \text{ Л}$).

Пример задачи 10.5. Количество лекарственного препарата в теле

Количество M лекарственного препарата находящегося в теле зависит от интенсивности, с которой лекарственный препарат выводится из тела, и от интенсивности приема препарата, причем скорость поглощения (вывода) лекарственного препарата телом пропорционально количеству препарата в теле. Дифференциальное уравнение для M :

$$\frac{dM}{dt} = -kM + p,$$

где k – коэффициент пропорциональности и p – интенсивность приема препарата.

- (a) Определить k , если период полувыводла препарата составляет 3 часа.
- (b) Пациент поступил в больницу и ему оказывается лечение лекарственным препаратом с интенсивностью 50 мг/ч. (Первоначально в теле пациента не было никакого лекарства.) Вывести выражение для M как функции времени.
- (c) Изобразить график M как функции времени в течение первых 24 часов.

Решение

- (a) Коэффициент пропорциональности может быть определен, если мы рассмотрим случай, когда препарат принят только один раз. В этом случае дифференциальное уравнение принимает вид:

$$\frac{dM}{dt} = -kM.$$

Это уравнение может быть решено с начальным условием $M = M_0$ при $t = 0$:

```
>> syms M M0 k t
>> Mt=dsolve('DM=-k*M', 'M(0)=M0')
Mt =
M0/exp(k*t)
```

Использование команды `dsolve`
для решения $\frac{dM}{dt} = -kM$.

Решение дает M как функцию времени:

$$M(t) = \frac{M_0}{e^{kt}}.$$

Период полураспада препарата 3 часа означает, что $M(t) = M_0/2$ при $t = 3$. Подставляя данную информацию в решение, получаем: $0.5 = \frac{1}{e^{3k}}$. Теперь константа k находится как решения этого последнего уравнения:

```
ks=solve('0.5=1/exp(k*3)')
ks =
.23104906018664843647241070715273
```

Использование команды `solve`
для решения $0.5 = e^{-3k}$.

- (b) Для этой части дифференциальное уравнение для M имеет вид:

$$\frac{dM}{dt} = -kM + p.$$

Постоянная k найдена в пункте (а), а $p = 50$ мг/ч – дано. Начальное условие состоит в том, что в начале в теле пациента нет приепарата, или $M = 0$ при $t = 0$. Решение этого уравнения с MATLAB:

```
>> syms p
>> MtB=dsolve('DM=-k*M+p', 'M(0)=0')
MtB =
(p-p/exp(k*t))/k
```

Использование команды `dsolve`
для решения $\frac{dM}{dt} = -kM + p$.

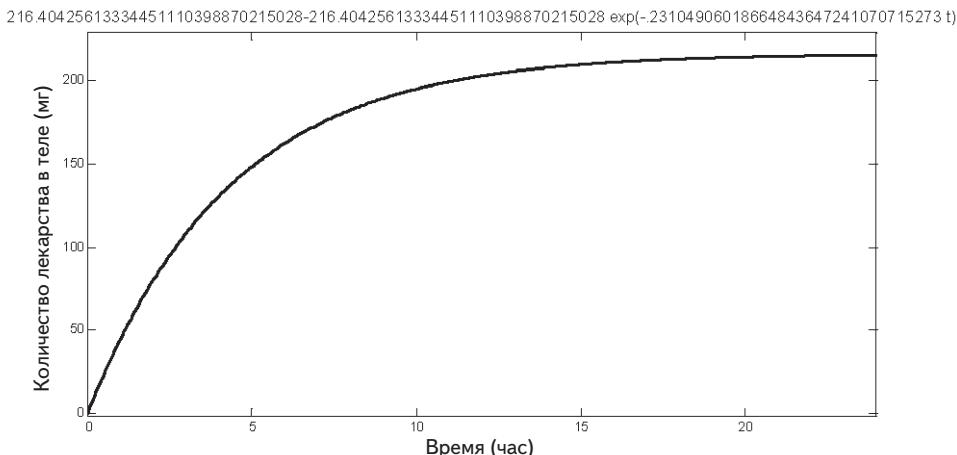
(с) График MtB как функция времени для $0 \leq t \leq 24$ может быть создан при использовании команды `ezplot`:

```
>> pgiven=50;
>> Mtt=subs(MtB, {p, k}, {pgiven, ks})
Mtt =
216.404-216.404/exp(0.231049*t)
>> ezplot(Mtt, [0, 24])
```

Подстановка числовых
значений вместо p и k .

Фактически при выводе на экран последнего выражения, которое было сгенерировано MATLAB ($Mtt = \dots$), числа имеют многое больше десятичных знаков, чем показано выше. Эти числа были укорочены с тем, чтобы они помещались на страницу.

Созданный график:



11.10. Задачи

1. Определите x как символьную переменную и создайте два символьных выражения

$$S_1 = x^2(x - 6) + 4(3x - 2) \quad \text{и} \quad S_2 = (x + 2)^2 - 8x.$$

Используйте символьные операции для нахождения самого простого вида каждого из следующих выражений:

$$(a) S_1 \cdot S_2; \quad (b) \frac{S_1}{S_2}; \quad (c) S_1 + S_2;$$

(d) Используйте команду `subs` для оценки численного значения результата пункта (c) при $x = 5$.

2. Определите x как символьную переменную и создайте два символьных выражения

$$S_1 = x(x^2 + 6x + 12) + 8 \quad \text{и} \quad S_2 = (x - 3)^2 + 10x - 5.$$

Используйте символьные операции для нахождения самого простого вида каждого из следующих выражений:

$$(a) S_1 \cdot S_2; \quad (b) \frac{S_1}{S_2}; \quad (c) S_1 + S_2.$$

(d) Используйте команду `subs` для оценки численного значения результата пункта (c) при $x = 3$.

3. Определите x и y как символьные переменные и создайте два символьных выражения

$$S = x + \sqrt{xy^2 + y^4} \quad \text{и} \quad T = \sqrt{x} - y^2.$$

Используйте символьные операции для нахождения самого простого вида $S \cdot T$.

Используйте команду `subs` для оценки численного значения этого результата при $y = 2$.

4. Определите x как символьную переменную.

(a) Выведите уравнение многочлена, который имеет корни $x = -2$, $x = -0.5$, $x = 2$ и $x = 4.5$.

(b) Найдите корни из многочлена

$$f(x) = x^6 - 6.5x^5 - 58x^4 + 167.5x^3 + 728x^2 - 890x - 1400$$

с использованием команды `factor`.

5. Используя команды из раздела 11.2 показать, что:

$$(a) \sin(4x) = 4 \sin(x) \cos(x) - 8 \sin^3(x) \cos(x),$$

$$(b) \cos(x) \cos(y) = [\cos(x - y) + \cos(x + y)]/2.$$

6. Используя команды из раздела 11.2 показать, что:

$$(a) \tan(3x) = \frac{3\tan x - \tan^3 x}{1 - 3\tan^2 x},$$

$$(b) \sin(x+y+z) = \sin(x)\cos(y)\cos(z) + \cos(x)\sin(y)\cos(z) + \\ + \cos(x)\cos(y)\sin(z) - \sin(x)\sin(y)\sin(z).$$

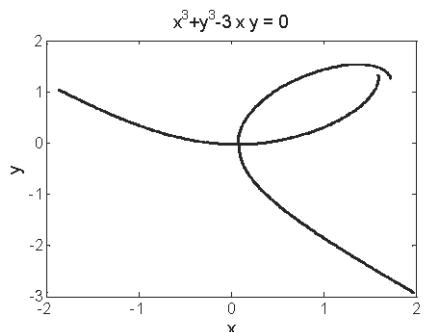
7. Декартов лист – это линия, показанная на рисунке. В параметрической форме его уравнения имеют вид:

$$x = \frac{3t}{1+t^3} \quad \text{и} \quad y = \frac{3t^2}{1+t^3} \quad \text{при } t \neq -1.$$

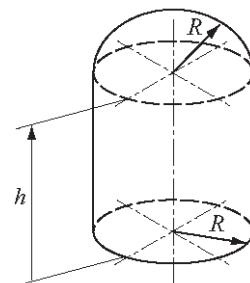
(a) Используя MATLAB покажите, что уравнение декартового листа может также быть записано так:

$$x^3 + y^3 = 3xy.$$

(b) Создайте график этого листа в области показанной на рисунке с использованием команды `ezplot`.



8. Водонапорная башня имеет геометрию показанную на рисунке (нижняя часть – цилиндр радиуса R и высоты h , а верхняя часть – полусфера радиуса R). Определите радиус R , если $h = 10$ м, а объем равен $1,050 \text{ м}^3$. Запишите уравнение для объема через радиус и высоту. Решите уравнение относительно радиуса, и используйте команду `double`, чтобы получить численное значение.



9. Соотношение между напряжением T и постоянной скоростью сокращения v в мышце задается уравнением Хилла:

$$(T+a)(v+b) = (T_0+a)/b,$$

где a и b – положительные константы, и T_0 – изометрическая сила, то есть, сила в мышце, когда $v = 0$. Максимальная скорость сокращения происходит когда $T = 0$.

(a) Используя символьные операции, создайте уравнение Хилла как символьное выражение. Затем используйте `subs` для подстановки $T = 0$ и затем решите относительно v , чтобы показать что $v_{\max} = (bT_0)/a$.

(b) Используйте результаты пункта (a) для исключения постоянной b из уравнения Хилла, и покажите, что $v = \frac{a(T_0 - T)}{T_0(T + a)} v_{\max}$.

10. Рассмотрим два эллипса в плоскости xOy , заданные уравнениями:

$$\frac{(x-1)^2}{6^2} + \frac{y^2}{3^2} = 1 \quad \text{и} \quad \frac{(x+1)^2}{2^2} + \frac{(y-5)^2}{4^2} = 1.$$

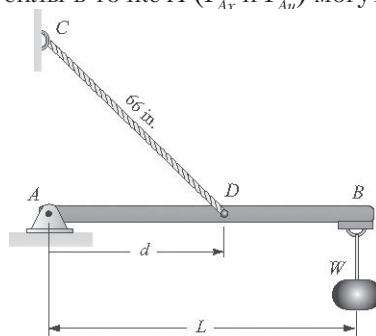
- (a) Используйте команду `ezplot`, чтобы графически изобразить эти два эллипса на одном рисунке `Figure`.
 (b) Определите координаты точек пересечения эллипсов.

11. Балка AB длиной 120 дюймов прикреплена к стене шарниров в точке A и кабелем CD длины 66 дюймов. Нагрузка 200 фунтов приложена к балке в точке B . Сила натяжения кабеля T и x и y компоненты силы в точке A (F_{Ax} и F_{Ay}) могут быть найдены из уравнений:

$$F_{Ax} - T \frac{d}{L_c} = 0,$$

$$F_{Ay} + T \frac{\sqrt{L_c^2 - d^2}}{L_c} - W = 0,$$

$$T \frac{\sqrt{L_c^2 - d^2}}{L_c} d - WL = 0,$$



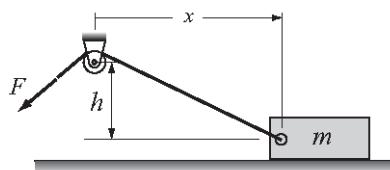
где L и L_c являются длинами балки и кабеля, соответственно, а d – это расстояние от точки A до точки D , где закреплен кабель.

- (a) Используйте MATLAB решите эти уравнения относительно сил T , F_{Ax} и F_{Ay} через величины d , L , L_c и W . Определите F_A по формуле $F_A = \sqrt{F_{Ax}^2 + F_{Ay}^2}$.
 (b) Используйте команду `subs` для подстановки $W = 200$ фунтов, $L = 120$ дюймов и $L_c = 66$ дюймов в выражения, выведенные в части (a). Это даст силу как функцию расстояния d .
 (c) Используйте команду `ezplot` для графического изображения сил T и F_A (оба графика в одном окне `figure` как функции d , когда d меняется от 20 и до 70 дюймов.)
 (d) Определите расстояние d , когда сила натяжения кабеля является наименьшей. Определите значение этой силы.

12. Ящик массы m тянет веревка как показано на рисунке. Сила F , приложенная к веревке как функция x может быть вычислена от уравнений:

$$-F \frac{x}{\sqrt{x^2 + h^2}} + \mu N = 0,$$

$$-mg + N + F \frac{h}{\sqrt{x^2 + h^2}} = 0,$$



где N и μ – нормальная сила и коэффициентом трения между ящиком и поверхностью, соответственно. Рассмотрите случай, когда $m = 18$ кг, $h = 10$ м, $\mu = 0.55$ и $g = 9.81$ м/с².

- (a) Используйте MATLAB для вывода выражения для F через x , h , m , g и μ .
- (b) Используйте команду `subs` для подстановки $m = 18$ кг, $h = 10$ м, $\mu = 0.55$ и $g = 9.81$ м/сек² в выражения, которые были выведены в пункте (a). Это даст силу как функцию расстояния x .
- (c) Используйте команду `ezplot` для графического изображения силы F как функции x для x , меняющегося от 5 и до 30 м.
- (d) Определите расстояние x , когда сила требуемая для вытягивания ящика, является наименьшей и определите величину этой силы.

13. Механическая выходная мощность P в сокращающейся мышце определяется по формуле:

$$P = T v = \frac{kvT_0 \left(1 - \frac{v}{v_{\max}}\right)}{k + \frac{v}{v_{\max}}},$$

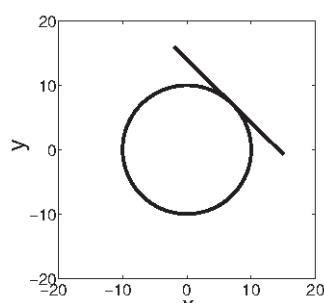
где T – сила мышцы, v – скорость сокращения (v_{\max} – максимальная), T_0 – изометрическая сила (то есть, сила в мышце, когда $v = 0$) и k является безразмерной константой, которая принимает значения между 0.15 и 0.25 для большинства мышц. Это уравнение может быть записано в безразмерной форме:

$$p = \frac{ku(1-u)}{k+u},$$

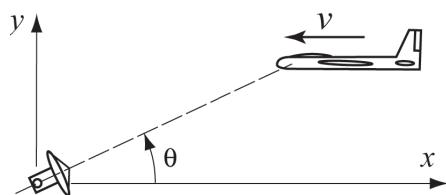
где $p = (Tv)/(T_0 v_{\max})$ и $u = v/v_{\max}$. Рассмотрите случай $k = 0.25$.

- (a) Постройте график p как функции u для $0 \leq u \leq 1$.
- (b) Используйте дифференцирование для нахождения значения u , где p максимально.
- (c) Найдите максимальное значение p .

14. Уравнение окружности есть $x^2 + y^2 = R^2$, где R – радиус окружности. Запишите программу в скрипте-файле, которая сначала выводит уравнение (символьно) касательной линии к окружности в точке (x_0, y_0) на верхней части окружности (то есть, для $-R < x_0 < R$ и $0 < y_0$). Затем для определенных значений R , x_0 и y_0 программа создает график окружности и касательной линии, как показано на рисунке справа. Выполните эту программу с $R = 10$ и $x_0 = 7$.



- 15.** Антенна радиолокационной станции сопровождения наведена на самолет, летящий на постоянной высоте 5 км и с постоянной скоростью 540 км/ч. Самолет движется вдоль пути, который проходит точно над радиолокационной станцией. Радар запускает слежение, когда самолет находится на расстоянии в 100 км.



- (a) Выведите выражение для угла антенны радара как функции времени.
 (b) Выведите выражение для угловой скорости антенны $d\theta/dt$, как функции времени.
 (c) Создайте два графика на одной странице, один график как функции времени, а другой $-d\theta/dt$ как функции времени, где угол в градусах, а время t – в минутах для $0 \leq t \leq 20$ мин.

- 16.** Вычислить следующие неопределенные интегралы:

$$(a) I = \int \frac{x^3}{\sqrt{1-x^2}} dx; \quad (b) I = \int x^2 \cos x dx.$$

- 17.** Определите x как символьную переменную и создайте символьное выражение

$$S = \frac{\cos^2 x}{1 + \sin^2 x}.$$

Постройте график S в области $0 \leq x \leq \pi$ и вычисляя интеграл $I = \int_0^\pi \frac{\cos^2 x}{1 + \sin^2 x} dx$.

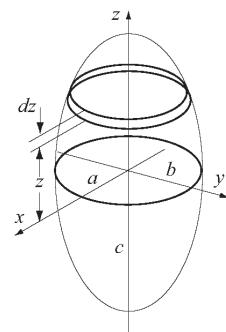
- 18.** Параметрические уравнения эллипсоида:

$x = a \cos(u) \sin(v)$, $y = b \sin(u) \sin(v)$, $z = c \cos(v)$,
где $0 \leq u \leq 2\pi$ и $-\pi \leq v \leq 0$.

Покажите, что дифференциальный элемент объема указанного эллипсоида задается формулой:

$$dV = -\pi abc \sin^3(v) dv.$$

Используйте MATLAB для вычисления интеграла dV от $-\pi$ до 0 символьно и покажите, что объем эллипсоида равен $V = 4\pi abc/3$.



- 19.** Одномерное уравнение диффузии имеет вид:

$$\frac{\partial u}{\partial t} = m \frac{\partial^2 u}{\partial x^2}.$$

Покажите, что следующие выражения являются решениями уравнения диффузии.

(a) $u = A \frac{1}{\sqrt{t}} \exp\left(\frac{-x^2}{4mt}\right) + B$, где A и B – константы.

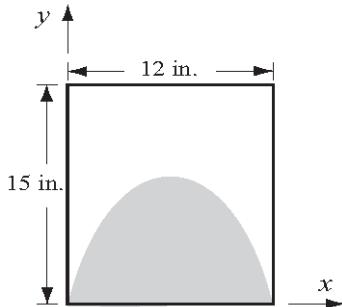
(b) $u = A \exp(-\alpha x) \cos(\alpha x - 2m\alpha^2 t + B) + C$,

где A, B, C и α – константы.

- 20.** Керамическая плитка имеет вид, показанный на рисунке. Затененная область имеет красный цвет, а остальная часть плитки является белой. Граница между красной и белой областями определяется уравнением

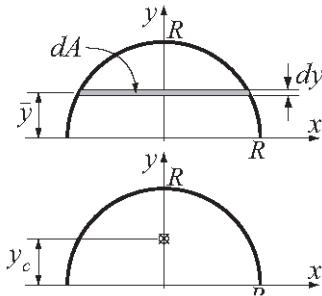
$$y = -kx^2 + 12kx.$$

Найдите k такое, что площадь белой и красной областей будут одинаковыми.



- 21.** Покажите, что расположение центроида (центра масс) y_c показанной справа области полукруга определяется формулой $y_c = \frac{4R}{3\pi}$. Координата y_c может быть вычислена по формуле:

$$y_c = \frac{\int_A \bar{y} dA}{\int_A dA}.$$



- 22.** Для области полукруга из предыдущей задачи покажите, что момент инерции I_x относительно оси x определяется формулой $I_x = \frac{1}{8}\pi R^3$. Момент инерции I_x может быть вычислен по формуле:

$$I_x = \int_A y^2 dA.$$

- 23.** Среднеквадратичное значение напряжения переменного тока определяется формулой:

$$v_{rms} = \sqrt{\frac{1}{T} \int_0^T v^2(t') dt'},$$

где T – период колебания.

- (a) Напряжение задано формулой $v(t) = V \cos(\omega t)$. Покажите, что $v_{rms} = \frac{V}{\sqrt{2}}$ и не зависит от ω . (Соотношение между периодом T и радианной частотой ω есть $T = 2\pi/\omega$.)

- (b) Напряжение задано формулой $v(t) = 2.5 \cos(350t) + 3$. Определите v_{rms} .

24. Распространение инфекции от одного индивидуума к совокупности N незараженных людей может быть описано уравнением

$$\frac{dx}{dt} = -Rx(N+1-x) \text{ с начальным условием } x(0) = N,$$

где x – число незараженных индивидуумов и R – положительная константа скорости процесса. Решите это дифференциальное уравнение символьно относительно $x(t)$. Кроме того, определите символьно время t , когда зараженность dx/dt максимальна.

25. Функция плотности вероятности Максвелла-Больцмана определяется формулой

$$f(v) = \sqrt{\frac{2}{\pi}} \left(\frac{m}{kT}\right)^{\frac{3}{2}} v^2 \exp\left(-\frac{mv^2}{2kT}\right),$$

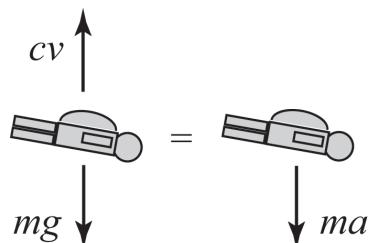
где m – масса каждой молекулы (кг), v – скорость (м/сек), T – температура (К) и $k = 1.38 \times 10^{-23}$ Дж/К – постоянная Больцмана. Наиболее вероятная скорость v_p соответствует максимальному значению $f(v)$ и может быть определена из уравнения $\frac{df(v)}{dv} = 0$. Создайте символьное выражение для $f(v)$, продифференцируйте его относительно v и покажите что $v_p = \sqrt{\frac{2kT}{m}}$. Вычислите v_p для молекул кислорода ($m = 5.3 \times 10^{-26}$ кг) при $T = 300$ К ($k = 1.38 \times 10^{-23}$ Дж/К). Создайте график $f(v)$ как функции от v для $0 \leq v \leq 2500$ м/с для молекул кислорода.

26. Скорость парашютиста, парашют которого еще не открыт, может быть смоделирована в предположении, что сопротивление воздуха пропорционально скорости. Из второго закона движения Ньютона соотношение между массой m парашютиста и его скоростью v имеет вид (положительное направление – вниз),

$$mg - cv = m \frac{dv}{dt},$$

где c – коэффициент сопротивления и g – гравитационная постоянная ($g = 9.81$ м/с²).

- (a) Решите это уравнение относительно v через m, g, c и t , предполагая, что начальная скорость парашютиста – нуль.
 (b) Замечено, что спустя 4 сек после прыжка из самолета 90-килограммового парашютиста, его скорость составляет 28 м/с. Определите постоянную c .



- (c) Создайте график скорости парашютиста как функция времени для $0 \leq t \leq 30$ сек.

- 27.** Сопротивление R ($R = 0.4$ Ом) и катушка индуктивности L ($L = 0.08$ Гн) соединены как показано на рисунке. Первоначально, переключатель находится в положении A и в цепи тока нет. При $t = 0$ переключатель находится в положении B так, что сопротивление и индуктор подключены к источнику тока v_s ($v_s = 6$ В) и в цепи начинает течь ток. Переключатель остается в положении B до тех пор, пока напряжение на сопротивлении не достигает 5 В. В этот момент времени (t_{BA}) переключатель возвращается в положение A .

Ток i в цепи может быть найден как решение дифференциальных уравнений:

$$iR + L \frac{di}{dt} = v_s \quad \text{— в течение времени от } t = 0 \text{ и до времени, когда переключатель возвращается в положение } A.$$

$$iR + L \frac{di}{dt} = 0 \quad \text{— от времени, когда переключатель вернулся в положение } A.$$

Напряжение через сопротивление v_R в любой момент времени задается формулой $v_R = iR$.

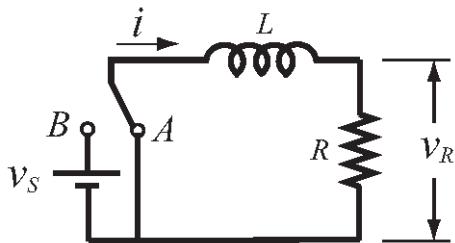
- (a) Найдите выражение для тока i через R , L , v_s и t для $0 \leq t \leq t_{BA}$, решая дифференциальное уравнение первого порядка.
 (b) Подставьте значения R , L и v_s в решение для i и определяю время t_{BA} , когда напряжение на сопротивлении достигает 5 В.
 (c) Выведите выражение для тока i через R , L и t для $t_{BA} \leq t$, решая второе дифференциальное уравнение.
 (d) Создайте два графика v_R как функции от t (на одной странице), один — для $0 \leq t \leq t_{BA}$, и другой — для $t_{BA} \leq t \leq 2t_{BA}$.

- 28.** Определите общее решение дифференциального уравнения

$$\frac{dy}{dx} = \frac{x^4 - 2y}{2x}.$$

Покажите, что это решение корректно. (Найдите первую производную решения, а затем подставьте в уравнение.)

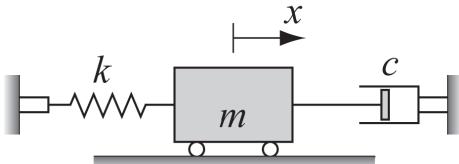
- 29.** Найдите решение следующего дифференциального уравнения, которое удовлетворяет заданным начальным условиям. Графически изобразите решение для $0 \leq t \leq 7$.



$$\frac{d^2y}{dt^2} - 0.08 \frac{dy}{dt} + 0.6t = 0, \quad y(0) = 2, \quad \left. \frac{dy}{dt} \right|_{t=0} = 3.$$

- 30.** Ток i в последовательной цепи RLC , когда переключатель замкнут в момент $t = 0$, может быть определен из решения ОДУ 2-го порядка

$$L \frac{d^2i}{dt^2} + R \frac{di}{dt} + \frac{1}{C} i = 0,$$

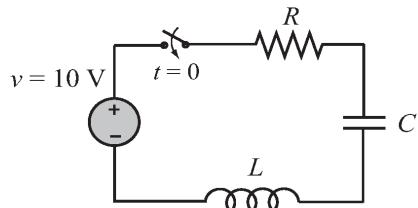


где R , L и C – сопротивление резистора, индуктивность катушки индуктивности и емкость конденсатора, соответственно.

- (a) Решите это уравнение относительно i через R , L , C и t , предполагая, что $i = 0$ и $di/dt = 8$ при $t = 0$.
- (b) Используйте команду `subs` для подстановки $L = 3$ Гн, $R = 10$ Ом и $C = 80$ °F в выражение, которое было получено в пункте (a). Создайте график i как функции t для $0 \leq t \leq 1$ сек. (Слабо демпфированная реакция.)
- (c) Используйте команду `subs` для подстановки $L = 3$ Гн, $R = 200$ Ом и $C = 1200$ °F в выражение, которое было получено в пункте (a). Создайте график i как функции t для $0 \leq t \leq 2$ сек. (Сильно демпфированная реакция.)
- (d) Используйте команду `subs` для подстановки $L = 3$ Гн, $R = 201$ Ом и $C = 300$ °F в выражение, которое было получено в пункте (a). Создайте график i как функции t для $0 \leq t \leq 2$ сек. (Критически демпфированная реакция.)

- 31.** Затухающие свободные колебания могут быть смоделированы блоком из массы m , которая присоединена к пружине и амортизатору как показано на рисунке. Из второго закона движения Ньютона смещение x массы, как функция времени может быть определено из дифференциального уравнения

$$m \frac{d^2x}{dt^2} + c \frac{dx}{dt} + kx = 0,$$



где k – константа пружины и c – коэффициент демпфирования амортизатора. Если масса будет сдвинута из ее положения равновесия и затем отпущена, то она начинает колебаться назад и вперед. Природа колебаний зависит от величины массы и от значений k и c .

Для системы, показанной на рисунке, $m = 10$ кг и $k = 28$ Н/м. В момент $t = 0$ масса сдвинута в положение $x = 0.18$ м и затем отпущена. Выведите выражения для смещения x и скорости v массы, как функций времени. Рассмотрите следующие два случая:

- (a) $c = 3 \text{ (Н·сек)/м.}$
- (b) $c = 50 \text{ (Н·сек)/м.}$

Для каждого случая постройте графики положения x и скорости v как функций времени (два графика на одной странице). Для случая (a) возьмите $0 \leq t \leq 20$ сек, и для случая (b) возьмите $0 \leq t \leq 10$ сек.



ПРИЛОЖЕНИЕ.

Сводка символов, команд и функций

Следующие таблицы представляют символы MATLAB, команды и функции, которые описаны в книге. Списки сгруппированы по темам.

Символы и арифметические операторы

Символ	Описание
+	Сложение.
-	Вычитание.
*	Умножение скаляров и массивов.
.*	Поэлементное умножение массивов.
/	Правое деление.
\	Левое деление.
. /	Поэлементное правое деление.
. \	Поэлементное левое деление.
^	Возведение в степень.
. ^	Поэлементное возведение в степень.
:	Двоеточие; создает векторы с равномерно распределенными элементами, представляет диапазон элементов в массивах.
=	Оператор присвоения.
()	Круглые скобки; установка приоритета, включают входные аргументы в функциях и нижние индексы массивов.
[]	Прямые скобки; образование массивов. Включают выходные аргументы в функциях.

Символ	Описание
,	Запятая; разделяет нижние индексы массива и аргументы функции, разделяет команды на одной строке.
;	Точка с запятой; подавляет вывод результатов на экран, конец строки в массиве.
'	Одинарная кавычка; транспонирование матрицы, создание строки (string).
...	Многоточие. Замещающий знак; продолжение строки.
%	Процент; обозначает комментарий, определяет формат вывода результата.

Операторы сравнения и логические операторы

Символ	Описание
<	Меньше чем.
>	Больше чем.
<=	Меньше или равно.
>=	Больше или равно.
==	Равно.
~=	Не равно.
&	Логическое И (AND).
	Логическое ИЛИ (OR).
~	Логическое НЕТ (NOT).

Команды управления

Команда	Описание
cd	Изменение текущего каталога.
clc	Очищает командное окно.
clear	Удаляет все переменные из памяти.
clear x y z	Удаляет переменные x, y и z из памяти.
close	Закрывает активное окно графиков Figure.
fclose	Закрывает файл.

Команда	Описание
figure	Открывает окно графиков Figure.
fopen	Открывает файл.
global	Объявляет глобальные переменные.
help	Вывод на экран справки функций MATLAB.
iskeyword	Вывод на экран ключевых слов.
lookfor	Поиск указанного слова во всех записях справки.
who	Вывод на экран переменных, находящихся в настоящий момент в памяти.
whos	Вывод на экран информации о переменных в памяти.

Предопределенные переменные

Переменная	Описание
ans	Значение последнего выражения.
eps	Наименьшая разность между двумя числами.
i	Мнимая единица.
inf	Бесконечность.
j	То же что и i.
NaN	Неопределенность, не число.
pi	Число π .

Форматы вывода в командном окне

Команда	Описание
format bank	Две десятичных цифры.
format compact	Устраняет пустые строки.
format long	Фиксированная точка с 14 десятичными цифрами.
format long e	Экспоненциальное представление с 15 десятичными цифрами.
format long g	Лучшее 15-значное с фиксированной или плавающей запятой.

Команда	Описание
format loose	Добавляет пустые строки.
format short	Фиксированная точка с 4 десятичными цифрами.
format short e	Экспоненциальное представление с 4 десятичными цифрами.
format short g	Лучшее 5-значное с фиксированной или плавающей запятой.

Элементарные математические функции

Функция	Описание
abs	Абсолютное значение.
exp	Показательная функция.
factorial	Факториал.
log	Натуральный логарифм.
log10	Логарифм по основанию 10.
nthroot	Действительный корень степени n вещественного числа.
sqrt	Квадратный корень.

Тригонометрические математические функции

Функция	Описание	Функция	Описание
acos	Арккосинус.	cos	Косинус.
acot	Арккотангенс.	cot	Котангенс.
asin	Арксинус.	sin	Синус.
atan	Арктангенс.	tan	Тангенс.

Гиперболические математические функции

Функция	Описание	Функция	Описание
cosh	Гиперболический косинус.	sinh	Гиперболический синус.
coth	Гиперболический котангенс.	tanh	Гиперболический тангенс.

Округление

Функция	Описание
ceil	Округление в сторону бесконечности.
fix	Округление ближе к нулю.
floor	Округление в сторону минус бесконечности.
rem	Остаток от деления x на y.
round	Округление до самого близкого целого числа.
sign	Сигнум-функция.

Создание массивов

Функция	Описание
diag	Создает диагональную матрицу из вектора. Создает вектор из диагонали матрицы.
eye	Создает единичную матрицу.
linspace	Создает равномерно распределенный вектор.
ones	Создает массив единиц.
rand	Создает массив со случайными числами.
randi	Создает массив со случайными целыми числами.
randn	Создает массив с нормально распределенными числами.
randperm	Создает вектор со случайной перестановкой целых чисел.
zeros	Создает массив из нулей.

Обработка массивов

Функция	Описание
length	Число элементов у вектора.
reshape	Перестройка вида матрицы.
size	Размер массива.

Функции массива

Функция	Описание
cross	Вычисляет векторное произведение двух векторов.
det	Вычисляет определитель.
dot	Вычисляет скалярное произведение двух векторов.
inv	Вычисляет обратную квадратной матрицы.
max	Возвращает максимальное значение.
mean	Вычисляет среднее значение.
median	Вычисляет медиану.
min	Возвращает минимальное значение.
sort	Располагает элементы в порядке возрастания.
std	Вычисляет стандартное отклонение.
sum	Вычисляет сумму элементов.

Ввод и вывод

Команда	Описание
disp	Выводит на экран результат.
fprintf	Вывод на экран или сохранение результата.
input	Предложение пользователю для ввода данных.
load	Загружает переменные в рабочее пространство.
save	Сохраняет переменные из рабочего пространства.
uiimport	Запускает Мастера импорта.
xlsread	Импорта данных из Excel.
xlswrite	Экспорт данных в Excel.

Двумерные графики

Команда	Описание
bar	Создает график вертикальных баров.
barh	Создает график горизонтальных баров.

Команда	Описание
errorbar	Создает график со значениями погрешностей.
fplot	График функции.
hist	Создает гистограмму.
hold off	Конец удержания hold on.
hold on	Сохраняет открытый текущий график.
line	Добавляет кривые к существующему графику.
loglog	Создает график с логарифмическим масштабом на обеих осях.
pie	Создает круговой график.
plot	Создает график.
polar	Создает график в полярных координатах.
semilogx	Создает график с логарифмическим масштабом на оси X.
semilogy	Создает график с логарифмическим масштабом на оси Y.
stairs	Создает ступенчатый график.
stem	Создает график стеблей (stem).

Трехмерные графики

Команда	Описание
bar3	Создает вертикальный 3D график баров.
contour	Создает 2D контурный график.
contour3	Создает 3D контурный график.
cylinder	Графическое изображение цилиндра.
mesh	Создает график типа сетки.
meshc	Создает график типа сетки с контурами.
meshgrid	Создает сетку для 3D графика.
meshz	Создает график типа сетки с занавесом.
pie3	Объемный круговой график.
plot3	Создает график.

Команда	Описание
pol2cart	Преобразует сетку полярных координат в сетку декартовых координатах.
scatter3	Создает диаграмму рассеяния.
sphere	Графически изображает сферу.
stem3	Создает график стеблей.
surf	Создает график поверхности.
surfcb	Создает график поверхности и контуров.
surfl	Создает график поверхности с освещением.
waterfall	Создает сеточный график с эффектом водопада.

Форматирование графиков

Команда	Описание
axis	Устанавливает пределы осей.
colormap	Устанавливает цвет.
grid	Добавляет сетку к графику.
gtext	Добавляет текст к графику.
legend	Добавляет легенду графику.
subplot	Создает множественные графики на одной странице Figure.
text	Добавляет текст к графику.
title	Добавляет заголовок к графику.
view	Управляет направлением просмотра 3D графика.
xlabel	Добавляет метку к оси X.
ylabel	Добавляет метку к оси Y.

Математические функции (создание, вычисление, решение)

Команда	Описание
feval	Оценивает (вычисляет) значение математической функции.
fminbnd	Определяет минимум функции.
fzero	Решает уравнение с одной переменной.

Численное интегрирование

Функция	Описание
quad	Интегрирует функцию.
quadl	Интегрирует функцию.
trapz	Интегрирует функцию.

Решатели обыкновенных дифференциальных уравнений

Команда	Описание
ode113	Решает ОДУ первого порядка.
ode15s	Решает ОДУ первого порядка.
ode23	Решает ОДУ первого порядка.
ode23s	Решает ОДУ первого порядка.
ode23t	Решает ОДУ первого порядка.
ode23tb	Решает ОДУ первого порядка.
ode45	Решает ОДУ первого порядка.

Логические функции

Функция	Описание
all	Определяет, являются ли все элементы массива ненулевыми.
and	Логическое И (AND).
any	Определяет, являются ли какие-либо элементы массива ненулевыми.
find	Находит индексы определенных элементов вектора.
not	Логическое НЕТ (NOT).
or	Логическое ИЛИ (OR).
xor	Логическое исключающее ИЛИ.

Команды управления потоками

Команда	Описание
break	Завершает выполнение цикла.
case	Условное выполнение команды.

Команда	Описание
continue	Завершает проход в цикле.
else	Условное выполнение команды.
elseif	Условное выполнение команды.
end	Завершает условные утверждения и циклы.
for	Выполнение повторений группы команд.
if	Условное выполнение команды.
otherwise	Условное выполнение команды.
switch	Переключатели из нескольких случаев, основанных на выражении.
while	Выполнение повторений группы команд.

Полиномиальные функции

Функция	Описание
conv	Умножает многочлены.
deconv	Делит многочлены.
poly	Определяет коэффициенты многочлена.
polyder	Определяет производную многочлена.
polyval	Вычисляет значение многочлена.
roots	Определяет корни из многочлена.

Подбор аппроксимирующей кривой и интерполяция

Функция	Описание
interp1	Одномерная интерполяция.
polyfit	Полиномиальная подгонка кривой по множеству точек.

Символьная математика

Функция	Описание
collect	Собирает члены в выражении.
diff	Дифференцирует выражение.

Функция	Описание
double	Преобразовывает число из символьной формы в числовую форму
dsolve	Решает обыкновенное дифференциальное уравнение.
expand	Раскладывает выражение.
ezplot	График выражения.
factor	Разложение на множители более низких степеней.
findsym	Выводит на экран символьные переменные в выражении.
int	Интегрирует выражение.
pretty	Выводит на экран выражение в математическом формате.
simple	Находит форму выражения с наименьшим количеством символов.
simplify	Упрощает выражение.
solve	Решает одно уравнение, или систему уравнений.
subs	Подставляет числа в выражение.
sym	Создает символьный объект.
syms	Создает символьный объект.



ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

3D

3D графики в полярных координатах 342

А

Анонимная функция 240

В

Вложенные условные операторы 208

Вложенные функции 250

Вложенные циклы 208

Встроенные логические функции 191

Встроенные функции для анализа
массивов 86

Г

Генерация случайных чисел 88

Гистограммы 163

Графики в полярных координатах 166

График линии

двумерный 143

трехмерный 333

График поверхности 337

в виде сети 337

График функции 148

Графическое окно Figure 145

Д

Двоеточие 55

Деление многочленов 274

Дескриптор функции 244

Добавление элементов массива 58

Дополнительные команды 3D графики 338

И

Индексация массивов 53

Интерполяция

линейная 284

одномерная 284

сплайновая 284

Интерфейс для подбора Basic fitting 287

К

Ключевые слова 30

Команда

clc 22

diff 372

dsolve 375

ezplot 378

findsym 362

pretty 367

solve 367

subs 381

точка с запятой 22

Команда прерывания break 210

Команда продолжения continue 210

Командное окно 18, 21

Команды

collect, expand и factor 363

simplify и simple 365

sym и syms 357

zeros, ones, eye 51

Комментарий 22

Координатная сетка 335

Корни многочлена 272

Л

Логарифмический масштаб осей 158

Логические операторы 188

М

Массив

одномерный 46

Мастер импорта 126

Матрица

обратная 80

Метод наименьших квадратов 277

Минимум функции 308

Многочлен 270

Многочлен в MATLAB 271

О

Окно графиков 19
 Окно истории команд 22
 Окно рабочего пространства 107
 Окно редактора 19
 Окно редактора переменных 107
 Окно справки 20
 Оператор переключения 198
 Операторы сравнения 186
 Операция
 вычитания 75
 деления матриц 79
 левого деления матриц 81
 поэлементная 83
 правого деления матриц 82
 сложения 75
 транспонирования 52
 умножения матриц 76
 Определитель матрицы 81

П

Подбор кривой 276
 другими функциями 280
 многочленами 276
 Подфункции 249
 Пользовательская функция 229
 Порядок символьных переменных по умолчанию 362
 Предопределенные переменные 31
 Приоритет операций 23, 190
 Производная многочлена 275

Р

Регрессионный анализ 276
 Редактор графика 158
 Редактор-отладчик 33
 Решатели ОДУ 314
 Решение ОДУ 313

С

Свойства линии 147
 Символ
 процент 22
 Символьная переменная
 задание 357
 по умолчанию 362
 Символьное выражение
 графическое изображение 378
 математический формат 367
 наиболее короткий вид 366
 подстановка значения 381
 преобразование в числовую форму 361
 приведение подобных 363

разложение 364
 разложение на сомножители 365
 создание 359
 упрощение 365
 численные расчеты 381
 Символьное дифференцирование 372
 Символьное интегрирование 373
 Символьное решение
 алгебраических уравнений
 одного уравнения 367
 системы уравнения 369
 дифференциальных уравнений 375
 Символьные объекты 357
 Скрипт-файл 32, 105
 Сложение многочленов 273
 Создание
 вектора столбца 47
 вектора строки 47
 матрицы 50
 Специальные функции 3D графики 340
 Спецификаторы линии 145
 Строки символов 63

Т

Текущий каталог 34

У

Удаление элементов массива 59
 Умножение многочленов 274
 Условные операторы 193

Ф

Файл сценария 32, 105
 Файл функции 230
 Входные и выходные аргументы 232
 Глобальные переменные 235
 Локальные переменные 235
 Строка Н1 234
 Строка определения функции 231
 Текстовые строки справки 234
 Форматы вывода MATLAB 24
 Функции просмотра графиков 343
 Функции управления массивами 60
 Функция
 нахождения локального максимума 309
 нахождения локального минимума 308
 решения ОДУ 314
 решения уравнения 306
 численного интегрирования 310
 Функция от функции 243

Ц

Цикл
 for-end 201

while-end 205
Циклы 201

Э

Элементарные математические функции 25

А

axis 157

В

break 210

С

char 65
collect 363
continue 210
conv 274

Д

deconv 274
diff 372
disp 111
dsolve 375

Е

expand 364
ezplot 378

Ф

factor 365
findsym 362
fminbnd 308
fplot 150
fprintf 113
fzero 305

Г

grid on и grid off 158

Н

hold on и hold off 153

И

if-else-end 195
if-elseif-else-end 195
if-end 194
input 109
interp1 285

Л

legend 155

line 153
linspace 49
load 122

М

mesh 337
meshgrid 335

Р

plot 143
plot3 333
poly 273
polyder 275
polyfit 278
polyval 271
pretty 367

Q

quad 310
quadl 311

Р

rand 88
randi 89
randn 90
roots 272

С

save 121
simple 366
simplify 365
solve 367
subplot 167
subs 381
surf 337
switch-case 198
sym и syms 357

Т

text и gtext 155
title 155
trapz 312

В

view 343

Х

xlabel и ylabel 154
xlsread 124
xswrite 125

Книги издательства «ДМК Пресс» можно заказать в торгово-издательском холдинге «Планета Альянс» наложенным платежом, выслав открытку или письмо по почтовому адресу: **115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.**

При оформлении заказа следует указать адрес (полностью), по которому должны быть высланы книги; фамилию, имя и отчество получателя. Желательно также указать свой телефон и электронный адрес.

Эти книги вы можете заказать и в интернет-магазине: www.aliants-kniga.ru.

Оптовые закупки: тел. +7(499)782-38-89.

Электронный адрес: books@aliants-kniga.ru.

Амос Гилат

MATLAB. Теория и практика

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com

Перевод с английского *Смоленцев Н. К.*

Корректор *Синяева Г. И.*

Верстка *Паранская Н. В.*

Дизайн обложки *Мовчан А. Г.*

Формат 70×100¹/₁₆. Гарнитура «Петербург».

Печать офсетная. Усл. печ. л. 33,8.

Тираж 200 экз.

Веб-сайт издательства: www.dmk.ru