ВВЕДЕНИЕ

Название MatLab является сокращением от Matan. Первоначально пакет MatLab предназначался для матричных вычислений и был только удобной оболочкой для имеющихся библиотек программ, но за последнее десятилетие его возможности существенно возросли. В настоящее время MatLab является одним из самых мощных среди всех универсальных вычислительных пакетов.

Научные работники и инженеры применяют MatLab для решения задач, возникающих в различных прикладных областях. Обработка сигналов и изображений, исследование и расчет физических процессов, визуализация данных, статистический анализ, матричный анализ, оптимизация, нейронные сети, нечеткая логика, моделирование нелинейных динамических систем — вот далеко не полный перечень задач, которые могут быть эффективно решены в MatLab. Уникальность пакета MatLab состоит еще и в том, что он идеально приспособлен для лабораторной поддержки курсов по методам вычислений и информатике, читаемых студентам технических факультетов. Специализированные пакеты (ToolBox'ы), входящие в состав MatLab, могут быть использованы при проведении лабораторных работ на старших курсах технических факультетов. Интегрирование MatLab с Word позволяет получать интерактивные документы для разработки электронных курсов лекций и практических занятий.

Удобный интерфейс и встроенный простой язык программирования, простота которого компенсируется обширной библиотекой графических, вычислительных и сервисных функций, способствуют быстрому созданию приложений для исследования и решения поставленной задачи. Визуальная среда программирования позволяет при наличии некоторого опыта программировать приложения с графическим интерфейсом пользователя. Сочетание MatLab и Excel существенно расширяет возможности Excel при оперировании с данными. Программы, написанные на современных языках программирования высокого уровня, могут прекрасно взаимодействовать с приложениями, созданными в MatLab.

Данное пособие посвящено основам работы в MatLab и охватывает только те возможности пакета, которые необходимы при дальнейшем изучении более специального круга вопросов, например, численных методов или использованию ToolBox'ов. Описана работа из командной строки, вычисление арифметических выражений, использование одномерных и двумерных массивов. Следует подчеркнуть, что успешная работа в пакете невозможна без понимания принципов оперирования с массивами данных в MatLab. Рассмотрены графические возможности MatLab для визуализации данных и построения графиков функций. Приведены базовые конструкции встроенного языка программирования и показано их использование при обработке различных типов данных и организации файлового ввода-вывода.

Каждый параграф содержит варианты заданий для самостоятельной работы. Изучение материала рассчитано на семестр.

§ 1. ВЫЧИСЛЕНИЕ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ

Арифметические выражения в MatLab состоят, как и в большинстве языков программирования, из чисел, знаков арифметических операций, знака ^ (возведение в степень), круглых скобок, переменных, и встроенных функций. Десятичная часть числа отделяется точкой. Для вычисления простейшего выражения следует набрать его в командной строке и нажать <Enter>. Ответ записывается в специальную переменную `ans` и результат выводится в командное окно:

```
» 1.5+2.9
ans =
    4.4000
```

Вид результата зависит от установленного формата, подробнее о форматах вывода написано ниже. После вычисления следующего выражения значение `ans` изменится. Для сохранения результатов промежуточных вычислений их следует записывать в переменные. При использовании переменных необходимо придерживаться правил:

- имя переменной может состоять из символов латинского алфавита, знака подчёркивания и цифр, но начинается обязательно с символа алфавита;
- прописные и строчные буквы различаются;
- пробел не входит в имя переменной.

В качестве знака присваивания используется =, например:

```
» a=3.25*(0.7-3.3/5.1)+2.3^3
a =
   12.3391
```

Обратите внимание, что результат сразу же выводится в командное окно. Для подавления вывода следует завершить строку с оператором присваивания точкой с запятой. Символ `e` предназначен для за-

писи чисел в экспоненциальной форме: числа 0.00125 и 1.25e–3 эквивалентны. Комплексные числа вводятся при помощи буквы `i`:

```
» b=5*(2.2+3.9i)+0.8
b =
  11.8000 +19.5000i
```

MatLab обладает большим набором встроенных математических функций. Некоторые из них приведены в табл. 1.1. При вызове математических функций аргумент заключается в круглые скобки. Полный список всех встроенных элементарных математических функций можно получить, набрав в командной строке `help elfun`. Команда `help` отображает в командном окне список разделов справочной системы. Для получения содержимого раздела необходимо указать через пробел его название после `help`, а для вывода детальной информации о какой-либо функции, следует ввести в строке с `help` имя функции.

Таблица 1.1

Основные математические функции

Тригонометрические функции (аргумент задаётся в радианах)	
`sin, cos, tan, cot`	Синус, косинус, тангенс и котангенс
`sec, csc`	Секанс, косеканс
Обратные тригонометрические функции (результат вычисляется в радианах)	
`asin, acos, atan, acot`	Арксинус, арккосинус, арктангенс и арккотангенс
`asec, acsc`	Арксеканс, арккосеканс
Гиперболические функции	
`sinh, cosh, tanh, coth`	Гиперболические синус, косинус, тангенс и котангенс
`sech, csch`	Гиперболические секанс и косеканс
`asinh, acosh, atanh, acoth`	Гиперболические арксинус, арккосинус, арктангенс и арккотангенс;
Экспоненциальная функция, логарифмы, степенные функции	
`exp`	Экспоненциальная функция
`log, log2, log10`	Натуральный логарифм, логарифмы по основанию 2 и 10
`sqrt`	Квадратный корень
Модуль, знак и функции для работы с комплексными числами	
`abs, sign`	Модуль и знак числа
`conj, imag, real`	Комплексно-сопряжённое, мнимая и вещественная часть

Пусть, например, требуется найти значение выражения при $x = 0.2$ и $y = -3.9$:

$$c = \sqrt{\frac{\sin(\frac{4}{3}\pi x) + e^{0.1y}}{\cos(\frac{4}{3}\pi x) + e^{0.1y}}} + \sqrt[3]{\frac{\sin(\frac{4}{3}\pi x) + e^{0.1y}}{\cos(\frac{4}{3}\pi x) + e^{0.1y}}}$$

Если набирать сразу все выражение, то получается достаточно длинная строка. Для переноса на следующую строку любой команды MatLab можно использовать три идущие подряд точки, после нажатия на <Enter> среда MatLab ждет продолжения ввода:

```
» x=0.2;
» y=-3.9;
» c=sqrt((sin(4/3*pi*x)+exp(0.1*y))/(cos(4/3*pi*x)+exp(0.1*y)))+...
((sin(4/3*pi*x)+exp(0.1*y))/(cos(4/3*pi*x)+exp(0.1*y)))^(1/3)
```

Проще всего решить поставленную задачу, используя промежуточные переменные:

```
» x=0.2;
» y=-3.9;
» a=sin(4/3*pi*x)+exp(0.1*y);
» b=cos(4/3*pi*x)+exp(0.1*y);
» c=sqrt(a/b)+(a/b)^(1/3)
c =
    2.0451
```

Обратите внимание на несколько важных особенностей. Все операторы присваивания, кроме последнего, завершены точкой с запятой для подавления вывода результата. Необязательно набирать выражение для b, похожее на только что введенное для a. После ввода третьей строки нажмите клавишу <↑>. В командной строке появится предыдущее выражение, внесите в него необходимые изменения, а именно, замените sin на cos, и нажмите <Enter>. Клавиши <↑> и <↓> служат для перехода по истории команд, т.е. для занесения ранее набранных команд в командную строку, а <←>, <→>, <Home>, <End> — для перемещения в пределах командной строки. Передвижение по экрану (только для просмотра команд, а не для редактирования) осуществляется клавишами <PageUp>, <PageDown> или вертикальной полосой скроллинга. Начиная с версии 6.0 в среду MatLab включено окно Command History для быстрого перехода по истории команд. В любой момент можно вывести значение переменной в командное окно, для чего следует набрать имя переменной в командной строке и нажать <Enter>, либо вызвать функцию disp:

```
» disp(c)
    2.0451
```

Возникающий в процессе вычислений комплексный результат не является ошибкой. MatLab автоматически переходит в область комплексных чисел, продолжая вычисления. Найдите, например, квадратный корень из –1. Более того, допустимы операции деления на ноль, которые приводят к стандартным переменным Inf или –Inf. Результат деления нуля на ноль есть NaN (Not a Number — не число). Переполнение или потеря точности в MatLab при выполнении операций с числами с плавающей точкой не вызывает прекращение вычислений.

Просмотр текущих переменных рабочей среды производится при помощи команды whos. Предположим, что ранее переменным a и b были присвоены значения:

```
» a=-1.34;
» b=2.98+3.86i;
```

Вызовите команду whos, указав через пробелы имена переменных

```
» whos a b
```

В командное окно выводится таблица, приведённая ниже. В столбике Class указан тип переменной, в Bytes — число байт, выделенных под хранение значения, а Size содержит информацию о размере. После таблицы размещена строка с указанием суммарного объема памяти в байтах.

```
Name      Size         Bytes  Class
a         1x1              8  double array
b         1x1             16  double array (complex)
Grand total is 2 elements using 24 bytes
```

Обратите внимание, что числовые переменные в MatLab являются двумерными массивами размера один на один. Представление всех данных в MatLab в виде массивов оказывается очень полезным, о чем подробнее будет сказано в следующих разделах.

Пользователь имеет возможность управлять видом результата, устанавливая подходящие форматы вывода. Существует два способа задания форматов. Выбор пункта Preferences… в меню File рабочей среды приводит к появлению диалогового окна с одноименным названием. В MatLab 5.3 панель Numeric Format вкладки General содержит переключатели, а в версии 6.x на панели Text display расположены раскрывающиеся списки Numeric Format и Numeric display (при выбранном пункте Command Window в списке левой панели окна). Возможно установить один из следующих форматов.

- short (default) — короткий формат с плавающей точкой с четырьмя цифрами после десятичной точки (используется по умолчанию).
- long — длинный формат с плавающей точкой с четырнадцатью цифрами после десятичной точки.
- short e — экспоненциальный формат с четырьмя цифрами после десятичной точки.
- long e — экспоненциальный формат с пятнадцатью цифрами после десятичной точки.
- short g — наилучшее представление числа либо в формате short, либо в short e (аналогично long g).
- hex — шестнадцатеричное представление числа.
- + — положительные и отрицательные числа отображаются знаками "+" и "–", а нулевые - пробелами.
- bank — формат для вывода денежных сумм с двумя знаками после десятичной точки.
- rat — вещественные числа приближённо представляются отношением двух небольших целых чисел.

Иногда возникает ситуация, когда MatLab не может уложиться в установленный формат при выводе результата. Пусть, например, установлен формат short. При вычислении 1/3333 результат отобра-

жается в формате short e, однако, автоматической смены формата для всех последующих вычислений не происходит.

MatLab предоставляет два способа вывода в командное окно.

- § compact — строки с результатами выводятся подряд.
- § loose — строки с результатами разделяются пустой строкой.

Команда `format` служит для установки формата из командной строки. К примеру, обращение

```
» format short e
```

аналогично выбору короткого экспоненциального формата в диалоговом окне Preferences. Вне зависимости от установленного формата все вычисления производятся с двойной точностью, следовательно, после смены формата с short на long не требуется повторно находить значения переменных. Достаточно снова вывести их значения в командном окне.

Задания для самостоятельной работы

Во всех заданиях требуется занести в некоторую переменную значения выражений при заданных $x = -1.75\times10^{-3}$ и $y = 3.1\rho$, отобразить результат в различных форматах и изучить информацию о переменных при помощи команды `whos`.

1. $$F = \left(\frac{e^x \sin y + 2^x \cos y}{200x + y}\right)^{2.3} + \ln|\sin y| - \sqrt{\frac{e^x \sin y + 2^x \cos y}{200x + y}}$$

2. $$Z = \operatorname{arctg} \frac{\sqrt[3]{x - \sin y}}{\sqrt{1 - x^2}} - \frac{|x|\sqrt{1 - x^2}}{\sqrt[3]{x - \sin y}}$$

3. $$T = \frac{(\sin y + \sin 2y + \sin 3y)^4}{1 + \dfrac{\sin y + \sin 2y + \sin 3y}{e^x}} + \sqrt{1 + \frac{\sin y + \sin 2y + \sin 3y}{e^x}}$$

4. $$W = \left(1 + \frac{\ln y}{x + \operatorname{tg} y}\right)^{1 + \frac{x + \operatorname{tg} y}{\ln y}}$$

5. $$R = \operatorname{sh} \frac{(x + \ln y)^3}{\sqrt{|x - \ln y|}} \times \operatorname{ch}\left[(x + \ln y)\sqrt{|x - \ln y|}\right]$$

6. $$H = \frac{\sqrt{\cos 2y + \sin 4y + \sqrt{e^x + e^{-x}}}}{(e^{-x} + e^x)^3 (\sin 4y + \cos 2y - 2)^2}$$

7. $$Q = \sqrt{e^x \sin y + e^{-x} \cos y + \sqrt{1 + \frac{e^x \sin y + e^{-x} \cos y}{\operatorname{tg} y}}}$$

8. $$A = \sqrt[5]{x(1 + x)^2 (1 + 2x)^3 + \sqrt[3]{\frac{x(1 + x)^2 (1 + 2x)^3}{\ln|\operatorname{ctg} y|}}}$$

9. $$S = \operatorname{arctg} \sqrt{\left|\frac{x - \sin y}{x + \sin y} + \frac{x + \sin y}{x - \sin y}\right|} + e^{(x - \sin y)(x + \sin y)}$$

10. $$B = \frac{1 + \arcsin(\cos 2y)}{2^x + 3^{-x}} + \left(\frac{2^x + 3^{-x} - 1}{x + \arcsin(\cos 2y)}\right)^2$$

§ 2. ВЕКТОР–СТРОКИ И ВЕКТОР–СТОЛБЦЫ

Массивы являются одним из самых распространенных способов хранения данных и используются во всех языках программирования и вычислительных пакетах. К особенностям работы с массивами в MatLab относится то, что одномерный массив может быть вектор-строкой или вектор-столбцом. Если способ представления массива важен, то мы будем подчеркивать, о строке или о столбце идет речь. Если же это несущественно, то будем говорить о вектор-строках и вектор-столбцах просто как о векторах или одномерных массивах (одномерный массив в MatLab есть двумерный, у которого один из размеров равен единице).

Для ввода вектора используются квадратные скобки, элементы вектора отделяются друг от друга:

- точкой с запятой, если требуется получить вектор–столбец;
- пробелом или запятой, если необходимо разместить элементы в вектор–строке.

Занесите вектор-столбцы и вектор-строки

$$a=\begin{bmatrix}0.2\\-3.9\\4.6\end{bmatrix}\quad b=\begin{bmatrix}7.6\\0.1\\2.5\end{bmatrix}\quad u=\begin{bmatrix}0.1 & 0.5 & -3.7 & 8.1\end{bmatrix}\quad v=\begin{bmatrix}5.2 & 9.7 & 3.4 & -0.2\end{bmatrix}$$

в соответствующие массивы, набрав в командной строке:

```
» a=[0.2; -3.9; 4.6];
» b=[7.6; 0.1; 2.5];
» u=[0.1 0.5 -3.7 8.1];
» v=[5.2 9.7 3.4 –0.2];
```

Точка с запятой в конце каждой строки поставлена для подавления вывода на экран, она никак не связана с точкой с запятой, которая является разделителем элементов в вектор-столбцах. Выведите в командное окно значения переменных `a`, `b`, `u`, `v` и посмотрите, как MatLab отображает содержимое вектор-строк и вектор-столбцов. Получите информацию о переменных при помощи команды `whos`. В предыдущем параграфе было замечено, что числа хранятся в двумерных массивах, каждый из размеров которых равен единице. Векторы также представляются двумерными массивами, один из размеров которых равен единице.

Для получения длины вектора предназначена функция `length`, вектор указывается в качестве ее входного аргумента:

```
» L=length(a)
L =
     3
```

Вектор-столбцы с одинаковым числом элементов можно складывать и вычитать друг из друга при помощи знаков "+" и "–". Аналогичное верно и для вектор-строк:

```
» c=a+b;
» w=u-v;
```

Сложение и вычитание вектор-строки и вектор-столбца или векторов разных размеров приводит к ошибке. Операция * предназначена для умножения векторов по правилу матричного умножения. Поскольку MatLab различает вектор-строки и вектор столбцы, то допустимо либо умножение вектор-строки на такой же по длине вектор-столбец (скалярное произведение), либо умножение вектор-столбца на вектор-строку (внешнее произведение, в результате которого получается прямоугольная матрица). Скалярное произведение двух векторов возвращает функция `dot`, а векторное — `cross`:

```
» s=dot(a,b)
» c=cross(a,b)
```

Разумеется, векторное произведение определено только для векторов из трех элементов.

Для операции транспонирования зарезервирован апостроф `'`. Если вектор содержит комплексные числа, то операция `'` приводит к комплексно-сопряженному вектору. При вычислении скалярного и векторного произведений функциями `cross` и `dot` не обязательно следить за тем, чтобы оба вектора были либо столбцами, либо строками. Результат получается верный, например, при обращении `c=cross(a,b')`, только `c` становится вектор-строкой.

MatLab поддерживает поэлементные операции с векторами. Наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения `.*` (точка со звездочкой). Данная операция применяется к векторам одинаковой длины и приводит к вектору той же длины, что исходные, элементы которого равны произведениям соответствующих элементов исходных векторов.

Например, для векторов a и b, введенных выше, поэлементное умножение дает следующий результат:

```
» c=a.*b
c =
    1.5200
   -0.3900
   11.5000
```

Аналогичным образом работает поэлементное деление `./` (точка с косой чертой). Кроме того, операция `.\` (точка с обратной косой чертой) осуществляет обратное поэлементное деление, то есть выражения `a./b` и `b.\a` эквивалентны. Возведение элементов вектора a в степени, равные соответствующим элементам b, производится с использованием `.^`. Для транспонирования вектор-строк или вектор-столбцов предназначено сочетание `.'` (точка с апострофом). Операции `'` и `.'` для вещественных векторов приводят к одинаковым результатам. Не обязательно применять поэлементные операции при умножении вектора на число и числа на вектор, делении вектора на число, сложении и вычитании вектора и числа. При выполнении, например, операции `a*2`, результат представляет собой вектор того же размера, что и a, с удвоенными элементами.

Векторы могут быть аргументами встроенных математических функций, таких, как `sin`, `cos` и т. д. В результате получается вектор с элементами, равными значению вызываемой функции от соответствующих элементов исходного вектора, например:

```
» q=sin([0 pi/2 pi])
q =
         0    1.0000    0.0000
```

Однако для вычисления более сложной функции от вектора значений, скажем

$$\frac{v\sin v + v^2}{v+1},$$

выражение `f=(v*sin(v)+v^2)/(v+1)` вызовет ошибку уже при попытке умножения v на `sin(v)`. Дело в том, что v является вектор-строкой длиной четыре, т. е. хранится в двумерном массиве размером один на четыре. Точно также представлен и `sin(v)`, следовательно, умножение при помощи звездочки (по правилу матричного умножения) лишено смысла. Аналогичная ситуация возникает и при возведении вектора v в квадрат, т. е., фактически, при вычислении `v*v`. Правильная запись выражения в MatLab требует использования поэлементных операций:

```
» f=(v.*sin(v)+v.^2)./(v+1)
```

Часто требуется вычислить функцию от вектора значений аргумента, отличающихся друг от друга на постоянный шаг. Для создания таких вектор-строк предусмотрено двоеточие. Последовательность команд

```
» x=-1.2:0.5:1.8;
» f=(x.*sin(x)+x.^2)./(x+1);
```

приводит к заполнению следующих векторов:

```
» x
x =
   -1.2000 -0.7000 -0.2000  0.3000  0.8000  1.3000  1.8000
» f
f =
  -12.7922  3.1365  0.0997  0.1374  0.6744  1.2794  1.7832
```

Шаг может быть отрицательным, в этом случае начальное значение должно быть больше, либо равно конечному для получения непустого вектора. Если шаг равен единице, то его можно не указывать, например:

```
» n=-3:4
n =
    -3    -2    -1     0     1     2     3     4
```

Ясно, что для заполнения вектор-столбца элементами с постоянным шагом следует транспонировать вектор-строку. Создание векторов при помощи двоеточия и умение производить поэлементные операции необходимо для визуализации массивов данных, о чем будет сказано в следующих разделах.

MatLab обладает большим набором встроенных функций для обработки векторных данных, часть из них приведена в табл. 2.1. Полный список имеющихся функций выводится в командное окно

при помощи `help datafun`, а для получения подробной информацию о каждой функции требуется указать ее имя в качестве аргумента команды `help`. Обратите внимание на то, что ряд функций допускает обращение к ним как с одним, так и с двумя выходными аргументами. В случае нескольких выходных аргументов они заключаются в квадратные скобки и отделяются друг от друга запятой.

Очень часто требуется обработать только часть вектора, или обратиться к некоторым его элементам. Разберем правила MatLab, по которым производится индексация векторных данных. Для доступа к элементу вектора необходимо указать его номер в круглых скобках сразу после имени переменной, в которой содержится вектор. Например, сумма первого и третьего элементов вектора `v` находится при помощи выражения

```
» s=v(1)+v(3);
```

Обращение к последнему элементу вектора можно произвести с использованием `end`, т.е. `v(end)` и `v(length(v))` приводят к одинаковым результатам.

Таблица. 2.1

Функции обработки данных

Функции	Назначение
`s=sum(a)`	Сумма всех элементов вектора `a`
`p=prod(a)`	Произведение всех элементов вектора `a`
`m=max(a)`	Нахождение максимального значения среди элементов вектора `a`
`[m,k]=max(a)`	Второй выходной аргумент `k` содержит номер максимального элемента в векторе `a`
`m=min(a)`	Нахождение минимального значения среди элементов вектора `a`
`[m,k]=min(a)`	Второй выходной аргумент `k` содержит номер минимального элемента в векторе `a`
`m=mean(a)`	Вычисление среднего арифметического элементов вектора `a`
`a1=sort(a)`	Упорядочение элементов вектора по возрастанию
`[a1,ind]=sort(a)`	Второй выходной аргумент `ind` является вектором из целых чисел от 1 до `length(a)`, который соответствует проделанным перестановкам

Указание номеров элементов вектора можно использовать и при вводе векторов, последовательно добавляя новые элементы (не обязательно в порядке возрастания их номеров). Команды:

```
» h=10;
» h(2)=20;
» h(4)=40;
```

приводят к образованию вектора:

```
» h
h =
    10    20     0    40
```

Заметьте, что для ввода первого элемента `h` не обязательно указывать его индекс, т.к. при выполнении оператора `h=1` создается вектор (массив размера один на один). Следующие операторы присваивания приводят к автоматическому увеличению длины вектора `h`, а пропущенные элементы (в нашем случае `h(3)`) получают значение ноль.

Индексация двоеточием позволяет выделить идущие подряд элементы в новый вектор. Начальный и конечный номера указываются в круглых скобках через двоеточие, например:

```
» z=[0.2 -3.8 7.9 4.5 7.2 -8.1 3.4];
» znew=z(3:6)
znew =
    7.9000    4.5000    7.2000   -8.1000
```

Применение встроенных функций обработки данных к некоторым последовательно расположенным элементам вектора не представляет труда. Следующий вызов функции `prod` вычисляет произведение элементов вектора `z` со второго по шестой:

```
» p=prod(z(2:6))
```

Индексация вектором служит для выделения элементов с заданными индексами в новый вектор. Индексный вектор должен содержать номера требуемых элементов, например:

```
» ind=[3 5 7];
» znew=z(ind)
znew =
    7.9000    7.2000    3.4000
```

Подумайте, как найти сумму элементов произвольного вектора `z` с четными индексами. Вот правильное решение:

```
» ind=2:2:length(z);
» s=sum(z(ind))
```

Конструирование новых векторов из элементов имеющихся векторов производится при помощи квадратных скобок. Следующий оператор приводит к образованию вектора, в котором пропущен пятый элемент вектора `z`

```
» znew=[z(1:4) z(6:end)]
znew =
    0.2000   -3.8000    7.9000    4.5000   -8.1000    3.4000
```

Задания для самостоятельной работы

Для заданных векторов *a* и *b* длины *n*:

1) вычислить их сумму, разность и скалярное произведение;

2) образовать вектор $c = [a_1, a_2, ..., a_n, b_1, b_2, ..., b_n]$, определить его максимальный и минимальный элементы и поменять их местами;

3) упорядочить вектор *c* по возрастанию и убыванию;

4) переставить элементы вектора *c* в обратном порядке и записать результат в новый вектор;

5) найти векторное произведение $u = [a_1, a_3, a_4]$ и $v = [b_2, b_3, b_4]$.

Варианты

1. *a* = [0.5 3.7 6.0 -4.3 1.2 -2.7 2.4 2.2]; *b* = [3.6 7.0 7.0 5.4 2.6 -2.7 -6.4 0.3].
2. *a* = [-4.8 -1.3 -1.0 0.7 4.0 5.8 4.3 -8.0]; *b* = [-1.1 -1.9 7.1 -2.1 6.8 2.8 0.3 1.6].
3. *a* = [1.0 -3.9 -2.3 -3.3 -1.7 2.2 -0.6 1.8]; *b* = [2.7 -2.7 -2.2 4.4 0.4 -6.0 -3.4 -5.2].
4. *a* = [-2.4 3.3 -0.1 3.6 7.4 -2.8 0.3 2.2]; *b* = [6.3 0.6 4.3 -3.7 -7.0 3.7 3.7 8.0].
5. *a* = [8.4 -5.9 -6.5 -0.9 6.9 -1.7 1.7 0.8]; *b* = [-0.0 2.0 -1.5 7.5 -4.0 -3.0 -6.2 0.0].
6. *a* = [5.3 6.8 -7.1 6.8 -4.0 -2.3 -4.4 -0.2]; *b* = [7.5 -1.5 -4.9 -4.6 -2.3 -5.3 5.5 2.3].
7. *a* = [1.2 -4.1 -0.8 -0.7 -2.2 1.7 3.3 -6.1]; *b* = [-1.5 2.2 1.0 -4.3 -0.0 -1.8 -1.5 2.4].
8. *a* = [6.6 -5.0 -2.7 8.3 3.8 1.9 1.1 2.7]; *b* = [-1.0 3.2 4.2 -6.4 1.9 -6.5 -6.2 -8.1].
9. *a* = [-1.9 0.4 1.8 4.2 -3.8 -4.7 4.0 -2.1]; *b* = [-8.7 -4.2 -1.4 2.8 -2.2 7.8 0.0 -0.1].
10. *a* = [0.9 1.7 -3.2 -3.8 7.3 6.0 -0.2 8.6]; *b* = [0.6 -0.4 -6.9 -2.2 1.6 3.8 -3.2 0.4].

Задания для самостоятельной работы

Вычислить значения функции на отрезке в заданном числе N равномерно отстоящих друг от друга точек.

Варианты

1. $y(x) = \dfrac{\sin x \cos x}{x^2 + 1}$ $[0, 2\pi]$ $N = 10$
2. $y(x) = \ln(x+1)\sqrt{e^x + e^{-x}}$ $[-0.2, 4]$ $N = 8$
3. $y(x) = x^2 \mathrm{tg}(\sqrt{\arcsin(x)})$ $[0, \frac{1}{3}]$ $N = 9$

4. $y(x) = x\sin(x) + x^3 \dfrac{e^x}{x+1}$ $[0,1]$ $N = 7$

5. $y(x) = \dfrac{1}{1 + \frac{x}{\sqrt{1+x}}}$ $[0,3]$ $N = 9$

6. $y(x) = \dfrac{e^{\sin(x)} + e^{\cos(x)}}{x^2}$ $[p, 3p]$ $N = 11$

7. $y(x) = \mathrm{ctg}(x^2 + 1) \times (\sin(2x) + \cos(2x))$ $[-1,1]$ $N = 7$

8. $y(x) = \log_2(x^2 + 1) \times \sin\dfrac{1}{x^2 + 1}$ $[-1,1]$ $N = 10$

9. $y(x) = |x^3 + 2x^2 - 3| \sin(px)$ $[-2,2]$ $N = 7$

10. $y(x) = \dfrac{\sqrt[3]{x+1}}{\sqrt{|x| + \frac{1}{2}}} \times \dfrac{\sin(x)+1}{\cos(x)+2}$ $[-2,2]$ $N = 9$

§ 3. МАТРИЦЫ

Матрицы небольших размеров удобно вводить из командной строки. Существует три способа ввода матриц. Например, матрицу

$$A = \begin{bmatrix} 0.7 & -2.5 & 9.1 \\ 8.4 & 0.3 & 1.7 \\ -3.5 & 6.2 & 4.7 \end{bmatrix}$$

можно ввести следующим образом: набрать в командной строке (разделяя элементы строки матрицы пробелами): `A=[0.7 -2.5 9.1` и нажать <Enter>. Курсор перемещается в следующую строку (символ приглашения командной строки » отсутствует). Элементы каждой следующей строки матрицы набираются через пробел, а ввод строки завершается нажатием на <Enter>. При вводе последней строки в конце ставится закрывающая квадратная скобка:

```
» A=[0.7 -2.5 9.1
8.4 0.3 1.7
-3.5 6.2 4.7]
```

Если после закрывающей квадратной скобки не ставить точку с запятой для подавления вывода в командное окно, то матрица выведется в виде таблицы.

Другой способ ввода матрицы основан на том, что матрицу можно рассматривать как вектор-столбец, каждый элемент которого является строкой матрицы. Поскольку точка с запятой используется для разделения элементов вектор-столбца, то ввод, к примеру, матрицы

$$B = \begin{bmatrix} 6.1 & 0.3 \\ -7.9 & 4.4 \\ 2.5 & -8.1 \end{bmatrix}$$

осуществляется оператором присваивания:

```
» B=[6.1 0.3; -7.9 4.4; 2.5 -8.1];
```

Введите матрицу B и отобразите ее содержимое в командном окне, набрав в командной строке `B` и нажав <Enter>.

Очевидно, что допустима такая трактовка матрицы, при которой она считается вектор-строкой, каждый элемент которой является столбцом матрицы. Следовательно, для ввода матрицы

$$C = \begin{bmatrix} 0.4 & -7.2 & 5.3 \\ 0.1 & -2.1 & -9.5 \end{bmatrix}$$

достаточно воспользоваться командой:

```
» C=[[0.4; 0.1] [-7.2; -2.1] [5.3; -9.5]]
```

Обратите внимание, что внутренние квадратные скобки действительно нужны. Оператор `C=[0.4; 0.1 -7.2; -2.1 5.3; -9.5]`является недопустимым и приводит к сообщению об ошибке, поскольку оказывается, что в первой строке матрицы содержится только один элемент, во второй и третьей — по два, а в четвертой — снова один.

Воспользуйтесь командой `whos` для получения информации о переменных `A`, `B` и `C` рабочей среды. В командное окно выводится таблица с информацией о размерах массивов, памяти, необходимой для хранения каждого из массивов, и типе — `double array`:

```
» whos A B C
  Name      Size         Bytes  Class
  A         3x3             72  double array
  B         3x2             48  double array
  C         2x3             48  double array
```

Функция `size` позволяет установить размеры массивов, она возвращает результат в виде вектора, первый элемент которого равен числу строк, а второй — столбцов:

```
» s=size(B)
s =
     3     2
```

Сложение и вычитание матриц одинаковых размеров производится с использованием знаков `+`, `-`. Звездочка `*` служит для вычисления матричного произведения, причем соответствующие размеры матриц должны совпадать, например:

```
» P=A*B
P =
   46.7700  -84.5000
   53.1200   -9.9300
  -58.5800  -11.8400
```

Допустимо умножение матрицы на число и числа на матрицу, при этом происходит умножение каждого элемента матрицы на число и результатом является матрица тех же размеров, что и исходная. Апостроф `'` предназначен для транспонирования вещественной матрицы или нахождения сопряженной к комплексной матрице. Для возведения квадратной матрицы в степень применяется знак `^`.

Вычислите для тренировки матричное выражение $R = (A - BC)^3 + ABC$, в котором A, B и C — определенные выше матрицы. Ниже приведена запись[1] в MatLab этого выражения:

```
» R=(A-B*C)^3+A*B*C
R =
  1.0e+006 *
   -0.0454    0.1661   -0.6579
    0.0812   -0.2770    1.2906
   -0.0426    0.1274   -0.7871
```

MatLab обладает многообразием различных функций и способов для работы с матричными данными. Для обращение к элементу двумерного массива следует указать его строчный и столбцевой индексы в круглых скобках после имени массива, например:

```
» C(1,2)
ans =
     -7.2000
```

Индексация двоеточием позволяет получить часть матрицы — строку, столбец или блок, например:

```
» c1=A(2:3,2)
c1 =
    0.3000
    6.2000
» r1=A(1,1:3)
r1 =
    0.7000   -2.5000    9.1000
```

1 Обратите внимание на общий множитель 1.0e+006, который относится ко всем элементам матрицы.

Для обращения ко всей строке или всему столбцу не обязательно указывать через двоеточие начальный (первый) и конечный индексы, то есть операторы `r1=A(1,1:3)` и `r1=A(1,:)` эквивалентны. Для доступа к элементам строки или столбца от заданного до последнего можно использовать `end`, так же как и для векторов: `A(1,2:end)`. Выделение блока, состоящего из нескольких строк и столбцов, требует индексации двоеточием как по первому измерению, так и по второму. Пусть в массиве `T` хранится матрица:

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

Для выделения ее элементов (обозначенных курсивом) со второй строки по третью и со второго столбца по четвертый, достаточно использовать оператор:

```
» T1=T(2:3,2:4)
T1 =
    -5    -6     3
     4     5    -1
```

Индексация двоеточием так же очень полезна при различных перестановках в массивах. В частности, для перестановки первой и последней строк в произвольной матрице, хранящейся в массиве `A`, подойдет последовательность команд:

```
» s=A(1,:);
» A(1,:)=A(end,:);
» A(end,:)=s;
```

MatLab поддерживает такую операцию, как вычеркивание строк или столбцов из матрицы. Достаточно удаляемому блоку присвоить пустой массив, задаваемый квадратными скобками. Например, вычеркивание второй и третьей строки из массива `T`, введенного выше, производится следующей командой:

```
» T(2:3,:)=[]
T =
     1     7    -3     2     4     9
    -6    -4     7     2     6     1
```

Индексация двоеточием упрощает заполнение матриц, имеющих определенную структуру. Предположим, что требуется создать матрицу

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Первый шаг состоит в определении нулевой матрицы размера пять на пять, затем заполняются первая и последняя строки и первый и последний столбцы:

```
» W(1:5,1:5)=0;
» W(1,:)=1;
» W(end,:)=1;
» W(:,1)=1;
» W(:,end)=1;
```

Проверьте, что в результате получается требуемая матрица. Ряд встроенных функций, приведенных в табл. 3.1, позволяет ввести стандартные матрицы заданных размеров. Обратите внимание, что во всех функциях, кроме `diag`, допустимо указывать размеры матрицы следующими способами:

- числами через запятую (в двух входных аргументах);
- одним числом, результат — квадратная матрица;
- вектором из двух элементов, равных числу строк и столбцов.

Последний вариант очень удобен, когда требуется создать стандартную матрицу тех же размеров, что и некоторая имеющаяся матрица. Если, к примеру, `A` была определена ранее, то команда `I=eye(size(A))` приводит к появлению единичной матрицы, размеры которой совпадают с размерами `A`, так как функция `size` возвращает размеры матрицы в векторе.

Разберем, как получить трехдиагональную матрицу размера семь на семь, приведенную ниже, с использованием функций MatLab.

$$T = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 5 & 3 & -3 & 0 & 0 & 0 \\ 0 & 0 & 5 & 4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 5 & -5 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & -6 \\ 0 & 0 & 0 & 0 & 0 & 5 & 7 \end{pmatrix}$$

Введите вектор `v` с целыми числами от одного до семи и используйте его для создания диагональной матрицы и матрицы со смещенной на единицу вверх диагональю. Вектор длины шесть, содержащий пятерки, заполняется, например, так: `5*ones(1,6)`. Этот вектор укажите в первом аргументе функции `diag`, а минус единицу — во втором и получите третью вспомогательную матрицу. Теперь достаточно вычесть из первой матрицы вторую и сложить с третьей:

```
» T=diag(v)-diag(v(1:6),1)+diag(5*ones(1,6),-1)
```

Таблица. 3.1.

Функции для создания стандартных матриц

Функция	Результат и примеры вызовов
`zeros`	Нулевая матрица `F=zeros(4,5)` `F=zeros(3)` `F=zeros([3 4])`
`eye`	Единичная прямоугольная матрица (единицы расположены на главной диагонали) `I=eye(5,8)` `I=eye(5)` `I=eye([5 8])`
`ones`	Матрица, целиком состоящая из единиц `E=ones(3,5)` `E=ones(6)` `E=ones([2 5])`
`rand`	Матрица, элементы которой — случайные числа, равномерно распределенные на интервале (0,1) `R=rand(5,7)` `R=rand(6)` `R=rand([3 5])`
`randn`	Матрица, элементы которой — случайные числа, распределенные по нормальному закону с нулевым средним и дисперсией равной единице `N=randn(5,3)` `N=randn(9)` `N=randn([2 4])`

`diag`	1) диагональная матрица, элементы которой задаются во входном аргументе векторе
	`D=diag(v)`
	2) диагональная матрица со смещенной на `k` позиций диагональю (положительные `k` — смещение вверх, отрицательные — вниз), результатом является квадратная матрица размера `length(v)+abs(k)`
	`D=diag(v,k)`
	3) выделение главной диагонали из матрицы в вектор
	`d=diag(A)`
	4) выделение k-ой диагонали из матрицы в вектор
	`d=diag(A,k)`

В предыдущем параграфе было описано применение поэлементных операций к векторам. Поэлементные вычисления с матрицами производятся практически аналогично, разумеется, необходимо следить за совпадением размеров матриц:

`A.*B`, `A./B`— поэлементные умножение и деление;

`A.^p` — поэлементное возведение в степень, `p` — число;

`A.^B` — возведение элементов матрицы `A` в степени, равные соответствующим элементам матрицы `B`;

`A.'` — транспонирование матрицы (для вещественных матриц `A'` и `A.'` приводят к одинаковым результатам);

Иногда требуется не просто транспонировать матрицу, но и "развернуть" ее. Разворот матрицы на 90^o против часовой стрелки осуществляет функция `rot90`:

```
» Q=[1 2;3 4]
Q =
     1     2
     3     4
» R=rot90(Q)
R =
     2     4
     1     3
```

Допустимо записывать сумму и разность матрицы и числа, при этом сложение или вычитание применяется, соответственно, ко всем элементам матрицы. Вызов математической функции от матрицы приводит к матрице того же размера, на соответствующих позициях которой стоят значения функции от элементов исходной матрицы.

В MatLab определены и матричные функции, например, `sqrtm` предназначена для вычисления квадратного корня. Найдите квадратный корень из матрицы

$$K = \begin{pmatrix} 3 & 2 \\ 1 & 4 \end{pmatrix}$$

и проверьте полученный результат, возведя его в квадрат (по правилу матричного умножения, а не поэлементно!):

```
» K=[3 2; 1 4];
» S=sqrtm(K)
S =
    1.6882    0.5479
    0.2740    1.9621
» S*S
ans =
    3.0000    2.0000
    1.0000    4.0000
```

Матричная экспонента вычисляется с использованием `expm`. Специальная функция `funm` служит для вычисления произвольной матричной функции[2].

Все функции обработки данных, приведенные в табл. 2.1, могут быть применены и к двумерным массивам. Основное отличие от обработки векторных данных состоит в том, что эти функции работают с двумерными массивами по столбцам, например, функция `sum` суммирует элементы каждого из столбцов и возвращает вектор-строку, длина которой равна числу столбцов исходной матрицы:

```
» M=[1 2 3; 4 5 6; 7 8 9]
M =
     1     2     3
     4     5     6
     7     8     9
» s=sum(M)
s =
    12    15    18
```

Для вычисления суммы всех элементов матрицы требуется дважды применить `sum`:

```
» s=sum(sum(M))
s =
    45
```

Очень удобной возможностью MatLab является конструирование матрицы из матриц меньших размеров. Пусть заданы матрицы:

[2] Для использования funm необходимо создать файл-функцию. Файл-функциям посвящен § 5.

$$M1=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\qquad M2=\begin{bmatrix}-2 & -3 & -5\\ -1 & -5 & -6\end{bmatrix}\qquad M3=\begin{bmatrix}9 & 8\\ -7 & -5\\ 1 & 2\end{bmatrix}\qquad M4=\begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix}$$

Требуется составить из $M1$, $M2$, $M3$ и $M4$ блочную матрицу M

$$M=\left[\begin{array}{c|c} M1 & M2\\ \hline M3 & M4\end{array}\right]$$

Можно считать, что M имеет размеры два на два, а каждый элемент является, соответственно, матрицей $M1$, $M2$, $M3$ или $M4$. Следовательно, для получения в рабочей среде MatLab массива M с матрицей M требуется использовать оператор:

```
» M=[M1 M2; M3 M4]
```

Задания для самостоятельной работы

Введите матрицы

$$A=\begin{bmatrix}-9.8 & 4.4 & 1.3\\ -5.7 & 0.1 & 0.8\\ 2.4 & 4.4 & 8.6\end{bmatrix}\qquad B=\begin{bmatrix}1 & 0 & 2\\ 3 & 0 & -1\\ 5 & 2 & 2\\ 8 & 9 & 0\end{bmatrix}\qquad C=\begin{bmatrix}0.1 & 0.2 & -1.3 & 0.7\\ -0.2 & 0.3 & 2.2 & 0.8\\ 1.9 & 2.3 & 6.5 & 4.9\end{bmatrix}$$

и найдите значения следующих выражений.

Варианты

1. $\left(A^3+CB\right)\left(A^2-3CB\right)^T$.
2. A^4+2A^3-ACB.
3. $BAC-4C^TB^T$.
4. $3BA^3C-BAC+2BC$.
5. $-3C^TAC-BB^T$.
6. $\left(BCB-4C^T\right)A^4$.
7. $\left(AB^T-C\right)\left(C+AB^T\right)^T-3A$.
8. $\left(AB^TB\right)^4-2A^3+CC^T$.
9. $C\left(BB^T+C^TC\right)C^T-8A$.
10. $2AA^T-\left(CB\right)^2+4A$.

Задания для самостоятельной работы

При помощи встроенных функций для заполнения стандартных матриц, индексации двоеточием и, возможно, поворота, транспонирования или вычеркивания получите следующие матрицы:

Варианты

1.
$$\begin{bmatrix}1&1&1&1&1&1&1\\ 1&2&0&0&0&7&1\\ 1&0&2&0&0&0&1\\ 1&0&0&2&0&0&1\\ 1&0&0&0&2&0&1\\ 1&0&0&0&0&2&1\\ 1&1&1&1&1&1&1\end{bmatrix}$$

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

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

4.
$$\begin{bmatrix}4&3&3&3&3&3&9\\ 3&4&3&3&3&3&3\\ 3&3&4&3&3&3&3\\ 3&3&3&4&3&3&3\\ 3&3&3&3&4&3&3\\ 3&3&3&3&3&4&3\\ 9&3&3&3&3&3&4\end{bmatrix}$$

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

6.
$$\begin{bmatrix}1&0&0&0&0&0&-1\\ 0&1&0&0&0&-1&0\\ 0&0&1&0&-1&0&0\\ 0&0&0&5&0&0&0\\ 0&0&-1&0&1&0&0\\ 0&-1&0&0&0&1&0\\ -1&0&0&0&0&0&1\end{bmatrix}$$

7.
$$\begin{bmatrix}2&1&1&1&1&1&1&1&1&1&1\\ 1&2&0&0&0&0&0&0&0&0&1\\ 1&0&2&0&0&0&0&0&0&0&1\\ 1&0&0&2&0&0&0&0&0&0&1\\ 1&0&0&0&2&0&0&0&0&0&1\\ 1&0&0&0&0&2&0&0&0&0&1\\ 1&1&1&1&1&1&2&1&1&1&1\end{bmatrix}$$

8.
$$\begin{bmatrix}0&1&1&1&1&1&1&1&1&1&1\\ 1&0&1&1&1&1&1&1&1&1&1\\ 1&1&0&1&1&1&1&1&1&1&1\\ 1&1&1&0&1&1&1&1&1&1&1\\ 1&1&1&1&0&1&1&1&1&1&1\\ 1&1&1&1&1&0&1&1&1&1&1\\ 0&0&0&0&0&0&0&0&0&0&0\end{bmatrix}$$

9. $\begin{bmatrix}1&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&1&0&0\end{bmatrix}$ 10. $\begin{bmatrix}1&0&0&0&1&0&0&0&0&0\\0&1&0&0&0&1&0&0&0&0\\0&0&1&0&0&0&1&0&0&0\\0&0&0&1&0&0&0&1&0&0\\0&0&0&0&1&0&0&0&1&0\\0&0&0&0&0&1&0&0&0&1\end{bmatrix}$

Задания для самостоятельной работы

Вычислить значения функции для всех элементов матрицы и записать результат в матрицу того же размера, что и исходная.

Варианты

1. $f(x)=x^3-2x^2+\sin x-4;$ $A=\begin{bmatrix}9.33&-4.01&8.19&2.64\\0.55&3.81&3.32&5.07\end{bmatrix}.$
2. $f(x)=\dfrac{e^x-x}{e^x+x};$ $A=\begin{bmatrix}9.32&0.21&-9.89&3.11\\0.54&4.99&5.01&-0.03\end{bmatrix}.$
3. $f(x)=\sqrt{1+\sqrt{|x|^3+1}};$ $A=\begin{bmatrix}-1.54&0.49&3.11&2.99\\4.05&-5.85&3.72&0.11\end{bmatrix}.$
4. $f(x)=e^x\sin x-e^{-x}\cos(x);$ $A=\begin{bmatrix}-9.04&3.36&3.09&-2.49\\-4.33&-5.09&9.74&1.65\end{bmatrix}.$
5. $f(x)=\ln(|x|)\sin(\pi x);$ $A=\begin{bmatrix}0.33&0.95&7.12&-9.22\\-0.64&3.76&1.34&-0.03\end{bmatrix}.$
6. $f(x)=e^{x^2+x+1};$ $A=\begin{bmatrix}-4.53&-2.12&-6.54&-3.21\\3.43&7.43&-0.25&1.64\end{bmatrix}.$
7. $f(x)=\dfrac{\sqrt[3]{x^2-1}}{|x|+3};$ $A=\begin{bmatrix}0.23&3.89&-4.23&-7.25\\5.84&5.13&-0.89&3.55\end{bmatrix}.$
8. $f(x)=\dfrac{1}{1+\dfrac{1+x}{1-x^2}};$ $A=\begin{bmatrix}-5.84&9.84&0.23&1.59\\-9.25&-0.25&1.54&0.43\end{bmatrix}.$
9. $f(x)=\dfrac{x^3+\sin x}{x^3-\cos x}\sqrt{e^x+1};$ $A=\begin{bmatrix}0.64&6.34&0.32&-4.23\\1.19&3.23&1.54&0.43\end{bmatrix}.$
10. $f(x)=\arcsin(\cos(x^2));$ $A=\begin{bmatrix}\pi&2.2\pi&-2\pi&0.3\pi\\3\pi&-\pi&0.1\pi&5\pi\end{bmatrix}.$

Задания для самостоятельной работы

Сконструировать блочные матрицы (используя функции для заполнения стандартных матриц) и применить функции обработки данных и поэлементные операции для нахождения заданных величин.

Варианты

1. $A=\begin{bmatrix}1&0&0&4&0&0\\0&1&0&0&4&0\\0&0&1&0&0&4\\2&2&2&3&3&3\\2&2&2&3&3&3\\2&2&2&3&3&3\end{bmatrix}$ $m=\max\limits_{j=1,K6}\left\{\sum\limits_{i=1}^{6}a_{ij}^2\right\}.$

2. $A=\begin{pmatrix} 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \end{pmatrix}$ $s=\sum_{i=1}^{6}\sum_{j=1}^{6}\left|a_{ij}\right|$.

3. $A=\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ -1 & -2 & -3 & -4 & -5 & -6 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 \end{pmatrix}$ $m=\min_{i,j=1,\text{K},6} a_{ij}^3$.

4. $A=\begin{pmatrix} -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 3 & 0 & 0 \\ 2 & 2 & 2 & 0 & 3 & 0 \\ 2 & 2 & 2 & 0 & 0 & 3 \end{pmatrix}$ $s=\sum_{k=1}^{6} a_{kk}^3$.

5. $A=\begin{pmatrix} 1 & 1 & -3 & -3 & -3 & -3 \\ 1 & 1 & -3 & -3 & -3 & -3 \\ -3 & -3 & 2 & 0 & 0 & 0 \\ -3 & -3 & 0 & 2 & 0 & 0 \\ -3 & -3 & 0 & 0 & 2 & 0 \\ -3 & -3 & 0 & 0 & 0 & 2 \end{pmatrix}$ $s=\sum_{i=1}^{5} a_{i\,i+1}$.

6. $A=\begin{pmatrix} -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 & 4 \end{pmatrix}$ $s=\sum_{i=1}^{6} a_{ii}+\sum_{i=1}^{5} a_{i\,i+1}$.

7. $A=\begin{pmatrix} 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 0 & 4 \\ 3 & 0 & 0 & 0 & 0 & 4 \\ 1 & 1 & 1 & 1 & 1 & -2 \end{pmatrix}$ $s=\sum_{i=1}^{6}\sum_{j=1}^{6}\sin\left(\frac{p}{6}a_{ij}^2\right)$.

8. $A=\begin{pmatrix} 1 & 2 & 3 & -1 & 0 & 0 \\ 1 & 2 & 3 & 0 & -1 & 0 \\ 1 & 2 & 3 & 0 & 0 & -1 \\ 0 & 0 & 7 & 2 & 2 & 2 \\ 0 & 7 & 0 & 2 & 2 & 2 \\ 7 & 0 & 0 & 2 & 2 & 2 \end{pmatrix}$ $m=\max_{i=1,\text{K},6}\min_{j=1,\text{K},6} a_{ij}$.

9. $A=\begin{pmatrix} 1 & 0 & 0 & 1 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 1 & 0 & 0 & 1 & -3 & -3 \\ -2 & -2 & -2 & -2 & 4 & 4 \\ -2 & -2 & -2 & -2 & 4 & 4 \end{pmatrix}$ $s=\sum_{i=1}^{6}\max_{j=1,\text{K},6}\left(a_{ij}+a_{ji}\right)$.

10. $A = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ -1 & -1 & -1 & 4 & 0 & 0 \\ -1 & -1 & -1 & 0 & 4 & 0 \\ -1 & -1 & -1 & 0 & 0 & 4 \end{pmatrix}$ $p = \prod_{i=1}^{6} \sum_{j=1}^{6} (a_{ij})^{a_{ij}}$.

§ 4. ГРАФИКА И ВИЗУАЛИЗАЦИЯ ДАННЫХ

MatLab обладает широким набором средств для построения графиков функций одной и двух переменных и отображения различных типов данных. Все графики выводятся в графические окна со своими меню и панелями инструментов. Вид графиков определяется аргументами графических команд и затем может быть изменен при помощи инструментов графического окна. Важно понимать, что для построения графиков функций на некоторой области изменения аргументов следует вычислить значения функции в точках области, часто для получения хороших графиков следует использовать достаточно много точек.

Разберем сначала, как получить график функции одной переменной, к примеру:

$$f(x) = e^x \sin \pi x + x^2$$

на отрезке $[-2,2]$. Первый шаг состоит в задании координат точек по оси абсцисс. Заполнение вектора `x` элементами с постоянным шагом при помощи двоеточия позволяет просто решить эту задачу. Далее необходимо поэлементно вычислить значения $f(x)$ для каждого элемента вектора `x` и записать результат в вектор `f`. Для построения графика функции осталось использовать какую-либо из графических функций MatLab. Достаточно универсальной графической функцией является `plot`. В самом простом случае она вызывается с двумя входными аргументами — парой `x` и `f` (т. е. `plot` выводит зависимость элементов одного вектора от элементов другого). Последовательность команд, приведенная ниже, приводит к появлению графического окна Figure No.1 с графиком функции (рис. 4.1).

```
» x=[-2:0.05:2];
» f=exp(x).*sin(pi*x)+x.^2;
» plot(x,f)
```

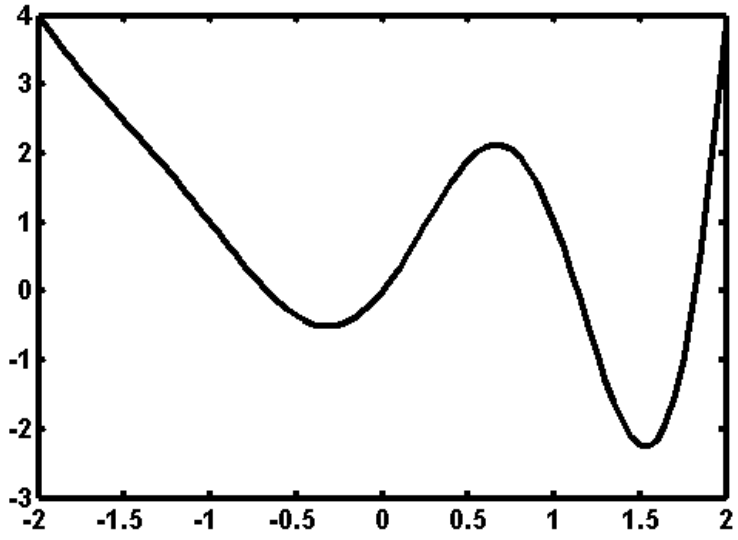

Рис. 4.1

Тип линии, цвет и маркеры определяются значением третьего дополнительного аргумента функции `plot`. Этот аргумент указывается в апострофах[3], например, вызов `plot(x,f,'ro:')` приводит к построению графика красной пунктирной линией, размеченной круглыми маркерами. Обратите внимание, что абсциссы маркеров определяются значениями элементов вектора `x`. Всего в дополнительном аргументе может быть заполнено три позиции, соответствующие цвету, типу маркеров и стилю линии. Обозначения для них приведены в табл. 4.1. Порядок позиций может быть произвольный, допустимо

[3] Он является строкой, о представлении строк и операциях с ними написано в § 7.

указывать только один или два параметра, например, цвет и тип маркеров. Посмотрите на результат выполнения следующих команд: `plot(x,f,'g')`, `plot(x,f,'ko')`, `plot(x,f,':')`.

Функция `plot` имеет достаточно универсальный интерфейс, она, в частности, позволяет отображать графики нескольких функций на одних осях. Пусть требуется вывести график не только $f(x)$, но и $g(x) = e^{-x^2} \sin 5\pi x$ на отрезке [-2, 2]. Сначала необходимо вычислить значения $g(x)$:

```
» g=exp(-x.^2).*sin(5*pi*x);
```

а затем вызвать `plot`, указав через запятую пары `x`, `f` и `x`, `g` и, при желании, свойства каждой из линий:

```
» plot(x,f,'ko-', x,g,'k:')
```

Таблица 4.1.

Сокращения для цвета, типа маркеров и стиля линий

Цвет		Тип маркера	
y	желтый	.	точка
m	розовый	o	кружок
c	голубой	x	крестик
r	красный	+	знак плюс
g	зеленый	*	звездочка
b	синий	s	квадрат
w	белый	d	ромб
k	черный	v	треугольник вершиной вниз
Тип линии		^	треугольник вершиной вверх
-	сплошная	<	треугольник вершиной влево
:	пунктирная	>	треугольник вершиной вправо
-.	штрих-пунктирная	p	пятиконечная звезда
--	штриховая	h	шестиконечная звезда

Допускается построение произвольного числа графиков функций, свойства всех линий могут быть различными. Кроме того, области построения каждой из функций не обязательно должны совпадать, но тогда следует использовать разные вектора для значений аргументов и вычислять значения функций от соответствующих векторов. Для получения графика кусочно-линейной функции:

$$y(x) = \begin{cases} \sin x & -4\pi \le x \le -\pi \\ 3(x/\pi + 1)^2 & -\pi < x \le 0 \\ 3e^{-x} & 0 < x \le 5 \end{cases}$$

достаточно выполнить последовательность команд:

```
» x1=[-4*pi:pi/10:-pi];
» y1=sin(x1);
» x2=[-pi:pi/30:0];
» y2=3*(x2/pi+1).^2;
» x3=[0:0.02:5];
» y3=3*exp(-x3)
» plot(x1,y1,x2,y2,x3,y3)
```

Заметьте, что графики ветвей функции отображаются различными цветами. Можно было поступить и по-другому, а именно: после заполнения `x1`, `y1`, `x2`, `y2`, `x3` и `y3` собрать вектор `x` для значений аргумента и вектор `y` для значений $y(x)$ и построить зависимость `y` от `x`:

```
» x=[x1 x2 x3];
» y=[y1 y2 y3];
» plot(x,y)
```

Несложно догадаться, как построить график параметрически заданной функции, используя то обстоятельство, что `plot` отображает зависимость одного вектора от другого. Пусть требуется получить график астроиды: $x(t) = \cos^3 t$, $y(t) = \sin^3 t$, $t \in [0, 2\pi]$. Следует задать вектор `t`, затем в векто-

ры x, y занести значения $x(t)$, $y(t)$ и воспользоваться plot для отображения зависимости y от x:

```
» t=[0:pi/20:2*pi];
» x=cos(t).^3;
» y=sin(t).^3;
» plot(x,y)
```

Функция comet позволяет проследить за движением точки по траектории параметрически заданной линии. Вызов comet(x,y) приводит к появлению графического окна[4], на осях которого рисуется перемещение точки в виде движения кометы с хвостом. Управление скоростью движения осуществляется изменением шага при определении вектора значений параметра.

В MatLab имеются графические функции, предназначенные для отображения графиков в логарифмическом и полулогарифмическом масштабах:

- loglog (логарифмический масштаб по обеим осям);
- semilogx (логарифмический масштаб только по оси абсцисс);
- semilogy (логарифмический масштаб только по оси ординат).

Входные аргументы этих функций задаются так же, как и при использовании plot. Для сравнения поведения двух функций со значениями разных порядков удобно применять plotyy. Функция plotyy вызывается от двух пар входных аргументов (векторов) и приводит к появлению двух линий графиков, каждой из которых отвечает своя ось ординат.

Графики оформляются в MatLab специальными командами и функциями. Сетка наносится на оси командой grid on, а убирается при помощи grid off. Заголовок размещается в графическом окне посредством функции title, входным аргументом которой является строка, заключенная в апострофы:

```
» title('Результаты эксперимента')
```

При наличии нескольких графиков требуется расположить легенду обратившись к legend. Надписи легенды, заключенные в апострофы, указываются во входных аргументах функции legend, их число должно совпадать с числом линий графиков. Кроме того, последний дополнительный входной аргумент определяет положение легенды:

–1 — вне графика в правом верхнем углу графического окна;
0 — выбирается лучшее положение в пределах графика так, чтобы как можно меньше перекрывать сами графики;
1 — в верхнем правом углу графика (это положение используется по умолчанию);
2 — в верхнем левом углу графика;
3 — в нижнем левом углу графика;
4 — в верхнем левом углу графика.

Функции xlabel и ylabel предназначены для подписей к осям, их входные аргументы так же заключаются в апострофы.

Обратимся теперь к визуализации векторных и матричных данных. Самый простой способ отображения векторных данных состоит в использовании функции plot с вектором в качестве входного аргумента. При этом получающийся в виде ломаной линии график символизирует зависимость значений элементов вектора от их индексов. Второй дополнительный аргумент может определять цвет, стиль линии и тип маркеров, например: plot(x,'ko'). Вызов функции plot от матрицы приводит к нескольким графикам, их число совпадает с числом столбцов матрицы, а каждый из них является зависимостью элементов столбца от их строчных индексов. Цвет и стиль линий и тип маркеров сразу для всех линий так же определяется вторым дополнительным аргументом.

Наглядным способом представления матричных и векторных данных являются разнообразные диаграммы. Простейшая столбцевая диаграмма строится при помощи функции bar:

```
» x=[0.7 2.1 2.5 1.9 0.8 1.3];
» bar(x)
```

Дополнительный числовой аргумент bar указывает на ширину столбцов (по умолчанию он равен 0.8), а значения большие единицы, например bar(x,1.2), приводят к частичному перекрытию столбцов. Указание матрицы во входном аргументе bar приводит к построению групповой диаграммы, число

[4] Графическое окно должно находится поверх остальных окон для наблюдения за движением с самого начала.

групп совпадает с числом строк матрицы, а внутри каждой группы столбиками отображаются значения элементов строк.

Круговые диаграммы векторных данных получаются с помощью функции `pie`, которая имеет некоторые особенности по сравнению с `bar`. Различаются два случая:

1) если сумма элементов вектора больше или равна единицы, то выводится полная круговая диаграмма, площадь каждого её сектора пропорциональна величине элемента вектора;
2) если сумма элементов вектора меньше единицы, то результатом является неполная круговая диаграмма, в которой площадь каждого сектора пропорциональна величине элементов вектора, в предположении что площадь всего круга равна единице.

Сравните, например `pie([0.1 0.2 0.3])` и `pie([1 2 3])`. Можно отделить некоторые секторы от всего круга диаграммы, для чего следует вызвать `pie` со вторым аргументом — вектором той же длины, что исходный. Ненулевые элементы второго вектора соответствуют отделяемым секторам. Следующий пример показывает, как отделить от диаграммы сектор, соответствующий наибольшему элементу вектора `x`:

```
» x=[0.3 2 1.4 0.5 0.9];
» [m,k]=max(x);
» v=zeros(size(x));
» v(k)=1;
» pie(x,v)
```

Подписи к секторам диаграммы указываются во втором дополнительном входном аргументе, который заключается в фигурные скобки[5]:

```
» pie([2400 3450 1800 5100],{'Март','Апрель','Май','Июнь'})
```

Функции `bar` и `pie` имеют аналоги:

- `barh` — построение столбцевой диаграммы с горизонтальным расположением столбцов;
- `bar3`, `pie3` — построение объемных диаграмм.

При обработке больших массивов векторных данных часто требуется получить информацию о том, какая часть данных находится в том или ином интервале. Функция `hist` предназначена для отображения гистограммы данных и нахождения числа данных в интервалах. Входным аргументом `hist` является вектор с данными, а выходным — вектор, содержащий количество элементов, попавших в каждый из интервалов. По умолчанию берется десять равных интервалов. Например, вызов `hist(randn(1,5000))` приводит к появлению на экране гистограммы данных, распределенных по нормальному закону, а `n=hist(randn(1,5000))` к заполнению вектора `n` длины десять (при этом гистограмма не строится). Число интервалов указывается во втором дополнительном аргументе `hist`. Можно задать интервалы, использовав в качестве второго аргумента не число, а вектор, содержащий центры интервалов. Более удобно задавать интервалы не центрами, а границами. В этом случае требуется сначала определить количество элементов в интервалах при помощи функции `histc`, а затем применить `bar` со специальным аргументом `'histc'`, например:

```
» x=randn(1,10000);
» int=[-2:0.5:2];
» n=histc(x,int);
» bar(int,n,'histc')
```

Визуализация функций двух переменных в MatLab может быть осуществлена несколькими способами, но все они предполагают однотипные предварительные действия. Рассмотрим здесь только построение графиков функций двух переменных на прямоугольной области определения[6]. Предположим, что требуется получить поверхность функции $z(x,y)=e^{-x}\sin(\pi y)$ на прямоугольнике $x\in[-1,1]$, $y\in[0,2]$. Первый шаг состоит в задании сетки на прямоугольнике, т. е. точек, которые будут использоваться для вычисления значений функции. Для генерации сетки предусмотрена функция `meshgrid`, вызываемая от двух входных аргументов — векторов, задающих разбиения по осям *x* и *y*. Функция `meshgid` возвращает два выходных аргумента, являющиеся матрицами.

[5] Второй аргумент является массивом ячеек, о массивах ячеек написано в § 8.

[6] MatLab поддерживает построение графиков функций двух переменных не только на прямоугольной, но и на произвольной многоугольной области определения.

```
» [X,Y]=meshgrid(-1:0.1:1,0:0.1:2);
```

Матрица `X` состоит из одинаковых строк, равных первому входному аргументу — вектору в `meshgrid`, а матрица `Y` — из одинаковых столбцов, совпадающих со вторым вектором в `meshgrid`. Такие матрицы оказываются необходимыми на втором шаге при заполнении матрицы `Z`, каждый элемент которой является значением функкции $z(x,y)$ в точках сетки. Несложно понять, что использование поэлементных операций при вычислении функции $z(x,y)$ приводит к требуемой матрице:

```
» Z=exp(-X).*sin(pi*Y);
```

Для построения графика $z(x,y)$ осталось вызвать подходящую графическую функцию, к примеру:

```
» mesh(X,Y,Z)
```

На экране появляется графическое окно, содержащее каркасную поверхность исследуемой функции (рис. 4.2). Обратите внимание, что цвет поверхности соответствует значению функции.

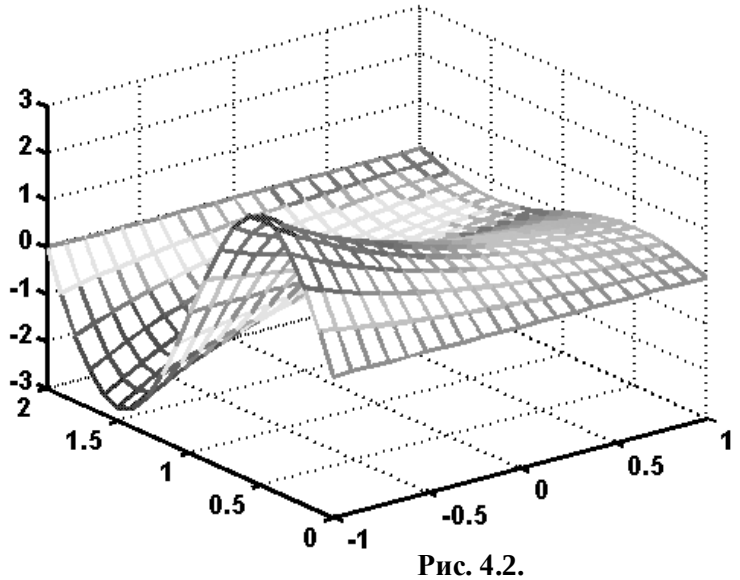

Рис. 4.2.

Команда `colorbar` приводит к отображению в графическом окне столбика, показывающего соотношение между цветом и значением $z(x,y)$. Цветовые палитры графика можно изменять, пользуясь функцией `colormap`, например `colormap(gray)` отображает график в оттенках серго цвета. Некоторые цветовые палитры приведены ниже:

- `bone` – похожа на палитру `gray`, но с легким оттенком синего цвета;
- `colorcube` – каждый цвет изменяется от темного к яркому;
- `cool` – оттенки голубого и пурпурного цветов;
- `copper` – оттенки медного цвета;
- `hot` – плавное изменение: черный-красный-оранжевый-желтый-белый;
- `hsv` – плавное изменение (как цвета радуги);
- `jet` – плавное изменение: синий-голубой-зеленый-желтый-красный;
- `spring` – оттенки пурпурного и желтого;
- `summer` – оттенки зеленого и желтого;
- `winter` – оттенки синего и зеленого;

MatLab предоставляет целый набор графических функций для визуализации функций двух переменных, среди них:

- `surf` — залитая цветом каркасная поверхность;
- `meshc`, `surfc` — поверхности с линиями уровня на плоскости *xy*;
- `contour` — плоский график с линиями уровня;
- `contourf` — залитый цветом плоский график с линиями уровня;
- `contour3` — поверхность, составленная из линий уровня;
- `surfl` — освещенная поверхность

Все перечисленные функции допускают то же самое обращение, что и `mesh`, например:

```
» sufr(X,Y,Z)
» contourf(X,Y,Z)
```

Остановимся подробнее на нескольких вопросах. Первый из них: как изменять установки, определённые по умолчанию, при отображении функций линиями уровня при помощи

contour, contourf и contour3. Число линий уровня задается в четвертом дополнительном аргументе, например:

```
» contourf(X,Y,Z,10)
```

Вместо числа линий уровня можно указать в векторе те значения $z(x,y)$, для которых требуется построить линии уровня:

```
» contour(X,Y,Z,[-0.51 -0.25 -0.01 0.89])
```

Несколько сложнее нанести подписи с соответствующим значением $z(x,y)$ к каждой линии уровня. Для этого придется вызвать contour с двумя выходными аргументами, первый из них — матрица с информацией о положении линий уровня, а второй — вектор с указателями[7] на линии. Полученные переменные следует использовать в качестве входных аргументов функции clabel:

```
» [CMatr, h] = contour(X, Y, Z,[-0.51 -0.25 -0.01 0.89]);
» clabel(CMatr, h)
```

Залитые цветом каркасные поверхности, построенные при помощи sufr и surfc, имеют постоянный цвет в пределах каждой ячейки. Команда shading interp, вызываемая после surf и surfc, служит для плавного изменения цвета в пределах ячеек и скрытия линий сетки на поверхности. Если желательно убрать сетку и сохранить постоянный цвет ячеек, то достаточно использовать shading flat, а shading faceted придает графику прежний вид.

Графические функции по умолчанию располагают поверхность так, что наблюдатель видит ее часть под некоторым углом, а другая — скрыта от взора. Положение наблюдателя определяется двумя углами: азимутом (AZ) и углом возвышения (EL). Азимут отсчитывается от оси, противоположной *y*, а угол возвышения — от плоскости *xy* (см. рис. 4.3, на котором положительные направления отсчета углов обозначены стрелками).

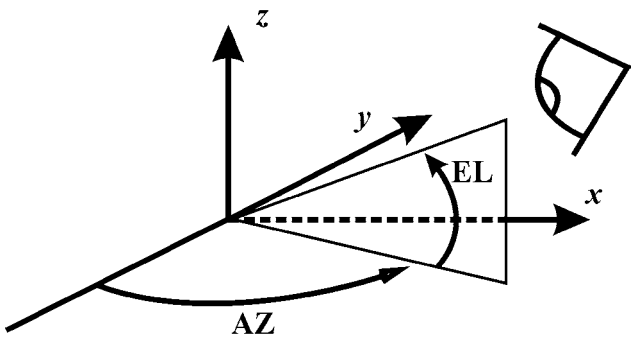

Рис. 4.3.

Осмотреть поверхность со всех сторон позволяет функция view. Вызов функции view с двумя выходными аргументами и без входных дает возможность определить текущее положение наблюдателя (углы выводятся в градусах):

```
» [AZ, EL]=view
AZ =
  -37.5000
EL =
    30
```

Эти значения MatLab использует по умолчанию при построении трехмерных графиков. Для задания положения наблюдателя следует указать азимут и угол возвышения (в градусах) в качестве входных аргументов view, например: view(0,90) показывает вид на график сверху. Перед поворотом графика целесообразно расставить обозначения к осям, используя, как и для двумерных графиков xlabel и ylabel, и zlabel для подписи к вертикальной оси. Функция view допускает еще несколько вариантов вызова:

§ view(3) — возврат к стандартным установкам;

[7] MatLab является объектно-ориентированной системой, все графические объекты (окна, оси, линии, поверхности и т. д.) выстроены в определённой иерархии и с каждым объектом связывается некоторый указатель. В данном случае, каждой линии уровня ставится в соответствие элемент вектора h.

§ `view(x,y,z)` — помещение наблюдателя в точку с координатами `x`, `y` и `z`.

Освещенная поверхность строится при помощи функции `surfl`, которая позволяет получить наглядное представление о поведении исследуемой функции. Следует учесть, что лучше сочетать вызов `surfl` с командой `shading interp` и цветовой палитрой, содержащей большое количество оттенков (`gray`, `copper`, `bone`, `winter` и т. д.), поскольку поверхность обладает свойствами рассеивания, отражения и поглощения света, исходящего от некоторого источника. Положение источника можно задавать в четвертом дополнительном аргументе `surfl`, причем либо вектором из двух элементов (азимут и угол возвышения источника), либо вектором из трех элементов (положение источника света в системе координат осей), например: `surfl(X,Y,Z,[20 80])` или `surfl(X,Y,Z,[6 8 11])`.

Разберем теперь работу с несколькими графиками. Первый вызов любой графической функции приводит к появлению на экране графического окна Figure No. 1, содержащего оси с графиком. Однако, при дальнейших обращениях к графическим функциям прежний график пропадает, а новый выводится в тоже самое окно. Команда `figure` предназначена для создания пустого графического окна. Если требуется получить несколько графиков в разных окнах, то перед вызовом графических функций следует прибегать к `figure`. Графические окна при этом нумеруются так: Figure No. 2, Figure No. 3 и т. д.

Каждое окно имеет свои оси, при наличии нескольких пар осей (в одном окне или в разных) вывод графиков производится в текущие оси. Последняя созданная пара осей является текущей. Для того, чтобы выбрать текущие оси из нескольких имеющихся, достаточно щелкнуть по ним правой кнопкой мыши перед вызовом графической функции. Возможна и обратная ситуация, когда в процессе работы требуется добавлять графики к уже имеющимся на некоторых осях. В этой ситуации перед добавлением графика следует выполнить команду `hold on`. Для завершения такого режима достаточно воспользоваться `hold off`.

В одном графическом окне можно расположить несколько осей со своими графиками. Функция `subplot` предназначена для разбиения окна на части и определения текущей из них. Предположим, что требуется вывести графики на шесть пар осей в одно графическое окно (две по вертикали и три по горизонтали). Создайте графическое окно при помощи `figure` и выполните команду:

```
» subplot(2,3,1)
```

В левом верхнем углу окна появились оси. Первые два аргумента в `subplot` указывают на общее число пар осей по вертикали и горизонтали, а последний аргумент означает номер данной пары осей. Нумерация идет слева направо, сверху вниз. Используйте `subplot(2,3,2)`, … , `subplot(2,3,6)` для создания остальных пар осей[8]. Вывод любой из графических функций можно направить в нужные оси, указав их при помощи `subplot(2,3,k)`, например:

```
» subplot(2,3,3)
» bar([1.2 0.3 2.8 0.9])
» subplot(2,3,6)
» surf(X,Y,Z)
```

Задания для самостоятельной работы

Построить графики функций одной переменной на указанных интервалах. Вывести графики различными способами:

§ в отдельные графические окна;
§ в одно окно на одни оси;
§ в одно окно на отдельные оси.

Дать заголовки, разместить подписи к осям, легенду, использовать различные цвета, стили линий и типы маркеров, нанести сетку.

Варианты

1. $f(x)=\sin x;$ $g(x)=\sin^2 x;$ $x\hat{I}\ [-2p,3p].$

   $u(x)=0.01x^2;$ $v(x)=e^{-|x|};$ $x\hat{I}\ [-0.2,9.4].$

2. $f(x)=\sin x^2;$ $g(x)=\cos x^2;$ $x\hat{I}\ [-p,p].$

[8] Предварительное создание всех пар осей не является обязательным. Допустимо применять `subplot` непосредственно перед вызовом графической функции.

	$u(x)=x/20;$	$v(x)=e^{x};$	$x\in[-2,2].$
3.	$f(x)=x^3+2x^2+1;$	$g(x)=(x-1)^4;$	$x\in[-1,1].$
	$u(x)=\sqrt{x};$	$v(x)=e^{-x^2};$	$x\in[0,1].$
4.	$f(x)=\ln x;$	$g(x)=x\ln x;$	$x\in[0.2,10].$
	$u(x)=x^{1/3};$	$v(x)=\sqrt{x};$	$x\in[0,8].$
5.	$f(x)=\lvert 2x\rvert^3;$	$g(x)=\lvert 2x\rvert^5;$	$x\in[-0.5,0.5].$
	$u(x)=\sqrt{\lvert x\rvert};$	$v(x)=x^{1/5};$	$x\in[-0.6,0.5].$
6.	$f(x)=x^2;$	$g(x)=x^3;$	$x\in[-1,1].$
	$u(x)=x^4;$	$v(x)=x^5;$	$x\in[-1,1].$
7.	$f(x)=\arcsin x;$	$g(x)=\arccos(x);$	$x\in[-1,1].$
	$u(x)=\operatorname{arctg} x;$	$v(x)=\operatorname{arctg} 3x;$	$x\in[-1,1].$
8.	$f(x)=\operatorname{sh} x;$	$g(x)=\operatorname{ch} x;$	$x\in[-1,1].$
	$u(x)=e^{x};$	$v(x)=e^{-x};$	$x\in[-0.6,0.6].$
9.	$f(x)=\dfrac{\sin x}{x};$	$g(x)=e^{-x}\cos x;$	$x\in[0.01,2\pi].$
	$u(x)=\sin(\ln(x+1));$	$v(x)=\cos(\ln(x+1));$	$x\in[0,2\pi].$
10.	$f(x)=x^x;$	$g(x)=x^{x^x};$	$x\in[0.1,1].$
	$u(x)=\dfrac{1}{1+x};$	$v(x)=\dfrac{1}{1+\frac{1}{1+x}};$	$x\in[0,1].$

Задания для самостоятельной работы

Построить график кусочно-линейной функции, отобразить ветви разными цветами и маркерами.

1. $f(x)=\begin{cases}-1, & -3\le x\le -1\\ x, & -1<x\le 1\\ e^{1-x}, & 1<x\le 3\end{cases}$

2. $f(x)=\begin{cases}\sqrt{x}, & 0\le x\le 1\\ x, & 1<x\le 3\\ (x-4)^2, & 3<x\le 5\end{cases}$

3. $f(x)=\begin{cases}\ln x, & 1\le x\le e\\ x/e, & e<x\le 9\\ e^{8-x}, & 9<x\le 12\end{cases}$

4. $f(x)=\begin{cases}\sin x, & -2\pi\le x\le 0\\ -x^3, & 0<x\le 1\\ \cos\pi x, & 1<x\le 3\pi\end{cases}$

5. $f(x)=\begin{cases}\arcsin x-1, & 0\le x\le 1\\ \frac{\pi}{2}-x, & 1<x\le\frac{\pi}{2}\\ \cos x, & \frac{\pi}{2}<x\le\pi\end{cases}$

6. $f(x)=\begin{cases}\lvert x\rvert, & -2\le x\le 1\\ \sin\frac{\pi}{2}x, & 1<x\le 2\\ (2-x)^3, & 2<x\le 3\end{cases}$

7. $f(x)=\begin{cases}(x-1)^2, & -2\le x\le 1\\ \cos\frac{\pi}{2}x, & 1<x\le 3\\ 1-e^{3-x}, & 3<x\le 8\end{cases}$

8. $f(x)=\begin{cases}e^{x}, & -2\le x\le -1\\ \frac{|x|}{e}, & -1<x\le 1\\ e^{-x}, & 1<x\le 2\end{cases}$

9. $f(x)=\begin{cases} e^{x+1}, & -2 \le x \le -1 \\ x^2, & -1 < x \le 1 \\ (2-x)^3, & 1 < x \le 2 \end{cases}$

10. $f(x)=\begin{cases} x^2 \log_2 x, & 1 \le x \le 2 \\ x^3/2, & 2 < x \le 3 \\ x^x/2, & 3 < x \le 3.5 \end{cases}$

Задания для самостоятельной работы

Построить график параметрически заданной функции, используя `plot` и `comet`.

Варианты

1. $x(t)=t-\sin t;\ y(t)=1-\cos t.$
2. $x(t)=2\sin t-\frac{2}{3}\sin 2t;\ y(t)=2\cos t-\frac{2}{3}\cos 2t.$
3. $x(t)=9\sin\frac{t}{10}-\frac{1}{2}\sin\frac{9}{10}t;\ y(t)=9\cos\frac{1}{10}t+\frac{1}{2}\cos\frac{9}{10}t.$
4. $x(t)=\cos t;\ y(t)=\sin(\sin t).$
5. $x(t)=e^{-t}\cos t;\ y(t)=\sin t.$
6. $x(t)=e^{-t}\cos t;\ y(t)=e^{t}\sin t.$
7. $x(t)=t(t-2\pi);\ y(t)=\sin t.$
8. $x(t)=\sin t(t-2\pi);\ y(t)=\sin t.$
9. $x(t)=\sin t(t-2\pi);\ y(t)=\sin t\times\cos t.$
10. $x(t)=\sin t+\cos^3 t;\ y(t)=\sin t\times\cos t.$

Задания для самостоятельной работы

Визуализировать функцию двух переменных на прямоугольной области определения различными способами:

- каркасной поверхностью;
- залитой цветом каркасной поверхностью;
- промаркированными линиями уровня (самостоятельно выбрать значения функции, отображаемые линиями уровня);
- освещенной поверхностью.

Расположить графики в отдельных графических окнах и в одном окне с соответствующим числом пар осей. Представить вид каркасной или освещенной поверхности с нескольких точек обзора.

Варианты

1. $z(x,y)=\sin x\times e^{-3y}$ $x\in[0,2\pi]$ $y\in[0,1]$
2. $z(x,y)=\sin^2 x\times\ln y$ $x\in[0,2\pi]$ $y\in[0,10]$
3. $z(x,y)=\sin^2(x-2y)\times e^{-|y|}$ $x\in[0,\pi]$ $y\in[-1,1]$
4. $z(x,y)=\dfrac{x^2y^2+2xy-3}{x^2+y^2+1}$ $x\in[-2,2]$ $y\in[-1,1]$
5. $z(x,y)=\dfrac{\sin xy}{x}$ $x\in[-0.1,5]$ $y\in[-\pi,\pi]$
6. $z(x,y)=\left(\sin x^2+\cos y^2\right)^{xy}$ $x\in[-1,1]$ $y\in[-1,1]$
7. $z(x,y)=\arctan(x+y)\left(\arccos x+\arcsin y\right)$ $x\in[-1,1]$ $y\in[-1,1]$
8. $z(x,y)=(1+xy)(3-x)(4-y)$ $x\in[0,3]$ $y\in[0,4]$
9. $z(x,y)=e^{-|x|}\left(x^5+y^4\right)\sin(xy)$ $x\in[-2,2]$ $y\in[-3,3]$
10. $z(x,y)=(y^2-3)\sin\dfrac{x}{|y|+1}$ $x\in[-2\pi,2\pi]$ $y\in[-3,3]$

§ 5. ФАЙЛ-ФУНКЦИИ И ФАЙЛ-ПРОГРАММЫ

Встроенный язык программирования MatLab достаточно прост, он содержит необходимый минимум конструкций, которые описаны в следующем параграфе. Прежде чем программировать в MatLab, необходимо понять, что все программы могут быть либо файл-функциями, либо файл-программами. Файл-программа является текстовым файлом с расширением m (М-файлом), в котором записаны команды и операторы MatLab. Разберем, как создать простую файл-программу.

В MatLab имеется редактор М-файлов, для запуска которого следует нажать кнопку New M-file на панели инструментов рабочей среды, либо выбрать в меню File в пункте New подпункт M-file. На экране появляется окно редактора. Наберите в нем какие-либо команды, например для построения графика (см. листинг 5.1):

Листинг 5.1. Простейшая файл-программа

```
x=[-1:0.01:1];
y=exp(x);
plot(x,y)
grid on
title('Экспоненциальная функция')
```

Для запуска программы или ее части есть несколько способов. Первый, самый простой — выделить операторы при помощи мыши, удерживая левую кнопку, или при помощи клавишы[9] <Shift> со стрелками, <PageUp>, <PageDown> и выбрать в меню View пункт Evaluate Selection (или нажать <F9>). Выделенные операторы выполняются последовательно, точно так же, как если бы они были набраны в командной строке. Очевидно, что работать в М-файле удобнее, чем из командной строки, поскольку можно сохранить программу, добавить операторы, выполнять отдельные команды не пробегаясь по истории команд, как в случае командной строки.

После того, как программа сохранена в М-файле, к примеру в myprog.m, для ее запуска можно использовать пункт Tools меню Run, либо просто набрать в командной строке имя М-файла (без расширения) и нажать <Enter>, то есть выполнить, как обычную команду MatLab. При таких способах запуска программы следует учесть важное обстоятельство — путь к каталогу с М-файлом должен быть известен MatLab. Сделайте каталог с файлом `myprog` текущим.

§ В MatLab 5.3 в меню File рабочей среды перейдите к пункту Set Path… Появляется диалоговое окно Path Browser (навигатор путей). В строке ввода Current Directory установите требуемый каталог. Воспользуйтесь кнопкой, расположенной справа от строки ввода, для выбора каталога.

§ В MatLab 6.x установка текущего каталога производится из окна Current Directory рабочей среды. Если это окно отсутствует, то следует выбрать пункт Current Directory меню View рабочей среды. Для выбора желаемого каталога на диске нажмите кнопку, расположенную справа от раскрывающегося списка.

Когда текущий каталог установлен, то все М-файлы, находящиеся в нем, могут быть запущены из командной строки, либо из редактора М-файлов. Все переменные файл-программы после ее запуска доступны в рабочей среде, т. е. являются глобальными. Убедитесь в этом, выполнив команду `whos`. Более того, файл-программа может использовать переменные рабочей среды. Например, если была введена команда:

```
» a=[0.1 0.4 0.3 1.9 3.3];
```

то файл-программа, содержащая строку `bar(a)`, построит столбцевую диаграмму вектора `a` (разумеется, если он не был переопределен в самой файл-программе).

Файл-функции отличаются от файл-программ тем, что они могут иметь входные и выходные аргументы, а все переменные, определенные внутри файл-функции, являются локальными и не видны в рабочей среде. М-файл, содержащий файл-функцию, должен начинаться с заголовка, после него записываются операторы MatLab. Заголовок состоит из слова `function`, списка выходных аргументов, имени файл-функции и списка входных аргументов. Аргументы в списках разделяются запятой. Листинг 5.2 содержит пример простейшей файл-функции с двумя входными и одним выходным аргументами.

Листинг 5.2. Файл-функция mysum

```
function c=mysum(a,b)
```

[9] Так, как выделяется блок текста в текстовом редакторе.

```
c=a+b;
```
Наберите этот пример в новом файле в редакторе и сохраните его. Обратите внимание, что MatLab предлагает в качестве имени М-файла название файл-функции, т.е. mysum.m. Всегда сохраняйте файл-функцию в М-файле, имя которого совпадает с именем файл-функции! Убедитесь, что каталог с файлом mysum.m является текущим и вызовите файл-функцию `mysum` из командной строки:
```
» s=mysum(2,3)
s =
     5
```
При вызове файл-функции `mysum` произошли следующие события:

- входной аргумент `a` получил значение 2;
- входной аргумент `b` стал равен 3;
- сумма `a` и `b` записалась в выходной аргумент `c`;
- значение выходного аргумента `c` получила переменная `s` рабочей среды и результат вывелся в командное окно.

Заметьте, что оператор `c=a+b` в файл-функции `mysum` завершен точкой с запятой для подавления вывода локальной переменной `c` в командное окно. Для просмотра значений локальных переменных при отладке файл-функций, очевидно, не следует подавлять вывод на экран значений требуемых переменных.

Практически все функции MatLab являются файл-функциями и хранятся в одноименных М-файлах. Функция `sin` допускает два варианта вызова: `sin(x)` и `y=sin(x)`, в первом случае результат записывается в `ans`, а во втором — в переменную `y`. Наша функция `mysum` ведет себя точно так же. Более того, входными аргументами `mysum` могут быть массивы одинаковых размеров или массив и число.

Разберем теперь, как создать файл-функцию с несколькими выходными аргументами. Список выходных аргументов в заголовке файл-функции заключается в квадратные скобки, сами аргументы отделяются запятой. В качестве примера на листинге 5.3 приведена файл-функция `quadec`, которая по заданным коэффициентам квадратного уравнения находит его корни.

Листинг 5.3. Файл-функция для решения квадратного уравнения
```
function [x1,x2]=quadeq(a,b,c)
D=b^2-4*a*c;
x1=(b+sqrt(D))/(2*a);
x2=(b-sqrt(D))/(2*a);
```
При вызове `quadeq` из командной строки используйте квадратные скобки для указания переменных, в которые будут занесены значения корней:
```
» [r1,r2]=quadeq(1,3,2)
r1 =
     2
r2 =
     1
```
Заметьте, что файл-функцию `quadec` можно вызвать без выходных аргументов, или только с одним выходным аргументом. В этом случае вернется только первый корень.

Файл-функция может и не иметь входных или выходных аргументов, заголовки таких файл-функций приведены ниже:

`function noout(a,b)`, `function [v,u]=noin`, `function noarg()`

Умение писать собственные файл-функции и файл-программы необходимо как при программировании в MatLab, так и при решении различных задач средствами MatLab (в частности, поиска корней уравнений, интегрирования, оптимизации[10]). Разберем только один пример, связанный с построением графика функции $f(x) = e^{-x}\left(\sin x + 0.1\sin(100\pi x)\right)$ на отрезке $[0,1]$. Запрограммируйте файл-функцию `myfun` для вычисления $f(x)$. Используйте поэлементные операции (см. листинг 5.4) для того, чтобы `myfun` можно было вызывать от вектора значений аргумента и получать вектор соответствующих значений функции.

Листинг 5.4. Файл-функция myfun
```
function y=myfun(x);
```

[10] Использование численных методов MatLab выходит за рамки этого пособия.

```
y=exp(-x).*(sin(x)+0.1*sin(100*pi*x));
```

График $f(x)$ можно получить двумя способами. Первый очевидный — надо создать вектор значений аргумента, скажем с шагом 0.01, заполнить вектор значений функции и вызвать `plot`:

```
» x=[0:0.01:1];
» y=myfun(x);
» plot(x,y)
```

В результате получается график, приведенный на рис. 5.1, а, который, очевидно, неверен. Действительно, при вычислении значений функции на отрезке $[0,1]$ с шагом 0.01 слагаемое $\sin(100\pi x)$ все время обращалось в ноль и `plot` построила график не $f(x)$, а другой функции. Непродуманный выбор шага часто приводит к потере существенной информации о поведении функции. В MatLab имеется встроенная функция `fplot` — некоторый аналог `plot`, но с автоматическим подбором шага при построении графика. Первым входным аргументом `fplot` является имя файл-функции, а вторым — вектор, элементы которого есть границы отрезков: `fplot('имя файл-функции', [a,b])`. Постройте теперь в новом окне график $f(x)$ при помощи `fplot`:

```
» figure
» fplot('myfun',[0,1])
```

Получился график, точно отражающий поведение функции (рис. 5.1, б).

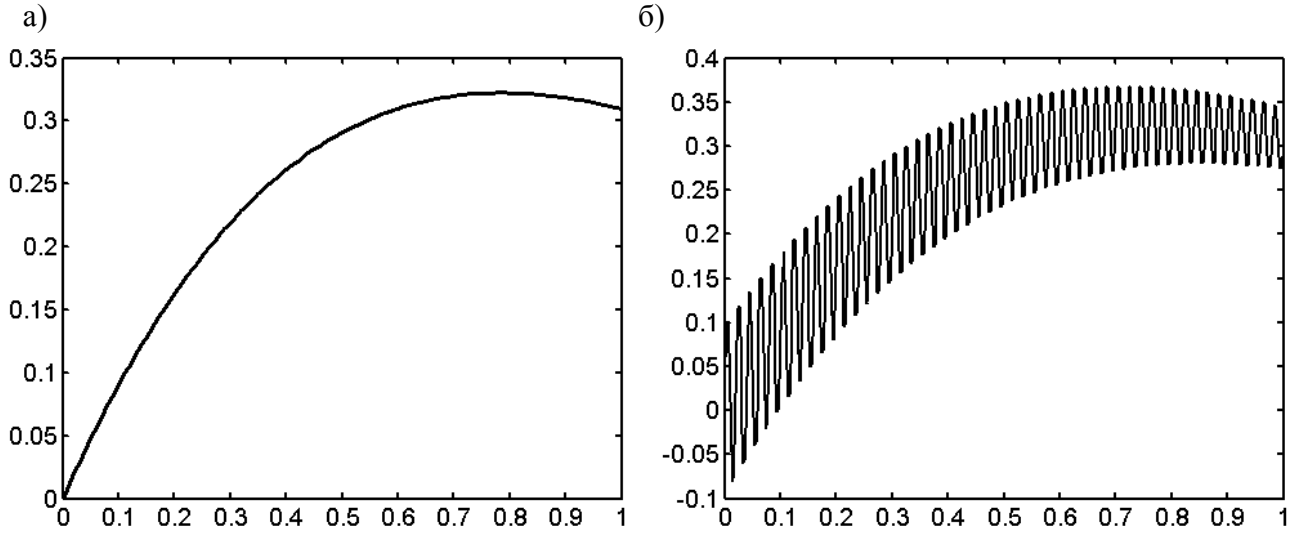

Рис. 5.1.

Задания для самостоятельной работы

Написать файл-функции и построить графики на заданном отрезке при помощи `plot` (с шагом 0.05) и `fplot` для следующих функций:

Варианты

1. $f(x)=\sin\frac{1}{x}$ $x\in[0.05,1]$.
2. $f(x)=e^{3x\sin 5\pi x}+e^{3x\cos 5\pi x}$ $x\in[0,1]$.
3. $f(x)=\dfrac{10}{11-10\sin 21\pi x}$ $x\in[0.05,1]$.
4. $f(x)=\sqrt{\dfrac{|\sin 21\pi x|}{2+\sin 20\pi x}}$ $x\in[0,1]$.
5. $f(x)=\dfrac{1}{\operatorname{arctg}\left(\dfrac{1}{1.1+\sin 5\pi x}\right)-\dfrac{3}{2}}$ $x\in[0,1]$.
6. $f(x)=\cos\left(\dfrac{1}{\frac{2\pi}{11}-\operatorname{arctg}x^x}\right)$ $x\in[0,1]$.
7. $f(x)=\sin\left(6\pi\left|x-\frac{2}{3}\right|x^3\right)$ $x\in[0,1]$.
8. $f(x)=\sin 2\pi\sqrt{\left|\sqrt{1-x^3}-\frac{4}{7}\right|}$ $x\in[0,1]$.
9. $f(x)=|\sin 20\pi x|$ $x\in[0.05,1]$.

10. $f(x) = \dfrac{1}{\sin(e^{2x} - e^{-2x}) + \cos(e^{2x} - e^{-2x}) - \frac{3}{2}}$ $x \in [-1,1]$.

Задания для самостоятельной работы

Написать файл-функцию для решения поставленной задачи.

Варианты

1. Написать файл-функцию, которая по заданному вектору определяет номер его элемента с наибольшим отклонением от среднего арифметического всех элементов вектора.
2. Написать файл-функцию, возвращающую сумму всех элементов вектора с нечетными индексами.
3. Написать файл-функцию, вычисляющую максимальное значение среди диагональных элементов заданной матрицы.
4. Написать файл-функцию, переставляющую первый столбец квадратной матрицы с ее диагональю.
5. Написать файл-функцию, которая суммирует все внедиагональные элементы заданной матрицы.
6. Написать файл-функцию, заменяющую максимальный элемент вектора средним значением всех его элементов.
7. Написать файл-функцию, заменяющую элемент матрицы с индексами 1,1 произведением всех элементов матрицы.
8. Написать файл-функцию, которая строит многоугольник (замкнутый) по заданным векторам `x` и `y` с координатами вершин.
9. Написать файл-функцию, которая отображает элементы заданного вектора синими маркерами, а максимальный элемент — красным и возвращает значение и номер максимального элемента.
10. Написать файл-функцию, переводящую время в секундах в часы, минуты и секунды.

§ 6. ПРОГРАММИРОВАНИЕ

Язык программирования MatLab достаточное простой, он содержит основной набор конструкций: операторы ветвления и циклы. Простота языка программирования окупается огромным количеством встроенных функций, которые позволяют решать задачи из различных областей.

Цикл `for` используется для повторения операторов в случае, когда число повторений заранее известно. В цикле `for` используется счетчик цикла, его начальное значение, шаг и конечное значение указываются через двоеточие. Блок операторов, размещенный внутри цикла, должен заканчиваться словом `end`. Листинг 6.1 содержит файл-программу для вывода графиков функции $f(x, b) = e^{bx} \sin x$ на отрезке $[-2,2]$, для значений параметра $b \in [-0.5,0.5]$.

Листинг 6.1. Графики функции при различных значениях параметра

```
x=[-2:0.01:2];
for beta=-0.5:0.1:0.5
   y=exp(beta*x).*sin(x);
   plot(x,y)
   hold on
end
hold off
```

Если шаг равен единице, то его указывать не обязательно. Например, для вычисления суммы

$$\sum_{k=1}^{10} \frac{x^k}{k!}$$

при различных значениях *x* потребуется файл-функция, текст которой приведен на листинге 6.2. Обратите внимание, что `sum10` может быть вызвана как от числа, так и от массива значений, благодаря применению поэлементных операций.

Листинг 6.2. Файл-функция для вычисления суммы

```
function s=sum10(x)
s=0;
for k=1:10
   s=s+x.^k/factorial(k);
end
```

Подумайте над тем, как избежать нахождения факториала в каждом слагаемом (при вычислении *k*-го слагаемого можно использовать значение $(k-1)!$, найденное на предыдущем шаге цикла).

Цикл `for` подходит для повторения заданного числа определенных действий. В том случае, когда число повторов заранее неизвестно и определяется в ходе выполнения блока операторов следует организовать цикл `while`. Цикл `while` работает, пока выполнено условие цикла. Файл-функция `negsum` (см. листинг 6.3) находит сумму всех первых отрицательных элементов вектора.

Листинг 6.3. Файл-функция negsum

```
function s=negsum(x)
s=0;
k=1;
while x(k)<0
   s=s+x(k);
   k=k+1;
end
```

В качестве операторов отношения используются символы: `>`, `<` , `>=`, `<=`, `==` (равно), `~=` (не равно). Файл-функция `negsum` имеет один недостаток: если все элементы массива — отрицательные числа, то `k` становится больше длины массива `x`, что приводит к ошибке, например:

```
» b=[-2 -7 -1 -9 -2 -5 -4];
» s=negsum(b)
???  Index exceeds matrix dimensions.
```

Кроме проверки значения `x(k)` следует позаботится о том, чтобы `k` не превосходила длины вектора `x`. Вход в цикл должен осуществляться только при одновременном выполнении условий `k<=length(x)` и `x(k)<0`, т. е. необходимо применить логический оператор "и", обозначаемый в MatLab символом `&`. Замените условие цикла на составное: `k<=length(x) & x(k)<0`. Если первое из условий не выполняется, то второе условие проверяется не будет, именно поэтому выбран такой порядок операндов. Теперь файл-функция `negsum` работает верно для любых векторов.

Логический оператор "или" обозначается символом вертикальной черты `|`, а отрицание — при помощи тильды `~`. Если требуется применить отрицание к некоторому логическому выражению, то его следует заключить в круглые скобки, например: `~(a<1 & b==3)`. В отличие от многих языков программирования, приоритет у логического "или" такой же, как и у логического "и", например, выражения `a==4 | x>-1 & x<2` и `a==4 | (x>-1 & x<2)` неэквивалентны. Для изменения порядка выполнения логических операторов используются круглые скобки.

Циклы могут быть вложены друг в друга. Например, для поиска суммы элементов матрицы, расположенных выше главной диагонали следует использовать два цикла `for`, причем начальное значение счетчика внутреннего цикла зависит от текущего значения счетчика внешнего цикла (см. листинг 6.4).

Листинг 6.4. Использование вложенных циклов

```
function s=upsum(A)
[n m]=size(A);
s=0;
for i=1:n
   for j=i+1:m
      s=s+A(i,j);
   end
end
```

Ветвление в ходе работы программы осуществляется при помощи конструкции `if-elseif-else`. Самый простой вариант ее использования (без `elseif` и `else`) реализован в файл-функции `possum` (см. листинг 6.5), которая предназначена для нахождения суммы всех положительных элементов вектора.

Листинг 6.5. Файл-функция для суммирования положительных элементов вектора

```
function s=apossum(x)
```

```
s=0;
for k=1:length(x)
   if x(k)>=0
      s=s+x(k);
   end
end
```

Если ход программы должен изменяться в зависимости от нескольких условий, то следует использовать полную конструкцию `if-elseif-else`. Каждая из ветвей[11] `elseif` в этом случае должна содержать условие выполнения блока операторов, размещенных после нее. Важно понимать, что условия проверяются подряд, первое выполненное условие приводит к работе соответствующего блока, выходу из конструкции `if-elseif-else` и переходу к оператору, следующему за `end`. У последней ветви `else` не должно быть никакого условия. Операторы, находящиеся между `else` и `end`, работают в том случае, если все условия оказались невыполненными. Предположим, что требуется написать файл-функцию для вычисления кусочно-заданной функции:

$$f(x)=\begin{cases}1-e^{-1-x}, & x<-1\\ x^2-x-2, & -1\le x\le 2\\ 2-x, & x>2\end{cases}$$

Первое условие `x<-1` проверяется в ветви `if`. Обратите внимание, что условие `–1<=x` не требуется включать в следующую ветвь `elseif` (см. листинг 6.6), поскольку в эту ветвь программа заходит, если предыдущее условие (`x<-1`) оказалось не выполнено. Условие `x>2` проверять не надо — если не выполнены два предыдущих условия, то `x` будет больше двух.

Листинг 6.6. Файл-функция для вычисления кусочно-заданной функции

```
function f=pwf(x)
if x<-1
   f=1-exp(-1-x);
elseif x<=2
   f=x.^2-x-2;
else
   f=2-x;
end
```

Ход работы программы может определятся значением некоторой переменной (переключателя). Такой альтернативный способ ветвления программы основан на использовании оператора переключения `switch`. Переменная-переключатель помещается после `switch` через пробел. Оператор `switch` содержит блоки, начинающиеся со слова `case`, после каждого `case` записывается через пробел то значение переключателя, при котором выполняется данный блок. Последний блок начинается со слова `otherwise`, его операторы работают в том случае, когда ни один из блоков `case` не был выполнен. Если хотя бы один из блоков `case` выполнен, то происходит выход из оператора `switch` и переход к оператору, следующему за `end`.

Предположим, что требуется найти количество единиц и минус единиц в заданном массиве и, кроме того, найти сумму всех элементов, отличных единицы и минус единицы. Следует перебрать все элементы массива в цикле, а в роли переменной-переключателя выступает текущий элемент массива. Листинг 6.7 содержит файл-функцию, которая по заданному массиву возвращает число минус единиц в первом выходном аргументе, число единиц — во втором, а сумму — в третьем.

Листинг 6.7. Файл-функция mpsum

```
function [m,p,s]=mpsum(x)
m=0;
p=0;
s=0;
for i=1:length(x)
   switch x(i)
   case -1
```

[11] Ветвей elseif может быть сколько угодно.

```
      m=m+1;
   case 1
      p=p+1;
   otherwise
      s=s+x(i);
   end
end
```

Блок `case` может быть выполнен не только при одном определенном значении переключателя, но и в том случае, когда переключатель принимает одно из нескольких допустимых значений. В этом случае значения указываются после слова `case` в фигурных скобках через запятую, например: `case {1,2,3}`.

Досрочное завершение цикла `while` или `for` осуществляется при помощи оператора `break`. Пусть, например, требуется по заданному массиву `x` образовать новый массив `y` по правилу `y(k)=x(k+1)/x(k)` до первого нулевого элемента `x(k)`, т.е. до тех пор, пока имеет смысл операция деления. Номер первого нулевого элемента в массиве `x` заранее неизвестен, более того, в массиве `x` может и не быть нулей. Решение задачи состоит в последовательном вычислении элементов массива `y` и прекращении вычислений при обнаружении нулевого элемента в `x`. Файл-функция, приведенная на листинге 6.8, демонстрирует работу оператора `break`.

Листинг 6.8. Использование оператора break для выхода из цикла

```
function y=div(x)
for k=1:length(x)-1
   if x(k)==0
      break
   end
   y(k)=x(k+1)/x(k);
end
```

Задания для самостоятельной работы

Написать файл-функцию для вычисления кусочно-заданной функции (см. варианты на стр. 35). Написать файл-функцию для решения поставленной задачи.

Варианты

1. Вычислить произведение элементов вектора, не превосходящих среднее арифметическое значений его элементов.
2. Подсчитать число нулей и единиц в заданной матрице.
3. Определить количество положительных элементов вектора, расположенных между его максимальным и минимальным элементами.
4. Просуммировать отрицательные элементы матрицы, лежащие ниже главной диагонали.
5. Заменить положительные элементы вектора суммой всех его отрицательных элементов.
6. Заполнить квадратную матрицу *A*, каждый элемент которой a_{ij} определяется следующим образом:

$$a_{ij} = \begin{cases} i - j, & i > j \\ i + j, & i = j \\ i^2 + j^2, & i < j \end{cases}.$$

7. Вычислить сумму:

$$s(x) = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{x^{i+j}}{(i+j)^2}.$$

8. Для матрицы $A = (a_{ij})$ размера *n* на *m* найти значение выражения:

$$w(x) = \sum_{i=1}^{n} \prod_{j=1}^{m} a_{ij} .$$

9. По заданному x найти максимальное значение n, для которого следующая сумма не превосходит 100:

$$s(x) = \sum_{k=1}^{n} kx^k .$$

10. Вычислить сумму

$$s(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} .$$

с заданной точностью e. Суммировать следует пока отношение текущего слагаемого к уже накопленной части суммы превосходит e. Сравнить результат с точным значением, построив графики e^x и $s(x)$ для $x \in [0,5]$.

11. Заданы окружности, координаты их центров содержатся в массивах `x` и `y`, а радиусы в массиве `r`. Известны координаты некоторой точки. Требуется вывести график, на котором маркером отмечено положение точки, синим цветом изображены те окружности, внутри которых лежит точка, а остальные окружности нарисованы красным цветом.

§ 7. РАБОТА СО СТРОКАМИ

Основным типом данных в MatLab является массив, вспомните, что даже числа представляются массивами размера один на один. Элементы массива не обязательно должны быть числами. Если элементы массива являются символами, то такой массив называется массивом символов, или строкой. Переменные, содержащие строки, будем называть строковыми переменными. Для ограничения строки используются апострофы, например, оператор присваивания

```
» str='Hello, World!'
```

приводит к образованию строковой переменной

```
str =
Hello, World!
```

Убедитесь, что строковая переменная `str` действительно является массивом, обратившись к ее элементам: `str(8)`, `str(1:5)`. Команда `whos` позволяет получить подробную информацию о `str`, в частности, `str` хранится в виде
вектор-строки, ее длина равна 13. Поскольку строковые переменные являются массивами, то к ним применимы некоторые функции и операции, рассмотренные нами ранее. Длина строковой переменной, т.е. число символов в ней, находится при помощи функции `length`. Допустимо сцепление строк как вектор-строк с использованием квадратных скобок. Создайте еще одну строковую переменную `str1`, содержащую текст ' My name is Igor.', и осуществите сцепление:

```
» strnew=[str str1]
strnew =
Hello, World! My name is Igor.
```

Обратите внимание, что для разделения сцепляемых строк в квадратных скобках следует использовать пробел (или запятую). Применение точки с запятой приведет к ошибке, поскольку точка с запятой служит для образования вектор-столбцов или отделения строк матриц. Но в матрице все строки должны быть одинаковой длины, а переменные `str` и `str1` состоят из различного числа символов. Далее рассмотрим, как создавать двумерные массивы символов (массивы строк), а пока обратимся к наиболее важным функциям MatLab, предназначенным для обработки строк.

Сцепление строк может быть проведено как с использованием квадратных скобок, так и при помощи функции `strcat`. Входными аргументами `strcat` являются сцепляемые строки, их число неограничено, а результат возвращается в выходном аргументе. Функция `strcat` игнорирует

пробелы в конце каждой строки. Для строк: `s1='abc  '` и `s2='def'` сцепления `s=[s1 s2]` и `s=strcat(s1,s2)` приведут к разным результатам.

Поиск позиций вхождения подстроки в строку производится при помощи функции `findstr`. Ее входными аргументами являются строка и подстрока, а выходным — вектор позиций, начиная с которых подстрока входит в строку, например:

```
» s='abcbcddefbcc';
» s1='bc';
» p=findstr(s,s1)
p =
     2     4    10
```

Порядок входных аргументов не важен, подстрокой всегда считается аргумент меньшего размера.

Функция `strcmp` предназначена для сравнения двух строк, которые указываются во входных аргументах: `strcmp(s1,s2)`. Результатом является логическая единица или ноль. Функция `strncmp` сравнивает только часть строк от первого символа до символа, номер которого указан в третьем входном аргументе: `strncmp(s1,s2,8)`. Данные функции имеют аналоги: `strcmpi` и `strncmpi`, которые проводят сравнение без учета регистра, т.е., например, символы `'A'` и `'a'` считаются одинаковыми.

Для замены в строке одной подстроки на другую служит функция `strrep`. Пусть, например, требуется заменить в строке `str` подстроку `s1` на `s2` и записать обновленную строку в `strnew`. Тогда вызов функции `strrep` должен выглядеть так: `strnew=strrep(str,s1,s2)`. Здесь важен порядок аргументов.

Преобразование всех букв строки в строчные (прописные) производит функция `lower (upper)`, например: `lower('MatLab')` приводит к `'matlab'`, а `upper('MatLab')` — к `'MATLAB'`.

То, что строки являются массивами символов, позволяет легко писать собственные файл-функции обработки строк. Предположим, что требуется переставить символы в строке в обратном порядке. Файл-функция, приведенная на листинге 7.1, решает поставленную задачу.

Листинг 7.1. Файл-функция для перестановки символов в строке

```
function sout=strinv(s)
L=length(s);
for k=1:L
   sout(L-k+1)=s(k);
end
```

Перейдем теперь к изучению массивов строк. Можно считать, что массив строк является вектор-столбцом, каждый элемент которого есть строка, причем длины всех строк одинаковы. В результате получается прямоугольная матрица, состоящая из символов. Например, для переменных `s1='March'`, `s2='April'`, `s3='May'`, операция `S=[s1; s2]` является допустимой, и приводит к массиву строк:

```
S =
March
April
```

Аналогичное объединение трех строк `S=[s1; s2; s3]` вызовет вывод сообщения об ошибке. Можно, конечно, дополнить при помощи сцепления, каждую строку пробелами справа до длины наибольшей из строк и затем производить формирование массива. Функция `char` как раз и решает эту задачу, создавая из строк разной длины массив строк:

```
» S=char(s1,s2,s3)
S =
March
April
May
```

Проверьте при помощи команды `whos`, что `S` является массивом размера 3 на 5. Для определения размеров массива строк, так же, как и любых массивов, используется `size`.

Обращение к элементам массива строк производится аналогично обращению к элементам числового массива — при помощи индексирования числами или двоеточием, например: `S(3,2)`, `S(2,:)`, `S(2:3,1:4)`. При выделении строки из массива строк в конце могут остаться пробелы. Если они не

нужны, то их следует удалить, воспользовавшись функцией deblank. Сравните, к чему приводят S(3,:) и deblank(S(3,:)).

Поиск подстроки в массиве строк выполняется функцией strmatch. Входными аргументами strmatch являются подстрока и массив строк, а выходным — вектор с номерами строк, содержащих подстроку:

```
» p=strmatch('Ma',S)
p =
     1
     3
```

Если требуется искать номера строк, точно совпадающих с подстрокой, то следует задать третий дополнительный входной аргумент 'exact'.

Некоторые функций обработки строк могут работать и с массивами строк. Например, при сцеплении массивов с одинаковым числом строк при помощи strcat, сцепление применяется к соответствующим строкам массивов, образуя новый массив строк. Среди входных аргументов strcat может быть и строка. В этом случае она добавляется ко всем строкам массива.

Среди символов строки могут быть цифры и точка, т.е. строка может содержать числа. Важно понимать, что оператор присваивания b=10 приводит к образованию числовой переменной b, а оператор s='10' — строковой. Преобразование целого числа в строку производится при помощи функции int2str, входным аргументом которой является число, а выходным — строка, например:

```
» str=['May, ' int2str(10)]
str =
May, 10
```

Заметьте, что нецелые числа перед преобразованием округляются. Для перевода нецелых чисел в строки служит функция num2str:

```
» str=num2str(pi)
str =
3.1416
```

Дополнительный второй входной аргумент num2str предназначен для указания количества цифр в строке с результатом: str=num2str(pi,11). Возможно более гибкое управление преобразованием при помощи строки специального вида, определяющей формат (экспоненциальный или с плавающей точкой) и количество позиций, отводимых под число. Строка с форматом задается в качестве второго входного аргумента num2str, начинается со знака процента и имеет вид '%A.ax', где:

- A — количество позиций, отводимое под все число;
- a — количество цифр после десятичной точки;
- x — формат вывода, который может принимать одно из следующих значений: f (с плавающей точкой), e (экспоненциальный) или g (автоматический подбор наилучшего представления).

Сравните результаты следующих преобразований числа в строку:

```
num2str(198.23981, '%12.5f')
num2str(198.23981, '%12.5e')
num2str(198.23981, '%12.5g')
```

Если входным аргументом функций int2str и num2str является матрица, то результатом будет массив строк.

Обратная задача, а именно, преобразование строковой переменной, содержащей число, в числовую переменную решается при помощи функции str2num. Входным аргументом str2num может быть строка, содержащая представление целого, вещественного или комплексного числа в соответствии с правилами MatLab, например: str2num('2.9e-3'), str2num('0.1'), str2num('4.6+4i').

Задания для самостоятельной работы

Написать файл-функцию для решения поставленной задачи.

Варианты

1. Подсчитать число вхождений подстроки в строку.
2. Найти количество пробелов в строке.

3. Определить количество цифр в строке.
4. Удалить идущие подряд одинаковые символы в строке.
5. Заменить идущие подряд одинаковые символы в строке на один.
6. Строка является предложением, в котором слова разделены пробелами. Переставить первое и последнее слово.
7. Образовать строку, состоящую из первых букв строк, входящих в массив строк.
8. Вывести номера одинаковых строк в массиве строк.
9. Определить количество символов в каждой строке массива строк без учета пробелов.
10. По заданному массиву строк образовать новый, исключив повторяющиеся строки.
11. Заменить в строке цифры числительными (вместо 1, 2,… — один, два, три,…).
12. Задана строка, содержащая текст и числа, выделить числа в числовой массив.

§ 8. МАССИВЫ СТРУКТУР И МАССИВЫ ЯЧЕЕК

Массивы MatLab могут состоять не только из чисел или символов, но содержать и более сложно организованную информацию. Пусть необходимо хранить и обрабатывать информацию о группе из пяти студентов. Информация о каждом студенте включает в себя:

- фамилию;
- имя;
- год рождения;
- экзаменационные оценки по четырем предметам.

Структура данных, относящихся к каждому из студентов, одинакова (имеет одинаковые поля), а содержание структуры (значения полей структуры) индивидуально для каждого из студентов. Для хранения в MatLab такой однотипно организованной информации предназначен массив структур. Назовем массив структур `GR521`, а его поля: `Fam`, `Name`, `Year` и `Marks`.

Обращение к структурам массива производится при помощи индексации, например: `GR521(4)` — четвертая структура массива `GR521`. Названия полей отделяется от структуры при помощи точки. Скажем, для получения имени второго студента, следует воспользоваться обращением `GR251(2).Name`. Заметьте, что поля `Fam` и `Name` содержат строки, поле `Year` — число, а `Marks` — вектор длины четыре (поскольку записаны оценки по четырем предметам). Создайте файл-программу `fillinfo` для заполнения массива структур `GR251` (см. листинг 8.1).

После запуска `fillinfo` в рабочей среде образовался массив структур `GR521`. Функция `size` позволяет получить его размеры, в данном случае один на пять[12]. Длину одномерного массива структур возвращает функция `length`. Введите имя массива в командную строку и нажмите <Enter> — содержимое каждой структуры не отображается, выводятся только сведения о размере массива и названия полей. Для отображения в командном окне содержимого каждой структуры следует обратиться непосредственно к ней, например: `GR521(2)` или `disp(GR521(2))`.

Листинг 8.1. Файл-программа fillinfo для заполнения массива структур

```
GR521(1).Fam='Алексеев'; GR521(1).Name='Иван';
GR521(1).Year=1982;      GR521(1).Marks=[4 5 5 4];
GR521(2).Fam='Иванов';   GR521(2).Name='Сергей';
GR521(2).Year=1981;      GR521(2).Marks=[3 4 4 5];
GR521(3).Fam='Николаев'; GR521(3).Name='Олег';
GR521(3).Year=1981;      GR521(3).Marks=[5 5 5 5];
GR521(4).Fam='Петрова';  GR521(4).Name='Анна';
GR521(4).Year=1982;      GR521(4).Marks=[5 5 5 4];
GR521(5).Fam='Федорова'; GR521(5).Name='Елена';
GR521(5).Year=1982;      GR521(5).Marks=[3 3 3 4];
```

При работе с массивами структур требуется производить некоторые операции либо над структурами, либо с их содержимым. Элемент массива структур всегда можно выделить в отдельную структуру и наоборот, заменить его на другую структуру (с аналогичным набором полей). Например, для перестановки первых двух структур в массиве `GR521` достаточно использовать вспомогательную структуру `help`:

[12] Массив структур может быть и двумерным, тогда для его заполнения и обращения к его элементам следует применять два индекса (как для матриц).

```
» help=GR521(2);
» GR521(2)=GR521(1);
» GR521(1)=help;
```

Доступ к данным структур осуществляется при помощи полей, причем обязательно учитывать, что именно содержит поле: строку, число или массив. Предположим, что требуется по заданной структуре вида GR521 сформировать массив строк с фамилиями и именами каждого из студентов. Файл-функция namesgroup, приведенная на листинге 8.2, решает поставленную задачу. Предполагается, что входной аргумент GR является структурой вида GR521. Сначала в строковые переменные f и n выделяются фамилия и имя первого студента, из которых формируется строка N. Затем в цикле из каждой структуры массива GR извлекаются фамилия и имя текущего студента, они объединяются в строку str, которая добавляется в массив строк при помощи функции char.

Листинг 8.2. Файл-функция, формирующая массив строк с фамилиями и именами

```
function N=namesgroup(GR)
f=GR(1).Fam;
n=GR(1).Name;
N=[f, ' ', n];
for k=2:length(GR)
   f=GR(k).Fam;
   n=GR(k).Name;
   str=[f, ' ', n];
   N=char(N, str);
end
```

Проверьте работу файл-функции namesgroup, вызвав ее от массива GR521 с выходным аргументом NAMES, а затем отобразите содержимое NAMES в командное окно при помощи функции disp:

```
» NAMES=namesgroup(GR521);
» disp(NAMES)
NAMES =
Алексеев Иван
Иванов Сергей
Николаев Олег
Петрова Анна
Федорова Елена
```

Структуры массива можно дополнить новым полем, для чего следует присвоить значение этому полю в какой-нибудь структуре массива, например, первой. Добавьте в массив GR521 поле NBook, предназначенное для хранения номера зачетной книжки студента:

```
» GR521(1).NBook=599001;
```

Созданное поле, разумеется, образуется во всех структурах массива, но только в первой из них будет содержать заданное значение (проверьте!). Поле NBook остальных структур массива следует заполнить отдельно. Обратная операция — удаление поля из всех структур массива — осуществляется при помощи функции rmfield. Входными аргументами rmfield являются имя массива структур и название поля, а выходным — имя массива, в который следует записать обновленный массив структур:

```
» g251=rmfield(GR521,'Year');
```

При работе с массивами структур часто возникает ситуация, когда название поля хранится в строковой переменной и требуется получить доступ к соответствующему полю. Функция getfield предназначена для получения значения поля структуры по строке с его названием. Входными аргументами getfield являются структура и строка с названием поля, а выходным — его значение.

```
» field2='Fam';
» fam1=getfield(GR521(1),field2);
```

Заметьте, что к аналогичному результату приводит fam1=GR512(1).Fam. Для изменения значения поля по строке с его названием служит функция setfield. Во входных аргументах setfield задаются: структура, строка с названием поля и его новое значение, а в выходном — возвращается измененная структура:

```
» field3='Year';
» GR521(1)=setfield(GR521(1),field3,1981);
```

На том же самом принципе основана функция struct, которая заполняет структуру по строкам с на-

званием полей. Входными ее аргументами являются пары 'название поля'-значение, а выходным — структура:

```
» GR521(6)=struct('Fam','Котов','Name','Петр','Year',1980,...
'Marks',[3 5 4 4],'NBook',59908)
```

Определение названий полей в заданном массиве структур или структуре производится при помощи функции `fieldnames`. Ее входным аргументом является массив структур или структура, а выходным — названия полей, записанные в массив ячеек. Про массив ячеек сказано ниже, приведем здесь только примеры, демонстрирующие получение названий полей с использованием `fieldnames`:

```
» Fields=fieldnames(GR521);
» field3=Fields(3)
field3 =
    'Year'
```

Массив ячеек является самым универсальным способом хранения разнородных данных. Его элементами могут быть числа, числовые массивы, строки, структуры и массивы строк и структур. Присваивание значений ячейкам массива требует заключения индексов в фигурные скобки, а при обращении к ячейкам можно использовать как фигурные, так и круглые скобки. Листинг 8.3 содержит пример заполнения двумерного массива ячеек `CMAS` размера два на два. Ячейка с номером {1,1} содержит матрицу, {1,2} — строку, {2,1} — массив строк, {2,2} — структуру с полями `Data` и `Time`.

Листинг 8.3. Заполнение массива ячеек

```
clear CMAS
CMAS{1,1}=[-2.2 0.9; 5.6 -8.3];
CMAS{1,2}='This is a string';
CMAS{2,1}=char('first string','second string');
CMAS{2,2}.Data=[3.981 9.765 4.442 0.003];
CMAS{2,2}.Time=[0.11 0.12 0.13 0.14];
```

Обратите внимание, что перед заполнением массива ячеек целесообразно использовать команду `clear CMAS` для удаления (возможно существующей в рабочей среде) переменной `CMAS`. Дело в том, что при наличии, к примеру, числовой переменной `CMAS`, последующее заполнение ее как массива ячеек приведет к сообщению об ошибке. Другие типы переменных не требуют вызова `clear` — последовательность присваиваний `s=1`, `s='string'` допустима в MatLab.

Для отображения данных, хранящихся в массиве ячеек, MatLab предлагает две функции: `celldisp`, которая выводит в командное окно содержимое каждой ячейки, и `cellplot`, предназначенная для визуализации содержимого массива в графическом окне. Программная обработка разнородных данных значительно облегчается благодаря ряду функций, которые позволяют установить тип содержимого ячейки[13]. Перечисленные ниже функции возвращают логическую единицу или ноль в зависимости от типа входного аргумента:

- `ischar(a)` — равно 1, если `a` является строкой или массивом строк;
- `isnumeric(a)` — равно 1, если `a` является числом или числовым массивом;
- `isstruct(a)` — равно 1, если `a` является структурой или массивом структур.

Задания для самостоятельной работы

Задан массив структур с информацией о группе студентов. Написать файл-функцию для решения следующей задачи.

Варианты

1. Подсчитать средний балл каждого студента и вывести столбцевую диаграмму успеваемости.
2. Найти фамилию наиболее успевающего студента.
3. Сформировать матрицу, строки которой содержат оценки каждого из студентов.
4. Определить, есть ли в группе студент с заданной фамилией.
5. Расположить структуры массива в соответствии с успеваемостью студентов.

Задания для самостоятельной работы

Задан одномерный массив ячеек, который может содержать данные различных типов. Написать файл-функцию для решения следующей задачи.

[13] Разумеется, при помощи данных функций можно определить тип любой переменной, а не только ячейки массива.

Варианты

1. Определить количество ячеек, содержащих числа или числовые массивы.
2. Найти номера ячеек, которые содержат только числа, но не массивы.
3. Объединить в один массив все строки и массивы строк, входящие в массив ячеек.
4. Отобразить столбцевыми диаграммами все данные, являющиеся векторами.
5. Выделить в массив структур все структуры, входящие в массив ячеек (предполагается, что поля всех структур одинаковы).

§ 9. ТЕКСТОВЫЕ ФАЙЛЫ

MatLab предлагает достаточно универсальные способы считывания данных из текстовых файлов и записи данных в требуемом виде в текстовый файл. Рассмотрим сначала оперирование с числовыми данными, представленными матрицами и векторами. Пусть в текстовом файле vec.dat в столбик записаны числа в соответствии с правилами MatLab, т.е. для отделения десятичных знаков используется точка, а для записи числа в экспоненциальном виде применяется символ `e`. Функция `load` позволяет занести содержимое файла vec.dat в числовой массив. Имя файла, заключенное в апострофы, указывается во входном аргументе `load`, а в выходном аргументе возвращается вектор-столбец[14]:

```
» v=load('vec.dat');
```

Обратная операция — запись значений вектор-столбца в файл — производится при помощи команды `save`, аргументами которой являются: имя файла, переменная, и дополнительный параметр `–ascii`, означающий запись в текстовом виде:

```
» v2=v.^2;
» save 'vec2.dat' v2 –ascii
```

Если числа в текстовом файле расположены в строку и разделены пробелами, то результатом считывания будет вектор-строка. Верно и обратное, запись значений вектор-строки приведет к занесению их в строку в текстовом файле.

Запись матрицы командой `save` приводит к образованию текстового файла с таблицей чисел. Текстовый файл, в каждой строке которого находится одинаковое количество чисел, отделенных друг от друга пробелами, считывается в двумерный массив функцией `load`.

Команды `load` и `save` предназначены только для простейшего ввода-вывода числовых данных. Разберем теперь более сложный случай, когда считываемая или записываемая информация содержит не только числа, но и текст. Работа с текстовыми файлами состоит из трех этапов:

1) открытие файла;
2) считывание или запись данных;
3) закрытие файла.

Для открытия файла служит функция `fopen`, которая вызывается с двумя входными аргументами: именем файла и строкой, задающей способ доступа к файлу. Выходным аргументом `fopen` является идентификатор файла, т.е. переменная, которая впоследствии используется при любом обращении к файлу. Функция `fopen` возвращает –1, если при открытии файла возникла ошибка. Существует четыре основных способов открытия файла:

1) `f=fopen('myfile.dat','rt')` — открытие текстового файла myfile.dat только для чтения из него;

2) `f=fopen('myfile.dat','rt+')` — открытие текстового файла myfile.dat для чтения и записи данных;

3) `f=fopen('myfile.dat','wt')` — создание пустого текстового файла myfile.dat только для записи данных;

4) `f=fopen('myfile.dat','wt+')` — создание пустого текстового файла myfile.dat для записи и чтения данных.

При использовании двух последних вариантов следует соблюдать осторожность — если файл myfile.dat уже существует, то его содержимое будет уничтожено. После открытия файла появляется возможность считывать из него информацию, или заносить ее в файл. По окончании работы с файлом необходимо закрыть его при помощи `fclose(f)`.

[14] Файл с данными должен находится в текущем каталоге MatLab, иначе требуется указать полное имя файла.

Построчное считывание информации из текстового файла производится при помощи функции `fgetl`. Входным аргументом `fgetl` является идентификатор файла, а выходным — текущая строка. Каждый вызов `fgetl` приводит к считыванию одной строки и переводу текущей позиции в файле на начало следующей строки. Команды, приведенные на листинге 9.1, последовательно считывают из файла myfile.dat первые три строки в строковые переменные `str1`, `str2`, `str3`.

Листинг 9.1. Считывание трех первых строк из текстового файла

```
f=fopen('myfile.dat','rt');
str1=fgetl(f);
str2=fgetl(f);
str3=fgetl(f);
fclose(f);
```

При последовательном считывании рано или поздно будет достигнут конец файла, при этом функция `fgetl` вернет минус 1. Лучше всего перед считыванием проверять, не является ли текущая позиция в файле последней. Для этого предназначена функция `feof`, которая вызывается от идентификатора файла и возвращает единицу, если текущая позиция последняя и ноль, в противном случае. Обычно последовательное считывание организуется при помощи цикла `while`. Листинг 9.2 содержит файл-функцию `viewfile`, которая считывает строки файла, и выводит их в командное окно.

Листинг 9.2. Файл-функция viewfile

```
function viewfile(fname)
f=fopen(fname,'rt');
while feof(f)==0
   s=fgetl(f);
   disp(s)
end
fclose(f);
```

Вызов `viewfile('viewfile.m')` приводит к отображению текста самой файл-функции в командном окне.

Строки записываются в текстовый файл при помощи функции `fprintf`, ее первым входным аргументом является идентификатор файла, а вторым — добавляемая строка. Символ `\n` служит для перевода строки. Если поместить его в конец добавляемой строки, то следующая команда `fprintf` будет осуществлять вывод в файл с новой строки, а если `\n` находится в начале, то текущая команда `fprintf` выведет текст с новой строки. Например, последовательность команд (листинг 9.3) приведет к появлению в файле my.txt текста из двух строк (листинг 9.4).

Листинг 9.3. Вывод строк в текстовый файл

```
f=fopen('my.txt','wt');
fprintf(f,'текст ');
fprintf(f,'еще текст\n');
fprintf(f,'а этот текст с новой строки');
fclose(f);
```

Листинг 9.4. Результат работы операторов листинга 9.3

```
текст еще текст
а этот текст с новой строки
```

Аналогичного результата можно добиться, переместив `\n` из конца строки второй функции `fprintf` в начало строки третьей функции `fprintf`. Аргументом `fprintf` может быть не только строка, но и строковая переменная. В этом случае для обеспечения вывода с новой строки ее следует сцепить со строкой `'\n'`.

Строка, предназначенная для вывода в текстовый файл, может содержать как текст, так и числа. Часто требуется выделить определенное количество позиций под число и вывести его в экспоненциальном виде или с плавающей точкой и с заданным количеством цифр после десятичной точки. Здесь не обойтись без форматного вывода при помощи `fprintf`, обращение к которой имеет вид:

```
fprintf(идентификатор файла, 'форматы', список переменных)
```

В списке переменных могут быть как числовые переменные (или числа), так и строковые переменные (или строки). Второй аргумент является строкой специального вида с форматами, в которых будут выводится все элементы из списка. Каждый формат начинается со знака процента. Для вывода строк используется формат `s`, а для вывода чисел — `f` (с плавающей точкой)

или e (экспоненциальный). Число перед s указывает количество позиций, отводимых под вывод строки. При выводе значений числовых переменных перед f или e ставится два числа, разделенных точкой. Первое из них означает количество позиций, выделяемых под все значение переменной, а второе — количество знаков после десятичной точки. Таким образом, под каждый из элементов списка отводится поле определенной длины, выравнивание в котором по умолчанию производится по правому краю. Для выравнивания по левому краю следует после знака процента поставить знак минус. Ниже приведены варианты вызова fprintf и результаты, незаполненные позиции после вывода (пробелы) обозначены символом ○.

```
fprintf(f,'%6.2f', pi)            ○○3.14
fprintf(f,'%-6.2f', pi)           3.14○○
fprintf(f,'%14.4e', exp(-5))      ○○○6.7379e-003
fprintf(f,'%-14.4e', exp(-5))     6.7379e-003○○○
fprintf(f,'%8s', 'текст')              ○○○текст
fprintf(f,'%-8s', 'текст')        текст○○○
```

В случае, когда зарезервированного поля не хватает под выводимую строку или число, MatLab автоматически увеличивает длину поля[15], например:

```
fprintf(f,'%5.4e', exp(-5))       6.7379e-003
```

При одновременном выводе чисел и текста пробелы могут появляться из-за неполностью заполненных полей, выделенных как под числа, так и под строки. Приведенные ниже операторы и результат их выполнения демонстрируют расположение полей в строке текстового файла. Волнистой линией подчеркнуты поля, выделенные под числа, а прямой — под строки, символ ○ по-прежнему обозначает пробелы.

```
x=0.55;
fprintf(f,'%-5s%6.2f%6s%20.8e','x=',x,'y=',exp(x))
x=○○○○○0.55○○○○y=○○○○○1.73325302e+000
```

Форматный вывод удобен при формировании файла с таблицей результатов. Предположим, что необходимо записать в файл f.dat таблицу значений функции $f(x) = x^2 \sin x$ для заданного числа n значений $x \in [a,b]$, отстоящих друг от друга на одинаковое расстояние. Файл с таблицей значений должен иметь такую структуру, как показано на листинге 9.5.

Листинг 9.5. Текстовый файл с таблицей значений функции

```
f(0.65)=2.55691256e-001
f(0.75)=3.83421803e-001
f(0.85)=5.42800093e-001
f(0.95)=7.34107493e-001
f(1.05)=9.56334106e-001
```

Очевидно, что следует организовать цикл от начального значения аргумента до конечного с шагом, соответствующим заданному числу точек, а внутри цикла вызывать fprintf с подходящим списком вывода и форматами. Символ \n, предназначенный для перевода строки, помещается в конец строки с форматами (см. листинг 9.6) .

Листинг 9.6. Файл-функция tab, выводящая таблицу значений функции

```
function tab(a,b,n)
h=(b-a)/(n-1);
f=fopen('f.dat','wt');
for x=a:h:b
   fprintf(f,'%2s%4.2f%2s%15.8e\n', 'f(',x,')=',x^2*sin(x))
end
fclose(f);
```

Обратимся теперь к считыванию данных из текстового файла. Функция fscanf осуществляет обратное действие по отношению к fprintf. Каждый вызов fscanf приводит к занесению данных,

[15] Лучше избегать такие ситуации и заранее рассчитывать длину поля вывода, иначе, например, при выводе таблицы может нарушиться структура данных в файле.

начинающихся с текущей позиции, в переменную. Тип переменной определяется заданным форматом. В общем случае, обращение к fscanf имеет вид:

```
a=fscanf(идентификатор файла, 'формат', число считываемых элементов)
```

Для считывания строк используется формат %s, для целых чисел — %d, а для вещественных — %g.

Предположим, что файл exper.dat содержит текст, приведенный на листинге 9.7. Данные отделены друг от друга пробелами. Требуется считать дату проведения эксперимента (число, месяц и год) в подходящие переменные, а результаты занести в числовые массивы TIME и DAT.

Листинг 9.7. Файл с данными exper.dat

```
Результаты экспериментов 10 мая 2002
t= 0.1 0.2 0.3 0.4 0.5
G= 3.02 3.05 2.99 2.84 3.11
```

Считывание разнородных данных (числа, строки, массивы) требует контроля, который достигается применением форматов. Первые два элемента файла exper.dat являются строками, следовательно можно сразу занести их в одну строковую переменную head. Очевидно, надо указать формат %s и число 2 считываемых элементов[16]. Далее требуется считать целое число 10, строку 'мая' и снова целое число 2002. Обратите внимание, что перед заполнением массива TIME еще придется предусмотреть считывание строки 't='. Теперь все готово для занесения пяти вещественных чисел в вектор-столбец TIME при помощи формата %g. Данные из последней строки файла exper.dat извлекаются аналогичным образом (листинг 9.8).

Листинг 9.8. Применение fscanf для считывания данных

```
f=fopen('exper.dat');
head=fscanf(f,'%s',2)
data=fscanf(f,'%d')
month=fscanf(f,'%s',1)
year=fscanf(f,'%d')
str1=fscanf(f,'%s',1)
TIME=fscanf(f,'%g',5)
str2=fscanf(f,'%s',1)
DATA=fscanf(f,'%g',5)
fclose(f);
```

Информация в текстовом файле может быть представлена в виде таблицы. Для считывания такой информации следует использовать массив структур с подходящим набором полей. Считывание всей информации реализуется в цикле while, в условии которого производится проверка на достижение конца файла при помощи функции feof.

Функция fscanf предоставляет возможность считывать числа из текстового файла не только в вектор-столбец, но и массив заданных размеров. Расположение чисел по строкам в файле не имеет значения. Размеры формируемого массива указываются в векторе в третьем входном аргументе fscanf. Рассмотрим считывание числовых данных из файла matr.txt (листинг 9.9).

Листинг 9.9. Текстовый файл matr.txt с матрицей

```
1.1 2.2 3.3 4.4
5.5 6.6 7.7 8.8
0.2 0.4 0.6 0.8
```

Функция fscanf формирует столбец матрицы, последовательно считывая числа из файла (т. е. построчно). Следовательно, для заполнения матрицы с тремя строками и четырьмя столбцами, необходимо считать данные из файла в массив четыре на три, а затем транспонировать его (листинг 9.10).

Листинг 9.10. Считывание матрицы из текстового файла

```
f=fopen('matr.txt');
M=fscanf(f,'%g', [4 3]);
M=M'
fclose(f)
```

[16] Заметьте, что слова заносятся в строковую переменную без пробела между ними. Для получения строки со словами, разделенными пробелом, необходимо считать их по отдельности в разные строковые переменные и сцепить их.

Обратите внимание, что массив может иметь размеры не только 3 на 4. Поскольку данные считываются подряд, то они могут быть занесены и в массив 2 на 6, и 4 на 3 и т. д.

Задания для самостоятельной работы

Написать файл-функцию для считывания данных из файла в структуру или массив структур с подходящими полями.

Варианты

1.
```
Алексеев    Сергей      1980    5 4 4 5 3 5
Иванов      Константин  1981    3 4 3 4 3 5
Петров      Олег        1980    5 5 5 4 4 5
```

2.
```
21  марта   2002    0.56  0.58  0.49  0.44
23  марта   2002    0.36  0.32  0.28  0.25
25  марта   2002    1.62  1.68  1.71  1.91
```

3.
```
195251  СПб  Политехническая      29
195256  СПб  Науки           49
195256  СПб  Науки           24
```

4.
```
Результаты наблюдений
Time= 0.0  0.1  0.2  0.3  0.4  0.5  0.6
      Mass=
           2.1  2.3  2.3  1.9  1.8  2.4
      0.8  0.7  0.5  1.1  3.2  0.3
```

5.
```
Алексеев    Иван        121-22-04
    Сидоров     Николай     101-21-99
Тимофеев    Сергей      570-00-03
```
(номера телефонов, должны быть записаны в поля структур как целые числа).

Задания для самостоятельной работы

Считать матрицы и вектора из файла в подходящие по размеру массивы. Обратите внимание, что в файлах содержится рядом две или три матрицы или вектора, их следует занести в разные массивы.

Варианты

1.
```
0.1  0.2  0.3      9.91
1.9  0.4  0.1      8.01
4.7  5.1  3.9      7.16
```

2.
```
1.399  2.001  9.921          3.21  0.12
0.129  1.865  8.341          9.33  8.01
9.136  8.401  7.133          3.12  3.22
```

3.
```
1  2  3  4        99  80
5  6  7  8        33  21
                  15  90
```

4.
```
10  20  40  50     12          19  21  32  44
-1  -2  -3  -4     32
                   10
```

5.	1 2 3 4		100
	6 7 8 9	0.1 0.2 0.3 0.4	200
		0.5 0.6 0.7 0.8	300

Библиографический список

1. Ануфриев И.Е. Самоучитель MatLab 5.3/6.X. СПб.: БХВ-Петербург, 2002. 736 с.
2. Гультяев А. Визуальное моделирование в среде MatLab. СПб.: Питер, 2000. 430 с.
3. Дьяконов В.П. MatLab. СПб.: Питер, 2001. 553 с.
4. Дьяконов В.П. MatLab 5.0/5.3. Система символьной математики. М.: Нолидж, 1999. 633 с.
5. Лазарев Ю.Ф. MatLab 5.X. Киев: BHV: Ирина, 2000. 383 с.
6. Мартынов Н.Н. MatLab 5.X. Вычисления, визуализация, программирование. М.: Кудиц-Образ, 2000. 332 с.
7. Потемкин В.Г. Система MatLab: Справ. пособие. М.: Диалог-МИФИ, 1997. 350 с.
8. Потемкин В.Г. MatLab 5 для студентов. 2-е изд., испр. и доп. М.: Диалог-МИФИ, 1999. 447 с.
9. Потемкин В.Г. Система инженерных и научных расчетов MatLab 5.X: В 2-х т. М.: Диалог-МИФИ, 1999. 670 с.