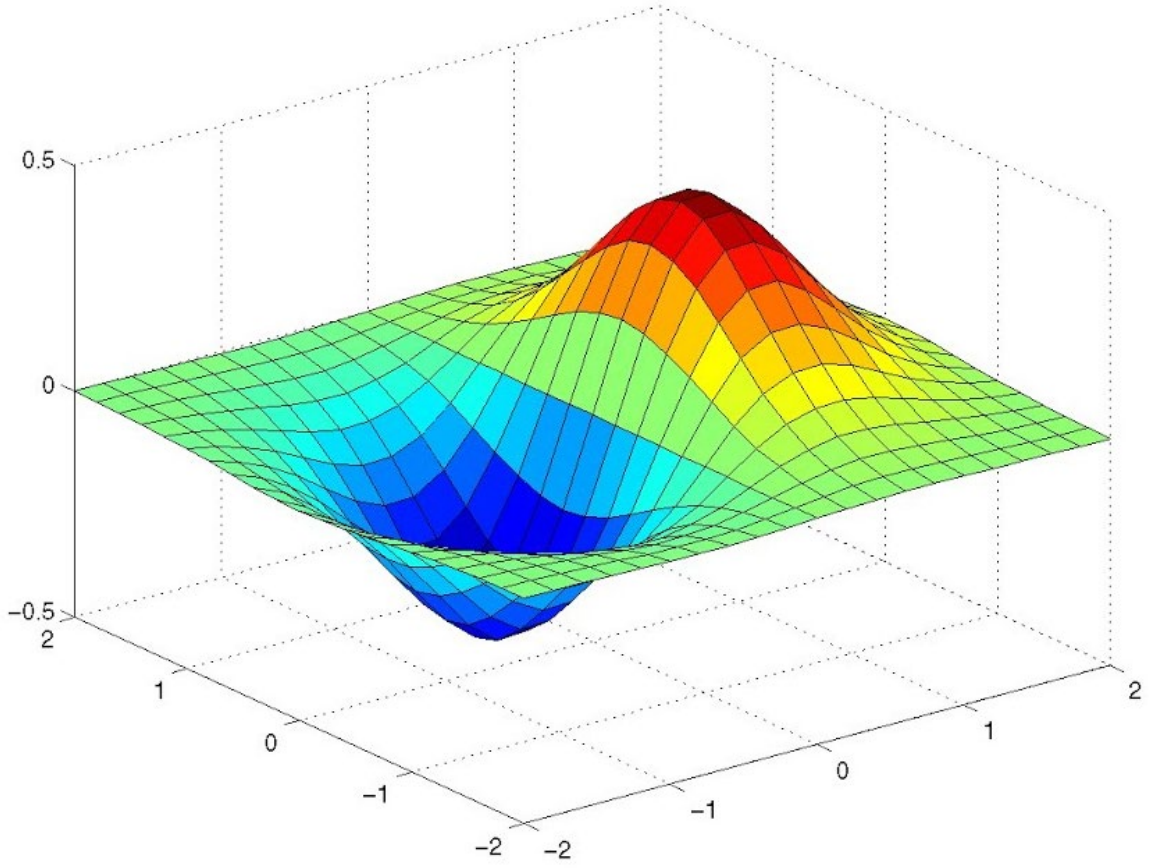
П. В. БАЛАБАНОВ, Д. А. ЛЮБИМОВА, Н. М. ГРЕБЕННИКОВА

МЕТОДЫ И ПРОГРАММНЫЕ СРЕДСТВА ОБРАБОТКИ ДАННЫХ



Тамбов
Издательский центр ФГБОУ ВО «ТГТУ»
2024

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Тамбовский государственный технический университет»

П. В. БАЛАБАНОВ, Д. А. ЛЮБИМОВА, Н. М. ГРЕБЕННИКОВА

МЕТОДЫ И ПРОГРАММНЫЕ СРЕДСТВА ОБРАБОТКИ ДАННЫХ

Утверждено Ученым советом университета в качестве практикума
для студентов 1 курса направления подготовки 27.03.02,
2 курса направления подготовки 15.03.06 всех форм обучения

Учебное электронное издание

Тамбов
Издательский центр ФГБОУ ВО «ТГТУ»
2024

УДК 00.004.78
ББК 39.972.53
Б20

Рецензенты:

Доктор технических наук, доцент, главный научный сотрудник, заведующий лабораторией ФГБНУ «ВНИИТиН»
А. В. Балашов

Кандидат технических наук, доцент, заведующий кафедрой «Системы автоматизированной поддержки принятия решений» ФГБОУ ВО «ТГТУ»
И. Л. Коробова

Балабанов, П. В.

Б20 Методы и программные средства обработки данных [Электронный ресурс] : практикум / П. В. Балабанов, Д. А. Любимова, Н. М. Гребенникова. – Тамбов : Издательский центр ФГБОУ ВО «ТГТУ», 2024. – 1 электрон. опт. диск (CD-ROM). – Системные требования : ПК не ниже класса Pentium II ; CD-ROM-дисковод ; 3,8 Mb ; RAM ; Windows 95/98/XP ; мышь. – Загл. с экрана.

ISBN 978-5-8265-2803-7

Содержит широкий перечень практических и лабораторных работ по обработке данных в среде MATLAB, а также на языке Python с использованием специализированных библиотек, с подробным теоретическим пояснением и наглядными примерами.

Предназначен для студентов 1 курса направления подготовки 27.03.02, 2 курса направления подготовки 15.03.06 всех форм обучения.

УДК 00.004.78
ББК 39.972.53

ISBN 978-5-8265-2803-7

ВВЕДЕНИЕ

Владение методами обработки данных при помощи систем автоматизированных расчетов, а также языков программирования высокого уровня неизменно является необходимым навыком для современного инженера, которому зачастую приходится работать с большими объемами информации. Данный практикум содержит широкий перечень практических и лабораторных работ по обработке данных в среде MATLAB, а также на языке Python с использованием специализированных библиотек, с подробным теоретическим пояснением и наглядными примерами.

MatLab – одна из старейших и тщательно проработанных систем автоматизации математических расчетов, созданная еще в конце 1970-х годов. Она обладает широкими возможностями, является гибкой и расширяемой системой, легко приспосабливаемой к решению различных классов задач, и характеризуется высокой скоростью вычислений в сравнении с другими подобными системами.

Python – это высокоуровневый язык программирования, отличающийся эффективностью, простотой и универсальностью использования. Он широко применяется в разработке веб-приложений и прикладного программного обеспечения, а также в машинном обучении и обработке больших данных. За счет простого и интуитивно понятного синтаксиса является одним из распространенных языков для обучения программированию.

Практикум может быть интересен студентам и специалистам широкого круга инженерных специальностей.

ПРАКТИЧЕСКАЯ РАБОТА 1
ОСНОВЫ РАБОТЫ В СРЕДЕ MATLAB

Цель работы: изучить интерфейс основного окна MatLab R2018b, математические и логические функции системы.

Главная отличительная черта системы – это легкость ее модификации и адаптация к конкретным задачам пользователя. Пользователь может ввести в систему любую новую команду, оператор или функцию и пользоваться затем ими так же просто, как и заведомо встроенными операторами и функциями. Новые определения в системе MatLab хранятся в виде файлов на диске, имеющих расширение *.m. Это делает набор операторов и функций практически неограниченным.

В базовый набор слов системы входят: спецзнаки; знаки арифметических и логических операций; арифметические, алгебраические, тригонометрические и некоторые специальные функции; функции быстрого преобразования Фурье и фильтрации; векторные и матричные функции; средства для работы с комплексными числами; операторы построения гистограмм, графиков в декартовой и полярной системах координат, трехмерных поверхностей и др.

Достоинством системы является достаточно простой и традиционный входной язык, напоминающий язык высокого уровня Бейсик. При этом система дает возможность редактировать программы с помощью любого текстового редактора.

При работе с системой возможны два режима работы: **режим прямых вычислений**, когда пользователь на клавиатуре вводит вычисляемое выражение, нажимает «Enter», а система выдает результат; и **режим программы**, когда пользователем последовательность необходимых вычислений предварительно оформляется в виде программы, а затем данная программа запускается на выполнение.

В режиме прямых вычислений команда вводится после приглашающего символа «>>». Ввод команды завершается клавишей «Enter».

В арифметических выражениях можно использовать следующие знаки арифметических операций: + (сложение); – (вычитание); * (умножение); / (деление); ^ (возведение в степень).

Например,

```
>> 2 + 2*2
            ans =
                        6
```

здесь *ans* – переменная, хранящая результат последней операции. Кроме этого, система содержит следующие системные переменные: *pi* – число «пи», *inf* – значение машинной бесконечности. В случае возникновения неопределенности типа *0*/*0* или *inf*/*inf* система выдает результат *NaN* (*not a number*).

Если используется операция присваивания –

Имя_переменной = *Выражение* [;]

то результат вычисления выражения помещается не в системную переменную ans, а в переменную с именем *«Имя_переменной»*. Символ «;» указывает на то, что результат вычисления не будет выводиться на экран. Этот символ используется также и для разделения нескольких операторов в одной строке.

Имена переменных должны формироваться из букв латинского алфавита, цифр, символа подчеркивания и могут начинаться только с буквы, например

```
>> k = 2 + 2*2          >> km2 = 2 + 2*2       >> k_1 = 2 + 2*2
       k =                    km2 =                  k_1 =
                 6                      6                      6
```

Замечание: **заглавные и строчные буквы системой различаются!**

Система MatLab может работать как с действительными, так и с комплексными числами, которые записываются следующим образом

```
>> 5 + 6*i
       ans =
                 5.0000 + 6.0000i
```

Математические функции системы MatLab

В MatLab имеются следующие математические функции (Z – в общем случае число комплексное, X и Y – только действительные числа):

abs(Z)	– вычисление модуля комплексного или действительного числа;
angle(Z)	– вычисление аргумента;
sqrt(Z)	– вычисление квадратного корня;
real(Z)	– вычисление действительной части числа;
imag(Z)	– вычисление мнимой части числа;
round(X)	– округление до целого;
fix(X)	– округление до ближайшего целого в сторону нуля;
floor(X)	– округление до ближайшего целого в сторону отрицательной бесконечности;
sign(X)	– вычисление функции знака $\begin{cases} -1, & \text{при } X < 0, \\ 0, & \text{при } X = 0, \\ 1, & \text{при } X > 0; \end{cases}$
rem(X, Y)	– вычисление остатка от деления X на Y;
exp(Z)	– вычисление экспоненты;
log(Z)	– вычисление натурального логарифма;
log10(Z)	– вычисление десятичного логарифма;
sin(Z)	– вычисление синуса;
cos(Z)	– вычисление косинуса;
tan(Z)	– вычисление тангенса;
asin(Z)	– вычисление арксинуса;

acos(Z)	– вычисление арккосинуса;
atan(Z)	– вычисление арктангенса;
sinh(Z)	– вычисление гиперболического синуса;
cosh(Z)	– вычисление гиперболического косинуса;
tanh(Z)	– вычисление гиперболического тангенса;
asinh(Z)	– вычисление гиперболического арксинуса;
acosh(Z)	– вычисление гиперболического арккосинуса;
atanh(Z)	– вычисление гиперболического арктангенса.

Например,

```
>> Z=20+20i; asin(Z)
        ans =
              0.7851 + 4.0355i
```

Операторы отношения и логические операции

В системе могут применяться следующие операторы отношения (сравнения):

<	– меньше чем,	>=	– больше или равно,
<=	– меньше или равно,	==	– равно,
>	– больше чем,	~=	– не равно.

Эти операторы используются, как правило, в режиме программы (а не прямых вычислений) для сравнения двух величин или арифметических выражений. Они возвращают значение «Неверно» или «Верно» в виде чисел 0 или 1 соответственно, например

```
>> Z=9; Z==0
     ans =
                 0
```

Возможно также выполнение логических операций:

&	– логическое умножение (И);
|	– логическое сложение (ИЛИ);
~	– логическое отрицание (НЕ).

ПРАКТИЧЕСКАЯ РАБОТА 2
МАТРИЦЫ И ОСНОВНЫЕ МАТРИЧНЫЕ ОПЕРАЦИИ В MATLAB

Цель работы: получить навыки работы с матрицами и векторами в системе MatLab; ознакомиться с основными матричными и векторными функциями.

Поскольку MatLab – матрично-ориентированная система, то в ней даже обычная переменная представляет собой матрицу размерностью 1×1. Чтобы задать вектор (матрицу из одной строки), состоящий, например, из трех элемен-

тов, то их значения необходимо перечислить в квадратных скобках, разделяя пробелами или запятыми:

```
>> P = [12  25  31]
       P =
              12       25       31
```

Получить значение каждого элемента вектора можно по его индексу, например, для второго элемента

```
>> P(2)
     ans =
                  25
```

Задание матрицы требует указания различных строк. Для различения строк используется знак «;», например:

```
>> Q = [3  6; 4  10]
       Q =
                  3        6
                  4       10
>> Q(2, 1)
     ans =
                    4
```

Обращение к элементам матрицы при помощи одного индекса, поясним на следующем примере:

```
>> A = [1  2  3; 4  5  6; 7  8  9]
A =
       1       2       3
       4       5       6
       7       8       9
```

```
>> A(2)
ans =
      4
```

```
>> A(6)
ans =
       8
```

Можно осуществлять вывод одной строки или одного столбца матрицы. Для этого используется знак «:», например,

Вывод 2-й строки

```
>> A(2, :)
ans =
     4      5      6
```

Вывод 3-го столбца

```
>> A(: , 3)
ans =
       3
       6
       9
```

Для преобразования матрицы в столбец используется также символ «:»

```
>> A(:)
```

Помимо поэлементного ввода векторов и матриц можно задавать значения в виде арифметической прогрессии с шагом 1:

```
>> x =[1:5]
      x =
           1   2   3   4   5
```

с произвольным шагом

```
>> x =[1:0.5:3]
      x =
            1.0000   1.5000   2.0000   2.5000   3.0000
```

или с шагом, заданным при помощи арифметического выражения:

```
>> x =[0:pi/4:pi]
     x =
                    0   0.7854   1.5708   2.3562   3.1416
```

Следующие матричные функции обеспечивают генерацию некоторых наиболее распространенных видов матриц размерностью M строк на N столбцов:

zeros(M, N)	– генерация матрицы с нулевыми элементами;
ones(M, N)	– генерация матрицы с единичными элементами;
rand(M, N)	– генерация матрицы с элементами, имеющими случайные значения;
eye(M, N)	– генерация матрицы с единичными диагональными элементами.

Например,

```
>> Q = eye(3, 4)
       Q =
                  1     0     0     0
                  0     1     0     0
                  0     0     1     0
```

Транспонирование квадратной матрицы осуществляется при помощи символа апостроф («'»):

```
>> A = [1  2 ; 3  4]
       A =
                  1     2
                  3     4
>> B = A'
       B =
                  1     3
                  2     4
```

Для сложения и вычитания матриц используются обычные знаки «+» и «–». Умножение двух матриц по правилам линейной алгебры осуществляется с использованием знака «*». В случае, когда требуется перемножить поэлементно две матрицы, используется знак «.*», например

```
>> A = [1  2 ; 3  4];  B = [3  2 ; 0  1]; C = A*B
        C =
                  3     4
                  9    10
>> C = A.*B
        C =
                  3     4
                  0     4
```

Аналогично, знаки «./», «.^» означают поэлементное деление и поэлементное возведение каждого элемента матрицы в степень соответственно.

Пример. Решение системы линейных алгебраических уравнений.

$$\begin{cases} 3x_1 + 2x_2 + x_3 = 4 \\ x_1 + x_2 - x_3 = 1 \\ x_1 - 2x_2 + x_3 = 3 \end{cases} \qquad \begin{pmatrix} 3 & 2 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 3 \end{pmatrix}$$

В матричной форме система записывается следующим образом A×X = B, где A – матрица линейных коэффициентов, X – искомый вектор неизвестных, B – вектор свободных членов. Тогда решение данной системы – $X = A^{-1} \times B$, где A^{-1} – обратная матрица. В MatLab обращению матрицы соответствует возведение ее в степень –1.

Выполним решение заданной системы уравнений:

```
>> A=[3 2 1; 1 1 -1; 1 -2 1]; B = [4; 1; 3]; X = A^(-1)*B
      X =
            1.7000
            -0.6000
            0.1000
```

Для вычисления определителя квадратной матрицы M используется функция det(M), например:

```
>> M = eye(3,3)
       M =
       1     0     0
       0     1     0
       0     0     1
>> det(M)
     ans =
                 1
```

Для определения размерности вектора-строки используется функция *length*, а размерности матрицы – *size*, например

```
>> A = [3 2 1];  B = [1 2 ; 3 4; 5 6];  k = length(A)
k =
     3
>> [m, n] = size(B)
m =
      3
n =
      2
```

Функции min (max) для матрицы возвращают вектор-строку минимальных (максимальных) значений каждого столбца матрицы, например:

```
>> A = [3  2  1 ; 1  2  3];  min(A)
       ans =
                    1       2       1
```

Применение этих функций к векторам-строкам позволяет определить наибольший (наименьший) элемент в строке, например:

```
>> A = [3  2  1];  max(A)
       ans =
                    3
```

а также его порядковый номер в строке

```
>> [M, m] = max(A)
     M =
                3
      m =
                1
```

Функции *sum*, *prod*, *mean* и *std* для матриц возвращают вектор-строку сумм, произведений, средних значений и стандартных квадратичных отклонений каждого столбца матрицы, а для вектора-строки возвращают сумму, произведение, среднее значение и стандартное квадратичное отклонение соответственно.

Для матрицы функция *sort* возвращает матрицу, у которой отсортированы в порядке возрастания элементы каждого столбца, а для вектора-строки – упорядочивает все его элементы, например:

```
>> A=[10 2 11; 23 1 0; 1 2 3]; B=sort(A)
     B =
           1     1     0
          10     2     3
                23        2     11
>> A=[10 2 11]; B=sort(A)
      B =
            2    10    11
```

Матричная функция *find* (*A*) возвращает индексы ненулевых элементов матрицы A, например:

```
>>A=[1 0 3; 0 0 4; 2 2 2]; find(A)
    ans =
              1
              3
              6
              7
              8
              9
```

Если в скобках указано логическое условие, то с помощью этой функции осуществляется поиск элементов матрицы, удовлетворяющих этому условию. При этом возвращаются индексы найденных элементов, например:

```
>> k=find(A>2)
   k =
         7
         8
```

```
>> k=find(A==0)
   k =
         2
         4
         5
```

ПРАКТИЧЕСКАЯ РАБОТА 3
РАБОТА С ГРАФИЧЕСКИМИ СРЕДСТВАМИ MATLAB

Цель работы: получить навыки работы с двумерными и трехмерными графиками; освоить приемы оформления графиков.

Построение графиков функций осуществляется с использованием специальных команд:

Команда	Описание
plot	Построение двумерного графика в декартовой системе координат
plot3	Построение трехмерного графика в декартовой системе координат
loglog	Построение двумерного графика в логарифмическом масштабе
semilogx	Построение двумерного графика в логарифмическом масштабе по x
semilogy	Построение двумерного графика в логарифмическом масштабе по y

Построение двумерного графика зависимости вектора y от вектора x осуществляется посредством функции plot(x, y), например:

```
>> x = 0 : 10;
>> y = x.^2;
>> plot(x, y)
```

При этом построение графика производится в отдельном окне Windows (рис. 3.1).

Команда *plot*(*x*, *y*, *s*) позволяет при помощи строкового параметра *s* указать способ отображения точек графика. Возможные значения элементов параметра *s* указаны в таблице.

Например (рис. 3.2):

`>> plot(x,y,'r')` – нарисовать красным цветом (линия сплошная без точек);

`>> plot(x,y,'k*')` – нарисовать черным цветом звездочками (линии нет);

`>> plot(x,y,'g--')` – нарисовать зеленым цветом пунктирной линией (точек нет);

`>> plot(x,y,'y-.h')` – нарисовать желтым цветом штрих-пунктирной линией 6-угольными звездочками.

Команда *plot* (*x1*, *y1*, *s1*, *x2*, *y2*, *s2*, *x3*, *y3*, *s3*, ...) позволяет объединить на одном графике несколько функций $y_1(x_1)$, $y_2(x_2)$, $y_3(x_3)$, ..., определив для каждой из них свой способ отображения посредством строковых переменных s_1, s_2, s_3, ..., например:

```
>> x = [1:10]; plot(x, sqrt(x), 'r', x, x.^2, 'k')
```

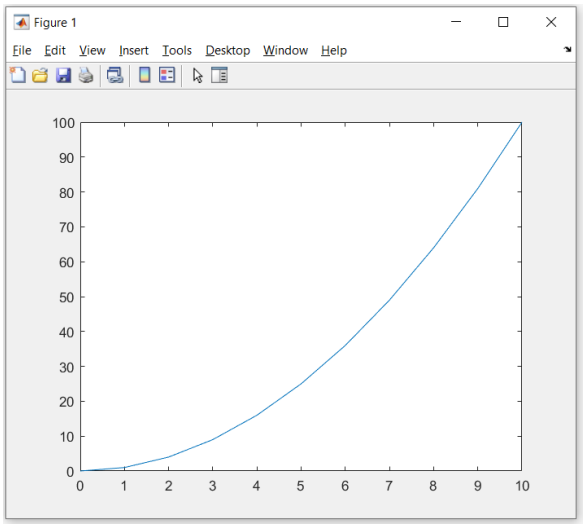


Рис. 3.1. Вид графического окна MatLab

Цвет		Тип точки		Тип линии	
y	– желтый	.	– точка	-	– сплошная линия
m	– сиреневый	o	– кружок	--	– пунктирная линия
c	– голубой	x	– крестик	-.	– штрих-пунктирная линия
r	– красный	*	– звездочка	:	– штриховая линия
b	– синий	+	– плюс		
g	– зеленый	s	– квадрат		
w	– белый	d	– ромб		
k	– черный	v	– треугольник (вниз)		
		^	– треугольник (вверх)		
		<	– треугольник (влево)		
		>	– треугольник (вправо)		
		p	– 5-угольные звездочки		
		h	– 6-угольные звездочки		

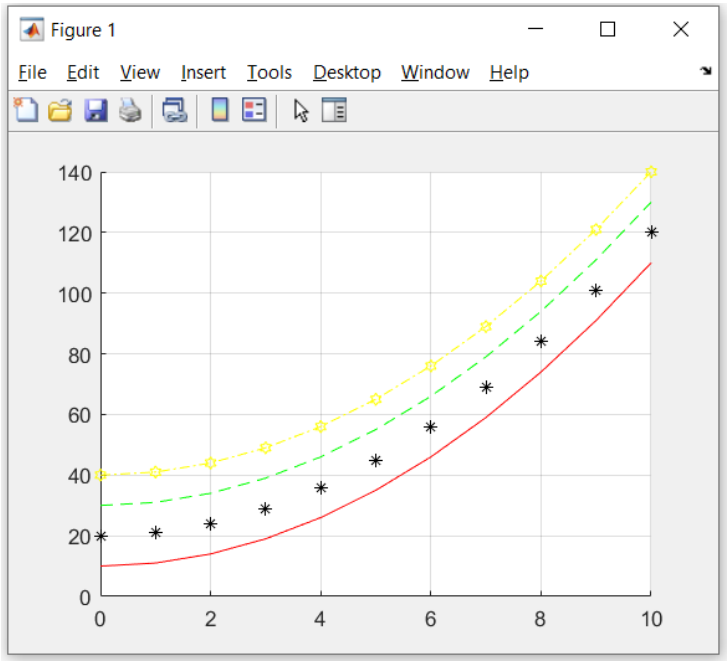


Рис. 3.2. Способы отображения точек графика

Аналогичную функцию выполняет и команда *hold on*, которая включает режим сохранения текущего графика и обеспечивает добавление к нему новых графиков.

Команда *hold off* выключает режим накопления графиков.

Для окончательного оформления графика служат следующие операторы.

Оператор *text*(*x*, *y*, *'текст'*) выводит надпись «текст» в точке, определяемой координатами x и y.

Оператор *title* (*'Заголовок'*) выводит текст «Заголовок» вверху графического окна в качестве заголовка. Текст автоматически центрируется в строке.

Операторы *xlabel*(*'Ось X'*) и *ylabel*(*'Ось Y'*) обеспечивают вывод на графике названия осей «Ось X» и «Ось Y».

Команда *grid on* наносит пунктирную координатную сетку на текущие оси, а команда *grid off* – удаляет координатную сетку.

Оператор *legend*(*'текст1'*, *'текст2'*, *'текст3'*, ...) добавляет к графику легенду (пояснение), в котором параметры *'текст1'*, *'текст2'*, *'текст3'*, ... являются поясняющими надписями к зависимостям, представленным на общем графике.

По умолчанию при построении графиков используется автоматическое масштабирование осей координат. Принудительное задание масштабов по осям X и Y обеспечивает оператор *axis* (*[x_min x_max y_min y_max]*), где вектор из четырех параметров определяет минимальные и максимальные значения по X и по Y. Оператор *axis* (*'square'*) устанавливает равные масштабы по обеим осям.

Для очистки экрана используются функции: *clc* – очистка текстового экрана от предыдущих вычислений и *clf* – очистка графического экрана от ранее построенных графиков.

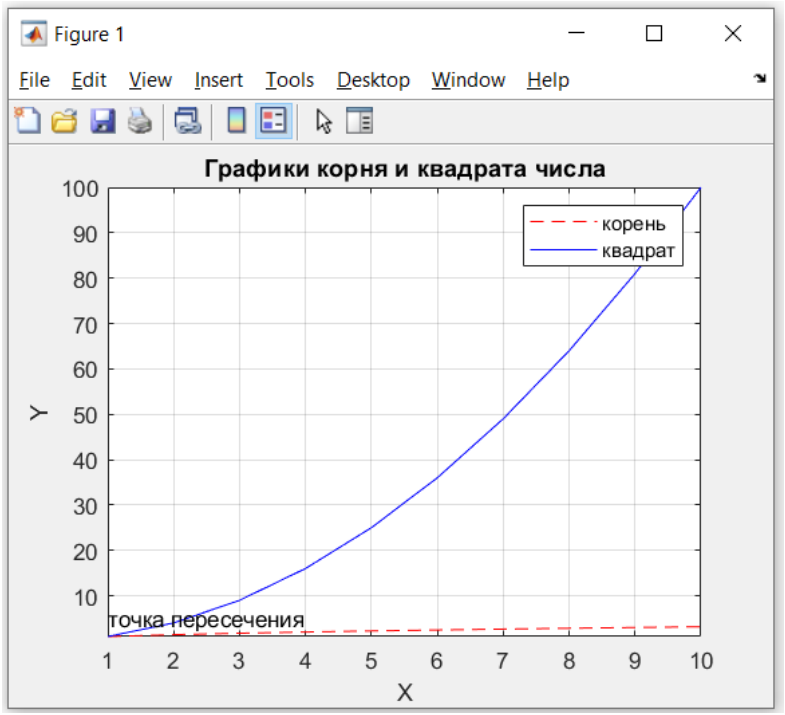


Рис. 3.3. Пример оформления графика

Пример построения графиков функций $y_1 = \sqrt{x}$ и $y_2 = x^2$ (рис. 3.3).

```
>> clf
>> x=[1:10];plot(x,sqrt(x), 'r--')
>> hold on
>> axis([1 10 1 100])
>> plot(x,x.^2, 'b-')
>> grid  on
>> title('Графики корня и квадрата числа')
>> xlabel('X');ylabel('Y')
>> legend('корень', 'квадрат')
>> text(1,5,'точка пересечения')
>> hold off
```

Если вместо функции *plot* (*x*, *y*, *s*) использовать функции *loglog* (*x*, *y*, *s*), то построение двумерного графика будет производиться в логарифмическом масштабе и по оси X, и по оси Y. Например:

```
>> x=1:0.5:100; y=x.^2; loglog (x, y, 'w+')
```

Построение графика в полулогарифмическом масштабе производится функциями *semilogx* (*x*, *y*, *s*) (log по оси X) и *semilogy* (*x*, *y*, *s*) (log по оси Y).

Для построения графика функции в полярных координатах применяется функция *polar* (*phi*, *ro*, *s*), где *phi* – угол в радианах, r_o – полярный радиус, например (рис. 3.4).

```
>> clf; phi=[0:pi/100:2*pi];
>> polar (phi, 1-(sin(phi).^2), 'r')
```

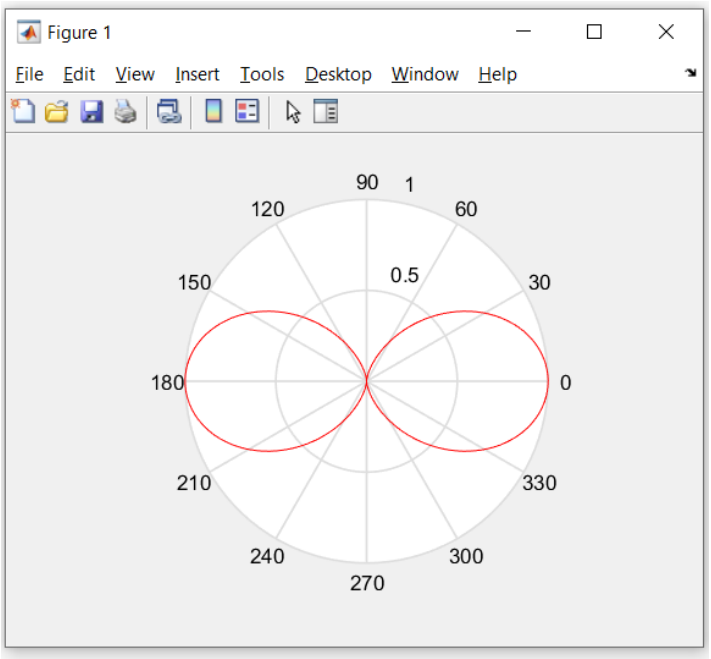


Рис. 3.4. Пример построения графика в полярных координатах

Для построения гистограмм в MatLab можно воспользоваться оператором *bar* (*x*). Он строит прямоугольники, высота которых пропорциональна значениям компонент вектора x, а их число определяется числом компонент этого вектора.

MatLab предоставляет возможность разбиения графического экрана на несколько частей и вывода в каждой из них отдельного графика или гистограммы. Для этого применяется команда *subplot* (*m*, *n*, *p*), где *m* определяет, на сколько частей разбивается экран по вертикали, *n* – по горизонтали, *p* – номер окна, куда будет выводиться очередной график, например (рис. 3.5):

```
>> clc;   clf
>> x=[1 : 10];
>> subplot (2,  1,  1); plot (x,   sqrt(x))
>> subplot (2,  1,  2); bar (x)
```

Таким образом, команда *subplot* (*m*, *n*, *p*) осуществляет переключение между частями экрана. Так, например, чтобы вывести координатную сетку в первой части экрана, необходимо выполнить следующие команды:

```
>> subplot (2,  1,  1);  grid  on
```

при этом во второй части экрана сетка не появится.

При работе с графическими средствами в программе MATLAB специализированные функции помещают графики, а также специальные знаки и надписи на них в отдельные графические окна. Типичный вид такого окна представлен на рис. 3.6.

Кнопки управления видом графического окна и их назначение представлены на рис. 3.7.

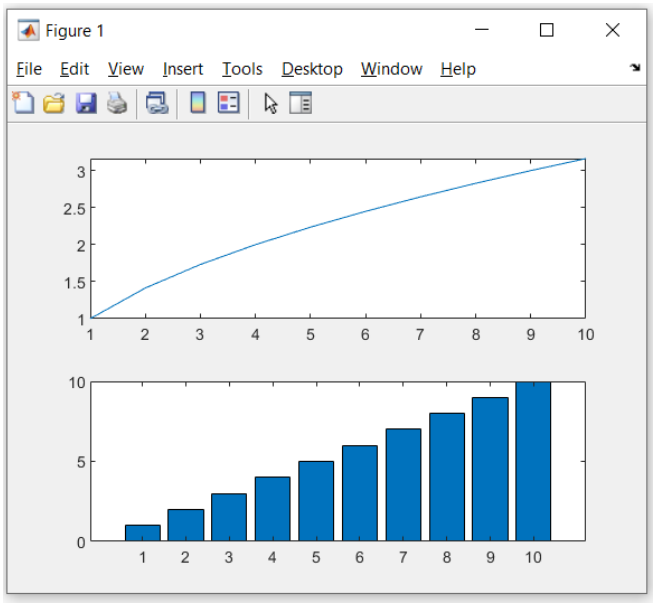


Рис. 3.5. Пример использования команды subplot

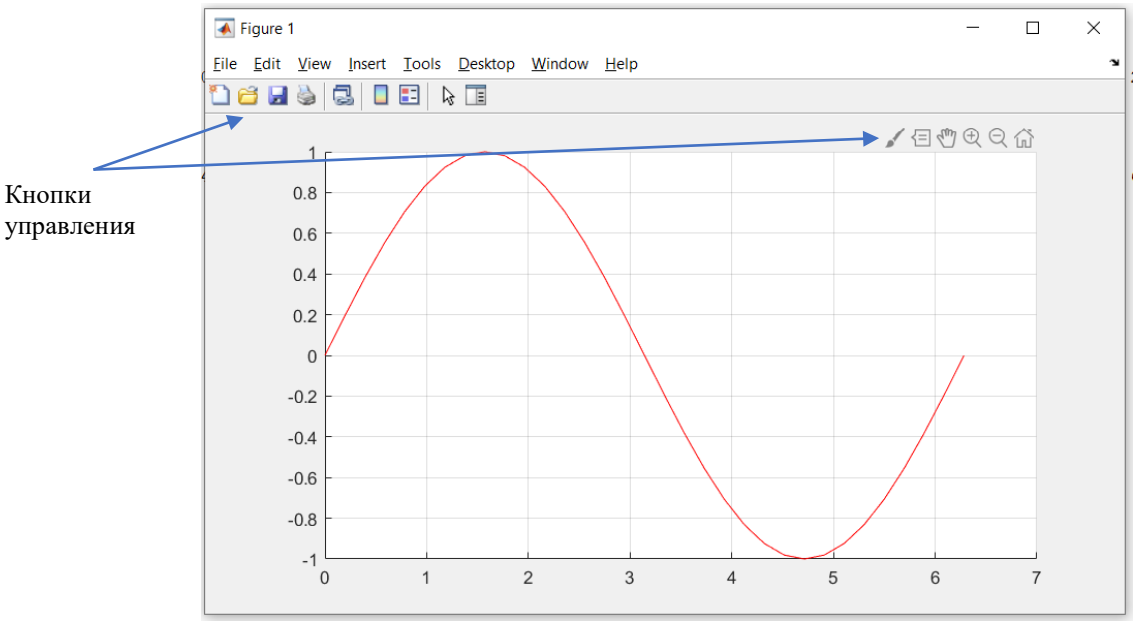


Рис. 3.6. Вид графического окна MATLAB

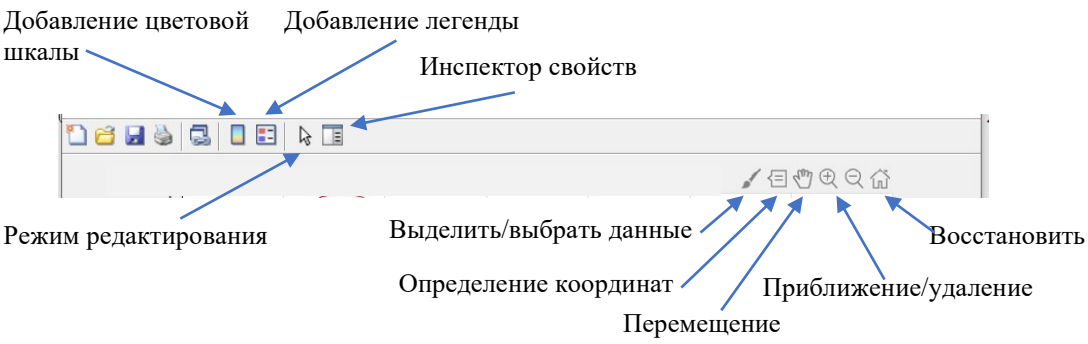


Рис. 3.7. Кнопки управления видом графического окна

Программа MatLab позволяет редактировать вид графического окна с помощью кнопок управления графическим окном. Кроме того, можно строить различные графики в т.н. режиме реального времени. Для задания этого режима можно ввести команду *plottools*.

Интерфейс режима управления графиками включает три панели.

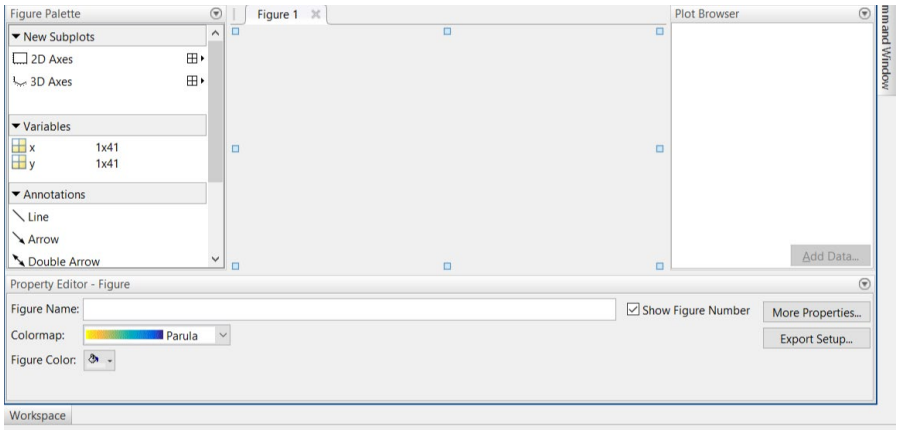
1. *Палитра графиков* (*Figure Palette*) – применяется для редактирования осевых линий графиков, вида рабочей области графика, а также для добавления пояснений. Для отображения палитры графиков можно также использовать специальную команду figurepalette.
2. *Браузер графиков* (*Plot Browser*) – применяется для выбора и управления графиками. Также можно добавить данные путем нажатия кнопки Add Data. Для вызова броузера графиков можно использовать команду plotbrowser.
3. *Редактор свойств* (*Property Editor*) – применяется для установления одинаковых свойств различным объектам. Для вызова редактора можно применить команду *propertyeditor*.

Рассмотрим алгоритм построения графика в интерактивном режиме.

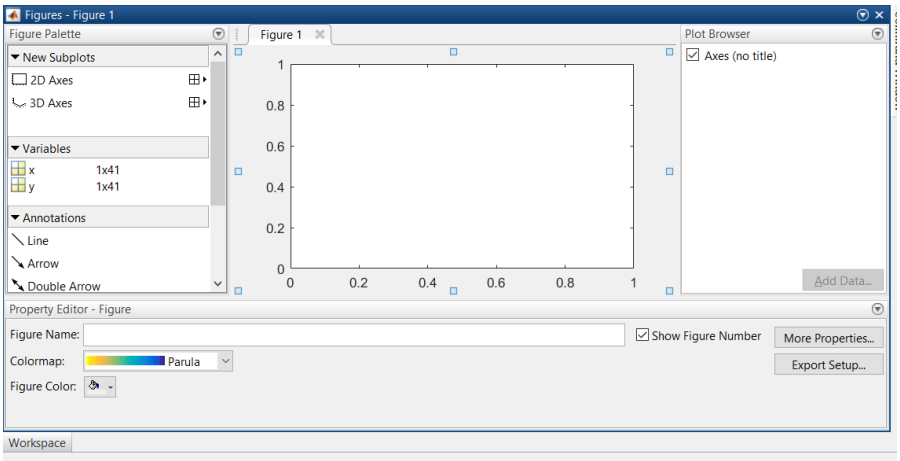
1. Зададим два вектора, используя команды:

```
x= -pi : pi/20:pi;
y= sin(x);
```

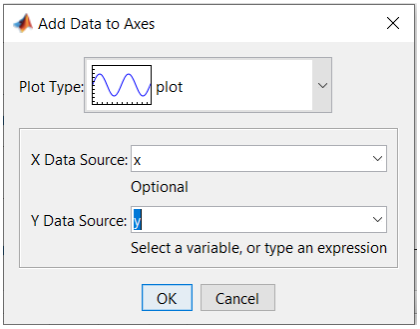
2. Задаем команду *plottools* в результате чего появится окно:



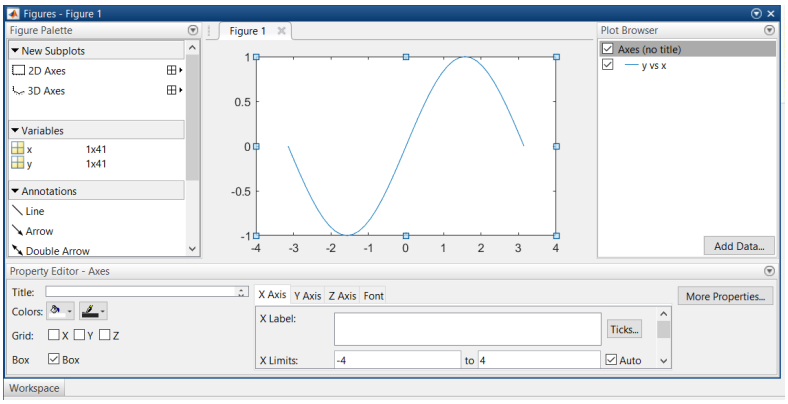
3. Выбираем 2-мерный вид графика на палитре графиков, нажав кнопку 2D Axes.



4. Нажимаем кнопку Add Data в окне браузера графиков и выбираем источники данных так, как показано на рисунке.



После нажатия кнопки OK появится график, представленный на рисунке:



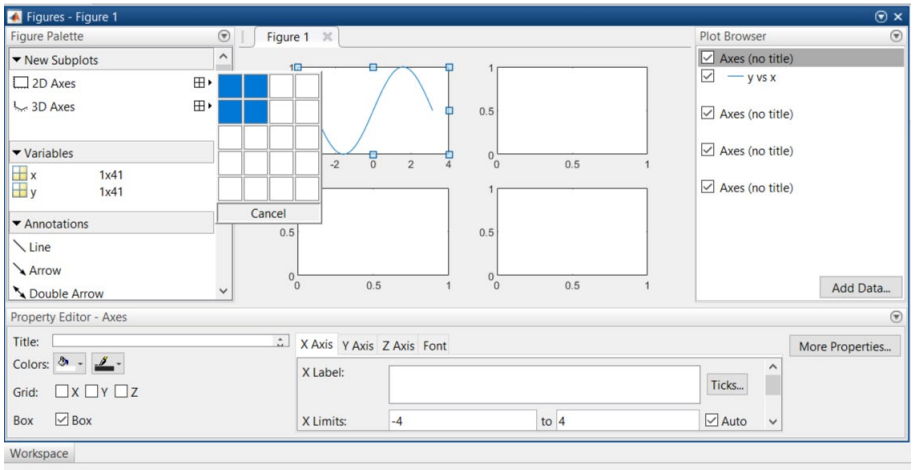
Рассмотрим более подробно палитры.

Палитра графиков (Figure Palette)

Палитра графиков включает три панели, позволяющих выполнять следующие задачи:

1. *Новые графики* (*New Subplots*) – добавлять окна для построения 2-3-мерных графиков.
2. *Переменные* (*Variables*) – изменять и строить переменные рабочей области.
3. *Аннотации* (*Annotations*) – добавлять поясняющие надписи на графики.

На следующем рисунке показан пример добавления 4-х окон для построения в них двумерных графиков:

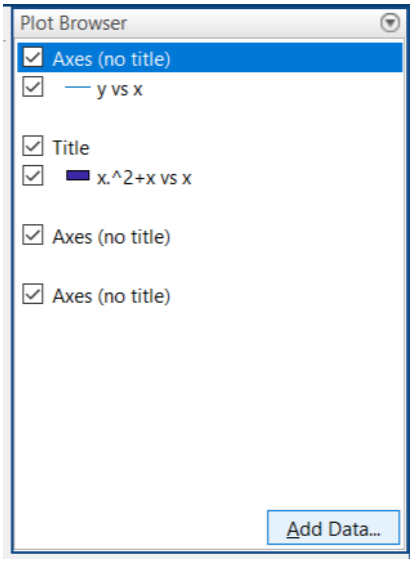


На панели *Variables* отражаются текущие массивы данных, использующиеся для построения графиков. На рисунке выше в первом окне построен график функции $y = \sin(x)$. При этом на панели *Variables* показаны две переменных x и y, размерность которых одна строка, 41 столбец.

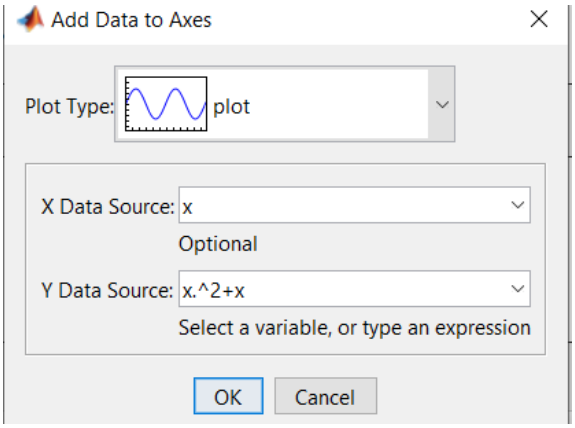
Браузер графиков (Plot Browser)

Обеспечивает возможности добавления и отображения легенды ко всем графикам. Кроме этого, можно управлять видимостью графиков в окне.

Например, для того, чтобы график функции $z = f(x)$ не отображался, достаточно убрать зеленую галочку напротив ----z. Также можно добавлять новые данные путем нажатия кнопки *Add Data*.

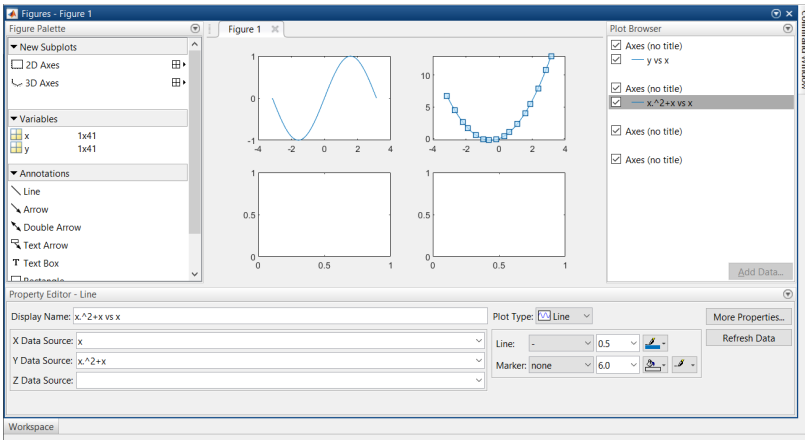


Можно добавлять новый график, используя не массив исходных данных, а выражение, записываемое в диалоговое окно. Например, на следующем рисунке строится график функции $x + x^2$.

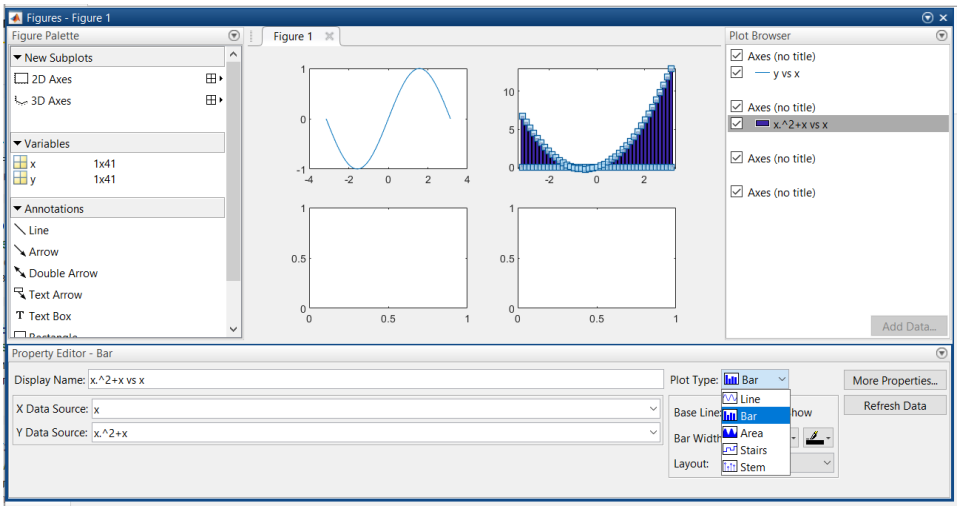


Редактор свойств

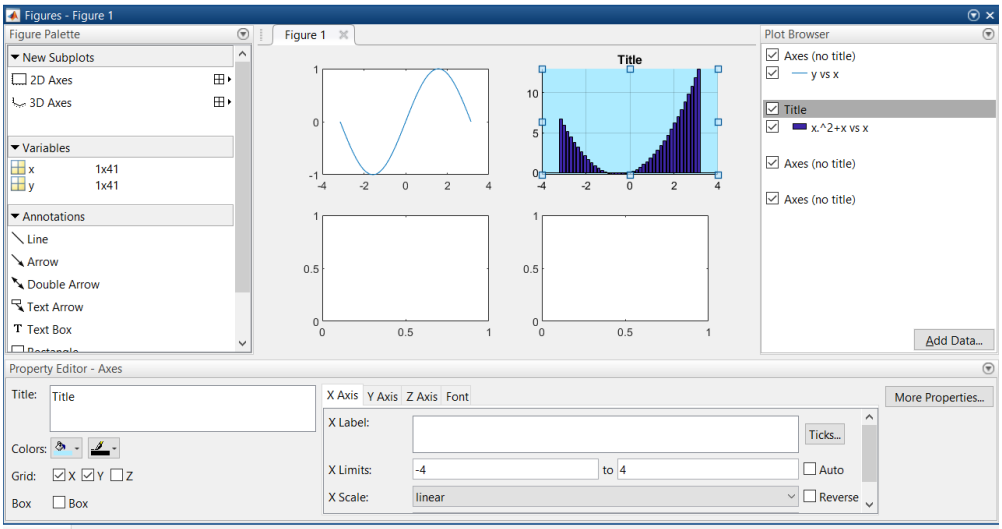
Используется для изменения типа графиков, добавления заголовка, подписи осей и т.д. Выделим график функции, при этом окно редактора свойств примет вид, изображенный на следующем рисунке.



Поменяем тип графика (Plot type) так, как показано на рисунке ниже.



Если выделить рабочую зону графика, то в окне редактора свойств можно добавить заголовок к графику, линии сетки, а также изменить подписи осей:



Построение трехмерных графиков

Для построения трехмерных графиков функций вида $z = f(x, y)$ сначала необходимо задать диапазоны изменения координат x и y, а затем определить сетку, представляющую собой совокупность всех возможных парных сочета-

ний значений x и y. Это осуществляется при помощи функции meshgrid. Например, пусть переменные $x1$ и $y1$ принимают значения 0 или 1, тогда в результате мы получим сетку из четырех пар значений:

```
>> x1=[0 1]; y1=[0 1];
>> [x, y]=meshgrid(x1, y1)
     x =
           0     1
           0     1
     y =
           0     0
           1     1
```

После этого на полученной координатной сетке рассчитывается функция $z = f(x, y)$, а окончательный вывод трехмерного графика производится функцией *mesh* (x, y, z). Например (рис. 3.8), требуется построить график функции $z = x^2 - y^2$, в области изменения $x = [-2; 2]$ и $y = [-2; 2]$.

```
>> x1=[-2:0.1:2]; y1=[-2:0.1:2];
>> [x, y]=meshgrid (x1, y1);
>> z=x.^2-y.^2;
>> mesh (x, y, z)
```

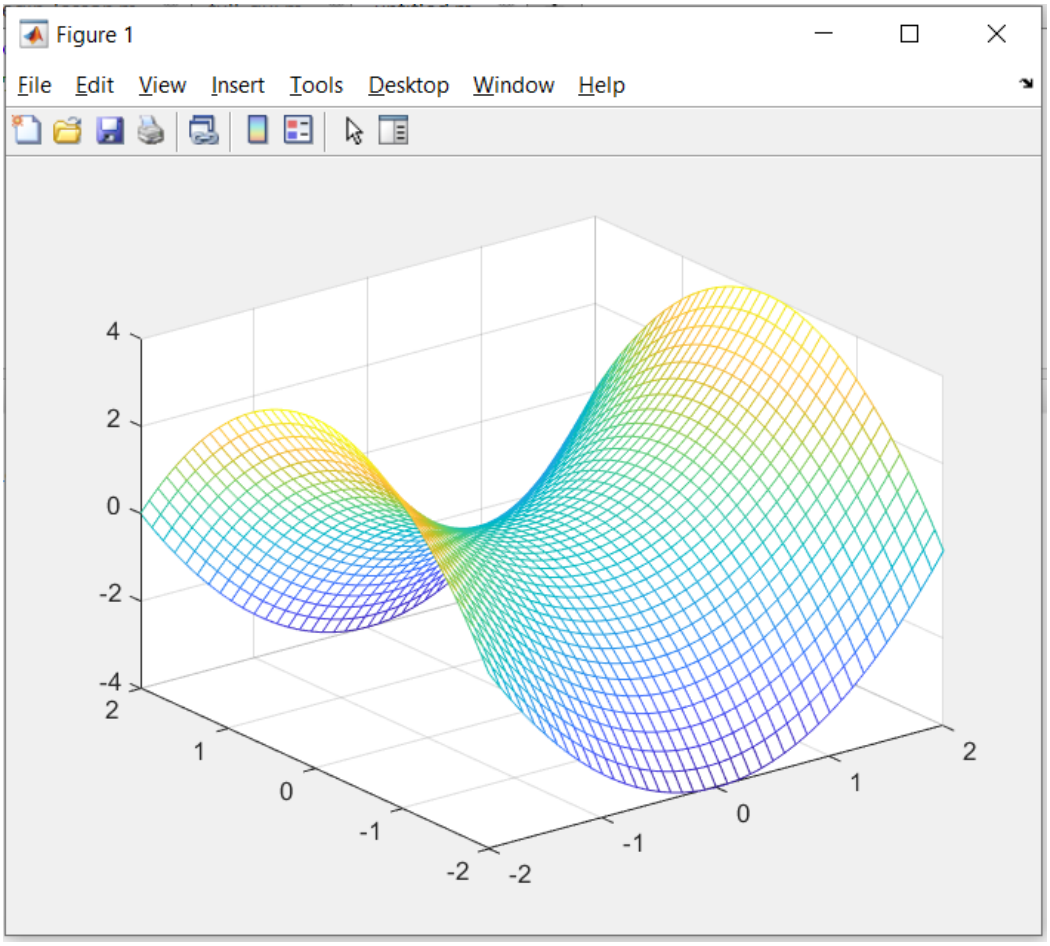


Рис. 3.8. Пример построения трехмерного графика

Поворот построенного трехмерного графика осуществляется при помощи оператора *view*(*az*, *el*), где *az* – угол горизонтального поворота (по часовой стрелке), *el* – угол вертикального возвышения (оба – в градусах.)

По умолчанию *az* = -37.5, el = 30. Для возврата к этому режиму используется оператор view(3).

Поскольку трехмерный график не имеет одного определенного цвета, то в нем производится изменение целиком цветовой гаммы. Для этого применяется оператор *colormap*(*map*), где *map* – имя конкретной цветовой гаммы. В MatLab возможны следующие значения параметра map: *hsv*, *hot*, *gray*, *bone*, *copper*, *pink*, *white*, *flag*, *lines*, *colorcube*, *jet*, *prism*, *cool*, *autumn*, *spring*, *winter*, *summer*. Например,

```
>> colormap(gray)
```

Оператор *mesh* (*x*, *y*, *z*) строит трехмерную поверхность в виде прозрачной сетки, а аналогичный ему оператор *surf* (*x*, *y*, *z*) – в виде непрозрачной сетки. Для построения контурных линий трехмерных поверхностей, часто именуемых линиями уровня, используется оператор *contou r*(*x*, *y*, *z*). Построение контурных линий особенно полезно для многоэкстремальных функций. Обычный трехмерный график в этом случае иногда не нагляден, так как один пик фигуры может скрывать другой, и он будет невидим. Совмещение контурных линий с самой поверхностью производится оператором *meshc* (*x*, *y*, *z*). Пример использования этих четырех операторов для одной функции (см. рис. 3.8) приведен на рис. 3.9.

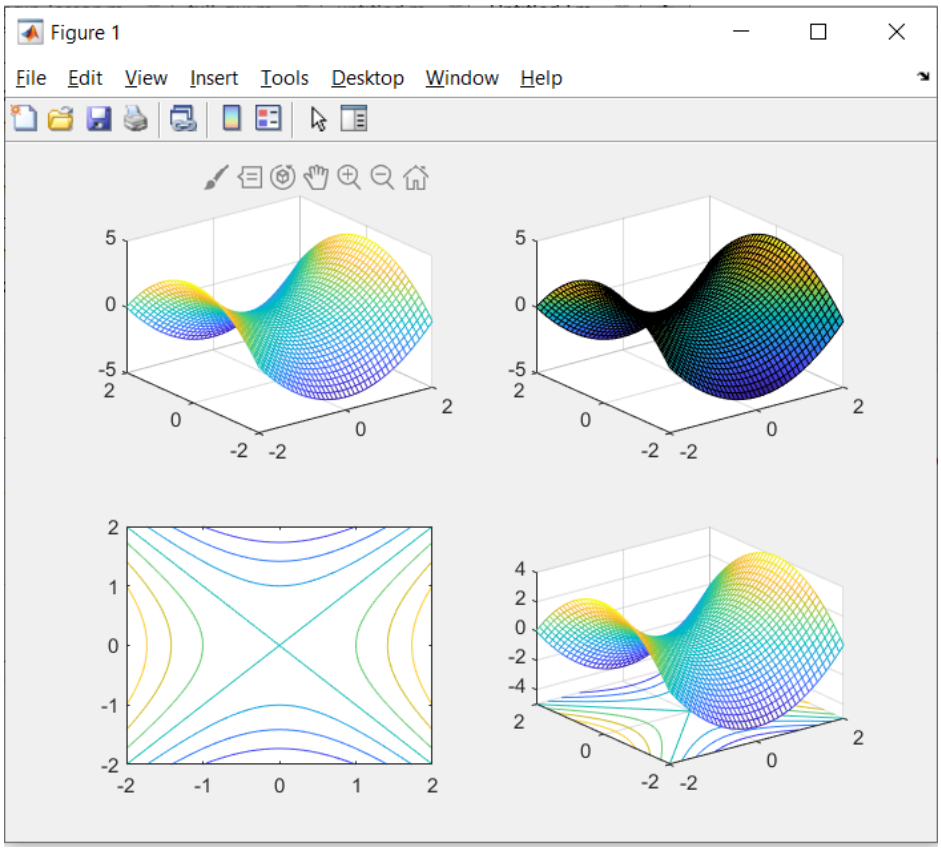


Рис. 3.9. Графики функции $z = x^2 - y^2$, построенные при помощи операторов *mesh*, *surf*, *contour* и *meshc*

Пример. Пусть требуется построить график поверхности, направляющая вычисляется по формуле $r = b + a*\cos(phi)$, а образующая равна 4.

Пример программы и результат ее выполнения (рис. 3.10) представлен ниже.

```
phi=[0:pi/10:2*pi];
a=1;
b=1;
r=b+a*cos(phi);
h=4;
x=r.*cos(phi);
y=r.*sin(phi);
step=10;
delta=[-h/2:h/step:h/2]';
X=ones(size(delta,1),1)*x;
Y=ones(size(delta,1),1)*y;
z=h/2*ones(1,size(x,2));
Z=delta*z;
surf(X,Y,Z)
xlabel('x')
ylabel('y')
zlabel('z')
```

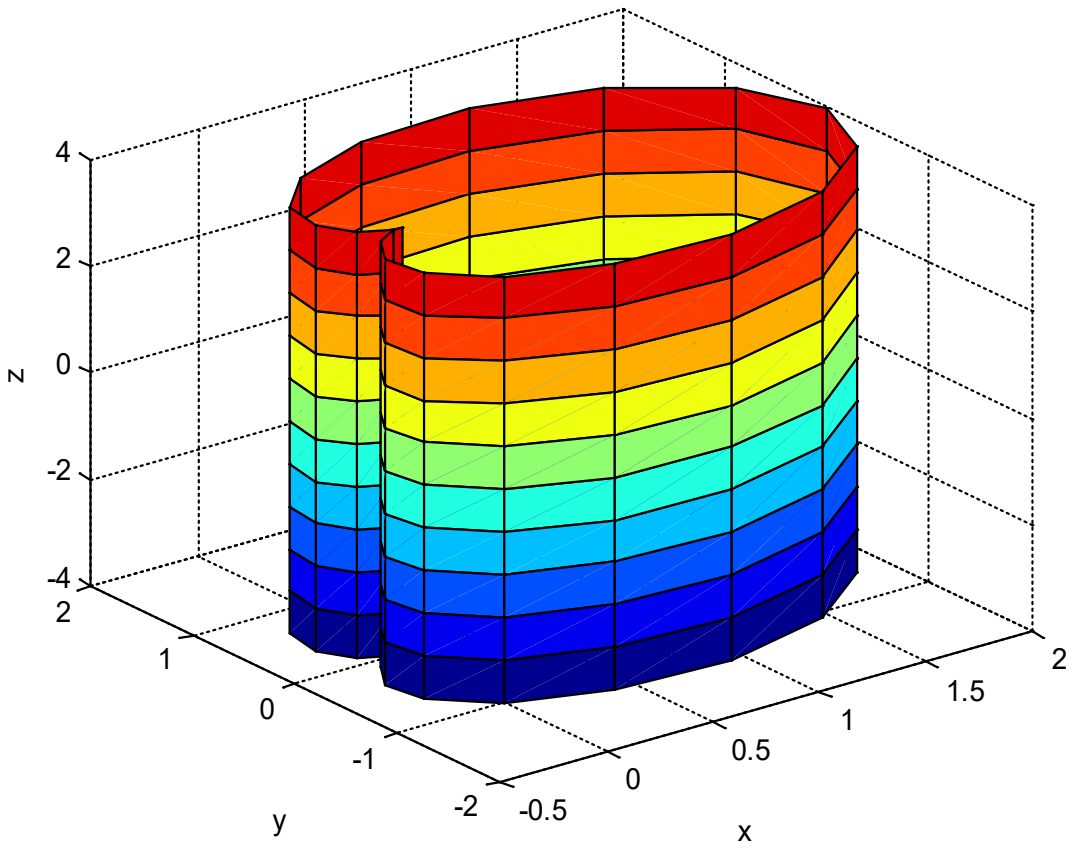


Рис. 3.10. Результат построения

ПРАКТИЧЕСКАЯ РАБОТА 4
ПРОГРАММИРОВАНИЕ В СИСТЕМЕ MATLAB

Цель работы: получить практические навыки разработки и программирования вычислительного процесса линейных, разветвляющихся и циклических структур в системе Matlab; получить навыки по отладке и тестированию программ.

Операторы и циклы

Программа в MatLab представляет собой запись в m-файле (файл с расширением «.m») всей той последовательности команд, которые потребовались бы для решения задачи в режиме прямых вычислений. Запуск на выполнение m-файла осуществляется введением в строке его имени без расширения и нажатием клавиши «Enter».

Ввод данных с клавиатуры в программе организуется при помощи оператора *input*, а вывод на дисплей – disp, например:

```
r=input ('Введите r');
disp(r)
```

Кроме того, управление выводом информации на экран может осуществляться посредством символа «;».

В системе MatLab оператор ветвления имеет следующий сокращенный вид:

```
if Условие
    Операторы
end
```

а в общем случае записывается следующим образом:

```
if Условие
    Операторы1
else
    Операторы2
end
```

В первом варианте «Операторы» выполняются только в том случае, если условие истинно, и не выполняются если ложно. Во втором варианте если условие истинно, то выполняется группа операторов «Операторы1», в противном случае, выполняется группа операторов «Операторы2».

Для организации циклов в программах используются операторы *for...end* (со счетчиком) и *while...end* (с предварительной проверкой условия). Первый из них записывается следующим образом:

```
for Счетчик=Нач_знач:Шаг:Кон_знач
      Тело цикла (операторы)
end
```

где *Счетчик* – имя переменной параметра цикла (не обязательно целочисленный тип), *Нач_знач и Кон_знач* – начальное и конечное значение переменной цикла, *Шаг* задает величину приращения переменной Счетчик при ее изменении от *Нач_знач* до *Кон_знач*.

Например, задать случайным образом вектор X из 10 элементов и подсчитать в нем число положительных элементов.

```
X=rand(1, 10)-0.5
n=0;
for i=1:1:10
  if X(i)>0
    n=n+1;
  end
end
disp (n)
```

Конструкция цикла *while...end* имеет вид:

```
while Условие
  Тело цикла (операторы)
end
```

В данном случае операторы цикла будут выполняться, пока будет истинно условие «*Условие*».

Создание функций

Центральным моментом программирования в среде MatLab является задание новых функций. Для этого используется конструкция:

```
function Имя_переменной=Имя_функции (Список_аргументов)
% Комментарий, выводимый при использовании команды help

Имя_переменной=Выражение
```

Все переменные, используемые внутри блока функции, **являются локальными**, т.е. область их действия распространяется только на блок функции.

Для того, чтобы создать свою функцию, можно войти в любой текстовый редактор, ввести весь текст с описанием функции и сохранить его в файле с именем «Имя_функции.m». Например, создав запись:

```
function y=func(x);
  y=sqrt(2*x+1);
```

и сохранив ее в файле *«func.m»*, мы получаем доступ к этой функции ($y = \sqrt{2x + 1}$), как, скажем, к стандартной функции sin(x), т.е. следующая команда даст результат:

```
>> z=func(4)
     z =
           3
```

Пример. Вычисление факториала числа. В файле с именем «fact.m» создаем запись:

```
function f=fact(x)
% fact(x) – функция вычисления факториала x!
  f=1;
  if x>0
    for i=1:x
      f=f*i;
    end
  end
```

Первая строка с комментарием (%) будет выводиться на экран в результате выполнения команды

```
>> help Имя_функции
```

т.е.

```
>> help fact
     fact(x) – функция вычисления факториала x!
```

Пример определения функции трех переменных $y = \frac{x_1^2}{2} + \frac{x_2^3}{3} + \frac{x_3^3}{4}$.

```
function y=func3(x)
% Файл func3.m
    y=x(1)^2/2+x(2)^3/3+x(3)^4/4;
```

Тогда вызов функции может быть осуществлен следующими способами:

```
>> x=[1 2 3]; func3(x)
     ans =
               23.4167
```

или

```
>> func3([1 2 3])
     ans =
               23.4167
```

Для изменения формата представления действительных чисел используется команда

```
>> format Тип_формата
```

В MatLab доступны следующие виды форматы:

short	– короткое представление в фиксированном формате (4 знака после десятичной запятой)
short e	– короткое представление в экспоненциальном формате (4 знака после запятой и 3 знака порядка)
long	– длинное представление в фиксированном формате (14 знаков после запятой)
long e	– длинное представление в экспоненциальном формате (14 знаков после запятой и 3 знака порядка)

Например,

```
>> format short; x=4/3
     x =
          1.3333
>> format short e; x=4/3
     x =
          1.3333e+000
```

Операции с многочленами

Многочлен, определяемый следующим выражением:

$$P(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0$$

задается в виде вектора, хранящего коэффициенты от a_n до a_0. Для нахождения корней полинома в MatLab используется функция *roots* (*P*), возвращающая вектор, элементы которого являются корнями многочлена, заданного вектором *P*. Например, пусть требуется найти корни уравнения

$$x^5 + 8x^4 + 31x^3 + 80x^2 + 94x + 20 = 0$$

```
>> P=[1 8 31 80 94 20];
>> roots (P)
     ans =
               -1.0000 + 3.0000i
               -1.0000 - 3.0000i
               -3.7321
               -2.0000
               -0.2679
```

Обратная функция *poly* (*R*), наоборот, восстанавливает по корням коэффициенты полинома:

```
>> R = [1 -1];
>> P = P=poly(R)
     P =
           1     0    -1
```

т.е. для корней $x = \pm 1$, мы получили полином $P = x^2 - 1$.

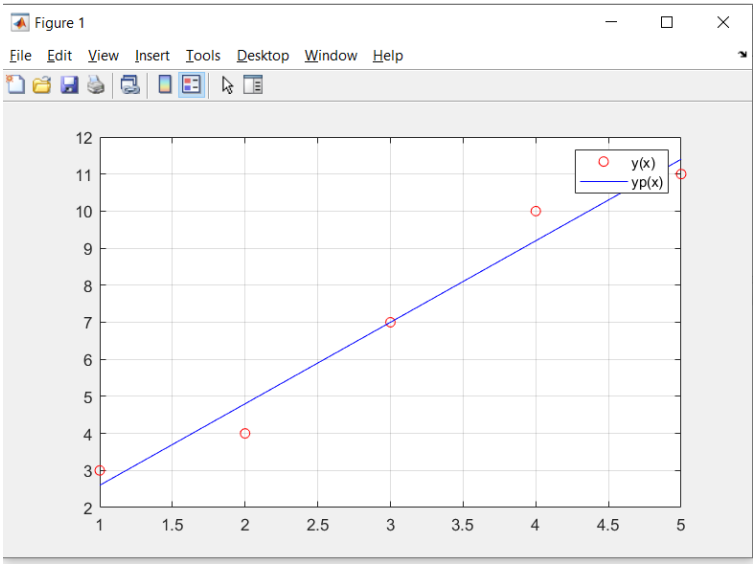


Рис. 4.1. Графики таблично заданной функции $y(x)$ и аппроксимирующей полиномиальной зависимости $y_p(x)$

Для проведения полиномиальной аппроксимации используется функция *polyfit*(*x*, *y*, *n*), которая реализует вычисление коэффициентов полинома *n*-й степени, аппроксимирующего зависимость *y*(*x*). Для вычисления значений аппроксимирующей полиномиальной функции в точках сетки используется функция *polyval*(*P*, *x*), где *P* – вектор коэффициентов аппроксимирующего полинома, x – точки сетки, например (рис. 4.1),

```
>> x = [1 2 3 4 5]; y = [3 4 7 10 11];
>> P = polyfit(x,y,2)
     P =
           0.0000    2.2000    0.4000
>> yp = polyval(P,x)
     yp =
           2.6000    4.8000    7.0000    9.2000   11.4000
>> plot (x, y, 'ro', x, yp, 'b')
```

В результате мы получили уравнение аппроксимирующей полиномиальной зависимости $y_p(x) = a_2x^2 + a_1x + a_0$, где $a_2 = 0; a_1 = 2{,}2; a_0 = 0{,}4$.

Для аппроксимации периодических функций на основе быстрого преобразования Фурье используется оператор interpft (*y*, *n*), возвращающий одномерный массив чисел, который является периодической функцией, определенной в n точках и аппроксимирующей одномерный массив *y*.

Аппроксимация и интерполяция

Интерполяция функции одной переменной осуществляется с помощью оператора *interp1*(*x*, *y*, *xi*, *'способ'*), где *xi* – сетка, на которой производится интерполяция зависимости *y*(*x*), заданной на более крупной сетке. При этом значения *xi*

должны лежать внутри области определения функции y(x). Строковый параметр *'способ'* определяет способ интерполяции. Возможны следующие способы:

'linear' – линейная интерполяция; 'spline' – интерполяция сплайнами; 'cubic' – интерполяция кубическими полиномами.

Пример. Выполнить аппроксимацию функции $y = \sin(x)$, которая задана 10 точками на интервале $[0; 3\pi]$ и провести интерполяцию (кубическую и сплайновую) данной функции на интервале $[\pi/2; \pi]$ (рис. 4.2).

```
clf
x=0:pi/3:3*pi; y=sin(x);
xa=0:pi/10:3*pi;
ya=interpft(y, length(xa));
plot(x, y, 'bo', xa, ya, 'r')
xi=pi/2:pi/10:pi;
yi=interp1(x, y, xi, 'cubic');
hold on
plot (xi, yi, 'k+')
yi=interp1(x, y, xi, 'spline');
plot (xi, yi, 'g')
legend('исход','аппрокс','кубич','сплайн')
grid
hold off
```

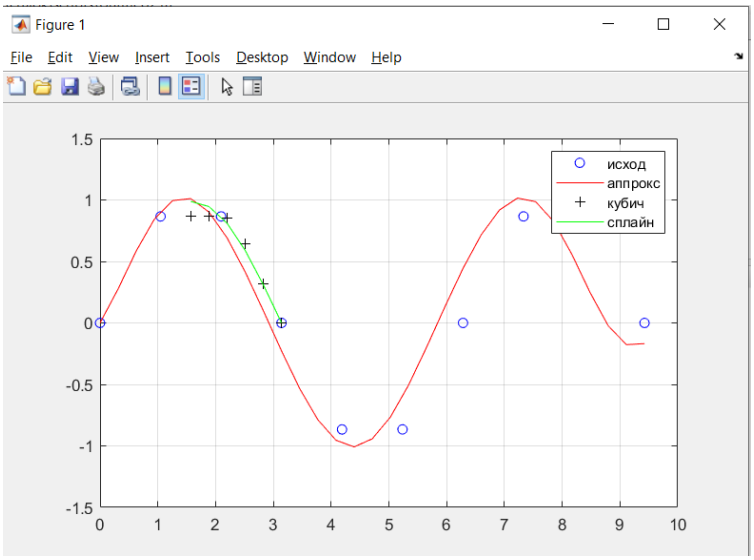


Рис. 4.2. Пример выполнения аппроксимации и интерполяции функции

Численное интегрирование в MatLab

Для выполнения операций численного интегрирования в системе MatLab используются две функции: реализующая метод трапеций – *trapz* и реализующая методом квадратур – *quad*.

Синтаксис функции *trapz* имеет вид:

```
trapz (x, y)
```

где x и y – векторы одинаковой размерности: x – переменная интегрирования, y – значения интегрируемой функции в точках x. Например, требуется определить значение интеграла $\int_0^{\pi} \sin(x)\,dx$. Его точное значение равно двум.

```
>> x=0:pi/100:pi; y=sin(x);
>> I=trapz(x,y)
     I =
                        1.9998
```

Функция вычисления интеграла по методу квадратур имеет следующий вид:

```
quad('имя_функции', ниж_пред, верх_пред, точность)
```

где «*имя_функции*» – имя функции (m-файла), в котором определена интегрируемая функция, «*ниж_пред*» и «*верх_пред*» – нижний и верхний пределы интегрирования, точность – задаваемая относительная погрешность вычисления интеграла (по умолчанию точность равна 0.001).

Например, для предыдущего примера

```
>> I=quad ('sin', 0, pi, 0.01)
     I =
          2.0003
```

Если осуществить вызов функции quad следующим образом:

```
>> [I, n]=quad('sin',0,pi,0.0001)
      I =
            2.0000
      n =
            33
```

то в первой переменной (I) будет возвращено значение интеграла, а во второй (n) – число выполненных итераций, обеспечивших расчет интеграла с заданной точностью.

Поиск минимума и нулей функции

Оператор *fmin* (*'Имя_функции'*, *x_0*, *x_k*) возвращает значение аргумента x из диапазона x_0 ≤ x ≤ x_k, при котором функция одной переменной с именем *«Имя_функции»* достигает своего минимума.

Например, найти минимум функции $y = x^2 - 2x - 3$ на отрезке $x \in [-5; 5]$. Для этого создадим файл с именем «mf.m», в котором описана минимизируемая функция:

```
function y=mf(x)
  y=x.^2-2*x-3;
```

Затем находим точку минимума и значение функции в этой точке

```
>> x_min = fmin ('mf', -5, 5)
     x_min =
                    1
>> y_min = mf (x_min)
     y_min =
                   -4
```

Для нахождения минимума функции нескольких переменных используется оператор *fmins* (*'Имя_функции'*, *X*). Здесь X – точка, в окрестности которой производится поиск.

Пример. Минимизировать функцию двух переменных $y = x_1 e^{-x_1^2 - x_2^2}$ в окрестности точки $x_1 = -2$, $x_2 = -2$. В файле с именем «myfun2.m» определяем минимизируемую функцию, график которой представлен на рис. 4.3:

```
function z=myfun2(x)
z = x(1)* exp(-x(1)^2 - x(2)^2);
```

и находим точку локального минимума *x_new* и значение функции *y_min* в этой точке:

```
>> x_0=[-2 -2];
>> x_min=fmins('myfun2', x_0)
     x_min =
                   -0.7071    0.0000
>> y_min=myfun2(x_min)
     y_min =
                   -0.4289
```

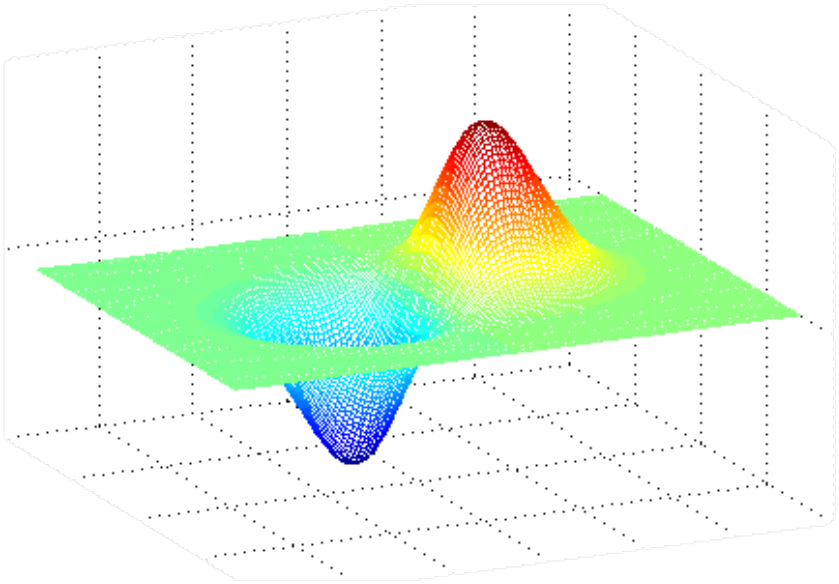


Рис. 4.3. График функции $y = x_1 e^{-x_1^2 - x_2^2}$

Оператор *fzero* (*'Имя_функции'*, *x_0*, *точность*) находит нуль функции одной переменной, заданной именем «Имя_функции» в окрестности точки x_0 с погрешностью, определяемой параметром «точность».

Например, требуется найти решение трансцендентного уравнения $2x^2 - 3x - \sin x - \cos x = 0$. Для этого определяем в файле mft.m функцию, стоящую в левой части исходного уравнения

```
function y=mft(x)
y = 2*x.^2-3*x-sin(x)-cos(x);
```

График данной функции на интервале [–1; 2,5] приведен на рис. 4.4. Найдем нули функции в окрестности точки x=0 и точки x=2.

```
>> x1=fzero('mft', 0, 0.01)
     x1 =
              -0.2263
>> x2=fzero('mft', 2, 0.01)
     x2 =
              1.7260
```

В результате мы нашли два корня уравнения x_1 и x_2, которые хорошо видны на графике (рис. 4.4).

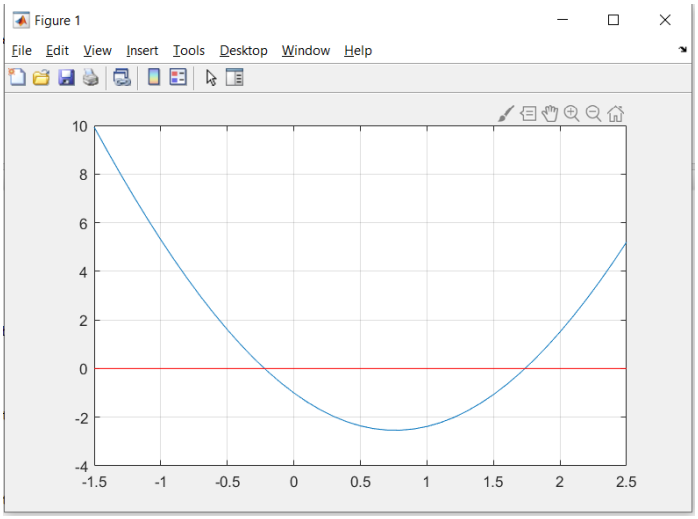


Рис. 4.4. График функции $y = 2x^2 - 3x - \sin x - \cos x$

ПРАКТИЧЕСКАЯ РАБОТА 5
СОЗДАНИЕ ЭЛЕМЕНТОВ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Цель работы: ознакомиться с основными приемами создания графического пользовательского интерфейса в MatLab.

Приложения *MatLab* являются графическими окнами, содержащими элементы управления (кнопки, списки, переключатели, флаги, меню), а также оси и текстовые области для вывода результатов работы.

MatLab создать интерфейс приложения можно двумя способами:

- вручную
- с помощью специального конструктора: для этого нужно в консоли MatLab прописать команду «guide» и выбрать необходимые настройки.

«Ручной» способ создания интерфейса. Основные команды для создания пользовательского интерфейса

Опытные пользователи нередко используют MATLAB для создания своих собственных систем. Этому во многом способствует идеология системы – хранение большей части команд и функций в виде m-файлов. Простота коррекции файлов и отсутствие необходимости явно объявлять создание новых команд и функций привели к появлению множества программных систем на базе MATLAB, особенно в таких областях, как решение задач линейной алгебры, моделирование различных систем и структур и т.д.

В MATLAB дескрипторная графика позволяет конструировать детали пользовательского интерфейса. Полный список команд и функций для проектирования пользовательского интерфейса можно получить, выполнив команду help uitools.

Ниже перечислены все команды и функции данного назначения. Функции пользовательского интерфейса GUI:

ulcontrol – создание управляющего элемента;

ulmenu – создание пользовательского меню;

ginput – графический ввод с помощью мыши.

dragrect – создание выделяющего прямоугольника с помощью мыши;

rbbox – растягивание прямоугольника мышью;

selectmoveresize – интерактивное выделение, перемещение и копирование объектов с помощью мыши;

waitforbuttonpress – ожидание нажатия клавиши клавиатуры или кнопки мыши в окне;

waltfor – прекращение выполнения программы в ожидании уничтожения заданного графического объекта или изменения его свойств;

uiwait – прекращение выполнения программы в ожидании вызова функции uiresume или закрытия заданного графического окна;

inresume – возобновить выполнение после блокировки;

uisuspend – прекращение интерактивного состояния фигуры;

uirestore – возобновление интерактивного состояния фигуры.

Средства проектирования пользовательского интерфейса:

guide – создание GUI;
align – выровнять положение объектов интерфейса;
cbedit – изменение повторного вызова объектов;
menuedit – изменение меню;
propedit – изменение свойств объектов.

Средства создания диалоговых окон:

dialog – создание диалогового окна;
axlimdlg – ограничение размеров диалогового окна;
errordlg – создание окна с сообщением об ошибке;
helpdlg – создание справочного окна;
inputdlg – создание окна диалога ввода;
listdlg – создание окна диалога для выбора вариантов параметра из списка;
menu – создание меню диалогового ввода;
msgbox – создание окна сообщений;
questdlg – создание окна запроса;
warndlg – создание окна предупреждения;
uigetfile – создание стандартного окна открытия файлов;
uiputfile – создание стандартного окна записи файлов;
uisetcolor – создание окна выбора цвета;
uisetfont – создание окна выбора шрифта;
pagedlg – создание диалогового окна параметров страницы;
printdlg – создание диалогового окна печати;
waitbar – создание окна с индикатором прогресса.

Создание меню:

makemenu – создать структуры меню;
menubar – устанавливать типовые свойства для объекта MenuBar;
umtoggle – изменять статус параметра «checked» для объекта uimenu;
winmenu – создать подменю для меню Window.
Создание кнопок панели инструментов и управление ими:
btngroup – создать кнопку панели инструментов;
btnstate – запросить статус кнопки;
btnpress – управление кнопкой;
btndown – нажать кнопку;
btnup – отпустить кнопку.

Утилиты задания свойств объектов figure/axes:

clruprop – удалить свойство объекта; getuprop – запросить свойство объекта; setuprop – установить свойство объекта.

Вспомогательные утилиты:

all child – запросить все порожденные объекты; findall – найти все объекты; hidegui – скрыть/открыть объекты GUI; edtext – интерактивное редактирование объектов text; getstatus – запросить свойства строки объекта figure; setstatus – установить свойства строки объекта figure; popupstr – запросить свойства строки выпадающего меню; remapflg – изменить положение объекта figure; setptr – установить указатель на объект figure; getptr – получить указатель на объект figure; overobj – запросить дескриптор объекта, над которым находится курсор мыши.

Таким образом, MATLAB содержит обширный набор команд и функций для создания типовых элементов пользовательского интерфейса. В данной работе мы ограничимся единственным примером: создание кнопки, на которой можно щелкнуть мышью и перевести ее в нажатое состояние.

Пример создания объекта интерфейса

Ниже представлена программа (распечатка m-файла с именем ui), которая при запуске создает четыре объекта интерфейса:

```
clear
kl=uicontrol('Style','pushbutton','Units','normalized',
'Position',[.6 .5 .2 .1],...
    'String','click here');
k2=uicontrol('Style','pushbutton','Units','normalized',
'Position',[.6 .3 .2 .1],...
    'String','click here');
ck = uicontrol('Style', 'pushbutton', 'String',
'Clear',...
    'Position', [150 150 100 70],'Callback', 'da');
hpop = uicontrol('Style', 'popup','String',
'hsv|hot|cool|gray',...
    'Position', [30 320 100 50],'Callback', 'setmap');
```

Первые два объекта *kl* и *k2* – это малые кнопки с надписью click here («щелкни здесь»). Объект ck – это большая кнопка Clear. Объект hpop – раскрывающийся список (тоже действующий, хотя и содержащий незаполненные поименованные позиции). Для создания всех этих объектов используется команда uicontrol с соответствующими параметрами, задающими стиль (вид) объекта интерфейса, место его размещения и надпись (на кнопках). На рисунке 5.1 построены все эти объекты, причем раскрывающийся список показан в открытом состоянии.

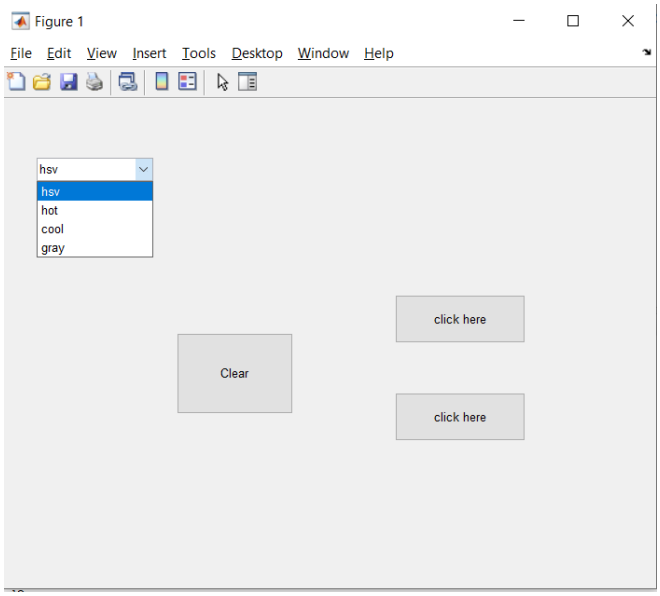


Рис. 5.1. Пример построения объектов пользовательского интерфейса

Дескрипторная графика MATLAB позволяет создавать любые детали современного пользовательского интерфейса.

Создание приложений графического интерфейса в среде GUIDE

После ввода и запуска команды «guide» в Command Window, откроется окно графического редактора:

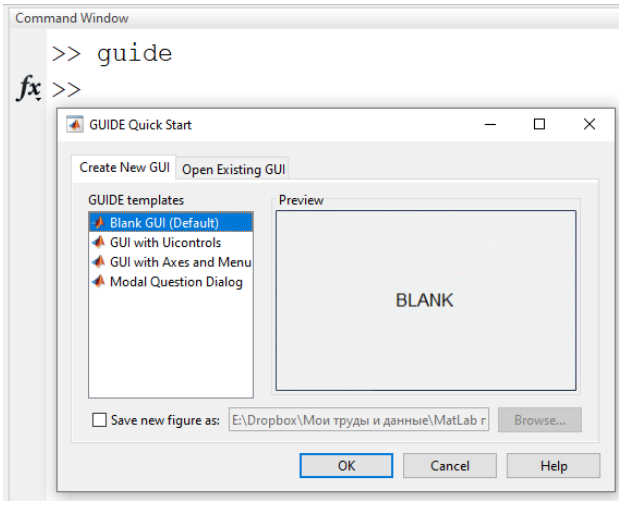


Рис. 5.2. Окно при запуске приложения Guide

Интерфейс редактора после создания нового графического окна:

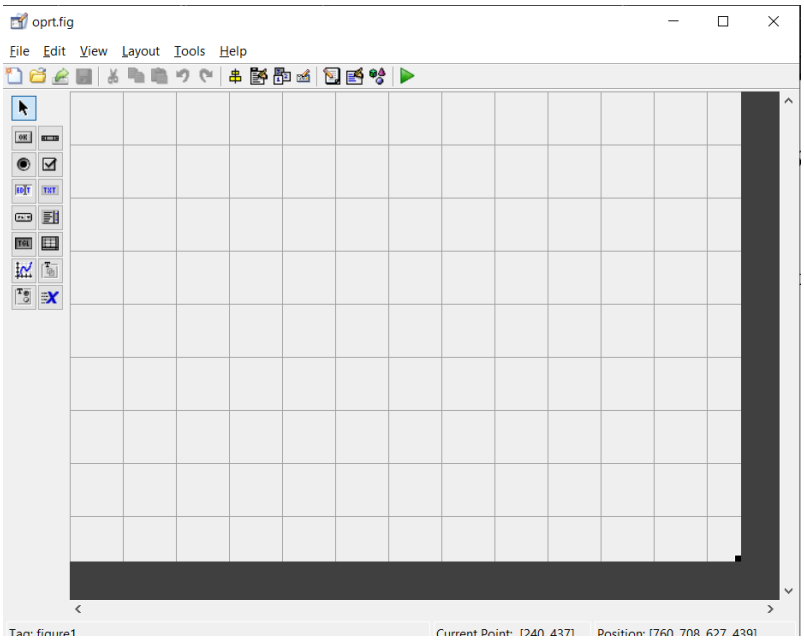


Рис. 5.3. Внешний вид графического редактора интерфейса

Приложение в *MatLab* хранится в двух файлах с расширениями *.fig* и *.m*. Первый хранит информацию о расположении элементов интерфейса, второй – является М-файлом с основной функцией и подфункциями. Добавление элемента приводит к автоматическому добавлению соответствующей подфункции. Данную подфункцию необходимо наполнить операторами, реализующими алгоритм события, возникающего при обращении к элементу интерфейса.

Перечень основных элементов графического окна MATLAB:

1. Кнопка (Button)
2. Слайдер (Slider)
3. Кнопка группового выбора (Radio-button)
4. Кнопка выбора (Check box)
5. Вводимый текст (Edit text)
6. Статический текст (Static text, Label)
7. Выпадающее меню (Pop-up menu)
8. Окно списка (List box)
9. Кнопка с фиксацией (Toggle button)
10. Таблица (Table)
11. График координат (Axes)
12. Панель (Panel)
13. Группировка кнопок (Button group)
14. Активная панель (ActiveX)

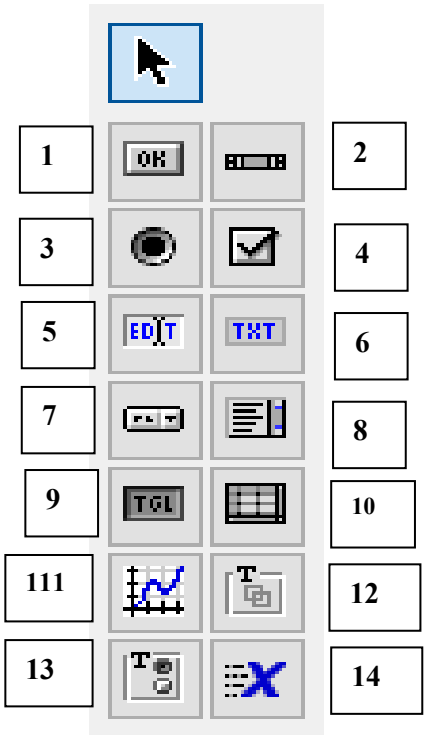


Рис. 5.3. Элементы графического окна MATLAB

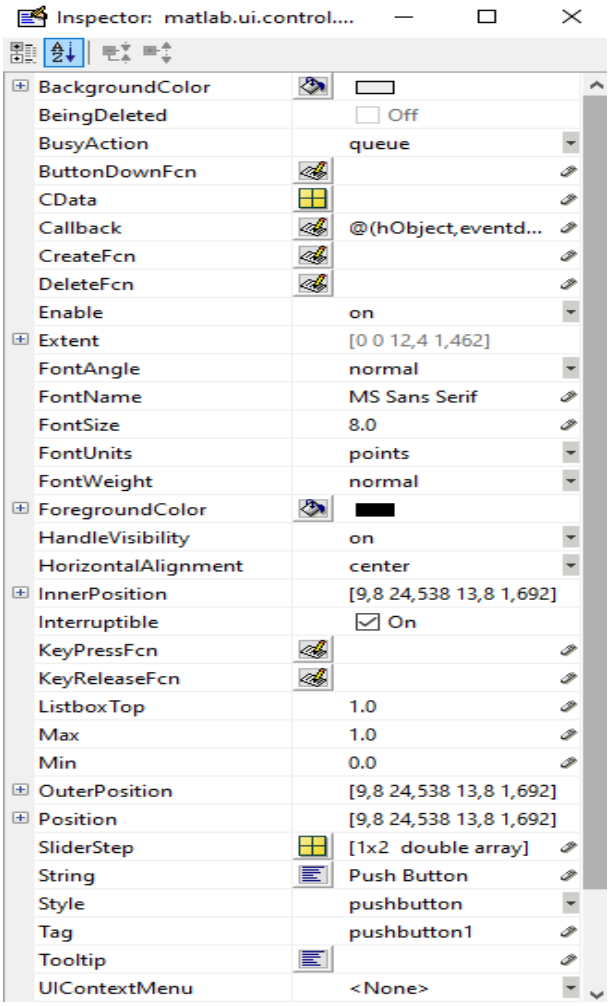


Рис. 5.4. Инспектор свойств

Если два раза кликнуть на элемент, то появится Inspector, который позволяет настраивать разные свойства объектов окна.

Для каждого созданного в конструкторе «виджета», MATLAB автоматически генерирует функции в основном коде, описывающем свойства графического окна (основной код генерируется после нажатия кнопки «Run figure» в графическом редакторе и сохранения файлов интерфейса):

```
function pushbutton1_
Callback(hObject, eventdata,
handles)
```

Пример построения графика функции при нажатии на кнопку:

```
function pushbutton1_
Callback(hObject,
eventdata, handles)
    x=[-2:0.1:2];
    y=exp(-x.^2);
    plot(x,y);
    grid on;
```

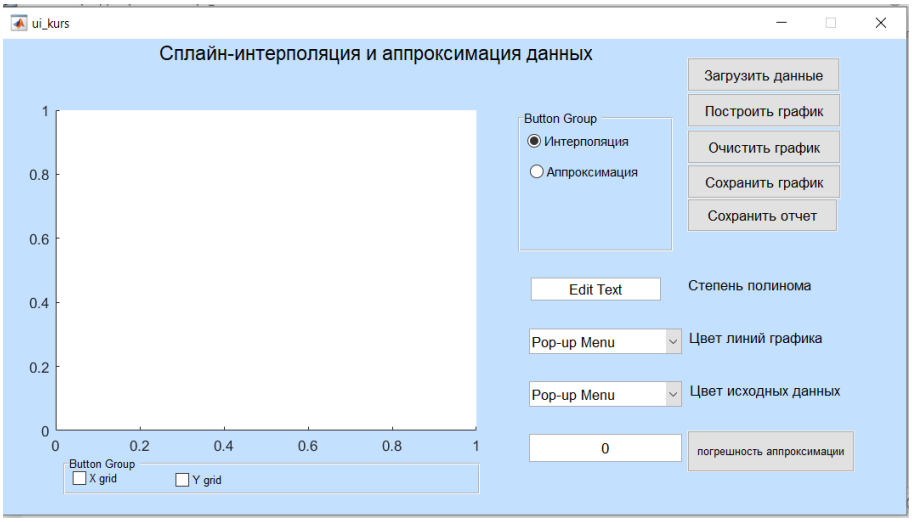


Рис. 5.5. Пример пользовательского интерфейса приложения

ПРАКТИЧЕСКАЯ РАБОТА 6
СТАТИСТИЧЕСКАЯ ОБРАБОТКА ДАННЫХ И ФИЛЬТРАЦИЯ СИГНАЛОВ

Цель работы: изучить следующие вопросы:

- Статистическая обработка массивов
- Сортировка элементов массивов
- Преобразования Фурье
- Свертка и обратная ей операция
- Дискретная фильтрация

Статистическая обработка данных

Нахождение максимального и минимального элементов массива.

Самый простой анализ данных, содержащихся в некотором массиве, заключается в поиске его элементов с максимальным и минимальным значениями. В системе MATLAB определены следующие быстрые функции для нахождения минимальных и максимальных элементов массива:

- max(A) – возвращает наибольший элемент, если A – вектор; или возвращает вектор-строку, содержащую максимальные элементы каждого столбца, если A – матрица, в многомерных массивах работает с первой не единичной размерности;
- max(A.B) – возвращает массив того же размера, что A и B, каждый элемент которого есть максимальный из соответствующих элементов этих массивов;
- max(A.[ ],dim) – возвращает наибольшие элементы по столбцам или по строкам матрицы в зависимости от значения скаляра dim. Например, max(A,[ ],1) возвращает максимальные элементы каждого столбца матрицы A;
- [C.I] =max(A) – кроме максимальных значений возвращает вектор индексов I этих элементов.

Для быстрого нахождения элемента массива с минимальным значением служит следующая функция:

- min(A) – возвращает минимальный элемент, если A – вектор; или возвращает вектор-строку, содержащую минимальные элементы каждого столбца, если A – матрица;
- min(A.B) – возвращает массив того же размера, что A и B, каждый элемент которого есть минимальный из соответствующих элементов этих массивов;
- min(A,[ ],dim) – возвращает наименьший элемент по столбцам или по строкам матрицы в зависимости от значения скаляра dim. Например, min (A,[ ],1) возвращает минимальные элементы каждого столбца матрицы A;
- [C,I] = min(A) – кроме минимальных значений возвращает вектор индексов этих элементов.

Работа указанных функций базируется на сравнении численных значений элементов массива A, что и обеспечивает высокую скорость выполнения операций.

Нахождение средних, срединных значений массива и стандартных отклонений

Элементарная статистическая обработка данных в массиве обычно сводится к нахождению их среднего значения, медианы (срединного значения) и стандартного отклонения. Для этого в системе MATLAB определены следующие функции:

- mean (A) – возвращает арифметическое среднее значение элементов массива, если A – вектор; или возвращает вектор-строку, содержащую средние значения элементов каждого столбца, если A – матрица. Арифметическое среднее значение есть сумма элементов массива, деленная на их число;
- mean(A.dim) – возвращает среднее значение элементов по столбцам или по строкам матрицы в зависимости от значения скаляра dim (dim=l по столбцам и dim=2 по строкам соответственно).
- median (A) – возвращает медиану, если A – вектор; или вектор-строку медиан для каждого столбца, если A – матрица;
- median(A.dim) – возвращает значения медиан для столбцов или строк матрицы в зависимости от значения скаляра dim.
- std(X) – возвращает стандартное отклонение элементов массива, вычисляемое по формуле, если X – вектор. Если X – матрица, то std(X) возвращает вектор-строку, содержащую стандартное отклонение элементов каждого столбца (обратите внимание, что оно отличается от среднеквадратического отклонения);

- std(X.flag) – возвращает то же значение, что и std(X), если flag=0; если flag=l, функция std(X.l) возвращает среднеквадратическое отклонение (квадратный корень из несмещенной дисперсии), вычисляемое по формуле
- std(X.flag.dim) – возвращает стандартное или среднеквадратическое отклонения по рядам (dim=2) или по столбцам(dim=1) матрицы X в зависимости от значения переменной dim.

Функции сортировки элементов массива

Многие операции статистической обработки данных выполняются быстрее и надежнее, если данные предварительно отсортированы. Кроме того, нередко представление данных в отсортированном виде более наглядно и ценно. Ряд функций служит для выполнения сортировки элементов массива. Они представлены ниже.

- sort (A) – в случае одномерного массива A сортирует и возвращает элементы по возрастанию их значений; в случае двумерного массива происходит сортировка и возврат элементов каждого столбца. Допустимы вещественные, комплексные и строковые элементы. Если A принимает комплексные значения, то элементы сначала сортируются по абсолютному значению, а затем, если абсолютные значения равны, по аргументу. Если A включает NaN-элементы, sort помещает их в конец;
- [B. INDEX] = sort(A) – наряду с отсортированным массивом возвращает массив индексов INDEX. Он имеет размер size(A), с помощью этого массива можно восстановить структуру исходного массива.
- sort(A.dim) – для матриц сортирует элементы по столбцам (dim=l) или по рядам в зависимости от значения переменной dim.
- sortrows(A) – выполняет сортировку рядов массива A по возрастанию и возвращает отсортированный массив, который может быть или матрицей, или вектором-столбцом;
- sortrows(A.column) – возвращает матрицу, отсортированную по столбцам, точно указанным в векторе column. Например, sortrows(A,[2 3]) сортирует строки матрицы A сначала по второму столбцу, и затем, если его элементы равны, по третьему;
- [B, index] = sort rows (A) – также возвращает вектор индексов index. Если A – вектор-столбец, то B=A(index). Если A – матрица размера *mxn*, то B=A(index.:).

Вычисление коэффициентов корреляции

Под *корреляцией* понимается взаимосвязь некоторых величин, представленных данными – векторами или матрицами. Общепринятой мерой линейной корреляции является *коэффициент корреляции*. Его близость к единице указы-

вает на высокую степень линейной зависимости. Данный раздел посвящен описанию функции для вычисления коэффициентов корреляции и определения ковариационной матрицы элементов массива. Приведенная ниже функция позволяет вычислить коэффициенты корреляции для входного массива данных.

- corrcoef(X) – возвращает матрицу коэффициентов корреляции для входной матрицы, строки которой рассматриваются как наблюдения, а столбцы – как переменные. Матрица S=corrcoef(X) связана с матрицей ковариаций C=cov(X) следующим соотношением: S(i.j)=C(i.j)/sqrt(C(i.i)C(j.j));
- Функция S = corrcoef (x,y), где x и y – векторы-столбцы, аналогична функции соггсоеЩх у]).

Вычисление матрицы ковариации

Приведенная далее функция позволяет вычислить матрицу ковариации для массива данных.

- cov(x) – возвращает смещенную дисперсию элементов вектора x. Для матрицы, где каждая строка рассматривается как наблюдение, а каждый столбец – как переменная, cov(x) возвращает матрицу ковариации. diag(cov(x» – вектор смещенных дисперсий для каждого столбца и sqrt(diag(cov(x))) – вектор стандартных отклонений.
- Функция C = cov(x.y), где x и y – векторы-столбцы одинаковой длины, равносильна функции cov([x y]).

Преобразования Фурье

Разработка преобразований Фурье сыграла огромную роль в появлении и развитии ряда новых областей науки и техники. Достаточно отметить, что электротехника переменного тока, электрическая связь и радиосвязь базируются на спектральном представлении сигналов. Ряды Фурье также можно рассматривать как приближение произвольных функций (определенные ограничения в этом известны) тригонометрическими рядами бесконечной длины. При конечной длине рядов получаются наилучшие среднеквадратические приближения. MATLAB содержит функции для выполнения быстрого одномерного и двумерного быстрого дискретного преобразования Фурье.

Прямое преобразование Фурье переводит описание сигнала (функции времени) из временной области в частотную, а обратное преобразование Фурье переводит описание сигнала из частотной области во временную. На этом основаны многочисленные методы фильтрации сигналов.

Функции одномерного прямого преобразования Фурье

В описанных ниже функциях реализован особый метод *быстрого преобразования Фурье* – Fast Fourier Transform (FFT, или БПФ), позволяющий резко уменьшить число арифметических операций в ходе приведенных выше преобразований. Он особенно эффективен, если число обрабатываемых элементов (отсчетов) составляет 2^m, где m – целое положительное число. Используется следующая функция:

- fft(X) – возвращает для вектора X дискретное преобразование Фурье, по возможности используя алгоритм быстрого преобразования Фурье. Если X – матрица, функция fft возвращает преобразование Фурье для каждого столбца матрицы;
- fft(X.n) – возвращает n-точечное преобразование Фурье. Если длина вектора X меньше n, то недостающие элементы заполняются нулями. Если длина X больше n, то лишние элементы удаляются. Когда X – матрица, длина столбцов корректируется аналогично;
- fft(X,[Ldirn) и fft(X,n,dim) – применяют преобразование Фурье к одной из размерностей массива в зависимости от значения параметра dim.

Для иллюстрации применения преобразования Фурье создадим трехчастотный сигнал на фоне сильного шума, создаваемого генератором случайных чисел:»

```
» t=0:0.0005:1:
» x=sin(2*pi*200*t)+0.4*sin(2*pi*150*t)
+0.4*sin(2*pi*250*t):
» y=x+2*randn(size(t));
» plot(y(l:100).'b')
```

Этот сигнал имеет среднюю частоту 200 рад/с и два боковых сигнала с частотами 150 и 250 рад/с, что соответствует амплитудно-модулированному сигналу с частотой модуляции 50 рад/с и глубиной модуляции 0.8 (амплитуда боковых частот составляет 0.4 от амплитуды центрального сигнала). На рисунке 6.1 показан график этого сигнала (по первым 100 отсчетам из 2000). Нетрудно заметить, что из него никоим образом не видно, что полезный сигнал – амплитудно-модулированное колебание, настолько оно забито шумами. Теперь построим график спектральной плотности полученного сигнала с помощью прямого преобразования Фурье, по существу переводящего временное представление сигнала в частотное. Этот график в области частот до 300 Гц (см. рис. 6.1) строится с помощью следующих команд:

```
Y=fft(y,1024);
Pyy=Y.*conj(Y)/1024;
f=2000*(0:150)/1024;
plot(f,Pyy(1:151))
grid on
```

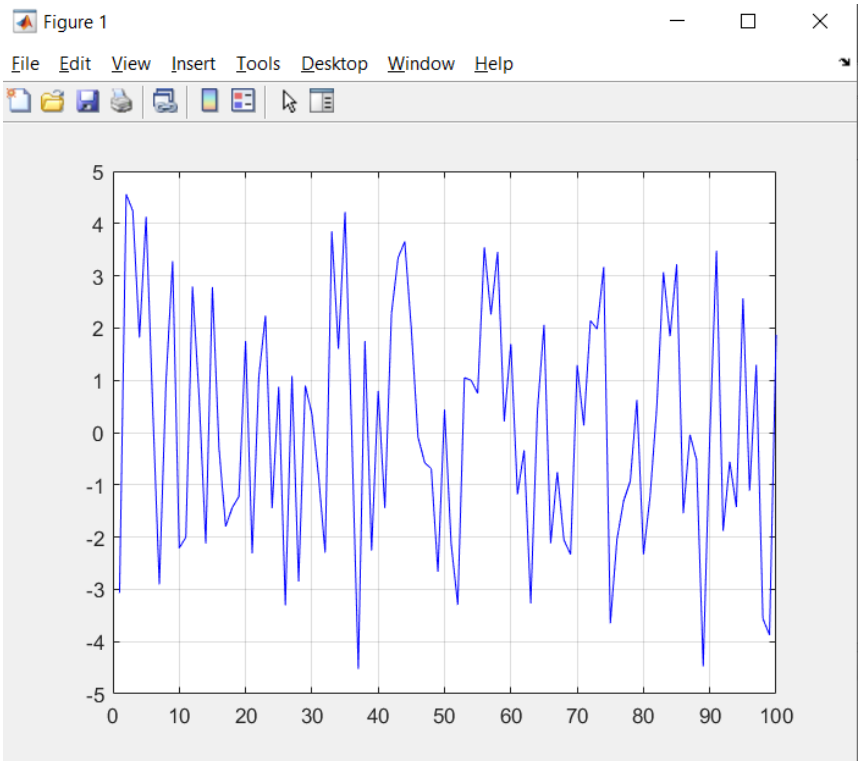


Рис. 6.1. Форма зашумленного сигнала

График спектральной плотности сигнала, построенный в этом примере, представлен на рис. 6.2. Даже беглого взгляда на рисунок достаточно, чтобы убедиться в том, что спектрограмма сигнала имеет явный пик на средней частоте амплитудно-модулированного сигнала и два боковых пика. Все эти три частотные составляющие сигнала явно выделяются на общем шумовом фоне. Таким образом, данный пример наглядно иллюстрирует технику обнаружения слабых сигналов на фоне шумов, лежащую в основе работы радиоприемных устройств.

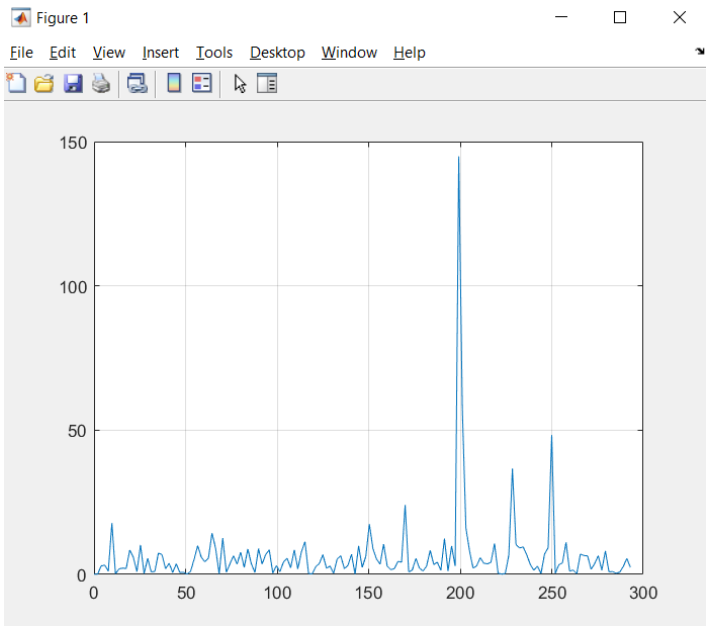


Рис. 6.2. График спектральной плотности приведенного на рис. 6.1 сигнала

Функции многомерного прямого преобразования Фурье

Для двумерного прямого преобразования Фурье используется функция fft2:

- fft2(X) – возвращает для массива данных X двумерное дискретное преобразование Фурье;
- fft2(X,m.n) – усекает массив X или дополняет его нулями, чтобы перед выполнением преобразования Фурье создать матрицу размера *mxn*. Результат – матрица того же размера.

Для многомерного прямого преобразования Фурье также существует функция:

- fftn(X) – возвращает результат N-мерного дискретного преобразования для массива X размерности *N*. Если X – вектор, то выход будет иметь ту же ориентацию;
- fftn(X.siz) – возвращает результат дискретного преобразования для массива X с ограничением размера, заданным переменной siz.

Функции обратного преобразования Фурье

Возможно одномерное обратное преобразование Фурье, реализуемое следующими функциями:

- ifft(F) – возвращает результат дискретного обратного преобразования Фурье вектора F. Если F – матрица, то if ft возвращает обратное преобразование Фурье для каждого столбца этой матрицы;
- ifft(F.n) – возвращает результат n-точечного дискретного обратного преобразования Фурье вектора F;
- ifft(F.[ ],dim) иу = ifft(X,n,dim) – возвращают результат обратного дискретного преобразования Фурье массива F по строкам или по столбцам в зависимости от значения скаляра dim.

Для любого *X* результат последовательного выполнения прямого и обратного преобразований Фурье ifft(fft(x)) равен X с точностью до погрешности округления. Если X – массив действительных чисел, ifft(fft(x)) может иметь малые мнимые части.

Пример:

```
» V=[l 1110000]:
» fft(V)
ans =
Columns 1 through 4
4.0000 1.0000 - 2.41421 0 1.0000 - 0.41421
Columns 5 through 8
 0 1.0000 + 0.41421 0 1.0000 + 2.41421
» 1fft(fft(V))
ans =
1 1 1 1 0 0 0 0
```

Аналогичные функции есть для двумерного и многомерного случаев:

- ifft2(F) – производит двумерное дискретное обратное преобразование Фурье для матрицы F;
- ifft2(F,m,n) – производит обратное преобразование Фурье размерности *mxn* для матрицы F;
- ifftn(F) – возвращает результат JV-мерного обратного дискретного преобразования Фурье для N-мерного массива F;
- ifftn(F.siz) – возвращает результат обратного дискретного преобразования Фурье для массива F с ограничением размера, заданным вектором siz. Если любой элемент siz меньше, чем соответствующая размерность F, то массив F будет урезан до размерности siz.

Дискретная одномерная фильтрация

MATLAB может использоваться для моделирования работы цифровых фильтров. Для обеспечения дискретной одномерной фильтрации используется функция *filter* в следующих формах записи:

- filter(B,A,X) – фильтрует одномерный массив данных X, используя дискретный фильтр, описываемый следующим конечноразностным уравнением:

$$a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + \ldots + b(nb+1)*x(n-nb) - a(2)*y(n-1) - \ldots - a(na+1)*y(n-na).$$

Если a(1) не равно 1, то коэффициенты уравнения нормализуются относительно a (1). Когда X – матрица, функция *filter* оперирует столбцами X. Возможна фильтрация многомерного (размерности *N*) массива.

- [Y. Zf]=filter(B, A, X, Zi) – выполняет фильтрацию с учетом ненулевого начального состояния фильтра Zi; возвращает помимо выходного сигнала Y конечное состояние фильтра Zf;
- filter(B.A.X,[ ].dim) или fi!ter(B,A,X.Zi,dim) – работает в направлении размерности dim

Двумерная фильтрация

Для осуществления двумерной фильтрации служит функция filter2:

- filter2(B,X) – фильтрует данные в двумерном массиве X, используя дискретный фильтр, описанный матрицей B. Результат Y имеет те же размеры, что и X;
- filter2(B,X, 'option') – выполняет то же, но с опцией, влияющей на размер массива Y:

'same' – size(Y)=size(X) (действует по умолчанию);

'valid' – size(Y) < size(X), центральная часть двумерной свертки, при вычислении которой не приходится дополнять массивы нулями;
'full' – size(Y) > size(X), полная двумерная свертка

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 7
ЧТЕНИЕ И ЗАПИСЬ ИНФОРМАЦИИ В ФАЙЛЫ

Цель работы: освоение базовых операций, связанных с записью и чтением результатов вычислений в файлы.

При разработке собственных инженерных приложений часто требуется сохранить результаты расчетов в файлы для их дальнейшего анализа, обработки, хранения и т.п. В связи с этим в MATLAB реализованы различные функции по работе с файлами, содержащие данные в самых разных форматах.

Рассмотрим наиболее простые и полезные функции для сохранения и загрузки результатов работы собственных инженерных приложений из файлов различных форматов.

Функции save и load

Для сохранения и последующей загрузки каких-либо данных в MATLAB предусмотрены две функции:

```
save <имя файла> <имена переменных> % сохранение данных
load <имя файла> <имена переменных> % загрузка данных
```

Функция *save* предназначена для сохранения данных из рабочей среды MATLAB в файл, который будет (по умолчанию) располагаться в рабочем каталоге (обычно поддиректория work). По умолчанию файл с данными будет иметь расширение .mat. Соответственно функция *load* позволяет загрузить из указанного mat-файла ранее сохраненные данные. Причем в один файл могут быть сохранены различные типы данных, как отдельные переменные и числовые массивы, так и строковые переменные.

Рассмотрим простой пример использования данных функций, для этого создадим пустой М-файл и введем в него следующий скрипт:

```
%% Использование функций save и load
% Очистка командного окна и рабочей среды MATLAB clc;
clear all;
% Инициализация переменных
x = [1:1.25:10];
y = 12555;
s = 'Text';
% Сохранение переменных в файл Var.mat
save Var x y s;
%% Загрузка переменных в рабочую среду MATLAB
```

```
% Очистка командного окна и рабочей среды MATLAB clc;
clear all;
% Загрузка переменных x, y, z из файла Var.mat
load Var x y s;
% Вывод переменных x, y, z в Command Window
disp(x);
disp(y);
disp(s);
```

Следует обратить внимание на то, что функция load позволяет загружать из mat-файла не все, а только указанные пользователем переменные, например:

```
load Var.mat x; % загружает только значение переменной x
load Var.mat y; % загружает только значение переменной y
load Var.mat x s; % загружает значения переменных x и s
```

Для управления форматами файлов следует в команде save в дополнение к имени файла и списку переменных использовать следующие флаги, приведенные в табл. 7.1.

7.1. Форматы файлов

Флаг	Пояснение
-mat	Двоичный MAT-файл (по умолчанию)
-ascii	ASCII-формат (8 цифр)
-ascii -double	ASCII-формат (16 цифр)
-ascii -double -tabs	Формат с разделителями и метками табуляции
-v4	Формат версии MATLAB 4
-append	Добавить данные к существующему MAT-файлу

При использовании флага *v4* можно сохранить только те данные, которые совместимы с данными, используемыми в версии MATLAB 4; это означает, что сохранить такие типы данных как массивы записей, ячеек, многомерные массивы или объекты нельзя.

Когда содержимое рабочей области сохраняется в ASCII-формате, то рекомендуется единовременно сохранять только одну переменную. Если сохраняется более одной переменной, то система MATLAB создаст файл ASCII-файл, который нельзя будет в дальнейшем загрузить в MATLAB, используя команду *load*.

Команды *load* и save допускают использование группового символа (*) в качестве замены ряда символов в шаблоне имени переменной.

Например, команда *save Test1 x** сохраняет все переменные, имена которых начинаются с символа x в файле с именем Test1.mat.

Точно также команда *load Test1 var*40* загружает все переменные, имена которых начинаются с символов *'var'* и заканчиваются символами *'40'*, независимо от того, какие символы размещены между ними.

Работа с текстовыми файлами

Ввод текстовой информации. Работа с текстовыми файлами состоит и трех этапов: открытие файла, считывание или запись данных, закрытие файла.

Для открытия файла используется команда *fopen*:

```
[fid, mes] = fopen(filename, permission)
```

– открывает файл с именем *filename* и параметром, определенным в permission, и возвращает идентификатор *fid*, присвоенный файлу (порядковый номер файла), в *mes* – содержится информация об открытии файла или сообщение об ошибке.

Идентификатор *fid* используется в качестве аргумента других функций и программ ввода-вывода. Имя файла *filename* может содержать путь к файлу.

Параметр *permission* может принимать одно из следующих значений:

'r' – открытие файла для чтения (по умолчанию);
' r+' – открытие файла для чтения и записи;
'w' – удаление содержимого существующего файла или создание нового и открытие его для записи;
'a' – создание и открытие нового файла или открытие существующего для записи с добавлением в конец файла.

Команда *fclose* закрывает файл. Она имеет следующие варианты.

```
status = fclose(fid)
```

– закрывает файл, если он открыт. Возвращает статус файла *status*, равный 0, если закрытие завершилось успешно, и -1 в противном случае. Аргумент *fid* – это идентификатор, связанный с открытым файлом.

Далее рассмотрим особенности записи информации в текстовые файлы.

Добавление строки в текстовый файл осуществляется функцией *fprintf* с двумя входными аргументами – идентификатором файла и строкой с текстом или любыми другими символами и знаками, вводимыми с клавиатуры.

Рассмотрим несколько примеров, иллюстрирующих применение команд *fopen*, *fclose* и *fprintf*:

```
%%Пример 1 – Применение команд fopen, fclose и fprintf
% Очистка командного окна и рабочей среды MATLAB
clc;
clear all;
% Создание файла Example1.txt и открытие его для записи
fid,mes]= fopen('Example1.txt', 'w')
```

```
% Запись строки в файл Example1.txt
fprintf(fid, 'Строка добавлена командой fprintf');
% Закрытие файла Example1.txt
fclose(fid);
```

Последующая команда *fprintf* выводит заданную строку сразу за предыдущей на следующую строку текстового файла.

Для ввода текста в текстовый файл с новой строки следует добавить символ перехода на новую строку \n, при этом возможны два варианта:

- в начале строковой переменной

```
fprintf(fid, '\n.');
```

- в конце строковой переменной

```
fprintf(fid, 'Перевод каретки на новую строку. \n ');
```

```
%%Пример 2 – Применение команд fopen, fclose и fprintf
clc;
clear all;
% Создание файла Example1.txt и открытие его для записи
[fid,mes]= fopen('Example1.txt', 'w')
% Запись строки в файл Example1.txt
fprintf(fid, 'Строка добавлена командой fprintf. ');
% Запись строковой переменной и перевод каретки на новую
строку
fprintf(fid, 'Второе предложение. \n ');
% Запись информации с новой строки
fprintf(fid, 'Текст добавлен с новой строки.');
% Закрытие файла Example1.txt
fclose(fid);
% Просмотр содержимого текстового файла в CommandWindow
type Example1.txt
```

Форматный вывод числовых данных. Символы текстового файла образуют строки со словами, предложениями или числами. Запись текстовых строк достаточно проста, а вот для записи в текстовый файл чисел используют специальные форматы. Для вывода цифровой информации в текстовый файл используется функция *fprintf*.

Занесение чисел или значений текстовых переменных требует использования форматного вывода. Схема использования команды fprintf при работе с числовыми переменными такова:

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

Первый аргумент команды *fprintf* является идентификатором файла, второй – строка со специальными символами, определяющими формат вывода чисел в текстовый файл, третий – список выводимых переменных.

В случае если идентификатор файла отсутствует, то вывод информации производится в Command Window, а не в текстовый файл.

Рассмотрим применение форматного вывода на следующем примере.

Необходимо записать значения переменных $x = 0{:}\pi/2{:}2\pi$ и функции $y = \sin(x)$ в текстовый файл, в формате с плавающей точкой, оставляя четыре цифры после запятой для значений x и две цифры для y. Для этого составим следующую программу:

```
% Очистка Command Window и Workspace
clc; clear all;
% Создание файла Example2.txt
[fid,mes] = fopen('Example2.txt', 'w');
% Вычисление значений аргумента x и функции y
x = 0:pi/2:2*pi;
y = sin(x);
% Конструирование матрицы значений x,y
M=[x;y];
% Вывод информации в файл 'Example2.txt'
fprintf(fid, 'Таблица значений функции \n y = sin(x)\n');
fprintf(fid, '_____________ fprintf(fid, '| x | y|\n');
\n');
fprintf(fid, '    \n');
% Форматный вывод значений матрицы
fprintf(fid, '|%7.3f |%10.4f |\n', M);
fprintf(fid, '_____________    \n');
% Закрытие файла Example2.txt
fclose(fid);
% Вывод содержимого файла Example2.txt в CommandWindow
type Example2.txt
```

В результате работы описанной программы создается файл Example2.txt, содержимое которого приведено ниже:

Таблица значений функции y = sin(x)

```
________-
| x | y |
________-
| 0.000 | 0.0000 |
| 1.571 | 1.0000 |
| 3.142 | 0.0000 |
| 4.712 | -1.0000 |
| 6.283 | -0.0000 |
________-
```

Обратите внимание на второй аргумент команды *fprintf*, предназначенной для форматного вывода значений матрицы M. Последовательность %7.3f %10.4f задает формат вывода переменных x и y, составляющих основу матрицы M. Знак процента указывает на начало формата, цифра 7 означает, что под значение переменной x всего отводится семь позиций, цифра три, после разделителя-точки – обеспечивает точность отображения результата – четыре цифры после запятой. Символ *f* указывает на то, что результат выводится в формате с плавающей точкой. Аналогичным образом работает формат %10.4f для переменной y. Таким образом, количество форматов вывода, должно соответствовать количеству выводимых в текстовый файл переменных.

Форматы между собой можно разделять пробелами и другой текстовой информацией, которая запишется в текстовый файл между соответствующими значениями выводимых переменных, например:

```
clc; clear all; a = 12.5;
b = 135.22;
fprintf(' Вывод значений переменных \n');
fprintf('| a = %4.2f | b = %6.3f | \n', a, b)
```

В результате работы программы в командное окно будет выведен следующий результат:

```
Вывод значений переменных
| a = 12.50 | b = 135.220 |
```

Более подробно с видами форматов, при выводе числовой информации в текстовый файл можно ознакомиться, набрав в командной строке fprintf и нажав клавишу F1.

Считывание информации из текстовых файлов. Для считывания информации из текстовых файлов и записи их в переменные подходящих типов предназначена функция *fscanf*, осуществляющая обратное действие по отношению к *fprintf*.

Вызов функции *fscanf* производится с тремя входными аргументами – идентификатором файла, строкой с форматом и числом, считываемых в данном формате объектов, и одним выходным аргументом, предназначенном для записи результата считывания:

```
A = fscanf(идентификатор, формат, число считываемых элементов)
```

Функция *fscanf* допускает считывать матрицы числовых значений размерностью *[n m]*, где *n* – количество строк, *m* – количество столбцов.

Как правило, в состав любого текстового файла входят такие элементы, как текст и числа. Текст всегда считывается в строковые переменные, а числа можно занести как в строковые, так и в числовые переменные.

Для считывания строки предназначен формат %s, для целых чисел – %d, для вещественных – %g.

Рассмотрим применение функции *fscanf* на следующем примере. Необходимо выполнить следующие действия:

1. Вычислить значение функции: $y = e^x$, где x = [0:0.1:1].
2. Записать результаты вычислений в текстовый файл *exp.txt*.
3. Считать информацию из текстового файла *exp.txt* в три переменные рабочей среды MATLAB:

H – заголовок таблицы;

Vr – обозначение аргумента и функции в текстовом файле; A – матрица значений аргумента и функции.

```
clc; clear all;
% Создание таблицы значений функции
x = 0:.1:1;
y = [x; exp(x)];
% Создание и открытие файла для записи
fid = fopen('exp.txt', 'w');
% Ввод информации в текстовый файл
fprintf(fid, 'Таблица значений функции \n');
fprintf(fid, '  x       y \n');
fprintf(fid, '%6.3f %12.4f\n', y);
% Закрытие файла
fclose(fid);
% Просмотр содержимого текстового файла в Command Window
type exp.txt
% Открытие текстового файла
fid = fopen('exp.txt');
% Чтение и запись информации в переменные рабочей среды
H = [fscanf(fid,'%s',1),' ',fscanf(fid,'%s',1),
' ',fscanf(fid,'%s',1)]
Vr = [fscanf(fid,'%s',1), ', ', fscanf(fid,'%s',1)]
A = fscanf(fid, '%g %g', [2 inf]);
% Закрытие файла
fclose(fid);
% Транспонирование матрицы – все строки записать
столбцами
A = A'
```

Результат работы программы будет представлен в командном окне в следующем виде:

```
Таблица значений функции
x    y
0.00 1.00000000
0.10 1.10517092
0.20 1.22140276
0.30 1.34985881
0.40 1.49182470
0.50 1.64872127
0.60 1.82211880
0.70 2.01375271
0.80 2.22554093
0.90 2.45960311
1.00 2.71828183
```

```
H = Таблица значений функции
Vr =x, y
A =
0 1.0000
0.1000 1.1052
0.2000 1.2214
0.3000 1.3499
0.4000 1.4918
0.5000 1.6487
0.6000 1.8221
0.7000 2.0138
0.8000 2.2255
0.9000 2.4596
1.0000 2.7183
```

Задание 1. Создайте текстовый файл, содержащий следующую информацию о студенте в следующем виде (указать свои данные):

```
Персональные данные студента

Фамилия: Иванов
Имя: Иван
Отчество: Иванович
Год рождения: 1990
Группа: БМР201
Специальность: Мехатроника и робототехника
```

Задание 2. Используя ранее полученные результаты при выполнении своего варианта *задания 1 Лабораторной работы 4*, запишите значения переменных x и функции y в текстовый файл, в формате с плавающей точкой, оставляя две цифры после запятой для значений аргумента x и три цифры для значения функции y.

Задание 3. Вычислите значения функции, соответствующей Вашему варианту в таблице, запишите значения аргумента и функции в текстовый файл, затем произведите чтение информации из текстового файла и создайте матрицу M, содержащую значения функции и аргумента.

№ варианта	Функция	Аргумент
1	$y = 2(1.2 – 3.2\ x^2) + 3\ \sin(x)$	x = [0:0.5:5];
2	$y = 5\ e^x + 2\ \cos(x)$	x = [0:0.25:4];
3	$y = 3\ \sin(x) + 2\ e^x$	x = [0:0.1:1];
4	$y = 4\ x^{1/3} – x^{2/3}$	x = [1:1.5:10];
5	$y = 2.5\ x + \ln(x)$	x = [10:2.5:20];
6	$y = \sin(2.5 + 4x)^3$	x = [0:0.25:5];
7	$y = 1 + x\ e^x$	x = [0:0.5:6];
8	$y = \ln(1 + x^2) + 1$	x = [3:1:12];
9	$y = e^x\ \cos^2 x$	x = [0:0.1:0.5];
10	$y = \ln^2 x/x$	x = [2:1:6];

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 8
ВОЗМОЖНОСТИ ЦИФРОВОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Цель работы: изучить возможности пакета Image Processing Toolbox для обработки цифровых изображений.

На сегодняшний день система Matlab, в частности, пакет прикладных программ Image Processing Toolbox, является наиболее мощным инструментом для моделирования и исследования методов обработки изображений. Он включает большое количество встроенных функций, реализующих наиболее распространенные методы обработки изображений. Рассмотрим основные возможности пакета Image Processing Toolbox.

Геометрические преобразования изображений

К наиболее распространенным функциям геометрических преобразований относится кадрирование изображений (imcrop), изменение размеров (imresize) и поворот изображения (imrotate).

Суть кадрирования состоит в том, что функция imcrop позволяет с помощью мыши в интерактивном режиме вырезать часть изображения и поместить ее в новое окно просмотра.

```
L=imread('cameraman.tif');
imshow(L);
imcrop;
```

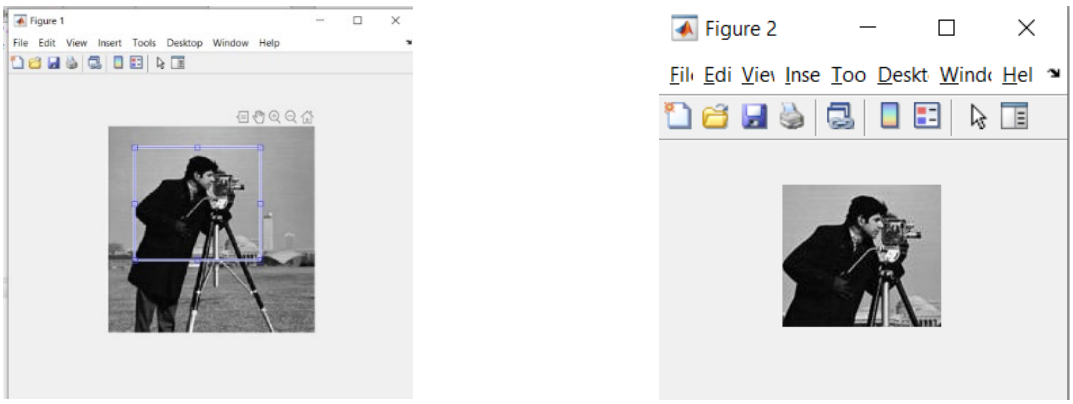


Рис. 8.1. Кадрирование изображения

Функция изменения размеров изображения *imresize* позволяет, используя специальные методы интерполяции, изменять размер любого типа изображения.

В пакете *Image Processing Toolbox* существует функция imrotate, которая осуществляет поворот изображения на заданный угол.

```
L1=imrotate(L,35,'bicubic');
figure,imshow(L1)
```

Рис. 8.2. Поворот изображения

Таким образом, приведенные выше функции позволяют поворачивать, вырезать части, масштабировать, т.е. работать с целым массивом изображения.

Анализ изображений

Для работы с отдельными элементами изображений используются такие функции как *imhist*, *impixel*, *mean2*, *corr2* и другие.

Одной из наиболее важных характеристик является гистограмма распределения значений интенсивностей пикселей изображения, которую можно построить с помощью функции *imhist*.

```
L=imread('cameraman.tif');
figure, imshow(L);
figure, imhist(L);
```

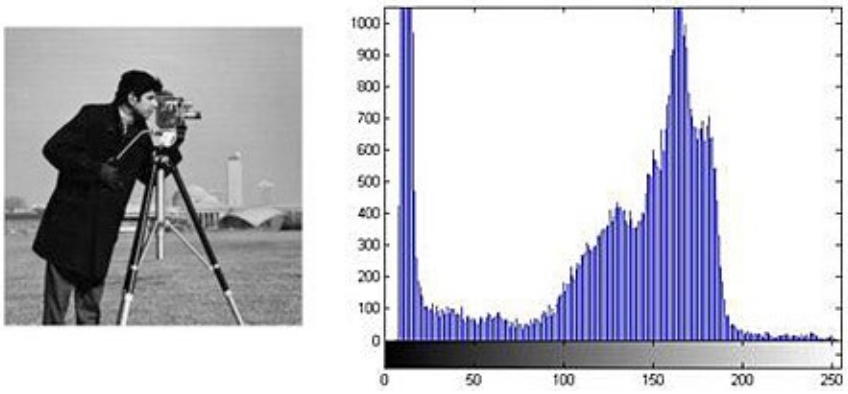


Рис. 8.3. Гистограмма распределения значений интенсивностей пикселей изображения

Довольно часто при проведении анализа изображений возникает необходимость определить значения интенсивностей некоторых пикселей. Для этого в интерактивном режиме можно использовать функцию *impixel*.

```
>> impixel
ans =
   173   173   173
   169   169   169
   163   163   163
    39    39    39
```

Следует отметить, что функция *impixel* по своим возможностям в некоторой степени повторяет опцию *Data Cursor*, пример использования которой при веден на изображении внизу.

Рис. 8.4. Пример использования функции *Data Cursor*

Еще одной широко применяемой функцией является функция *mean2* – она вычисляет среднее значение элементов матрицы.

Функция *corr2* вычисляет коэффициент корреляции между двумя матрицами. Другими словами, с помощью функции *corr2* можно сказать насколько две матрицы или изображения похожи между собой. Эта функция широко применяется при решении задач распознавания.

Улучшение изображений

Среди встроенных функций, которые реализуют наиболее известные методы улучшения изображений, выделим следующие – *histeq*, *imadjust* *imfilter*(*fspecial*).

Как уже отмечалось ранее, гистограмма изображения является одной из наиболее информативных характеристик. На основе анализа гистограммы можно судить о яркостных искажениях изображения, т.е. сказать о том, является ли изображение затемненным или засветленным. Известно, что в идеале на цифровом изображении в равном количестве должны присутствовать пиксели со всеми значениями яркостей, т.е. гистограмма должна быть равномерной. Перераспределение яркостей пикселей на изображении с целью получения равномерной гистограммы выполняет метод эквализации, который в системе Matlab реализован в виде функции *histeq*.

```
L=imread('cameraman.tif');
figure, imshow(L);
L1=histeq(L);
Figure, imsow(L1);
```

а) *б*)

Рис. 8.5. Изображение до (*а*) и после (*б*) эквализации гистограммы

Довольно часто при формировании изображений не используется весь диапазон значений интенсивностей, что отрицательно отражается на качестве визуальных данных. Для коррекции динамического диапазона сформированных изображений используется функция *imadjust*.

```
L=imread('cameraman.tif');
figure, imshow(L);
L1=imadjust(L);
figure, imshow(L1);
figure, imhist(L);
figure, imhist(L1);
```

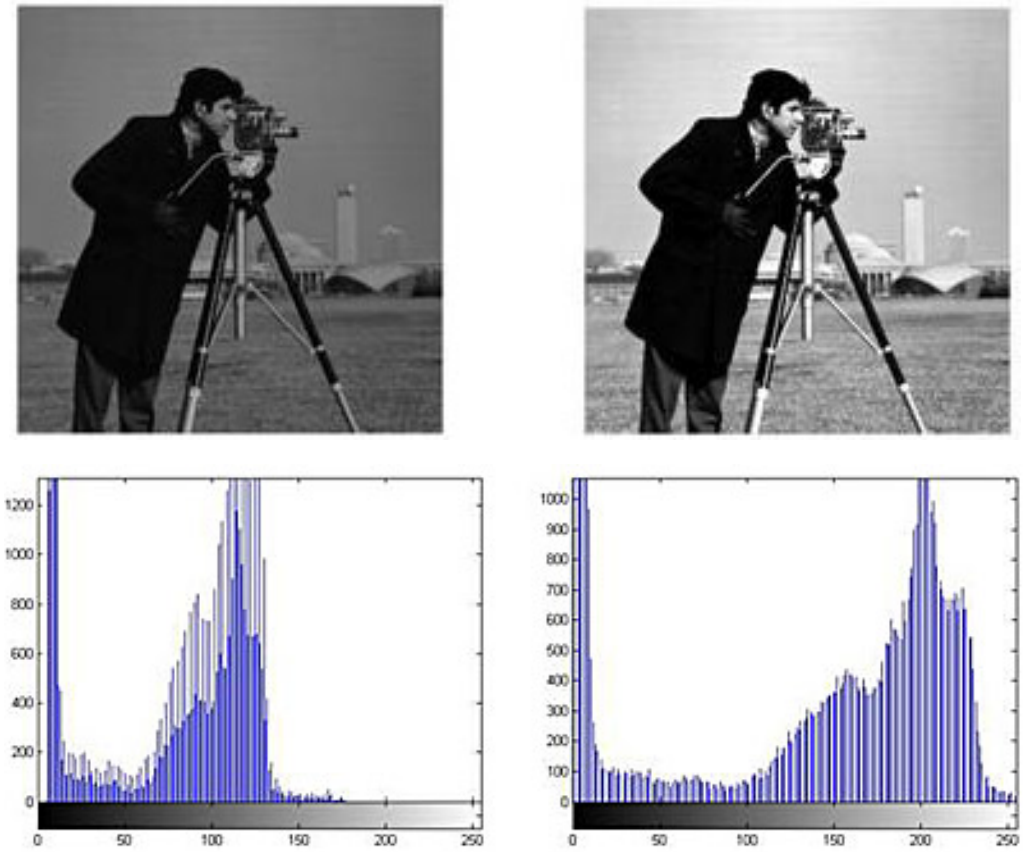


Рис. 8.6. Коррекция динамического диапазона изображения

Также при решении задач улучшения изображений используется функция *imfilter* в паре с функцией *fspecial*. Функция *fspecial* позволяет задавать различные типы масок фильтра. Рассмотрим пример использования маски фильтра, повышающего резкость изображения.

```
L=imread('cameraman.tif');
figure, imshow(L);
H = fspecial('unsharp');
L1 = imfilter(L,H,'replicate');
figure, imshow(L1);
```

Рис. 8.7. Применение маски фильтра к изображению

Фильтрация изображений

Пакет Image Processing Toolbox обладает очень мощным инструментарием по фильтрации изображений. Среди множества встроенных функций, которые решают задачи фильтрации изображений, особо следует выделить *fspecial*, *ordfilt2*, *medfilt2*.

Функция fspecial является функцией задания маски предопределенного фильтра. Эта функция позволяет формировать маски:

1) высокочастотного фильтра Лапласа;
2) фильтра, аналогичного последовательному применению фильтров Гаусса и Лапласа, так называемого лапласиана-гауссиана;
3) усредняющего низкочастотного фильтра;
4) фильтра, повышающего резкость изображения.

Рассмотрим примеры применения названных выше фильтров:

```
L=imread('cameraman.tif');
figure, imshow(L);
h=fspecial('laplasian',.5);
L1 = imfilter(L,h,'replicate');
figure, imshow(L1);
```

Рис. 8.7. Высокочастотный фильтр Лапласа

```
h=fspecial('log', 3, .5);
L1 = imfilter(L,h,'replicate');
figure, imshow(L1);
```

Рис. 8.8. Фильтр лапласиана-гауссиана

```
h=fspecial('average', 3);
L1 = imfilter(L,h,'replicate');
figure, imshow(L1);
```

Рис. 8.9. Усредняющий низкочастотный фильтр

```
h=fspecial('unsharp', .5);
L1 = imfilter(L,h,'replicate');
figure, imshow(L1);
```

Рис. 8.10. Повышение резкости изображения

Сегментация изображений

Среди встроенных функций пакета Image Processing Toolbox, которые применяются при решении задач сегментации изображений, следует выделить *qtdecomp*, *edge и roicolor*.

Функция qtdecomp выполняет сегментацию изображения методом разделения и анализа однородности неперекрывающихся блоков изображения.

```
I = imread('cameraman.tif');
S = qtdecomp(I,.27);
blocks = repmat(uint8(0),size(S));

for dim = [512 256 128 64 32 16 8 4 2 1];
  numblocks = length(find(S==dim));
  if (numblocks > 0)
    values = repmat(uint8(1),[dim dim numblocks]);
```

```
    values(2:dim,2:dim,:) = 0;
       blocks = qtsetblk(blocks,S,dim,values);
  end
end

blocks(end,1:end) = 1;
blocks(1:end,end) = 1;

imshow(I), figure, imshow(blocks,[])
```

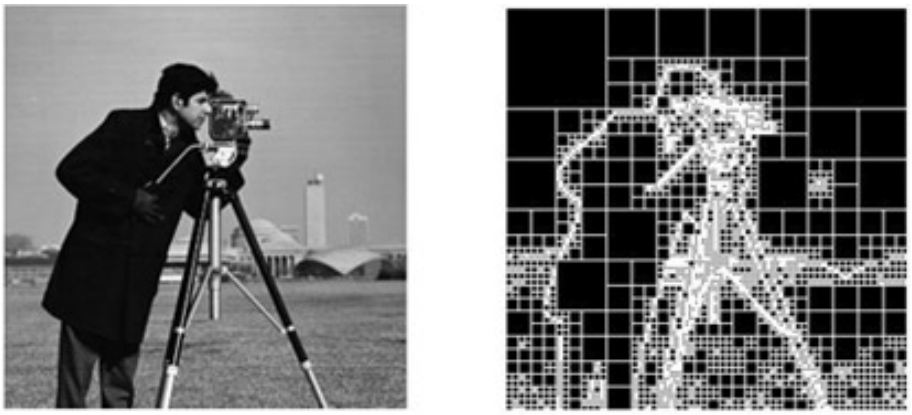


Рис. 8.11. Сегментация изображения

Одной из наиболее часто применяемых является функция выделения границ edge, которая реализует такие встроенные методы – Собела, Превит, Робертса, лапласиан-гауссиана, Канни и др.

Приведем примеры реализации функции edge с использованием различных фильтров.

```
clear;
I = imread('cameraman.tif');
BW1=edge(I,'sobel');
figure,imshow(BW1);title('sobel');
BW2=edge(I,'prewitt');
figure,imshow(BW2);title('prewitt');
BW3=edge(I,'roberts');
figure,imshow(BW3);title('roberts');
BW4=edge(I,'log');
figure,imshow(BW4);title('log');
BW5=edge(I,'zerocross');
figure,imshow(BW5);title('zerocross');
BW6=edge(I,'canny');
figure,imshow(BW6);title('canny');
```

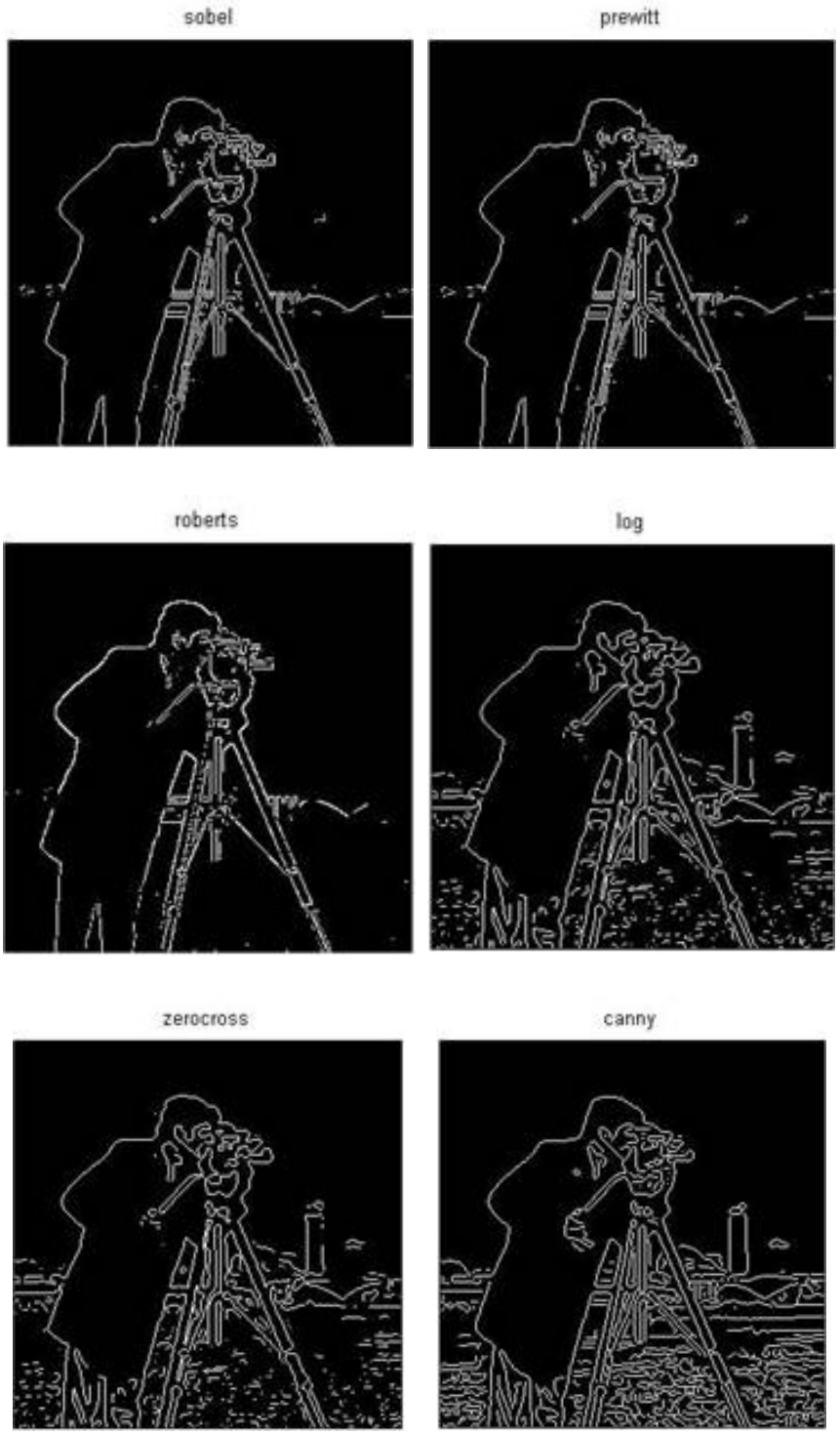


Рис. 8.12. Функция edge с использованием различных фильтров

Еще одной функцией, которая применяется для бинаризации по заданным цветам, является функция roicolor. Приведем пример ее использования.

```
I = imread('cameraman.tif');
figure, imshow(I);
BW = roicolor(I,128,255);
imshow(I);
figure, imshow(BW);
```

Рис. 8.13. Применение функции roicolor

Морфологические операции над бинарными изображениями

Система Matlab владеет довольно мощным инструментарием морфологической обработки бинарных изображений. Среди основных операций выделим следующие – эрозия, наращивание, открытие, закрытие, удаление изолированных пикселей, построение скелета изображения и т.п.

В качестве примера приведем решение задачи разделения слипшихся объектов с помощью морфологических операций.

```
clear;
L=imread('circles.bmp');
L=L(:,:,1);
imshow(L);
BW1=L<150;
figure,imshow(BW1);
BW2=bwmorph(BW1,'erode',12);
figure,imshow(BW2);
BW2=bwmorph(BW2,'thicken',Inf);
figure,imshow(BW2);
BW1=BW1&BW2;
figure,imshow(BW1);
```

Кроме перечисленных возможностей, пакет прикладных программ Image Processing Toolbox владеет широкими возможностями при решении задач анализа изображений, в частности, при поиске объектов и вычислении их признаков.

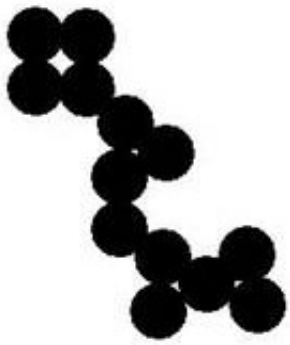
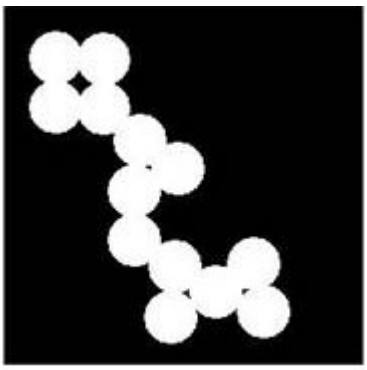
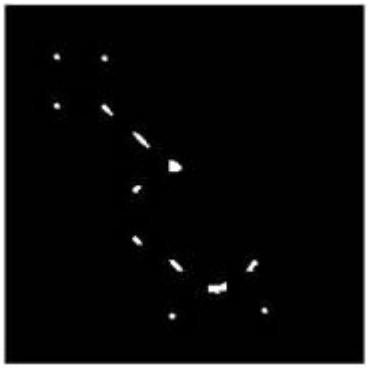


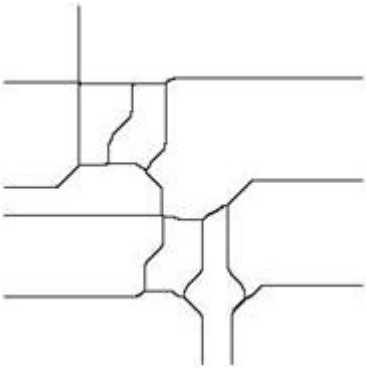
Рис. 8.13. Исходное изображение



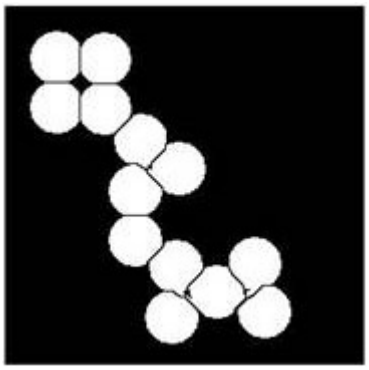
Бинарное изображение



Эрозия бинарного изображения



Утолщение объектов



Результат разделения слипшихся объектов

Рис. 8.14. Изображение после преобразований

Из представленного материала видно, что пакет прикладных программ Image Processing Toolbox обладает мощным инструментарием для обработки и анализа цифровых изображений. Это приложение является очень удобной средой для разработки и моделирования различных методов. Возможности каждой из приведенных выше, а также других функций будут детально рассмотрены в последующих материалах.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 9
РАБОТА С ДАННЫМИ НА ЯЗЫКЕ PYTHON

Цель работы: изучить основы применения языка Python для работы с данными, ознакомиться с библиотеками для обработки и визуализации данных, получить навык обработки данных при помощи библиотеки Pandas

Основы применения Python для обработки данных

Python – интерпретируемый язык программирования высокого уровня, который в настоящее время является одним из самых популярных в мире. Он удобен, так как обладает несложным синтаксисом и возможностью подключения дополнительных библиотек для решения широкого круга задач. Чаще всего Python используют для веб-разработки, системном администрировании, разработке игр, а также в сферах Data Science (аналитика данных) и Machine Learning (машинного обучения), где ему практически нет равных.

Работа с данными осуществляется в несколько этапов – сбор, обработка, анализ, построение моделей и визуализация – использовать Python можно на любом из них. Применение Python в Data Science и Machine Learning – тема для отдельного курса, в рамках данного практикума мы лишь коснемся основ применения языка Python для работы с данными.

Для того, чтобы работать с данными на Python существует множество дополнительных импортируемых библиотек, рассмотрим наиболее известные и популярные из них:

1. Получение данных. Для поиска данных инженеры используют Scrapy и BeautifulSoup. С помощью Scrapy можно создавать программы, которые собирают структурированные данные в сети. BeautifulSoup применяется там, где получить данные из API не выходит; он собирает данные и расставляет их в определенном формате.

2. Обработка и моделирование данных. На этом этапе в числе самых используемых библиотек NumPy и Pandas. NumPy (Numerical Python) используется для сортировки больших наборов данных. Он упрощает математические операции и их векторизацию на массивах. Pandas предлагает два структуры данных: Series (список элементов) и Data Frames (таблица с несколькими колонками). Эта библиотека конвертирует данные в формат Data Frame (рис. 9.1), позволяя удалять и добавлять новые колонки, а также выполнять разные операции.

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino

Рис. 9.1. Data Frame

3. Визуализация данных. Matplotlib и Seaborn (рис. 9.2) широко используются для визуализации данных. Они помогают конвертировать огромные списки чисел в удобные графики, гистограммы, диаграммы, тепловые карты и т.д.

Конечно, существуют и другие библиотеки. Python предлагает бесчисленное количество инструментов для проектов в сфере анализа данных и может помочь при выполнении любых задач в процессе.

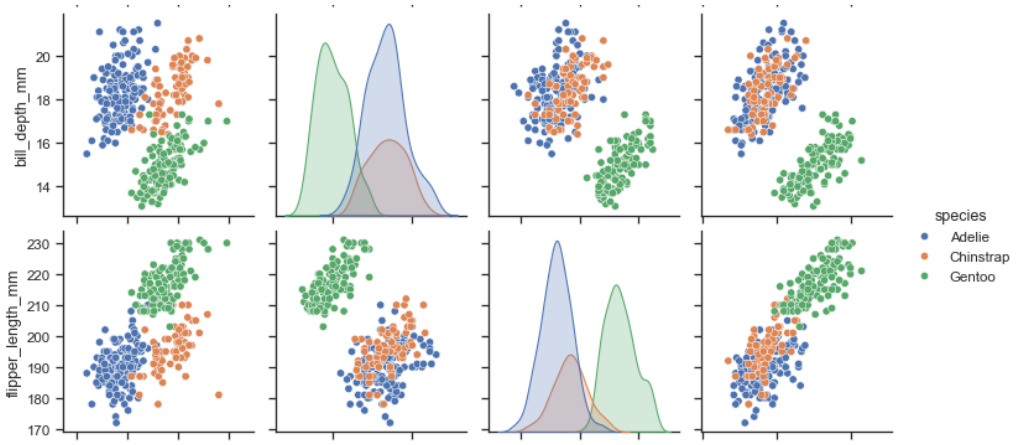


Рис. 9.2. Графическая библиотека Seaborn

4. Среда разработки. Существует большое количество редакторов кода, позволяющих писать и редактировать код на Python, однако в мире Data Science одним из популярных инструментов для анализа данных и быстрого прототипирования уже несколько лет считается Jupyter-ноутбук. В Jupyter Notebook две основные части: веб-приложение и ноутбуки (блокноты) – файлы с расширением **.ipynb*, в которых работают с исходным кодом программы, запускают его и выводят данные в разных форматах. Для экспорта ноутбуков используют два формата – PDF и HTML. Что можно делать в веб-приложении:

- запускать и редактировать код в браузере;
- показывать результаты расчетов, используя схемы и графики;
- использовать язык разметки Markdown и LaTeX.

Один из плюсов этого инструмента в том, что код можно разделить на кусочки и работать над ними в любом порядке. Например, написать скрипт и сразу посмотреть, как он работает. Остальные фрагменты кода при этом запускать не нужно, результат появляется тут же, под частью кода. Так специалисты по Data Science выводят предварительные результаты исследований, строят графики и диаграммы (рис. 9.3).

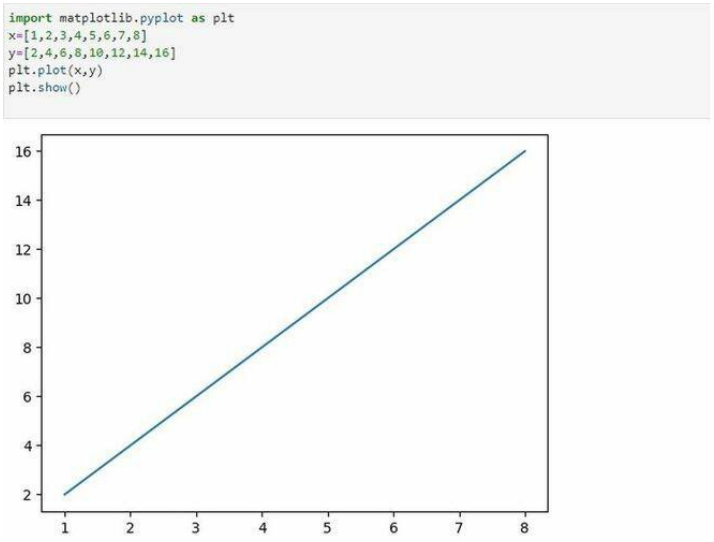


Рис. 9.3. Линейный график в Jupyter-ноутбук

Чтобы установить ноутбук, необходимо ввести в терминале команду: *pip install notebook*. Если выводится сообщение *Successfully installed*, то Jupyter-ноутбук готов к запуску. Чтобы начать работу, используют команду *jupyter notebook*. Второй способ установить Jupyter-ноутбук на компьютер – загрузить Anaconda. Это дистрибутив Python и репозиторий файлов, который используют в основном для анализа данных и машинного обучения. Jupyter Notebook и его расширение JupyterLab как раз входят в этот дистрибутив.

Кроме того, существуют и другие популярные сервисы на основе Jupyter Notebook со схожим принципом работы: *Google Colab*, Datalore от Jet Brains. Они предоставляет возможность писать и запускать код на языке Python, используя только браузер, без установки специальных программ на компьютер.

NumPy. NumPy расшифровывается как «Numeric Python», что можно перевести как «Числовой Python». Данная библиотека предоставляет возможности работы с многомерными массивами. В ее основе лежит класс **ndarray** – многомерный контейнер элементов одного типа данных и размера. Объекты этого класса занимают меньше памяти, а операции над ними производятся быстрее, чем с обычными списками Python. С чем это связано?

Меньший расход памяти обеспечивается за счет определенно заданного размера, так как все элементы представляют собой один тип данных, например, int32 или float32. Быстрота объясняется возможностью не использовать циклы. NumPy использует низкоуровневый язык программирования *C*, а многомерный объект ndarray разворачивается в одномерный массив *C*.

Список (list) в Python содержит массив указателей на объекты. Поэтому извлечение элементов списка подразумевает обращение к объекту и проверку типов. Проведем бытовую аналогию: представьте, что вам нужно перетащить кучу коробок, которые находятся в 20 метрах от вас. При этом вам нужно заглядывать в каждую коробку, чтобы проверить ее содержимое. С точки зрения языка *C* это бы выглядело так: коробки находятся прямо перед вами, заглядывать внутрь не нужно, к тому же вы всегда знаете точное их количество и массу каждой коробки. Вот поэтому массивы NumPy так эффективны, но накладывают определенные ограничения.

Pandas. «Panel Data», сокращено Pandas, на русский язык можно перевести как «Панельные данные». Данный термин в эконометрии используется для набора данных, включающих многократные наблюдения за одними и теми же лицами.

В основе Pandas лежит класс **DataFrame**, предоставляющий возможности работы с двумерными массивами неоднотипных данных. Объект DataFrame составлен из объектов **Series** – одномерных массивов NumPy ndarray, объединенных под одним названием. Поэтому в DataFrame каждый столбец может иметь свой тип данных. На рисунке схематично показано архитектура DataFrame:

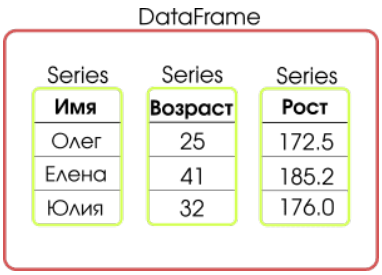


Рис. 9.4. Представление DataFrame и Series

В отличие от NumPy, библиотека Pandas не требует знаний линейной алгебра или умений работать с многомерными массивами (тензорами), которые мы разбирали в этой статье. DataFrame можно представить как таблицу, где каждая строка представляет единичное наблюдение. А со строками и столбцами удобно работать. Python-библиотека Pandas поддерживает такие же операции, как и NumPy: индексацию массивов, статистические исследования,

работу с пустыми (Nan) значениями, изменение формы массивов, извлечение отдельных элементов, конкатенацию. Помимо этого, Pandas также позволяет:

- чтение и запись данных, например, csv или excel таблица;
- группировку данных;
- средства визуализации данных;
- работу с датами;
- работу со строковыми значениями и т.д.

На практике при решении задач Data Science и Machine Learning приходится работать с обеими рассмотренными библиотеками, так как они позволяют быстро и просто обрабатывать большие объемы данных. Даже на онлайн-площадке соревнований по машинному обучению Kaggle обычно первые строчки кода программистов начинаются с `import pandas as pd`, а ниже можно увидеть операцию чтения датасета – `pd.read_csv(...)`. Действительно, в первую очередь необходимо взглянуть на датасет, проанализировать его, построить необходимые графики. Здесь и нужен Pandas.

NumPy, в свою очередь, используется практически в везде, включая другие библиотеки Python, связанные с Data Science и Machine Learning. Например, уже упомянутый Pandas, Seaborn, Statsmodel, в анализе сигналов – Scipy, PyWavelets, в обработке изображений – Scikit-learn и OpenCV, в симуляции моделей – PyDSTool, в глубоком обучении – Keras. Таким образом, эти простые библиотеки являются фундаментом Data Science в Python.

Далее все примеры кода будут представлены в среде Jupyter-ноутбук.

Series – это проиндексированный одномерный массив значений. Он очень похож на массив NumPy, и в его основе как раз и лежит массив NumPy. Отличие Series в том, что в нем для каждого элемента можно указать некоторое название (метка – label). В результате можно обращаться к отдельному элементу Series не только по его номеру внутри массива, как в NumPy, но и по значению метки. В этом отношении Series очень похож на словарь (Dictionary). Метки могут быть любого типа данных, необязательно числа или строки.

Давайте рассмотрим это на примерах.

Чтобы импортировать библиотеку Pandas и создать объект Series на основе списков Python, необходимо выполнить следующий код:

```
In [1]: import pandas as pd
        import numpy as np

In [2]: mycolors = ['black','yellow','green', 'blue']
        data=['000000', 'FFFF00', '008000', '0000F']
        my_series=pd.Series(data=data)

In [3]: my_series

Out[3]: 0    000000
        1    FFFF00
        2    008000
        3     0000F
        dtype: object
```

Замена индекса по умолчанию на индекс пользователя:

```
In [4]: my_series=pd.Series(data=data,index=mycolors)

In [5]: my_series

Out[5]: black     000000
        yellow    FFFF00
        green     008000
        blue       0000F
        dtype: object
```

Создание Data Frame из списка Python:

```
In [6]: df = pd.DataFrame(data=data)

In [7]: df

Out[7]:
```

	0
0	000000
1	FFFF00
2	008000
3	0000F

Создание Data Frame из файла csv

Если файл .csv находится **в той же** папке, что и файл .py или блокнот .ipynb, то можно просто указать название этого файла .csv. Например:

```
df = pd.read_csv('some_file.csv')
```

В другом случае указывается полный путь к файлу. Например:

```
df = pd.read_csv(«C:\\Users\\myself\\files\\some_file.csv»).
```

Некоторые основные функции и методы по работе с Data Frame в Pandas (df- имя DataFrame).

Имя функции/метода	**Описание**	**Пример**
columns	Получить информацию о столбцах	df.columns
index	Получить информацию об индексе DataFrame	df.index
head()	Вывод первых нескольких строк DataFrame, по умолчанию – 5 строк	df.head(3)
tail()	Вывод последних строк DataFrame, по умолчанию – 5 строк	df.tail(3)
info()	Вывод информации о DataFrame : количество строк, столбцов, типы данных, отсутствующие значения	df.info(3)
describe()	Вывод множества статистических значений в нескольких столбцах (медиана, стандартное отклонение, 25 и 75-й процентили, максимальное и минимальные значения)	df.describe()

Примеры редактирования DataFrame

Описание	Код
Обращение к одной колонке	df['total_bill']
Обращение к нескольким колонкам	df[['total_bill','tip']]
Создание новой колонки при помощи математического выражения	df['tip_percentage'] = 100* df['tip'] / df['total_bill']
Изменение данных в существующих колонках (np-numpy)	df['price_per_person'] = np.round(df['price_per_person'],2)
Удаление колонок	df = df.drop(«tip_percentage»,axis=1)
Назначение индекса	df.set_index('Payment ID')
Сброс индекса	df = df.reset_index()
Вывод одной строки на основании номера строки вили ее названия	df.iloc[0] df.loc['Sun2959']
Вывод нескольких строк	df.iloc[0:4]
Вывод нескольких строк по индексу	df.drop('Sun2959',axis=0).head()
Вывод данных, соответствующих условию	bool_series = df['total_bill'] > 30
Объединение нескольких условий: | – логическое ИЛИ (OR) & – логическое И (AND) ~ (тильда) – оператор NOT	df[(df['total_bill'] > 30) & (df['sex']=='Male')]
Isin() фильтрация данных по набору из нескольких указанных значений (аналог оператора in в других языках программирования)	options = ['Sat','Sun'] df['day'].isin(options)
Заполнение пустых значений	df['col1'].fillna (0)

Задание.

1. Прочитать файл с данными, полученный от преподавателя (csv-формат);
2. Вывести на экран блокнота первые 4 строки;
3. Добавить столбец (по формуле) по согласованию с преподавателем;
4. Удалить столбец;
5. Заменить пустые значения, если они имеются в DataFrame;
6. Вывести данные, соответствующие условию, по заданию от преподавателя.

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 10
РЕАЛИЗАЦИЯ АЛГОРИТМА ЛИНЕЙНОЙ РЕГРЕССИИ НА ЯЗЫКЕ PYTHON

Цель работы: ознакомиться с основными алгоритмами машинного обучения, применяемыми в анализе данных. Научиться применять алгоритм линейной регрессии, используя библиотеку scikit-learn.

1. Алгоритмы машинного обучения

Машинное обучение (ML) – это способ наделить компьютеры способностью обучиться выполнению определенных задач без непосредственного программирования этих задач. Это осуществляется за счет того, что вычислительной системе передается информация, которую она преобразует в модели принятия решений, используемые для прогнозирования результатов в дальнейшем.

Восемь главных алгоритмов машинного обучения:

1. Линейная регрессия.
2. Логистическая регрессия.
3. Деревья решений.
4. Метод опорных векторов.
5. Метод k-ближайших соседей.
6. Алгоритм случайный лес.
7. Метод k-средних.
8. Метод главных компонент.

Импорт основных библиотек для построения алгоритмов ML

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

В рамках данного пособия рассматриваются основы алгоритма линейной и полиномиальной регрессии на Python, в интересах читателей самостоятельно ознакомиться с реализацией других алгоритмов ML на языке Python, используя доступные открытые источники.

1.1. Линейная регрессия

Возможно, это самый популярный алгоритм машинного обучения на данный момент. Иногда из двух алгоритмов с одинаковой производительностью лучше выбирать тот, что проще. Линейная регрессия – это алгоритм контролируемого машинного обучения, который прогнозирует результат, основанный на непрерывных функциях. Линейная регрессия универсальна в том смысле, что она имеет возможность запускаться с одной входной *переменной* (*простая линейная регрессия*) или с зависимостью от нескольких переменных (*множественная регрессия*). Суть этого алгоритма заключается в назначении оптимальных весов для переменных, чтобы создать линию или кривую, которая будет использоваться для прогнозирования вывода.

Реализация линейной регрессии на Python, импорт:

```
from sklearn import linear_model
```

Начало работы

```
from sklearn import linear_model
df = pd.read_csv('linear_regression_df.csv')
df.columns = ['X', 'Y']
```

```
df.head()
Визуализация
sns.set_context(«notebook», font_scale=1.1)
sns.set_style(«ticks»)
sns.lmplot('X','Y', data=df)
plt.ylabel('Response')
plt.xlabel('Explanatory')
```

Реализация

```
linear = linear_model.LinearRegression()
trainX = np.asarray(df.X[20:len(df.X)]).reshape(-1, 1)
trainY = np.asarray(df.Y[20:len(df.Y)]).reshape(-1, 1)
testX = np.asarray(df.X[:20]).reshape(-1, 1)
testY = np.asarray(df.Y[:20]).reshape(-1, 1)
linear.fit(trainX, trainY)
linear.score(trainX, trainY)
print('Coefficient: \n', linear.coef_)
print('Intercept: \n', linear.intercept_)
print('R² Value: \n', linear.score(trainX, trainY))
predicted = linear.predict(testX)
```

1.2. Логистическая регрессия

Логистическая регрессия – алгоритм контролируемой классификации. Она позволяет предсказывать значения непрерывной зависимой переменной на интервале от 0 до 1. Некоторые из основных аспектов логистической регрессии лежат в основе других важных алгоритмов машинного обучения, например, для повышения точности прогноза нейросетевой модели.

Импорт:

```
from sklearn.linear_model import LogisticRegression
```

1.3. Деревья решений

Метод деревьев решений (decision trees) – это один из наиболее популярных методов решения задач классификации и прогнозирования. Опыт показывает, что чаще всего данный алгоритм используется именно для классификации. На входе модель принимает набор атрибутов, характеризующих некую сущность, а затем спускается по дереву, тестируя их в зависимости от того, какие значения может принимать целевая функция. Таким образом, классификация каждого нового случая происходит при движении вниз до листа, который и укажет нам значение целевой функции в каждом конкретном случае. Деревья принятия решений становятся все более популярными и могут служить очень сильным инструментом для аналитики данных, особенно в сочетании с простыми методами композиции, такими как случайный лес, бустинг и бэггинг.

Импорт:

```
from sklearn import tree
```

1.4. Метод опорных векторов

Метод опорных векторов, также известный как SVM, является широко известным алгоритмом классификации, который создает разделительную линию между разными категориями данных. Вектор вычисляется путем оптимизации линии таким образом, что ближайшие точки в каждой из групп будут наиболее удалены друг от друга.

Импорт:

```
from sklearn import svm
```

1.5. Метод k-ближайших соседей

K-Nearest Neighbors, или KNN, представляет собой контролируемый алгоритм обучения, который используется преимущественно для решения задач классификации. Данный алгоритм наблюдает за разными центрами (центроидами) и сравнивает расстояние между ними, используя для этого различные функции (обычно евклидово расстояние). Затем определяется, к какому классу принадлежит большинство ближайших объектов обучающей выборки – к этому классу относится и неизвестный объект.

Импорт:

```
from sklearn.neighbors import KNeighborsClassifier
```

1.6. Случайный лес

Случайный лес – популярный алгоритм контролируемого обучения, заключающийся в использовании комитета (ансамбля) решающих деревьев. «Ансамбль» означает, что он берет кучу «слабых учеников» и объединяет их, чтобы сформировать один сильный предиктор. «Слабые ученики» – это все случайные реализации деревьев решений, которые объединяются для формирования сильного предсказателя – случайного леса.

Импорт:

```
from sklearn.ensemble import RandomForestClassifier
```

1.7. Метод k-средних

Метод k-средних – это популярный алгоритм неконтролируемой классификации обучения, обычно используемый для решения проблемы кластеризации. K обозначает количество введенных пользователем кластеров. Алгоритм начинается со случайно выбранных точек, а затем оптимизирует кластеры при помощи функций (евклидово расстояние и т. д), чтобы найти наилучшую группировку точек данных. В итоге специалистам по большим данным необходимо выбрать правильное значение K.

Импорт:

```
from sklearn.cluster import KMeans
```

1.8. Метод главных компонент

PCA (Principal Component Analysis) – алгоритм сокращения размерности, который может быть очень полезен для аналитиков. Главное – это то, что данный алгоритм может значительно уменьшить размерность данных при работе с сотнями или даже тысячами различных функций. Данный алгоритм не контролируется, но пользователь должен анализировать результаты и следить за тем, чтобы сохранялось 95% или около этой цифры первоначального набора данных.

Импорт:

```
from sklearn import decomposition
```

Подробнее о линейной регрессии в Python

Рассмотрим подробнее пример реализации алгоритма линейной регрессии на Python, с использованием библиотеки Python Scikit-Learn, которая является одной из самых популярных библиотек машинного обучения для Python.

Существует два типа алгоритмов контролируемого машинного обучения: регрессия и классификация. Первый предсказывает непрерывные выходы значений, в то время как второй предсказывает дискретные выходы. Например, предсказание цены дома в долларах – это проблема регрессии, тогда как предсказание того, является ли новообразование злокачественным или доброкачественным, – это проблема классификации.

Термин «линейность» в алгебре относится к линейной зависимости между двумя или более переменными. Если мы нарисуем эту связь в двумерном пространстве (в данном случае между двумя переменными), то получим прямую линию.

Давайте рассмотрим сценарий, в котором мы хотим определить линейную зависимость между количеством часов, которые студент изучает, и процентом оценок, которые студент набирает на экзамене. Мы хотим выяснить, что, учитывая количество часов, которые студент готовит к тесту, о том, как высокий балл может достичь студент? Если мы построим независимую переменную (часы) на оси x и зависимую переменную (процент) на оси y, линейная регрессия даст нам прямую линию, которая наилучшим образом соответствует точкам данных, как показано на рис. 10.1.

Мы знаем, что уравнение прямой линии имеет вид:

$$y = mx + b,$$

где b – пересечение с осью OY, а m – наклон линии.

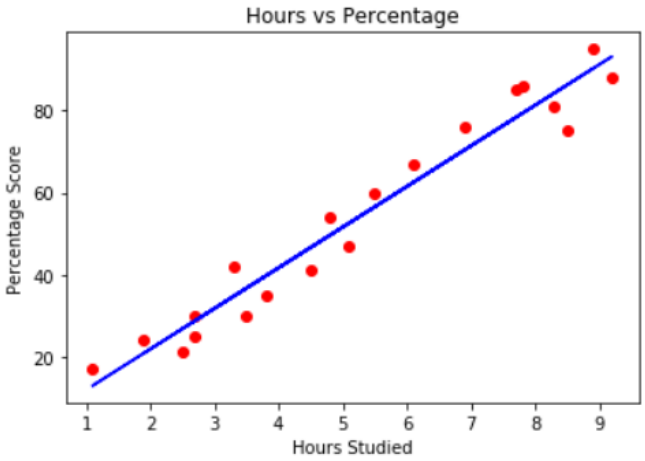


Рис. 10.1. Линейная регрессия

Таким образом, в основном алгоритм линейной регрессии дает нам наиболее оптимальное значение для пересечения и наклона (в двух измерениях). Переменные *y* и *x* остаются неизменными, поскольку они являются функциями данных и не могут быть изменены. Значения, которые мы можем контролировать, – это пересечение и наклон. В зависимости от значений коэффициентов может быть несколько прямых. По сути, алгоритм линейной регрессии помещает несколько строк в точки данных и возвращает строку, которая приводит к наименьшей ошибке.

Эта же концепция может быть распространена на случаи, когда существует более двух переменных. Этот случай называется множественной линейной регрессией. Например, рассмотрим сценарий, в котором вы должны спрогнозировать цену дома, основываясь на его площади, количестве спален, среднем доходе людей в этом районе, возрасте дома и т.д. В этом случае зависимая переменная зависит от нескольких независимых переменных. Регрессионную модель, включающую несколько переменных, можно представить в виде:

$$y = b0 + m1b1 + m2b2 + m3b3 + \ldots \ldots mnbn$$

Это уравнение гиперплоскости. Помните, что линейная регрессионная модель в двух измерениях – это прямая линия, в трех измерениях–плоскость, а в более чем трех измерениях–гиперплоскость.

В данной регрессионной задаче мы будем предсказывать процент оценок, которые студент должен набрать, исходя из количества часов, которые он изучал. Это простая задача линейной регрессии, поскольку она включает в себя всего две переменные.

Чтобы импортировать необходимые библиотеки для этой задачи, выполните следующие инструкции импорта:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Набор данных, используемый для этого примера, общедоступен и может быть загружен по этой ссылке:

https://drive.google.com/open?id=1oakZCv7g3mlmCSdv9J8kdSaqO5_6dIOw

Примечание. Этот пример был выполнен на компьютере под управлением Windows, и набор данных был сохранен в “D:\datasets. Вы можете загрузить файл в другое место, если соответствующим образом измените путь к набору данных.

Следующая команда импортирует набор данных CSV с помощью pandas:

```
dataset = pd.read_csv('D:\Datasets\student_scores.csv')
```

Теперь давайте немного изучим наш набор данных. Для этого выполните следующий сценарий:

```
dataset.shape
```

После этого вы должны увидеть следующее:

```
(25, 2)
```

Это означает, что наш набор данных имеет 25 строк и 2 столбца. Давайте посмотрим, как на самом деле выглядит наш набор данных. Для этого используйте метод *head*() :

```
dataset.head()
```

Описанный выше метод извлекает первые 5 записей из нашего набора данных, который будет выглядеть следующим образом:

```
   Hours  Scores
0    2.5      21
1    5.1      47
2    3.2      27
3    8.5      75
4    3.5      30
```

Чтобы увидеть статистические данные набора данных, мы можем использовать describe() :

```
dataset.describe()
```

Вывод:

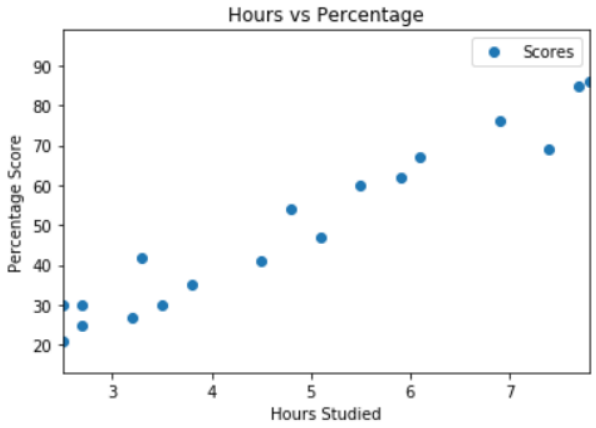
```
            Hours     Scores
count  25.000000  25.000000
mean    5.012000  51.480000
std     2.525094  25.286887
min     1.100000  17.000000
25%     2.700000  30.000000
50%     4.800000  47.000000
75%     7.400000  75.000000
max     9.200000  95.000000
```

И наконец, давайте построим наши точки данных на графике, чтобы посмотреть на наш набор данных и понять, сможем ли мы вручную найти какую-либо связь между этими данными.

```
dataset.plot(x='Hours', y='Scores', style='o')
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.show()
```

В приведенном выше скрипте мы используем функцию *plot*() и передаем ей имена столбцов для *x* и *y* координат, которые являются «Часами» и «Баллами» соответственно.

Результат будет выглядеть так:



Из приведенного выше графика мы ясно видим, что существует положительная линейная зависимость между количеством изученных часов и процентом баллов.

Подготовка данных. Теперь у нас есть представление о статистических деталях наших данных. Следующим шагом является разделение данных на «атрибуты» и «метки». Атрибуты являются независимыми переменными, а метки–зависимыми переменными, значения которых должны быть предсказаны. В нашем наборе данных у нас есть только два столбца. Мы хотим предсказать процентный балл в зависимости от изученных часов. Поэтому наш набор атрибутов будет состоять из столбца «Часы», а метка–из столбца «Оценка». Чтобы извлечь атрибуты и метки, выполните следующий сценарий:

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

Атрибуты хранятся в переменной X . Мы указали «-1» в качестве диапазона для столбцов, так как хотели, чтобы наш набор атрибутов содержал *все* столбцы, кроме последнего, который является «Баллами». Аналогично переменная *y* содержит метки. Мы указали 1 для столбца метки, так как индекс для столбца «Оценки» равен 1. Помните, что индексы столбцов начинаются с 0, причем 1 является вторым столбцом.

Теперь, когда у нас есть атрибуты и метки, следующий шаг–разделить эти данные на обучающие и тестовые наборы. Мы сделаем это с помощью встроенного метода Scikit-Learn *train_test_split*() :

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

Приведенный выше код разбивает 80% данных на обучающий набор, а 20% – на тестовый. Переменная *test_size* – характеризует долю тестового набора.

Обучение алгоритма. Мы разделили наши данные на обучающие и тестовые наборы, и теперь, наконец, пришло время обучить наш алгоритм. Выполните следующую команду:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

С помощью Scikit-Learn очень просто реализовать модели линейной регрессии, так как все, что вам действительно нужно сделать, это импортировать класс *LinearRegression*, создать его экземпляр и вызвать метод *fit*() вместе с обучающими данными.

Линейная регрессионная модель в основном находит лучшее значение для коэффициентов. Чтобы увидеть их значение, вычисленное алгоритмом линейной регрессии для нашего набора данных, выполните следующий код.

Чтобы восстановить свободный коэффициент:

```
print(regressor.intercept_)
```

Результирующее значение, которое вы видите, должно быть приблизительно 2.01816004143.

Для получения наклона (коэффициент x):

```
print(regressor.coef_)
```

Результат должен быть примерно 9.91065648.

Это означает, что на каждую единицу изменения в изученных часах изменение балла составляет около 9,91%. Или, проще говоря, если студент учится на один час больше, чем он ранее учился на экзамене, он может ожидать увеличения на 9,91% балла, достигнутого студентом ранее.

Прогнозирование. Теперь, когда мы обучили наш алгоритм, пришло время сделать некоторые прогнозы. Для этого мы воспользуемся нашими тестовыми данными и посмотрим, насколько точно наш алгоритм предсказывает процентный балл. Чтобы сделать прогнозы по тестовым данным, выполните следующий сценарий:

```
y_pred = regressor.predict(X_test)
```

y_pred – это массив *numpy*, содержащий все предсказанные значения для входных значений в серии *X_test*.

Чтобы сравнить фактические выходные значения для *X_test* с прогнозируемыми значениями, выполните следующий сценарий:

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df
```

Вывод выглядит следующим образом:

```
0 20  16.884145

1 27  33.732261

2 69  75.357018

3 30  26.794801

4 62  60.491033
```

Хотя наша модель не очень точна, предсказанные проценты близки к реальным.

Примечание. Значения в приведенных выше столбцах могут отличаться в вашем случае, поскольку функция *train_test_split* случайным образом разбивает данные на тестовые и проверочные, и ваши разбиения, скорее всего, отличаются от приведенных в этой статье.

Оценка алгоритма. Заключительным шагом является оценка производительности алгоритма. Этот шаг особенно важен для сравнения того, насколько хорошо различные алгоритмы работают с конкретным набором данных. Для регрессионных алгоритмов обычно используются три оценочные метрики:

1. Средняя абсолютная ошибка (MAE) – это среднее значение абсолютного значения ошибок. Он рассчитывается как:

$$\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|$$

2. Среднеквадратичная ошибка (MSE) – это среднее значение квадратов ошибок и вычисляется как:

$$\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|^2$$

3. Среднеквадратичная ошибка (RMSE) – это квадратный корень из среднего значения квадратов ошибок:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}|Actual - Predicted|^2}$$

К счастью, нам не нужно выполнять эти вычисления вручную. Библиотека Scikit-Learn поставляется с предустановленными функциями, которые могут быть использованы для определения этих значений для нас.

Давайте найдем значения этих метрик, используя наши тестовые данные. Выполните следующий код:

```
from sklearn import metrics
print('Mean Absolute Error:',
metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:',
metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

Вывод будет выглядеть примерно так:

```
Mean Absolute Error: 4.183859899
Mean Squared Error: 21.5987693072
Root Mean Squared Error: 4.6474476121
```

Вы можете видеть, что значение среднеквадратичной ошибки составляет 4,64, что составляет менее 10% от среднего значения процентов всех студентов, т.е. 51,48. Это означает, что наш алгоритм проделал достойную работу.

Задание. Получить у преподавателя текстовый файл с данными (датасет), провести предварительную обработку данных, обучить модель линейной (и полиномиальной) регрессий, предсказать значение переменной, зависимой от признаков, рассчитать метрики (параметры качества модели).

Порядок работы

1. Прочитать набор данных, соответствующий варианту (либо использовать обучающий набор данных по согласованию с преподавателем);
2. Получение информации о датасете, проверка отсутствующих значений, выявить зависимую переменную и признаки, визуализировать данные зависимости на графиках;
3. Разделение на тестовую и проверочные выборки;
4. Обучение модели;
5. Предсказание значения;
6. Расчет метрик модели.

ВАРИАНТЫ ЗАДАНИЙ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

ЛАБОРАТОРНАЯ РАБОТА № 1 РАБОТА С МАТРИЦАМИ В СИСТЕМЕ MATLAB

Вариант № 1

1. Задать вектор X из 6 элементов случайным образом в диапазоне значений от –5 до 5. Изменить 3-й элемент на 0. Для вектора X найти минимальный элемент и его порядковый номер.

2. Задать две матрицы A и B размерностью 5 x 5 с клавиатуры. Найти определитель матрицы A, произведение матриц A и B, а также разделить поэлементно A на B.

Вариант № 2

1. Задать вектор X, элементы которого представляют собой арифметическую последовательность от $\pi/16$ до π с шагом $\pi/16$, и вектор Y – от $\pi/8$ до 2π с шагом $\pi/8$. Найти произведение максимального элемента вектора X на минимальный элемент вектора Y. Найти вектор попарных произведений элементов векторов X и Y.

2. Задать матрицу A размерностью 6×7 случайным образом в диапазоне значений от 1 до 10. Транспонировать матрицу A и выполнить сортировку столбцов матрицы A^T по возрастанию. Изменить 3-й элемент 4-й строки на 1.

Вариант № 3

1. Задать вектор X из 8 значений с клавиатуры. Для вектора X найти максимальный элемент и его порядковый номер. Преобразовать вектор-строку X в вектор-столбец.

2. Задать матрицу A размерностью 4×4 с клавиатуры. Прибавить к ней единичную матрицу. Возвести все элементы полученной матрицы в квадрат. Обнулить 4-й элемент 3-й строки полученной матрицы.

Вариант № 4

1. Задать вектор Y из 8 элементов случайным образом в диапазоне значений от –5 до 0. Увеличить 4-й элемент вектора Y на 1. Выполнить сортировку элементов вектора и найти произведение его элементов.

2. Задать две матрицы A и B размерностью 5×5 с клавиатуры. Найти определитель матрицы B и максимальный элемент для обеих матриц. Найденное значение определителя матрицы B поместить на место 2-го элемента 3-й строки матрицы A.

Вариант № 5

1. Задать вектор X, элементы которого представляют собой арифметическую последовательность от 1 до 10 с шагом 1, и вектор Y – от 0,5 до 5 с шагом 0,5. Найти вектор попарных произведений упорядоченных векторов X и Y.

2. Задать матрицу A размерностью 5×5 случайным образом в диапазоне значений от 1 до 2. Найти сумму максимальных элементов каждого столбца. Вычислить обратную матрицу. В обратной матрице обнулить первый и последний элементы.

Вариант № 6

1. Задать векторы X и Y, состоящие из 7 значений, с клавиатуры. Упорядочить вектор X-Y. Найти сумму элементов вектора X и произведение элементов вектора Y. Увеличить на 1 вторые элементы векторов X и Y.

2. Задать матрицу A размерностью 7×7 случайным образом в диапазоне значений от 0 до 100. Отнять от нее единичную матрицу. Найти определитель полученной матрицы.

Вариант № 7

1. Задать векторы X и Y размерностью 6 случайным образом в диапазоне значений от –10 до 10. Найти минимальный элемент для обоих векторов. Произвести сложение векторов.

2. Задать матрицу A размерностью 4×4 с клавиатуры. Обнулить первый и последний элементы 3-й строки матрицы A. Найти определитель матрицы A^T. Рассчитать сумму всех элементов матрицы A.

Вариант № 8

1. Задать вектор Y из 9 элементов случайным образом в диапазоне значений от –3 до 6. Изменить 8-й элемент на 2. Для вектора Y найти максимальный элемент и его порядковый номер.

2. Задать матрицу A размерностью 6×6 с клавиатуры. Обнулить второй элемент 4-й строки матрицы A. Найти определитель обратной матрицы.

Вариант № 9

1. Задать векторы X и Y размерностью 7 случайным образом в диапазоне значений от –1 до 1. Найти сумму всех элементов обоих векторов. Произвести попарное деление векторов.

2. Задать две матрицы A и B размерностью 4×4 с клавиатуры. Найти определители матриц A и B, произведение матриц A и B, а также сумму всех элементов обеих матриц.

Вариант № 10

1. Задать вектор X, элементы которого представляют собой арифметическую последовательность от $\pi/6$ до π с шагом $\pi/12$, и вектор Y – от 1 до 11 с шагом 1. Найти произведение минимального элемента вектора X на максимальный элемент вектора Y. Найти вектор суммы элементов векторов X и Y.

2. Задать матрицу A размерностью 5×5 случайным образом в диапазоне значений от –10 до 0. Обнулить второй элемент 3-й строки. Прибавить к матрице A единичную матрицу. Найти вектор средних значений каждого столбца полученной матрицы.

Вариант № 11

1. Задать векторы X и Y, состоящие из 7 значений, с клавиатуры. Упорядочить вектор попарных произведений X на Y. Найти произведение средних значений элементов векторов X и Y. Увеличить на 2 третьи элементы векторов X и Y.

2. Задать матрицу A размерностью 7×6 случайным образом в диапазоне значений от 10 до 100. Транспонировать матрицу A и найти минимальный элемент матрицы A^T. Изменить 2-й элемент 6-й строки на 3.

Вариант № 12

1. Задать вектор X из 9 значений с клавиатуры. Для вектора X найти минимальный элемент и его порядковый номер. Преобразовать вектор-строку X в вектор-столбец. Обнулить его 3-й элемент.

2. Задать матрицу B размерностью 6 x 6 случайным образом в диапазоне значений от 1 до 5. Найти произведения средних значений каждого столбца. Транспонировать матрицу.

Вариант № 13

1. Задать вектор X, элементы которого представляют собой арифметическую последовательность от 0 до 100 с шагом 10, и вектор Y – от 0 до 100π с шагом 10π. Найти вектор почленного деления вектора X на вектор Y.

2. Задать матрицу A размерностью 5×6 случайным образом в диапазоне значений от 0,5 до 1. Транспонировать матрицу A и найти вектор средних значений для матрицы A^T. Увеличить 2-й элемент 3-й строки на 1.

Вариант № 14

1. Задать вектор Y из 8 элементов случайным образом в диапазоне значений от –5 до 5. Увеличить 3-й элемент вектора Y на 3. Выполнить сортировку элементов вектора и найти сумму его элементов.

2. Задать матрицу A размерностью 5×5 с клавиатуры. Отнять от нее единичную матрицу. Возвести все элементы полученной матрицы в куб. Обнулить 3-й элемент 2-й строки полученной матрицы.

Вариант № 15

1. Задать с клавиатуры вектор X и вектор Y, состоящие из 10 элементов каждый. Найти произведение суммы элементов вектора X на среднее значение элементов вектора Y. Разделить поэлементно вектор X на вектор Y.

2. Задать матрицу A размерностью 6×6 случайным образом в диапазоне значений от –50 до 100. Найти определитель матрицы A. Удвоить первый и последний элементы 3-й строки матрицы.

Вариант № 16

1. Задать вектор Y из 8 элементов случайным образом в диапазоне значений от –30 до –10. Увеличить первый элемент вектора на 2. Для вектора Y найти сумму, произведение и среднее значение его элементов.

2. Задать матрицу A размерностью 3×4 и матрицу B размерностью 4×3 с клавиатуры. Найти матрицу C = A·B. Рассчитать сумму и произведение всех элементов матрицы C. Найти определитель матрицы C.

Вариант № 17

1. Задать вектор Z из 7 значений с клавиатуры. Для вектора Z найти максимальный элемент и его порядковый номер. Транспонировать вектор Z. Обнулить 5-й элемент вектора Z.

2. Задать матрицу B размерностью 6×5 случайным образом в диапазоне значений от 100 до 200. Транспонировать матрицу B. Найти сумму произведений столбцов матрицы. Добавить к матрице B единичную матрицу размерностью 6×5.

Вариант № 18

1. Задать векторы X и Y, состоящие из 8 значений, с клавиатуры. Упорядочить вектор попарных произведений X на Y. Найти произведение средних значений элементов векторов X и Y. Увеличить на 3 третьи элементы векторов X и Y.

2. Задать матрицу A размерностью 4×4 случайным образом в диапазоне значений от –3 до –2. Найти сумму минимальных элементов каждого столбца. Вычислить обратную матрицу. В обратной матрице обнулить первый и последний элементы.

ЛАБОРАТОРНАЯ РАБОТА № 2
РАБОТА С ГРАФИЧЕСКОЙ ПОДСИСТЕМОЙ MATLAB

Задание. Разбить графическое окно MatLab на 6 частей. В каждом из окон построить следующие графики:

а) $y = f(x)$, $z = f(x)$ в обычном масштабе; нанести сетку, легенду, название графика и осей координат (варианты заданий – в табл. 2.1);

б) $y = f(x)$ в логарифмическом масштабе по X и Y;

в) гистограмму значений функции z;

г) график функции ρ(φ) в полярных координатах, φ = [0…2π], Δφ = π/16;

д) график функции $y = f(x_1, x_2)$ в виде прозрачной сетки (mesh); нанести названия всех осей x_1, x_2 и y (варианты заданий – в табл. 2.2);

е) $y = f(x_1, x_2)$ в виде контурных линий (contour).

По графику сделать вывод о наличии экстремума (минимума или максимума) исследуемой функции.

2.1. Варианты заданий (окна а, б, в, г)

№	Функции $y = f(x)$ и $z = f(x)$	Диапазон и шаг изменения аргумента x	Функция ρ(φ)
1	$y = sin(x); z = sin(2x + 10)$	$x \in [0..3\pi]$, $\Delta x = \pi/6$	ρ(φ)=1-cos²φ sin²φ
2	$y = 2x + 10; z = \sqrt{x} + 2$	$x \in [0..10]$, $\Delta x = 0{,}5$	ρ(φ)=1-sin²φ
3	$y = e^{-x}; z = e^{x}$	$x \in [0..2]$, $\Delta x = 0{,}05$	ρ(φ)=1-cos²φ
4	$y = cos(3x); z = sin(3x)$	$x \in [0..2\pi]$, $\Delta x = \pi/8$	ρ(φ)=1-2sin²φ
5	$y = e^{x/10}; z = e^{(x/10-1)}$	$x \in [10..20]$, $\Delta x = 0{,}1$	ρ(φ)=1-cos²(φ/2) sin²φ
6	$y = \sqrt{10x}; z = 2\sqrt{5x}$	$x \in [10..20]$, $\Delta x = 0{,}1$	ρ(φ)=1-sin²(φ/2)
7	$y = 2x^2 - 3x + 1; z = x^2 - 2$	$x \in [0..10]$, $\Delta x = 0{,}1$	ρ(φ)=1-cos²(φ/2)
8	$y = \mathrm{tg}(x); z = \mathrm{tg}(x/2)$	$x \in [0..\pi/4]$, $\Delta x = \pi/64$	ρ(φ)=1-sin²(2φ)
9	$y = 1/\sqrt[3]{x}; y = 1/\sqrt[4]{x}$	$x \in [5..10]$, $\Delta x = 0{,}2$	ρ(φ)=φ sin²φ
10	$y = log(x); z = e^{-x/10}$	$x \in [1..6]$, $\Delta x = 0{,}2$	ρ(φ)=φ cos²φ
11	$y = \mathrm{tg}(x); z = \mathrm{th}(x)$	$x \in [0..\pi/4]$, $\Delta x = \pi/32$	ρ(φ)=1-φ sin²φ
12	$y = ln(x); z = \log_{10}(x)$	$x \in [1..2]$, $\Delta x = 0{,}05$	ρ(φ)=2sin²φ -1
13	$y = \mathrm{sh}(x); z = \mathrm{ch}(x)$	$x \in [0..4\pi]$, $\Delta x = \pi/16$	ρ(φ)=1-2cos²φ sin²φ
14	$y = \lvert x \rvert; z = 2x + 4$	$x \in [-10..10]$, $\Delta x = 0{,}1$	ρ(φ)=1-4sin²φ
15	$y = x^2 - 3x + 2; z = x^3 - x^2$	$x \in [-5..5]$, $\Delta x = 0{,}01$	ρ(φ)=φ²+cosφ
16	$y = \frac{e^{-x^2}}{x+2}; z = \frac{x}{6}$	$x \in [0..3]$, $\Delta x = 0{,}005$	ρ(φ)=φ-sinφ
17	$y = \frac{1}{x-2}; z = -x - 2$	$x \in [-10..0]$, $\Delta x = 0{,}01$	$\rho(\phi) = \sqrt{cos\,\phi}$
18	$y = \frac{1}{2x+4}; z = 1 + x$	$x \in [-10..10]$, $\Delta x = 0{,}02$	$\rho(\phi) = \sqrt[3]{sin\,\phi}$

2.2. Варианты заданий (окна д, е)

№	Функции $y = f(x_1, x_2)$	Диапазон и шаг изменения аргументов x_1 и x_2
1	$[1{,}5 - x_1(1 - x_2)]^2 + [2{,}25 - x_1(1 - x_2^2)]^2 + +[2{,}625 - x_1(1 - x_2^3)]$	$x_1 \in [0..5]$, $x_2 \in [-2..2]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
2	$x_1^3 + x_2^2 - 3x_1 - 2x_2 + 2$	$x_1 \in [-3..3]$, $x_2 \in [-3..3]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
3	$1 - 2x_1 - 2x_2 - 4x_1x_2 + 10x_1^2 + 2x_2^2$	$x_1 \in [-2..2]$, $x_2 \in [-2..2]$, $\Delta x_1 = 0{,}05$, $\Delta x_2 = 0{,}05$
4	$x_1^4 + x_2^4 + 2x_1^2x_2^2 - 4x_1 + 3$	$x_1 \in [-2..3]$, $x_2 \in [-2..2]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
5	$(x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	$x_1 \in [0..5]$, $x_2 \in [0..5]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
6	$x_1^3 + x_2^2 - 3x_1 - 2x_2 + 2$	$x_1 \in [-5..5]$, $x_2 \in [-5..5]$, $\Delta x_1 = 0{,}2$, $\Delta x_2 = 0{,}2$
7	$100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$x_1 \in [-2..3]$, $x_2 \in [-2..3]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
8	$[(e^{-0{,}1x_1} - e^{-0{,}1x_2}) - (e^{-0{,}1} - e^{-1})]^2$	$x_1 \in [-3..3]$, $x_2 \in [5..15]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}2$
9	$[1{,}5 - x_1(1 - x_2)]^2 + [2{,}25 - x_1(1 - x_2^2)]^2 + +[2{,}625 - x_1(1 - x_2^3)]$	$x_1 \in [-1..4]$, $x_2 \in [-3..3]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
10	$1 - 2x_1 - 2x_2 - 4x_1x_2 + 10x_1^2 + 2x_2^2$	$x_1 \in [-4..4]$, $x_2 \in [-3..3]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
11	$x_1^4 + x_2^4 + 2x_1^2x_2^2 - 4x_1 + 3$	$x_1 \in [-3..3]$, $x_2 \in [-2..3]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
12	$(x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	$x_1 \in [1..6]$, $x_2 \in [1..6]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
13	$100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$x_1 \in [-3..3]$, $x_2 \in [-3..3]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
14	$[(e^{-0{,}2x_1} - e^{-0{,}2x_2}) - (e^{-0{,}2} - e^{-2})]^2$	$x_1 \in [-4..4]$, $x_2 \in [3..15]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}2$
15	$(x_1 - 2)^2 + (x_2 - 5)^2$	$x_1 \in [0..5]$, $x_2 \in [0..10]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}1$
16	$x_1^2 + 4x_1x_2 + 5x_2^2$	$x_1 \in [-5..5]$, $x_2 \in [-5..5]$, $\Delta x_1 = 0{,}05$, $\Delta x_2 = 0{,}05$
17	$3(x_1 - 4)^2 + 5(x_2 + 3)^2$	$x_1 \in [0..10]$, $x_2 \in [-5..1]$, $\Delta x_1 = 0{,}1$, $\Delta x_2 = 0{,}05$
18	$\frac{x_1^2}{5} + \frac{x_2^2}{3}$	$x_1 \in [-5..5]$, $x_2 \in [-7..7]$, $\Delta x_1 = 0{,}01$, $\Delta x_2 = 0{,}05$

ЛАБОРАТОРНАЯ РАБОТА № 3
ПРОГРАММИРОВАНИЕ В MATLAB. ЦИКЛЫ И УСЛОВНЫЕ ОПЕРАТОРЫ

Задание 1. С помощью операторов цикла и условных операторов написать программу, предназначенную для вычисления функций (варианты заданий – в табл. 3.1).

3.1. Варианты задания

Вариант	Функция	Условие	Исходные данные	Диапазон и шаг изменения аргумента
1	$Y=\begin{cases} at^2\cdot \ln t \\ 1 \\ e^{at}\cdot \cos b\,t \end{cases}$	$1\le t\le 2$ $t<1$ $t>2$	$a=-0.5$ $b=2$	$t\in[0;3]$ $\Delta t=0.15$
2	$Y=\begin{cases} \pi x^2-\dfrac{7}{x^2} \\ ax^3+7\sqrt{x} \\ lg\left(x+7\sqrt{x}\right) \end{cases}$	$x<1.3$ $x=1.3$ $x>1.3$	$a=1.5$	$x\in[0.8;2]$ $\Delta x=0.1$
3	$W=\begin{cases} ax^2+bx+c \\ \dfrac{a}{x}+\sqrt{x^2+1} \\ \dfrac{(a+bx)}{\sqrt{x^2+1}} \end{cases}$	$x<1.2$ $x=1.2$ $x>1.2$	$a=2.8$ $b=-0.3$ $c=4$	$x\in[1;2]$ $\Delta x=0.05$
4	$Q=\begin{cases} \pi x^2-\dfrac{7}{x^2} \\ ax^3+7\sqrt{x} \\ ln\left(x+7\sqrt{\lvert x+a\rvert}\right) \end{cases}$	$x<1.4$ $x=1.4$ $x>1.4$	$a=1.65$	$x\in[0.7;2]$ $\Delta x=0.1$
5	$Y=\begin{cases} 1{,}5\cdot \cos^2 x \\ 1{,}8\cdot ax \\ (x-2)^2+6 \\ 3\cdot tgx \end{cases}$	$x<1$ $x=1$ $1<x\le 2$ $x>2$	$a=2.3$	$x\in[0.2;2.8]$ $\Delta x=0.2$
6	$W=\begin{cases} e^{-ax}\cdot \cos a\,x \\ x\cdot \sin a\,x \\ x\cdot \sqrt[3]{x-a} \end{cases}$	$x<a$ $x=a$ $x>a$	$a=2.5$	$x\in[1;5]$ $\Delta x=0.5$
7	$Q=\begin{cases} bx-lg\,b\,x \\ 1 \\ bx+lg\,b\,x \end{cases}$	$bx<1$ $bx=1$ $bx>1$	$b=1.5$	$x\in[0.1;1]$ $\Delta x=0.1$
8	$Y=\begin{cases} \sin x\cdot lg\,x \\ \cos^2 x \end{cases}$	$x>3.5$ $x\le 3.5$		$x\in[2;5]$ $\Delta x=0.25$
9	$F=\begin{cases} lg(x+1) \\ \sin^2 \sqrt{ax} \end{cases}$	$x>1$ $x\le 1$	$a=20.3$	$x\in[0.5;2]$ $\Delta x=0.2$
10	$Z=\begin{cases} \cos x+t\cdot \sin^2 x \\ \dfrac{1}{x}+\sqrt{x+t} \\ \dfrac{(\ln^3 x+x^2)}{\sqrt{x+t}} \end{cases}$	$x>0.5$ $x=0.5$ $x<0.5$	$t=2.2$	$x\in[0.2;2]$ $\Delta x=0.2$

Продолжение табл. 3.1

Вариант	Функция	Условие	Исходные данные	Диапазон и шаг изменения аргумента
11	$S=\begin{cases}\dfrac{a+b}{e^x+\cos x}\\ \dfrac{a+b}{x+1}\\ e^x+\sin x\end{cases}$	$x<2.8$ $2.8\le x<6$ $x\ge 6$	$a=2.6$ $b=-0.39$	$x\in[0;7]$ $\Delta x=0.5$
12	$Y=\begin{cases}a\cdot lg\,x+\sqrt[3]{\lvert x\rvert}\\ 2a\cdot\cos x+3x^2\end{cases}$	$x>1$ $x\le 1$	$a=0.9$	$x\in[0.8;2]$ $\Delta x=0.1$
13	$W=\begin{cases}\dfrac{a}{i}+bi^2+c\\ ai+bi^3\end{cases}$	$i<4$ $4\le i\le 6$ $i>6$	$a=2.1$ $b=1.8$ $c=-20.5$	$i\in[0;12]$ $\Delta i=1$
14	$Z=\begin{cases}a\cdot sin\left(\dfrac{i^2+1}{n}\right)\\ cos\left(i+\dfrac{1}{n}\right)\end{cases}$	$sin\left(\dfrac{i^2+1}{n}\right)>0$ $sin\left(\dfrac{i^2+1}{n}\right)<0$	$a=0.3$ $n=10$	$i\in[1;10]$ $\Delta i=1$
15	$Q=\begin{cases}\sqrt{at^2+b\cdot sin\,t+1}\\ at+b\\ \sqrt{at^2+b\cdot\cos t+1}\end{cases}$	$t<0.1$ $t=0.1$ $t>0.1$	$a=2.5$ $b=0.4$	$t\in[-1;1]$ $\Delta t=0.2$

Задание 2. Обработать на ЭВМ массив в соответствии с вариантом задания, указанного в табл. 3.2.

3.2. Варианты задания

Вариант	Имя матрицы и размеры	Действия	Условия и ограничения
1	$A(10,15)$	Вычислить и запомнить сумму и число положительных элементов каждого столбца матрицы. Результаты отпечатать в виде двух строк	$a_{ij}>0$
2	$A(N,M)$	Вычислить и запомнить суммы и числа элементов каждой строки матрицы. Результаты отпечатать в виде двух столбцов	$N<20$ $N<15$
3	$B(N,N)$	Вычислить сумму и число элементов матрицы, находящихся под главной диагональю и на ней	$N<12$
4	$C(N,N)$	Вычислить сумму и число положительных элементов, находящихся над главной диагональю	$c_{ij}>0$ $N<12$
5	$D(K,K)$	Записать на место отрицательных элементов матрицы нули и вывести ее на печать в общепринятом виде	$K<10$

Продолжение табл. 3.2

Вариант	Имя матрицы и размеры	Действия	Условия и ограничения
6	$D(10,10)$	Записать на место отрицательных элементов матрицы нули, а на место положительных – единицы. Вывести на печать нижнюю треугольную матрицу в общепринятом виде	
7	$F(N,M)$	Найти в каждой строке матрицы максимальный и минимальный элементы и поместить их на место первого и последнего элементов строки соответственно. Матрицу напечатать в общепринятом виде	$N < 20$ $M < 10$
8	$F(10,8)$	Транспонировать матрицу и вывести на печать элементы главной диагонали и диагонали, расположенной под главной. Результаты поместить в двух строках	
9	$N(10,10)$	Для целочисленной матрицы найти для каждой строки число элементов, кратных 5, и наибольший из полученных результатов	$n_{ij}/5 * 5 = n_{ij}$
10	$N(10,10)$	Из положительных элементов матрицы N сформировать матрицу $M(10, k\,max)$, располагая их в строках матрицы подряд, где $kmax$ – максимальное число положительных элементов строки матрицы N. Записать нули на место отсутствующих элементов. Отпечатать обе матрицы в общепринятом виде	
11	$P(N,N)$	Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали. Отпечатать полученную матрицу в общепринятом виде	$N < 15$
12	$R(K,N)$	Найти наибольший и наименьший элементы матрицы и поменять их местами	$K < 20$ $N < 10$
13	$S(5,6)$	Ввести исходные данные в первые 4 строки и 5 столбцов. Вычислить среднее арифметическое каждое столбца и записать его в 5-ю строку. Отпечатать полученную матрицу в общепринятом виде	
14	$T(N,M)$	Найти строки с наибольшей и наименьшей суммой элементов. Вывести на печать найденные строки и суммы их элементов	$M < 15$ $N < 20$
15	$V(15,10)$	Упорядочить по возрастанию элементы каждой строки матрицы. Отпечатать полученную матрицу в общепринятом виде	

ЛАБОРАТОРНАЯ РАБОТА № 4
АППРОКСИМАЦИЯ ПОЛИНОМАМИ ТАБЛИЧНО ЗАДАННЫХ ФУНКЦИЙ

Задание.

1. Аппроксимировать полиномом таблично заданную функцию $y = f(x)$. Степень полинома выбрать с учетом критерия:

$$\text{а)} \sum_{i=1}^{n} (y_i - y_{ia})^2 \to \min; \qquad \text{б)} \sum_{i=1}^{n} |(y_i - y_{ia})| \to \min,$$

где n – количество точек заданной функции; y_{ia} – значения функции аппроксимирующего полинома в точках x_i.

2. Найти корни полученного полинома.

3. Вывести график заданной функции и аппроксимирующей полиномиальной функции.

Варианты заданий даны в табл. 4.1.

4.1. Варианты заданий

№	Критерий	Данные $y = f(x)$									
1	а	x	1	2	3	4	5	6	7		
		y	2,45	2,56	2,35	2,48	2,51	2,66	2,98		
2	б	x	1	2	3	4	5	6	7	8	9
		y	10,56	10,98	10,54	10,78	10,32	10,58	10,91	10,25	10,56
3	а	x	11	22	33	44	55	66	77	88	
		y	121,2	132,6	129,6	130,5	127,6	126,4	135,1	128,2	
4	б	x	1,2	1,3	1,4	1,5	1,6	1,7			
		y	18,0	17,5	16,8	17,4	18,6	18,1			
5	а	x	1	2	3	4	5	6	7		
		y	121	131	141	125	135	138	129		
6	б	x	1,2	1,3	1,4	1,5	1,6	1,7	1,8	1,9	2,0
		y	4,52	4,35	4,28	4,68	4,55	4,70	4,29	4,56	4,85
7	а	x	5	5,5	6	6,5	7	7,5	8	8,5	
		y	21,3	22,1	21,9	22	22,6	23,1	22,4	21,8	
8	б	x	17	18	19	20	21	22	23		
		y	0,56	0,63	0,58	0,49	0,63	0,55	0,59		
9	а	x	14	15	16	17	18	19			
		y	121	124	126	128	130	124			
10	б	x	2,5	3,5	4,5	5,5	6,5	7,5	8,5	9,5	
		y	14,23	14,56	14,28	14,98	14,11	14,78	15,11	14,32	
11	а	x	120	130	140	150	160	170	180		
		y	6,33	6,52	6,42	6,81	6,12	6,35	6,45		
12	б	x	1	2	3	4	5	6	7	8	9
		y	55,23	54,96	53,89	56,36	51,69	54,32	55,31	52,89	54,56
13	а	x	0,5	1	1,5	2	2,5	3	3,5	4	
		y	5	7,3	6,4	8,1	4,9	6,3	5,6	7,2	

Продолжение табл. 4.1

№	Критерий	Данные $y = f(x)$								
14	б	x	71	75	79	83	87	91			
		y	2,36	2,45	2,8	2,12	2,65	2,74			
15	а	x	12	13	14	15	16	17	18	19	
		y	281	254	266	239	255	263	278	245	
16	б	x	65	70	75	80	85	90	95	100	
		y	16,36	15,98	16,21	16,54	16,83	17,23	15,99	16,87	
17	а	x	7	8	9	10	11	12	13	14	15
		y	7,41	7,23	7,56	7,12	7,9	8,1	6,98	7,56	7,49
18	б	x	11	11,5	12	12,5	13	13,5			
		y	33,3	12,4	0,5	44,5	12	27,6			

ЛАБОРАТОРНАЯ РАБОТА № 5

АППРОКСИМАЦИЯ ПЕРИОДИЧЕСКИМИ ФУНКЦИЯМИ И ИНТЕРПОЛЯЦИЯ

Задание.

1. *Изучение аппроксимации и интерполяции в MatLab.*

Аппроксимировать определенную заданным количеством точек на указанном интервале функцию периодической функцией (с использованием быстрого преобразования Фурье) на более мелкой сетке x.

Провести кубическую интерполяцию и интерполяцию сплайнами функции внутри указанного интервала на более мелкой сетке x.

Вывести зависимости заданной, аппроксимирующей и интерполяционных функций на одном графике. Варианты заданий даны в табл. 5.1.

5.1. Варианты заданий

№	Функция	Заданный интервал	Количество точек	Интервал интерполяции
1	$\frac{1}{\sin x \cdot \cos x}$	π/20, 2π/5	10	π/4, π/3
2	$tgx \cdot ctg2x$	π/20, 2π/5	8	π/4, π/3
3	$\frac{\sin x}{\sin 2x} + 1$	π/15, 3π/7	12	π/5, π/4
4	$4\sin^3 x + 2\sin 3x$	0, 2π	6	π/2, π
5	$tgx + \cos x$	π/20, 2π/5	8	π/5, π/4
6	$\sin 2x + \sin 4x + \sin 6x$	0, 2π	10	π,3π/2
7	$\frac{1}{\sin x} + \frac{2}{\cos x}$	π/20, 2π/5	15	π/5, π/4
8	$\cos x \cos 4x$	-π, π	8	0, π/2
9	$8\cos^3 x + 3\sin 3x$	-π, 2π	12	-π/2, 0
10	$\sin x \cos 3x$	π, 4π	7	2π, 5π/2

Продолжение табл. 5.1

№	Функция	Заданный интервал	Количество точек	Интервал интерполяции
11	$\frac{tg3x}{\sin x} - 1$	π/10, 9π/10	8	π/2, 2π/3
12	$\cos x + \cos 3x - \sin 2x$	π, 3π	9	2π, 5π/2
13	$tg2x - \cos x$	π/15, 3π/7	15	π/5, π/4
14	$2\sin^2 2x + 1$	0, 2π	10	0, 2π/3
15	$\cos x \sin 4x$	0, 2π	14	0, 2π/5
16	$\sin 3x - \cos x$	0, π	6	0, 2π/7
17	$\frac{\sin 2x}{\text{tg}x}$	π/15, 3π/7	7	π/15, 4π/15
18	$\cos x \cdot \sin x + \cos^2 3x$	-2π, 2π	12	-π/2, π

ЛАБОРАТОРНАЯ РАБОТА № 6
ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ В MATLAB

Задание. Вычислить значение определенного интеграла $\int_a^b f(x)dx$ методами трапеций и квадратур. Определить относительные погрешности численного интегрирования, сравнив полученные результаты с точным значением интеграла $F(b) - F(a)$, F – первообразная функции $f(x)$. Варианты заданий даны в табл. 6.1.

6.1. Варианты заданий

№	Подынтегральная функция $f(x)$	Первообразная $F(x)$	Интервал Интегрирования $[a, b]$
1	$\frac{1}{\sqrt{9+x^2}}$	$ln\left\|x + \sqrt{x^2+9}\right\|$	[0, 2]
2	$x^x(1 + ln\,x)$	x^x	[1, 3]
3	$\sqrt{x^2+16}$	$x\sqrt{x^2+16} + 8\,ln\left\|x + \sqrt{x^2+16}\right\|$	[2, 5]
4	$\frac{1+2x}{x^2(1+x^2)}$	$-\frac{1}{x} + 2\,ln\,x - ln(1+x^2) - \text{arctg}x$	[6, 8]
5	$e^x \cos^2 x$	$e^x(0{,}4 + 0{,}2\cos^2 x + 0{,}2\sin 2x)$	[0, π]
6	$\sqrt{1-x^2}$	$0.5\left(arcsin\,x + x\sqrt{1-x^2}\right)$	[–0,5, 0,5]
7	$x^3/(3+x)$	$x^3/3 - 3x^2/2 + 9x - 27\,ln(x+3)$	[1, 2]
8	$1/\cos^2 3x$	$\text{tg}3x/3$	[–π/8, 0]
9	$\frac{ln^2 x}{x}$	$\frac{ln^3 x}{3}$	[1, 4]
10	$5\sin x + \cos x$	$-5\cos x + \sin x$	[0, π/2]
11	$\sqrt{2x-1}$	$\sqrt{(2x-1)^3}/3$	[2, 5]
12	$\frac{1}{sin(1/x)\cdot x^2}$	$-ln\left(\frac{1}{sin(1/x)} - ctg(1/x)\right)$	[1, 2,5]

Продолжение табл. 6.1

№	Подынтегральная функция $f(x)$	Первообразная $F(x)$	Интервал Интегрирования $[a, b]$
13	$\frac{1}{(x+2)(x+4)}$	$-\frac{1}{2} ln \left\|\frac{x+4}{x+2}\right\|$	[1, 3]
14	$cos(ln\|x\|)$	$x[sin(ln\|x\|) + cos(ln\|x\|)]/2$	[3, 6]
15	$\frac{x^2+x+2}{x^4-5x+4}$	$\frac{1}{3} ln \left\|\frac{(x+1)^2(x-2)}{(x-1)(x+2)^2}\right\|$	[3, 7]
16	$x\text{arctg}x$	$\frac{\text{arctg}(x^2+1)-x}{2}$	[0, 3]
17	xe^{3x}	$\frac{e^{3x}(3x-1)}{9}$	[0.7, 3.1]
18	$\frac{x}{x^4+3x^2+2}$	$\frac{1}{2} ln \frac{-x^2-1}{x^2+2}$	[2, 4]

ЛАБОРАТОРНАЯ РАБОТА № 7
ПОИСК МИНИМУМА И НУЛЕЙ ФУНКЦИИ

Задание 1. Найти минимум функции на заданном промежутке. Построить график функции (см. табл. 7.1).

Таблица 7.1

№	Функция одной переменной	Интервал поиска	Функция нескольких переменных	Область поиска		
				x_1	x_2	x_3
1	$x^3 - x^2 + 2x$	–2,4	$100(x_2 - x_1^2)^2 + (4 - x_1)^2$	–2,5	0,10	
2	$sin(x - 0.8) - cos\, 2\, x$	–0,3, 0,7	$7x_1^2 + 4x_2^2 + (x_3^2 + 10)$	–1,1	–2,2	–3,3
3	$0.3x^2e^{-x} - 2$	–1,1	$3x_1^2 + 6(x_2^2 - 3) + 18(x_3^2 + 1)$	–1,1	–2,2	–3,3
4	$x^3 - 2x + 10$	1,5	$6x_1^2 + x_2^2 + (x_3^2 - 11) - 4x_2x_3$	2,5	–4,6	–1,2
5	$e^{-x} - 10\, sin\, 2\, x$	0,1	$16(x_2 - x_1^2)^2 + (x_2 - x_1)^2$	–5,5	0,10	
6	$(x-2)^2 + 1 + e^{-x}$	1,4	$15x_1^2 + 7x_2^2 + 5(x_3 + 1.5)^2$	–2,2	–2,2	–2,2
7	$-sin\, x^2 - (x-1)^2 + 2$	0,2, 2,9	$(2x_1 + x_2)^2 - (x_2 + 4)^2$	–3,3	–5,5	
8	$x^3 - x$	–1,4	$(x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2$	–1,2	–1,4	–1,6
9	$2x - 3\, ln\, x - 3$	0, 1,6	$40(x_1 - x_2^2)^2 + (14 - x_2)^2$	150, 250	2,20	
10	$2e^{-x} + 4(1-x)(2-x)$	1,5	$3x_1^2 + 5(x_2^2 - 2)$	–2,2	0,3	
11	$2x^3 - 6x + 3$	0, 2,5	$10(x_2 - x_1^2)^2 + (9 - x_2)^2 + 3$	0,5	0,20	
12	$x^2 - 5x + 10$	–1,3	$(x_2 - x_1)^2 + (2x_3 - x_2)^2$	–1,1	–10,0	–1,1
13	$e^{-2x} - cos(x + 0.5)$	0,1	$4x_1^2 + x_3^2 + (x_2^2 - 18)$	–2,2	0,5	–2,2
14	$sin\, 2\, x + cos\, x$	π/2,π	$4x_1^2 + 4x_1x_2 + x_2^2 + (x_1 - 3)^2$	0,5	–10,0	

Задание 2. Определить приближенно графическим способом все корни уравнения на заданном промежутке и уточнить их с заданной точностью. Варианты заданий в табл. 7.2.

7.2. Варианты заданий

№	Функция	Интервал	Точность
1	$2x - 3\,ln\,x - 3$	0.5,3	0.0001
2	$3x^2 - 8x - 4$	–2,5	0.000001
3	$cos\,2\,x + sin\,2\,x$	0, 2π	0.00001
4	$sin\,x\,cos\,x$	–π, 2π	0.00005
5	$3x - 4\,ln\,x - 5$	2,5	0.000005
6	$0.78e^x - e^{-x} - 1.7$	0,5	0.0001
7	$8x^2 - 12x - 5$	–1,4	0.0001
8	$\frac{1}{sin\,x + 0.1} + cos\,4\,x$	0, 0.9π	0.00001
9	$3 - 2x + sin\,2\,x - ln(1 + x)$	0,6	0.000015
10	$0.4 + arctg\sqrt{x} - x$	1,2	0.0001
11	$x + sin(x^{0.32} + 1)$	0.2,3.5	0.0001
12	$\frac{1}{sin\,x + 0.05} + sin\,3\,x$	0, 0.75π	0.00001
13	$8x^2 - 12x - 5$	–2,4	0.0001
14	$arccos\,x - \sqrt{1 - 0.3x^2}$	0,1	0.0001
15	$cos\,x + sin(x/2)$	0, 5π	0.00001

ЗАКЛЮЧЕНИЕ

Приведенные в учебном пособии методические рекомендации по выполнению практических заданий, рассматривают возможности применения для обработки данных системы автоматизации математических расчетов MATLAB, а также высокоуровневый язык программирования Python. Работы посвящены широкому спектру вопросов, начиная с основ работы в среде MATLAB, – матрицы и графические средства, программирование, статистическая обработка данных и фильтрация сигналов, обработка изображений, а также работа с данными на языке Python. Довольно подробно изложена теория по данным вопросам, приведены примеры, что позволяет достаточно глубоко освоить материал и применять его на практике. Также в пособии приведены задания к лабораторным работам по обработке данных в среде MATLAB.

Изучение материала пособия позволит повысить уровень практических навыков студентов и возможность применения материалов этого пособия при самостоятельной подготовке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Балабанов, П. В. Система автоматизированных расчетов MATLAB : лабораторный практикум / П. В. Балабанов, Н. А. Конышева, Д. А. Любимова. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2015 – 80 с. – URL : https://clck.ru/36XZ6f (дата обращения: 13.05.2024).
2. Алибеков, И. Ю. Теория вероятностей и математическая статистика в среде MATLAB : учебное пособие для вузов / И. Ю. Алибеков. – 2-е изд., стер. – СПб. : Лань, 2021. – 184 с. – Лань : электронно-библиотечная система. – URL : https://e.lanbook.com/book/152661 (дата обращения: 13.05.2024).
3. Ревинская, О. Г. Символьные вычисления в MatLab : учебное пособие для вузов / О. Г. Ревинская. – СПб. : Лань, 2020. – 528 с. – Лань : электронно-библиотечная система. – URL : https://e.lanbook.com/book/149344 (дата обращения: 13.05.2024).
4. Тихомирова, Л. В. Автоматизация математических расчетов в системе MathCAD : учебное пособие / Л. В. Тихомирова. – Комсомольск-на-Амуре : КНАГУ, 2018. – 92 с. – Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/151715 (дата обращения: 13.05.2024).
5. Дьяконов, В. П. MATLAB 7.*/R2006/R2007: Самоучитель [Электронный ресурс]/ В. П. Дьяконов. – М. : ДМК-Пресс, 2009. – 768 с. – URL : http://e.lanbook.com/books/element.php?pl1_cid=25&pl1_id=1178 (дата обращения: 13.05.2024).
6. Scikit-learn documentation. Machine Learning in Python. – URL : https://scikit-learn.org/stable/modules/classes.html (дата обращения: 13.05.2024)
7. Pandas documentation. – URL : https://pandas.pydata.org/pandas-docs/stable/reference/index.html (дата обращения: 13.05.2024)).
8. Маккинни, У. Python и анализ данных. Первичная обработка данных с применением pandas, NumPy и Jupiter : справочник / У. Маккинни ; пер. с английского А. А. Слинкина. – 3-е изд. – М. : ДМК Пресс, 2023. – 536 с. – Лань : электронно-библиотечная система. – URL : https://e.lanbook.com/book/348086 (дата обращения: 13.05.2024).
9. Митина, О. А. Технологии и инструментарий машинного обучения : учебное пособие / О. А. Митина, В. В. Жаров. – М. : РТУ МИРЭА, 2023. – 203 с. – Лань : электронно-библиотечная система. – URL : https://e.lanbook.com/book/368633 (дата обращения: 13.05.2024).
10. Котельников, Е. В. Введение в машинное обучение и анализ данных : учебное пособие / Е. В. Котельников, А. В. Котельникова. – Киров : ВятГУ, 2023. – 68 с. – Лань : электронно-библиотечная система. – URL : https://e.lanbook.com/book/390698 (дата обращения: 13.05.2024).
11. Пылов, П. А. Основы работы с моделями машинного и глубокого обучения : учебное пособие / П. А. Пылов, Р. В. Майтак, А. В. Дягилева. – Вологда : Инфра-Инженерия, 2023. – 256 с. // Лань : электронно-библиотечная система. – URL : https://e.lanbook.com/book/346529 (дата обращения: 13.05.2024).

ОГЛАВЛЕНИЕ

Учебное электронное издание

БАЛАБАНОВ Павел Владимирович
ЛЮБИМОВА Дарья Александровна
ГРЕБЕННИКОВА Наталия Михайловна

МЕТОДЫ И ПРОГРАММНЫЕ СРЕДСТВА ОБРАБОТКИ ДАННЫХ

Практикум

Редактирование Е. С. Мордасовой
Графический и мультимедийный дизайнер Т. Ю. Зотова
Обложка, упаковка, тиражирование Е. С. Мордасовой

ISBN 978-5-8265-2803-7

Подписано к использованию 29.08.2024.
Тираж 50 шт. Заказ № 93

Издательский центр ФГБОУ ВО «ТГТУ»
392000, г. Тамбов, ул. Советская, д. 106, к. 14
Тел./факс (4752) 63-81-08.
E-mail: izdatelstvo@tstu.ru