СПРАВОЧНЫЕ МАТЕРИАЛЫ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ОСНОВЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ»

ISBN 978-985-477-931-7

9 789854 779317

МГПУ им. И.П. Шамякина

Министерство образования Республики Беларусь

Учреждение образования
«Мозырский государственный педагогический университет
имени И. П. Шамякина»

СПРАВОЧНЫЕ МАТЕРИАЛЫ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ «ОСНОВЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ»

МГПУ им. И.П. Шамякина

Мозырь
МГПУ им. И. П. Шамякина
2024

УДК 519.85(075.8)
ББК 22.18я73
С74

Составитель
А. В. Макаревич, кандидат физико-математических наук, доцент,
доцент кафедры теоретической физики и прикладной информатики
УО МГПУ им. И. П. Шамякина

Рецензенты:
кандидат физико-математических наук, доцент, декан факультета физики
и информационных технологий ГГУ им. Ф. Скорины
А. Л. Самофалов;
кандидат физико-математических наук, доцент,
заместитель заведующего ЦДС Института физики НАН Беларуси
П. И. Ропот

Печатается по решению редакционно-издательского совета учреждения образования
«Мозырский государственный педагогический университет имени И. П. Шамякина»

Справочные материалы к выполнению лабораторных работ по учебной дисциплине «Основы математического моделирования» / сост. А. В. Макаревич. – Мозырь : МГПУ им. И. П. Шамякина, 2024. – 76 с.
С74
ISBN 978-985-477-931-7.

Справочные материалы адресованы студентам специальности 6-05-0533-04 «Компьютерная физика» и предназначены в качестве вспомогательных к выполнению лабораторных работ по дисциплине «Основы математического моделирования». Содержат систематически подобранный материал, касающийся основ работы в пакете Matlab.

УДК 519.85(075.8)
ББК 22.18я73

ISBN 978-985-477-931-7

© Макаревич А. В., составление, 2024
© УО МГПУ им. И. П. Шамякина, 2024

МГПУ им. И.П. Шамякина

ОГЛАВЛЕНИЕ

МГПУ им. И.П. Шамякина

ПРЕДИСЛОВИЕ

Издание посвящено знакомству с пакетом Matlab, используемому для решения различных задач, включая математические вычисления, работу с массивами, построение двумерных и трехмерных графиков функций, численное решение обыкновенных дифференциальных уравнений и их систем. В качестве справочного материала рассмотрены примеры использования Matlab для практической реализации перечисленных задач с приведением соответствующего программного кода и графических иллюстраций, способствующих наглядности и доступности усвоения представленной информации.

Материал издания оформлен там образом, что учащиеся, работая с ним, могут получить не только необходимые теоретические сведения, но и соответствующие практические навыки. Для закрепления теоретического материала студентам предлагается самостоятельно выполнить практические задания, которые по каждой рассматриваемой теме вынесены в отдельные приложения.

Издание главным образом ориентировано на организацию учебного процесса по специальности 6-05-0533-04 «Компьютерная физика» и предназначено для самостоятельной подготовки студентов к выполнению лабораторных работ по дисциплине «Основы математического моделирования». Также оно может быть полезно для реализации задач компьютерного моделирования физических систем, процессов и явлений в пакете Matlab.

МГПУ им. И.П. Шамякина

1. ВЫПОЛНЕНИЕ ПРОСТЕЙШИХ ВЫЧИСЛЕНИЙ В MATLAB

В настоящее время пакет прикладных программ Matlab представляет собой мощный инструмент, используемый для численного анализа, вычислений, моделирования, а также разработки алгоритмов и приложений. Он широко применяется в научных исследованиях, инженерии, математике и других дисциплинах, требующих обработки больших объемов данных и выполнения сложных вычислений. Ниже рассмотрим некоторые основы работы в этом пакете.

После запуска Matlab полезно задать его вид по умолчанию, применив последовательность команд Desktop → Desktop Layout → Default, а дальнейшие вычисления удобно производить в М-файле (File → New → M-File). Для сохранения М-файла следует использовать команды File → Save As…, в поле «Имя файла:» указать его необходимое имя и нажать «Сохранить».

Работу с М-файлом целесообразно начинать с использования в нем команд:

```
clc; clear all;
```

Функция **clc** очищает командное окно (Command Window) системы Matlab, а функция **clear all** удаляет из памяти Matlab ранее сохраненные переменные при каждом выполнении М-файла.

Чтобы найти значение выражения

$$\frac{(\cos(8{,}16\pi)-\sin(3{,}15\pi))^2}{2\tan(5{,}6)-\tan(3{,}4)\cdot\cos(3{,}38\pi)}+\sqrt{\lg(15{,}7)}\cdot e^{-1/3}$$

необходимо ввести его в рабочую область М-файла

```
(cos(8.16*pi)-sin(3.15*pi))^2/...
(2*tan(5.6)-tan(3.4)*cos(3.38*pi))+...
sqrt(log10(15.7))*exp(-1/3)
```

и запустить вычисления, выбрав либо пункт Run с именем М-файла в меню Debug, либо нажав соответствующую пиктограмму на панели инструментов М-файла, либо нажав клавишу F5 на клавиатуре. При этом в командном окне программы (Command Window) будет отображен результат вычисления введенного выражения:

```
ans=
    -0.3726
```

МГПУ им. И.П. Шамякина

Для того чтобы результат вычисления не выводился в командное окно, в конце выражения необходимо поставить точку с запятой «`;`»:

```
(cos(8.16*pi)-sin(3.15*pi))^2/...
(2*tan(5.6)-tan(3.4)*cos(3.38*pi))+...
sqrt(log10(15.7))*exp(-1/3);
```

Чтобы найдите значения выражений

$$a = \frac{\dfrac{\tan(2{,}15)}{\ln(6{,}45)} - \sqrt{\dfrac{\log_2(5{,}8)}{\cos(3{,}4\pi)} + \dfrac{\sin(2{,}8\pi)}{\lg(1{,}6)}}}{\dfrac{\tan(2{,}15)}{\ln(6{,}45)} - \dfrac{\sin(2{,}8\pi)}{\lg(1{,}6)}},$$

$$b = \frac{\left(\dfrac{\sin(2{,}8\pi)}{\lg(1{,}6)}\right)^2 - \sqrt{\dfrac{\tan(2{,}15)}{\ln(6{,}45)}}}{\sqrt{\dfrac{\log_2(5{,}8)}{\cos(3{,}4\pi)}}},$$

можно, воспользовавшись присвоением переменных, значениям $\dfrac{\tan(2{,}15)}{\ln(6{,}45)}$, $\dfrac{\log_2(5{,}8)}{\cos(3{,}4\pi)}$ и $\dfrac{\sin(2{,}8\pi)}{\lg(1{,}6)}$ присвоить, соответственно, x, y и z:

```
x=tan(2.15)/log(6.45);
y=log2(5.8)/cos(3.4*pi);
z=sin(2.8*pi)/log10(1.6);
```

и далее выполнить непосредственно вычисления исходных выражений, записав в М-файле следующие команды:

```
a=(x-sqrt(y+z))/(x-z)
b=z^2-sqrt(x)/sqrt(y)
```

Выполнив компиляцию программы, в командном окне увидим результаты вычислений приведенных выражений:

```
a =
  -0.8202 + 0.6238i

b =
    7.9760
```

МГПУ им. И.П. Шамякина

Примечание – При необходимости можно «закомментировать» введенные в М-файл строки, выделив их с помощью мыши и нажав комбинацию клавиш Ctrl+R («раскомментировать» выделенные строки можно комбинацией клавиш Ctrl+T).

Ниже в таблице 1 приведены обозначения некоторых распространенных команд при работе с числами в Matlab.

Таблица 1 – Команды для работы с числами в Matlab

Команда	***Результат***	***Команда***	***Результат***
x+y	сложение чисел x и y	**acos(x)**	$\arccos(x)$
x-y	разность чисел x и y	**atan(x)**	$\mathrm{arctg}(x)$
x*y	произведение чисел x и y	**round(x)**	округление числа x
x/y	деление числа x на y	**fix(x)**	целая часть числа x
x^y	возведение числа x в степень y	**floor(x)**	округление x к ближайшему меньшему целому числу
abs(x)	модуль числа x	**ceil(x)**	округление x к ближайшему большему целому числу
sign(x)	знак числа x	**gcd(m,n)**	НОД(m, n)
sqrt(x)	корень квадратный из числа x	**lcm(m,n)**	НОК(m, n)
exp(x)	e^x	**primes(n)**	простые числа $\leq n$
log(x)	$\ln(x)$	**isprime(n)**	проверка числа n на простоту
log2(x)	$\log_2(x)$	**factor(n)**	разложение числа n на простые множители
log10(x)	$\lg(x)$	**factorial(n)**	$n!$
sin(x)	$\sin(x)$	**complex(a,b)**	комплексное число $a+bi$
cos(x)	$\cos(x)$	**real(z)**	действительная часть комплексного числа z
tan(x)	$\mathrm{tg}(x)$	**imag(z)**	мнимая часть комплексного числа z
asin(x)	$\arcsin(x)$	**angle(z)**	угол (аргумент) комплексного числа z

Для закрепления навыков простейших вычислений в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ А.

МГПУ им. И.П. Шамякина

2. РАБОТА С МАССИВАМИ. ВЕКТОР-СТОЛБЦЫ И ВЕКТОР-СТРОКИ

Все данные Matlab представляет в виде массивов (векторов и матриц), и даже отдельно взятое число в нем представляется, по сути, в виде матрицы размером 1×1. Поэтому очень важно правильно понять, как эффективно использовать массивы и работать с ними в Matlab, поскольку без этого затруднительно построение в нем графиков, решение задач линейной алгебры, статистики, обработки данных и многих других. Массивы представляют собой самые распространенные объекты языка программирования системы Matlab, поэтому ниже приведено описание основ работы с векторами и матрицами.

Допустим, необходимо вычислить сумму вектор-столбцов

$$a = \begin{pmatrix} 2,7 \\ 4,2 \\ 6,9 \end{pmatrix} \text{ и } b = \begin{pmatrix} 5,8 \\ 1,6 \\ 4,2 \end{pmatrix}.$$

Для хранения векторов в Matlab будут использоваться массивы **a** и **b**, для создания которых в М-файле необходимо ввести сначала массив **a**, применяя квадратные скобки и разделяя элементы вектор-столбца точкой с запятой «**;**», а затем ввести массив **b** таким же способом. В конце каждой строки также нужно поставить точку с запятой, чтобы не загромождать промежуточными данными командное окно. Для нахождения суммы векторов используется знак «**+**». Если результат необходимо вывести в командное окно, записав его, например, в массив **c**, то точку с запятой в конце ставить не следует:

```
a=[2.7; 4.2; 6.9];
b=[5.8; 1.6; 4.2];
c=a+b
```

После запуска вычислений в командном окне будет выведен следующий результат:

```
c =
    8.5000
    5.8000
    11.1000
```

Предположим, необходимо вывести третий элемент вектор-строки d = (0,6 6,3 9,4 1,7 5,2).

Для вычислений вектор-строк следует записать их также в квадратных скобках, но между значениями элементов поставить запятые или пробелы:

```
d=[0.6 6.3 9.4 1.7 5.2];
```

Для того чтобы вывести лишь один элемент из вектор-столбца или вектор-строки, необходимо написать имя массива и далее в круглых скобках указать номер того элемента, который необходимо вывести, например:

```
d(3)
```

После запуска вычислений в командном окне должен отображаться следующий результат:

```
ans=
    9.4000
```

Если необходимо заменить какой-то из элементов на другое значение, например, третий элемент массива **d** на значение 8,1, то после **d(3)** следует поставить знак присваивания и ввести новое значение:

```
d(3)=8.1
d=
    0.6000    6.3000    8.1000    1.7000    5.2000
```

Из элементов массива можно формировать новые массивы. Например, можно создать массив **e** из пятого, первого и третьего элементов массива **d**. Производится это следующим образом:

```
e=[d(5); d(1); d(3)]
e=
    5.2000
    0.6000
    8.1000
```

С элементами вектор-столбцов и вектор-строк можно выполнять и иные операции. Допустим, используя вектор-строку $f = (0{,}3\ 6{,}2\ 7{,}1\ 9{,}4\ 5{,}1\ 3{,}9\ 2{,}2)$, необходимо создать массив **f1**, заменив нулями элементы массива **f** с третьего по седьмой; создать массив **f2**, используя элементы массива **f** со второго по четвертый; составить массив **f3**, содержащий элементы **f**, кроме пятого, используя сцепление строк.

Для обращения к блокам последовательно расположенных элементов вектора или вектор-строки служит индексация при помощи знака двоеточия. Реализация указанных действий приведена ниже:

МГПУ им. И.П. Шамякина

```
f=[0.3 6.2 7.1 9.4 5.1 3.9 2.2];
f1=f;
f1(3:7)=0
f2=f(2:4)
f3=[f(1:4) f(6:7)]
```

После компиляции программы в командном окне должен отображаться следующий результат:

```
f1 =
    0.3000    6.2000    0    0    0    0    0
f2 =
    6.2000    7.1000    9.4000
f3 =
    0.3000    6.2000    7.1000    9.4000    3.9000
2.2000
```

Допустим, необходимо перемножить элементы вектор-столбца $g=\begin{pmatrix}3{,}6\\5{,}4\\1{,}7\\6{,}9\\2{,}2\\4{,}3\end{pmatrix}$, найти минимальный и максимальный элементы этого вектора, а также порядковый номер максимального элемента.

Перемножение элементов вектор-столбца или вектор-строки осуществляется при помощи функции **prod**:

```
g=[3.6; 5.4; 1.7; 6.9; 2.2; 4.3];
p=prod(g)
p =
   2.1572e+003
```

Для нахождения минимального и максимального значений из элементов вектора служат встроенные функции **min** и **max**:

```
m=min(g)
m =
1.7000

M=max(g)
M =
   6.9000
```

МГПУ им. И.П. Шамякина

Вывод порядкового номера максимального элемента осуществляется следующим образом:

```
[M,k]=max(g)
M =
    6.9000
k =
    4
```

Аналогично можно определить порядковый номер и минимального элемента.

Далее, например, нужно упорядочить следующую представленную вектор-строку h = (8,4 –6,3 2,5 –1,2 0,6 5,7):

а) по возрастанию ее элементов;

б) по убыванию ее элементов;

в) в порядке возрастания модулей ее элементов;

г) по возрастанию ее элементов с двумя выходными аргументами (это приведет к образованию массива индексов соответствия элементов упорядоченного и исходного массивов).

```
h=[8.4 -6.3 2.5 -1.2 0.6 5.7];
```

а) для упорядочения элементов вектора по возрастанию используется функция **sort**:

```
R1=sort(h)
R1 =
    -6.3000  -1.2000  0.6000  2.5000  5.7000  8.4000
```

б) чтобы упорядочить вектор-строку по убыванию, необходимо функцию **sort** записать со знаком минус и также указать саму вектор-строку со знаком минус:

```
R2=-sort(-h)
R2 =
    8.4000  5.7000  2.5000  0.6000  -1.2000  -6.3000
```

в) упорядочение элементов в порядке возрастания их модулей производится с привлечением команды **abs**:

```
R3=sort(abs(h))
R3=
    0.6000   1.2000   2.5000   5.7000   6.3000   8.4000
```

МГПУ им. И.П. Шамякина

г) упорядочение элементов по возрастанию с двумя выходными аргументами, последний из которых указывает номер элемента массива, выглядит следующим образом:

```
[R4, n]=sort(h)
R4=
   -6.3000  -1.2000  0.6000  2.5000  5.7000  8.4000
n=
   2   4   5   3   6   1
```

Также можно выполнять поэлементные действия с вектор-столбцами и вектор-строками. Допустим, необходимо выполнить математические операции с вектор-строками l = (9 2 –5 4) и m = (3 7 –6 1):

а) перемножить поэлементно вектор-строки;

б) возвести во вторую степень элементы вектор-строки l;

в) все элементы вектор-строки l возвести в степень, равную соответствующим элементам второй вектор-строки m;

г) разделить поэлементно l на m и m на l;

д) ко всем элементам вектор-строки m прибавить число 1,8, вычесть из результата вектор-строку l, поэлементно умноженную на число 3, и разделить поэлементно весь полученный результат на число 5.

Следует отметить, что операция «`.*`» приводит к поэлементному умножению векторов одинаковой длины, при помощи «`.^`» осуществляется поэлементное возведение в степень, а деление соответствующих элементов векторов одинаковой длины выполняется с привлечением «`./`». Сумма и разность при использовании поэлементных операций обозначаются стандартно, как «`+`» и «`-`» соответственно.

а)
```
l=[9 2 -5 4];
m=[3 7 -6 1];
S1=l.*m
S1 =

    27    14    30     4
```

б)
```
S2=l.^2
S2 =

    81     4    25    16
```

в)
```
S3=l.^m
S3 =

    729.0000    128.0000    0.0001    4.0000
```

г)

```
S4=l./m
S4 =
    3.0000    0.2857    0.8333    4.0000

S5=m./l
S5 =
    0.3333    3.5000    1.2000    0.2500
```

д)

```
S6=(m+1.8-l.*3)./5
S6 =
    -4.4400    0.5600    2.1600   -1.8400
```

Для закрепления навыков работы с вектор-столбцами и вектор-строками в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ Б.

МГПУ им. И.П. Шамякина

3. ДВУМЕРНЫЕ МАССИВЫ И МАТРИЦЫ

Предположим, необходимо выполнить следующие математические матричные операции:

а) найти сумму и разность матриц $A = \begin{pmatrix} 4 & 5 & -2 \\ -6 & -3 & 7 \end{pmatrix}$ и $B = \begin{pmatrix} 4 & -7 & 8 \\ 2 & 6 & 0 \end{pmatrix}$;

б) умножить матрицы A и $C = \begin{pmatrix} 3 & 5 & -2 \\ 7 & 2 & -8 \\ -4 & 6 & 0 \end{pmatrix}$;

в) полученную матрицу умножить на 5.

Перечисленные действия и их реализация в Matlab будут выглядить следующим образом:

а)
```
A=[4 5 -2; -6 -3 7];
B=[4 -7 8; 2 6 0];
S=A+B
S =
     8    -2     6
    -4     3     7

R=A-B
R =
     0    12   -10
    -8    -9     7
```

б)
```
C=[3 5 -2; 7 2 -8; -4 6 0];
P=A*C
P =
    55    18   -48
   -67     6    36
```

в)
```
T=P*5
T =
   275    90  -240
  -335    30   180
```

Для нахождения значения выражения $(A-B)C^2(A+B)^T$, где значок «T» обозначает транспонирование матрицы – операцию над матрицей, при которой строки и столбцы матрицы меняются местами (в Matlab она реализуется с применением значка «`'`»), можно использовать следующую запись:

МГПУ им. И.П. Шамякина

```
(A-B)*C^2*(A+B)'
ans =
        4328        -1820
       -4166         5505
```

Вычисление произведения матриц $\begin{pmatrix} 3 & 2 & -4 \end{pmatrix} \cdot \begin{pmatrix} 6 & 3 & 0 \\ -8 & 5 & -1 \\ -7 & 2 & -4 \end{pmatrix} \cdot \begin{pmatrix} -6 \\ 5 \\ 1 \end{pmatrix}$ выполняется следующим образом:

```
d=[3 2 -4];
E=[6 3 0; -8 5 -1; -7 2 -4];
f=[-6; 5; 1];
d*E*f
ans =
    -111
```

Ниже в таблице 2 приведены обозначения некоторых распространенных команд при работе с матрицами и массивами в Matlab.

Таблица 2 – Команды для работы с матрицами (массивами) в Matlab

Команда	*Результат*	*Команда*	*Результат*
A+B	сумма матриц A и B	**size(A)**	размеры матрицы A
A-B	разность матриц A и B	**size(A,1)**	число строк матрицы A
a*A	умножение числа a на матрицу A	**size(A,2)**	число столбцов матрицы A
A*B	произведение матриц A и B	**flipud(A)**	переворот матрицы A «вверх ногами»
A^n	возведение матрицы A в степень n	**fliplr(A)**	переворот матрицы A «задом наперед»
A.*B	покомпонентное произведение матриц A и B	**rot90(A)**	переворот матрицы A на 90° против часовой стрелки
A./B	покомпонентное деление матриц A и B	**det(A)**	определитель матрицы A
A.^B	покомпонентная степень матриц A и B	**inv(A)**	обратная матрица A^{-1}
A′	транспонированная матрица A^T	**diag(A)**	главная диагональ матрицы A

Для закрепления навыков работы с матрицами в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ В.

МГПУ им. И.П. Шамякина

4. БЛОЧНЫЕ МАТРИЦЫ

Блочные матрицы – это матрицы, составленные из подматриц (блоков). Соответствующие размеры блоков при этом должны совпадать.

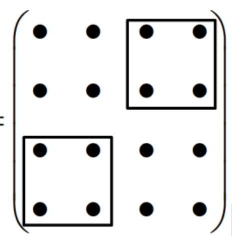
Допустим необходимо создать из квадратных матриц $A = \begin{pmatrix} 3{,}5 & 4 \\ 7 & 6 \end{pmatrix}$, $B = \begin{pmatrix} 8 & -1{,}2 \\ -3 & 5 \end{pmatrix}$, $C = \begin{pmatrix} -8 & 4 \\ 7 & 9 \end{pmatrix}$ и $D = \begin{pmatrix} 15 & 5 \\ -32 & 3 \end{pmatrix}$ размерностью 2×2 блочную матрицу $K = \left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$. В этом случае необходимая последовательность действий в Matlab будет иметь следующий вид:

```
A=[3.5 4; 7 6];
B=[8 -1.2; -3 5];
C=[-8 4; 7 9];
D=[15 5; -32 3];
K=[A B; C D]
K =
     3.5000    4.0000    8.0000   -1.2000
     7.0000    6.0000   -3.0000    5.0000
    -8.0000    4.0000   15.0000    5.0000
     7.0000    9.0000  -32.0000    3.0000
```

Для составления блочной матрицы $M = \left(\begin{array}{c|c} S & a \\ \hline b & 3{,}5 \end{array}\right)$, где $S = \begin{pmatrix} 5 & 0 \\ 6 & 4 \end{pmatrix}$, $a = \begin{pmatrix} 8 \\ 7 \end{pmatrix}$, $b = \begin{pmatrix} -6 & 3 \end{pmatrix}$ можно использовать команды:

```
S=[5 0; 6 4];
a=[8; 7];
b=[-6 3];
M=[S a; b 3.5]
M =
     5.0000         0    8.0000
     6.0000    4.0000    7.0000
    -6.0000    3.0000    3.5000
```

Если необходимо выделить указанные блоки из полученной матрицы K, а также 3-ю строку из матрицы M, создав при этом массивы **K1**, **K2** и вектор-строку **m**, то команды для реализации этих действий будут иметь следующий вид:

$K =$ 

МГПУ им. И.П. Шамякина

```
K1=K(1:2, 3:4);
K1 =

    8.0000   -1.2000

   -3.0000    5.0000

K2=K(3:4, 1:2)
K2 =

    -8     4

     7     9

m=M(3, :)
m =

   -6.0000    3.0000    3.5000
```

Чтобы удалить первую 1-ю строку и 3-й столбец матрицы *K* применяют команду **[]**:

```
K(1, :)=[];
K(:, 3)=[]
K =
     7     6     5
    -8     4     5
     7     9     3
```

Для создания прямоугольной матрицы *E* размером 3×4, заполненной нулями, используют функцию *zeros*:

```
E=zeros(3, 4)
E =
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Создание квадратной матрицы *F* размером 3×3, заполненной единицами, реализуется функцией *ones*:

```
F=ones(3)
F =
     1     1     1
     1     1     1
     1     1     1
```

МГПУ им. И.П. Шамякина

Для создания прямоугольной матрицы G размером 3×5, заполненной случайными числами, можно использовать функцию **rand**:

```
G=rand(3, 5)
G =
    0.8147    0.9134    0.2785    0.9649    0.9572
    0.9058    0.6324    0.5469    0.1576    0.4854
    0.1270    0.0975    0.9575    0.9706    0.8003
```

Для того чтобы создать квадратную матрицу H размером 4×4, у которой диагональные элементы являются элементами вектор-строки $r = (1\ 2\ 3\ 4)$, а все остальные элементы равны 0, можно применить команду **diag**:

```
r=[1 2 3 4];
H=diag(r)
H =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

Matlab также предполагает возможность создания более сложных блочных матриц и их сохранения, используя описанные выше команды и функцию **save**. Допустим, необходимо ввести и сохранить следующие матрицы L и N:

$$L = \begin{pmatrix} -2 & 0 & 0 & 0 & 0 & 5 & 5 & 5 & 5 & 5 \\ 0 & -2 & 0 & 0 & 0 & 5 & 5 & 5 & 5 & 5 \\ 0 & 0 & -2 & 0 & 0 & 5 & 5 & 5 & 5 & 5 \\ 0 & 0 & 0 & -2 & 0 & 5 & 5 & 5 & 5 & 5 \\ 0 & 0 & 0 & 0 & -2 & 5 & 5 & 5 & 5 & 5 \\ 2 & 2 & 2 & 2 & 2 & 1 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 0 & 1 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 0 & 0 & 1 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 & 0 & 0 & 0 & 1 & 0 \\ 2 & 2 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

МГПУ им. И.П. Шамякина

$$N = \begin{pmatrix} 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \end{pmatrix}.$$

В этом случае программный код может иметь следующий вид:

```
L=[-2*eye(5) 5*ones(5); 2*ones(5) eye(5)]
L =
   -2     0     0     0     0     5     5     5     5     5
    0    -2     0     0     0     5     5     5     5     5
    0     0    -2     0     0     5     5     5     5     5
    0     0     0    -2     0     5     5     5     5     5
    0     0     0     0    -2     5     5     5     5     5
    2     2     2     2     2     1     0     0     0     0
    2     2     2     2     2     0     1     0     0     0
    2     2     2     2     2     0     0     1     0     0
    2     2     2     2     2     0     0     0     1     0
    2     2     2     2     2     0     0     0     0     1

save('L.mat', 'L')

N=4*eye(7)+diag(ones(1,6), 1)+diag(ones(1, 6),-1)
N =
     4     1     0     0     0     0     0
     1     4     1     0     0     0     0
     0     1     4     1     0     0     0
     0     0     1     4     1     0     0
     0     0     0     1     4     1     0
     0     0     0     0     1     4     1
     0     0     0     0     0     1     4

save('N.mat', 'N');
```

МГПУ им. И.П. Шамякина

Для загрузки сохраненных массивов, используются следующие команды:

```
load('L.mat')
load('N.mat')
```

Отображение загруженных массивов можно увидеть в окне Workspace.

Подобные операции сохранения и последующей загрузки массивов могут быть полезны для работы с массивами данных, полученными в результате громоздких математических вычислений, требующих больших временных машинных затрат. Поэтому такие массивы можно отдельно единожды рассчитать, сохранить и в последующем загружать в необходимый момент реализации программного кода, не дожидаясь их повторного расчета, что позволяет значительно сэкономить время выполнения основной программы.

Для закрепления навыков работы с блочными матрицами в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ Г.

МГПУ им. И.П. Шамякина

5. ВИЗУАЛИЗАЦИЯ МАТРИЦ И ПОЭЛЕМЕНТНЫЕ ОПЕРАЦИИ НАД НИМИ

Рассмотрим поэлементные операции с матрицами на примере матрицы:

$$M = \begin{pmatrix} 2 & -3 & -5 \\ 6 & -2 & 4 \\ 3 & -1 & 1 \end{pmatrix}.$$

Сумма элементов матрицы по столбцам и по строкам выполняется с использованием команды **sum** следующим образом:

```
M=[2 -3 -5; 6 -2 4; 3 -1 1];
s1=sum(M)
s1 =
    11    -6     0

s2=sum(M, 2)
s2 =
    -6
     8
     3
```

Для сортировки элементов матрицы в порядке возрастания их столбцов и строк применяется команда **sort**:

```
MC=sort(M)
MC =
     2    -3    -5
     3    -2     1
     6    -1     4

MR=sort(M, 2)
MR =
    -5    -3     2
    -2     4     6
    -1     1     3
```

Определение максимальных и минимальных элементов по столбцам матрицы *M*, а также номеров этих элементов реализуется следующим образом:

МГПУ им. И.П. Шамякина

```
[mx1, n_mx1]=max(M)
mx1 =
     6    -1     4
n_mx1 =
     2     3     2

[mn1, n_mn1]=min(M)
mn1 =
     2    -3    -5
n_mn1 =
     1     1     1
```

Чтобы определить максимальные и минимальные элементы по строкам матрицы *M*, а также номера этих элементов, можно использовать приведенные ниже команды:

```
[mx2, n_mx2]=max (M, [], 2)
mx2 =
     2
     6
     3
n_mx2 =
     1
     1
     1

[mn2, n_mn2]=min(M, [], 2)
mn2 =
    -5
    -2
    -1
n_mn2 =
     3
     2
     2
```

Чтобы определить наибольшие и наименьшие элементы матрицы *M*, применяется следующий алгоритм:

```
MX=max(max(M))
MX =
     6

MN=min(min(M))
MN =
    -5
```

МГПУ им. И.П. Шамякина

Для создания матриц заданного вида возможно последовательное использование нескольких команд. Например, для создания квадратной матрицы L размера 5×5, состоящей из случайных целых неотрицательных чисел от 0 до 10, можно применить следующий алгоритм:

```
L=round(10*rand(5))
L =
     8     7     8     4     5
     7     0     7     4     4
     4     3     3     8     6
     7     0    10     8     7
     2     1     0     2     8
```

Создание квадратной матрицы K размера 8×8

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

реализуется последовательностью команд

```
K=[ones(4) diag(1:4); zeros(4)-5...
rot90(diag(-4:-1))]
K =
     1     1     1     1     1     0     0     0
     1     1     1     1     0     2     0     0
     1     1     1     1     0     0     3     0
     1     1     1     1     0     0     0     4
    -5    -5    -5    -5     0     0     0    -1
    -5    -5    -5    -5     0     0    -2     0
    -5    -5    -5    -5     0    -3     0     0
    -5    -5    -5    -5    -4     0     0     0
```

Используя заданные условия, также можно, например, легко заменить все значения матрицы, равные –5, на –10, положительные элементы матрицы на 7, а элементы, лежащие в диапазоне значений [–4; –1], на 10.

МГПУ им. И.П. Шамякина

```
K(K==-5)=-10;
K(K>0)=7;
K(K>=-4 & K<=-1)=10
K =
     7     7     7     7     7     0     0     0
     7     7     7     7     0     7     0     0
     7     7     7     7     0     0     7     0
     7     7     7     7     0     0     0     7
   -10   -10   -10   -10     0     0     0    10
   -10   -10   -10   -10     0     0    10     0
   -10   -10   -10   -10     0    10     0     0
   -10   -10   -10   -10    10     0     0     0
```

Чтобы найти сумму всех элементов последней полученной матрицы, достаточно приписать команды

```
S=sum(sum(K))
S =
    20
```

Для закрепления навыков работы с элементами матриц в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ Д.

МГПУ им. И.П. Шамякина

6. ПОСТРОЕНИЕ ДВУМЕРНЫХ ГРАФИКОВ ФУНКЦИЙ

Графические возможности системы Matlab являются мощными и разнообразными. Ниже уделим внимание рассмотрению вопросов построения и оформления различных графиков функций.

Построение графиков функций одной переменной в линейном масштабе осуществляется при помощи функции `plot`. В зависимости от входных аргументов функция `plot` позволяет строить один или несколько графиков в одних координатных осях, изменять цвет и стиль линий, добавлять маркеры на графики.

Пример вывода простейшего графика функции $y(x) = e^{-x}\sin(10x)$ с шагом 0,05 при использовании `plot` приведен ниже (рисунок 1).

```
x=0:0.05:1;
y=exp(-x).*sin(10.*x);
plot(x,y)
```

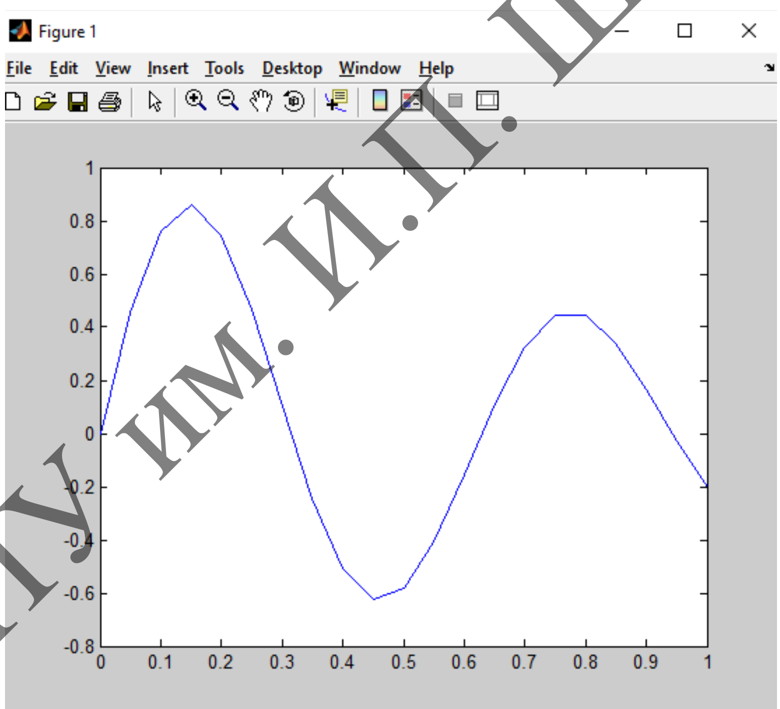


Рисунок 1 – Построение графика одной функции с использованием `plot`

Обратите внимание, что фактически мы строим график табличной функции, то есть зависимость одного массива данных **y** от другого **x**. Неудачный выбор шага по оси абсцисс может исказить реальную картину поведения кривой (на рисунке 1 она выглядит ломаной). Поэтому при построении графиков функций следует уделять повышенное внимание выбору шага вычисления.

МГПУ им. И.П. Шамякина

Сравнение нескольких функций легко производить, построив графики на одних координатных осях. Построение графиков функций $f(x)=e^{-0{,}1x}\sin^2 x$ и $g(x)=e^{-0{,}2x}\sin^2 x$ на отрезке $[-2\pi;2\pi]$ с шагом 0,01 представлено на рисунке 2.

```
x=-2*pi:0.01:2*pi;
f=exp(-0.1*x).*sin(x).^2;
g=exp(-0.2*x).*sin(x).^2;
plot(x,f,x,g)
```

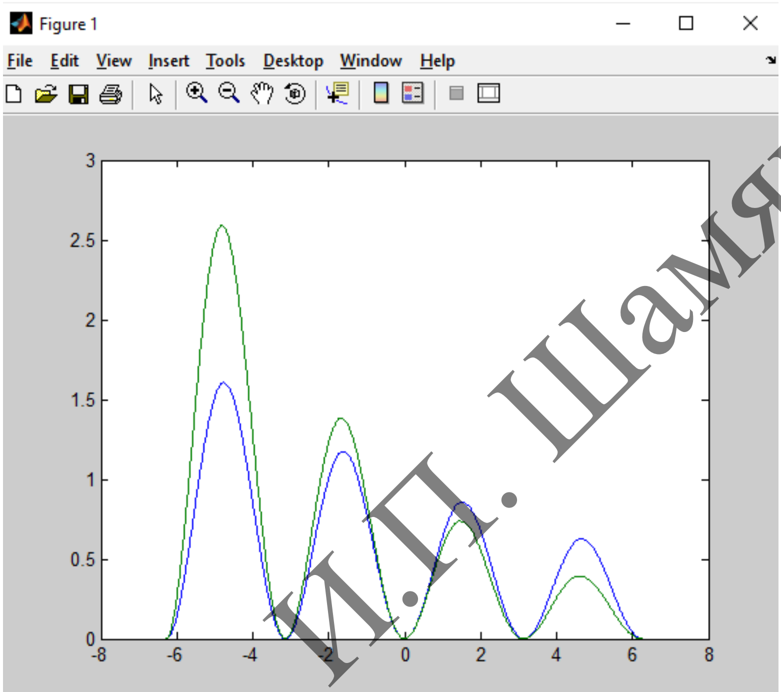


Рисунок 2 – Построение графиков двух функции с использованием `plot`

Функции необязательно должны быть определены на одном и том же отрезке. В этом случае при построении графиков Matlab выберет максимальный отрезок, содержащий остальные.

Иногда требуется сравнить поведение двух функций, значения которых сильно отличаются друг от друга. В таком случае график функции с небольшими значениями практически сливается с осью абсцисс, и визуально установить его вид не удается. В этой ситуации помогает функция **`plotyy`**, которая выводит графики в окно с двумя вертикальными осями, имеющими подходящий масштаб. Сравним, например, две функции $f(x)=x^{-3}$ и $h(x)=1000\cdot(x+0{,}5)^{-4}$, результат построения которых представлен на рисунке 3.

```
x=0.5:0.01:3;
f=x.^-3;
h=1000.*(x+0.5).^-4;
plotyy(x,f,x,h)
```

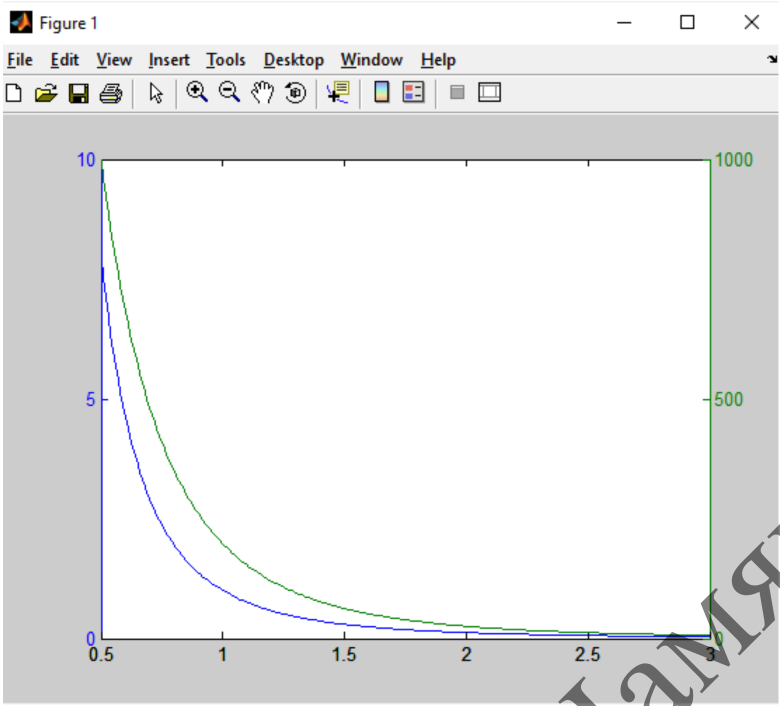


Рисунок 3 – Сравнение функций при помощи `plotyy`

В этом случае следует обратить внимание, что цвет графика совпадает с цветом соответствующей ему оси ординат.

Функция **`plot`** использует линейный масштаб по обеим координатным осям. Однако Matlab предоставляет пользователю возможность строить графики функций одной переменной в логарифмическом или полулогарифмическом масштабе.

Для построения графиков в логарифмическом и полулогарифмическом масштабах служат функции:

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

Аргументы **`loglog`**, **`semilogx`** и **`semilogy`** задаются в виде пары векторов значений абсцисс и ординат так же, как и для функции **`plot`**.

Построение, например, графиков функций $f(x)=\ln(0,5x)$ и $g(x)=\sin(\ln x)$ на отрезке [0,1; 10] в логарифмическом масштабе по оси *Ox* (рисунок 4) реализуется следующим образом:

```
x=0.1:0.01:10;
f=log(0.5.*x);
g=sin(log(x));
semilogx(x,f,x,g)
```

МГПУ им. И.П. Шамякина

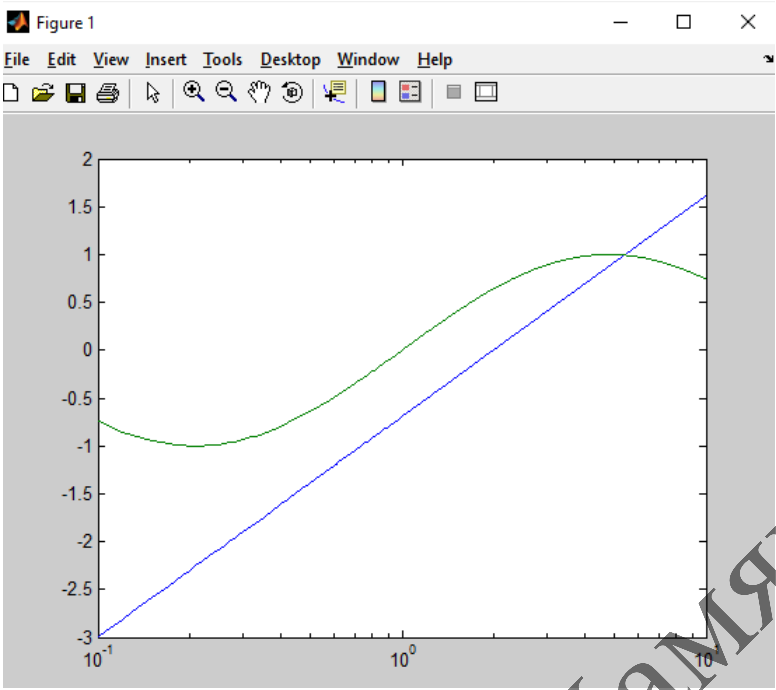


Рисунок 4 – Построение графиков двух функции с использованием `semilogx`

Функции **`loglog`** и **`semilogy`** вызываются аналогично.

Построенные графики функций должны быть максимально удобными для восприятия. Часто требуется нанести маркеры, изменить цвет линий, а при подготовке к монохромной печати – задать тип линии (сплошная, пунктирная, штрих-пунктирная и т. д.). Matlab предоставляет возможность управлять видом графиков, построенных при помощи **`plot`**, **`loglog`**, **`semilogx`** и **`semilogy`**, для чего служит дополнительный аргумент, помещаемый за каждой парой векторов. Этот аргумент заключается в апострофы и состоит из символов, которые определяют цвет, тип маркера и тип линии. В таблице 3 приведены возможные значения данного аргумента с указанием результата.

Таблица 3 – Свойства линии графика функции

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

МГПУ им. И.П. Шамякина

Например, для построения первого графика на рисунке 2 красными точечными маркерами без линии, а второго черными звездочками следует задать соответствующие команды в функции **plot** (рисунок 5).

```
x=-2*pi:0.1:2*pi;
f=exp(-0.1*x).*sin(x).^2;
g=exp(-0.2*x).*sin(x).^2;
plot(x,f,'r.',x,g,'k*')
```

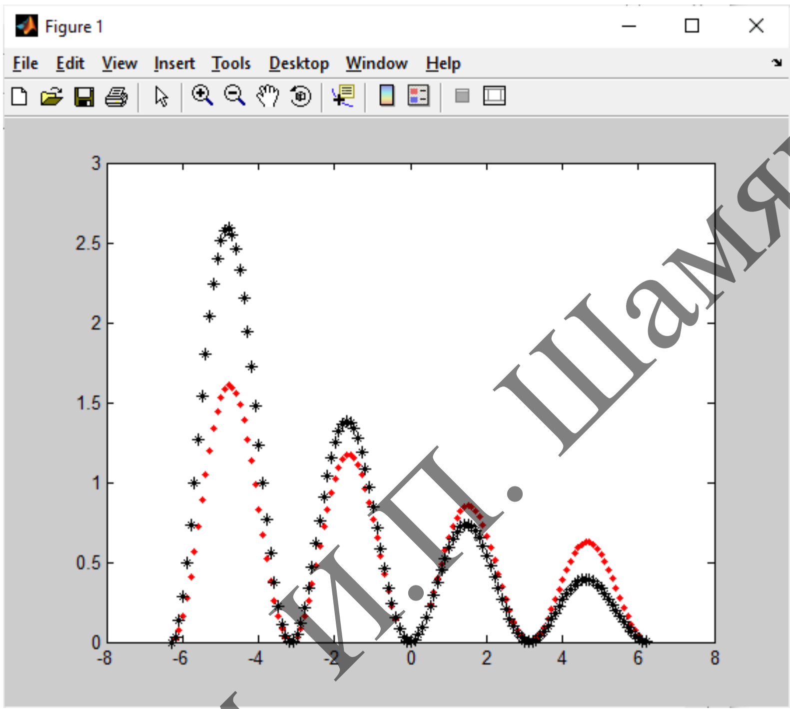


Рисунок 5 – Построение графиков двух функции с использованием plot, но при изменении свойств выводимых линий

Обратите внимание, что абсциссы маркеров совпадают со значениями аргумента, содержащимися в массиве **x**. Это не всегда хорошо, ведь для получения гладкой кривой требуется вычислить вектор значений функции в достаточно большом числе точек, что приводит к слишком частому расположению маркеров или даже их перекрытию. Простой прием позволяет поместить маркеры в заранее выбранные позиции. Строится два графика функции, один – сплошной линией, а второй – только маркерами для небольшого набора значений аргумента (рисунок 6).

```
x=-1:0.01:1;
y=sin(2*pi*x.^2);
xm=-1:0.2:1;
ym=sin(2*pi*xm.^2);
plot(x,y,'k',xm,ym,'ko')
```

МГПУ им. И.П. Шамякина

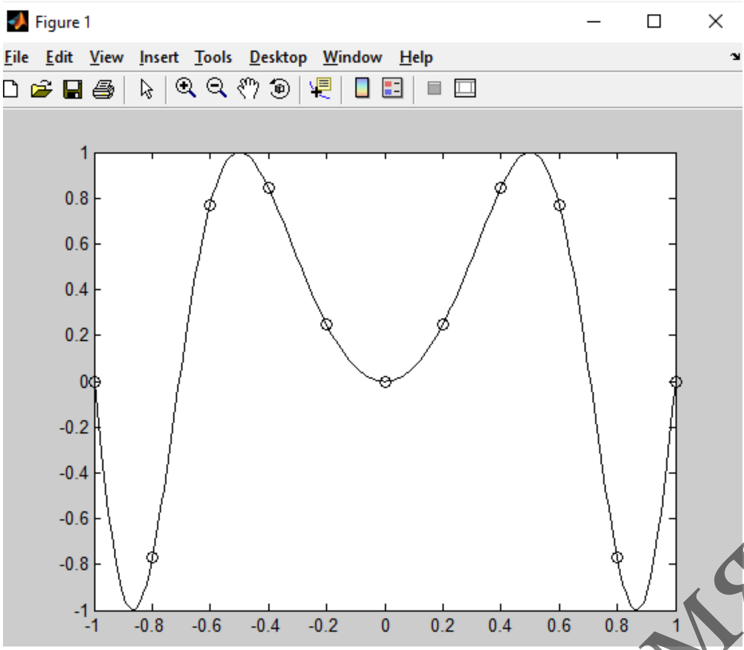


Рисунок 6 – Построение графика функции с использованием plot, но при выводе маркеров с большим шагом

Удобство использования графиков во многом зависит от дополнительных элементов оформления: координатной сетки, подписей к осям, заголовка и легенды. Такие возможности реализуются либо с помощью дополнительных параметров, задающих свойства объектов, либо с помощью вспомогательных команд и функций. Перечислим основные из них. Координатная сетка наносится командой **grid on**, функции **xlabel** и **ylabel** служат для размещения подписей к осям, а **title** – для заголовка. Если нужно сопроводить график легендой, следует использовать функцию **legend**. Все перечисленные команды применимы к графикам как в линейном, так и в логарифмическом и полулогарифмическом масштабах. Следующие команды выводят графики изменения суточной температуры, изображенные на рисунке 7, которые снабжены всей необходимой информацией.

```
time=[0 4 7 9 10 11 12 13 13.5 14 14.5 15 16 17 ...
18 20 22];
temp1=[14 15 14 16 18 17 20 22 24 28 25 20 16 13 ...
13 14 13];
temp2=[12 13 13 14 16 18 20 20 23 25 25 20 16 12 ...
12 11 10];
plot(time, temp1, 'ro-', time, temp2, 'go-')
grid on
title('Суточные температуры', 'Fontname',...
'Arial Unicode MS')
xlabel('Время (часы)', 'Fontname',...
'Arial Unicode MS')
ylabel('Температура (\circ C)', 'Fontname',...
'Arial Unicode MS')
legend('10 мая', '11 мая', 'Fontname',...
'Arial Unicode MS')
```


МГПУ им. И.П. Шамякина

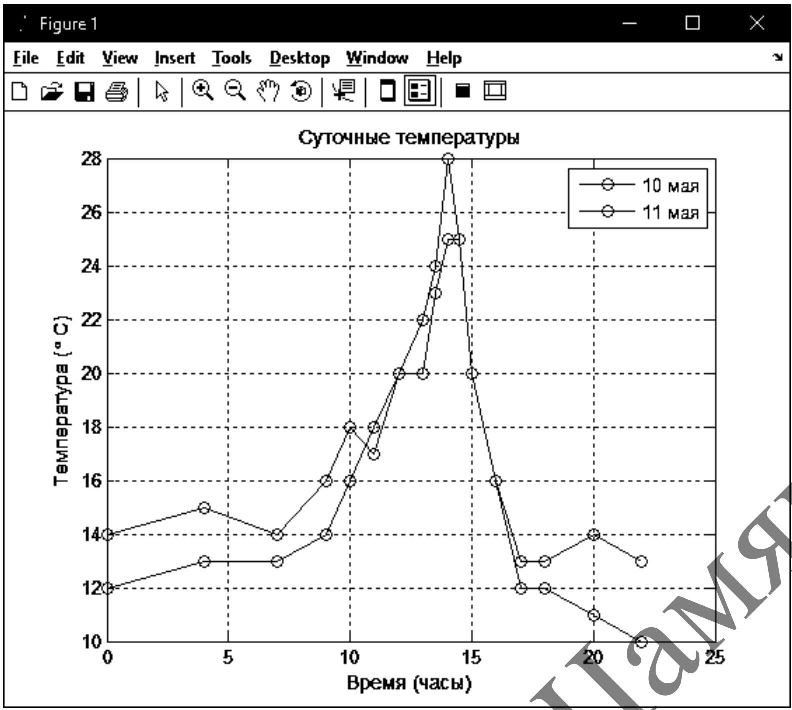


Рисунок 7 – Графики изменения суточной температуры, снабженные информацией, с использованием дополнительных параметров их вывода

Примечание – Символы кириллицы могут некорректно отображаться на графиках в Matlab. Один из способов решения этой проблемы заключается в изменении шрифта для выводимых на графиках подписей. В данном случае установлен шрифт Arial Unicode MS.

При размещении легенды следует учесть, что порядок и количество аргументов команды **`legend`** должны соответствовать линиям на графике. Последним дополнительным аргументом **`legend`** может быть положение легенды в графическом окне:

`–1` – вне графика в правом верхнем углу графического окна;

`0` – выбирается лучшее положение в пределах графика так, чтобы как можно меньше перекрывать сами графики;

`1` – в верхнем правом углу графика (это положение используется по умолчанию);

`2` – в верхнем левом углу графика;

`3` – в нижнем левом углу графика;

`4` – в нижнем правом углу графика.

Кроме того, имеется и другая возможность для указания положения легенды за счет привлечения ее свойства **`Location`**. Восемнадцать допустимых значений этого свойства приведены в справочной системе Matlab, вызываемой с использованием Help → Product Help, далее необходимо воспользоваться индексным поиском по слову legend.

МГПУ им. И.П. Шамякина

Для построения функций, заданных параметрически, следует сперва сгенерировать вектор значений аргумента. Затем необходимо вычислить значения функций и записать их в векторы, которые и надо использовать в качестве аргументов **plot**. График функции $x(t) = 0,5\sin t$, $y(t) = 0,7\cos t$ для $t \in [0; 2\pi]$ (эллипс), приведенный на рисунке 8, получается при помощи следующих команд:

```
t=0:0.01:2*pi;
x=0.5*sin (t);
y=0.7*cos (t);
plot(x,y)
```

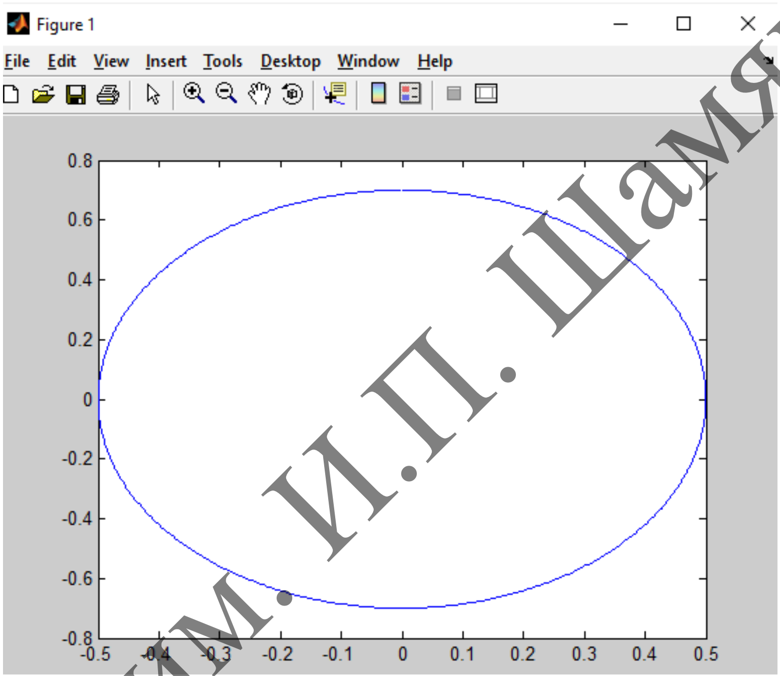


Рисунок 8 – График параметрически заданной функции

Также, например, можно построить график кусочно-заданной функции (кусочно-функциональной зависимости)

$$y(x) = \begin{cases} \pi \cdot \sin x, & -2\pi \le x \le -\pi; \\ \pi - |x|, & -\pi < x < \pi; \\ \pi \cdot \sin^3 x, & \pi \le x \le 2\pi. \end{cases}$$

Для этого сначала необходимо вычислить каждую из трех ветвей, то есть фактически получить три пары массивов **x1** и **y1**, **x2** и **y2**, **x3** и **y3**, затем объединить значения всех абсцисс в вектор **x**, а значения всех ординат в вектор **y** и вывести график функции, задаваемой парой массивов **x** и **y**. При таком подходе получается график, изображенный на рисунке 9.

МГПУ им. И.П. Шамякина

```
x1=-2*pi:pi/30:-pi;
y1=pi*sin(x1);
x2=-pi:pi/30:pi;
y2=pi-abs(x2);
x3=pi:pi/30:2*pi;
y3=pi*sin(x1).^3;
x=[x1 x2 x3];
y=[y1 y2 y3];
plot(x,y)
```

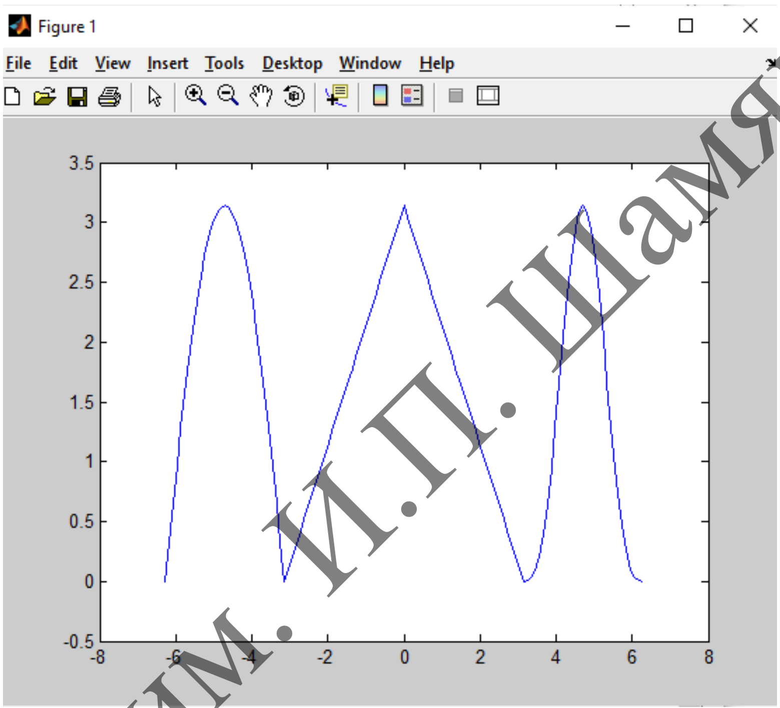


Рисунок 9 – График функции, заданной кусочным образом

Можно поступить и по-другому – построить графики трех ветвей как три различные функции, каждую своим цветом и маркером. В этом случае график имеет более наглядный вид, так как каждая ветвь функции отображается по-разному (рисунок 10):

```
x1=-2*pi:pi/30:-pi;
y1=pi*sin(x1);
x2=-pi: pi/30:pi;
y2=pi-abs(x2);
x3=pi: pi/30:2*pi;
y3=pi*sin(x1).^3;
plot(x1,y1,'r+',x2,y2,'kx',x3,y3,'bs')
```

МГПУ им. И.П. Шамякина

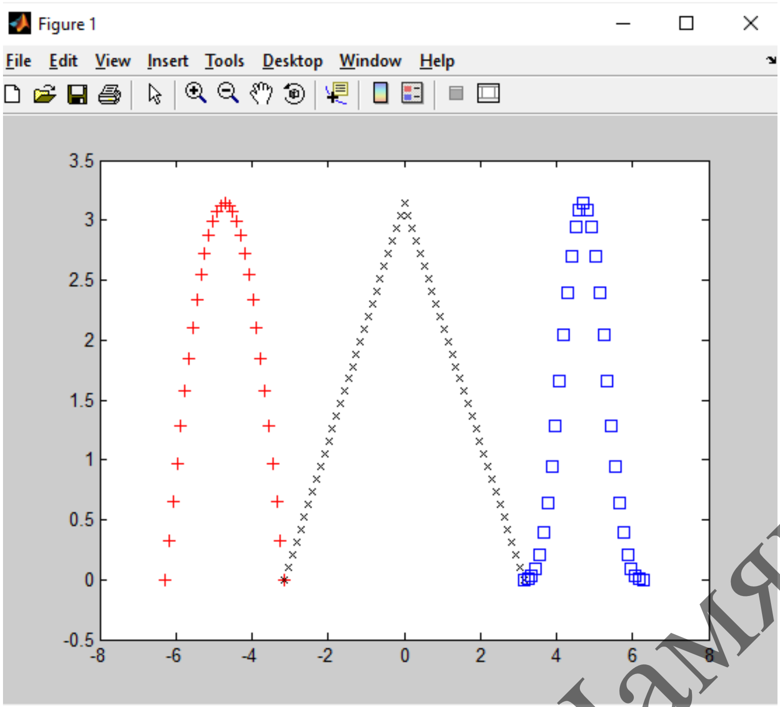


Рисунок 10 – График функции, заданной кусочным образом, но с различным оформлением частей графика

Для закрепления навыков работы с двумерными графиками функций в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ Е.

МГПУ им. И.П. Шамякина

7. ПОСТРОЕНИЕ ТРЕХМЕРНЫХ ГРАФИКОВ ФУНКЦИЙ

Обычно при построении трехмерных графиков функций для изменения координат вдоль осей абсцисс и ординат генерируется координатная сетка с использованием функции **`meshgrid`**, вызываемой с двумя входными и двумя выходными аргументами. Входными аргументами являются векторы, элементы которых соответствуют сетке на прямоугольной области построения функции. Если область построения функции квадрат и шаг сетки по обоим направлениям одинаков, то допустимо указать один аргумент, например, **`[X,Y]=meshgrid(-1:0.05:1)`**. Выходными аргументами являются матрицы с абсциссами и ординатами узлов сетки.

Рассмотрим основные возможности, предоставляемые Matlab для визуализации функций двух переменных, на примере построения графика

$$z(x,\, y) = 4\sin(2\pi x)\cos(1{,}5\pi y)(1-x^2)y(1-y)$$

на прямоугольной области определения $x \in [-1;1]$, $y \in [0;1]$ с шагом изменения координат, равным 0,05. Для этого необходимо подготовить матрицы с координатами узлов сетки и значениями функции. В случае построения каркасной поверхности, изображенной на рисунке 11, следует использовать функцию **`mesh`**, вызываемую с тремя аргументами.

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
mesh(X,Y,Z)
```

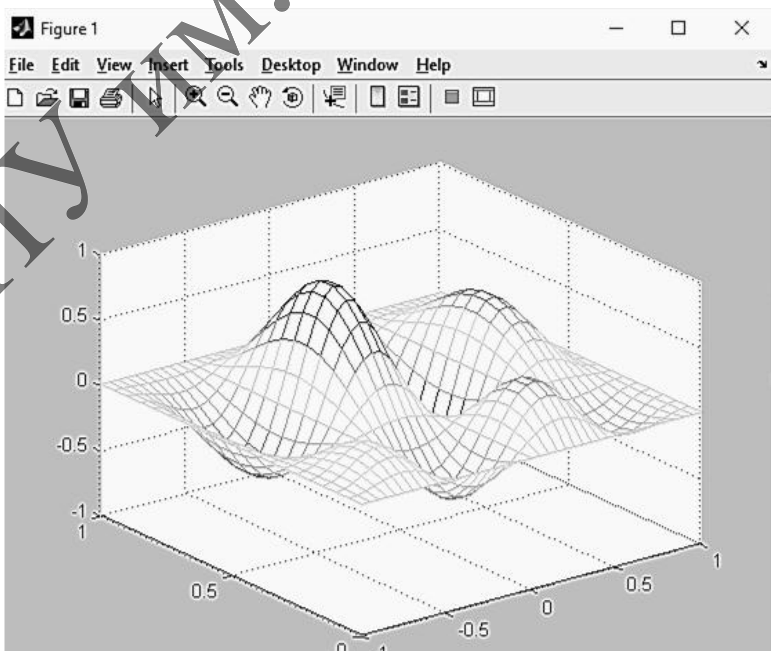


Рисунок 11 – Получение каркасной поверхности с использованием функции `mesh`

МГПУ им. И.П. Шамякина

Примечание – Кроме приведенного способа обращения к **mesh** с тремя входными аргументами (матрицами), допускается также ряд других. В частности, если указан только один входной аргумент – матрица, то на осях абсцисс и ординат откладываются значения, соответственно, столбцевых и строчных индексов ее элементов. Вместо номеров строк и столбцов можно указать векторы, состоящие из требуемых чисел.

Цвет линий поверхности соответствует значениям функции. При помощи команды **hidden off** можно сделать каркасную поверхность прозрачной (рисунок 12). Команда **hidden on** возвращает графику прежний вид.

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
mesh(X,Y,Z)
hidden off
```

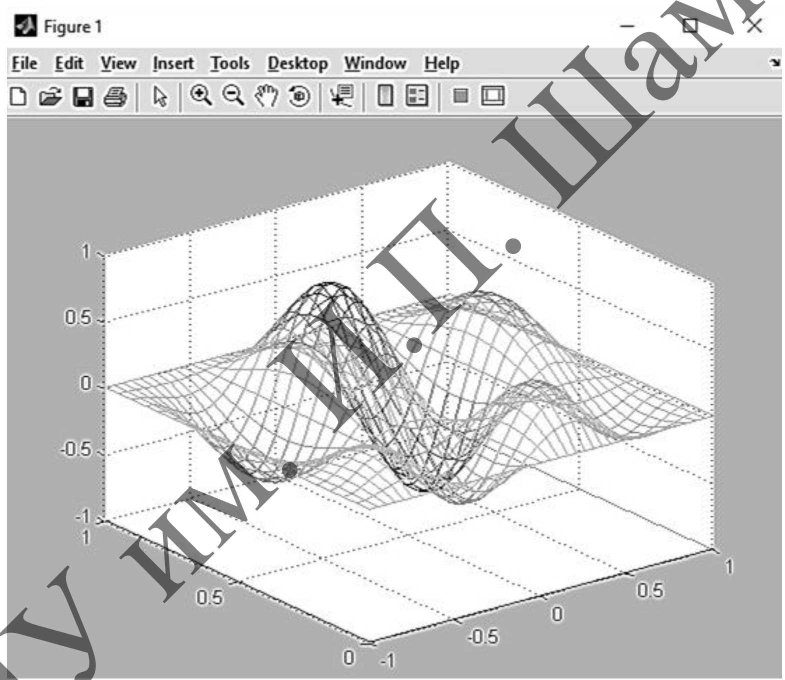


Рисунок 12 – Получение каркасной поверхности с использованием функций mesh и hidden off

Функция **surf** строит каркасную поверхность графика функции и заливает каждую клетку поверхности определенным цветом, зависящим от значения функции в точках, соответствующих узлам клетки. Команда **surf(X,Y,Z)** приводит к графику, изображенному на рисунке 13.

```
[X,Y]=meshgrid(-1:0.05:1,0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
surf(X,Y,Z)
```

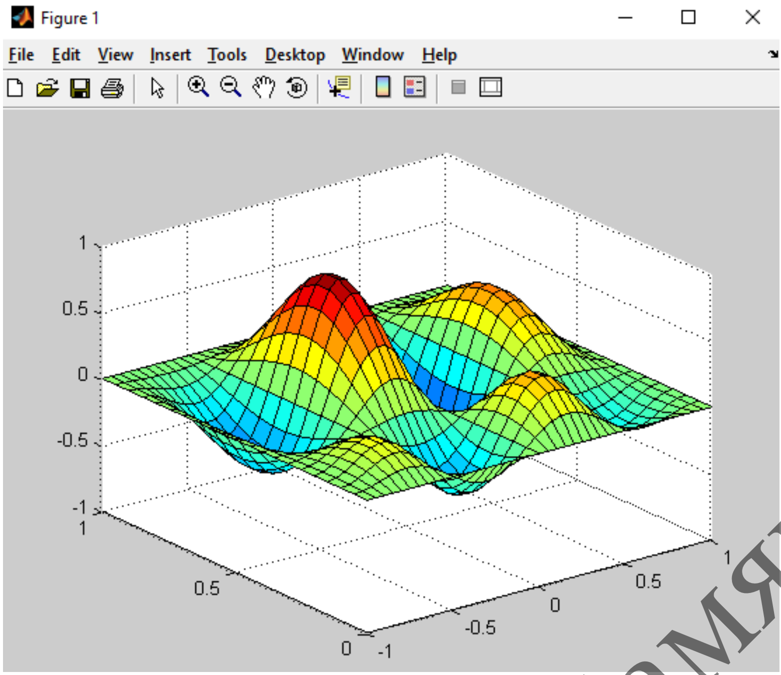


Рисунок 13 – Получение каркасной поверхности залитой цветом с использованием функции surf

В пределах каждой клетки цвет постоянный. Команда **`shading flat`** позволяет убрать каркасные линии (рисунок 14).

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
surf(X,Y,Z)
shading flat
```

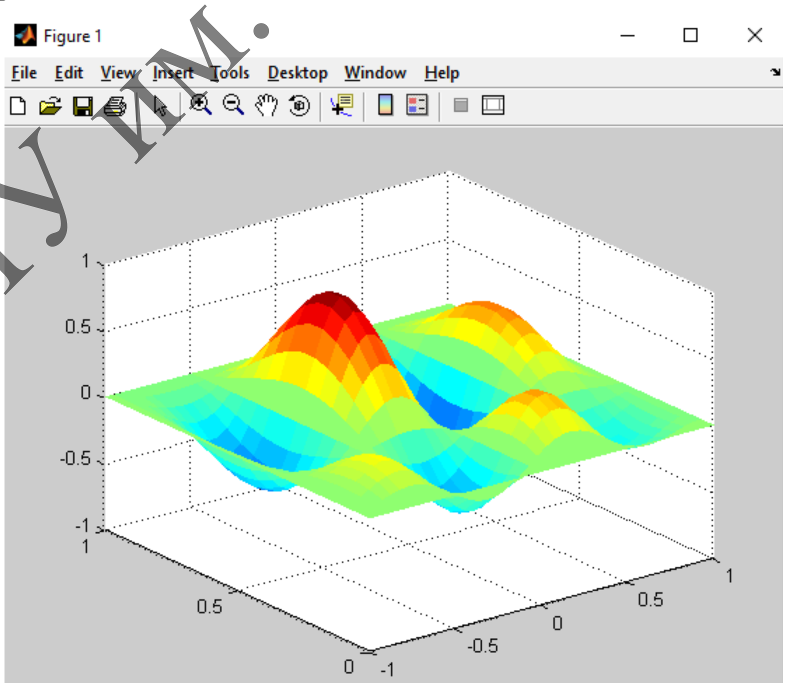


Рисунок 14 – Получение поверхности залитой цветом без каркасной сетки с использованием функций surf и shading flat

МГПУ им. И.П. Шамякина

Для получения поверхности, плавно залитой цветом, зависящим от значений функции (рисунок 15), предназначена команда **`shading interp`**.

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
surf(X,Y,Z)
shading interp
```

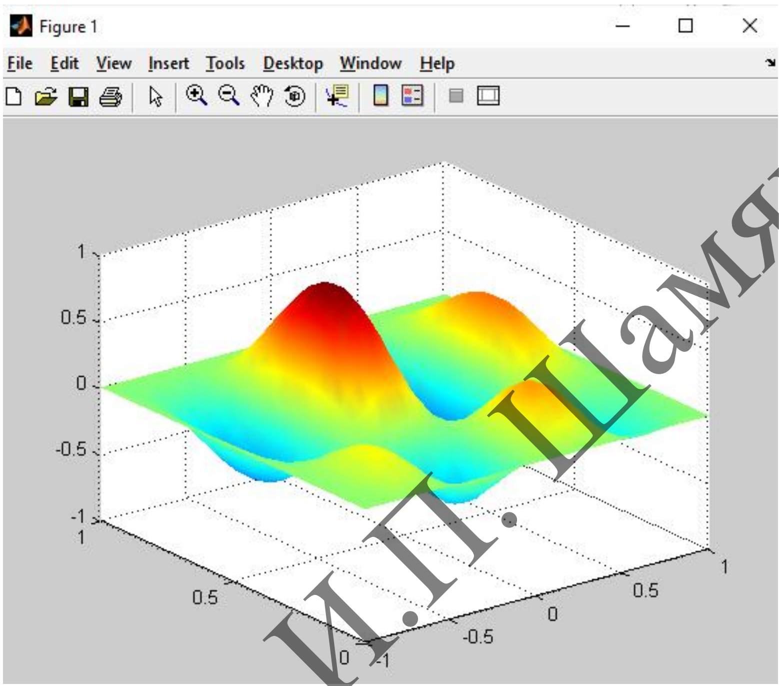


Рисунок 15 – Получение поверхности плавно залитой цветом с использованием функций `surf` и `shading interp`

Трехмерные графики, изображенные на рисунках 11–15, удобны для получения представления о форме поверхности, однако по ним трудно судить о значениях функции. В Matlab определена команда **`colorbar`**, которая выводит рядом с графиком цветовую шкалу, устанавливающую соответствие между цветом и значением функции. Построим при помощи **`surf`** график поверхности и дополним его информацией о цвете. Рисунок 16 иллюстрирует получающийся результат. Окно вывода графика несколько уменьшается из-за того, что рядом размещается цветовая шкала. Команду **`colorbar`** можно применять в сочетании со всеми функциями, строящими трехмерные объекты.

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
surf(X,Y,Z)
colorbar
```

МГПУ им. И.П. Шамякина

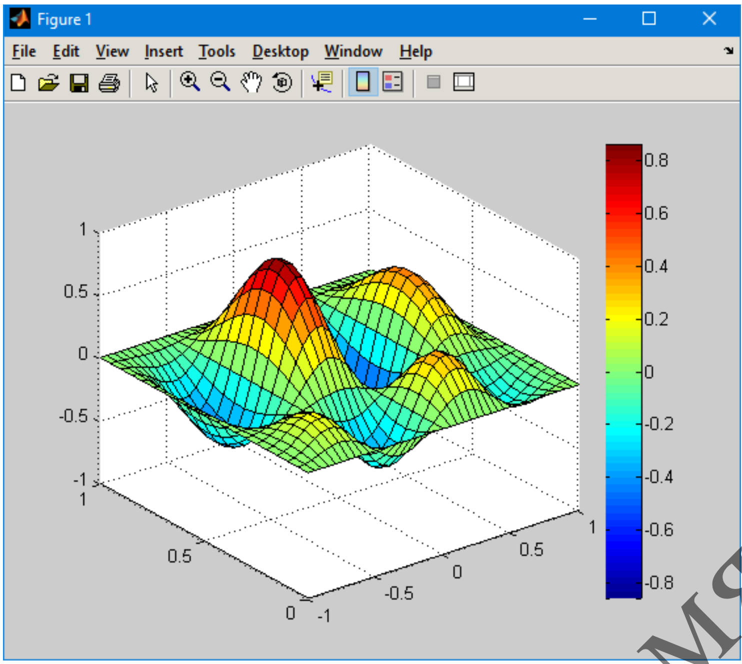


Рисунок 16 – Соответствие цвета и значений функции при использовании `colorbar`

Более информативным является график, содержащий линии уровня функции, то есть линии постоянства значений функции, на плоскости *Oxy*. Для получения такого графика следует использовать **`meshc`** или **`surfc`** вместо **`mesh`** или **`surf`** соответственно (рисунки 17 и 18).

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
meshc(X,Y,Z)
colorbar
```

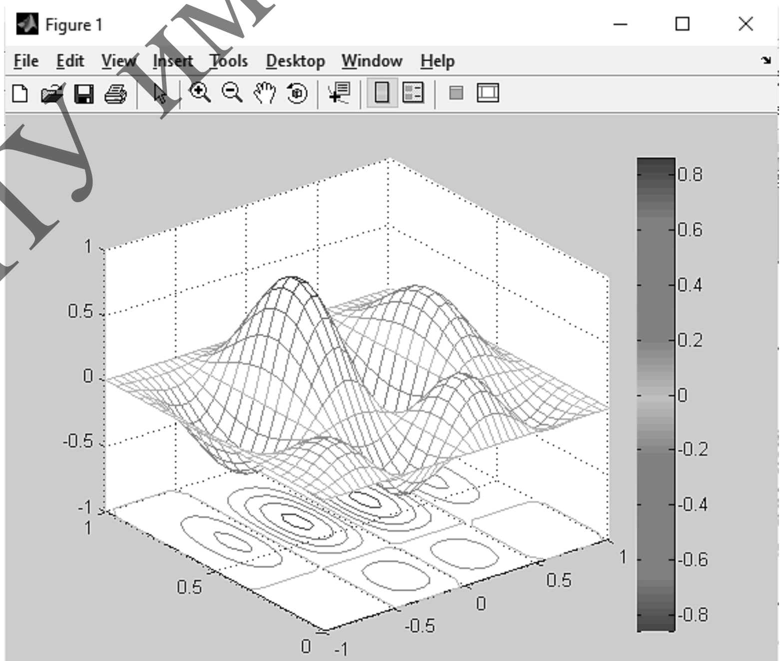


Рисунок 17 – Построение графика трехмерной функции с использованием функций `meshc` и `colorbar`

МГПУ им. И.П. Шамякина

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
surfc(X,Y,Z)
colorbar
```

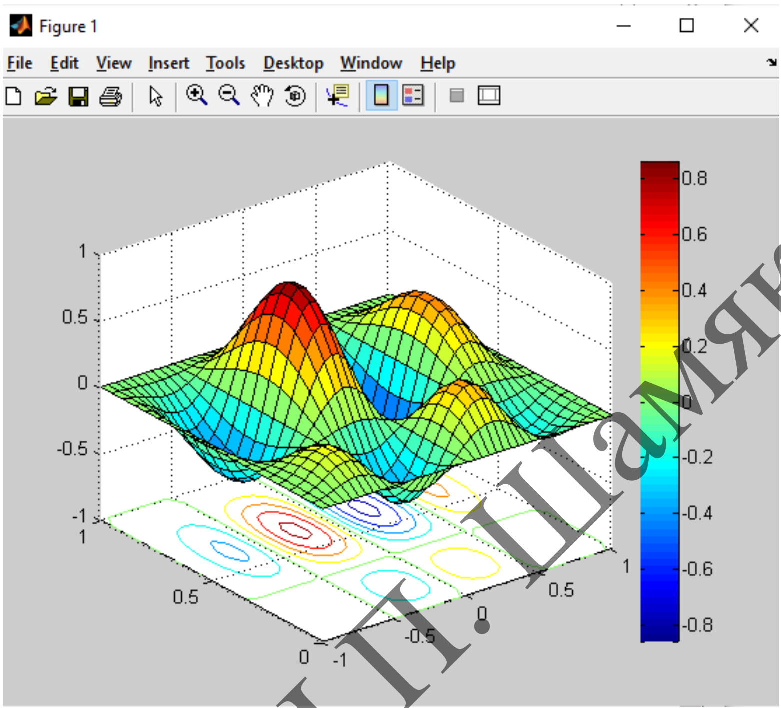


Рисунок 18 – Построение графика трехмерной функции с использованием функций `surfc` и `colorbar`

Matlab позволяет построить поверхность, состоящую из линий уровня, при помощи функции **`contour3`**. Эту функцию можно использовать так же, как и описанные выше **`mesh`**, **`surf`**, **`meshc`** и **`surfc`** с тремя аргументами. При этом число линий уровня выбирается автоматически. Имеется возможность задать четвертым аргументом в **`contour3`** либо число линий уровня, либо вектор, элементы которого равны значениям функции, отображаемым в виде линий уровня. Задание вектора удобно, когда требуется исследовать поведение функции в некоторой области ее значений (срез функции). Построение, например, поверхности, состоящей из линий уровня, соответствующих значениям функции от 0 до 0,5 с шагом 0,01 представлено ниже (рисунок 19).

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
levels=0:0.01:0.5;
contour3(X,Y,Z,levels)
colorbar
```

МГПУ им. И.П. Шамякина

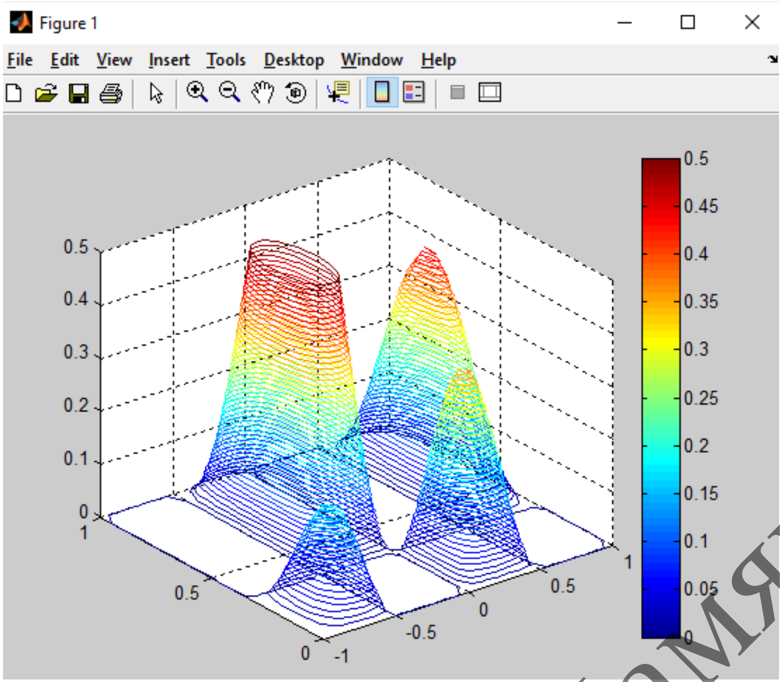


Рисунок 19 – График среза функции, состоящий из линий уровня с использованием функций `contour3` и `colorbar`

Matlab предоставляет возможность получать различные типы контурных графиков при помощи функций **`contour`** и **`contourf`** (рисунок 20).

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
contour(X,Y,Z)
```

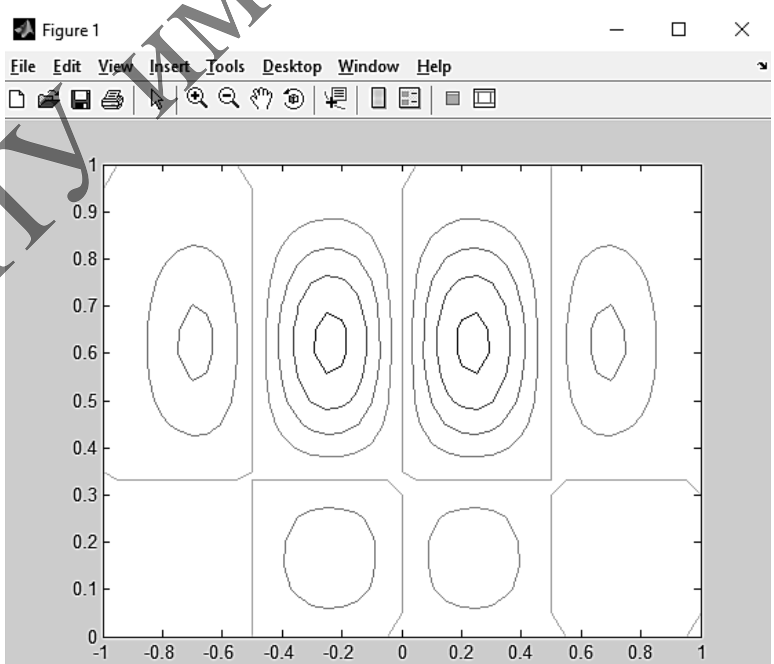


Рисунок 20 – Отображение линий уровня функции при использовании `contour`

МГПУ им. И.П. Шамякина

Такой график является малоинформативным, он не позволяет узнать значения функции на каждой из линий уровня. Использование команды **`colorbar`** также не позволит точно определить значения функции. Каждую линию уровня можно снабдить ярлыком с соответствующим значением исследуемой функции при помощи определенной в Matlab функции **`clabel`**. Функция **`clabel`** вызывается с двумя аргументами: матрицей, содержащей информацию о линиях уровня и указателем на график, на котором следует нанести разметку. Оказывается, пользователю не нужно самому создавать аргументы **`clabel`**. Функция **`contour`**, вызванная с двумя выходными параметрами, не только строит линии уровня, но и находит требуемые для **`clabel`** параметры. В этом случае можно использовать **`contour`** с выходными аргументами (в матрице **`CMatr`** содержится информация о линиях уровня, а в векторе **`h`** – указатели). Вызов **`contour`** следует завершить точкой с запятой для подавления вывода на экран значений выходных параметров и далее для большей информативности можно нанести на график сетку. Полученный результат приведен на рисунке 21.

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
[CMatr, h]=contour(X,Y,Z);
clabel(CMatr,h)
grid on
```

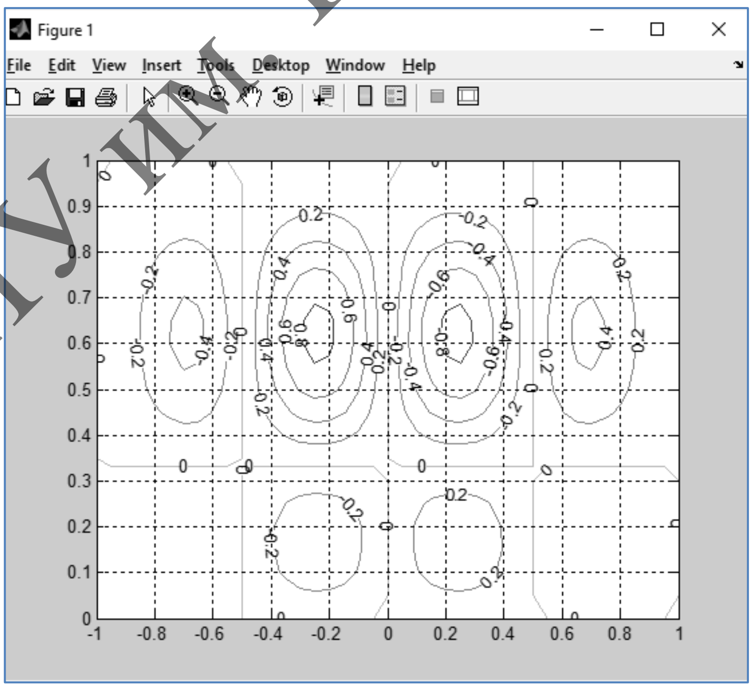


Рисунок 21 – Получение маркированных линий уровня с использованием функций `contour` и `clabel`

МГПУ им. И.П. Шамякина

Наглядную информацию об изменении функции дает заливка области определения цветом, зависящим от значения функции. Для построения таких графиков предназначена функция **contourf**. В следующем примере выводится график, который состоит из двадцати линий уровня (рисунке 22).

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4.*sin(2.*pi.*X).*cos(1.5.*pi.*Y).*...
(1-X.^2).*Y.*(1-Y);
contourf(X,Y,Z,20);
colorbar
```

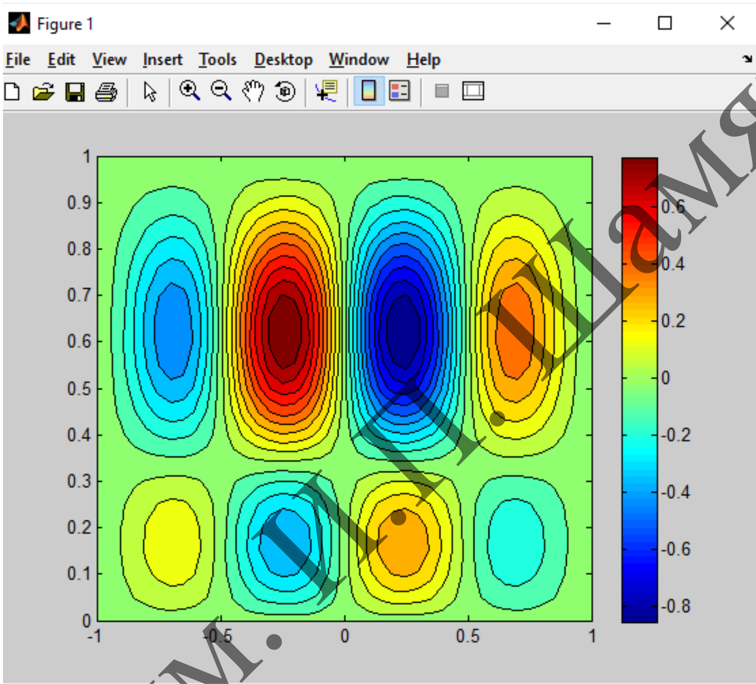


Рисунок 22 – Заливка цветом промежутков между линиями уровня с использованием функций contourf и colorbar

Для закрепления навыков работы с трехмерными графиками функций в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ Ж.

МГПУ им. И.П. Шамякина

8. ОФОРМЛЕНИЕ ГРАФИКОВ ФУНКЦИЙ

Простым, но эффективным способом изменения цветового оформления графика является установка цветовой палитры при помощи функции **colormap**. Следующий пример (рисунок 23) демонстрирует, как подготовить график функции для печати на монохромном принтере, используя палитру **gray**.

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y);
surfc(X, Y, Z)
colorbar
colormap(gray)
title('График функции z(x, y)')
xlabel('x')
ylabel('y')
zlabel('z')
```

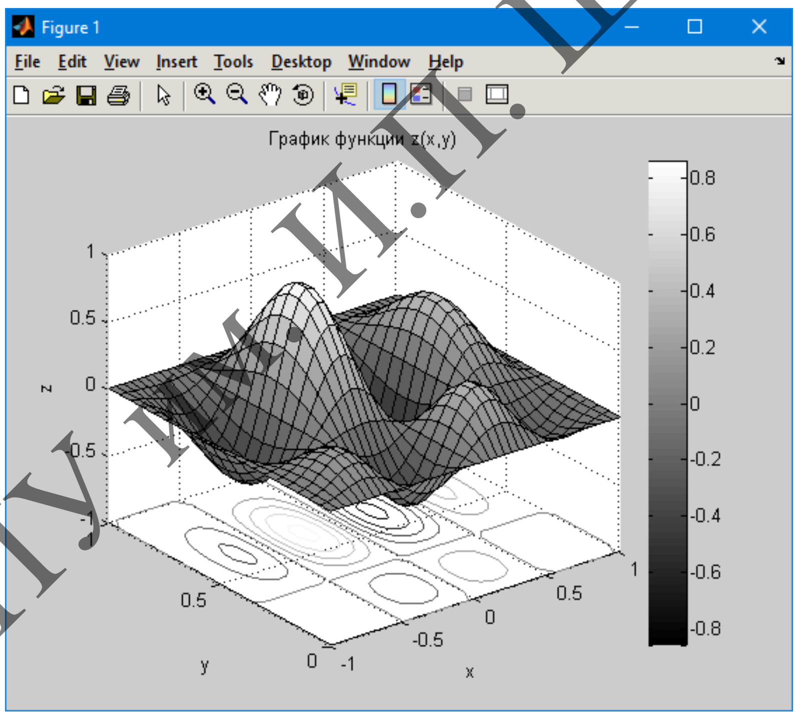


Рисунок 23 – Отображение поверхности с использованием палитры gray

Обратите внимание, что команда **colormap(gray)** изменяет палитру графического окна, то есть следующие графики будут выводиться в этом окне также в серых тонах. Для восстановления первоначального значения палитры следует применить команду **colormap('default')**. Цветовые палитры, доступные в Matlab, приведены в таблице 4.

МГПУ им. И.П. Шамякина

Таблица 4 – Палитры цвета

Палитра	Изменение цвета
`autumn`	Плавное изменение: красный-оранжевый-желтый
`bone`	Похожа на палитру gray, но с легким оттенком синего цвета
`colorcube`	Каждый цвет изменяется от темного к яркому
`cool`	Оттенки голубого и пурпурного цветов
`copper`	Оттенки медного цвета
`flag`	Циклическое изменение: красный-белый-синий-черный
`gray`	Оттенки серого
`hot`	Плавное изменение: черный-красный-оранжевый-желтый-белый
`hsv`	Плавное изменение (как цвета радуги)
`jet`	Плавное изменение: синий-голубой-зеленый-желтый-красный
`pink`	Похожа на палитру gray, но с легким оттенком коричневого цвета
`prism`	Циклическое изменение: красный-оранжевый-желтый-зеленый-синий-фиолетовый
`spring`	Оттенки пурпурного и желтого
`summer`	Оттенки зеленого и желтого
`vga`	Палитра Windows из шестнадцати цветов
`white`	Один белый цвет
`winter`	Оттенки синего и зеленого

МГПУ им. И.П. Шамякина

Добавление заголовка графика и подписей к осям осуществляется теми же командами **`title`**, **`xlabel`**, **`ylabel`**, что применялись при визуализации графиков функций одной переменной. Несложно догадаться, что для вертикальной оси предназначена команда **`zlabel`**. Часто требуется добавить формулу в заголовок или рядом с вертикальной осью. Использование в аргументах команд некоторых математических обозначений в формате TeX позволяет добавлять формулы на график. В заголовок графика, изображенного на рисунке 23, можно поместить формулу отображаемой функции

$$z(x,y) = 4\sin(2\pi x)\cos(1{,}5\pi y)(1-x^2)y(1-y).$$

Для этого применяется код с добавлением строки, которая приводит к появлению требуемого заголовка (рисунок 24).

```
    [X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
    Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y);
    surfc(X, Y, Z)
    colorbar
    colormap(gray)
    title('{\itz\rm(\itx\rm, \ity)} = 4sin(2\pi{\itx})
cos(1.5\pi{\ity})(1 - {\itx}^{2}){\ity}(1 - {\ity})')
    xlabel('x')
    ylabel('y')
    zlabel('z')
```

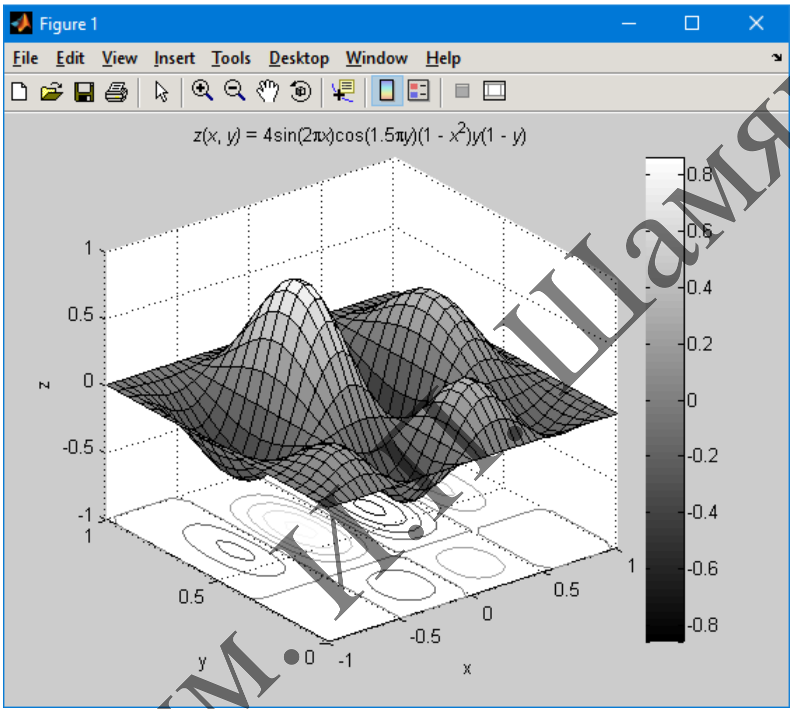


Рисунок 24 – Добавление в заголовок графика формулы отображаемой функции

Правила набора формул и изменения свойств шрифтов, приведены в таблице 4.

Таблица 5 – Правила набора формул и изменения свойств шрифтов

Что требуется	Команда TeX	Результат
Выделение курсивом одного символа или текста	`{\itx}`	*x*
	`1.2{\itP}`	1.2*P*
	`{\itГиперболический} синус`	*Гиперболический* синус
Выделение жирным шрифтом одного символа или текста	`Шаблон матрицы {\bfM}`	Шаблон матрицы **M**
	`{\bfАЧХ} фильтра`	**АЧХ** фильтра
Набор символа или текста жирным курсивом	`Векторы {\bf\itx} и {\bf\ity}`	Векторы ***x*** и ***y***
	`{\bf\itОптимальная} кривая`	***Оптимальная*** кривая

МГПУ им. И.П. Шамякина

Продолжение таблицы 5

Изменение шрифта и его размера	`{\fontname{arial}\fontsize{14}Z-функция}`	Z-функция
Степень, верхний индекс	`x^{2}`	x^2
	`{\itx}^{2.5}`	$x^{2.5}$
	`{\ite}^{\it-x}`	e^{-x}
Нижний индекс	`f_{5}`	f_5
	`f_{\itxx}`	f_{xx}

Примечание – Прямой шрифт текста в Matlab устанавливается командой `\rm`. Также для получения обычного прямого шрифта можно не указывать никаких команд.

Возможно использование греческих букв и специальных символов, например **`title('Зависимость при a = \pi')`** приводит к заголовку: «Зависимость при a = π». В таблицах 6 и 7 приведены команды для вставки некоторых прописных и строчных греческих букв и специальных символов.

Таблица 6 – Греческие буквы

Команда	**Символ**	**Команда**	**Символ**	**Команда**	**Символ**
\alpha	α	\lambda	λ	\chi	χ
\beta	β	\mu	μ	\psi	ψ
\gamma	γ	\nu	ν	\omega	ω
\delta	δ	\xi	ξ	\Gamma	Γ
\epsilon	ε	\rho	ρ	\Delta	Δ
\eta	η	\sigma	σ	\Theta	Θ
\theta	θ	\tau	τ	\Lambda	Λ
\kappa	κ	\phi	φ	\Phi	Φ

Таблица 7 – Специальные символы

Команда	**Символ**	**Команда**	**Символ**
\leq	≤	\leftrightarrow	↔
\qeq	≥	\leftarrow	←
\pm	±	\rightarrow	→
\propto	∞	\downarrow	↓
\partial	∂	\uparrow	↑

МГПУ им. И.П. Шамякина

Указанные команды можно использовать в качестве аргумента функций **title**, **xlabel**, **ylabel** при построении двумерных графиков и в тех же командах вместе с **zlabel** для трехмерных графиков.

Еще один пример построения трехмерного графика с подписями представлен на рисунке 25. Также ниже приведена последовательность команд, обеспечивающих требуемый результат.

```
[X,Y]=meshgrid(0:0.05:1, -2:0.05:0);
Z=-exp(-Y.^2).*cos(3*pi*X).*X.*(1-X).*Y;
surfc(X, Y, Z)
colormap(gray)
colorbar
title('График функции {\itz\rm(\itx\rm, \ity)} = 4sin (2\pi{\itx})cos(1.5\pi{\ity})(1 - {\itx}^{2}) {\ity}(1 - {\ity})')
xlabel('x')
ylabel('y')
zlabel('z')
```

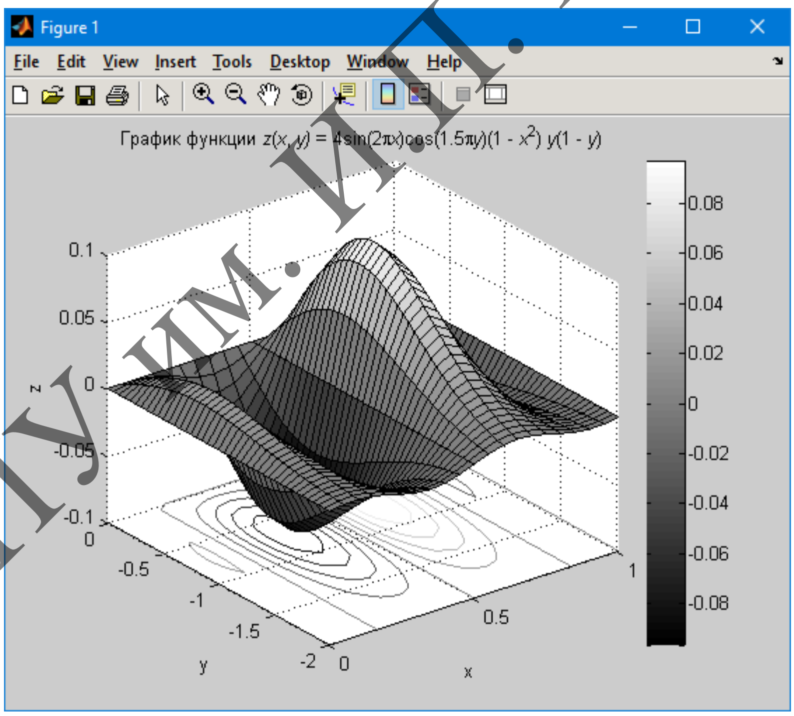


Рисунок 25 – Использование команд для создания подписей к графику функции

Для закрепления навыков оформления графиков функций в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ И.

МГПУ им. И.П. Шамякина

9. РАБОТА С НЕСКОЛЬКИМИ ГРАФИКАМИ

Во всех примерах, рассмотренных выше, графики выводились в специальное графическое окно с заголовком Figure 1. При следующем построении графика предыдущий пропадал, а новый выводился в то же самое окно. Matlab предоставляет следующие возможности работы с несколькими графиками:

- вывод каждого графика в свое окно;
- вывод нескольких графиков в одно окно (на одни координатные оси);
- отображение в пределах одного окна нескольких графиков, каждого на своих осях.

Команда **figure**, определенная в Matlab, служит для создания пустого графического окна и отображения его на экране. Окно становится текущим, то есть все последующие графические функции будут осуществлять построение графиков в этом окне. Для получения нового графического окна следует снова использовать **figure**. Например, последовательность команд

```
[X,Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y)
figure
mesh(X, Y, Z)
figure
surfl(X, Y, Z)
```

приводит к появлению на экране двух графических окон: Figure 1, содержащего каркасную поверхность, и Figure 2 с освещенной поверхностью. Окно Figure 2 является текущим, так как было создано последним. Команды, набираемые далее, например:

```
colormap ('copper')
shading interp
```

приведут к изменениям именно в этом окне. Для того чтобы сделать графическое окно Figure 1 текущим, следует щелкнуть на нем мышкой, вернуться в рабочую среду Matlab и продолжать ввод команд. Команды повлекут изменения в окне Figure 1. Для очистки всего текущего окна используется команда **clf** (сокращение от clear figure), а для того, чтобы убрать только график, но оставить оси, заголовок и названия осей, следует применить **cla** (сокращение от clear axes).

Вышеописанным способом можно получить сколько угодно графических окон и вывести в них графики различных функций или визуализировать векторные и матричные данные. Однако для изменения того или иного графика придется искать его окно на экране и делать его текущим при помощи щелчка мыши. Есть более универсальный и удобный способ работы с несколькими окнами. При создании каждого нового графического окна

МГПУ им. И.П. Шамякина

при помощи **figure** следует вызвать ее с выходным аргументом. Этот аргумент называется в Matlab указателем на графическое окно. Значением выходного аргумента является число, совпадающее с номером графического окна. Для того чтобы сделать графическое окно текущим, следует вызвать **figure**, прописав в качестве входного аргумента указатель на требуемое графическое окно.

Рассмотрим использование указателей на следующем примере. Требуется создать два графических окна, построить в них графики функций $f(x) = \sin x$ и $g(x) = \ln x$, а затем оформить их – дать заголовки и нанести сетку на второй график. Последовательность команд, приведенная ниже, позволяет получить желаемый результат, изображенный на рисунке 26.

```
sinGr=figure;
lnGr=figure;
x=[0.1:0.05:10];
f=sin(x);
g=log(x);
figure(sinGr)
plot(x,f)
figure(lnGr)
plot(x,g)
figure(sinGr)
title('{\itf}({\itx}) = sin\itx')
figure(lnGr)
title('{\itg}({\itx}) = ln\itx')
grid on
```

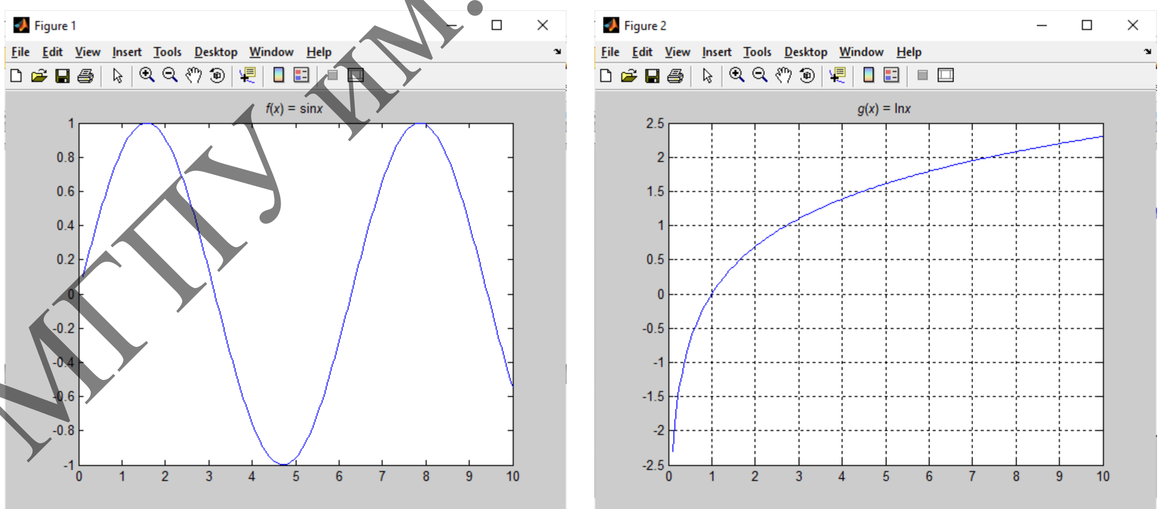


Рисунок 26 – Вывод графиков в разные окна

Для того чтобы очистить графическое окно с указателем **lnGr**, следует использовать **clf(lnGr)**. Удаление графика из первого окна, на которое указывает **sinGr**, производится при помощи **cla(sinGr)**.

МГПУ им. И.П. Шамякина

Возможность отображения нескольких графиков функций одной переменной на одних осях использовалась при изучении **plot**, **plotyy**, **semilogx**, **semilogy**, **loglog**. Перечисленные команды позволяют выводить графики нескольких функций, задавая соответствующие векторные аргументы парами, например, **plot(x,f,x,g)**. Однако при построении трехмерных графиков или различных типов графиков объединять их на одних осях не было возможности. Для объединения графиков предназначена команда **hold on**, которую нужно задать перед построением следующего графика. В приведенном ниже примере выводится пересечение плоскости и фигуры, заданной параметрически. Результат отображен на рисунке 27. Команда **hidden off** применена для того, чтобы показать часть фигуры, находящейся под плоскостью.

```
u=(-2*pi:0.1*pi:2*pi)';
v=-2*pi:0.1*pi:2*pi;
X=0.3*u*cos(v);
Y=0.3*u*sin(v) ;
Z=0.6*u*ones(size(v));
surf(X,Y,Z)
[X,Y]=meshgrid(-2:0.1:2) ;
Z=0.5*X+0.4*Y;
hold on
mesh(X,Y,Z)
hidden off
```

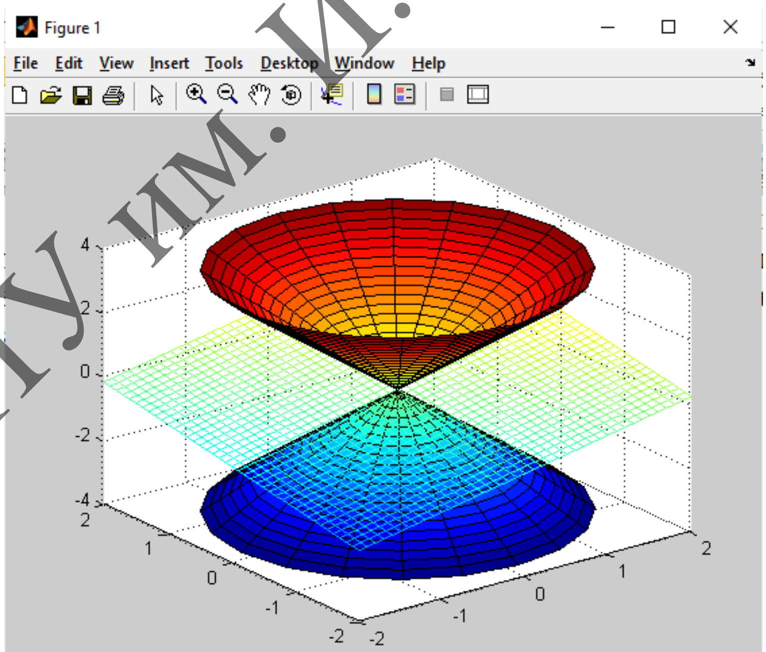


Рисунок 27 – Вывод графиков в одном графическом окне с использованием функции hold on

Обратите внимание, что **hold on** распространяется на все последующие выводы графиков. Для размещения графиков на новых осях следует

МГПУ им. И.П. Шамякина

выполнить команду **hold off**. Команда **hold on** может применяться и для графиков функций одной переменной, например,

```
plot(x,f,x,g)
```

эквивалентно следующей последовательности.

```
plot(x,f)
hold on
plot(x,g)
```

Matlab позволяет разместить в графическом окне несколько осей и вывести на них различные графики. Самый простой способ заключается в разбиении окна на определенное число частей по вертикали и горизонтали с использованием функции **subplot**, которая располагает оси в виде матрицы и используется с тремя параметрами: **subplot(i,j,n)**. Здесь **i** и **j** – число подграфиков по горизонтали и вертикали соответственно, а **n** – номер подграфика, который надо сделать текущим. Номер отсчитывается от левого верхнего угла построчно. Последовательность вызовов

```
subplot(2,2,1)
subplot(2,2,2)
subplot(2,2,3)
subplot(2,2,4)
```

приводит к размещению четырех осей координат в графическом окне (рисунок 28).

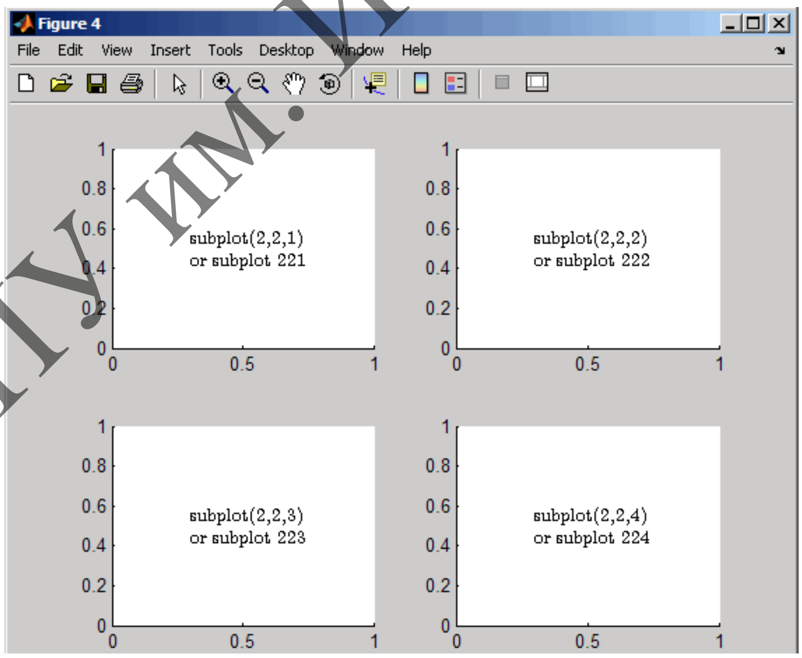


Рисунок 28 – Вывод в одном графическом окне 4-х графических осей с использованием функции subplot

Следующая комбинация использования входных аргументов в функции **subplot** приводит к образованию несимметричного расположения графических окон (рисунок 29).

МГПУ им. И.П. Шамякина

```
subplot(2,2,[1 3])
subplot(2,2,2)
subplot(2,2,4)
```

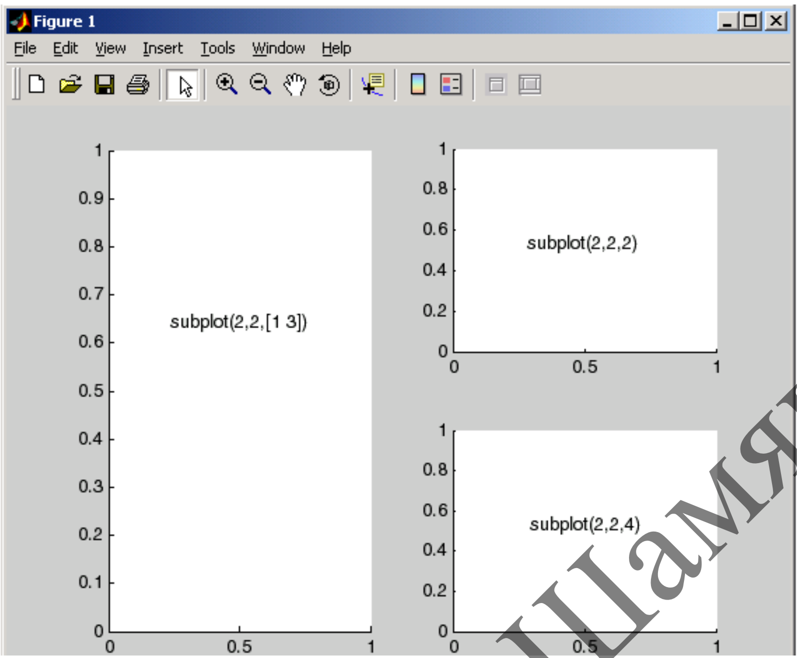


Рисунок 29 – Вывод в одном графическом окне 3-х графических осей с применением функции `subplot` при объединении воедино первого столбца графиков

Также возможно, например, и объединение графиков по горизонтали (рисунок 30) при использовании следующей комбинации **`subplot`**.

```
subplot(2,2,1:2)
subplot(2,2,3)
subplot(2,2,4)
```

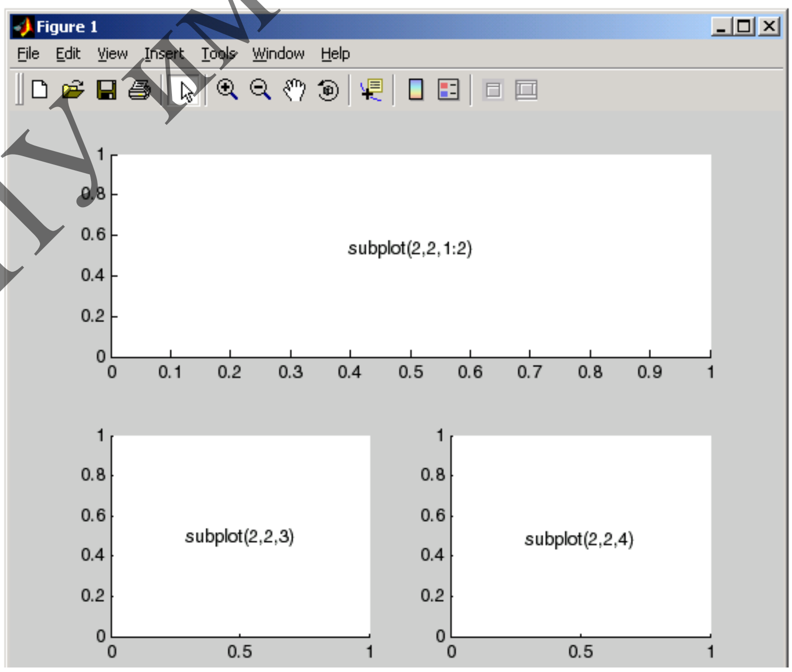


Рисунок 30 – Вывод в одном графическом окне 3-х графических осей с применением функции `subplot` при объединении воедино первой строки графиков

МГПУ им. И.П. Шамякина

В качестве завершающего упражнения построим графики функции

$$z(x, y) = 4 \cdot \sin 2\pi x \cdot \cos 1{,}5\pi y \cdot (1 - x^2) \cdot y \cdot (1 - y)$$

на прямоугольной области определения $x \in [-1; 1]$, $y \in [0; 1]$ всеми известными способами, размещая их на отдельных подграфиках. Названия команд, применяемых для построения графиков, включим в заголовки подграфиков. В результате выполнения приведенной ниже последовательности действий должно получится графическое окно, представленное на рисунке 31.

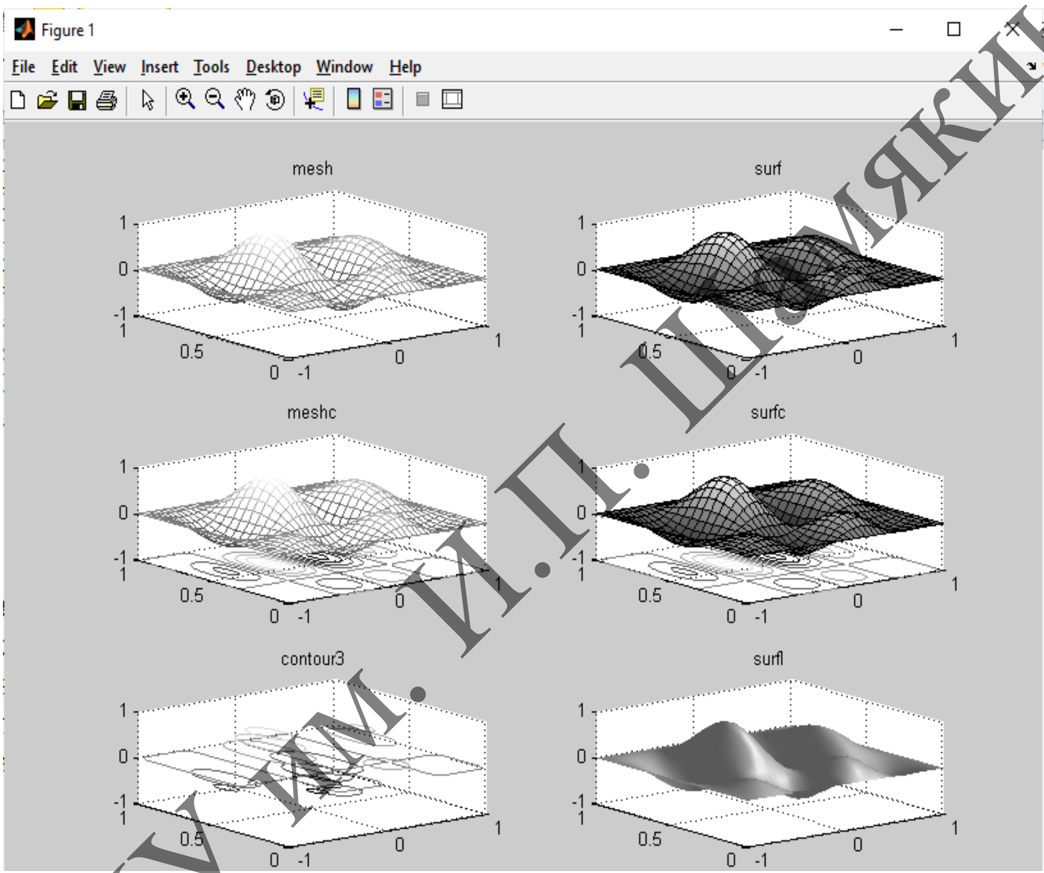


Рисунок 31 – Вывод в одном графическом окне 6-ти графических осей и располагающихся в них графиков с использованием функции subplot

```
[X, Y]=meshgrid(-1:0.05:1, 0:0.05:1);
Z=4*sin(2*pi*X).*cos(1.5*pi*Y).*(1-X.^2).*Y.*(1-Y);
subplot(3,2,1)
mesh(X,Y,Z)
title('mesh')
subplot(3,2,2)
surf(X,Y,Z)
title('surf')
subplot(3,2,3)
meshc(X,Y,Z)
```

МГПУ им. И.П. Шамякина

```
title('meshc')
subplot(3,2,4)
surfc(X,Y,Z)
title('surfc')
subplot(3,2,5)
contour3(X,Y,Z)
title('contour3')
subplot(3,2,6)
surfl(X,Y,Z)
shading interp
title('surfl')
colormap(gray)
```

Обратите внимание, что последняя команда co1ormap(gray) изменяет палитру всего графического окна, а не подграфиков по отдельности.

Для закрепления навыков работы с несколькими графиками функций в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ К.

МГПУ им. И.П. Шамякина

10. РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ И ИХ СИСТЕМ

Для численного решения обыкновенных дифференциальных уравнений произвольного порядка и их систем в Matlab предназначены специальные функции, которые в вычислительной математике называют солверы. Matlab имеет достаточно большой набор солверов, основанных на различных численных методах. К ним относятся, например, **ode45**, **ode23**, **ode113** и т. д.

Очень часто солвер **ode45** дает вполне хорошие результаты, им стоит воспользоваться в первую очередь. Он основан на формулах Рунге-Кутты 4-го и 5-го порядка точности. Солвер **ode23** также основан на формулах Рунге-Кутты, но уже более низкого порядка точности. Имеет смысл применять **ode23** в задачах, содержащих небольшую жесткость, когда требуется получить решение с невысокой степенью точности. Если же требуется получить решение нежесткий задачи с высокой точностью, то наилучший результат даст **ode113**, основанный на методе переменного порядка Адамса-Бэшфорта-Милтона. Солвер **ode113** оказывается особенно эффективным для нежестких систем дифференциальных уравнений, правые части которых вычисляются по сложным формулам. Все солверы по умолчанию пытаются найти решение с относительной точностью 10^{-3} и абсолютной – 10^{-6}. Хорошим тестом качества приближенного решения является увеличение точности вычислений.

Если все попытки применения **ode45** и **ode113** не приводят к успеху, то возможно, что решаемая система является жесткой. Для решения жестких систем подходит солвер **ode15s**, основанный на многошаговом методе Гира, который допускает изменение порядка точности.

В общем случае вызов солвера для решения задачи Коши производится следующим образом:

```
[T,Y]=solver(@odefun, interval, Y0, options);
```

Здесь **odefun** – функция для вычисления вектор-функции правой части системы уравнений, **interval** – массив из двух чисел, задающий промежуток для решения уравнения, **Y0** – заданный вектор начальных значений искомой вектор-функции, **options** – структура для управления параметрами и ходом вычислительного процесса. Аргумент **options**, задается функцией **odeset**. Обычно используемые параметры **odeset** включают допустимые значения относительной погрешности **RelTol** и абсолютной погрешности **AbsTol**:

```
options=odeset('RelTol',1e-8,'AbsTol',1e-10);
```

МГПУ им. И.П. Шамякина

Любой солвер позволяет получить массив **T** с координатами узлов сетки, в которых найдено решение, и соответствующую массиву **T** матрицу **Y**, каждый столбец которой содержит решение определенного дифференциального уравнения.

Схема решения обыкновенных дифференциальных уравнений в Matlab состоит из следующих этапов:

1) приведение дифференциального уравнения порядка выше первого к системе дифференциальных уравнений первого порядка (если изначально задана такая система, то в этом нет необходимости);

2) написание специальной функции для системы уравнений;

3) вызов подходящего солвера;

4) визуализация результата.

Разберем решение обыкновенных дифференциальных уравнений на примере обыкновенного дифференциального уравнения второго порядка.

Интегрирование обыкновенного дифференциального уравнения порядка выше первого всегда можно свести к интегрированию системы обыкновенных дифференциальных уравнений первого порядка. Например, требуется проинтегрировать обыкновенное дифференциальное уравнение второго порядка

$$\frac{d^2x}{dt^2} = (1 - x^2)\cdot\frac{dx}{dt} - x.$$

Вводя новые переменные $\frac{dx}{dt} = y_1$ и $x = y_2$, приводим это уравнение к системе уравнений первого порядка.

$$\begin{cases} \dfrac{dy_1}{dt} = (1 - y_2^2)\cdot y_1 - y_2, \\ \dfrac{dy_2}{dt} = y_1. \end{cases}$$

Следует отметить, что решение систем дифференциальных уравнений в Matlab можно, а иногда оказывается даже и удобно, выполнять с использованием двух М-файлов, первый из которых содержит непосредственно сами уравнения для математического описания моделируемого процесса, а второй, главным образом, применяется для их решения и визуализации в графическом и/или динамическом виде полученных результатов. Отметим, что оба М-файла при таком подходе решения дифференциальных уравнений и их систем рекомендуется располагать на компьютере в одной папке для того, чтобы система Matlab при выполнении второго М-файла имела возможность «беспрепятственно» сослаться на первый М-файл и содержащиеся в нем уравнения.

МГПУ им. И.П. Шамякина

При этом следует обратить внимание, что имя первого М-файла и содержащейся в нем функции должны полностью совпадать. Таким образом, последовательность команд для решения представленной выше системы дифференциальных уравнений на двух М-файлах может иметь следующий вид.

1-й М-файл (имя «ode»)

```
function dy=ode(t,y)

%%% создаем вектор, в который будет вноситься
% решение системы дифференциальных уравнений.
dy=zeros(2,1);

%%% Записываем первое и второе уравнения системы.
dy(1)=(1-y(2)^2)*y(1)-y(2);
dy(2)=y(1);
```

2-й М-файл (имя «ode_solution»)

```
clc; clear all;
options=odeset('reltol',1e-4,'abstol',1e-7);
[T,Y]=ode45('@ode', [0:0.1:100], [0.1 0.2],...
options);
figure(1)
plot(T, Y(:,1)); %рисуем график y1(t).
figure(2)
plot(T, Y(:,2)); %рисуем график y2(t).
figure(3)
plot(Y(:,2), Y(:,1),'-'); %рисуем фазовую
% траекторию y1(y2).
```

Второй М-файл под именем «ode_solution» включает как решение самих дифференциальных уравнений, приведенных в первом М-файле, так и вывод полученных результатов в графическом виде.

Для закрепления навыков работы решения обыкновенных дифференциальных уравнений и их систем в Matlab предлагается выполнить задания для самостоятельной работы, приведенные в ПРИЛОЖЕНИИ Л.

МГПУ им. И.П. Шамякина

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ануфриев, И. Е. Matlab 7. Наиболее полное руководство / И. Е. Ануфриев, Л. Б. Смирнов, Е. Н. Смирнова. – СПб. : БХВ-Петербург, 2005. – 1104 с.

2. Потемкин, В. Г. Вычисления в среде MATLAB / В. Г. Потемкин. – М. : Диалог-МИФИ, 2004. – 720 с.

3. Чернецова, Е. А. Лабораторный практикум «Введение в MATLAB» / Е. А. Чернецова. – СПб. : РГГМУ, 2006. – 88 с.

4. Кривилев, А. В. Основы компьютерной математики с использованием системы MATLAB / А. В. Кривилев. – М. : Лекс-Книга, 2005. – 483 с.

5. Поршнев, С. В. Компьютерное моделирование физических процессов в пакете Matlab / С. В. Поршнев. – СПб. : Лань, 2011. – 736 с.

6. Коткин, Г. Л. Компьютерное моделирование физических процессов с использованием Matlab : учеб. пособие для вузов / Г. Л. Коткин, Л. К. Попов, В. С. Черкасский. – М. : Юрайт, 2019. – 202 с.

7. Горбаченко, В. И. Вычислительная линейная алгебра с примерами на MATLAB / В. И. Горбаченко. – СПб. : БХВ-Петербург, 2011. – 320 с.

8. Бордовский, Г. А. Физические основы математического моделирования : учеб. пособие для вузов / Г. А. Бордовский, А. С. Кондратьев, А. Д. Р. Чоудери. – М. : Академия, 2005. – 320 с.

9. Самарский, А. А. Математическое моделирование: Идеи. Методы. Примеры / А. А. Самарский, В. П. Михайлов. – М. : Физматлит, 2001. – 320 с.

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Выполнение простейших вычисления в Matlab

1. Создайте и сохраните в своей рабочей папке новый M-файл «Practice_1» и после введения в нем команд

```
clc; clear all;
```

вычислите значения выражений

$$a = \frac{\tan(3,2) + \sqrt{e^{1,6} + \cos(8,4\pi)}}{(\sin(1,6\pi) - 3\tan(2,8))^2} - e^{-1/4} \cdot (1 + \cos(3,45\pi)),$$

$$b = \frac{\sqrt{6\tan(4,8)} - \cos(1,3\pi)}{\sin(0,7\pi) + \log^2(12,2)} - \sqrt{\log(13,5)} \cdot (1 - e^{1/2}),$$

$$c = \frac{\log_5(125) + \log_3(81)}{\cos^2\left(\frac{\pi}{6}\right) - \sin^2\left(\frac{\pi}{6}\right)}.$$

2. Найдите значения выражений, используя присвоение переменных.

$$d = \frac{\frac{\cos(9,1\pi)}{\log_5(3,4)} + \frac{\sqrt{\ln(2,6)}}{\sin(5,4\pi)} - \sqrt{\frac{\tan(5,48)}{\lg(3,6)}}}{\sqrt{\frac{\tan(5,48)}{\lg(3,6)}} - \left(\frac{\cos(9,1\pi)}{\log_5(3,4)}\right)^2},$$

$$e = \frac{\frac{\sqrt{\ln(2,6)}}{\sin(5,4\pi)} + \sqrt{\frac{\cos(9,1\pi)}{\log_5(3,4)}}}{\sqrt{\frac{\tan(5,48)}{\lg(3,6)}}},$$

$$f = \frac{\left(\frac{\cos(4,2\pi)}{\sin(7,6\pi)}\right)^2 - \frac{\tan(7,46)}{\sin(5,4\pi)} + \frac{\ln(3,2)}{2\lg(7,3)}}{\frac{\cos(4,2\pi)}{\sin(7,6\pi)} + \sqrt{\frac{\tan(7,46)}{\sin(5,4\pi)}}},$$

$$g = \frac{\sqrt{\frac{\ln(3,2)}{2\lg(7,3)}} + \left(\frac{\cos(4,2\pi)}{\sin(7,6\pi)} - \frac{\tan(7,46)}{\sin(5,4\pi)}\right)}{\left(\frac{\tan(7,46)}{\sin(5,4\pi)}\right)^2}.$$

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ Б

Работа с массивами. Вектор-столбцы и вектор-строки

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_2» и после введения в нем команд

```
clc; clear all;
```

вычислите сумму вектор-столбцов $a = \begin{pmatrix} 9,2 \\ 8,6 \\ 1,4 \end{pmatrix}$ и $b = \begin{pmatrix} 7,2 \\ 2,5 \\ 4,9 \end{pmatrix}$.

2. Выведите второй элемент вектор-строки d = (0,2 8,3 7,8 3,1 6,4), замените четвертый элемент вектор-строки на значение 5,7, создайте массив **`e`**, состоящий из первого, пятого и третьего элементов массива **`d`**.

3. Используя вектор-строку f = (1,8 6,4 9,3 0,5 2,1 3,7 2,9), создайте массив **`f1`**, заменив нулями элементы массива **`f`** с третьего по пятый; создайте массив **`f2`**, используя элементы массива **`f`** со второго по пятый; составьте массив **`f3`**, содержащий элементы **`f`**, кроме третьего и пятого, используя сцепление строк.

4. Перемножьте элементы вектор-столбца $g = \begin{pmatrix} 3,7 \\ 2,4 \\ 1,5 \\ 0,2 \\ 9,6 \\ 5,3 \end{pmatrix}$, найдите минимальный и максимальный элементы этого вектора, а также порядковые номера максимального и минимального элементов.

5. Упорядочьте вектор-строку h = (–0,2 6,3 –9,4 3,8 7,4 0,1):

а) по возрастанию ее элементов;

б) по убыванию ее элементов;

в) в порядке возрастания модулей ее элементов;

г) по возрастанию ее элементов с двумя выходными аргументами.

6. Выполните поэлементные математические операции с вектор-строками l = (2 8 –4 3) и m = (–5 7 –3 1):

а) перемножите поэлементно вектор-строки;

б) возведите в третью степень элементы вектор-строки l;

в) все элементы вектор-строки m возведите в степень, равную соответствующим элементам второй вектор-строки l;

г) разделите поэлементно l на m и m на l;

д) ко всем элементам вектор-строки m прибавите число –3,6, вычтите из результата вектор-строку l, поэлементно деленную на число 5, и умножьте поэлементно весь полученный результат на число 7.

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ В

Двумерные массивы и матрицы

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_3» и после введения в нем команд

```
clc; clear all;
```

выполните следующие математические операции:

а) найдите сумму и разность матриц $A=\begin{pmatrix}1 & 3 & -5\\ -2 & -6 & 9\end{pmatrix}$ и $B=\begin{pmatrix}9 & -5 & 4\\ 1 & 7 & 0\end{pmatrix}$;

б) умножьте матрицы A и $C=\begin{pmatrix}6 & 3 & -7\\ 5 & 4 & -9\\ -6 & 2 & 1\end{pmatrix}$;

в) полученную матрицу умножьте на 4.

2. Найдите значение выражения $(A+B)C^3(B-A)^T$.

3. Вычислите выражение $\begin{pmatrix}6 & 1 & -5\end{pmatrix}\cdot\begin{pmatrix}5 & 3 & 1\\ -7 & 4 & -1\\ 8 & 6 & -9\end{pmatrix}\cdot\begin{pmatrix}-2\\ 9\\ 1\end{pmatrix}$.

4. С последовательным использованием команд **`flipud`**, **`fliplr`** и **`rot90`** преобразуйте матрицу $D=\begin{pmatrix}1 & 4 & 8\\ 3 & -7 & 2\\ 5 & -3 & 9\end{pmatrix}$ к матрице $E=\begin{pmatrix}5 & 3 & 1\\ -3 & -7 & 4\\ 9 & 2 & 8\end{pmatrix}$.

5. Вычислите определитель матрицы E.

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ Г

Блочные матрицы

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_4» и после использования в нем команд

```
clc; clear all;
```

введите четыре квадратные матрицы $A=\begin{pmatrix} 5 & 1 \\ -20 & 7 \end{pmatrix}$, $B=\begin{pmatrix} 5 & 15 \\ 18 & -4 \end{pmatrix}$, $C=\begin{pmatrix} -3 & 9 \\ -5 & 11 \end{pmatrix}$, $D=\begin{pmatrix} 1 & -4 \\ 0 & 11 \end{pmatrix}$ размерностью 2×2 и создайте из них блочную матрицу $K=\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$.

2. Составьте блочную матрицу $M=\left(\begin{array}{c|c} S & a \\ \hline b & 4{,}7 \end{array}\right)$, где $S=\begin{pmatrix} 3 & 7 \\ 5 & 8 \end{pmatrix}$, $a=\begin{pmatrix} 11 \\ 15 \end{pmatrix}$, $b=\begin{pmatrix} 7 & -3 \end{pmatrix}$.

3. Выделите блоки из полученной матрицы $K=\begin{pmatrix} \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{pmatrix}$ и 1-ю строку из матрицы M, создав при этом массивы **K1**, **K2** и вектор-строку **m**.

4. Удалите 2-ю строку и 1-й столбец матрицы K.

5. Выполните создание матриц с различными элементами:

а) создайте прямоугольную матрицу E произвольного размера, заполненную нулями;

б) создайте квадратную матрицу F произвольного размера, заполненную единицами;

в) создайте прямоугольную матрицу G произвольного размера, заполненную случайными числами, используя функцию **rand.**

г) создайте квадратную матрицу H произвольного размера, у которой диагональные элементы являются элементами предварительно созданного произвольного вектор-столбца r, а все остальные элементы равны 0.

6. Заполните и сохраните следующие матрицы

МГПУ им. И.П.Шамякина

$$L = \begin{pmatrix} 3 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 7 \\ 1 & 3 & 1 & 1 & 1 & 0 & 0 & 0 & 7 & 0 \\ 1 & 1 & 3 & 1 & 1 & 0 & 0 & 7 & 0 & 0 \\ 1 & 1 & 1 & 3 & 1 & 0 & 7 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 3 & 7 & 0 & 0 & 0 & 0 \\ 5 & 9 & 9 & 9 & 9 & 5 & 5 & 5 & 5 & 5 \\ 9 & 4 & 9 & 9 & 9 & 5 & 5 & 5 & 5 & 5 \\ 9 & 9 & 3 & 9 & 9 & 5 & 5 & 5 & 5 & 5 \\ 9 & 9 & 9 & 2 & 9 & 5 & 5 & 5 & 5 & 5 \\ 9 & 9 & 9 & 9 & 1 & 5 & 5 & 5 & 5 & 5 \end{pmatrix},$$

$$N = \begin{pmatrix} 5 & 4 & 3 & 2 & 1 & 0 & 0 & 0 & 0 \\ 4 & 5 & 4 & 3 & 2 & 1 & 0 & 0 & 0 \\ 3 & 4 & 5 & 4 & 3 & 2 & 1 & 0 & 0 \\ 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 4 & 3 & 2 \\ 0 & 0 & 1 & 2 & 3 & 4 & 5 & 4 & 3 \\ 0 & 0 & 0 & 1 & 2 & 3 & 4 & 5 & 4 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix}.$$

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ Д

Визуализация матриц и поэлементные операции над ними

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_5» и после записи в нем команд

```
clc; clear all;
```

введите произвольную матрицу M размера 4×4.

2. Вычислите сумму элементов матрицы M по столбцам и по строкам.

3. Отсортируйте элементы матрицы M в порядке возрастания их столбцов и строк.

4. Определите максимальные и минимальные элементы по столбцам матрицы M, а также номера этих элементов.

5. Определите максимальные и минимальные элементы по строкам матрицы M, а также номера этих элементов.

6. Определите наибольший и наименьший элементы матрицы M.

7. Создайте квадратную матрицу L размера 10×10, состоящую из случайных целых чисел от –10 до 10.

8. Создайте квадратную матрицу K размера 8×8 следующего вида

$$K = \begin{pmatrix} -3 & -3 & -3 & -3 & 0 & 0 & 0 & 1 \\ -3 & -3 & -3 & -3 & 0 & 0 & 1 & 0 \\ -3 & -3 & -3 & -3 & 0 & 1 & 0 & 0 \\ -3 & -3 & -3 & -3 & 1 & 0 & 0 & 0 \\ 9 & 10 & 10 & 10 & 3 & -2 & -2 & -2 \\ 10 & 8 & 10 & 10 & -2 & 3 & -2 & -2 \\ 10 & 10 & 7 & 10 & -2 & -2 & 3 & -2 \\ 10 & 10 & 10 & 6 & -2 & -2 & -2 & 3 \end{pmatrix}.$$

9. Замените все отрицательные элементы матрицы K на равные по модулю положительные элементы, элементы больше 9 на –10, а элементы, лежащие в диапазоне значений [0; 1], на 7.

10. Найдите сумму всех элементов матрицы K.

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ Е

Построение двумерных графиков функций

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_6» и после введения в нем команд

```
clc; clear all;
```

постройте график функции одной переменной $y(x) = e^x \cos(5x)$ на отрезке $[-\pi; \pi]$ с шагом $\pi/20$ при использовании функции **`plot`**.

2. При использовании функции **`plot`** на одних координатных осях изобразите графики функций $f(x) = e^{0,1x} \cos^2 x$ и $g(x) = e^{-0,1x} \cos^2 x$ на интервале $[-2\pi; 2\pi]$ с шагом $\pi/10^3$.

3. Измените оформление построенных в пункте 2 графиков на свое усмотрение, изменив свойства их линий.

4. Сравните графики двух самостоятельно выбранных функций с использованием **`plotyy`**, шаг изменения функций также задайте самостоятельно.

5. С использованием **`loglog`** постройте в одних координатных осях графики функций $f(x) = x^5$ и $h(x) = \dfrac{x^{-4}}{e^x}$ на отрезке $[-100; 100]$ с шагом 1/10.

6. Используя табличные данные зависимости относительной влажности воздуха от времени 31 августа и 1 сентября 2024 года, взятые с ресурса pogoda.by, постройте графические зависимости этих величин, применив координатную сетку, подписи координатных осей, легенду и заголовок графика.

Таблица 8 – Зависимость относительной влажности воздуха от времени 31 августа и 1 сентября 2024 года

Время, часы	$14^{\underline{00}}$	$15^{\underline{00}}$	$16^{\underline{00}}$	$17^{\underline{00}}$	$18^{\underline{00}}$	$19^{\underline{00}}$	$20^{\underline{00}}$	$21^{\underline{00}}$	$22^{\underline{00}}$	$23^{\underline{00}}$
Относительная влажность воздуха, % 31 августа	22	20	20	22	25	29	35	40	44	46
Относительная влажность воздуха, % 1 сентября	35	30	29	32	38	42	51	59	64	64

МГПУ им. И.П. Шамякина

7. С применением **`plot`** постройте график фигуры Лиссажу, которая задается параметрически функциями $x(t) = \sin(4t)$ и $y(t) = \cos(5t)$ при $t \in [0;\, 2\pi]$. Шаг изменения t выберите, равным 0,01.

8. Постройте с шагом π/50 и оформите на свое усмотрение график кусочно-заданной функции.

$$y(x) = \begin{cases} \pi \cdot \cos x, & -2\pi \le x \le -\pi; \\ -2\pi + |x|, & -\pi < x < \pi; \\ -\pi \cdot \cos^2 x, & \pi \le x \le 2\pi. \end{cases}$$

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ Ж

Построение трехмерных графиков функций

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_7» и после введения в нем команд

```
clc; clear all;
```

с использованием функции **mesh** на квадратной области определения $x \in [-\pi; \pi]$, $y \in [-\pi; \pi]$ с шагом 0,2 постройте график

$$z(x, y) = \sin(x)\cos(y)$$

и сделайте полученную каркасную поверхность прозрачной.

2. С применением команды **surf** постройте на квадратной области $x \in [-5; 5]$, $y \in [-5; 5]$ с шагом 0,5 плавно залитую цветом поверхность графика функции

$$z(x, y) = x^2 y^2 + 50,$$

убрав каркасные линии.

3. С использованием команды **surf** постройте трехмерный график параметрически заданной в пространстве сферы единичного радиуса, координаты точек поверхности которой определяются уравнениями

$$\begin{gathered} x(\alpha, \beta) = \cos(\alpha)\cos(\beta), \\ y(\alpha, \beta) = \sin(\alpha)\cos(\beta), \\ z(\beta) = \sin(\beta), \\ \alpha \in [0; 2\pi], \ \beta \in \left[-\frac{\pi}{2}; \frac{\pi}{2}\right]. \end{gathered}$$

Шаг изменения параметров (углов) α и β задайте самостоятельно. Выведите рядом с полученным графиком цветовую шкалу, устанавливающую соответствие между цветом и значением параметрически заданной функции.

4. Для функций, приведенных в пунктах 1 и 2, с использованием соответственно команд **meshc** или **surfc** постройте графики, содержащие линии уровня указанных функций на плоскости *Oxy*, применив также к каждому из выводимых графиков команду **colorbar**.

5. С использованием **surf** постройте трехмерный график параметри-

МГПУ им. И.П. Шамякина

чески заданной в пространстве спирали, координаты точек которой определяются уравнениями

$$
\begin{gathered}
x(\alpha,\beta)=\cos(\alpha)(\cos(\beta)+3),\\
y(\alpha,\beta)=\sin(\alpha)(\cos(\beta)+3),\\
z(\alpha,\beta)=\sin(\beta)+\alpha,\\
\alpha\in[-2\pi;2\pi],\ \beta\in\left[-\pi;2\pi\right].
\end{gathered}
$$

Шаг изменения параметров (углов) α и β задайте самостоятельно. Рядом с полученным графиком выведите цветовую шкалу, устанавливающую соответствие между цветом и значением функции, просмотрите полученный результат.

Далее замените команду **`surf`** на команду **`contour3`**, задав ее четвертым аргументом число линий уровня в количестве ста и также просмотрите получившийся результат.

6. Постройте с использованием команды **`surf`** трехмерный график параметрически заданной «морской раковины», координаты точек которой определяются уравнениями

$$
\begin{gathered}
x(\alpha,\beta)=\alpha\cos(\alpha)(\cos(\beta)+1),\\
y(\alpha,\beta)=\alpha\sin(\alpha)(\cos(\beta)+1),\\
z(\alpha,\beta)=\alpha\sin(\beta)-\left(\frac{\alpha+3}{8}\pi\right)^2-20,\\
\alpha\in[0;8\pi],\ \beta\in\left[-\pi;\pi\right].
\end{gathered}
$$

Просмотрите полученный результат и повращайте выведенный график, чтобы убедиться, что построенная фигура имеет вид «морской раковины». Далее замените команду **`surf`** на **`contour`** с применением **`clabel`**, после чего отобразите на плоскости *Oxy* пять линий уровня построенного графика с соответствующими числовыми значениями исследуемой функции.

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ И

Оформление графиков функций

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_8» и после введения в нем команд

```
clc; clear all;
```

постройте поверхность динамически распространяющейся сферической волны, задаваемой уравнением

$$z(x, y, t) = A\cos(\omega t - \frac{2\pi}{\lambda}),$$

где A – амплитуда, ω – циклическая частота и λ – длина волны, а t – время. Для этого введите приведенный ниже код, содержащий также некоторые комментарии к нему.

```
%%% Вводим параметры волны
A=0.1; % амплитуда волны (м).
w=5; % циклическая частота волны (рад/с).
l=0.2; % длина волны (м).

%%% Вводим параметры области наблюдения волны
a=1; % длина стороны квадратной области наблюдения
% волны (м).
step=0.005; % шаг разбивки стороны квадратной
% области наблюдения волны (м).
[X,Y] = meshgrid(-a/2:step:a/2);

%%% Для лучшей визуализации результатов численных
% расчетов осуществляем вывод увеличенного
% графического окна для отображения распространения
% волны. Для этого изначально определяем разрешение
% экрана в пикселях.
SCRsize=get(0, 'ScreenSize');

%%% Левый край графического окна располагаем возле
% левого края экрана.
left=SCRsize(1);
```

МГПУ им. И.П. Шамякина

```
%%% Нижний край графического окна «приподнимаем» на
% 50 пикселей от нижнего края экрана для возможности
% отображения панели задач Windows, которая также
% может понадобиться пользователю для выполнения
% иных задач при работе с компьютером.
bottom=SCRsize(2)+50;

%%% Задаем ширину области графического окна, равную
% ширине экрана.
width=SCRsize(3);

%%% Уменьшаем высоту выводимого графического окна с
% учетом ранее выполненного его «приподнимания» на
% 50 пикселей от нижнего края экрана.
height=SCRsize(4)-50;

%%% Создаем графическое окно с заданными
% параметрами.
hF=figure('OuterPosition', [left bottom width...
height]);

%%% Вычисляем расстояния от источника волны до всех
% точек области ее наблюдения.
r=sqrt(X.^2+Y.^2);

%%% Задаем цикл для отображения вида волновой
% поверхности в различные моменты времени с шагом 0.1
% секунды.
t=0;
for i=1:100;
Z=A.*cos(w.*t-2.*pi./l.*r);
t=t+0.1;
surf(X,Y,Z);

%%% Используем функцию drawnow для покадрового
% вывода прописанных в циклах команд, реализуя тем
% самым анимацию моделируемого процесса
% распространения сферической волны.
drawnow
end
```

Просмотрите полученный результат, используя компиляцию программы.

2. Для изменения ракурса наблюдения графика (функция **view**)

МГПУ им. И.П. Шамякина

и изменения его цветовой палитры (функция **colormap**), добавьте команды

```
view(-30,55)
colormap(winter)
```

после функции **surf**. Просмотрите полученный результат и обратите внимание на то, что ракурс выводимого графика изменился и стал более приемлемым для наблюдения. Для ознакомления с возможностями команды **view** можно обратиться к справочной системе Matlab, вызываемой с использованием Help → Product Help, и далее осуществить индексный поиск по слову view.

3. Используя приведенные выше в издании таблицы 5 и 6, добавьте к графику заголовок в виде уравнения волны, при этом все латинские и греческие буквы в этом уравнении задайте жирным курсивом, а остальные символы, входящие в это уравнение, – прямым жирным шрифтом. Также к координатным осям *Ox*, *Oy* и *Oz* соответственно добавьте подписи «***x*, *м***», «***y*, *м***» и «***z*, *м***». Для введения всех подписей используйте шрифт Arial Unicode MS.

4. Уберите с графика каркасные линии, введя после **colormap(winter)** на следующей строке **shading flat**, а также замените функцию **surf** на **surfl** и просмотрите полученный результат.

Добавьте к графику освещение, введя после **shading flat** на следующей строке

```
lightangle(-45,30)
```

и также просмотрите полученный результат, обратив внимание на визуальные изменения, произошедшие с отображаемой волновой поверхностью. Для ознакомления с возможностями команды **lightangle**, а также ее входными аргументами, обратитесь к справочной системе Matlab при использовании индексного поиска по слову lightangle.

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ К

Работа с несколькими графиками

1. Создайте и сохраните в своей рабочей папке новый М-файл «Practice_9» и после введения в нем команд

```
clc; clear all;
```

постройте в двух одновременно выводимых окнах с использованием **`mesh`** и **`surfl`** графики поверхности заданной самостоятельно функции двух переменных. Также самостоятельно задайте их цветовую гамму, подписи к ним и к координатным осям.

2. С помощью команды **`hold on`** получите в одном графическом окне пересечение параметрически заданного шара и плоскости. Примените команду **`hidden off`** для отображения невидимых частей выводимых объектов.

3. Постройте графики произвольно заданных вами трех функций одной переменной при использовании **`subplot`**, объединив воедино первую строку графиков (графическое окно должно иметь вид подобный, представленному на рисунке 30).

4. В сети Internet найдите уравнения для любой понравившейся вам функции, задаваемой параметрически, и постройте в одном графическом окне при использовании **`subplot`** шесть ее различных отображений с применением **`mesh`**, **`surf`**, **`meshc`**, **`surfc`**, **`contour3`** и **`surfl`**. Число подграфиков по горизонтали выберите равное 2-м, а по вертикали – 3-м. Названия команд для построения графиков включите в заголовки подграфиков.

МГПУ им. И.П. Шамякина

ПРИЛОЖЕНИЕ Л

Решение обыкновенных дифференциальных уравнений и их систем в Matlab

1. Создайте и сохраните в своей рабочей папке новую папку «Practice_10». Далее создайте и сохраните в этой папке новый М-файл под именем «Movement_of_the_Earth».

Используя указанные выше в теоретической части данной темы особенности решения систем обыкновенных дифференциальных уравнений, введите в этом М-файле приведенную ниже систему обыкновенных дифференциальных уравнений для математического описания движения Земли вокруг Солнца.

$$\begin{cases} \dfrac{d\upsilon_x}{dt} = -\dfrac{GMx}{(x^2+y^2)^{3/2}}, \\ \dfrac{d\upsilon_y}{dt} = -\dfrac{GMy}{(x^2+y^2)^{3/2}}, \\ \dfrac{dx}{dt} = \upsilon_x, \\ \dfrac{dy}{dt} = \upsilon_y. \end{cases}$$

Здесь υ_x, υ_y и x, y – соответственно проекции вектора скорости и координаты Земли в момент времени t, G – гравитационная постоянная, M – масса Солнца.

Также в этом М-файле после строки

```
function dy=Movement_of_the_Earth(t,y)
```

введите необходимые для решения приведенной системы дифференциальных уравнений значения гравитационной постоянной $G = 6{,}67 \cdot 10^{-11}\,\dfrac{Н \cdot м^2}{кг^2}$ и массы Солнца $M = 1{,}9885 \cdot 10^{30}\ кг$. Сохраните М-файл.

2. В папке «Practice_10» создайте второй М-файл под именем «Movement_of_the_Earth_solution» и выполните решение в нем указанной системы дифференциальных уравнений, используя следующие значения начальных условий:

$\upsilon_x(0) = 0$ – проекция вектора скорости Земли на ось Ox в перигелии (ближайшей к Солнцу точке орбиты);

МГПУ им. И.П. Шамякина

$\upsilon_y(0) = 30,27 \cdot 10^3 \dfrac{м}{с}$ – проекция вектора скорости Земли на ось *Oy* в перигелии;

$x(0) = 147,1 \cdot 10^9\ м$ – начальная координата Земли относительно оси *Ox* в перигелии;

$y(0) = 0$ – начальная координата Земли относительно оси *Oy* в перигелии.

При решении системы дифференциальных уравнений используйте солвер **`ode113`**.

3. С применением функции **`plot`** и команды **`drawnow`** выведите в динамике движение Земли вокруг Солнца, отметив эти объекты круглыми маркерам, а также прорисуйте траекторию движения планеты. В данном случае при использовании приведенной выше системы дифференциальных уравнений предполагается, что Солнце находится в точке с координатой (0; 0).

4. Добавьте к графику заголовок «***Движение Земли вокруг Солнца***» и подписи «***Ox, м***», «***Oy, м***» к координатным осям *Ox* и *Oy* соответственно

МГПУ им. И.П. Шамякина

Справочное издание

СПРАВОЧНЫЕ МАТЕРИАЛЫ К ВЫПОЛНЕНИЮ
ЛАБОРАТОРНЫХ РАБОТ ПО УЧЕБНОЙ ДИСЦИПЛИНЕ
«ОСНОВЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ»

Составитель
Макаревич Александр Викторович

Корректор *В. В. Кузьмич*
Оригинал-макет: *Д. С. Москалевич, М. В. Бобкова*

Иллюстративный материал на первой странице обложки заимствован
из общедоступных интернет-ресурсов, не содержащих ссылок на авторов этих материалов
и ограничения на их заимствование.

Подписано в печать 26.12.2024. Формат 60x84 1/16. Бумага офсетная.
Цифровая печать. Усл. печ. л. 4,42. Уч.-изд. л. 3,01.
Тираж 49 экз. Заказ 39.

Издатель и полиграфическое исполнение: учреждение образования
«Мозырский государственный педагогический университет имени И. П. Шамякина».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/306 от 22 апреля 2014 г.
Ул. Студенческая, 28, 247777, Мозырь, Гомельская обл.
Тел. (0236) 24-61-29.

МГПУ им. И.П. Шамякина