

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению лабораторных работ
по дисциплине «МОДЕЛИРОВАНИЕ ПРОЦЕССОВ И СИСТЕМ»
для студентов направления подготовки
09.03.03 Прикладная информатика

(ЭЛЕКТРОННЫЙ ДОКУМЕНТ)

Ставрополь, 2026

Лабораторная работа № 1

ОСНОВЫ РАБОТЫ С MATLAB

Цель занятия: изучение интерфейса пользователя системы MATLAB и основ работы с системой в режиме прямых вычислений.

1.1 Основные теоретические сведения

Исторически MATLAB разрабатывался как диалоговая среда для матричных вычислений (MATrix LABoratory). Со временем пакет был оснащен хорошей графической системой, дополнен средствами компьютерной алгебры от Maple и усилен библиотеками команд (или Toolboxes), предназначенными для эффективной работы со специальными классами задач.

В состав MATLAB входят интерпретатор команд, графическая оболочка, редактор-отладчик, библиотеки команд, компилятор, символьное ядро пакета Maple для проведения аналитических вычислений, математические библиотеки MATLAB на C/C++, генератор отчетов и богатый инструментарий (Toolboxes).

Интерфейс MATLAB вполне отвечает современным канонам (см. рисунок 1.1). Он многооконный и имеет ряд средств прямого доступа к различным компонентам системы. Следует обратить внимание на следующие кнопки панели инструментов:

- New M-file** — выводит пустое окно редактора m-файлов;
- Open file** — открывает окно для загрузки файлов Matlab;
- Simulink** — открывает окно браузера библиотек Simulink;
- Help** — открывает окно справки.

Эти функции дублируются в очень простом меню системы MATLAB.

В левой части окна системы появились окна со вкладками **Launch Pad/Workspace** доступа к компонентам системы и вкладками текущей директории **Current Directory** и истории сессии **History**. Они обеспечивают оперативный контроль за состоянием системы. Выводимые на экран окна интерфейса MATLAB могут быть включены или отключены из пункта меню View.

Вся работа организуется через командное окно (**Command Window**), которое появляется при запуске программы. В процессе работы данные располагаются в памяти (**Workspace**) в виде матриц.

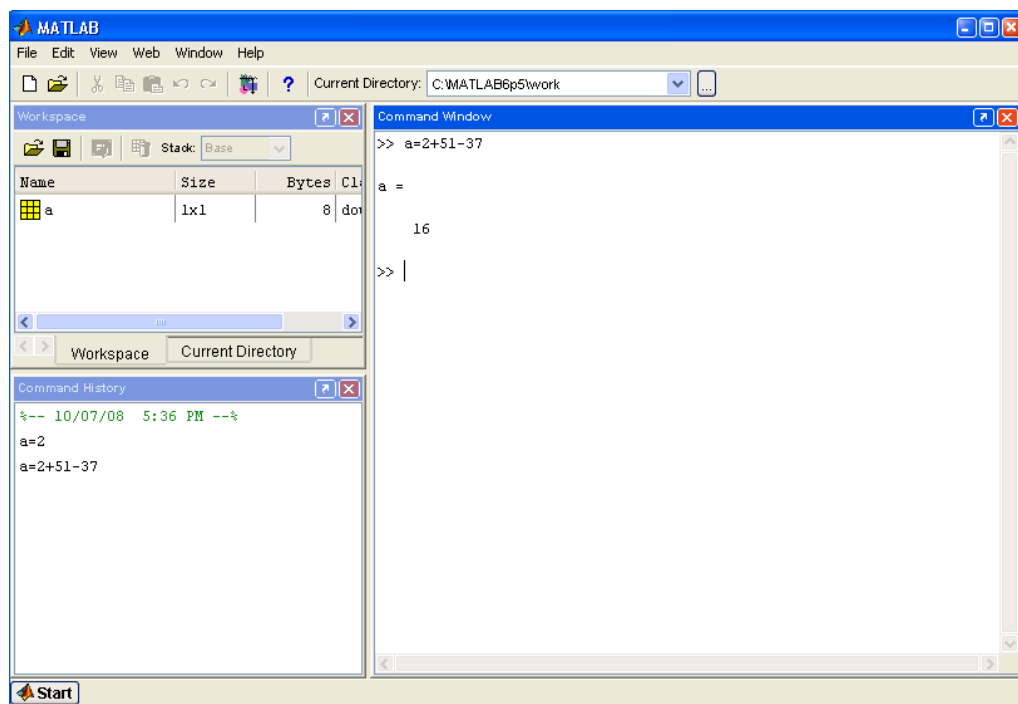


Рисунок 1.1 – Интерфейс программы MATLAB

Все расчеты в MATLAB выполняются с двойной точностью, а для представления чисел на экране имеются разные форматы. Нужный формат может быть определен в меню (**File/Preferences**) либо при помощи команды **format**. Существуют следующие способы представления чисел (табл.1.1).

Таблица 1.1 Форматы вывода на экран

Формат	Представление
short	Число отображается с 4 цифрами после десятичной точки или в формате short e
short e	Число в экспоненциальной форме с мантиссой из 5 цифр и показателем из 3 цифр
rat	Представление в виде рационального дробного числа
long	Число с 16 десятичными цифрами
long e	Число в экспоненциальной форме с мантиссой из 16 цифр и показателем из 3 цифр
hex	Число в шестнадцатеричной форме

Переменные в MATLAB не нужно предварительно описывать, указывая их тип. Все данные хранятся в виде массивов: числовые переменные (внутренний тип numeric), текстовые строки (char), ячейки (cell) и структуры (struct). Двумерный массив – это матрица, одномерный – вектор, а скаляр – матрица размера 1x1. Имя переменной должно начинаться с буквы, за ней могут идти буквы, цифры и символ подчеркивания. Допустимы имена любой длины, но MATLAB идентифицирует их по первым 31 символам и различает большие и малые буквы. В MATLAB имеется ряд констант (табл.1.2).

Таблица 1.2 Зарезервированные имена констант

Имя	Описание
ans	Результат последней операции
i, j	Мнимая единица
pi	Число π
eps	Машинная точность
realmax	Максимальное вещественное число
realmin	Минимальное вещественное число
inf	Бесконечность
NaN	Нечисловая переменная
end	Наибольшее значение индекса размерности массива

Отметим, что имя NaN (Not-a-Number) зарезервировано для результата операций $0/0$, $0*\text{inf}$, $\text{inf}-\text{inf}$ и т.п.

Таблица 1.3 Специальные символы

Символ	Назначение
[]	Квадратные скобки используются при задании матриц и векторов
	Пробел служит для разделения элементов матриц
,	Запятая применяется для разделения элементов матриц и оператора в строке ввода
;	Точка с запятой отделяет строки матриц, а точка с запятой в конце оператора (команды) отменяет вывод результата на экран
:	Двоеточие используется для указания диапазона (интервала изменения величины) и в качестве знака групповой операции над элементами матриц
()	Круглые скобки применяются для задания порядка выполнения математических операций, а также для указания аргументов функций и индексов матриц
.	Точка отделяет дробную часть числа от целой его части, а также применяется в составе комбинированных знаков ($.*$, $.^$, $./$, $.\backslash$)
...	Три точки и более в конце строки отмечают продолжение выражения на следующей строчке
%	Знак процента означает начало комментария
'	Апостроф указывает на символьные строки, а для включения самого апострофа в символьную строку нужно поставить два апострофа подряд

В командном окне в режиме диалога проводятся вычисления. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает введенное значение и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках. Строка ввода помечена знаком `>>`.

При работе с MATLAB в командном режиме действует простейший строчный редактор. Обратите особое внимание на применение клавиш **Up** и

Down (стрелки курсора "Вверх" и "Вниз"). Они используются для подстановки после маркера строки ввода `>>` ранее введенных строк из специального стека, например, для их исправления, дублирования или дополнения. При этом указанные клавиши обеспечивают перелистывание ранее введенных строк снизу вверх или сверху вниз.

Имена переменных должны начинаться с буквы. Знак `=` соответствует операции присваивания. Нажатие клавиши *Enter* заставляет систему вычислить выражение и показать результат. Если запись оператора не заканчивается символом `«;»`, то результат выводится в командное окно, в противном случае – не выводится. Если оператор не содержит знака присваивания `«=»`, то значение результата присваивается системной переменной *ans* (см. рисунок 1.2).

Все значения переменных, вычисленные в течение текущего сеанса работы, сохраняются в специально зарезервированной области памяти компьютера, называемой рабочим пространством системы MATLAB (**Workspace**).

Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу *Enter*.

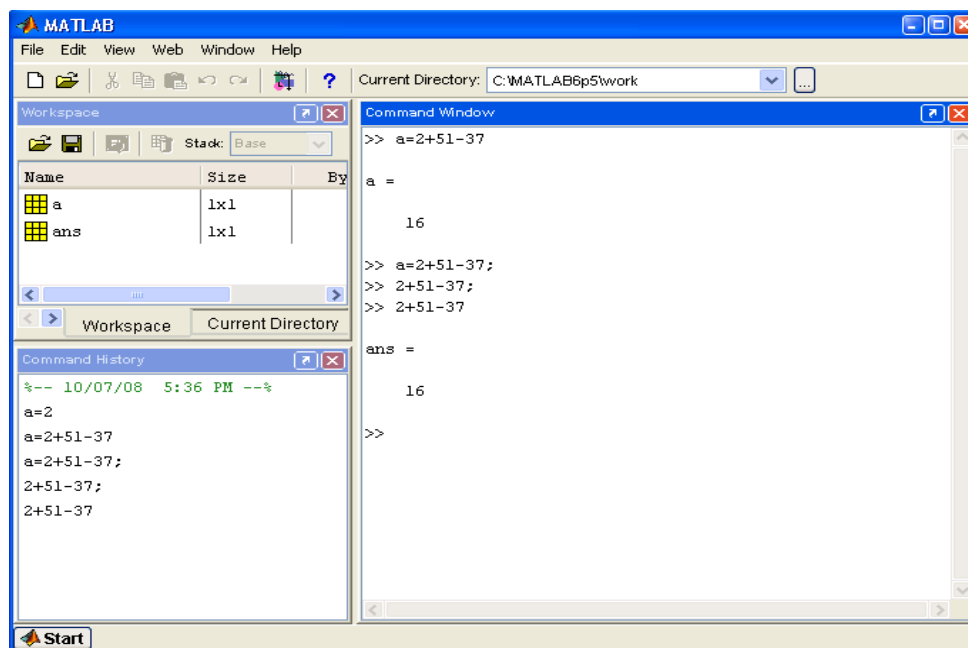


Рисунок 1.2 – Демонстрация выполнения команды присваивания

После окончания сеанса работы с системой MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню **File \Save Workspace As ...**. По умолчанию расширение имени файла *mat*, поэтому такие файлы принято называть МАТ-файлами.

Система MATLAB работает как с действительными, так и с комплексными числами. Перед использованием операций с комплексными числами необходимо определить переменную $i = \sqrt{-1}$ или $j = \sqrt{-1}$. В арифметических выражениях применяются следующие знаки операций:

- `+`, `-` – сложение, вычитание,
- `*` – умножение,

/ – деление слева направо;
 \ – деление справа налево;
 ^ – возведение в степень.

Система MATLAB позволяет вычислять различные математические функции. Следующие элементарные алгебраические функции имеют в качестве аргумента одно или два действительных (x , y) или одно комплексное (z) число (табл. 1.4).

Таблица 1.4 Элементарные алгебраические функции

Функция	Описание
abs (z), abs (x),	Вычисление модуля комплексного числа z или абсолютного значения действительного числа x .
angle (z)	Вычисление аргумента z .
sqrt (z), sqrt (x)	Вычисление квадратного корня чисел z и x
real (z)	Вычисление действительной части комплексного числа z .
imag (z)	Вычисление мнимой части комплексного числа z .
round (x)	Округление до целого.
fix (x)	Округление до ближайшего целого в сторону нуля.
rem (x , y)	Вычисление остатка от деления x на y .
exp (z)	Вычисление e в степени x .
log (z)	Вычисление натурального логарифма числа x .
log10 (z)	Вычисление десятичного логарифма числа x .

Система MATLAB предоставляет возможности для вычисления следующих тригонометрических и обратных тригонометрических функций переменной x (табл. 1.5).

Таблица 1.5 Тригонометрических функций

Функция	Описание
sin (x)	Вычисление синуса
cos (x)	Вычисление косинуса
tan (x)	Вычисление тангенса
asin (x)	Вычисление арксинуса
acos (x)	Вычисление арккосинуса
atan (x)	Вычисление арктангенса
atan2 (y , x)	Вычисление арктангенса по координатам точки

1.2 Порядок выполнения

1. В командном окне задать значения переменных, согласно варианту задания, представленному в таблице 1.6.
2. Записать выражение на языке MATLAB.

Таблица 1.6 Варианты заданий

№	Выражение	Переменные
1	2	3
1	$y = \sin \frac{a-x}{c} + 10^4 \sqrt[3]{\frac{a-kx^2}{2b}} + \frac{\cos kx^2}{\operatorname{tg} 3} - \frac{bc}{ax}$	$a = -1.3; b = 0.91;$ $c = 0.75; x = 2.32; k = 8.$
2	$y = -\frac{(x-d)(x^2+b^2)}{\sqrt[3]{x^2+b^2-cd}} + 10^{-3} \operatorname{tg} kn - \frac{\cos kx}{\sin 5}.$	$d = 1.25; b = 0.75; n = 4;$ $c = 2.2; x = 0.32; k = 2.$
3	$y = \operatorname{tg} ik + 10^3 e^{-5} + \sqrt[3]{\frac{10^2 xk }{(a+b)^2}} - \frac{ax^3-b}{(a+b)^2}.$	$i = 5; b = 2.35;$ $a = 25.2; x = 0.1; k = -2.$
4	$y = \frac{\sqrt{ c-d +(a+c)^2}}{\sin 2i} + 10^{-3} e^{ix} - \frac{ c-d +a^2}{\sqrt[3]{(a+c)^2}}.$	$a = -1.25; d = 2.5; i = 5;$ $c = 0.05; x = 1.35.$
5	$y = \frac{\ln kx }{\sin 7} - \sqrt{ x-a^2 } - \frac{10^4 a-b}{\cos kx} + \sqrt[3]{x-a^2} + c^3 x.$	$a = 0.93; b = 5.61;$ $c = 0.31; x = -2.5;$ $k = 2.$
6	$y = 10^4 \frac{ax}{b^2} - \left \frac{a-b}{kx} \right + \frac{\ln 3}{\sqrt[3]{ax^2+b^2}} - e^{-kx}$	$b = 0.35;$ $a = 3.5; x = 1.523; k = -2.$
7	$y = -\frac{ b-a }{kx} + 10^4 \sqrt[5]{ \cos kx } + \sqrt{\frac{abc}{2.4}} - \frac{0.7abc}{\sin 7}.$	$a = 1.7; b = -1.25;$ $c = -0.3; x = 2.5; k = 3.$
8	$y = \frac{ a^2-b^2 }{\sin kx} + 10^4 \sqrt[5]{ \sin kx-bc } - \frac{k^2+\operatorname{tg} 3k}{e^{kx}}.$	$a = 1.3; b = 2.42;$ $c = 0.83; x = 1.5;$ $k = 2.$
9	$y = \frac{\sqrt[3]{\ln x+a^2}}{0.47x^2} - \left 0.47x^2 - \frac{10^4}{7} \cos^2 k \right - \frac{c}{x}$	$c = 1.52;$ $a = -2.4; x = 0.29; k = 3.$
10	$y = \frac{1.5(a-b)^2}{ a-b c} + \frac{i}{5} + 10^3 \sqrt{ a-b } - \frac{(a+x^2)\cos 7}{ix^2+a^2bc}$	$a = -2.5; b = 1.35; i = 3;$ $c = -0.72; x = 2.75.$
11	$y = 10^4 \sin^2 i - \frac{0.32x^3+4x+b}{\cos ia} \sqrt[6]{0.32x^3-b+ b }$	$a = 3.5; b = -0.7; i = 2;$ $x = 0.8.$
12	$y = -\frac{\cos i}{\sin kx} + \frac{ax^2+ d }{(a+b)^2} - 10^4 \sqrt[6]{\frac{kx}{(a+b)^2}}.$	$d = -0.01; b = 1.25;$ $a = 4.72; i = 2;$ $x = 2.25; k = 3.$
13	$y = \cos k(x-a) + 10^{-4} \frac{(x+a)^3+x^4d}{k(x-a)^3} + \frac{\sqrt[5]{ x+a }}{2.4b}.$	$d = 0.95; b = 0.05; a = -3.25$ $x = 8.2; k = 4.$

Продолжение таблицы 1.6		
1	2	3
14	$y = \sqrt[5]{ ax^2 - b^3 } + \ln kx - \frac{e^{kx} + c^2}{\sin kx} - 10^{-3} \sqrt{2157}.$	$c = 1.72; b = -0.31; a = 2.01;$ $x = 0.48; k = 3.$
15	$y = \frac{1}{9} - 10^{-4} e^{kx} + \cos \sqrt{(x^2 + b)} + \frac{\sqrt{x^2 + b}}{0.4x} + \frac{\sin 3}{(x^2 + b)n}.$	$x = 2.5; b = 0.04;$ $k = 3; n = 5.$

1.3 Содержание отчета

1. Цель работы.
2. Пример расчета и вывода данных.

1.4 Контрольные вопросы

1. Для чего служит команда HELP?
2. Перечислите основные команды MATLAB для работы в режиме прямых вычислений.
3. С помощью какой команды устанавливается формат чисел?
4. Перечислите основные системные переменные MATLAB.
5. Приведите примеры математических функций системы MATLAB.

Лабораторная работа № 2

ОПЕРАЦИИ С ВЕКТОРАМИ И МАТРИЦАМИ В СИСТЕМЕ MATLAB

Цель занятия: изучение реализации средствами системы MATLAB основных операций с векторами и матрицами.

2.1 Основные теоретические сведения

По умолчанию все числовые переменные в MATLAB считаются матрицами, так что скалярная величина есть матрица первого порядка, а векторы являются матрицами, состоящими из одного столбца или одной строки. Матрицу можно ввести, задав ее элементы или считав данные из файла, а также в результате обращения к стандартной или написанной пользователем функции.

Матричные данные размещаются в памяти последовательно по столбцам. Элементы матрицы в пределах строки отделяются пробелами или запятыми. Непосредственное задание матрицы можно осуществить несколькими способами. Например, вектор-столбец, то есть матрица, вторая размерность которой равна единице, может быть присвоена переменной *A* вводом одной строки:

```
>> A=[7+4i; 4; 3.2]           % Ввод вектора-столбца
```

```
A =
```

```
7.0000 + 4.0000i
```

```
4.0000
```

```
3.2000
```

или вводом нескольких строк

```
>> A = [           % ввод вектора по строкам
```

```
7+4i
```

```
4
```

```
3.2];
```

Векторы могут быть сформированы как диапазоны – при помощи двоеточий, разделяющих стартовое значение, шаг и предельное значение. Если величина шага отсутствует, то по умолчанию его значение равно единице.

В результате *n:m:k* будет сформирован вектор, последний элемент которого не больше *k* для положительного шага *m*, и не меньше – для отрицательного: *[n, n+m, n+m+m, ...]*

Например:

```
>> a=1:2:5
```

```
a =
```

```
1 3 5
```

Задание диапазона используется также при организации цикла.

В таблице 2.1 представлен некоторый набор функций для создания матриц специального вида.

Таблица 2.1. Функции описания матриц

Функция	Описание
eye (m,n)	Единичная матрица размерности $m \times n$
zeros (m,n)	Нулевая матрица размерности $m \times n$
ones (m,n)	Матрица, состоящая из одних единиц размерности $m \times n$
rand (m,n)	Возвращает матрицу случайных чисел равномерно распределенных в диапазоне от 0 до 1, размерность $m \times n$
randn (m, n)	Возвращает матрицу размерности $m \times n$, состоящих из случайных чисел, имеющих гауссовское распределение
tril (A), triu (A)	Выделение нижней треугольной и верхней треугольной частей матрицы A
inv (A)	Нахождение обратной матрицы A
det (A)	Нахождение определителя (детерминанта) квадратной матрицы A

Обращение к элементу матрицы производится по правилу, – в круглых скобках после имени матрицы даются индексы, которые должны быть положительными целыми числами, указывающими номер строки и через запятую, номер столбца. Например, A(2,1) означает элемент из второй строки первого столбца матрицы A.

Для дальнейших примеров введем матрицу 2x2:

```
>> A=[1 2+5*i; 4.6 3]
A =
    1.0000    2.0000 + 5.0000i
    4.6000    3.0000
```

Чтобы изменить элемент матрицы, ему нужно присвоить новое значение:

```
>> A(2,2)=10          % Второй элемент второй строки

A =
    1.0000    2.0000 + 5.0000i
    4.6000    10.0000
```

Размер матрицы можно уточнить по команде **size**, а результат команды **size** можно использовать для организации новой матрицы.

Например, нулевая матрица того же порядка, что и матрица A, будет сформирована по команде

```
>> A2=zeros(size(A))
A2 =
     0     0
     0     0
```

С помощью двоеточия легко выделить часть матрицы. Например, вектор из первых двух элементов второго столбца матрицы A задаётся выражением:

```
>> A(1:2, 2)
ans =
    2.0000 + 5.0000i
    10.0000
```

Двоеточие само по себе означает строку или столбец целиком. Для удаления элемента вектора достаточно присвоить ему пустой массив – пару квадратных скобок []. Чтобы вычеркнуть одну или несколько строк (столбцов) матрицы нужно указать диапазон удаляемых строк (столбцов) для одной размерности и поставить двоеточие для другой размерности. Для нахождения длины вектора можно воспользоваться также командой **length**.

Набор арифметических операций в MATLAB для работы с матрицами состоит из стандартных операций сложения – вычитания, умножения – деления, операции возведения в степень и дополнены специальными матричными операциями (табл.2.2). Если операция применяется к матрицам, размеры которых не согласованы, то будет выведено сообщение об ошибке.

Для поэлементного выполнения операций умножения, деления и возведения в степень применяются комбинированные знаки (точка и знак операции). Например, если за матрицей стоит знак (^), то она возводится в степень, а комбинация (.^) означает возведение в степень каждого элемента матрицы. При умножении (сложении, вычитании, делении) матрицы на число соответствующая операция всегда производится поэлементно.

Таблица 2.2 Знаки операций

Символ	Назначение
+, -	Символы плюс и минус обозначают знак числа или операцию сложения и вычитания матриц, причем матрицы должны быть одной размерности
*	Знак умножения обозначает матричное умножение, для поэлементного умножения матрицы применяется комбинированный знак (.*)
'	Апостроф обозначает операцию транспонирования (вместе с комплексным сопряжением), транспонирование без вычисления сопряжения обозначается при помощи комбинированного знака (.)'
/	Левое деление
\	Правое деление
^	Оператор возведения в степень, для поэлементного возведения в степень применяется комбинированный знак (.^)

Проиллюстрируем различие обычного и поэлементного умножений при помощи следующего примера.

Введём матрицу *H* размера 2x2 и матрицу *D* из единиц той же размерности:

```
>> H=[0 1; 2 3], D=ones(size(H))
```

```
H =
```

```
0 1
2 3
```

```
D =
```

```
1 1
1 1
```

Перемножим матрицы, используя обычное умножение:

```
>> H*D
```

```
ans =
```

```
1 1
5 5
```

Теперь применим поэлементную операцию:

```
>> H.*D
```

```
ans =
```

```
0 1
2 3
```

Система MATLAB имеет ряд функций, предназначенных для обработки данных, заданных в матричной или векторной форме (таблица 2.3).

Таблица 2.3 Функции для работы с матрицами

Функция	Описание
size(A)	Возвращает массив, состоящий из числа строк и числа столбцов матрицы.
sum(A)	Возвращает сумму всех элементов по столбцу
mean(A)	Возвращает среднее значение столбца матрицы
std(A)	Возвращает среднеквадратическое отклонение столбца матрицы
min(A), max(A)	Возвращает минимум и максимум соответственно, по столбцу матрицы
sort(A)	Сортирует столбец матрицы по возрастанию
prod(A)	Вычисляет произведение всех элементов столбцов

Символы и текстовые строки в MATLAB вводятся при помощи простых кавычек. Во внутреннем представлении символы даны целыми числами. Конвертировать массив символов в числовую матрицу позволяет команда **double**. Обратная операция совершается по команде **char**. Печатаемые символы из стандартного набора ASCII представлены числами от 32 до 255.

Приведем примеры для данных команд. Вначале введем строку:

```
>> s = 'Привет'
```

```
s =
```

```
Привет
```

Отметим, что для ввода русских букв следует выбрать в меню File/ Preferences/ Command Windows Font шрифт с русской кодировкой.

```
>> h = [v + ' от MATLAB']
```

```
v =
```

```
Привет от MATLAB
```

Тот же результат получится, если вместо переменной *v* использовать строковую переменную *s*.

Для перевода численных данных в строковые переменные имеется ряд команд преобразования. В таблице 2.4 приведены некоторые функции для этих и обратных операций, а полный список можно получить по команде **help strfun**.

Таблица 2.4 Функции работы со строковыми переменными

Функция	Действие
num2str	Перевод числа в строку
int2str	Перевод целого числа в строку
mat2str	Преобразование матрицы в строку
str2mat	Объединение строк в матрицу
str2num	Преобразование строки в число
strcat	Объединение строк

2.2 Порядок выполнения

1. Ввод с клавиатуры векторов и матриц.

Ввести:

- произвольную вектор-строку (*v*), размерность 2;
- произвольный вектор-столбец (*w*), размерность 2;
- произвольную матрицу (*m*), размерности 2×2.

2. Генерация матриц специального вида.

Создать:

- матрицу с нулевыми элементами (*m0*), размерности 2×2;
- матрицу с единичными элементами (*m1*), размерности 2×2;
- матрицу с элементами, имеющими случайные значения (*mr*), размерности 2×2;
- матрицу с единичными диагональными элементами (*me*), размерности 2×2.

3. Вычисление матрицы *M* по формуле, представленной в таблице 2.5.

4. Изучение функций обработки данных:

- определение числа строк и столбцов матрицы *M*;
- определение максимального элемента матрицы *M*;
- определение минимального элемента матрицы *M*;
- суммирование элементов матрицы *M*;
- перемножение элементов матрицы *M*.

Таблица 2.5 Варианты заданий

№ варианта	Задание	№ варианта	Задание
1	$M = v * w + m + mr * me$	11	$M = m * w + mr * v'$
2	$M = m + mr * me$	12	$M = m * mr + w * v$
3	$M = (v/m) * (mr + me)$	13	$M = m + mr - 100$
4	$M = w * v + mr * me$	14	$M = v' + w + mr * w$
5	$M = m * mr + me$	15	$M = m + m1' * me'$
6	$M = m * mr + 100$	16	$M = (v/m) * (mr + me)$
7	$M = v * w + mr - m$	17	$M = v * mr + v * m1$
8	$M = m + mr * me - 10$	18	$M = m' + mr / 100$
9	$M = m * w + mr * v'$	19	$M = 10 * v + w' * mr * m$
10	$M = m' + mr * me$	20	$M = m' + mr * me$

2.3 Содержание отчета

1. Цель работы.
2. Описание ввода с клавиатуры векторов и матриц.
3. Описание команд генерации матриц специального вида.
4. Описание основных функций обработки данных.

2.4 Контрольные вопросы

1. Как осуществляется ввод вектора–строки?
2. Как осуществляется ввод вектора–столбца?
3. Как осуществляется ввод матрицы?
4. Для чего служат команды zeros, ones, rand, eye?
5. Как определяется число строк и столбцов матрицы?
6. Какие операции служат для определения минимального и максимального элемента матрицы?

Лабораторная работа № 3

ПРОГРАММИРОВАНИЕ В СРЕДЕ MATLAB

Цель занятия: ознакомиться с операциями отношения, логическими операциями и условными операторами, приобрести навыки их использования при разветвленных вычислениях.

3.1 Основные теоретические сведения

В MATLAB особое значение имеют файлы двух типов — с расширениями `.mat` и `.m`. Первые являются бинарными файлами, в которых могут храниться значения переменных, вторые представляют собой текстовые файлы, содержащие внешние программы, определения команд и функций системы. Именно к ним относится большая часть команд и функций, в том числе задаваемых пользователем для решения своих специфических задач.

Многооконный редактор–отладчик с пустым окном редактирования *m-файлов* можно вызвать командой **Edit** из командной строки или командой меню **File > New > M-file** (рисунок 3.1).

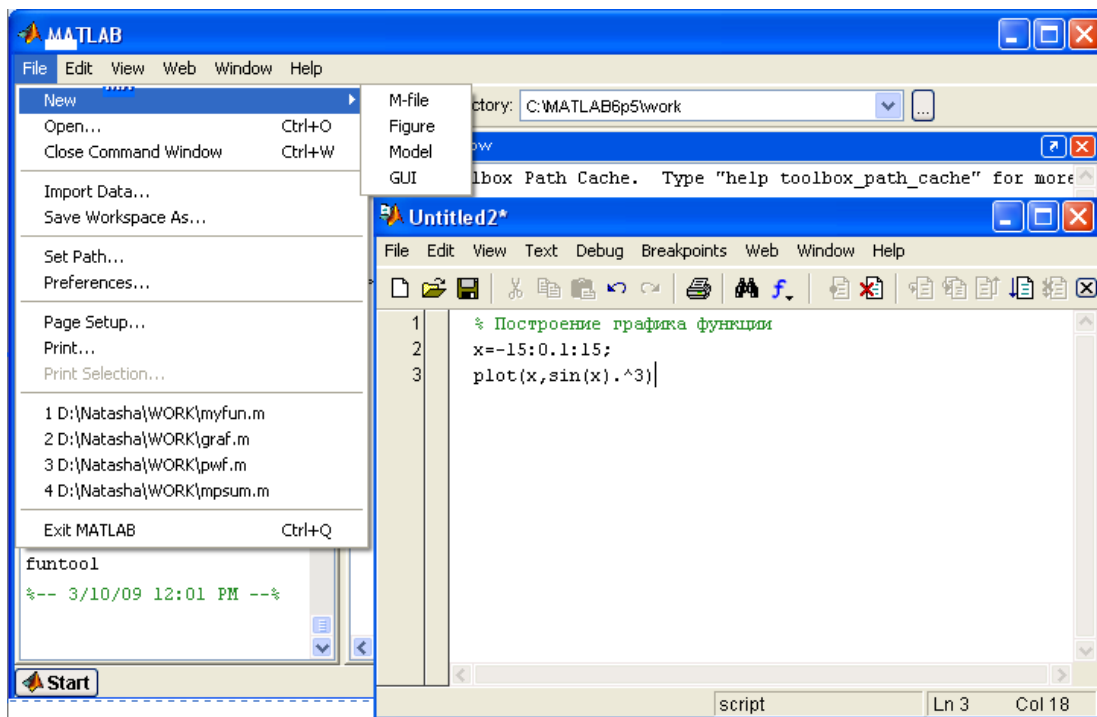


Рисунок 3.1– Многооконный редактор–отладчик

После этого в окне редактора можно создать свой файл, а также пользоваться средствами его отладки и запуска. Для запуска файла его необходимо записать на диск, используя команду **Save as** в меню **File** редактора. Редактор–отладчик *m-файлов* выполняет синтаксическую проверку программного кода по мере ввода текста. При этом используется следующее цветовое выделение:

- ключевые слова языка программирования — синий цвет;
- операторы, константы и переменные — черный цвет;

- комментарии после знака % — зеленый цвет;
- символьные переменные (в апострофах) — коричневый цвет;
- синтаксические ошибки — красный цвет.

Благодаря цветовому выделению вероятность синтаксических ошибок резко снижается

М-файлы, создаваемые редактором-отладчиком, делятся на два класса: файлы-сценарии, не имеющие входных параметров и файлы-функции, имеющие входные параметры. Файл-сценарий, именуемый также *script-файлом*, является просто записью серии команд без входных и выходных параметров. Он имеет следующую структуру:

```
%Основной комментарий
%Dополнительный комментарий
Тело файла с любыми выражениями
```

Важны следующие свойства файлов – сценариев:

- 1) они не имеют входных и выходных аргументов;
- 2) работают с данными из рабочей области;
- 3) в процессе выполнения не компилируются;
- 4) представляют собой зафиксированную в виде файла последовательность операций, полностью аналогичную той, что используется в сессии.

Рассмотрим следующий файл-сценарий (рис. 3.2):

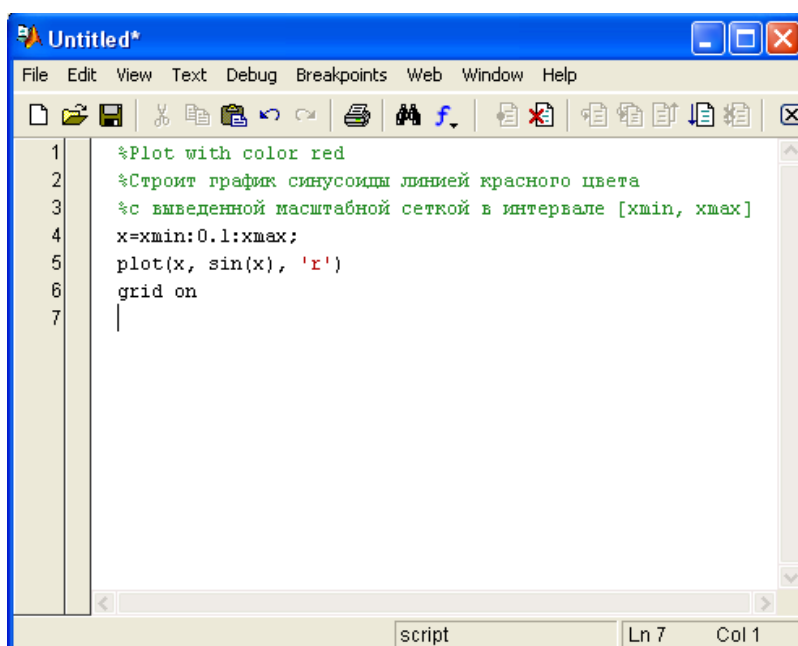


Рисунок 3.2 – Создание файла-сценария в MATLAB

Первые три строки здесь — это комментарий, остальные — тело файла. Обратите внимание на возможность задания комментария на русском языке. Знак % в комментариях должен начинаться с первой позиции строки. Необходимо отметить, что такой файл нельзя запустить без предварительной подготовки, сводящейся к заданию значений переменным *xmin* и *xmax*, использо-

ванным в теле файла. Это следствие первого свойства файлов-сценариев — они работают с данными из рабочей области. Имена файлов-сценариев нельзя использовать в качестве параметров функций, поскольку файлы-сценарии не возвращают значений. Можно сказать, что файл-сценарий — это простейшая программа на языке программирования MATLAB.

М-файл-функция является типичным объектом языка программирования системы MATLAB. Одновременно он является полноценным модулем с точки зрения структурного программирования, поскольку содержит входные и выходные параметры и использует аппарат локальных переменных. Структура такого модуля с одним выходным параметром выглядит следующим образом:

```
function var = f_name(Список_параметров)  
%Основной комментарий  
%Дополнительный комментарий  
Тело файла с любыми выражениями  
var=выражение
```

М-файл-функция имеет следующие свойства:

- 1) он начинается с объявления `function`, после которого указывается имя переменной `var` — выходного параметра, имя самой функции `f_name` и список ее входных параметров;
- 2) функция возвращает свое значение и может использоваться в математических выражениях;
- 3) все переменные, имеющиеся в теле файла-функции, являются локальными, т. е. действуют только в пределах тела функции;
- 4) файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;
- 5) правила вывода комментариев те же, что у файлов-сценариев;
- 6) при обнаружении файла-функции он компилируется и затем исполняется, а созданные машинные коды хранятся в рабочей области системы MATLAB.

Последняя конструкция `var = выражение` вводится, если требуется, чтобы функция возвращала результат вычислений. Приведенная форма файла-функции характерна для функции с одним выходным параметром. Если выходных параметров больше, то они указываются в квадратных скобках после слова `function`. При этом структура модуля имеет следующий вид:

```
function [var1,var2,...] = f_name(Список_параметров)  
%Основной комментарий  
%Дополнительный комментарий  
Тело файла с любыми выражениями  
var1=выражение  
var2=выражение
```

Если функция используется как имеющая единственный выходной параметр, но имеет ряд выходных параметров, то для возврата значения будет использоваться первый из них. Это зачастую ведет к ошибкам в математических вычислениях. Поэтому, как отмечалось, данная функция используется как отдельный элемент программ вида `[var1, var2] = f_name (Список_параметров)`. После его применения переменные выхода `var1`, `var2` становятся определенными и их можно использовать в последующих математических выражениях и иных сегментах программы.

Для организации диалогового ввода и вывода используются следующие операторы, представленные в таблице 3.1.

Таблица 3.1 Операторы диалогового ввода/вывода

Оператор	Синтаксис	Назначение
INPUT	<code>x = input('<приглашение>')</code>	Для ввода данных с клавиатуры
DISP	<code>disp (<переменная или текст в апострофах>)</code>	Для вывода на дисплей

Приведем простой пример диалоговой программы, которая служит для многократного вычисления длины окружности по вводимому пользователем значению радиуса r (рис.3.3).

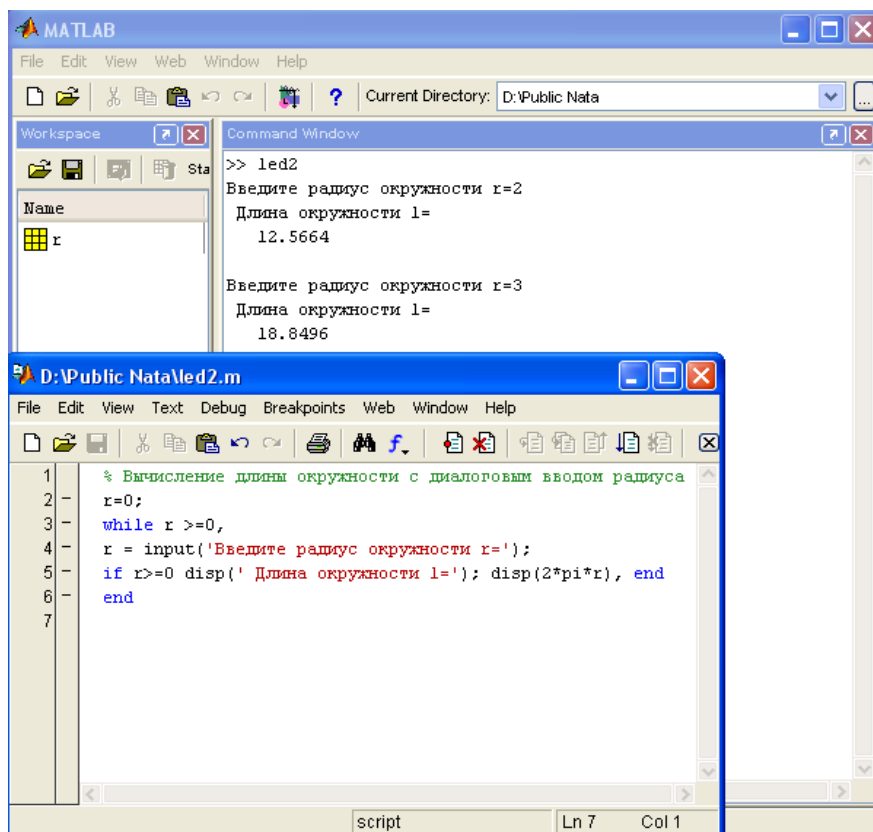


Рисунок 3.3 – Пример диалоговой программы

Для организации ветвлений служат условные операторы. Конструкции условных операторов:

```

1)
if <условие>
<операторы>
end

```

Операторы (тело выражения) выполняются только в том случае, если условие истинно, если условие ложно, то тело выражения не выполняется.

```

2) if <условие>
<операторы 1>
else
<операторы 2>
end

```

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

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

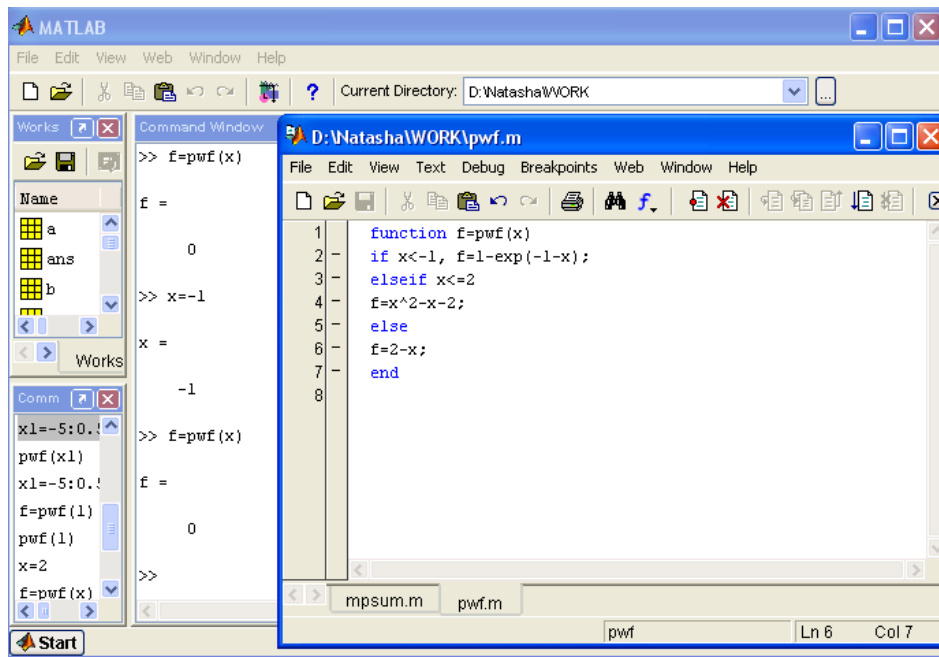


Рисунок 3.4 – Листинг программы для вычисления значения функции

В системе MATLAB могут применяться следующие операторы сравнения, приведенные в таблице 3.2.

Таблица 3.2 Операторы сравнения

Символ	Назначение	Имя функции
<	Меньше	lt
>=	Больше или равно	ge
>	Больше	gt
<=	Меньше или равно	le
==	Равно	eq
~=	Не равно	ne

Операции (==, ~=) проводят сравнение вещественных и мнимых частей комплексных чисел, а операции (>, <, >=, <=) – только вещественных частей.

Логические операции можно записывать в виде функций (табл. 3.3).

Таблица 3.3 Логические операции

Символ	Назначение	Имя функции
&	Логическое «и»	and
	Логическое «или»	or
~	Отрицание	not

Результатом логических операций являются числа 0 (false) и 1(true).

В системе MATLAB есть две разновидности операторов цикла – условный и арифметический. Для повторения операторов нефиксированное число раз используется оператор цикла с предусловием:

```
while <условие>
<операторы>
end
```

Операторы выполняются, если переменная <условие> имеет ненулевые элементы.

Арифметический оператор цикла имеет следующий вид:

```
for <имя> = <НЗ>: <Шаг>: <КЗ>
<операторы>
end,
```

где <имя> – имя управляющей переменной цикла,

<НЗ> – начальное значение управляющей переменной,

<КЗ> – конечное значение управляющей переменной,

<Шаг> – приращение значений переменной <имя> в ходе ее изменения от значения <НЗ> до значения <КЗ>. Если параметр <Шаг> не указан, по умолчанию его значение принимается равным единице.

При работе с циклом **for** допустимо использование оператора прерывания цикла **break**. При работе данного оператора работа цикла завершается, и управление передается на следующий после конца цикла оператор.

Ход работы программы может определяться значением некоторой переменной (переключателя). Такой альтернативный способ ветвления программы основан на использовании оператора переключения **switch**.. Оператор **switch** содержит блоки, начинающиеся со слова **case**, после каждого **case** записывается через пробел то значение переключателя, при котором выполняется данный

блок. Последний блок начинается со слова **otherwise**, его операторы работают в том случае, когда ни один из блоков **case** не был выполнен. Если хотя бы один из блоков **case** выполнен, то происходит выход из оператора **switch** и переход к оператору, следующему за **end**.

Предположим, что требуется найти количество единиц и минус единиц в заданном массиве и, кроме того, найти сумму всех элементов, отличных от единицы и минус единицы. Листинг программы содержит файл–функцию, которая по заданному массиву возвращает число минус единиц в первом выходном аргументе, число единиц — во втором, а сумму — в третьем (рис.3.5).

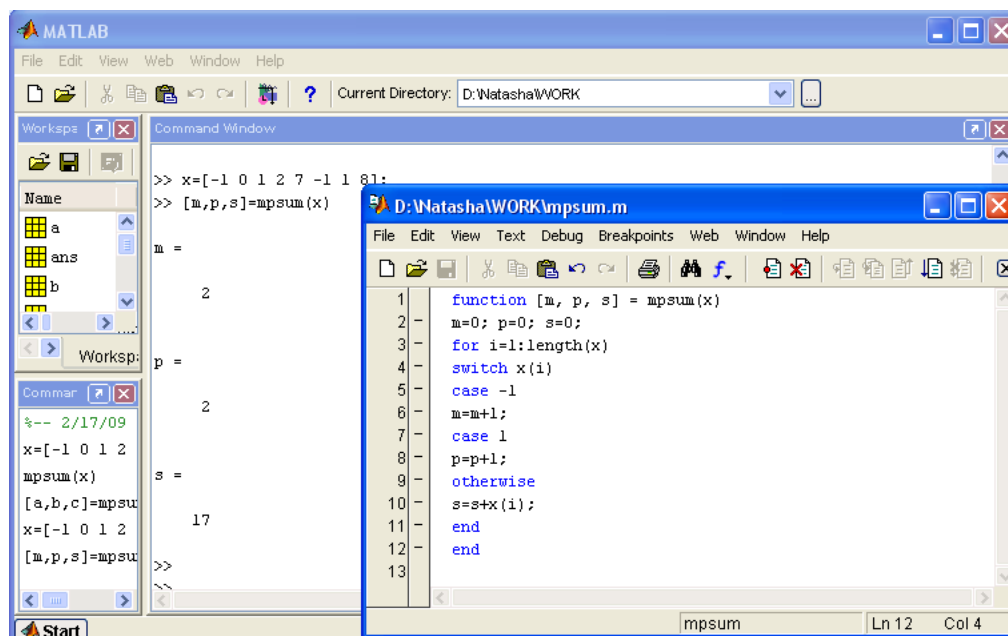


Рисунок 3.5 – Листинг программы

Для остановки программы используется оператор **pause**. Он используется в следующих формах:

- a) `pause` — останавливает вычисления до нажатия любой клавиши;
- b) `pause(N)` — останавливает вычисления на N секунд;
- c) `pause on` — включает режим отработки пауз;
- d) `pause off` — выключает режим отработки пауз.

3.2 Порядок выполнения

1. Из файла-сценария с помощью функции диалогового ввода ввести с клавиатуры все необходимые данные. Выполнить расчет с использованием условных операторов и вывести результаты в командное окно (таблица 3.4).

Таблица 3.4 Варианты заданий

№ варианта	Задание
1	2
1	Найти сумму положительных из четырех заданных переменных.

2	Найти максимальное значение из четырех заданных переменных и вывести ее.
3	Заданы четыре переменные. Наименьшую из них заменить на сумму остальных.
4	Заданы четыре переменные. Подсчитать количество отрицательных и количество нулевых из них.
5	Найти произведение отрицательных из четырех заданных переменных.
6	Заданы две фигуры: квадрат задан длиной стороны, а круг – длиной радиуса. Определить, какая из них имеет большую площадь и во сколько.
7	Заданы четыре переменные. Все отрицательные из них заменить абсолютными значениями и увеличить в 2 раза.
8	Заданы четыре переменные, подсчитать количество равных нулю, положительных и отрицательных.
9	Даны четыре переменные. Найти среди них переменные, наиболее близкие по значению к x .
10	Заданы четыре переменные. Все положительные из них заменить отрицательными значениями, умноженными на 5.
11	Найти минимальное и максимальное значения из четырех заданных переменных.
12	Заданы четыре значения. Определить какие из них целые.
13	Заданы четыре переменные. Подсчитать количество и произведение значений, попавших в интервал $[1\ 5]$.
14	Заданы четыре переменные. Подсчитать количество отрицательных и количество нулевых из них.
15	Найти произведение отрицательных из четырех заданных переменных.

2. Написать файл-функцию с использованием операторов ветвления и циклов, на основании вариантов задания, представленных в таблице 3.5.

Таблица 3.5 Варианты заданий

№	Вход. массив	Формируемый массив	Задача
1	2	3	4
1	$A_{3 \times 3}$	$B_{3 \times 3}, b_{ij} = \begin{cases} a_{ij}, i < j \\ a_{ji}^2, i \geq j \end{cases}$	Сформировать массив $A1$ из минимальных элементов строк матрицы A и массив $B1$ из минимальных элементов строк матрицы B . Среди элементов $A1$ и $B1$ найти максимальный
2	A_3	$B_3, b_i = \sin(i^2), i = 1 \dots 3$	Сформировать массив C – сумму элементов массивов A и B . Найти максимальное значение массивов A, B, C .

3	$A_{3 \times 3}$	$B_{3 \times 3}, b_{ij} = \sin(i) * \sin(j)$ $i = 1 \dots 3, j = 1 \dots 3.$	Определить минимальные элементы в матрицах A и B (mA и mB). Вычислить $C=A*B*mA*mB$.
4	$A_{3 \times 3}$	$B_{3 \times 3}, b_{ij} = \begin{cases} 2*i+3*j, i=j, \\ 5*i+2*j, иначе. \end{cases}$	Сформировать массив $A1$ из максимальных элементов строк матрицы A и массив $B1$ из максимальных элементов строк матрицы B . Упорядочить массив $A1$ по возрастанию, а массив $B1$ – по убыванию.
5	$A_{3 \times 3}$	$B_3, b_i = \sin(i) + \cos(i)$ $i = 1 \dots 3.$	Определить максимальные элементы в матрице A и массиве B (mA и mB). Вычислить $C=A*B*mA*mB$.
6	$A_{3 \times 3}$	$B_3, b_i = \log(2i + \cos(i)),$ $i = 1 \dots 3.$	Сформировать массив $A1$ из средних значений элементов строк матрицы A . Упорядочить массив $A1$ по возрастанию, а B – по убыванию. Осуществить поэлементное умножение $A1$ и B .
7	$A_{3 \times 3}$	$B_3, b_i = \sin(\ln(i) + \cos(i)),$ $i = 1 \dots 3.$	Заменить строку матрицы A , содержащую максимальный элемент, массивом B . Вычислить сумму элементов полученной матрицы.
8	A_3	$B_{3 \times 3}, b_{ij} = \begin{cases} 1 + \cos(i - j), i > j, \\ 1 - \sin(i + j), иначе. \end{cases}$	Массив A упорядочить по возрастанию и заменить им последнюю строку матрицы B .
9	A_3	$B_3, b_i = i * \log(i^2) + \sin(i),$ $i = 1 \dots 3.$	Упорядочить по возрастанию массивы A и B . Осуществить поэлементное деление упорядоченных массивов. Определить произведение элементов результирующего массива.
10	$A_{3 \times 3}$	$B_3, b_i = i * \sin(j) * \log(i)$ $i = 1 \dots 3$	Вычислить произведение элементов матрицы A (pA) и сумму элементов матрицы B (cB). Вычислить матрицу $C=pA*cB*A*B'$.
11	$A_{3 \times 3}$	$B_3,$ $b_i = i * \sin(i) + j * \cos(i)$ $i = 1 \dots 3, j = 1 \dots 3.$	Определить минимальные элементы в матрицах A и B (mA и mB). Вычислить $C=A*B*mA*mB$.
12	$A_{3 \times 3}$	$B_3,$ $b_i = \sin(2i) + \cos(3i)$ $i = 1 \dots 3.$	Сформировать массив $A1$ из максимальных элементов строк матрицы A . Осуществить поэлементное умножение $A1*B$. Упорядочить массив $A1$ по возрастанию.
13	$A_{3 \times 3}$	$B_3,$ $b_i = \sin(\ln(i) + i * \cos(i))$ $i = 1 \dots 3.$	Заменить строку матрицы A , содержащую минимальный элемент, массивом B . Вычислить произведение элементов полученной матрицы.

Продолжение таблицы 3.5

1	2	3	4
14	A_3	$B_{3 \times 3},$ $b_{ij} = \begin{cases} i + \cos(i + j), i > j, \\ j - \sin(i - j), \text{ иначе.} \end{cases}$ $i = 1 \dots 3, j = 1 \dots 3.$	Массив A упорядочить по убыванию и заменить им первую строку матрицы B .
15	A_3	$B_3,$ $b_i = \cos(i^2), i = 1 \dots 3.$	Сформировать массив C – произведение элементов массивов A и B . Найти максимальные и минимальные значения массивов A, B, C .

3.3 Содержание отчета

1. Цель занятия.
2. Листинг программ и результаты выполнения программ.

3.4 Контрольные вопросы

1. Как осуществляется диалоговый ввод и вывод?
2. Для чего используются условные операторы?
3. Чем отличаются файлы-сценарии от файлов-функций?

Лабораторная работа № 4

РАБОТА С ГРАФИКОЙ СРЕДСТВАМИ MATLAB

Цель занятия: изучение основных операторов графики системы MATLAB и создание программ, реализующих графический вывод.

4.1 Основные теоретические сведения

Одно из достоинств системы MATLAB — обилие средств графики, начиная от команд построения простых графиков функций одной переменной в декартовой системе координат и кончая комбинированными и презентационными графиками с элементами анимации, а также средствами проектирования графического пользовательского интерфейса (GUI). Особое внимание в системе уделено трехмерной графике с функциональной окраской отображаемых фигур и имитацией различных световых эффектов.

Для отображения функций одной переменной $y(x)$ используются графики в декартовой (прямоугольной) системе координат. При этом обычно строятся две оси — горизонтальная X и вертикальная Y , и задаются координаты x и y , определяющие узловые точки функции $y(x)$. Поскольку MATLAB — матричная система, совокупность точек $y(x)$ задается векторами X и Y одинакового размера.

Команда **plot** (X, Y) служит для построения графиков функций в декартовой системе координат, координаты точек (x, y) берутся из векторов одинакового размера Y и X . Если X или Y — матрица, то строится семейство графиков по данным, содержащимся в колонках матрицы.

Команда **plot**(X, Y, S) аналогична команде **plot**(X, Y), но тип линии графика можно задавать с помощью строковой константы S .

Значениями константы S могут быть следующие символы, которые представлены в таблице 4.1.

Таблица 4.1 Задание типа линии

Маркер типа линии	
Маркет	Тип линии
-	Непрерывная
--	Штриховая
:	Пунктирная (точками)
-.	Штрих-пунктирная
Маркер цвета графика	
Продолжение таблицы 4.1	
Маркер	Цвет графика
c	Голубой
m	Фиолетовый
y	Желтый

Продолжение таблицы 4.1	
r	Красный
g	Зеленый
b	Синий
w	Белый
k	Черный
Тип проставляемой точки	
Маркер	Тип точки
.	Точка
+	Плюс
*	Звездочкой
o	Кружком (указывается латинская буква o)
x	Крестиком (указывается латинская буква x)

Таким образом, с помощью строковой константы S можно изменять цвет линии, представлять узловые точки различными отметками (точка, окружность, крест, треугольник с разной ориентацией вершины и т. д.) и менять тип линии графика.

Команда **plot**($X1, Y1, S1, X2, Y2, S2, X3, Y3, S3, \dots$) строит на одном графике ряд линий, представленных данными вида (X, Y, S) , где X и Y — векторы или матрицы, а S — строки. С помощью такой конструкции возможно построение, например, графика функции линией, цвет которой отличается от цвета узловых точек. При отсутствии указания на цвет линий и точек он выбирается автоматически из таблицы цветов (белый исключается). Если линий больше шести, то выбор цветов повторяется.

Иногда требуется сравнить поведение двух функций, значения которых сильно отличаются друг от друга. График функции с небольшими значениями практически сливается с осью абсцисс, и установить его вид не удастся. В этой ситуации помогает функция **plotyy**, которая выводит графики в окно с двумя вертикальными осями, имеющими подходящий масштаб.

Трехмерные поверхности обычно описываются функцией двух переменных $z(x, y)$. Специфика построения трехмерных графиков требует не просто задания ряда значений x и y , то есть векторов x и y . Она требует определения для X и Y двумерных массивов — матриц.

Для создания таких массивов служит функция **meshgrid**. В основном она используется совместно с функциями построения графиков трехмерных поверхностей. Функция **meshgrid** записывается в следующих формах:

– $[X, Y, Z] = \text{meshgrid}(x, y, z)$ — возвращает трехмерные массивы, используемые для вычисления функций трех переменных и построения трехмерных графиков;

– $[X, Y] = \text{meshgrid}(x, y)$ — преобразует область, заданную векторами x и y , в массивы X и Y , которые могут быть использованы для вычисления функ-

ции двух переменных и построения трехмерных графиков. Строки выходного массива X являются копиями вектора x , а столбцы Y — копиями вектора y .

Команда **plot3(...)** является аналогом команды **plot(...)**, но относится к функции двух переменных $z(x, y)$. Она строит аксонометрическое изображение трехмерных поверхностей и представлена следующими формами:

- **plot3**(x, y, z) — строит массив точек, представленных векторами x, y и z , соединяя их отрезками прямых. Эта команда имеет ограниченное применение;

- **plot3**(X, Y, Z, S) — обеспечивает построения со спецификацией стиля линий и точек;

- **plot3**($x1, y1, z1, s1, x2, y2, z2, s2, \dots$) — строит на одном рисунке графики нескольких функций $z1(x1, y1)$, $z2(x2, y2)$ и т. д. со спецификацией линий и маркеров каждой из них.

Наиболее представительными и наглядными являются сетчатые графики поверхностей с заданной или функциональной окраской. В названии их команд присутствует слово **mesh**. Имеются три группы таких команд:

- **mesh**(X, Y, Z, C) — выводит в графическое окно сетчатую поверхность $Z(X, Y)$ с цветами узлов поверхности, заданными массивом C ;

- **mesh**(X, Y, Z) — аналог предшествующей команды при $C=Z$.

В данном случае используется функциональная окраска, при которой цвет задается высотой поверхности. Функция **mesh** возвращает дескриптор для объекта класса **surface**. Ниже приводится пример применения команды **mesh**:

```
>> [X, Y]=meshgrid([-3:0.15:3]);  
>> Z=X.^2+Y.^2;  
>> mesh(X, Y, Z)
```

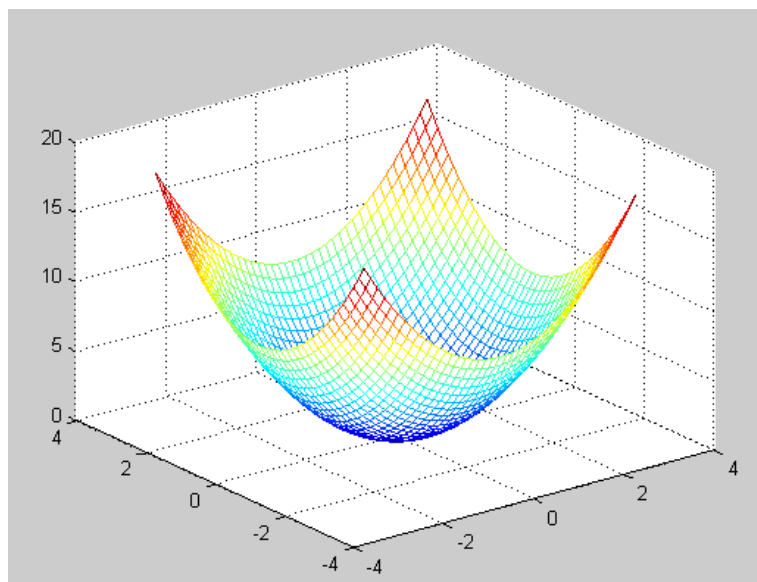


Рисунок 4.1 – График поверхности, созданный командой **mesh(X,Y,Z)**

После того как график уже построен, MATLAB позволяет выполнить его форматирование или оформление в нужном виде. Так, для установки над графиком титульной надписи используется следующая команда **title('string')** —

установка на двумерных и трехмерных графиках титульной надписи, заданной строковой константой 'string'.

Для установки надписей возле осей x , y и z используются следующие команды: **xlabel('String')**, **ylabel('String')**, **zlabel('String')**.

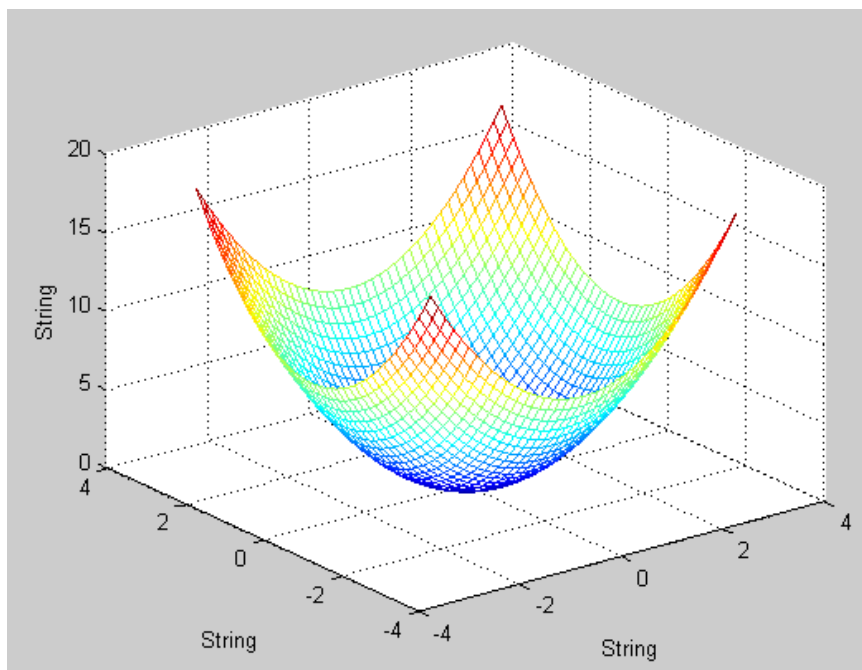


Рисунок 4.2 – Установка надписей с использованием команды: **xlabel('String')**, **ylabel('String')**, **zlabel('String')**

Часто возникает необходимость добавления текста в определенное место графика, например для обозначения той или иной кривой графика. Для этого используется команда **text**:

- **text(X,Y, 'string')** — добавляет в двумерный график текст, заданный строковой константой 'string', так что начало текста расположено в точке с координатами (X, Y). Если X и Y заданы как одномерные массивы, то надпись помещается во все позиции [x(i), y(i)];

- **text(X,Y, Z, 'string')** — добавляет в трехмерный график текст, заданный строковой константой 'string', так что начало текста расположено в позиции, заданной координатами X, Y и Z.

Очень удобный способ ввода текста предоставляет команда **gtext**:

- **gtext('string')** — задает выводимый на график текст в виде строковой константы 'string' и выводит на график перемещаемый мышью маркер в виде крестика. Установив маркер в нужное место, достаточно щелкнуть любой кнопкой мыши для вывода текста.

Пояснение в виде отрезков линий со справочными надписями, размещаемое внутри графика или около него, называется легендой. Для создания легенды используются различные варианты команды **legend**:

- **legend(string1, string2,..., strings)** — добавляет к текущему графику легенду в виде строк, указанных в списке параметров;

```
>> legend('график')
```

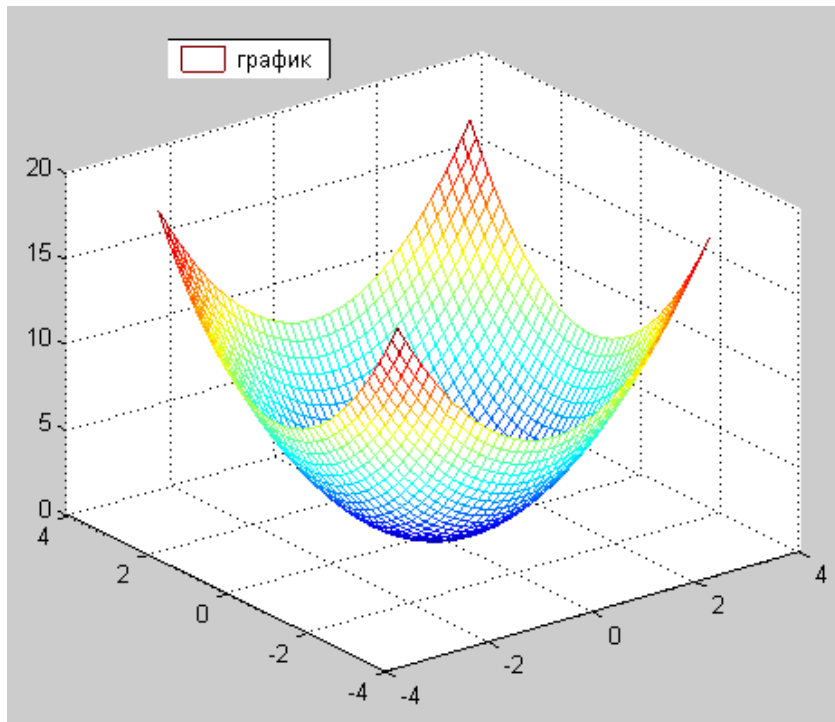


Рисунок 4.3 – График с пояснениями

legend (Pos) — помещает легенду в точно определенное место, специфицированное параметром Pos:

- Pos=0 — лучшее место, выбираемое автоматически;
- Pos=1 — верхний правый угол;
- Pos=2 — верхний левый угол;
- Pos=3 — нижний левый угол;
- Pos=4 — нижний правый угол;
- Pos=-1 — справа от графика.

При добавлении легенды следует учесть, что порядок и количество аргументов команды **legend** должны соответствовать порядку вывода графиков и их количеству

Обычно графики выводятся в режиме автоматического масштабирования. Следующие команды класса **axis** меняют эту ситуацию:

- **axis**([XMIN XMAX YMIN YMAX]) — установка диапазонов координат по осям x и y для текущего двумерного графика;
- **axis**([XMIN XMAX YMIN YMAX ZMIN ZMAX]) – установка диапазонов координат по осям x, y и z текущего трехмерного графика;
- **axis** auto — установка параметров осей по умолчанию;

В математической, физической и иной литературе при построении графиков в дополнение к разметке осей часто используют масштабную сетку. Команды **grid** позволяют задавать построение сетки или отменять это построение:

- **grid** on — добавляет сетку к текущему графику;
- **grid** off — отключает сетку.

Во многих случаях желательно построение многих наложенных друг на друга графиков в одном и том же окне. Для этого служит команда продолжения графических построений **hold**. Она используется в следующих формах:

- **hold on** — обеспечивает продолжение вывода графиков в текущее окно, что позволяет добавлять последующие графики к уже существующим;
- **hold off** — отменяет режим продолжения графических построений;

Бывает, что в одном окне надо расположить несколько координатных осей с различными графиками без наложения их друг на друга. Для этого используются команды **subplot**, применяемые перед построением графиков:

- **subplot(*m*, *n*, *p*)** — разбивает графическое окно на $m \times n$ подокон, при этом *m* — число подокон по горизонтали, *n* — число подокон по вертикали, а *p* — номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам).

Проиллюстрируем работу функции **subplot** (см рис. 4.4):

```
>> subplot(3, 2, 1); plot(x,y);  
>> subplot(3, 2, 4); plot(x,y);  
>> subplot(3, 2, 5); plot(x,y);
```

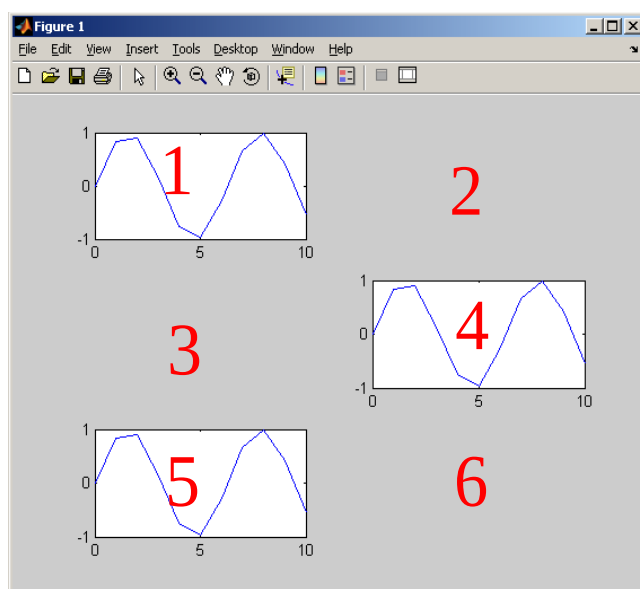


Рисунок 4.4 – Работа функции subplot

Было сформировано 3 строки и два столбца полей для вывода графиков. Обращение к каждому конкретному полю происходит с указанием его номера. Нумерация происходит слева направо и снизу вверх

4.2 Порядок выполнения

1. Составление и отладка программы для вывода графиков функций f_1 , f_2 , f_3 на основании задания из таблицы 4.2. Вывод графиков должен быть осуществлен в одном окне, графики должны быть подписаны, отмасштабированы.

Таблица 4.2 Варианты заданий

Номер варианта	$f1$	$f2$	$f3$	$f4$
1	2	3	4	5
1	$\sin(x)$	$\cos(x)$	x^2	$\cos(r)/r$
2	e^x	x^2	x	$\cos^2(r)/r$
3	$\sin(x) + \cos(x)$	$\cos(x) + x^2$	$x^2 + \lg(x)$	$\cos(r^2)/r$
4	$\sin(x) + e^x$	$\sin(x) + x^2$	$\sin(x) + x$	$\cos(r)/r^2$
5	$x * \sin(x)$	$x * \cos(x)$	x^2	$(\cos(r)/r)^2$
6	xe^x	$\sin(x) + x^2$	$\sin(x) + x$	$\sin^2(r)/r$
7	$\sin(x) * \cos(x)$	$\cos(x) * x^2$	$x^2 \lg(x)$	$\sin(r^2)/r$
8	$\sin(x)e^x$	$\sin(x) * x^2$	$\sin(x) * x$	$\sin(r)/r^2$
9	$\sin^2(x)$	$\cos^2(x)$	x	$(\sin(r)/r)^2$
10	$\sin(x) * e^x$	$\sin(x) * x^2$	$\sin(x) * x$	$r + \cos(r)/r$
11	$\sin^2(x) + \cos^2(x)$	$\cos(x) + x^2$	$x^2 + \lg(x)$	$r + \cos^2(r)/r$
12	$\sin(x) + e^x$	$\sin^2(x) + x^2$	$\sin^2(x) + x$	$r + \cos(r^2)/r$
13	$x * \sin(x)$	$\sin(x) + x^2$	$\sin(x) + x$	$r + \cos(r)/r^2$
14	$\sin(x) + \cos(x)$	$\cos(x) * x^2$	$x^2 \lg(x)$	$r + (\cos(r)/r)^2$
15	$\sin^2(x)$	$\sin(x) + x^2$	$\sin(x) * x$	$\sin^2(r)/r$

2. Составление и отладка программы для вывода графика трехмерной поверхности для функции $f4$ ($r = \sqrt{x^2 + y^2}$) задания из таблицы 4.2.

3. Написать файл-функцию для вычисления кусочно-заданной функции (табл. 4.3) и построить ее график.

Таблица 4.3 Варианты заданий кусочно-заданной функции

№ вари- анта	Функция	№ вари- анта	Функция
1	2	3	4
1	$y = \begin{cases} \frac{1+x^2}{\sqrt{1+x^4}}, x \leq 0 \\ 2x + \frac{\sin^2 x}{3+x}, x > 0 \end{cases}$	2	$y = \begin{cases} 3\sin x - \cos^2 x, x \leq 0 \\ \frac{3\sqrt{1+x^4}}{\ln(x+5)}, x > 0 \end{cases}$
3	$y = \begin{cases} \frac{3+\sin^2(2x)}{1+\cos^2 x}, x \leq 0 \\ 2x + \frac{\sin^2 x}{3+x}, x > 0 \end{cases}$	4	$y = \begin{cases} \frac{3x^2}{1+x^2}, x \leq 0 \\ \sqrt{1 + \frac{2x}{e^{0.5x} + x^2}}, x > 0 \end{cases}$

Продолжение таблицы 4.3			
1	2	3	4
5	$y = \begin{cases} \frac{3 + \sin^2 x}{1 + x^2}, x \leq 0 \\ 2x^2 \cos^2 x, x > 0 \end{cases}$	6	$y = \begin{cases} \sqrt{1 + 2x^2 + \sin^2 x}, x \leq 0 \\ \frac{2 + x}{\sqrt[3]{2 + e^{-0.5x}}}, x > 0 \end{cases}$
7	$y = \begin{cases} \frac{\sqrt{1 + x^2}}{1 + x}, x \leq 0 \\ \frac{1}{\sqrt[3]{1 + e^{-0.2x}} + 1}, x > 0 \end{cases}$	8	$y = \begin{cases} \sqrt{1 + x }, x \leq 0 \\ \frac{1 + 3x}{\sqrt[3]{1 + x + 2}}, x > 0 \end{cases}$
9	$y = \begin{cases} \frac{\sqrt{1 + x }}{2 + x }, x \leq 0 \\ \frac{1 + x}{2 + \cos^3 x}, x > 0 \end{cases}$	10	$y = \begin{cases} \sqrt[3]{1 + x^2}, x \leq 0 \\ \sin^2 x + \frac{1 + x}{1 + e^x}, x > 0 \end{cases}$
11	$y = \begin{cases} \frac{1 + x }{\sqrt[3]{1 + x + x^2}}, x \leq -1 \\ \frac{1 + \cos^4 x}{3 + x}, x > -1 \end{cases}$	12	$y = \begin{cases} 2 \ln(1 + x^2), x \leq -1 \\ (1 + \cos^2 x)^{\frac{3}{5}}, x > -1 \end{cases}$
13	$y = \begin{cases} \frac{1 + x}{\sqrt[3]{1 + x^2}}, x \leq 0 \\ -x + 2e^{-2x}, x > 0 \end{cases}$	14	$y = \begin{cases} 3x + \sqrt{1 + x^2}, x \leq 0 \\ 2 \cos x e^{-2x}, x > 0 \end{cases}$
15	$y = \begin{cases} \frac{\sqrt{1 + x^2}}{1 + x}, x \leq 0 \\ \frac{1}{\sqrt[3]{1 + e^{-0.2x}} + 1}, x > 0 \end{cases}$	16	$y = \begin{cases} \sqrt{1 + 2x^2 + \sin^2 x}, x \leq 0 \\ \frac{2 + x}{\sqrt[3]{2 + e^{-0.5x}}}, x > 0 \end{cases}$

4.3 Содержание отчета

1. Цель занятия.
2. Листинг программы для вывода графиков функций.

4.4 Контрольные вопросы

1. С помощью какой команды осуществляется построение графиков декартовой системы координат?
2. Для чего служит команда mesh?
3. Как осуществляется задание надписей?
4. Для чего используется команда grid?
5. Как осуществляется разбивка окна на меньшие окна?
6. Для чего используется команда hold?

Лабораторная работа № 5

РЕШЕНИЕ ТИПОВЫХ ЗАДАЧ АЛГЕБРЫ И АНАЛИЗА

Цель занятия: ознакомиться с возможностями системы MATLAB в решении типовых задач алгебры и анализа, изучение встроенного пакета символьных вычислений и операций Symbolic Math Toolbox.

5.1 Основные теоретические сведения

В системе MATLAB для решения систем линейных уравнений предусмотрены знаки операций / и \. Чтобы решить систему линейных уравнений вида

$$A \cdot y = B, \quad (1)$$

где A – заданная квадратная матрица размером $N \times N$,

B – заданный вектор–столбец длины N , достаточно применить операцию \ и вычислить выражение $A \setminus B$.

Операция \ называется левым делением матриц и, будучи примененная к матрицам A и B в виде $A \setminus B$, примерно эквивалентна вычислению выражения $\text{inv}(A) \cdot B$. Здесь под $\text{inv}(A)$ понимается вычисление матрицы, обратной к матрице A . Операцию / называют правым делением матрицы. Выражение A/B примерно соответствует вычислению выражения $B \cdot \text{inv}(A)$. Значит, эта операция позволяет решать системы линейных уравнений вида $y \cdot A = B$.

Решение уравнения $F(x)=0$, или нахождение нулей функции, осуществляется с помощью функции **fzero(name, x_0)**. В качестве первого аргумента ей передается имя функции, задающей исходное уравнение, вторым аргументом служит начальное приближение к корню. Определим, в качестве примера, нули функции $\cos(x)$ на отрезке от 0 до π . В качестве начального приближения примем $x_0=1$.

```
>> x=fzero('cos', 1)
x = 1.5708
```

Если требуется найти корень функции, отличной от стандартной (встроенной в систему MATLAB) и тем самым не имеющей в рамках системы MATLAB фиксированного имени, то нужно приписать некоторое имя выражению, вычисляющему функцию.

Пусть, например, требуется найти корни уравнения $\cos(x)=x$, что эквивалентно нахождению нулей функции, вычисляемой по формуле $y=\cos(x)-x$, не имеющей в рамках системы MATLAB фиксированного имени. В этом случае нужно создать Mat-функцию вида

```
function y=MyFunction1(x)
y=cos(x)-x
```

После этого можно воспользоваться функцией `fzero`:

```
>> x=fzero('MyFunction1',pi/2)
x = 0.7391
```

Если найдено абсолютно точное значение корня, то значение функции в этой точке равно нулю. Таким образом, величина функции в приближенно найденном нуле косвенно характеризует погрешность результата. Чтобы управлять погрешностью, нужно осуществлять вызов функции **fzero** с тремя аргументами **fzero(name, x_0 , tol)**, где **tol** задает требуемую величину погрешности (ошибки). Необходимо отметить, что функция **fzero** находит нули только вещественнозначных функций одной вещественной переменной. Однако часто бывает необходимо найти комплексные корни вещественнозначных функций, особенно в случае многочленов. Для этой цели в системе MATLAB существует специальная функция **roots(p)**, которой в качестве аргумента передается массив коэффициентов многочлена (p). Например, для многочлена $p = x^4 - 3x^3 + 3x^2 - 3x + 2$ нужно сначала сформировать массив его коэффициентов (расположив по порядку убывания степени x):

```
>> p = [ 1 -3 3 -3 2 ];
>> r = roots( p)
r = 2.0000
-0.0000 +1.0000i
-0.0000 -1.0000i
1.0000
```

В задаче о нахождении нулей функции сложным моментом является нахождение начального приближения к нулю функции, а также априорная оценка их количества. Поэтому важно параллельно с применением функций типа **roots** или **fzero** визуализировать поведение искомых функций на том или ином отрезке значений аргумента.

В системе MATLAB имеются специальные функции для поиска минимума заданных функций. При этом возможен поиск минимума как для функции одной вещественной переменной, так и для функций многих переменных. Для функций одной переменной их минимумы разыскивает функция $x_{\min} = \mathbf{fmin}(\text{name}, x_0, x_1)$. Здесь **name** представляет имя функции, у которой находятся минимумы, а x_0 и x_1 задают отрезок поиска. Для поиска минимума функции нескольких переменных применяется функция **fmins**: $x_{\min} = \mathbf{fmins}(\text{name}, x_0)$. Здесь **name** является именем функции нескольких переменных, для которой ищется минимум, а x_0 – это вектор ее аргументов, с которого начинается поиск. Для иллюстративного примера создадим простую функцию двух переменных имеющую минимумом точку (0,0).

```
function y = MyFunc2(x)
y = x(1)^2+ x(2)^2;
```

После этого можно вызвать функцию **fmins**, которая приближенно находит вектор $x_{\min}$ координат точки минимума:

```
>> xmin = fmins( 'MyFunc2', [1,1] );
>> xmin(1)
ans =-2.1023e-005
>> xmin(2)
ans =2.5484e-005
```

Для вычисления интегралов методом трапеций в системе MATLAB предусмотрена функция **trapz**: $\text{Integ} = \text{trapz}(x, y)$. Точность вычисления интеграла зависит от величины шага интегрирования: чем меньше этот шаг, тем больше точность.

Вычислим простой интеграл $\int_0^{\pi} \cos(x) dx$ методом трапеций с шагом интегрирования $\pi/10$.

```
>> dx = pi/10;
>> x = 0:dx:pi;
>> y=cos(x);
>> I1 = trapz(x,y);
I1 = 5.5511e-017
```

Обычно для достижения высокой точности требуется выполнять интегрирование с очень малыми шагами, а контроль достигнутой точности осуществлять путем сравнения последовательных результатов. При одном и том же шаге интегрирования методы более высоких порядков точности достигают более точных результатов. В системе MATLAB методы интегрирования более высоких порядков точности реализуются функциями: **quad** (метод Симпсона) и **quad8** (метод Ньюто-на-Котеса 8-го порядка точности).

Двойные интегралы в системе MATLAB вычисляются с помощью специальной функции **dblquad**.

Вычислим интеграл вида $\int_0^1 \int_0^2 (x \sin(y) + y \sin(x)) dx dy$.

Оформим подынтегральную *mat*-функцию и вызовем функцию **dblquad**:

```
function z = integ(x, y)
z = x.*sin(y) + y.*sin(x);
>> J = dblquad( 'integ', 0, 1, 1, 2 );
J = 1.1678
```

Возможности встроенного пакета символьных вычислений и операции Symbolic Math Toolbox достаточно обширны, рассмотрим лишь некоторые его возможности. Символьный объект создается при помощи функции **syms**. Команда создает три символьные переменные x , a и b

```
» syms x a b
```

Конструирование символьных функций от переменных класса *sym object* производится с использованием обычных арифметических операций и обозначений для встроенных математических функций. Запись формулы для выра-

жения в одну строку не всегда удобна, более естественный вид выражения выводит в командное окно функция **pretty**:

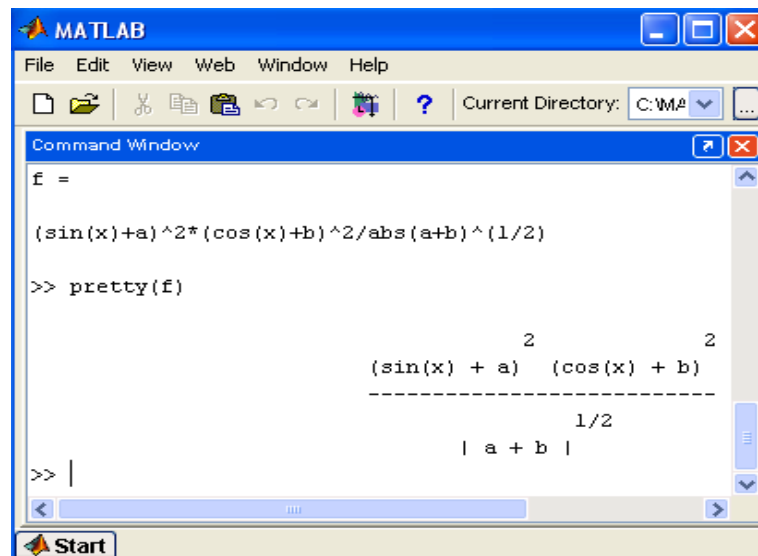


Рисунок 5.1– Демонстрация работы функции **pretty**

Упростим выражение $\frac{x^2 - y^2}{x - y}$, используя функцию **simplify**.

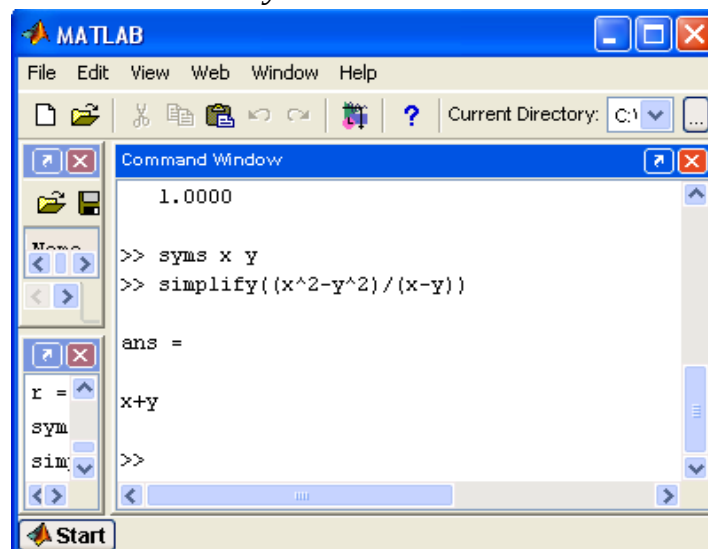


Рисунок 5.2– Демонстрация работы функции **simplify**

Символьную функцию можно создать без предварительного объявления переменных при помощи **sym**, входным аргументом которой является строка с выражением, заключенная в апострофы:

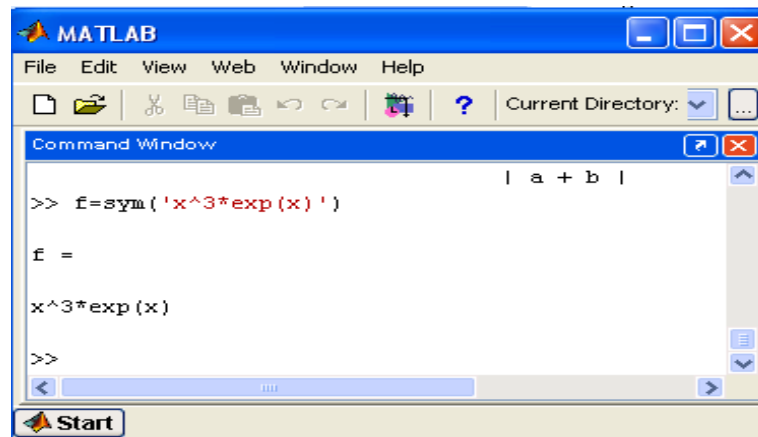


Рисунок 5.3 – Демонстрация создания символьной переменной без предварительного объявления

Упрощение тригонометрических, логарифмических, экспоненциальных функций и полиномов осуществляется функцией **expand**, формат обращения к которой имеет следующий вид: `rez=expand(S)`, где S – символьное выражение, которое нужно упростить, `rez` – результат упрощения.

Например:

```
>> syms x y;
>> rezl=expand(sin(x+y))
rezl = sin(x)*cos(y)+cos(x)*sin(y)
```

С помощью функции **factor** можно раскладывать многочлены на простые множители, а целые числа – в произведение простых чисел

```
>> factor(sym('x^5 - 1'))
ans =
(x-1)*(x^4+x^3+x^2+x+1)
```

Функция **subs** осуществляет подстановку новых выражений для указанных символьных переменных:

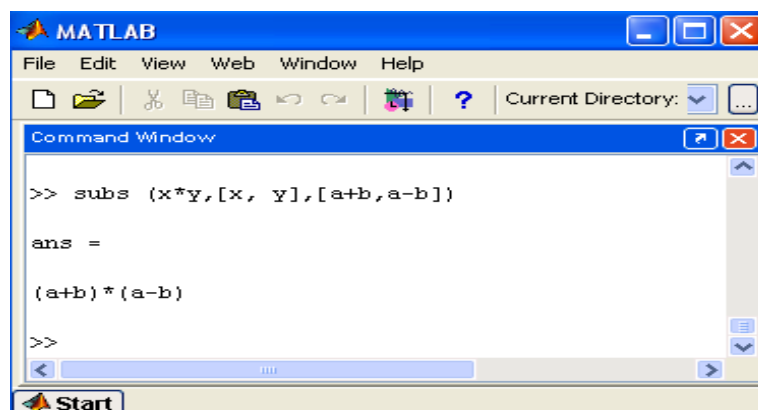


Рисунок 5.4– Демонстрация работы функции **subs**

Symbolic Math Toolbox позволяет работать как с неопределенными интегралами, так и с определенными. Неопределенные интегралы от символьных функций вычисляются при помощи функции **int**, в качестве входных аргумен-

тов указываются символьная функция и переменная, по которой происходит интегрирование (см. рис.5.5).

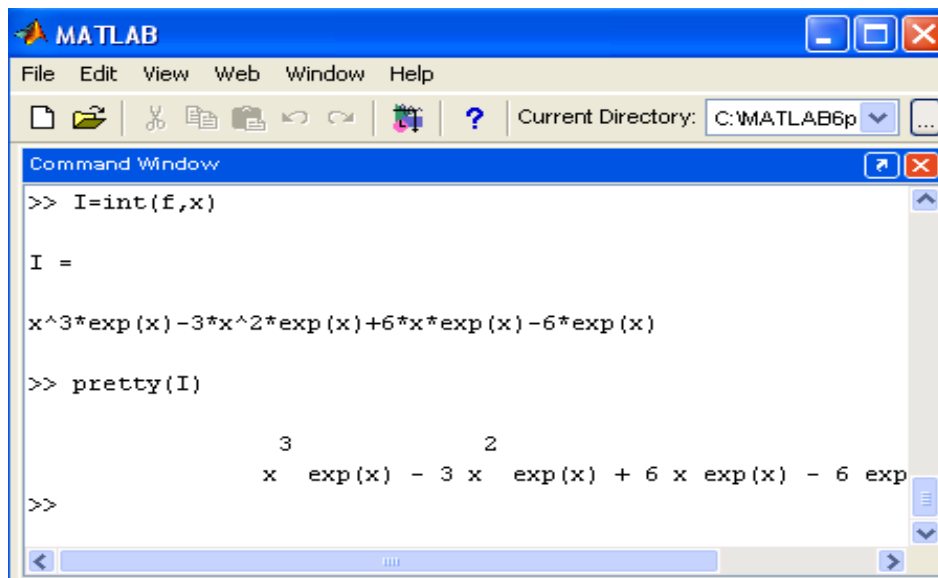


Рисунок 5.5– Демонстрация работы функции **int**

Для нахождения определенного интеграла в символьном виде следует задать нижний и верхний пределы интегрирования, соответственно, в третьем и четвертом аргументах **int**:

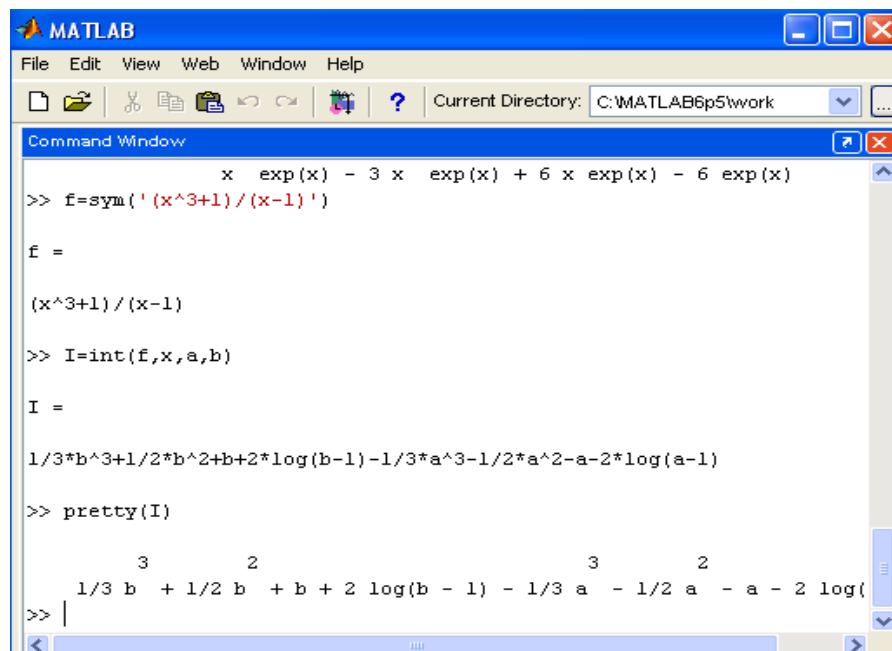


Рисунок 5.6– Демонстрация работы функции **int** для нахождения определенного интеграла

Перечислим еще несколько функций, часто используемых при символьных вычислениях:

`inv` – вычисляет обратную матрицу;

`limit` –вычисляет пределы;

taylor – осуществляет разложение функций в ряд Тейлора;
 solve – решает алгебраическое уравнение и систему алгебраических уравнений.

Для работы с символьными данными предусмотрена оболочка **funtool**. Она представляет собой интерактивный графический калькулятор, позволяющий быстро построить графики двух функций $f(x)$ и $g(x)$. Интерфейс данного приложения представлен на рисунке 5.7. При запуске выводятся три автономных окна: два графически и управляющее. Управляющее окно содержит два поля для ввода функций, поле ввода пределом изменения переменной x , поле ввода масштабирующего коэффициента.

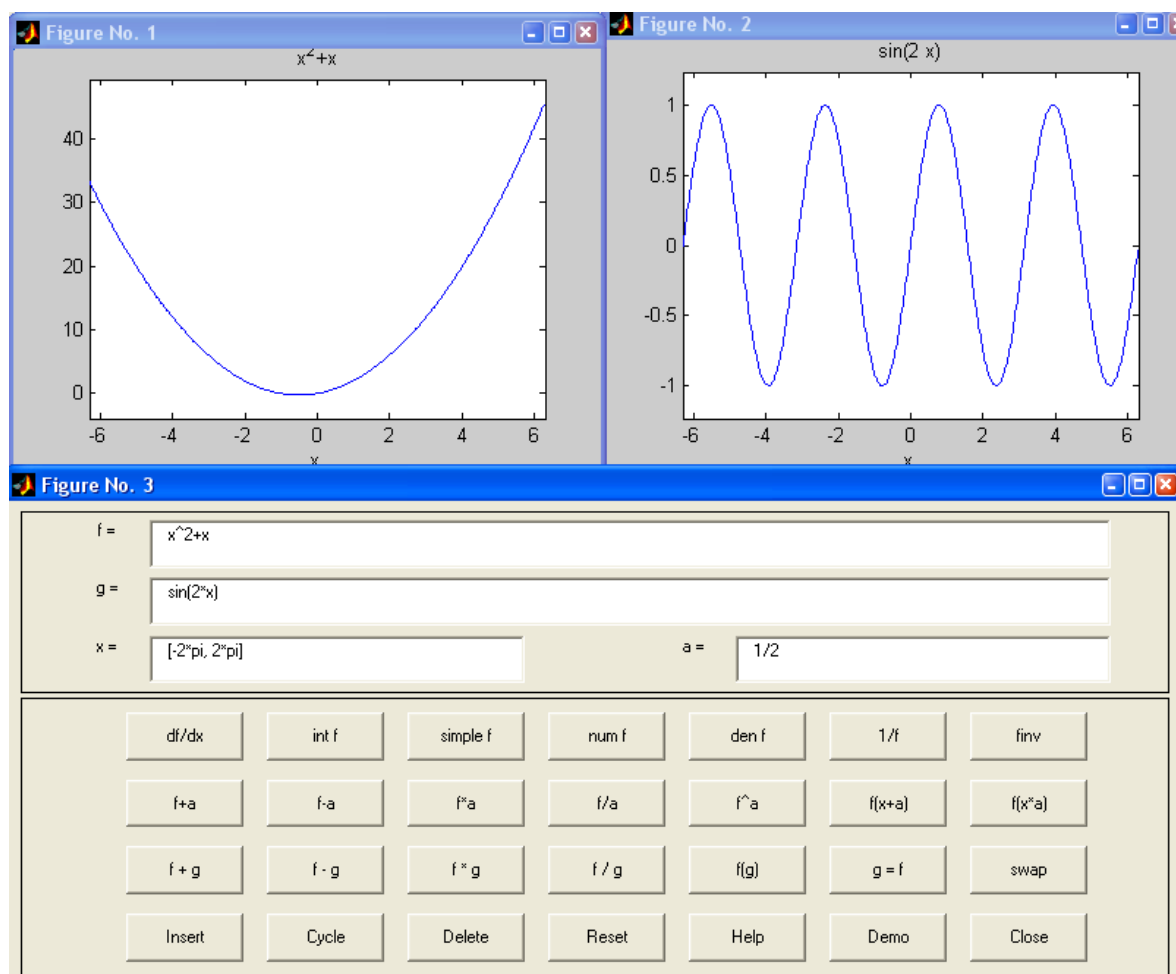


Рисунок 5.7 – Интерфейс приложения **funtool**

5.2 Порядок выполнения

1. Составить и отладить программы для нахождения корней уравнения $f_1(x) = 0$ и $f_2(x) = 0$ и вывести графики функции на основании задания из таблицы 5.1.
2. Найти определенный интеграл для подынтегральной функции, заданной в таблице 5.2.

Таблица 5.1– Варианты заданий для нахождения корней уравнения

№ вари- анта	$f_1(x)$ – полином 3-ей степе- ни с коэффициентами a				$f_2(x)$
1	2	3	4	5	6
1	0	-1	4	-1	$0.2e^x - 20$
2	0	2	-2	-15	$40 \cdot \cos(x) $
3	0	1	4	-1	$10 \cdot \log(x + 5.5)$
4	0	9	-8	-70	$100 \cdot \sin(x) $
5	0	-4	4	50	$70 \cdot \cos(x)$
6	1	-5	4	40	$60e^{ 0.1x } - 100$
7	2	-3	2	30	$20 \cdot \sin(2x)$
8	3	-6	1	50	$e^{ x } \sin(2x)$
9	4	-9	1	70	$e^{ x } \cos(3x)$
10	5	-7	5	60	$-60 \cdot \cos(x) $
11	-1	-4	9	60	$15 \cdot \log(x + 1.5)$
12	-2	-6	-7	55	$-50 \cdot \lg(x + 1.5)$
13	-3	-9	-8	75	$-100 \cdot \cos(x) $
14	-4	7	8	-75	$20 \cdot \sin(x/2)$
15	-5	1	4	-1	$40 \cdot \cos(x/2)$

Таблица 5.2 – Варианты функций для нахождения значения интеграла

№ вари- анта	Функция	Интервал интегрирова- ния	
		начало ин- тервала	конец интервала
1	2	3	4
1	$f(x) = x^2 - e^x$	-2	2
2	$f(x) = x - \cos(x)$	-2	2
3	$f(x) = \sin(x) - 2\cos(x)$	-2	2
4	$f(x) = \cos(x) + (1 + x^2)^{-1}$	-2	2
5	$f(x) = (x - 2)^2 - \ln(x)$	-0.5	4.5
6	$f(x) = 2x - \operatorname{tg}(x)$	-1.4	1.4
7	$f(x) = x^2 - 20\cos(x + 1)$	-5	5
8	$f(x) = 20\ln(x^2 + 1) - 0.1x^3$	-5	15
9	$f(x) = x^2 - e^x$	-4	2
10	$f(x) = 2x - \cos(x)$	-2	2
11	$f(x) = 3\sin(x) - \cos(x)$	-2	2

Продолжение таблицы 5.2			
1	2	3	4
12	$f(x) = x^2 - 20\cos(x + 1)$	-2	5
13	$f(x) = 20\ln(x^2 + 1) - 0.1x^3$	-1	1
14	$f(x) = x - \cos(x)$	-2	2
15	$f(x) = (x - 2)^2 - \ln(x)$	-2	2

3. Найти определенный интеграл для подынтегральной функции, заданной в таблице 5.2 с использованием пакета символьных вычислений.

5.3 Содержание отчета

1. Цель занятия.
2. Листинг программы и результаты выполнения.

5.4 Контрольные вопросы

1. Для чего служит функция fmin?
2. Для чего служит функция fzero?
4. Перечислите основные внешние расширения системы MATLAB для поиска экстремумов функций.
5. Какая функция служит для вычисления определенного интеграла?
6. Назовите функции работы с символьными переменными.

Лабораторная работа № 6

АППРОКСИМАЦИЯ И ИНТЕРПОЛЯЦИЯ ДАННЫХ. МЕТОДЫ РЕШЕНИЯ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Цель занятия: изучение основных методов аппроксимации и интерполяции данных в системе MATLAB и способов решения ОДУ.

6.1 Основные теоретические сведения

Под аппроксимацией обычно подразумевается описание некоторой, порой не заданной явно, зависимости или совокупности представляющих ее данных с помощью другой, обычно более простой или более единообразной зависимости. Часто данные находятся в виде отдельных узловых точек, координаты которых задаются таблицей данных. Результат аппроксимации может не проходить через узловые точки. Для обработки данных MATLAB использует различные функции интерполяции и аппроксимации данных.

Одна из наиболее известных аппроксимаций — полиномиальная. В системе MATLAB определены функции аппроксимации данных полиномами по методу наименьших квадратов — полиномиальной регрессии. Это выполняет функция, приведенная ниже:

– **polyfit**(x, y, n) — возвращает вектор коэффициентов полинома $p(x)$ степени n , который с наименьшей среднеквадратичной погрешностью аппроксимирует функцию $y(x)$.

Под интерполяцией обычно подразумевают вычисление значений функции $f(x)$ в промежутках между узловыми точками. Линейная, квадратичная и полиномиальная интерполяция реализуются при полиномиальной аппроксимации. В ряде случаев очень удобна сплайновая интерполяция и аппроксимация таблично заданных функций. При ней промежуточные точки ищутся по отрезкам полиномов третьей степени — это *кубическая сплайновая интерполяция*.

При этом обычно такие полиномы вычисляются так, чтобы не только их значения совпадали с координатами узловых точек, но также, чтобы в узловых точках были непрерывны производные первого и второго порядков. Такое поведение характерно для гибкой линейки, закрепленной в узловых точках, откуда и происходит название **spline** (сплайн) для этого вида интерполяции (аппроксимации).

Для одномерной табличной интерполяции используется функция **interp1**:

– $y_i = \mathbf{interp1}(x, y, x_i)$ — возвращает вектор y_i , содержащий элементы, соответствующие элементам x_i и полученные интерполяцией векторов x и y . Вектор x определяет точки, в которых задано значение y .

– $y_i = \mathbf{interp1}(x, y, x_i, \text{method})$ — позволяет с помощью параметра **method** задать метод интерполяции:

– 'nearest' — ступенчатая интерполяция;

– 'linear' — линейная интерполяция (принята по умолчанию);

- 'spline' — кубическая сплайн-интерполяция;
- 'cubic' или 'pchip' — интерполяция многочленами Эрмита.

Сплайн – интерполяция используется для представления данных отрезками полиномов невысокой степени – чаще всего третьей. При этом кубическая интерполяция обеспечивает непрерывность первой и второй производных результата интерполяции в узловых точках. Реализуется сплайн-интерполяция следующей функцией:

– $y_i = \text{spline}(x, y, x_i)$ — использует векторы x и y , содержащие аргументы функции и ее значения, и вектор x_i , задающий новые точки.

Решение большинства задач интерполяции и аппроксимации функций и табличных данных обычно сопровождается их визуализацией. Она, как правило, заключается в построении узловых точек функции (или табличных данных) и в построении функции аппроксимации или интерполяции (рис.6.1).

В MATLAB 6.5 совмещение функций аппроксимации с графической визуализацией доведено до логического конца — предусмотрена аппроксимация рядом методов точек функции, график которой построен. И все это выполняется прямо в окне редактора графики **Property Editor**. Для этого в позиции **Tools** графического окна имеются две новые команды:

Basic Fitting – основные виды аппроксимации (регрессии);

Data Statistics – статистические параметры данных.

Команда **Basic Fitting** открывает окно, дающее доступ к ряду видов аппроксимации и регрессии: сплайновой, эрмитовой и полиномиальной со степенями от 1 (линейная аппроксимация) до 10. В том числе со степенью 2 (квадратичная аппроксимация) и 3 (кубическая аппроксимация) (рис.6.2).

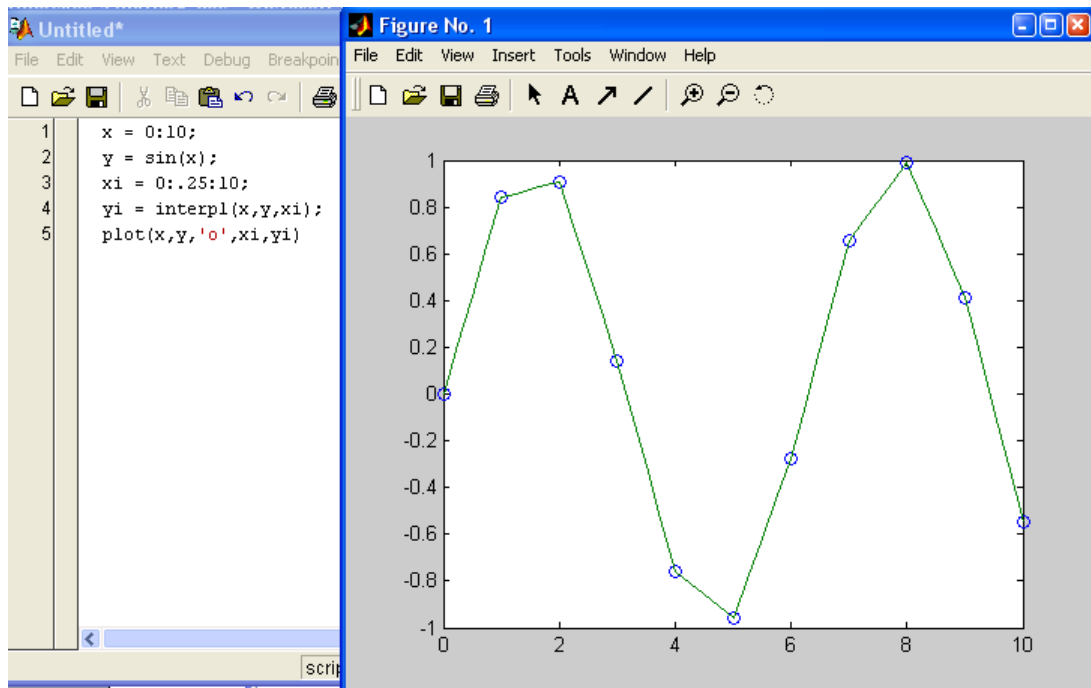


Рисунок 6.1 – Пример визуализации процесса интерполяции

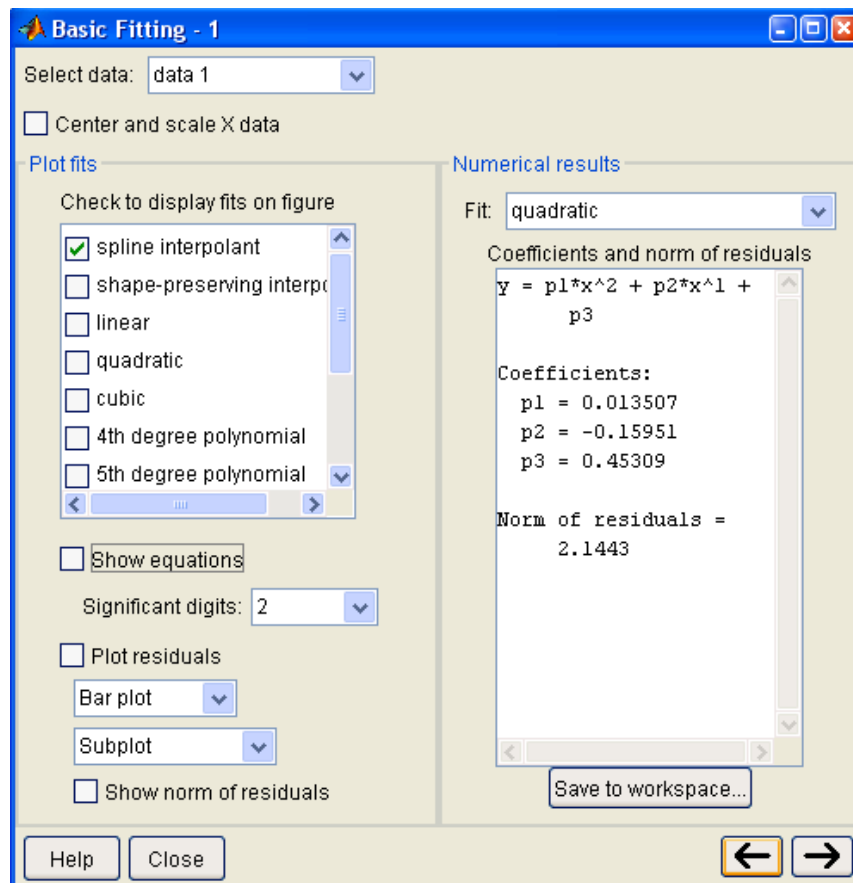


Рисунок 6.2 – Окно доступа к видам аппроксимации и регрессии

Команда **Data Statistics** открывает окно с результатами простейшей статистической обработки данных (рис.6.3).

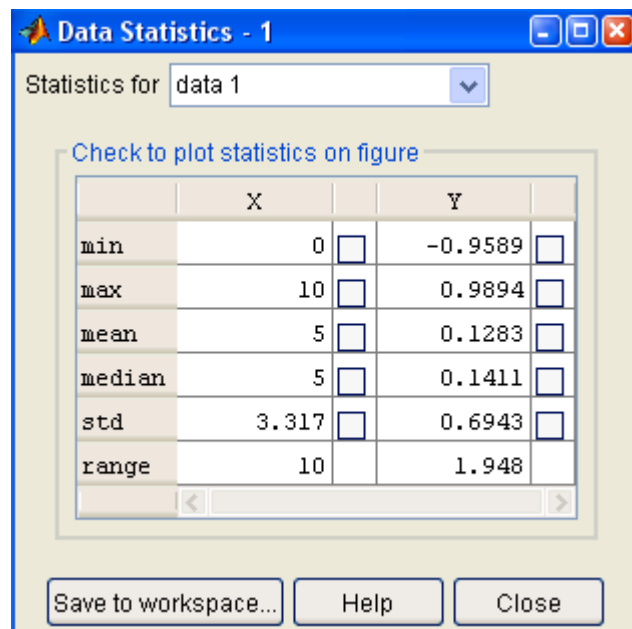


Рисунок 6.3 – Окно результатов статистической обработки

Анализ поведения многих систем и устройств в динамике базируются на решении систем обыкновенных дифференциальных уравнений (ОДУ). Их, как

правило, представляют в виде системы из дифференциальных уравнений первого порядка в форме Коши:

$$\begin{aligned}\frac{dy}{dt} &= y'; \\ y' &= f(x, y), \quad y(t_0, t_{end}) = b.\end{aligned}\tag{2}$$

где t_0, t_{end} — начальные и конечные точки интервалов.

Параметр t не обязательно означает время, хотя чаще всего решение дифференциальных уравнений ищется во временной области. Вектор b задает начальные и конечные условия.

Для решения систем ОДУ в MATLAB реализованы различные методы. Их реализации названы решателями ОДУ. Решатели реализуют следующие методы решения систем дифференциальных уравнений, причем для решения жестких систем уравнений рекомендуется использовать, только специальные решатели **ode45**, **ode23**:

- **ode45** — одношаговые явные методы Рунге-Кутты 4-го и 5-го порядка. Это классический метод, рекомендуемый для начальной пробы решения.

- **ode23** — одношаговые явные методы Рунге-Кутты 2-го и 4-го порядка.

В описанных далее функциях для решения систем дифференциальных уравнений приняты следующие обозначения и правила:

- **options** — аргумент, создаваемый функцией **odeset** (еще одна функция позволяет вывести параметры, установленные по умолчанию);

- **tspan** — вектор, определяющий интервал интегрирования $[t_0 \ t_{final}]$. Для получения решений в конкретные моменты времени $t_0, t_1, \dots, t_{final}$ (расположенные в порядке уменьшения или увеличения) нужно использовать **tspan** = $[t_0 \ t_1 \dots t_{final}]$;

- **y0** — вектор начальных условий;

- $p_1, p_2, \dots$ — произвольные параметры, передаваемые в функцию **F**;

- **T, Y** — матрица решений **Y**, где каждая строка соответствует времени, возвращенном в векторе-столбце **T**.

Перейдем к описанию функций для решения систем дифференциальных уравнений:

- **[T, Y] = solver(@F, tspan, y0)** — где вместо **solver** подставляем имя конкретного решателя — интегрирует систему дифференциальных уравнений вида $y' = F(t, y)$ на интервале **tspan** с начальными условиями **y0**, **@F** — дескриптор ODE-функции. Каждая строка в массиве решений **Y** соответствует значению времени, возвращаемому в векторе-столбце **T**;

- **[T, Y] = solver(@F, tspan, y0, options)** — дает решение, подобное описанному выше, но с параметрами, определяемыми значениями аргумента **options**, созданного функцией **odeset**.

Технология решения дифференциальных уравнений в системе MATLAB такова:

- 1) Создание m-файла. Независимо от вида системы он имеет вид:

```
function dy = solverDE(t, y)
dy = zeros(n, 1);
dy(1) = f1 (t, y(1), y(2), ..., y(n));
dy(2) = f2 (t, y(1), y(2), ..., y(n));
.....
dy(n) = fn (t, y(1), y(2), ..., y(n));
```

2) Получение решения и сопровождающий его график:

```
>> [T, Y] = solver('solverDE', [t0 tfinal], [y10 y20 ... yn0]);
>> plot(T, Y)
```

Пусть, к примеру, требуется решить дифференциальное уравнение:

$$y''' - 2y'' - y' + 2y = 0 \quad (3)$$

с единичными начальными условиями.

Данное дифференциальное уравнение второго порядка приведем к системе дифференциальных уравнений первого порядка:

$$\begin{cases} \frac{dy_1}{dt} = y_2; \\ \frac{dy_2}{dt} = y_3; \\ \frac{dy_3}{dt} = 2y_3 + y_2 - 2y_1, \end{cases} \quad (4)$$

с начальными условиями $y_1(0)=1, y_2(0)=1, y_3(0)=1$.

Вектор dy/dt правых частей системы уравнений, вычисляем с помощью собственной функции ex21 (рисунок 6.4):

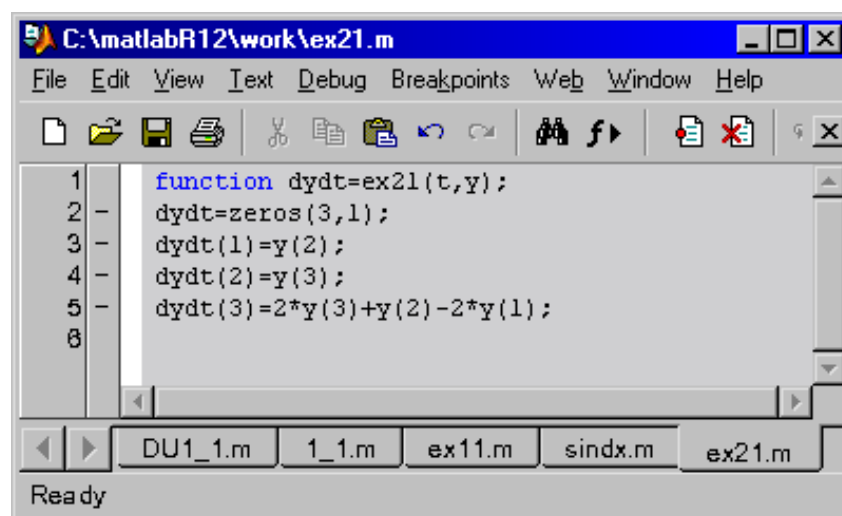


Рисунок 6.4 – Пример создания функции для решения системы ОДУ

Теперь можно вызывать функцию **ode45**, находящую решение нашей системы дифференциальных уравнений с начальными условиями [1,1,1] на отрезке [0,20] (рисунок 6.5)

```
>> y0=[1 1 1];  
>> tspan=[0 20];  
>> [T,Y]=ode45('ex21',tspan,y0);  
>> plot(T,Y)
```

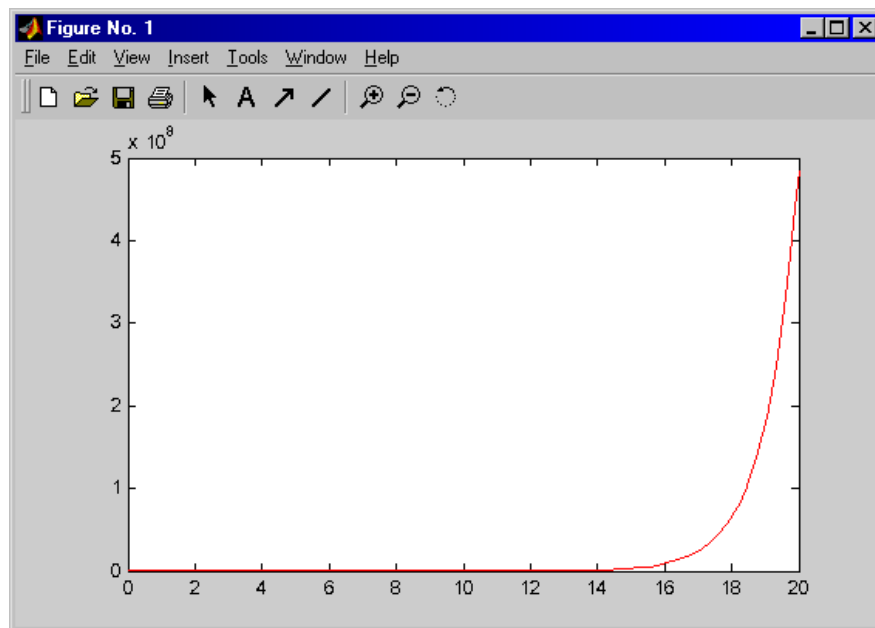


Рисунок 6.5 – Результат работы программы

Для решения дифференциальных уравнений в MATLAB зарезервирована функция **dsolve**, которая имеет следующие форматы обращения и возвращает аналитическое решение системы дифференциальных уравнений с начальными условиями:

– $y = \text{dsolve}('Dy(x)')$, где $Dy(x)$ – уравнение, y – возвращаемые функцией **dsolve** решения.

– $y = \text{dsolve}('Dy(x)', 'HY')$, где $Dy(x)$ – уравнение, HY – начальные условия.

Первая производная функции обозначается Dy , вторая производная – $D2y$ и так далее. Функция **dsolve** предназначена также для решения системы дифференциальных уравнений. В этом случае она имеет следующий формат обращения:

– $[f,g] = \text{dsolve}('Df(x), Dg(x)', 'HY')$, где $Df(x), Dg(x)$ – система уравнений, HY – начальные условия.

Решить дифференциальное уравнение (3) и использованием функции **dsolve** (рисунок 6.6).

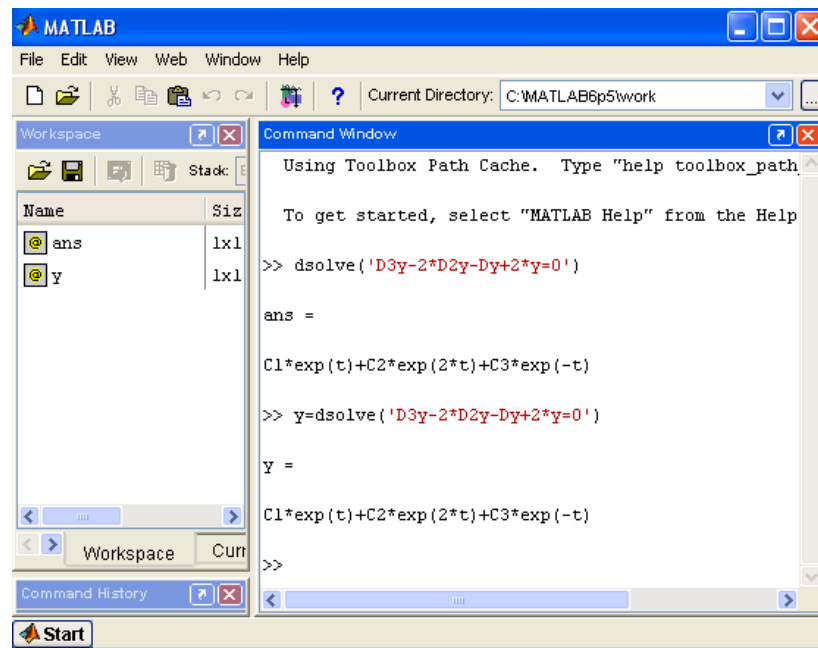


Рисунок 6.6 – Пример использования команды **dsolve**

6.2 Порядок выполнения

1. С помощью интерполяции найти значение таблично заданной функции в указанной точке таблица 6.1.
2. Выполнить аппроксимацию таблично заданной функции таблица 6.1.

Таблица 6.1– Варианты заданий

№ варианта	Табличная функция для интерполяции и аппроксимации								
1	2								
1	X	0	1	2	3	4	5	6	7
	Y	1	2.5	3	5	5.5	7	8	9
2	X	0	1	2	3	4	5	6	8
	Y	-1	-0.5	1	5	6.5	9	11.5	15
3	X	0	1	2	3	4	5	7	9
	Y	0.5	1	2.5	5	8.5	9.5	15	17.5
4	X	0	2	4	6	8	10	12	14
	Y	1	4	9	13	15	21	23	29
5	X	0	3	6	9	12	15	18	21
	Y	-2	8	16	25	35	45	52	62
6	X	0	2	4	6	10	16	20	25
	Y	-22	-14	-2	6	15	25	35	50
7	X	0	1	2	3	4	5	6	7
	Y	2	4	5	6	6.5	7	7.5	8
8	X	0	1	2	3	4	5	6	7
	Y	1	2.5	7	14.5	25	38.5	55	74.5

Продолжение таблицы 6.1									
1	2								
9	X	0	1	2	3	4	5	6	7
	Y	-1	0	3	8	15	24	35	48
10	X	0	1	2	3	4	5	6	7
	Y	-2	0	0.82	1.46	2	2.47	2.9	3.29
11	X	0	1	2	3	4	5	6	7
	Y	1	2.5	7	14.5	25	38.5	55	74.5
12	X	0	1	2	3	4	5	7	9
	Y	0.5	1	2.5	5	8.5	9.5	15	17.5
13	X	0	1	2	3	4	5	6	7
	Y	-1	0	3	8	15	24	35	48
14	X	0	2	4	6	10	16	20	25
	Y	-22	-14	-2	6	15	25	35	50
15	X	0	1	2	3	4	5	6	7
	Y	1	2.5	3	5	5.5	7	8	9

3. На отрезке $[a, b]$ найти решение дифференциального уравнения в виде $\frac{d^2 y}{dx^2} = f(x, y, y')$ с начальными условиями $y(0)$, $y'(0)$. Варианты заданий представлены в таблице 6.2. Построить график функции.

Таблица 6.2– Варианты заданий

№ варианта	$f(x, y, y')$	Начальные условия			
		a	b	$y(0)$	$y'(0)$
1	2	3	4	5	6
1	$f = e^x y + \cos(x)$	1	5	0	0
2	$f = e^{-x} + y \sin(x)$	2	3	1	0
3	$f = y \cos(x) + tg(x)$	0	1	0	0
4	$f = x^3 y + \cos(x)$	0	1	0	1
5	$f = e^x y / (1 - x) + x$	2	4	0	0
6	$f = x^2 y + 1 / (1 + x)$	1	3	0	0
7	$f = y \cos(x) + \cos^2(x)$	1	2	0	0
8	$f = (2 + x)y + \arctg(x)$	0	3	0	1
9	$f = (5 - x)y + x$	2	4	0	1
10	$f = e^{-xy} + 2e^{-x}$	0	1.5	1	0
11	$f = e^{-xy} / x + x$	-3	-2	1	1
12	$f = e^x y / (1 - x) + x$	2	4	0	0
13	$f = (10 - x)y + x$	1	5	1	1
14	$f = x^3 y + \cos(x)$	0	1	0	1

Продолжение таблицы 6.2					
1	2	3	4	5	6
15	$f = y \cos(x) + tg(x)$	0	1	0	0

4. Решить систему ОДУ, представленную в таблице 6.3, при заданных начальных условиях с помощью функции **dsolve**.

Таблица 6.3 Варианты заданий

№ варианта	Система ОДУ	Начальные условия			
		$u(0)$	$u'(0)$	$v(0)$	$v'(0)$
1	2	3	4	5	6
1	$\begin{cases} u'' = 2v + u, \\ v'' = 4v - 2u. \end{cases}$	1.5	1.5	1	1
2	$\begin{cases} u'' = -v + 3u, \\ v'' = v - 2u. \end{cases}$	-1	1	-1.5	3
3	$\begin{cases} u'' = 2v + u, \\ v'' = 4v - 2u. \end{cases}$	1.5	1.5	1	1
4	$\begin{cases} u'' = 5v, \\ v'' = v + 2u + t. \end{cases}$	1	1.5	0	2
5	$\begin{cases} u'' = v + u + t, \\ v'' = v + 2u - t. \end{cases}$	0.5	1.5	-1	2
6	$\begin{cases} u'' = 2v + u + t, \\ v'' = 4v. \end{cases}$	0.5	2	1	2
7	$\begin{cases} u'' = -v + t, \\ v'' = 5v - 7u. \end{cases}$	5	5	-1	1
8	$\begin{cases} u'' = v - 5u, \\ v'' = 2v + u + t. \end{cases}$	1.5	1	3	1
9	$\begin{cases} u'' = 1/2 + u, \\ v'' = 4 - u + t. \end{cases}$	2	0	-1	1
10	$\begin{cases} u'' = -v + t, \\ v'' = v + 3u. \end{cases}$	-1	2	-1.5	0
11	$\begin{cases} u'' = v - u - t, \\ v'' = 2v + u. \end{cases}$	1.5	1.5	-1	-1
12	$\begin{cases} u'' = 5v + t, \\ v'' = 3v + u. \end{cases}$	-1	1.5	0	2

Продолжение таблицы 6.3					
1	2	3	4	5	6
13	$\begin{cases} u'' = v + u, \\ v'' = v + u - t. \end{cases}$	0.5	1	-1	2
14	$\begin{cases} u'' = 2v - u, \\ v'' = 4v + t. \end{cases}$	0	-2	0	2
15	$\begin{cases} u'' = v - 2t, \\ v'' = v + 3u. \end{cases}$	3	3	-1	1

6.3 Содержание отчета

1. Цель занятия.
2. Листинг программы.
3. Результаты выполнения.

6.4 Контрольные вопросы

1. Для чего служит функция interp1?
2. Какой функцией реализуется сплайн-интерполяция в MATLAB?
3. Какие существуют методы решения систем дифференциальных уравнений?

Лабораторная работа № 7

ОСНОВНЫЕ ВОЗМОЖНОСТИ ПАКЕТА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ SIMULINK 4.0

Цель занятия: ознакомиться с возможностями пакета математического моделирования Simulink.

7.1 Основные теоретические сведения

Программа Simulink является приложением к пакету MATLAB. При моделировании с использованием Simulink реализуется принцип визуального программирования, в соответствии с которым, пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты. При работе с Simulink пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков.

Для запуска программы необходимо предварительно запустить пакет MATLAB. Основное окно пакета MATLAB показано на рисунке 7.1. Там же показана подсказка появляющаяся в окне при наведении указателя мыши на ярлык Simulink в панели инструментов.

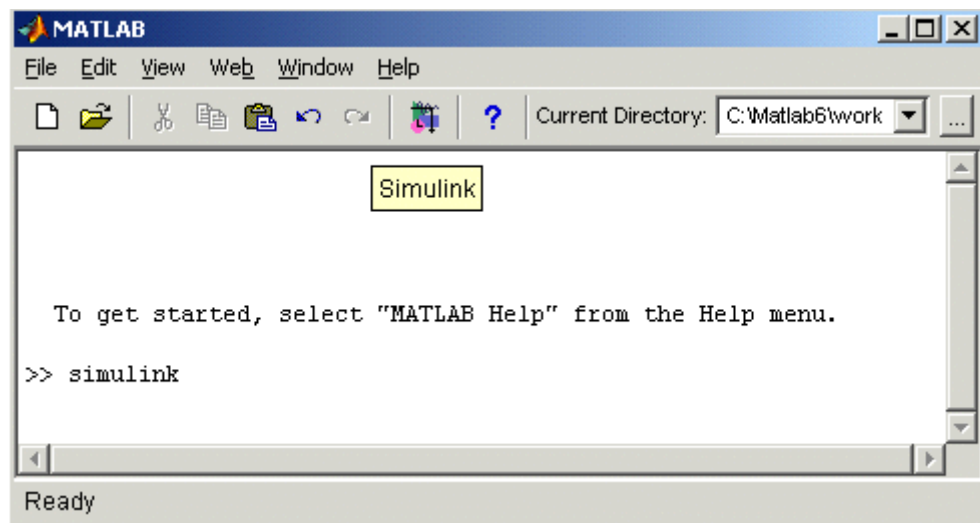


Рисунок 7.1 – Основное окно программы MATLAB

После открытия основного окна программы MATLAB нужно запустить программу Simulink. Это можно сделать одним из трех способов:

1. Нажать кнопку (Simulink) на панели инструментов командного окна MATLAB.
2. В командной строке главного окна MATLAB напечатать Simulink и нажать клавишу Enter на клавиатуре.
3. Выполнить команду **Open...** в меню **File** и открыть файл модели (mdl - файл).

Последний вариант удобно использовать для запуска уже готовой и отлаженной модели, когда требуется лишь провести расчеты и не нужно добавлять новые блоки в модель.

Использование первого и второго способов приводит к открытию окна обозревателя разделов библиотеки Simulink (рис. 7.2).

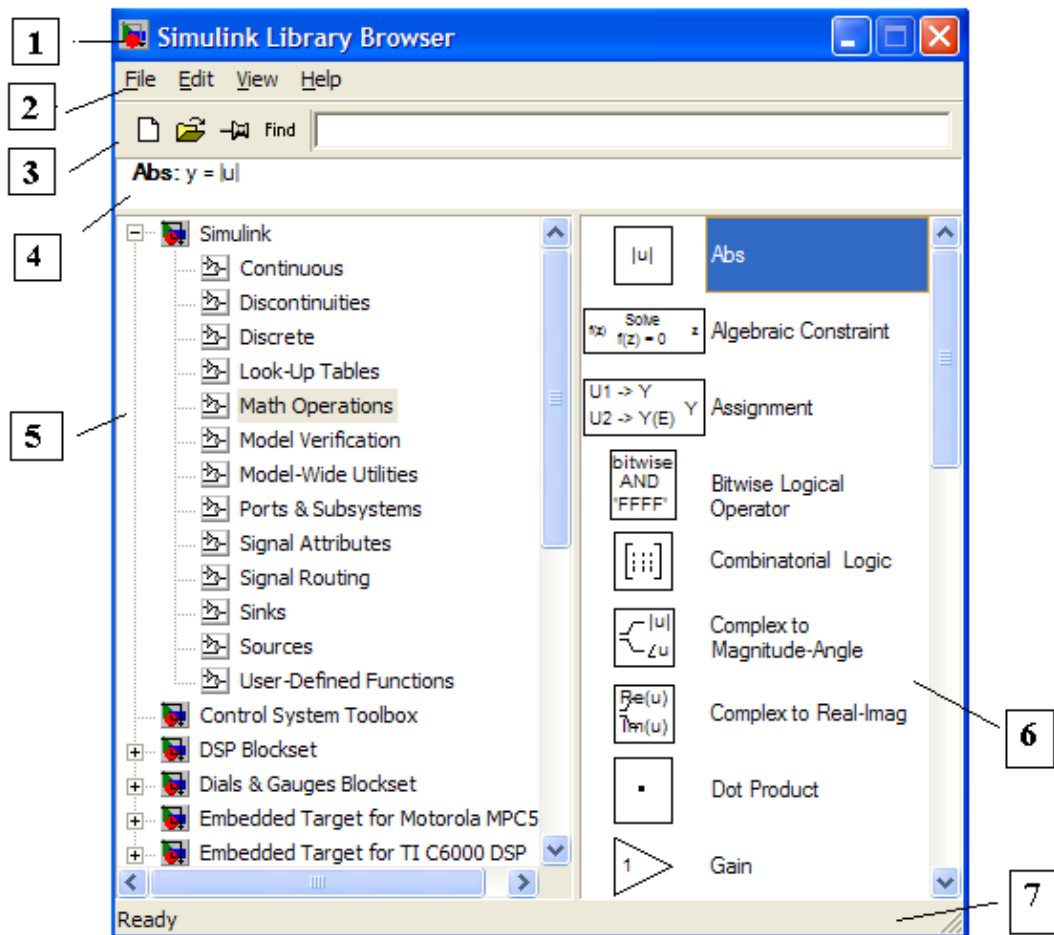


Рисунок 7.2 – Окно обозревателя разделов библиотеки Simulink

Окно обозревателя библиотеки блоков содержит следующие элементы:

- 1) Заголовок, с названием окна – Simulink Library Browser.
 - 2) Меню, с командами **File**, **Edit**, **View**, **Help**.
 - 3) Панель инструментов, с ярлыками наиболее часто используемых команд.
 - 4) Окно комментария для вывода поясняющего сообщения о выбранном блоке.
 - 5) Список разделов библиотеки, реализованный в виде дерева.
 - 6) Окно содержимого раздела библиотеки (список вложенных разделов библиотеки или блоков)
 - 7) Строка состояния, содержащая подсказку по выполняемому действию.
- На рисунке 7.2 выделена основная библиотека Simulink (в левой части окна) и показаны ее разделы (в правой части окна).

Библиотека Simulink содержит следующие основные разделы:

1. **Continuous** – линейные блоки.

2. **Discrete** – дискретные блоки.
3. **User-Defined Functions** – функции и таблицы.
4. **Math Operations** – блоки математических операций.
5. **Discontinuities** – нелинейные блоки.
6. **Signals Attribute, Signals Routing** – сигналы и системы.
7. **Sinks** - регистрирующие устройства.
8. **Sources** — источники сигналов и воздействий.
9. **Ports & Subsystems** – порты и блоки подсистем.

Список разделов библиотеки Simulink представлен в виде дерева, и правила работы с ним являются общими для списков такого вида:

- Пиктограмма свернутого узла дерева содержит символ "+", а пиктограмма развернутого содержит символ "-".
- Для того чтобы развернуть или свернуть узел дерева, достаточно щелкнуть на его пиктограмме левой клавишей мыши.

При выборе соответствующего раздела библиотеки в правой части окна отображается его содержимое (рис. 7.3).

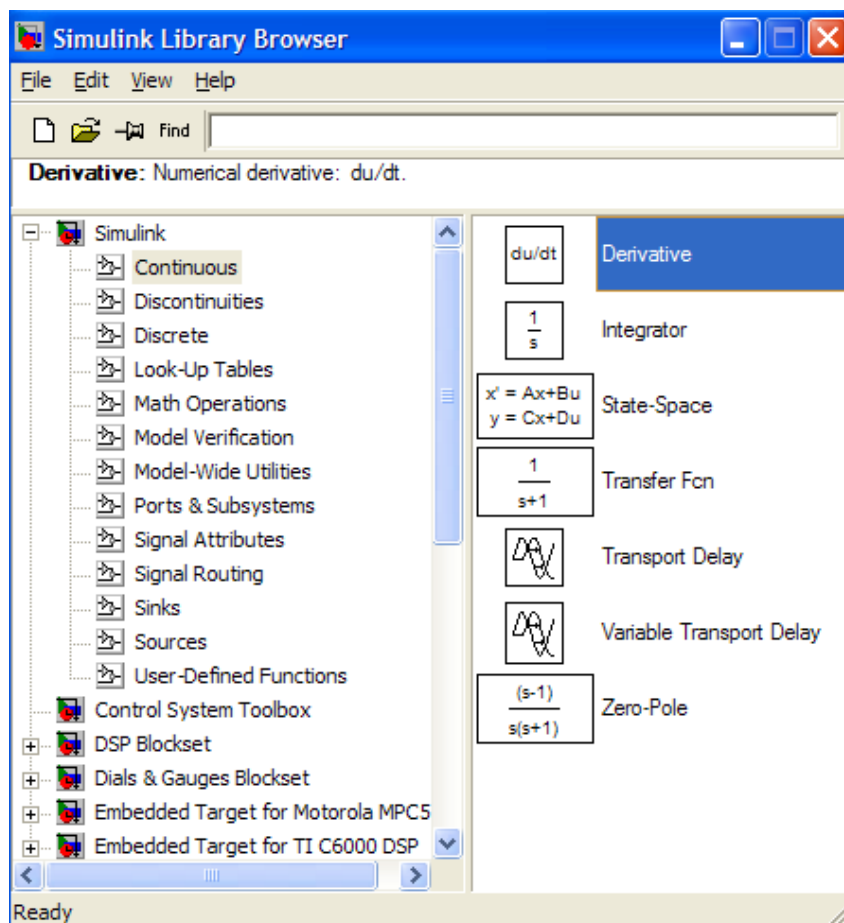


Рисунок 7.3 – Содержимое раздела **Continuous**

Для создания модели в среде Simulink необходимо последовательно выполнить ряд действий:

1. Создать новый файл модели с помощью команды **File/New/Model**, Вновь созданное окно модели показано на рисунке 7.4.

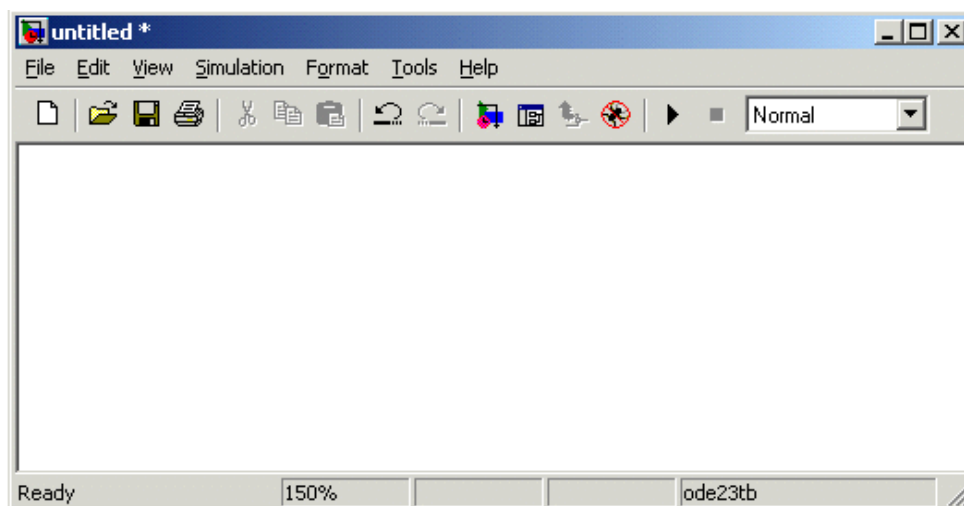


Рисунок 7.5 – Пустое окно модели

2. Расположить блоки в окне модели. Для этого необходимо открыть соответствующий раздел библиотеки (Например, **Sources** - Источники). Далее, указав курсором на требуемый блок и нажав на левую клавишу “мыши” - “перетащить” блок в созданное окно. Клавишу мыши нужно держать нажатой. На рисунке 7.6 показано окно модели, содержащее блоки.

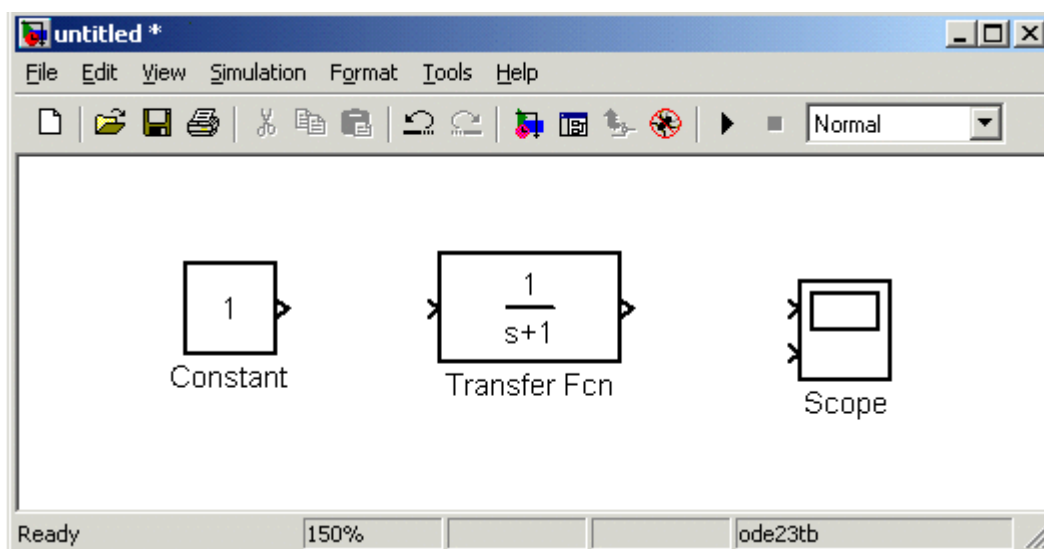


Рисунок 7.6 – Окно модели, содержащее блоки

Для изменения параметров блока необходимо дважды щелкнуть левой клавишей “мыши”, указав курсором на изображение блока. Откроется окно редактирования параметров данного блока. При задании численных параметров следует иметь в виду, что в качестве десятичного разделителя должна использоваться точка, а не запятая. После внесения изменений нужно закрыть окно кнопкой **ОК**.

После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы. Для соединения блоков необходимо указать курсором на “выход” блока, а затем, нажав и, не отпуская левую клавишу “мыши”, провести линию к входу другого блока. После чего отпустить клавишу. В случае правильного соединения изображение стрелки на входе

блока изменяет цвет. Для создания точки разветвления в соединительной линии нужно подвести курсор к предполагаемому узлу и, нажав правую клавишу “мыши”, протянуть линию. Схема модели, в которой выполнены соединения между блоками, показана на рисунке 7.7.

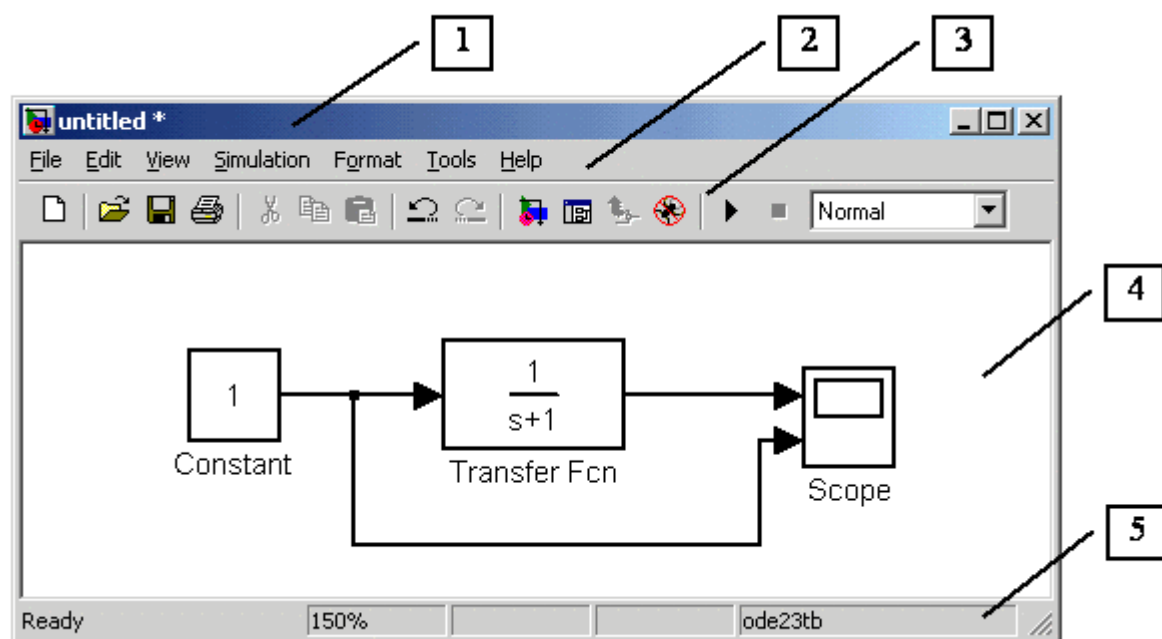


Рисунок 7.7 – Схема модели

Окно модели содержит следующие элементы (см.рис.7.7):

1. Заголовок, с названием окна. Вновь созданному окну присваивается имя Untitled с соответствующим номером.
2. Меню с командами **File**, **Edit**, **View** и т.д.
3. Панель инструментов.
4. Окно для создания схемы модели.
5. Строка состояния, содержащая информацию о текущем состоянии модели.

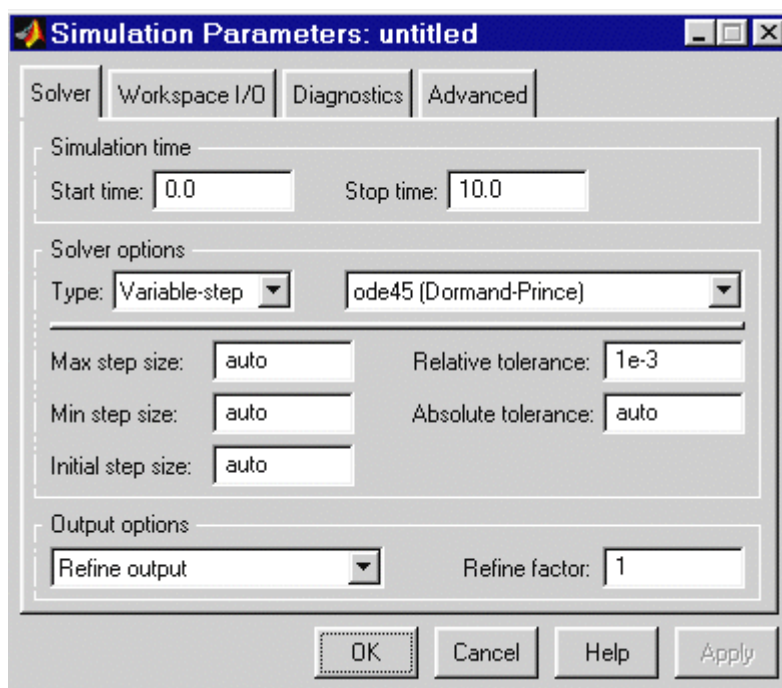
Меню окна содержит команды для редактирования модели, ее настройки и управления процессом расчета, работы файлами и т.п.:

- a) **File** (Файл) — Работа с файлами моделей.
- b) **Edit** (Редактирование) — Изменение модели и поиск блоков.
- c) **View** (Вид) — Управление показом элементов интерфейса.
- d) **Simulation** (Моделирование) — Задание настроек для моделирования и управление процессом расчета.
- e) **Format** (Форматирование) — Изменение внешнего вида блоков и модели в целом.
- f) **Tools** (Инструментальные средства) — Применение специальных средств для работы с моделью (отладчик, линейный анализ и т.п.)
- g) **Help** (Справка) — Вывод окон справочной системы.

Для повышения наглядности модели удобно использовать текстовые надписи. Для создания надписи нужно указать мышью место надписи и дважды щелкнуть левой клавишей мыши. После этого появится прямоугольная

рамка с курсором ввода. Аналогичным образом можно изменить и подписи к блоками моделей. Следует иметь в виду, что рассматриваемая версия программы (Simulink 4) не адаптирована к использованию кириллических шрифтов, и применение их может иметь самые разные последствия: отображение надписей в нечитаемом виде, обрезание надписей, сообщения об ошибках, а также невозможность открыть модель после ее сохранения. Поэтому, применение надписей на русском языке для текущей версии Simulink крайне не желательно.

Перед выполнением расчетов необходимо предварительно задать параметры расчета. Задание параметров расчета выполняется в панели управления меню **Simulation/Parameters**. Вид панели управления приведен на рисунке 7.8.



Рисунке 7.8– Панель управления

Окно настройки параметров расчета имеет 4 вкладки:

- **Solver** (Расчет) — Установка параметров расчета модели.
- **Workspace I/O** (Ввод/вывод данных в рабочую область) — Установка параметров обмена данными с рабочей областью MATLAB.
- **Diagnostics** (Диагностика) — Выбор параметров диагностического режима.
- **Advanced** (Дополнительно) — Установка дополнительных параметров.

Для визуализации процесса моделирования используются виртуальные регистраторы (блоки получателей информации **Sinks**).

Каждый регистратор имеет свое окно настройки, которое появляется при активизации его пиктограммы в окне компонентов или в окне модели (рисунок 7.9).

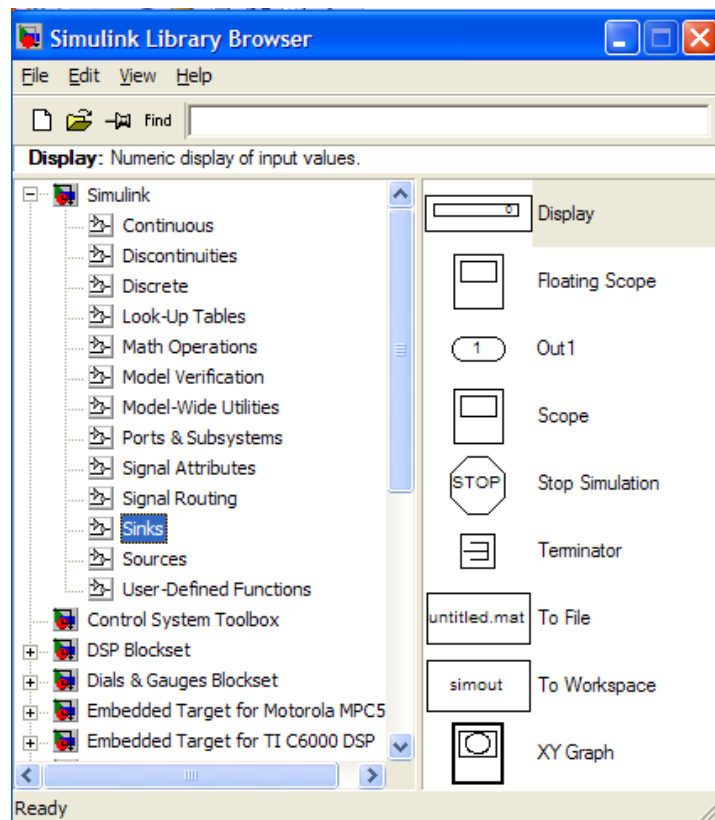


Рисунок 7.9 – Содержимое раздела **Sinks**

Виртуальный осциллограф (**Scope**), позволяет представить результаты моделирования в виде временных диаграмм тех или иных процессов в форме, которая напоминает выполненные лучами разного цвета осциллограммы современного высокоточного осциллографа с оцифрованной масштабной сеткой.

Здесь особое внимание надо обратить на кнопки масштабирования, позволяющие (наряду с командами контекстного меню) менять размер осциллограммы. Весьма удобной является кнопка автоматического масштабирования (**Autoscale**) – обычно она позволяет установить такой масштаб, при котором изображение осциллограммы имеет максимально возможный размер по вертикали и отражает весь временной интервал моделирования (рисунок 7.10).

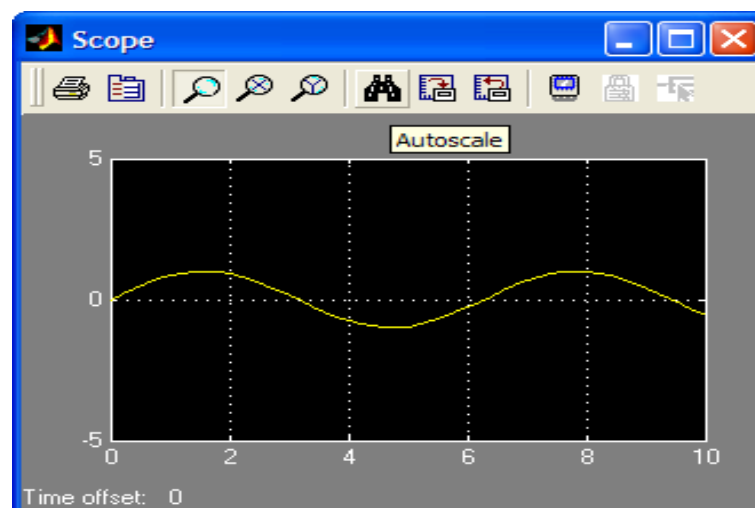


Рисунок 7.10 – Окно виртуального осциллографа

Виртуальный графопостроитель (**XY Graph**) имеет входы по осям X и Y , что позволяет строить графики функций в полярной системе координат, фигуры Лиссажу, фазовые портреты и т. д. (рисунок 7.11).

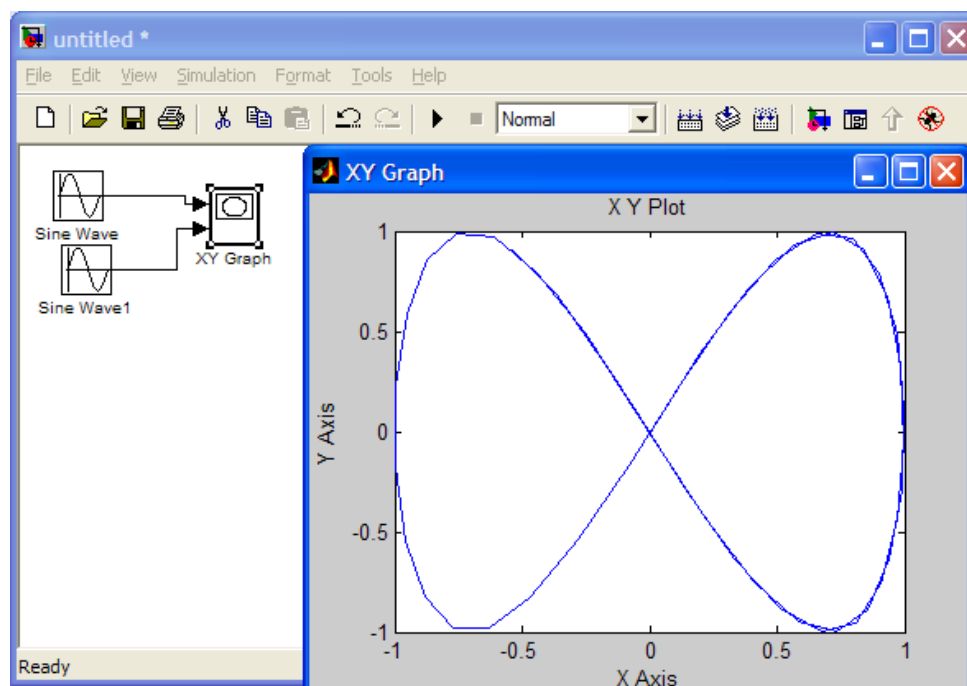


Рисунок 7.11– График фигуры Лиссажу

Для выполнения арифметических операций используется вкладка **Math Operations** (рисунок 7.12). К числу наиболее простых математических блоков относятся блоки арифметических операций: вычисления абсолютного значения числа **Abs**, скалярного произведения **Dot Product**, обычного произведения **Product**, а также суммы **Sum**.

Обратите внимание на то, что в окне настройки блока сложения / вычитания можно установить вид представления блока (круглый или квадратный) и число входов с выполняемыми по ним операциями. Число входов и операции задаются шаблоном **List of sign**. Например, шаблон $|++$ означает, что блок имеет два суммирующих входа, а $|+-+$ — что он имеет три входа, причем средний — вычитающий, а крайние — суммирующие (рисунок 7.13).

Блок **Product** (Умножение) предназначен для умножения и деления ряда входных сигналов. При этом операции задаются подобно тому, как это было описано для блока суммирования/вычитания с применением знаков умножения $*$ или деления $/$ в шаблоне.

Для контроля знака служит блок **Sign**. Он возвращает -1 при отрицательном входном аргументе, 0 — при нулевом входном аргументе и 1 — при положительном входном аргументе.

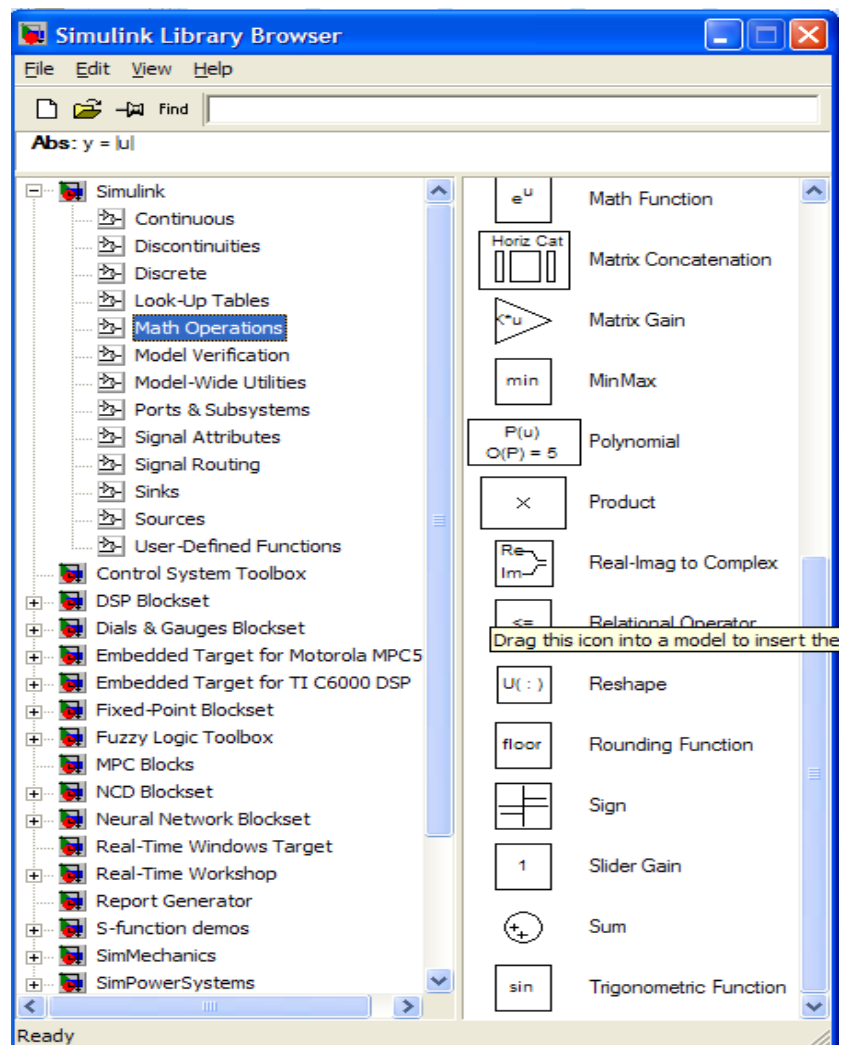


Рисунок 7.12 – Содержимое раздела **Math Operations**

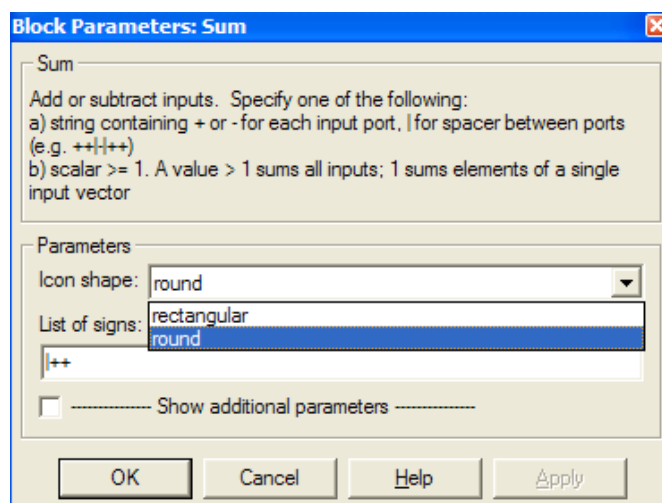


Рисунок 7.13 – Окно настройки блока сложения / вычитания

Блоки масштабирования **Gain** и **Slider Gain** служат для масштабирования данных (умножения их на заданный коэффициент — константу). В блоке **Gain** константа вводится в окне параметров (по умолчанию 1), а в блоке **Slider Gain** ее можно выбирать с помощью ползунка. Для масштабирования матричных данных служит блок **Matrix Gain**.

Блок **Math Function** служит для задания математических функций одной переменной u по правилам, принятым для языка программирования базовой системы MATLAB. В частности, это означает, что в теле функции могут встречаться встроенные функции системы. Окно параметров этого блока содержит описание правил задания функции и раздел параметров **Parameters**, в котором задаются выражение для функции и длина вектора выхода. Если она должна совпадать с длиной вектора входного сигнала, то вводится значение -1 (рисунок 7.14).

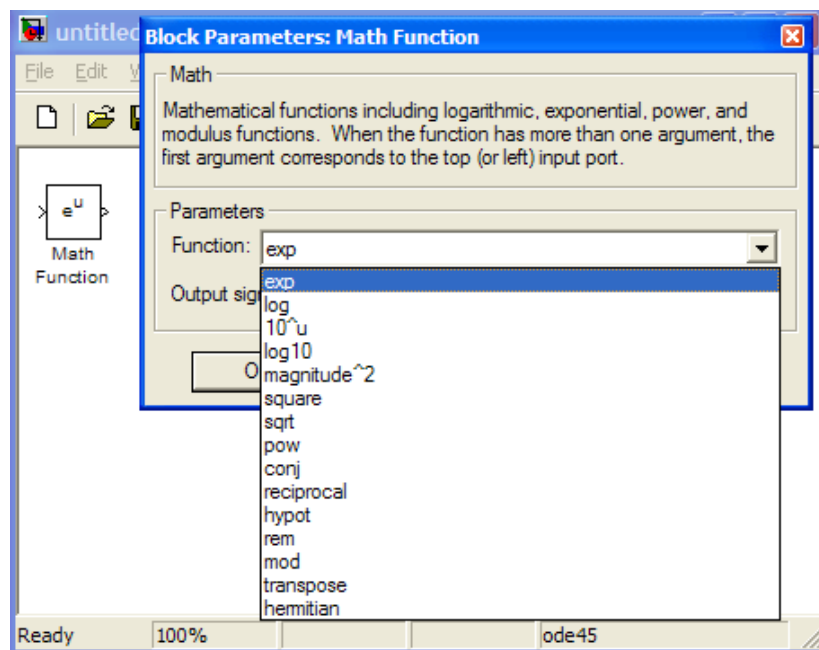


Рисунок 7.14 – Окно параметров блока **Math Function**

Дополнительная библиотека блоков Simulink находится в разделе **Simulink Extras**.

7.2 Порядок выполнения

1. Ознакомиться со структурой иерархической библиотеки Simulink.
2. Набрать модель системы, выданную преподавателем.
3. Произвести моделирование системы.
4. Сравнить результаты, полученные путем изменения параметров моделирования.

7.3 Содержание отчета

1. Схема модели.
2. Результаты моделирования.

7.4 Контрольные вопросы

1. Из каких библиотек состоит пакет Simulink?
2. Как собрать модель в пакете Simulink?

3. Как изменить параметры моделирования?
4. Какие существуют способы визуализации процесса моделирования?

Лабораторная работа №8

Знакомство с пакетом MATLAB 7.2.

1. Структура пакета и принципы работы

MATLAB – система многоцелевого назначения, которая вышла на рынок программных продуктов почти двадцать лет назад и с тех пор непрерывно совершенствовалась фирмой MathWorks. Но первоначально ее основу составляли алгоритмы решения систем линейных уравнений и задач на собственные значения, откуда и произошло ее название «матричная лаборатория» (MATrix LABoratory). Затем система была расширена за счет специальных приложений, таких как Simulink (для моделирования ИС), Wavelet (для применения вэйвлетов), Symbolic Math Toolbox (для проведения символьных вычислений) и пр.

Среда MATLAB включает интерпретатор команд на языке высокого уровня, графическую систему, пакеты расширений и реализована на языке C. Вся работа организуется через командное окно (Command Window), которое появляется при запуске программы matlab.exe. В процессе работы данные располагаются в памяти (Workspace), для изображения кривых, поверхностей и других графиков создаются графические окна. В командном окне в режиме диалога проводятся вычисления. Пользователь вводит команды или запускает на выполнение файлы с текстами на языке MATLAB. Интерпретатор обрабатывает введенное и выдает результаты: числовые и строковые данные, предупреждения и сообщения об ошибках. Строка ввода помечена знаком `>>`. В командном окне показываются вводимые с клавиатуры числа, переменные, а также результаты вычислений.

Инструментальная панель командного окна позволяет выполнять требуемые действия простым нажатием на соответствующую кнопку. Большинство кнопок имеют стандартный вид и выполняют стандартные, подобные другим программам действия. Следует обратить внимание на кнопку Path Browser, которая позволяет прокладывать пути к разным директориям и делать необходимую директорию текущей, а также на кнопку Workspace Browser, позволяющую просматривать и редактировать переменные в рабочей области. Принято работать с включенными помимо консоли окнами Command History и Workspace; включить их можно, расставив галочки в раскрытом меню Desktop, а упорядочить – проведя докировку (кривая стрелка около стандартного крестика закрытия окна). Если вы нарушили порядок следования окон, то воспользуйтесь командой Desktop/Desktop Layout/Default.

Все значения переменных, вычисленные в течение текущего сеанса работы, сохраняются в специально зарезервированной области памяти компьютера, называемой рабочим пространством системы MATLAB (Workspace). После окончания сеанса работы с системой MATLAB все ранее вычисленные переменные теряются. Чтобы сохранить в файле на диске компьютера содержимое рабочего пространства системы MATLAB, нужно выполнить команду меню File / Save Workspace As. По умолчанию расширение имени файла mat, поэтому такие файлы принято называть МАТ-файлами. Для загрузки в память компьютера ранее сохраненного на диске рабочего пространства нужно вы-

полнить команду меню: File / Load Workspace. Обе операции можно реализовать в режиме командной строки в формате, например, save <имя_файла> (без расширения .mat).

Команда help <имя_функции> позволяет получить на экране справку по конкретной функции. Например, команда help eig позволяет получить оперативную справку по функции **eig** - функции вычисления собственных значений матрицы. С некоторыми возможностями системы можно познакомиться с помощью команды demo. В окне MATLAB помимо собственно команд MATLAB можно использовать системные команды DOS. Удобным свойством системы является возможность использовать клавиши-стрелки ↑↓ для доступа к стеку с ранее введенными командами. Командой clc можно стереть содержимое командного окна, однако это не затронет содержимого рабочего пространства. Когда исчезает необходимость в хранении ряда переменных в текущем сеансе работы, их можно стереть из памяти компьютера командой clear или clear(имя1, имя2, ...). Первая команда удаляет из б памяти все переменные, а вторая – переменные с именами имя1 и имя2. Командой who (или более сильной whos) можно вывести список всех переменных, входящих в данный момент в рабочее пространство системы. Для просмотра значения любой переменной из текущего рабочего пространства системы достаточно набрать ее имя и нажать клавишу Enter. Для целей программирования важно знать, что часть строки, следующая за знаком %, является комментарием.

Легче всего протокол сессии получить с помощью команды diary. Вызов команды diary <имя_файла> приведет к тому, что все появившееся далее на экране (кроме графики) будет записано в файл <имя_файла>. Несколько последовательно набранных команд могут быть сохранены в М-файле (т.е. *.m), который с точки зрения операционных систем представляет аналог BAT-файла, а с точки зрения программирования – зародыш будущей программы. Самый простой путь для этого – в окне Command History выделить нужные команды, исполнить команду Create M-file контекстного меню; при этом откроется редактор М-файлов.

Пример:

Для получения информации о магическом квадрате наберем в консоли lookfor magic, получим, подождав некоторое время (для прерывания поиска использовать DOS-овскую комбинацию Ctrl+C):

```
>> lookfor magic
MAGIC Magic square.
TWEBMAGIC Example standalone test of webmagic function.
WEBMAGIC Magic squares into HTML table.
```

Затем получим справку о команде MAGIC:

```
>> help magic
```

MAGIC Magic square.

MAGIC(N) is an N-by-N matrix constructed from the integers 1 through N^2 with equal row, column, and diagonal sums.

Produces valid magic squares for all $N > 0$ except $N = 2$.

Reference page in Help browser

[doc magic](#)

Щелкнем по ссылке, открыв окно помощи. Построим уже реальный магический квадрат, набрав в консоли для $N=10$:

```
>> XMag=magic(10)
```

XMag =

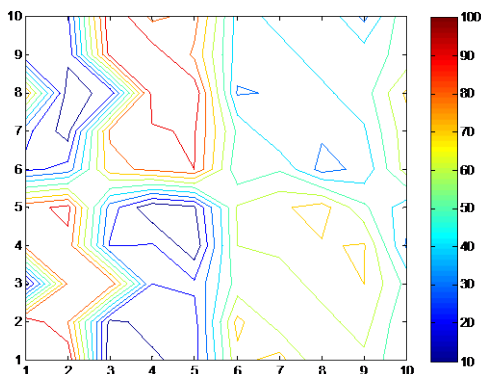
92	99	1	8	15	67	74	51	58	40
98	80	7	14	16	73	55	57	64	41
4	81	88	20	22	54	56	63	70	47
85	87	19	21	3	60	62	69	71	28
86	93	25	2	9	61	68	75	52	34
17	24	76	83	90	42	49	26	33	65
23	5	82	89	91	48	30	32	39	66
79	6	13	95	97	29	31	38	45	72
10	12	94	96	78	35	37	44	46	53
11	18	100	77	84	36	43	50	27	59

Подсчитаем, сколько занимает в памяти переменная XMag:

```
>> whos XMag
```

Name	Size	Bytes	Class
XMag	10x10	800	double array

Grand total is 100 elements using 800 bytes



Выберем закладку Workspace («рабочее пространство»), в контекстном меню исполним команду `countour`. При этом появится специальное окно, где MATLAB позволяет редактировать графические объекты – в частности, добавим расшифровку цветов (пиктограмма `Insert ColorBar`). Обратим внимание, что наши манипуляции не остались незамеченными средой, и в консоли появилась команда:

```
>> contour (XMag, 'DisplayName', 'XMag', 'ZDataSource', 'XMag'); figure(gcf)
```

Очистим рабочую область командой `clear`, затем консоль – `clc`. При этом история команд по-прежнему доступна; выделим шапку со списком сегодняшней даты и запишем в М-файл, вызвав контекстное (с выделенным) меню.

```
lookfor MAGIC
help magic
XMag=magic(10)
whos XMag
contour (XMag, 'DisplayName', 'XMag', 'ZDataSource', 'XMag'); figure(gcf)
clear
clc
```

Задание:

1. Запустить MATLAB и установите текущий каталог. Получить справку по команде `diary`. Начать запись дневника в файл.
2. В справочной системе, начав с консоли, получить информацию о гиперболическом котангенсе, а также переводу угла из радианной меры в градусную (и наоборот).
3. Вычислить $\sin(3.3a+b*\text{cth}(a+b))$ для $a=15^0$, $b=23^0$ (предварительно проведя расчет для нулевых a и b).
4. В справочной системе найти информацию о том, какую функцию следует вызывать для вычисления ближайшего к X целого. Предварительно следует показать преподавателю таблицу простейших функций вещественного переменного (перенеся ее в MS Word).

Примечание: необходимо иметь навык работы с HELP-системами (уходя от консоли, вызов, как обычно, по клавише F1 или командой Help– MatLab Help главного меню). Существует две стратегии поиска: по ключевым словам (индексу) или по дереву содержания (закладка Content окна помощи). Вторая стратегия предпочтительнее, но ее использование требует некоторого эвристического опыта. Разумеется, знание английского языка более чем необходимо.

5. Сформировать матрицу из случайных элементов размера 10×10 . Используя пользовательские средства Workspace, построить ее 3D-график, сохранить картинку средствами MATLAB.
6. Записать текущую сессию в mat-файл. Сохранить историю команд в m-файл.
7. Очистить экран и переменные. Заккрыть MATLAB.
8. Показать преподавателю все сохраненные файлы. Открыть MATLAB и с консоли загрузить последний mat-файл.
9. Завершить работу в MATLAB окончательно.

Лабораторная работа №9

Проведение вычислений без М-файлов

1. Элементарные матричные вычисления.

В MATLAB все данные рассматриваются как матрицы. Этот принцип *векторизации* имеет как положительные, так и отрицательные стороны. Тип результата определяется автоматически по виду выражения. Для имен обычных переменных принято выбирать строчные, а для имен матриц или даже векторов – прописные (большие). Вычислять можно в режиме калькулятора, но чаще используют присвоение (обычный знак «=», для проверки условий зарезервировано двойное равенство «==») какой-либо переменной. Любое уже определенное значение можно вызвать из рабочей области по имени переменной (либо в консоли, либо контекстным меню в окне Workspace). Помните, что можно провести несколько последовательных вычислений, подавив вывод в консоль промежуточных результатов – для этого после каждой команды нужно поставить «;».

Для чтения/записи данных из файлов (прежде всего ASCII-формата) MATLAB обладает внушительным арсеналом средств (см. раздел помощи MATLAB Programming – Using Import Functions with Text Data). В случае, когда импортируемый файл является перечислением чисел в регулярных колонках и строках (без различных шапок и легенд), достаточно применить команду load. Среди дружественных MATLAB внешних программ прежде всего следует отметить Excel (см. раздел помощи Excel Link – What Is Excel Link?), который со своей стороны имеет средства общения с MATLAB.

При ручном вводе, если компоненты вектора не умещаются на строке, используется троеточие в качестве разделителя. Иногда требуется организовать доступ к блоку внутри имеющейся матрицы, что реализуется через знак «:» – конструкция A(2:5,:) дает нам строки со 2-й по 5-ю матрицы A. Если нужно выполнить одинаковую операцию с каждым элементом матрицы, то нужно перед знаком поставить точку – V.*V, например, возвращает вектор квадратов, т.е. $(-1,5) \rightarrow (2,25)$. Для создания многомерных матриц использовать функцию cat.

Пример:

Введем скаляр, трехмерные вектор-строку и вектор-столбец, рассматриваемые как матрицы размера $1 \times n$ и $n \times 1$ ($n=3$). Сделаем это чуть различающимися способами (разделителем в массиве может либо пробел, либо запятая, либо оба знака вместе).

```
>> x=-1.25e-1
```

```
x =
```

```
-0.1250
```

```
>> u=[1,2.0 4]
```

```
u =
```

```
1 2 4
```

```
>> v=[-x; -2.0;7]
```

```
v =
```

```
0.1250  
-2.0000  
7.0000
```

Приготовим заранее в рабочем каталоге файл dm3.txt, состоящий из 3 строк по 3 числа, разделенных пробелом (ами) или знаком табуляции. Загрузим его данные в матрицу W.

```
>> Wfrom='dm3.txt';W=load(Wfrom);W
```

```
W =
```

```
1.2000 4.3000 5.0000  
-34.0000 2.1000 4.0000  
3.0000 3.0000 -6.0000
```

Заметим, мы ввели литерную переменную Wfrom, подавили вывод в консоль, и при этом исходный файл был таким:

```
1.2 4.3 5.0  
-3.4e+1 2.1 4  
3 3 -6
```

Попробуем сгенерировать новую матрицу 10×3 , где столбцы являются арифметическими прогрессиями. Вначале создадим матрицу 3×10 , а затем ее транспонируем.

```
>> W1=[-1.4:0.2:0.4; linspace(-2,5,10); -1.4:7.7]
```

```
W1 =
```

```
-1.4000 -1.2000 -1.0000 -0.8000 -0.6000 -0.4000 -0.2000 0  
0.2000 0.4000  
-2.0000 -1.2222 -0.4444 0.3333 1.1111 1.8889 2.6667 3.4444  
4.2222 5.0000  
-1.4000 -0.4000 0.6000 1.6000 2.6000 3.6000 4.6000 5.6000  
6.6000 7.6000
```

```
>> W1=W1'
```

```
W1 =
```

```
-1.4000 -2.0000 -1.4000  
-1.2000 -1.2222 -0.4000  
-1.0000 -0.4444  0.6000  
-0.8000  0.3333  1.6000  
-0.6000  1.1111  2.6000  
-0.4000  1.8889  3.6000  
-0.2000  2.6667  4.6000  
    0    3.4444  5.6000  
 0.2000  4.2222  6.6000  
 0.4000  5.0000  7.6000
```

Заметим, что второй столбец W1 получился в результате применения вызова функции от трех аргументов linspace, а первый – иллюстрирует типичное применения знака «:». Третий столбец иллюстрирует особенности последней операции. Интерпретатор MATLAB выдал бы ошибку, если бы в первой команде мы заменили 0.4 на 0.6 или в последней 7.7 на 9.9. Причина заключается в несоответствии размеров столбцов, ведь второй декларируется как имеющий 10 элементов.

Получим новую матрицу размера 3×3 таким способом: домножим транспонированного двойника W1 на W1 и переставим угловой элемент с центральными местами.

```
>> W2=W1'*W1; Wtemp=W2(1,1); W2(1,1)=W2(end-1,end-1); W2(end-1,end-1)=Wtemp; clear Wtemp; W2
```

```
W2 =
```

```
72.4074  5.3333  1.0000  
 5.3333  5.8000 110.6667  
 1.0000 110.6667 178.6000
```

Появившаяся выше переменная Wtemp была «мавром, сделавшим свое дело».

Сформируем новую матрицу на основе всех предыдущих – как линейную комбинацию W, W2 и случайной матрицы. Ниже в указанных форматах вызова: функция eig ищет наименьшее по величине собственное значение матрицы, diag(rand(...)) – формирует почти нулевую матрицу с ненулевой случайной наддиагональю, det и inv – вычисляют определитель матрицы и обратную ей. Очевидно также, что (u*v) – скаляр, (v*u) – матрица 3×3.

```
>> W3=x*inv(W)+W2/eigs(W2,1,'sm')-((u*v)+det(v*u))*diag(rand(1,2),1)
```

Iteration 1: a few Ritz values of the 3-by-3 matrix:

0

W3 =

```
-1.4996 -4.9030 -0.0201
-0.1275 -0.1219 -2.6727
-0.0304 -2.2877 -3.6801
```

Запишем полученный результат в файл EXCEL в определенный диапазон ячеек на первом листе. Флаг W3status равен 1, если все прошло успешно.

```
>> W3status=xlswrite('dm3_res.xls',W3,'B3:H10')
```

W3status =

1

Задание

1. Ввести две скалярные величины $x=-0.85$, $y=978.56 \cdot 10^{-2}$.
2. Заготовить XLS-файл с двумя матрицами A и B размера 3×5 и 5×3 .
3. Считать их в одноименные структуры MATLAB.

Примечание: Необходимо воспользоваться одним из подразделов помощи MATLAB Programming

4. Изменить центральные элементы обеих матриц. Для A – домножить его на x и вычесть 1, для B – домножить его на $\sin y$ и прибавить 1.

5. Осуществить поэлементное умножение матрицы A на транспонированную матрицу B.

6. Вычислить произведение AB и прибавить к нему матрицу C, где C – трехдиагональная: на главной диагонали единицы, наддиагональ – из x-ов, поддиагональ – из y-ков.

7. Результат вывести в виде текстового файла.

8. Вычислить след (trace) матрицы, обратной к последней вычисленной матрице.

9. Сохранить последовательность команд в файле matrix1.m

2. Элементарные функциональные вычисления и построение графиков.

Комплексные числа вводятся в виде $Re+Imi$ или $Re+Im*j$ (предпочтительнее вариант без умножения, поскольку может возникнуть путаница с перекрытием имен). Попробуйте набрать `>> j=5;[1+j,1+1j]` и почувствуйте разни-

цу! Однако в случае, когда коэффициентом перед мнимой единицей является не число, а переменная, между ними следует обязательно использовать знак умножения. Полезно знать три константы MATLAB: π – число π , inf – бесконечность, NaN – неопределенное значение. Они могут выступать аргументами функций: `>> [sin(inf),sin(1/inf),3*NaN]`. В системной переменной `ans` хранится результат вычислений, а в `eps` – номинальная точность вычислений, что удобно с точки зрения численных методов, например, решения уравнений способом дихотомии. Значения этих переменных можно изменять: по умолчанию `eps=2^(-52)`, но очень часто делают присвоение `eps=1e-6`; удобно проводить цепи вычислений (пример: `>> ans+1`). В MATLAB широкий набор самых разнообразных встроенных математических функций. Вы можете задавать и собственные функции (доступные, однако, только в текущем сеансе). Они называются **анонимные**, их значение вычисляется только одним оператором, а формат задания и вызова таков:

`<имя>=@(<arg1>,...) <выражение>      ...    <имя> (<arg1>,...)`

При этом инициализируется переменная-указатель на функцию, и как раз по указателю функция и вызывается (полезно посмотреть Workspace). При написании собственных функций весьма желательно предусмотреть матричные операнды, что сильно облегчает жизнь при построении графиков (так, для вычисления $f(x)=x^3$ конструкция `x^3` хуже правильного варианта `x.^3` с точкой поэлементной операции). Для реализации более сложных и серьезных функций необходимо обращаться к программированию М-файлов. Приведем лишь некоторые из часто используемых:

`log(x)` – натуральный логарифм

`mod(x,y)` – остаток от деления x на y (применима и для нецелых аргументов)

`angle(z)` – угол поворота комплексного числа z

`polyval(p,t)` – вычисление полинома от аргумента t с заданными вектором p коэффициентами.

`bessel(nu,z)` – вычисляет функцию Бесселя, причем если аргумент z матричный, то и результат также матричный. Константа nu нецелая. Пользователь может не только использовать различный синтаксис вызова (например, количество аргументов), но и определять вариант получения результата – в этом состоит важная и удобная особенность функций в MATLAB.

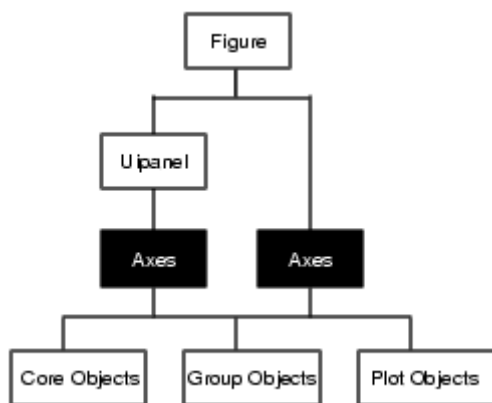
`quad1(fun,a,b)` – вычисление определенного интеграла Римана по адаптивному методу Гаусса-Лобатто на отрезке $[a,b]$ от функции $f(x)$, заданной указателем `fun`. Если функция анонимная, то вместо `fun` подставляется просто `<имя>`, но если функция задана в М-файле, то нужно провести разыменование и подставлять `<@имя>`.

Теперь рассмотрим функции вспомогательные для расчета, но весьма полезные:

`str2num` – переводит строку в число

`t1c <операции> t0c` – сколько времени (в сек) потребовалось для совершения операций

clock – возвращает текущую дату и время



Важное значение имеет графическое представление результатов, и MATLAB обладает обширным набором средств (см. раздел помощи в закладке Contents MATLAB – Mathematics - Plotting Mathematical Functions, более подробно изложено в MATLAB – Functions -- By Category –Graphics–Specialized Plotting). Каждый раз при получении графика создается графический объект, который автоматически открывается в окне графического редактора (можно открыть несколько объектов

– см. команда `h=figure(·)` и смежные; об иерархии графических объектов см. MATLAB – Handle Graphics Object Properties). Там вы можете манипулировать подписями осей, легендами, цветами линий, сохранять изображение в разных форматах. Существует сложная иерархия графических объектов, и каждому действию в окне графического редактора соответствует команда, модифицирующая заданный посредством указателя объект. Поэтому не следует забывать, что управлять параметрами изображения можно из консоли (разумеется, большинство инструкций приходится брать из системы помощи, например `set`) – этот способ вряд ли можно считать прогрессивным, но в некоторых случаях (поскольку доступные операции в меню окна графического редактора не исчерпывают всего многообразия) незаменим. Нужно помнить, что такие операции, как выбор цветовой гаммы, применяются к тому объекту, который объявлен текущим.

Не меньшее значение имеют пользовательские данные, представленные, как правило, матрицами большого размера (например, 100×300). Часть данных, таких как значения функций, может вычисляться «на лету», но надежнее вычислить все необходимое до построения графика. Всякий график предполагает наличие сетки, прямоугольной или треугольной (если ранее применялись численные методы, то в роли такой сетки может выступать сетка для исчисления конечных разностей, которая изначально и определяет детальность представления результатов; в общем случае обе сетки не совпадают, и тогда дополнительно следует применять интерполяцию – см. `griddata`). Самый простой путь для получения сеток для 3D-поверхностей – использование функции `meshgrid`.

Наиболее общеупотребительны (для целей физико-математических расчетов и/или проектирования РЭА) следующие виды графиков (рядом указано имя специальной функции):

А. Простой график вида $y=f(x)$ с маркерами и без– `plot(x,y,...)`, `fplot(...)`. Последняя команда удобна тем, что кривая строится по указателю функции, или даже по текстовому описанию `fplot('sin(x).*cos(2x)', [0 5*pi])`

В. Графики, построенные в преобразованных координатах (например,

двойных логарифмических, т.е. данные имеют вид $(x, f(x))$, а координаты – $(\ln x, \ln f(x))$. В том числе в полярных координатах – `cart2pol`, `loglog(x,y)`, `polar(fi,r)`.

С. Параметрическое семейство кривых $y=f(x,p)$, p – параметр, значение которого указывается в легенде для каждой кривой $y_p=f(x)$. Все кривые имеют разный цвет – `hold on`, `plot`, параметр `LineStyle` функции `plot`.

Д. Контурный график, изображающий линии уровня (изолинии) $z=\text{const}$ для поверхности $z=f(x,y)$. Промежутки, отделяющие изолинии, могут заполняться цветом, а могут и нет. Географическая карта, где отображаются океаны и горы, - классический пример контурного графика – `contourc`, `contourf`.

Е. 3D-график для поверхности $z=f(x,y)$ или траектории, заданной параметрически $\mathbf{r}(t)$ – `colormap(hsv(128))`, `mesh`, `surf`, `plot3`

Ф. Анимированные графики. Обычно для траекторий частиц, получили большое распространение в последнее время – `comet`, `getframe...movie`.

Пример:

1. Сформировать случайные векторы $\mathbf{u}$ и $\mathbf{v}$ размера 1×4 .

```
>> n=4;u=rand(1,n);v=rand(1,n);[u;v]
```

```
ans =  
    0.4740    0.9090    0.5962    0.3290  
    0.4782    0.5972    0.1614    0.8295
```

2. На основе этих векторов составить матрицу $\mathbf{A}=[\mathbf{a}_{ij}]$, $\mathbf{a}_{ij}=\mathbf{u}_i \mathbf{v}_j^{j-1}$.

```
>> % Выполним все без циклов. Начнем с формирования вектора из  
единиц
```

```
>> w=[linspace(1,1,n)];w=w';  
>> % Сделаем первую заготовку  
>> A1=(w*u)'
```

```
A1 =  
  
    0.4740    0.4740    0.4740    0.4740  
    0.9090    0.9090    0.9090    0.9090  
    0.5962    0.5962    0.5962    0.5962  
    0.3290    0.3290    0.3290    0.3290
```

```
>> % Составим вторую заготовку, содержащие степени  
>> q=[0:1:n-1]; A2=w*q  
A2 =
```

```
    0    1    2    3
```

```

0 1 2 3
0 1 2 3
0 1 2 3
>> % Осуществим поэлементную операцию возведения в степень
>> A3=w*v;A3=[A3.^A2]

```

A3 =

```

1.0000 0.5972 0.0261 0.5707
1.0000 0.5972 0.0261 0.5707
1.0000 0.5972 0.0261 0.5707
1.0000 0.5972 0.0261 0.5707
>> % Поэлементным умножением матриц приходим к искомому
>> A=A1.*A3
A =
0.4740 0.2831 0.0124 0.2705
0.9090 0.5428 0.0237 0.5188
0.5962 0.3561 0.0155 0.3403
0.3290 0.1964 0.0086 0.1877
>> % Стираем ненужное
>> clear ('A1','A2','A3','w')

```

3. Для каждого элемента матрицы A вычислить значение полинома с коэффициентами (u+iv). Результаты записать в матрицу B. Превратить ее в вещественную, взяв от каждого элемента его угол поворота. Сколько чистого времени потребовалось на выполнение п.3?

```

>> tic; B=polyval(u+i*v,A);B=angle(B);toc
Elapsed time is 0.003572 seconds.

```

4. Составить анонимную функцию для вычисления площади треугольника по формуле Герона ($S = \sqrt{p(p-a)(p-b)(p-c)}$, p- полупериметр, a,b,c – стороны). Вычислить площадь для египетского треугольника.

```

>> % Возможен только один оператор. Площадь полагаем нулю,
если из заданных отрезков нельзя составить треугольник
>> geron=@(a,b,c) real(0.25*sqrt((a+b+c)*(a+b-c)*(a+c-b)*(b+c-a)));
>> geron(3,4,5)

```

```

ans =
6

```

5. Построить график зависимости S(a) на отрезке [0,πe], приняв b=е,c=π. Оформить подписи, цвет кривой сделать фиолетовым.

>> % Есть два пути решения. 1-й состоит в переписывании анонимной функции.

>>% Команда figure избыточна. Hfig – указатель на текущий графический объект

```
>> geron1=@(a,b,c) real(0.25*sqrt((a+b+c).*(a+b-c).*(a+c-b).*(b+c-a)));
```

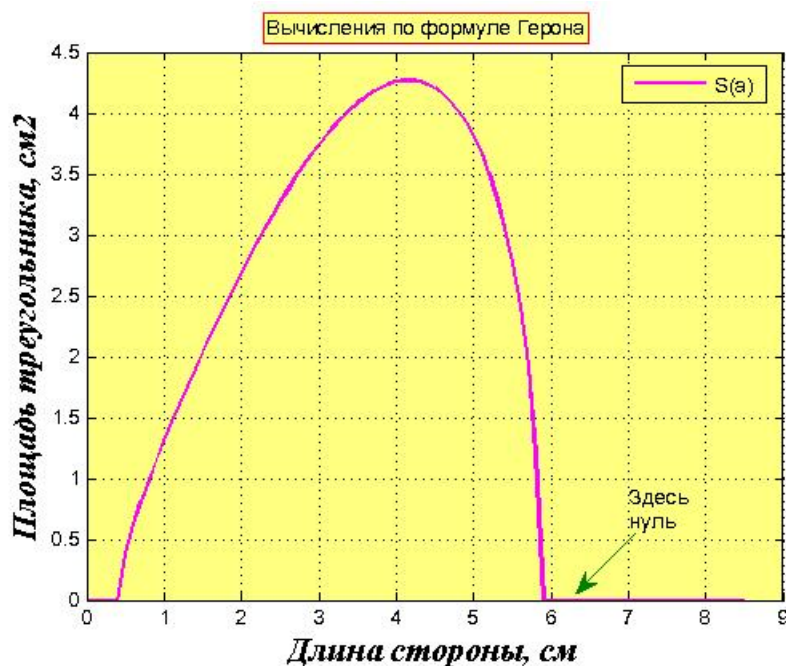
```
>> x=0:0.1:(pi*exp(1));Hfig=figure(1);plot(x,geron1(x,exp(1),pi));
```

>>% Второй путь, более общий, заставляет обратиться к программированию. Некоторые конструкции типа for...end доступны и в консоли.

>> % Ниже формируется вектор y, длина которого постоянно увеличивается. Затем непосредственно вывод графика.

```
>> x=0:0.1:(pi*exp(1));y=[ ]; for k=x y=[y,geron(k,exp(1),pi)]; end;
```

```
>> Hfig_ =figure(2);plot(x,y);
```



В окне Figure2 щелчком мышью по иконке со стрелкой, перейдя к режим редактирования. Вызовем контекстное меню, связанное с кривой – пункт Color, и выберем цвет. Аналогичным способом выберем цвет фона. Сделаем и другие манипуляции с осями, подписями и пр. Полезен пункт главного меню Insert,

пункт контекстного меню – Show Properties. Удобно также подключить редактор свойств (View – Property Editor). Исполнив File– Generate M-file, можно просмотреть совершенные нами действия в виде команд.

6. Построить семейство кривых $S(a)$ при пяти значениях $b=0,1,2,3,4$ и фиксированном $c=5$. Величина a лежит в пределах $[0;8]$. Точки маркировать.

```
>> % Закрыли 2-й график, открыли новый с указанием рисовать все там
```

```
>> close(Hfig_);figure(3);hold on;
```

```
>> % Инициализируем переменные
```

```
>> a=0:0.5:8;b=0:1:4;c=5;
```

>>% Команды занимают две строки. Организуем цикл, формируем

массив указателей на линии, цвет линии в формате RGB (смесь красного, зеленого и синего) случайный

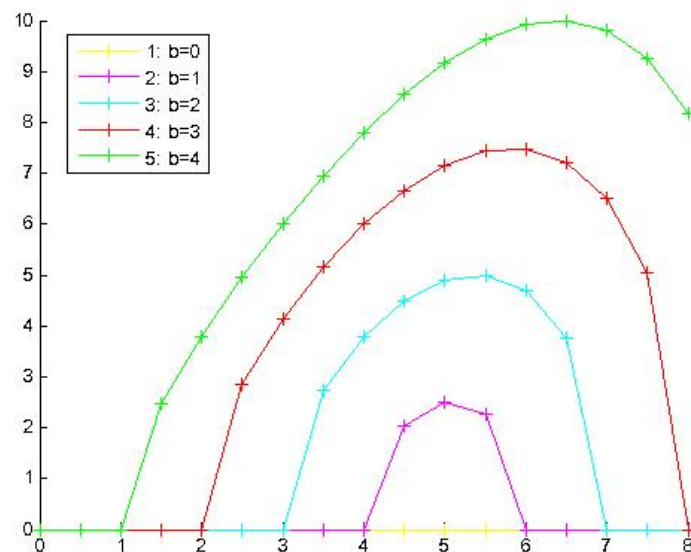
```
>> Hlines=[ ];for j=1:5, Hline=plot(a,geron1(a,b(j),c)); Hlines=[Hlines  
Hline]; ...
```

```
set(Hline,'Marker','+','Color',[rand rand rand]); end; hold off
```

>> %Изменим цвета линий на привычные (см. ColorSpec) и добавим подписи к линиям. Вот где понадобились указатели! Названия получаем преобразованием чисел в литералы

```
>> MyColor=['y' 'm' 'c' 'r' 'g' 'b' 'w' 'k'];
```

```
>> for j=1:5, Lname=strcat(num2str(j),': b= ',num2str(b(j)));  
set(Hlines(j),'Color',MyColor(j),'DisplayName', Lname); end; legend('show');
```



7. Построить контурный график на основе базиса (u,v) и матрицы значений В, которая содержит высоты «гор». Площадь построения – [-2;2]× [-2;2].

```
>> % Вначале создадим функцию растягивания координат, поскольку  
0<ui&vi<1
```

```
>>stretch=@(x) 4/(max(max(x))-min(min(x)))*x-  
2*(max(max(x))+min(min(x)))/(max(max(x))-min(min(x)));
```

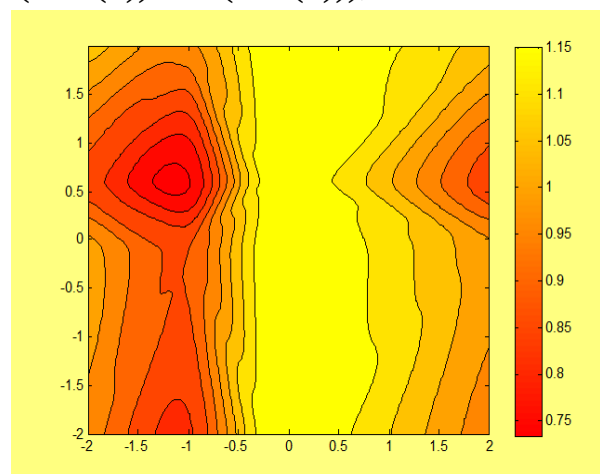
```
>> % Построим первоначальную сетку
```

```
>>
```

```
[Xbig,Ybig]=meshgrid(u,v)
```

```
Xbig =
```

```
0.4740 0.9090 0.5962  
0.3290  
0.4740 0.9090 0.5962
```



0.3290

0.4740	0.9090	0.5962	0.3290
0.4740	0.9090	0.5962	0.3290

Ybig =

0.4782	0.4782	0.4782	0.4782
0.5972	0.5972	0.5972	0.5972
0.1614	0.1614	0.1614	0.1614
0.8295	0.8295	0.8295	0.8295

```
>> % Построим подробную сетку в области, заданной векторами  
u,v. Эта область обязана не превосходить существенно предыдущую об-  
ласть [Xbig Ybig]
```

```
>>
```

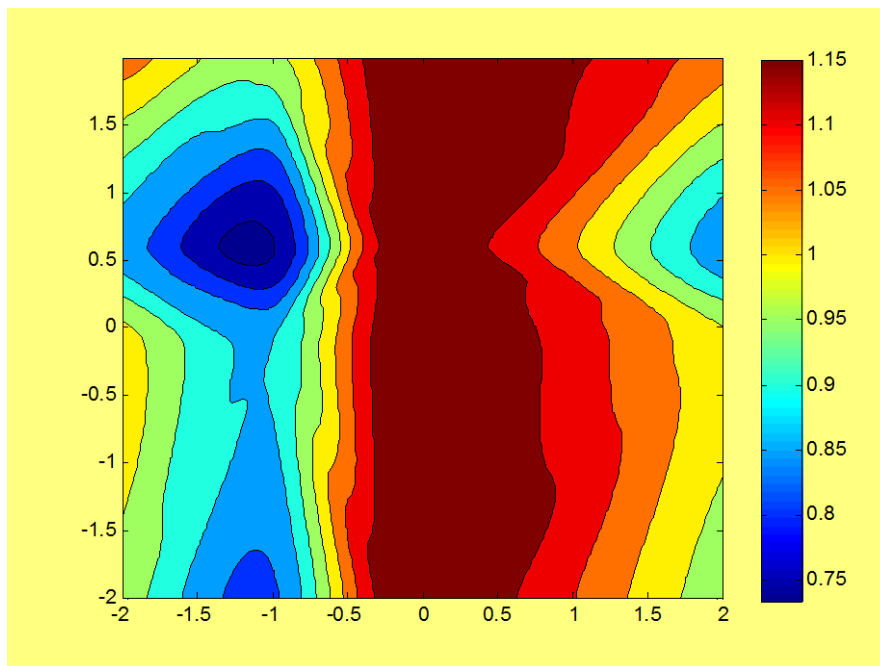
```
[X,Y]=meshgrid(linspace(min(u),max(u),100),linspace(min(v),max(v),100));
```

```
>> % Проведем интерполяцию с использованием кубических полиномов
```

```
Z=griddata(Xbig,Ybig,B,X,Y,'cubic');
```

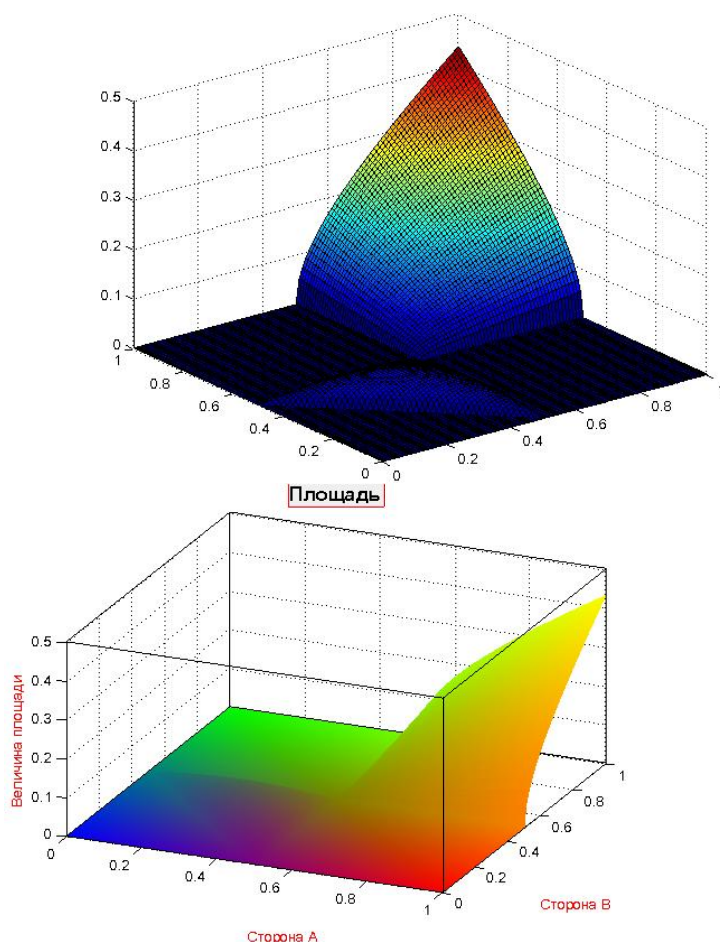
```
>> % Построим контурный график с заливкой для квадратной площадки  
contourf(stretch(X),stretch(Y),Z)
```

Для того, чтобы скопировать рисунок в буфер обмена, исполнить пункт Edit – Copy Figure меню графического окна, а если хотим скопировать его вместе с фоном, то поставить галочку File – Preferences – File Copy Template – Copy Options – Use Figure Color. Чтобы добавить расшифровку цветов, исполнить Insert – Colorbar. Можно цветовую гамму выбрать монотонной; для этого имеется пункт Edit – Colormap..., а в появившемся окне можно выбрать осеннюю гамму цветов (Tools – Standart colormaps – autumn) – см. рис. Сверху-справа. Для того, чтобы обрезать у рисунка неиспользуемые поля – поставить галочку File – Export Setup – (Properties=Size) – Expand Axes....



8. Построить поверхность площадей $S(a,b,c)$, вычисленную для изопериметрических треугольников (т.е. $a+b+c=1$).

```
>> % Подготовим сетки и значения, цвет выберем потом
>> a=0:0.01:1;b=0:0.01:1; [X,Y]=meshgrid(a,b);Z=geron1(X,Y,1-X-Y);
    >> % Инициализируем графический объект и построим в первом
приближении. Функция surf позволяет указывать вектора вместо матриц,
т.е. как бы функция meshgrid выполняется автоматически
>> figure(5);Hsur=surf(a,b,Z)
    >> % Обычно цветом отмечается высота точки. Введем свои цвета
(в палитре RGB) и немного изменим угол зрения. Также уберем линии
сетки поверхности.
    >> C=cat(3,X,Y,1-X-Y); set(Hsur,'CData',C, 'LineStyle','none');
view([23.5 36]);
    >> xlabel('Сторона A','Color',[1 0 0]); ylabel('Сторона B','Color',[1 0
0]); zlabel ('Величина площади', 'Color',[1 0 0],'Rotation', 90);
    >>title('Площадь','BackgroundColor',[0.9412 0.9412 0.9412],..
'EdgeColor',[1 0 0],'FontSize',14,'FontWeight','bold','LineWidth',1);
    >>axis([0 1 0 1 0 0.5]); box('on');
```



Примечание: Укажите ошибку вычислений при выполнении п.8. Как ее исправить? См. п.10.

9. Построить в декартовых координатах кривую, заданную параметрически на $[-\pi; \pi]$:

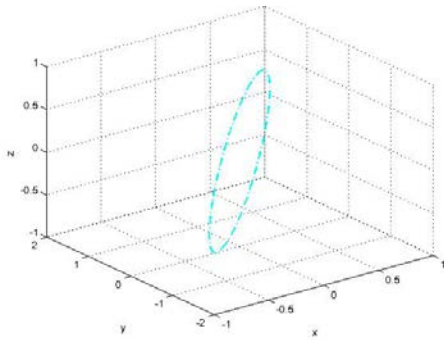
$$\vec{r}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \tilde{u} \sin t + \tilde{v} \cos t, \text{ «тильда» — первые три компонента одноименных}$$

векторов.

```
>> MyCurve1=@(t) u(1)*sin(t)+v(1)*cos(t); MyCurve2=@(t)
u(2)*sin(t)+v(2)*cos(t);
```

```
>> % Обратите внимание, что мы здесь не делаем матриц, используя об-
легченный вызов функции-рисовалки – поэтому не можем применять индекса-
цию по осям
```

```
>> MyCurve3=@(t) u(3)*sin(t)+v(3)*cos(t);
ezplot3(MyCurve1,MyCurve2,MyCurve3,[-pi,pi])
```



10. Создать киноленту, отображающую эволюцию поверхности треугольников при изменении третьей стороны треугольника.

```
>> % Формируем сетку и область переменной времени, она же равна c
>> a=0:0.01:1;b=0:0.01:1;c=0:0.01:2;[X,Y]=meshgrid(a,b);
>> % Последовательно создаем графические объекты: рисунок, про-
    пространство осей, поверхность (пока пробную)
>> HFig1=figure(1); axes();Hsur=surf(X,Y,ones(length(X)));set(HFig1, 'Pa-
    perType', 'A4', 'Color','y');
>> % Модифицируем параметры объектов. Убираем сетку, делаем цве-
    товую гамму неподвижной (важен порядок вызова установки пределов третьей
    оси и цветовой легенды
>> colormap('jet'); set(gca,'Box','on'); caxis([0 0.5]); colorbar;
    set(Hsur,'LineStyle','none')
>> % Основной цикл – смена данных построения. На лету формируем
    массив из скриншотов-картинок
>> for j=1:length(c) set(Hsur,'Zdata',geron1(X,Y,c(j))); MyMovie(j) = get-
    frame(HFig1); end;
>> % Проигрываем кино 5 раз подряд. Записываем avi-файл, его можно
    проиграть стандартными средствами Windows.
>> movie(MyMovie,5); movie2avi(MyMovie,'mymovie1.avi','fps',4)
```

Задание:

1. Сформировать случайную комплекснозначную матрицу A размера 3×3 , вычислить матричную экспоненту $B = e^A$. Решить систему линейных уравнений $Az = (1, 1, 1)^T$.

2. Сформировать, не используя циклов, матрицу вида $\begin{pmatrix} AB & B \\ A & [a_{ij}b_{ij}] \end{pmatrix}$ и выделить из нее вещественную ($D1$) и комплексную ($D2$) часть

3. Вычислить с помощью анонимной функции электрический потенциал системы точечных электрических зарядов, заданных координатами ($D1, D2$), в точке с координатами $(0, 0)$. Заряды положительные, по модулю равны 1.

4. Построить контурный график потенциала на площади $[-2; 2] \times [-2; 2]$. Выполнить цветовую легенду в логарифмическом масштабе. В графическом редакторе оформить график красиво.

5. Построить в полярных координатах 2D-график напряженностей поля (velocity plot).

6. Построить 3D-график п.4.

7. Создать анимацию контурного графика в п.4. при движении координат зарядов по закону:

$$D1_{ij} = D1_{ij}(1 + a \cos(j + it)) \quad D2_{ij} = D2_{ij}(1 + a \sin(i + jt)), \quad 0 < t < 2\pi, \quad a = 0.2$$

Примечание: поскольку графика требует времени для своего построения, то использовать команду `pause` для задания задержек. Более предпочтительный путь – команда `drawnow` до взятия фрейма.

Лабораторная работа №10

Проведение вычислений с помощью М-файлов

1. Создание и отладка М-файлов.

MATLAB является интерпретирующим языком непосредственных вычислений, т.е. выражения, которые вы вводите, интерпретируются и вычисляются. Поскольку все вычисления в MATLAB выполняются с двойной точностью, формат вывода может управляться с помощью следующих команд

<code>format short</code>	с фиксированной точкой и 4 знаками после точки (по умолчанию)
<code>format long</code>	с фиксированной точкой и 14 знаками после точки
<code>format short e</code>	научная нотация с 4 десятичными знаками
<code>format long e</code>	научная нотация с 15 десятичными знаками

После вызова одного из приведенных выше форматов он сохраняется до вызова другого. Команда `format compact` подавляет большинство пустых строк, позволяя большее количество информации вывести на экран или страницу. Она не зависит от других команд формата. Команда `eval` чрезвычайно полезна, позволяя обмениваться между процедурами (особенно, внешними) строкой-параметром, которая принуждает систему к выполнению заданной этой строкой команды (или серии команд). Напомним, что MATLAB различает большие и маленькие буквы в именах команд, функций и переменных, а спецзнаки могут быть записаны через обратный слэш.

Для простых операций удобен интерактивный режим, но если вычисления нужно многократно выполнять или необходимо реализовывать сложные алгоритмы, то следует использовать m-файлы MATLAB (расширение файла состоит из одной буквы m). Познакомимся со script-m-файлами (или сценариями) – текстовыми файлами, содержащими инструкции на языке MATLAB, подлежащими исполнению в автоматическом пакетном режиме. Создать такой файл удобнее с помощью редактора системы MATLAB (не забудем способ создания, исходя из окна истории команд). Он вызывается из командного окна системы MATLAB командой меню File/New/M-file (или самой левой кнопкой на полосе инструментов, на которой изображен чистый белый лист бумаги). Записанные в script-файлы команды будут выполнены, если в командной строке ввести имя script-файла (без расширения). Переменные, определяемые в командном окне и переменные, определяемые в сценариях, составляют единое рабочее пространство системы MATLAB, причем переменные, определяемые в сценариях, являются глобальными, их значения заместят значения таких же переменных, которые были использованы до вызова данного script-файла. Текст реально имеющихся m-файлов (системы MATLAB, например `polar`, или ваших собственных) можно просмотреть с помощью команды `type <имя_функции>`.

Однако, большинство m-файлов являются файлами-функциями, т.е. программами. При том же порядке вызова в отличие от простого линейного скрипта, хотя бы тот и содержал, например, циклические конструкции, нормальный m-файл содержит тело, окаймленное следующими декларациями:

```

function [out1, out2, ...] = funname(in1, in2, ...)
    <body>
end
Тело может быть представлено в таком формате:
<body>=
%Строки основного комментария, доступного по команде help
% Далее три четыре основных повторяющихся типа конструкций

% Объявление собственных переменных и команды MATLAB
<переменная=выражение>; <команда>
% Объявление подфункций (пишется обычно с отступом)
    function [out]=nested_funname (in)
        <операторы;>
    end;
% Конструкции языка типа for, if,...
If <условие>
    <операторы;>
End

```

В заголовке явном виде пишутся выходные аргументы, что несколько отличается, например, от Си-программ (там слово `function` равнозначно `procedure`). Сходные декларации имеют место для подфункций и пишутся внутри тела, которое по сути тождественно пакету команд. MATLAB рассматривает `m`-файл как одну «главную» (`primary`) функцию, причем ее имя обязано совпадать именем `m`-файла; внутри этой функции могут располагаться декларации локальных (вложенных, `nested`) функций, доступных только изнутри данного `m`-файла. Ниже главной функции в редких случаях могут располагаться подфункции, но доступ к ним возможен только из главной – поэтому их лучше все-таки реализовывать как вложенные. Внутри `m`-файлов можно ссылаться на другие `m`-файлы, в том числе и на самого себя рекурсивно. Тем самым реализуются, хотя и не полностью, принципы процедурного программирования. Переменные в функциях являются по умолчанию локальными, но в версиях 4.0 и выше разрешено объявлять требуемые переменные глобальными (`global`). Знак “;” получает двойное толкование: с одной стороны – это подавление вывода в консоль, с другой – привычный разделитель между операторами языка.

Как и в библиотечных функциях MATLAB, используется принцип избыточного задания аргументов. Это позволяет варьировать действия в зависимости от формы и количества аргументов (см. `nargin`, `nargout` и особенно параметр декларации функции `varargin`). Аналогично, доступ к результатам выполнения может быть частичным. При вызове функции для нее создается стек памяти, в который помещаются помимо локальных переменных копии одноименных переменных вызова. Для простого интерфейса, использующего только консоль, предусмотрены операторы `disp`, `input`, `error`. Как и в большинстве языков/сред программирования, MATLAB обладает стандартными оператора-

ми ветвления, выбора и цикла (кроме goto). Особенность for заключается в использовании массива/матрицы в качестве индекса, при этом счетчик последовательно пробегает значения, быть может вещественные, элементов/столбцов матрицы. Например если вам необходимо выполнить <оператор> только для тех элементов матрицы, которые больше 3, то удобно это сделать следующим образом:

```
for i=find(A>3) <оператор> end;
```

Редактор/отладчик предоставляет как средства редактирования текста m-файла, так и средства пошаговой его отладки. Один из способов вызова редактора – вызов из командной строки MATLAB с помощью команды edit. Редактор, используемый в системе, имеет синтаксическую раскраску, т.е. слово или символ по мере ввода приобретают тот цвет, который соответствует их типу. С помощью пункта меню Tools-Fonts можно настроить такие важные параметры, как используемый шрифт. Это особенно важно для работы с русским текстом, поскольку не все шрифты правильно воспроизводят русский текст. Редактор имеет стандартный набор возможностей (запуск M-файла, расстановка точек останова – breakpoints), так и специальные: переключение в Cell Mode (режим ячеек), публикация html, пункт Evaluate Selection, который позволяет вычислять значение выделенного выражения и помещать результат в консоль, пункт Edit-Paste to Workspace. В режиме ячеек легко отлаживать и читать программу; с той же целью введена пиктограмма Show Functions, позволяющая перескакивать по заголовкам вложенных функций. Более детальные сведения можно прочесть, нажав пункт меню редактора Help – Using the M-file Editor.

Справа от редактируемого файла находится линейка номеров строк, а слева (рядом с полосой прокрутки) – линейка ошибок. Если в строке найдена синтаксическая ошибка, то имеем красный маркер на линейке ошибок; если ошибка некритичная (warning), то цвет маркера желтый. Код может быть предварительно проверен (это чем-то напоминает этап компиляции) - с помощью пункта меню Tools – Check Code with M-Lint (вызывается специальное окно с перечнем найденных ошибок). Помимо синтаксических есть ошибки времени выполнения. Ошибки времени выполнения выявить более сложно, потому что локальная рабочая область m-функции оказывается потерянной, если ошибка приводит к возврату в рабочую область системы MATLAB. Чтобы определить причину такой ошибки, можно использовать один из следующих приемов:

- реализовать вывод результатов промежуточных вычислений на дисплей, удалив в соответствующих операторах точки с запятой, которые подавляют вывод на экран промежуточных результатов;
- добавить в m-файл команды keyboard, которые останавливают выполнение m-файла и разрешают проверить и изменить переменные рабочей области вызываемой m-функции. В этом режиме появляется специальное приглашение K». Возврат к выполнению функции реализуется командой return;
- закомментировать заголовок функции и выполнить m-файл как сценарий. Это позволяет проследить результаты промежуточных вычисле-

ний в рабочей области системы;

- использовать отладчик системы MATLAB (см. несколько пиктограмм на панели справа) – только в случае отсутствия входных аргументов.

Пример:

1. Построить и оформить графики трех функций $y(t) = t \ln t + \sqrt[3]{1+t^2} \sin t$, $z(t) = \frac{1+t^3}{1+t^2}$, $w(t) = z(t) - y(t)$ на $[0;3]$. Из серии набранных команд собрать М-файл-сценарий.

Создадим анонимные функции

```
>> funy=@(x) (x.^2).*log(x+eps)+(1+x.^2).^(1/3);  
>> funz=@(x) (1+x.^3)./(1+x.^2); funw=@(x) funz(x)-funy(x);
```

Проверим правильность задания

```
>> X=[0 1 2];[funy(X); funz(X); funw(X)], clear('X')
```

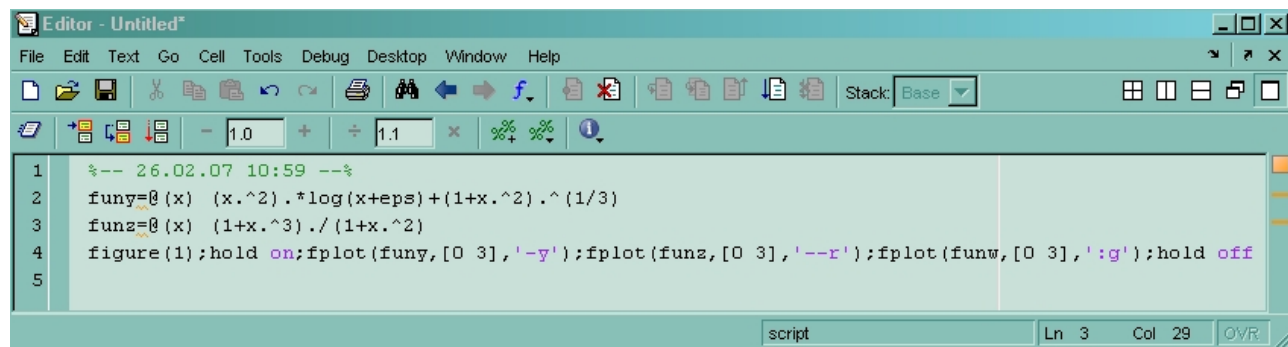
ans =

```
1.0000 1.2599 4.4826  
1.0000 1.0000 1.8000  
0 -0.2599 -2.6826
```

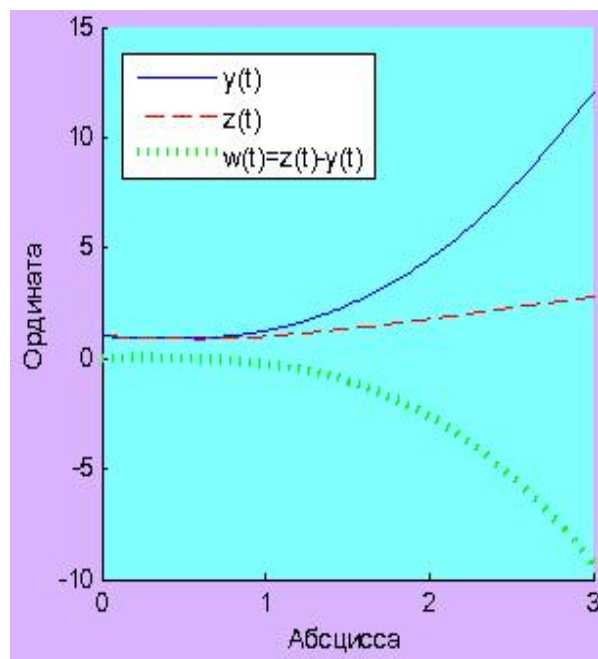
Строим графики простейшим путем

```
>> figure(1);hold on;fplot(funy,[0 3],'-y');fplot(funz,[0 3],'-r');fplot(funw,[0 3],':g');hold off
```

С помощью контекстного меню (Create...) в History Commands формируем М-файл текущей сессии (лишние команды можно затем стереть в редакторе)



Проведем украшение графика (скажем, заменим желтый цвет линии на синий и добавим легенду). Исполним пункт File/Generate M-file. Откроется новый файл Untitled-2. Скопируем все секции оттуда (кроме первой, начиная со строки %Create Figure). Сохраним файл под именем MyGraph.m в текущий каталог.



К сожалению, несмотря на полезные конструкции сгенерированного кода, он был исполнен в виде функции, поэтому придется провести ряд изменений, чтобы сделать его работоспособ-

ным. В первой секции заменим `figure1 = figure()` на `set(gcf, -` ведь полотно графиков уже создано. Во второй секции первая строка приобретет вид `set(gca,'Color',[0.502 1 1]);`. Также следует убрать третью ось в команде прорисовки осей `axis`. С секциями «% Create plot» сложнее, здесь придется операцию создания кривой заменять на поиск указанной кривой и установку ее свойств (см. `findobj`). Это позволит нам заменить все «пустые» ссылки (дескрипторы).

Общий вид кода и смасштабированного результата его выполнения (по команде `MyGraph` в консоли) см. на рис. снизу и слева.

```

1  %-- 26.02.07 10:59 --%
2  - funy=@(x) (x.^2).*log(x+eps)+(1+x.^2).^(1/3);
3  - funz=@(x) (1+x.^3)./(1+x.^2);
4  - funw=@(x) funz(x)-funy(x);
5  - figure(1);hold on;fplot(funy,[0 3],'-y');...
6  -     fplot(funz,[0 3], '--r');fplot(funw,[0 3],':g');hold off
7
8  % Create figure
9  - set(gcf,...
10     'Color',[0.8549 0.702 1],...
11     'InvertHardcopy','off',...
12     'Name','Три графика',...
13     'NumberTitle','off',...
14     'PaperSize',[20.98 29.68]);
15
16 % Create axes
17 - set(gca,'Color',[0.502 1 1]);
18 - axis([0 3 -10 15]);
19 - hold('all');
20
21 % Change plots
22 - HLines = findobj(gca,'Type','line');
23 - set(HLines(1),'Color',[0 0 1]);
24 - set(HLines(2),'Color',[1 0 0],'LineStyle','--');
25 - set(HLines(3),'Color',[0 1 0],'LineStyle',':','LineWidth',3);
26
27 % Create xlabel
28 - xlabel('Абсцисса');
29
30 % Create ylabel
31 - ylabel('Ордината');
32
33 % Create legend
34 - legend1 = legend(gca,{'y(t)','z(t)','w(t)=z(t)-y(t)'},...
35     'Position',[0.1896 0.7425 0.4257 0.1909]);
36

```

2. Записать в виде М-функции вычисления по формуле Герона в файле `geroncore`. Вычислить площадь при наборах (1,2,1.5) и (1,2,4).

```

function S=geroncore(a,b,c)
    p=(a+b+c)/2;
    S=sqrt(p.*(p-a).*(p-b).*(p-c));S=real(S);
end
>> geroncore([1 1],[2 2],[1.5 4])
ans =

```

0.7262 0

3. Записать в виде М-функции MyTriangle решатель треугольника, т.е. по трем заданным его элементам находящий остальные – число аргументов и результатов переменное.

Приготовим шаблон М-файла, разобьем его на секции (см. иконку «%%+» на второй панели редактора) и сразу его запишем

```
function [MyS, varargout]=MyTriangle (varargin)
% Эта функция "решает" треугольник. Стандартный вызов предусматривает
% 4 входных и 4 выходных аргумента. Первые три аргумента -
длины трех
% сторон, четвертый - строка вида 'abA', где большими буквами
указаны
% углы, а малыми - стороны (угол A противолежит стороне a). Пятый аргумент
% предусматривает дополнительный расчет биссектрис, медиан или высот.
%%
% Основные внутренние переменные
a=1;b=1;c=1;A=60;B=A;C=A;MyS=0;MyLegend='abc';
%%
%Проверка данных

%%
%Запись данных

%%
%Вычисление результата

%%
%Вывод результатов
varargout(1)=varargin(1);

end
```

Будем наполнять содержание каждой секции. Начнем предварительную проверку данных. При желании формат данных можно выверять тщательнее. Пока же проверяем число аргументов и формат легенды, которая пишется как массив символов в MyLegend. Для примера мы сохранили закомментированную «точку останова» keyboard. Особенность переменной varargin состоит в том, что она является массивом ячеек (cell array), типом промежуточным между классической записью и массивом. Поэтому для извлечения данных используется функция char. Проще, но менее привычно, было бы записать

varargin {4} вместо varargin(4).

```
%Проверка данных
switch nargin
    case {0,1,2}
        warning('Слишком мало аргументов'), flag=false;
    case {6,7,8}
        warning('Слишком много аргументов'), flag=false;
    otherwise
        flag=true;
        if (nargin~=3) MyLegend=char(varargin(4)); end;
        %keyboard;
end;
if (~flag) return; end;
MyLegend=MyLegend(1:3);flag=true;NAngles=0;
for s=MyLegend
    switch s
        case {'a','b','c'}
        case {'A','B','C'}
            NAngles=NAngles+1;
        otherwise
            flag=false;
    end;
    if (~flag) break; end
end;
if (NAngles==3) error('По трем углам нельзя решить треугольник'),
    return; end;
if (~flag) error('Неправильный формат легенды данных'), return;
end;
```

Для записи данных используем простой цикл и полезную команду eval.

```
%Запись данных
```

```
for (j=1:3) eval(strcat(MyLegend(j),'=',num2str(varargin{j}),',')); end;
```

Для расчета нам нужно разделить секцию на две части, в первой – описания функций, а во второй – вызывающий их код. Заключительная команда вызывает функцию `geopscore` из одноименного М-файла. Мы также оставили точку останова `keyboard`; поучительно рассмотреть одинаково называющиеся переменные, имеющие разную область действия – см. во время второго останова селектор `Workspace-Stack`. В стеке функции `b=5`, а стеке программы `b=4` – интерпретатор не путается в действии принципа «локальное имя закрывает глобальное». Теперь программу можно осмысленно запускать, временно раскомментаривая последнюю строчку – например, `>> MyTriangle(3,4,5,'abc')`.

```

% Сначала список функций, математически все сводится к 5 слу-
чаям
%%
% Теорема синусов. Два угла прилежат стороне.
% Находится сторона у первого угла
function res=SIN_side (a,alfa,beta)
    if (alfa+beta>180) warning('Два тупых угла невозможны');
    res=NaN;return;
    end;
    res=a.*sin(deg2rad(beta))./sin(deg2rad(alfa+beta));
end
% Теорема косинусов. Находится сторона возле угла
function res=COS_side (b,c,gamma)
    a1=c.^2-(b.*sin(deg2rad(gamma)))^2;
    if (a1<0) warning('Такого треугольника не существует');
    res=NaN;return;
    end;
    a2=b+sqrt(a1);a1=b-sqrt(a1);
    if (a1<0) warning('Найдено два таких треугольника!'); end;
    res=a2;
end
% Теорема косинусов. Находится сторона против угла
function res=COS_opposite (a,b,gamma)
    res=sqrt(a.^2+b.^2-2*a.*b.*cos(deg2rad(gamma)));
end
% Теорема косинусов. Находится угол (в градусах)
function res=COS_angle (a,b,c)
    if ((a<eps)|| (b<eps)|| (a+b<c))
        warning('Не выполнено неравенство треугольника');
        res=NaN;return;
    end;
    keyboard;
    res=acosd((a.^2+b.^2-c.^2)./(2*a.*b));
end
%%
% Главный блок расчета
switch MyLegend
case 'abc'
    C=COS_angle(a,b,c); if(C==NaN) return; end;
    B=COS_angle(a,c,b); if(B==NaN) return; end;
    A=COS_angle(c,b,a); if(A==NaN) return; end;
case 'abC'
    c=COS_opposite(b,a,C);if(c==NaN) return; end;
    B=COS_angle(a,c,b); if(B==NaN) return; end;
    A=180-B-C;

```

```

case 'abA'
    c=COS_side(b,a,A);if(c==NaN) return; end;
    B=COS_angle(a,c,b); if(B==NaN) return; end;
    C=180-A-B;
case 'aBC'
    b=SIN_side(a,C,B);if(b==NaN) return; end;
    c=COS_opposite(a,b,C);if(c==NaN) return; end;
    A=180-B-C
case 'aAB'
    C=180-A-B; if(C<0) return; end;
    b=SIN_side(a,C,B);if(b==NaN) return; end;
    c=COS_opposite(a,b,C);if(c==NaN) return; end;
otherwise
    disp('Будет улучшено в след. версиях')
end

MyS=geroncore(a,b,c);
%[a b c A B C]

```

Проведем дополнительные вычисления медиан, высот и т.д., в соответствии в 5-м параметром вызова. Обратите внимание, что подфункция может вызывать ранее определенную подфункцию; также – несмотря на краткость определений функций, каждая должна занимать три строки.

```

%%
% Дополнительные расчеты по пятому параметру:
% med - медианы, bis - биссектрисы, hhh - высоты
MyS=geroncore(a,b,c);

% Объявление новых функций
function res=median(a,b,gamma)
    res=COS_opposite(a,0.5*b,gamma);
end
function res=bisectrix(a,b,gamma)
    res=2*MyS./((a+b).*sind(gamma/2));
end
function res=height(c)
    res=2*MyS./c;
end
% Расчет линий
for i=1:3, varargout{i}=i; end;
if (nargin==5)
    switch varargin{5}
        case 'med'
            varargout{4}=median(b,a,C);varargout{5}=median(a,b,C);
            varargout{6}=median(b,c,A);

```

```

case 'bis'
    varargout{4}=bisectrix(c,b,A);varargout{5}=bisectrix(c,a,B);
    varargout{6}=bisectrix(a,b,C);
case 'hhh'
    varargout{4}=height(a);varargout{5}=height(b);
    varargout{6}=height(c);
otherwise
    ;
end
end;

```

Последняя секция короткая. Список аргументов varargout, как и varargin, - это массив ячеек, поэтому присвоения стоят в {}, а не в привычных (). При выводе графика оси рисуются после построения графика, в противном случае оси будут масштабироваться по данным.

```

%%
%Вывод результатов
varargout{1}=[a A];varargout{2}=[b B];varargout{3}=[c C];
% Как последний штрих - вывод изображения
if (nargin==5)&&(strcmp(varargin{5},'grh'))
    figure(1);plot([1 a.*cosd(C)+1 b+1 1], [1 a.*sind(C)+1 1 1]);
    axis([0 (a+b+c)/2 0 (a+b+c)/2]);
end

```

Для считывания результата приходится делать вызов программы не слишком красивым, например, [x,y,z,t,u,v,w]=MyTriangle(3,4,5,'abc','grh'). Все числа можно было бы расположить компактно в одной переменной-массиве (в качестве результата).

Задание:

1. Построить график функции, заданной в цилиндрических координатах уравнениями: $z = 1 + \sqrt{Ar} \sin \varphi$, $r = \cos^2 A\varphi$, $\varphi \in [0; 2\pi]$. Составить М-файл-сценарий и запустить его несколько раз. На что похожа кривая? Сделать ее «кометой».

Указание: Использовать функции pol2cart, plot3, comet3. Параметр A>0 свободный.

2. Оформить функцию myfun1, имеющую два аргумента (вектор и строку) и один результат – скаляр. Вычисление скаляра зависит от вида строки (предусмотреть три варианта расчета) – например, может рассчитываться среднее арифметическое. Протестировать М-файл.

3. Написать программу, вычисляющую средний балл студента. Пользователь должен в консоли вводить имя студента, название предмета и полученный балл. Первым, в качестве базы, вводится

название предмета.

Указание: Приветствуется замена циклов встроенными функциями (find,sort), базу данных объявить глобальной переменной (global) и реализовать как массив ячеек (cell array).

2. Решение некоторых стандартных математических задач

В практике математического моделирования часто приходится сталкиваться с необходимостью решать численно некоторые стандартные задачи. К их числу можно отнести:

- Вычислений значений спецфункций
- Задача интерполяции – узнать каково значение функции в некоторой точке интервала, если заданы ее значения в нескольких его точках.
- Поиск решения системы линейных уравнений, вычисление собственных векторов и пр.
- Поиск корней полинома $P_n(x)=0$ и вообще, нелинейного уравнения $f(x)=0$
- Вычисление коэффициентов разложения Фурье (в т.ч. быстрое Фурье–преобразование)
- Вычисление значений производных, дивергенций, определенных интегралов от функций
- Задачи поиска наименьшего и наибольшего значений (т.е. локальных и глобальных экстремумов)
- Поиск решений обыкновенных дифференциальных уравнений и их систем (задача Коши)
- Краевые задачи для некоторых уравнений в частных производных

Всю необходимую информацию можно почерпнуть в разделе справки MATLAB-Mathematics, а также в разделе MATLAB-Functions_by_Category-Mathematics. Кратко о некоторых полезных функциях (по их имени легко восстановить формат и особенности вызова):

`legendre(n,X)` – вычисление значений полиномов Лежандра порядка n в точках вектора X .

`yy = spline(x,Y,xx)` – интерполяция кубическими сплайнами $yy(xx)$ (с возможной экстраполяцией), xx – как скаляр, так и вектор.

`poly(A)` – дает коэффициенты характеристического многочлена матрицы A .

`roots(p)` – дает корни полинома, заданного вектором коэффициентов.

`fzero(fun,x0)` – находит нуль функции, заданной указателем `fun`, в окрестности x_0 .

`optimset` – устанавливает параметры оптимизации (например, для `fzero` – заметим, что решение алгебраических уравнений может быть сведено к оптимизационной задаче; например, $f(x^*)=0 \leftrightarrow \min f^2(x)$). Параметр

DisplayLevel='iter' позволяет отображать на экране ход итераций процесса и тем самым «почувствовать» успешность оптимизации.

fminsearch(fun,x0,options) – ищет безусловный экстремум функций многих переменных.

Для решения более сложных оптимизационных задач см. приложение MATLAB Optimization Toolbox.

gradient(F) – дает градиент функции, заданной таблично.

dblquad(fun,xmin,xmax,ymin,ymax) – двукратный интеграл Римана на прямоугольнике.

fft2(X) – быстрое двумерное Фурье-преобразование.

odeset – устанавливает параметры для решения обыкновенных дифференциальных уравнений (ОДУ)

ode45 – решение ОДУ методом Рунге-Кутты 4-го порядка точности.

dde23 – решение ОДУ с запаздыванием.

Пример:

1. Составить М-функцию, содержащую несколько независимых вложенных функций. Параметров вызова два – строковая переменная S, по которой определяется вызываемая подфункция, и массив коэффициентов K.

```
function M=MyMathSolutions(S,K)
%%
function M1=M1(x,y)
M1=1;
end

%%
function M2=M2(x,y)
M2=2;
end
%%

function M3=M3(x,y)
M3=3;
end

%%
function M4=M4(x,y)
M4=4;
end
%%
function M5=M5(x,y)
M5=1;
end
```

```

%%
switch (S)
    case '1'
        ;
    case '2'
        ;
    case '3'
        ;
    otherwise
        disp([S 'не найдено такого кода']);
end
M=1;
end

```

2. Вычислить производную в точке $x=(x_1, x_2)$ следующей функции $W(x)$:

$$W(x) = u(x, b) J_{2/3}(x_1 - a) - \Gamma(bx_2)(ax_1 + bx_2), \quad u(x, b) = x_1 + bx_2 + |x|^b, \quad a, b = \text{const}$$

Здесь J – функция Бесселя первого рода, Γ – гамма-функция, u – определенная пользователем функция

Прочитаем справку по используемым спецфункциям. Выпишем первую секцию (введем «защиту от дураков» в виде модуля, поскольку $\Gamma(x < 0)$ не определено, как и в нуле):

```

%%
function M1=M1(x,p)
    user_def_fun=@(x,b) x(1)+b*x(2)+norm(x).^b;
    M1=user_def_fun(x,p(2)).*besselj(2/3,x(1)-p(1))-
    gamma(abs(p(2)*x(2)))*(p*x');
end

```

В последней секции исправим (нижняя строка K – коэффициенты, верхняя – координаты точки):

```

case '1'
    M=M1(K(1,1:2),K(2,1:2));
....%M=1;

```

После вызова-теста `>> MyMathSolutions('1',[1 1;1 -1])` получим верный ответ 0. Пробный расчет дает `MyMathSolutions('1',[2 3;2 1])=-14`.

3. В пределах одного полотна построить два графика: один – контурный для $W(x)$, второй – график скоростей для градиента $\text{grad}W(x)$. Принять $a=1.5, b=2.3$, площадка – первая четверть круга радиуса π в центром в $(0,0)$. Шаг сетки для графика скоростей равен $\pi/4$.

Для простоты можно было бы использовать циклы, поскольку фактическая трехмерность матрицы данных мешает использовать `ezcontourf`. И все-

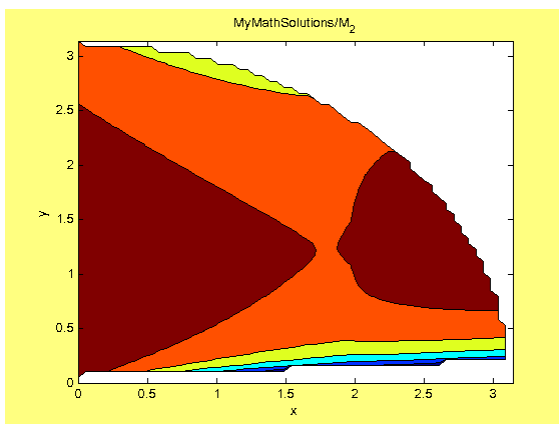
таким образом имеет место такая короткая формулировка (данные транспонируются):

```
%%
function M2=M2(x,y)
    if (x.^2+y.^2>pi^2) M2=NaN;
    else M2=real(M1([x.' y.'],K(2,1:2)));
    end
end
```

И в первом приближении нужно добавить в последнюю секцию:

```
case '2'
    ezcontourf(@M2,[0,pi,0,pi]);M=2;
```

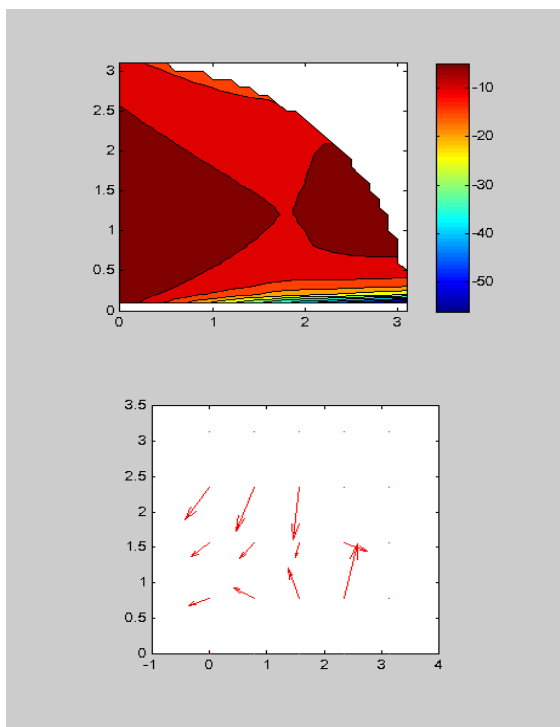
В качестве проверки запустим `>> MyMathSolutions('2',[2 3;2 1])` и получим график. Неровная белая полоса снизу получается по причине $\Gamma(+0)=-\infty$.



В ядре MATLAB отсутствует функция по вычислению градиента по заданному указателю функции, что вынуждает нас использовать матрицы. Поскольку градиент берется из данных интерполяции, то нужно сначала построить подробную сетку, взять градиент в ее узлах, затем полученные матрицы вновь интерполировать по нужным нам узлам; в противном случае точность не будет высокой.

```
%%
```

```
function M3=M3(x,y)
```



```
[X,Y]=meshgrid(0:0.1:pi);Z=X;
for i=1:length(X),
    for j=1:length(Y),
        Z(i,j)=real(M2(X(i,j),Y(i,j)));
    end
end
[Zx,Zy]=gradient(Z,0.05,0.05);
```

```
xbase=0:pi/4:pi;[xx,yy]=meshgrid(xbase);
Fx = griddata(X,Y,Zx,xx,yy);Fy =
griddata(X,Y,Zy,xx,yy);
scrsz = [1 1 0.4
0.8].*get(0,'ScreenSize');figure('Name',...
'Simulation Plot Window',
'NumberTitle',...
'off','Position',scrsz);
```

```
subplot(2,1,1);contourf(X,Y,Z);colorbar;axis square;
subplot(2,1,2);quiver(xx,yy,Fx,Fy,'Color','r');axis square;
```

```

M3=0;
end

```

В данном случае вызов `>> MyMathSolutions('3',[2 3;2 1])`. Неровность снизу контурного графика исчезает; вектора на втором графике показывают направление возрастания функции. Попробуйте сделать шаг вдвое меньше и рассчитать для других параметров.

4. Вычислить объем n-мерной сферы радиуса 1. Вывести график зависимости V(n).

Математически задача сводится к вычислению n-кратного интеграла, а в отношении программирования более чем уместна рекурсия, т.е. вызов функцией копии самой себя. Однако, MATLAB позволяет вычислять только одно-, дву- и трехкратные интегралы, а рекурсия в нем не предусмотрена, поэтому воспользуемся методом Монте-Карло:

$$V(n) = \int \int \dots \int_{\sum_k x_k^2 \leq 1} dx_1 \dots dx_n \quad V(2) = \pi \quad V(3) = \frac{4}{3}\pi$$

$$V(n) = 2^n \int \int \int_{R_+^n} f(x) \rho(x) dx = \overline{f(\xi)} \quad \xi - \text{случайная многомерная величина с плотностью } \rho(x)$$

$$\rho(x) \equiv 1 \quad f(x) = \begin{cases} 0, & (x, x) > 1 \\ \mu(x) \equiv 1, & (x, x) \leq 1 \end{cases}$$

Заметим, что метод Монте-Карло обладает плохой сходимостью, пропорциональной корню из числа испытаний $\sqrt{N}$.

Соответствующие фрагменты выглядят так (была проведена нормировка на число π):

```

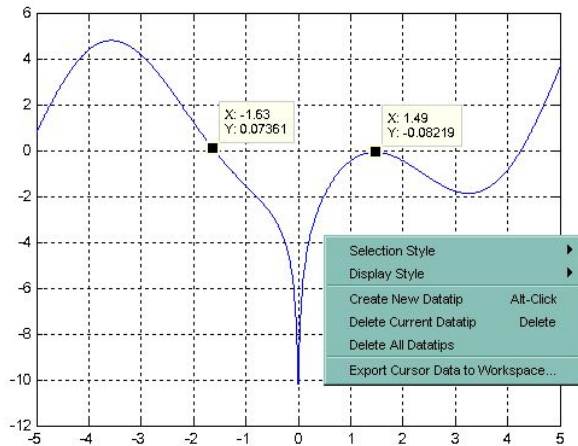
function M4=MySphereVolume(x,p,flag)
% x- вещественный аргумент, p - целые параметры, например, размер-
ность
% при начальном вызове x равен радиусу, а p - скаляр
MassOfPoint=@(x) 1;
function Region=Region(x)
    if (sum(x.*x)>1) Region=0; else Region=MassOfPoint(x); end
end
N=2;summa=1;medsumma=0;medsumma1=0.5;eps=1e-8;
while ((abs(medsumma-medsumma1)>eps)&&(N*sqrt(eps)<1))
    ksi=rand(1,p);summa=summa+Region(ksi);N=N+1;
    medsumma=medsumma1;medsumma1=summa/N;
    %если захотим увидеть ход процесса, каждый сотый шаг
    if (flag&&(~mod(N,100))) ['процессинг: ' num2str(medsumma) ...
        ' ' num2str(medsumma1)], end;
end
%вторым аргументом идет погрешность расчета
M4=[medsumma1*((2*x).^p) 1/sqrt(N)];
end

```

```
....
case '4'
    Z=[];
    for n=1:10, res=MySphereVolume(1,n,false), Z=[Z res(1)/pi]; end;
    plot(Z);
```

5. а) Вычислить все корни уравнения $8x^3 - 6x + 1 = 0$.
 б) Вычислить все корни трансцендентного уравнения на отрезке $[-5; 5]$: $x \cdot \cos(x) + \ln(x^2) = A$ (для некоторых A).

Примечание: излагается решение без M-файла



```
>> format long; roots([8 0 -6 1])
```

```
ans =
```

```
-0.93969262078591
0.76604444311898
0.17364817766693
```

Интересующий нас раздел помощи MATLAB->Mathematics->Function Functions->Minimizing Functions and Finding Zeros. Чтобы «почувствовать» задачу «на лету» построим график:

```
A=1; transcen=@(x) x.*cos(x)+log(x.^2)-A; x=-5:0.01:5; plot(x,transcen(x)),
grid on
```

Можно попробовать вручную, мышкой, найти корень. Для этого в графическом редакторе выбрать пиктограмму , в контекстном меню на кривой выбрать Selection Style – Mouse Position, Display Style – DataTip, выбрать стартовую точку, и, не отпуская кнопку мыши, провести по кривой. График позволяет нам выделить участки перемены знака. Найдём третий по возрастанию корень:

```
>> fzero(transcen,[4 5])
```

```
ans =
```

```
4.25053115278136
```

Однако график не позволяет нам определить, является ли второй корень двойным, или мы имеем два близких корня. Здесь, несмотря на многочисленность параметров (см. optimset) fzero, нам придется использовать функцию оптимизации fminbnd:

```
>>
```

```
transcen1=@(x)
```

```
transcen(x);[x,fx]=fminbnd(transcen1,1,2,optimset('Display','iter'))
```

Func-count	x	f(x)	Procedure
1	1.38197	0.0935767	initial
2	1.61803	0.11398	golden
3	1.23607	0.170065	golden

4	1.47297	0.0815729	parabolic
5	1.47129	0.0815597	parabolic
6	1.46961	0.0815551	parabolic
7	1.46955	0.0815551	parabolic
8	1.46952	0.0815551	parabolic

Optimization terminated:

the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-004

```
x =
    1.46955240774355
fx =
    0.08155506454236
```

Если бы уравнение действительно имело корень, то ордината fx была бы отрицательной. Таким образом, впечатление от графика (о наличии корня) ложное.

6. а) Графически решить систему уравнений (a,b – параметры):

$$\begin{cases} x^2 - ax + y^2 = b^2 \\ x + by = a \end{cases}$$

б) Найти все решения системы:

$$\begin{cases} ax + by = z \\ ax^2 + by^2 = z^2 \\ (x + a)(y + b)(z + 1) = 1 \end{cases}$$

По пункту (а) – мы используем контурные графики для линий уровня, соответствующих нулю. Добавим в наш М-файл секцию:

```
function M5=M5(a,b)
    %Решение графически системы двух уравнений
    %Сетка адаптирована к параметрам a,b
    f1=@(x,y) x.^2+y.^2-a*x-b^2;f2=@(x,y) x+b*y-1;
    x=fix(-b+a/2-1):b/100:fix(b+a/2+1);y=fix(-b-1):0.001:fix(b+1);
    [X,Y]=meshgrid(x,y);Z1=f1(X,Y);Z2=f2(X,Y);
    figure(1);hold on;contour(X,Y,Z1,[0 0],'-r','DisplayName','1');
    contour(X,Y,Z2,[0 0],'-g','DisplayName','2');
    axis equal; grid on; legend show; hold off;
    M5=0;
end
...
case '5'
    M5(K(1,1),K(2,1));
```

По пункту (б) – задача сводится к поиску нулевого экстремума. Добавим еще секцию:

```
%%
function M6=M6(a,b)
```

```

%Решение системы нескольких уравнений
%Область поиска локальных экстремумов разбивается на
%несколько случайным числом
f1=@(x) a*x(1)+ b*x(2)-x(3);
f2=@(x) a*x(1).^2+b*x(2).^2-x(3).^2;
f3=@(x) (x(1)+a).*(x(2)+b).*(x(3)+1)-1;
%Критерий корня - экстремум суммы квадратов
OurFun=@(x) sqrt(f1(x).^2+f2(x).^2+f3(x).^2);
Res=[];
for n=1:100,
    display(['Номер итерации: ' int2str(n)]), pause(0.05);
    opt=optimset('Display','off');p=10*rand(1,3)-5;
    [x,fx]=fminsearch(OurFun,p,opt);
    if (abs(fx)<1e-4) Res=[Res;x fx]; end;
end;
%убрать повторы
if (size(Res,1)==0) display('Решений нет'),
else
    Res=Res';Rest=Res(:,1);%keyboard;
    for p=Res,
        flag=true;
        for q=Rest,
            %keyboard;
            if (norm(p-q)<1e-3) flag=false; end;
        end;
        if (flag) Rest=[Rest p]; end;
    end;
end;
%проверка правых частей, 3 последних строки
Rest=[Rest(1:3,:);Rest(1:3,:)];n=0;
for p=Rest, p(4)=f1(p(1:3));p(5)=f2(p(1:3));p(6)=f3(p(1:3));
    n=n+1;Rest(:,n)=p;
    display([int2str(n) '-Аналитическое решение #(x,y,z): '
num2str(p(1:3))]),
    %display(['Погрешности от уравнений: ' num2str(p(4:6))]),
end;
M6=Rest;
End
...
case '6'
    M=M6(K(1,1),K(2,1));
    При a=1 и b=2 вызов W=MyMathSolutions('6',K) дает всего 5 решений
    (решения №3,4 можно получить аналитически – при y=0 x=z=±2-1/2-1):
    ...
    Номер итерации: 100

```

1-Аналитическое решение $\#(x,y,z)$: -0.93181 1.8636 2.7954
 2-Аналитическое решение $\#(x,y,z)$: 0.16215 -0.32434 -0.4865
 3-Аналитическое решение $\#(x,y,z)$: -0.29292 1.8644e-005 -0.29288
 4-Аналитическое решение $\#(x,y,z)$: -1.7071 -2.6909e-006 -1.7071
 5-Аналитическое решение $\#(x,y,z)$: 1.103 -2.2059 -3.3089

7. Решить систему обыкновенных дифференциальных уравнений и начертить траекторию движения частицы (параметр a изменяется от -1 до 1 с шагом 0.5):

$$\begin{cases} dx/dt = ay - z \cdot \cos t \\ dy/dt = az - x \cdot \sin t \\ dz/dt = ax - yt \end{cases} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{t=0} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad 0 < t < \pi$$

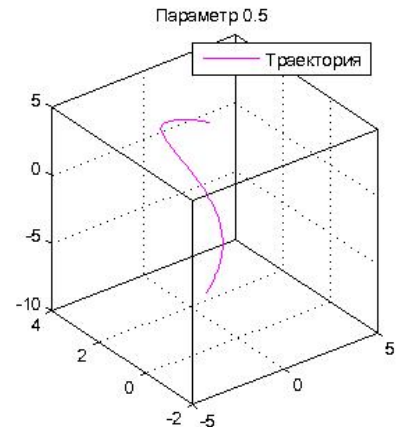
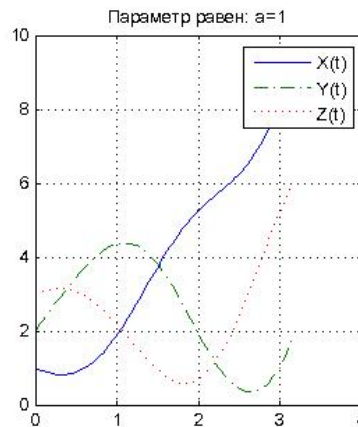
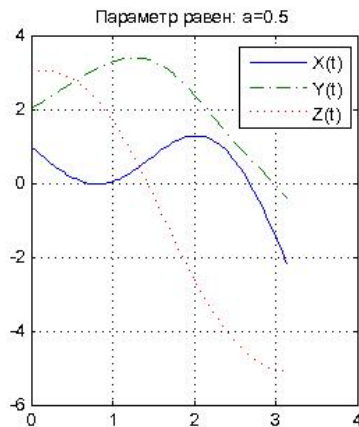
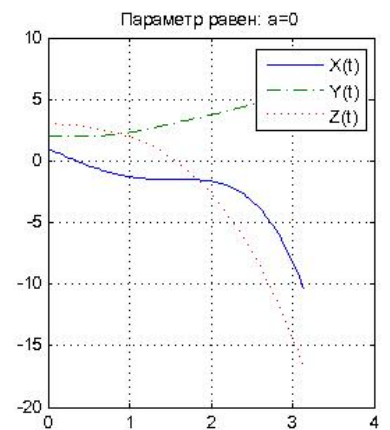
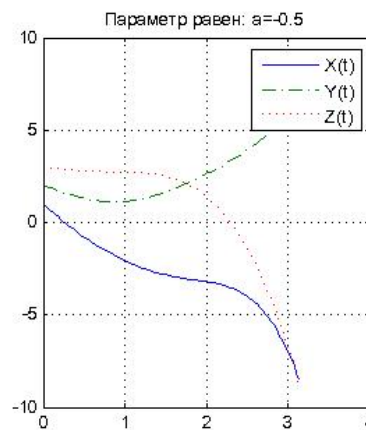
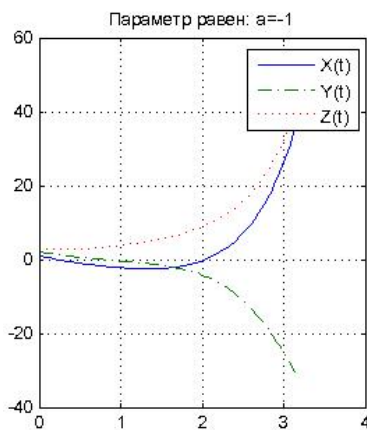
Особенность решения ОДУ состоит в том, что выходные вектора многокомпонентны, и точки расположены неравномерно (последняя проблема решается, однако, командой `deval`). По параметру a мы создаем семейство решений и записываем их в 3-х-мерную матрицу. Входная матрица K равна $[0 \ 1; 3 \ 2]$.

```
%%
function M7=M7(a,b)
    function RateX=RateX(x,y,z,t)
        RateX=a*y-z.*cos(t);
    end
    function RateY=RateY(x,y,z,t)
        RateY=a*z-x.*sin(t);
    end
    function RateZ=RateZ(x,y,z,t)
        RateZ=a*x-y.*t;
    end
    function Rate=Rate(time,arg)
        Rate=[RateX(arg(1),arg(2),arg(3),time);...
            RateY(arg(1),arg(2),arg(3),time); RateZ(arg(1),arg(2),arg(3),time)];
    end
    Data=[];Ti=linspace(0,pi,100)';
    for a=-1:0.5:1,
        [T,Y]=ode45(@Rate,[0 pi],b);
        %Шаг интегрирования нельзя задавать постоянным.
        %Интерполируем данные
        Y1 = interp1(T,Y(:,1),Ti,'spline'); Y2 = interp1(T,Y(:,2),Ti,'spline');
        Y3 = interp1(T,Y(:,3),Ti,'spline'); Data=cat(3,Data,[Y1 Y2 Y3]);
    end;
    %Данные подсчитаны. Приступим к построению графиков на одном
    полотне
    scrsz = get(0,'ScreenSize');
    figure('Name','Траектории','Position',[0.1*scrsz(3:4) 0.8*scrsz(3:4)]);
    for j=1:5,
        subplot(2,3,j);
```

```

plot(Ti,Data(:,1,j),'-',Ti,Data(:,2,j),'-', Ti,Data(:,3,j),'');
grid on;legend('X(t)','Y(t)','Z(t)');legend('show');
title(gca,['Параметр равен: a=' num2str(j/2-1.5)]);
end;
% На лишнем месте построим 3D-кривую для a=0.5
subplot(2,3,6);h=plot3(Data(:,1,4),Data(:,2,4),Data(:,3,4));
set(h,'Color','m'); axis square;grid on;box('on');title('Параметр 0.5');
legend({'Траектория'},'Orientation','horizontal','location','NorthEast');
M7=1;
end
...
case '7'
M=M7(K(1,1),[K(1,2);K(2,2);K(2,1)]);

```



Задание:

1. Попробуйте п.4. Примера пересчитать с большей точностью, а результаты пересчета отразить на одном графике (для этого немного измените второй фрагмент кода). Поясните полученные результаты. Улучшите код и постройте единый график, адекватный истинным результатам (отобразить их отдельной кривой). Они определяются аналитическим выражением через гамма-функцию: $V(n) = \pi^{n/2} / \Gamma(n/2 + 1)$.

2. Создать М-файл MyMath.m, вычисляющий значение функции $P(x)=1+2x+3x^2+4x^3+5x^5$, а также дающий его корни. На отрезке $[-1;1]$ вычислить коэффициенты быстрого Фурье-разложения $P(x)$ (см. fft) кол-ве 10 шт. Сравнить их с коэффициентами классического Фурье-разложения c_n .

Замечание: Для функции $f(x)$ коэффициенты ряда Фурье задаются формулами (для отрезка $[a;b]$):

$$a_n = \frac{2}{b-a} \int_a^b f(x) \cos\left(\frac{2\pi}{b-a} kx\right) dx, \quad b_n = \frac{2}{b-a} \int_a^b f(x) \sin\left(\frac{2\pi}{b-a} kx\right) dx, \quad c_n = 0.5 \sqrt{a_n^2 + b_n^2}$$

3. В рамках того же М-файла реализовать вычисление лапласиана от функции:

$$U_1(x, y, z) = x^2 yz - (1 + x^2) \frac{\partial}{\partial z} (a \cdot xz^2)$$

$$U_2(x, y, z) = a \cdot P(xy) + \operatorname{erf}(x - y + bz) - b * \frac{\partial}{\partial z} (z L_5^{(0)}(\sin(zx + ay)))$$

Здесь erf – функция ошибок, L_5 – полином Лежандра 5-й степени, a,b-параметры.

Указание: вычислить лапласиан для U_1 аналитически, но не программировать ее. Сверить результаты аналитики и программирования. Затем адаптировать процедуру для U_2 . Раздел справки (а также математические определения) см. в MATLAB-> Functions -- By Category-> Mathematics->Data Analysis-> Finite Differences and Integration, а также Mathematics-> Specialized Math.

4. В рамках того же М-файла реализовать построение 4-х графиков на одном полотне: 1,2,3D-мерного и одного параметрического. На базе данных от функции $U_{1,2}(x,y,z)$.

5. Графическим путем (через пересечение двух кривых) с точностью до 4-го знака найти все корни уравнения $0.1P(x)=U_1(x,1,1)$ – при $a=2$. На отрезке $[-1;2]$ аппроксимировать $U_1(x,1,1)$ полиномом 5-й степени $Q(x)$. Найти корни уравнения $0.1P(x)=Q(x)$.

6. Пользуясь результатами п.6 Примера, найти все локальные экстремумы функции $U_2(x,y,z)$ при наборах: $(a,b)=(1,1),(1,0),(0,1),(-1,0)$.

7. а) Решить следующую систему на отрезке $[-5;5]$ при $a=\pi$, $b=2$, $c=-3$:

$$\begin{cases} dx/dt = \cos(ax + by + cz) \\ dy/dt = \cos(ay + bz + cx) \\ dz/dt = \cos(az + bx + cy) \end{cases} \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{t=0} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad 0 < t < 2\pi$$

б) НАЙТИ ТОЧКИ ПУАНКАРЕ СЕЧЕНИЯ ТРАЕКТОРИЕЙ ПЛОСКОСТИ $Z=0.3$. СОЕДИНИТЬ ЭТИ ТОЧКИ С СОСЕДНИМИ С ПОМОЩЬЮ ФУНКЦИИ GPLOT.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Гультяев, А. П. Визуальное моделирование в среде MATLAB: учеб. курс/ А. П. Гультяев.— СПб.: Питер, 2000.— 432 с.
2. Дьяконов, В. П. MATLAB: учеб. курс/ В.П. Дьяконов. — СПб.: Питер, 2001.— 560 с.
3. Дьяконов, В.П. Simulink 4. Специальный справочник / В.П. Дьяконов.— СПб.: Питер, 2002.— 528 с.
4. Кетков, Ю.Л. MATLAB 6.x . Программирование численных методов / Ю Л Кетков, А. Ю Кетков, М.М. Шульц.— СПб.: БХВ-Петербург, 2004.—672 с.
5. Конев, В.Ю. Основные функции пакета MATLAB: учеб. пособие/ В.Ю. Конев, Л.А. Мироновский.— 2-ое издание.— СПб.: ГААПСПб., 1994.—76 л.
6. Потемкин, В. Г. Система MATLAB: справочное пособие/ В. Г. Потемкин.— М.: ДИАЛОГ-МИФИ, 1997.— 350 с.