Практическое занятие № 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.

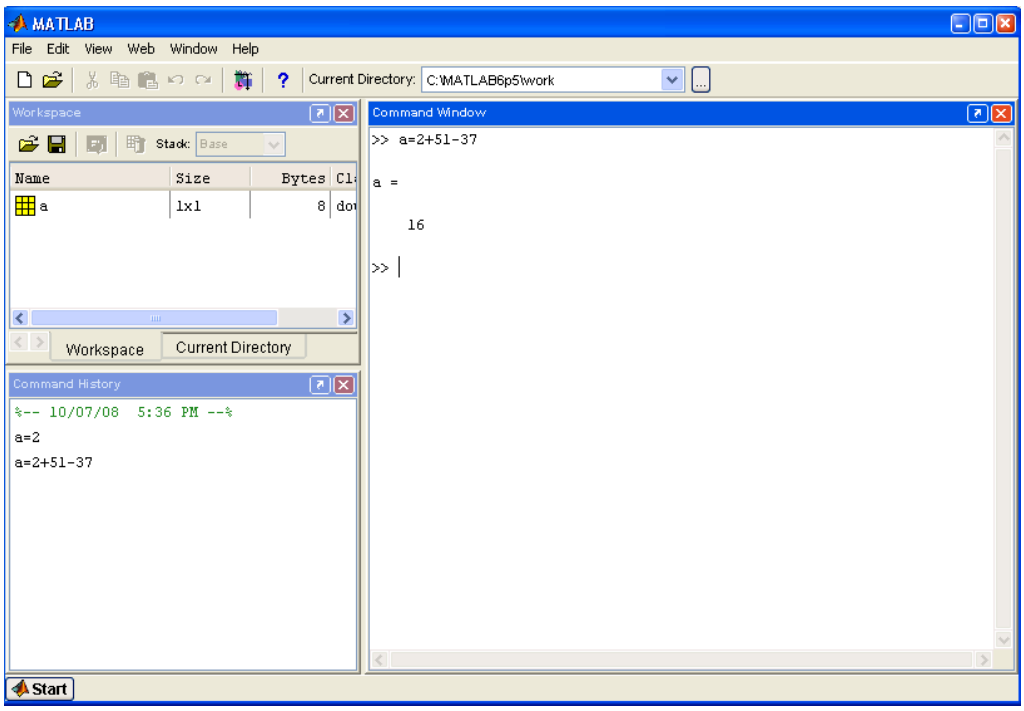

Рисунок 1.1 – Интерфейс программы MATLAB

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*inf, inf-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 (.*, .^, ./, .\)
…	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.

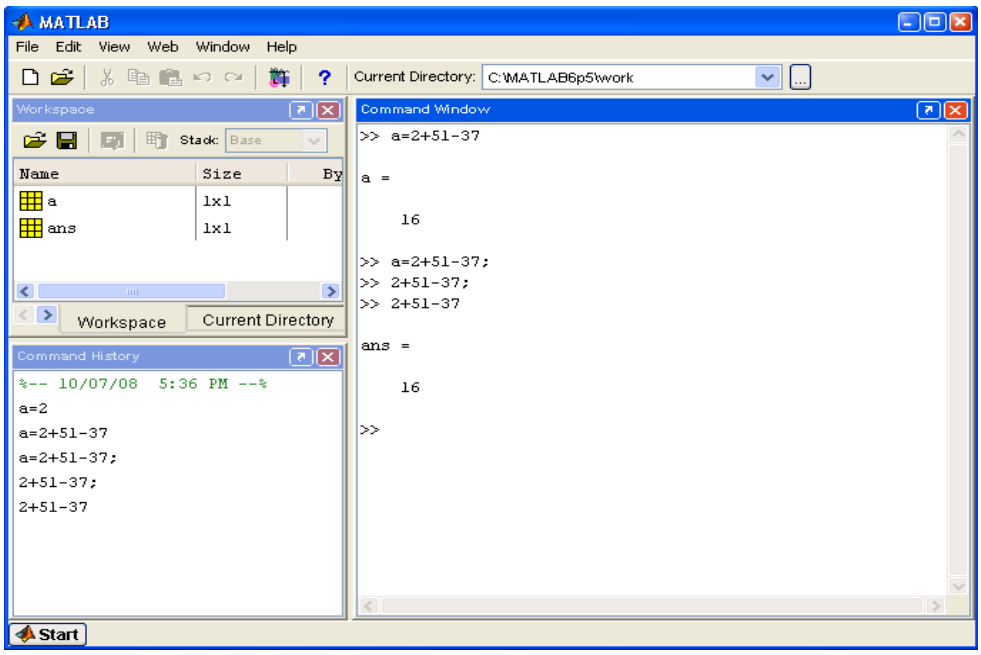

Рисунок 1.2 – Демонстрация выполнения команды присваивания

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 MAT-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	$y = \sin\frac{a-x}{c} + 10^4\sqrt[3]{\frac{d-kx}{2b}} + \frac{\cos kx^2}{tg3} - \frac{bc}{ax}$	$a = -1.3; b = 0.01;$ $c = 0.75; x = 2.32; k = 8.$ $d = 1.25; b = 0.75; n = 4;$
2	$y = -\frac{(x-d)(x^2+b^2)}{\sqrt[3]{x^2+b^2-cd}} + 10^{-3}tgkn - \frac{\cos kx}{\sin 5}.$	$c = 2.2; x = 0.32; k = 2.$ $i = 5; b = 2.35;$
3	$y = tgik + 10^3 e^{-5} + \sqrt[3]{\frac{10^2\lvert xk\rvert}{(a+b)^2}} - \frac{ax^3-b}{(a+b)^2}.$	$a = 25.2; x = 0.1; k = -2.$ $a = 1.3; b = 2.42;$
4	$y = \frac{\lvert a^2-b^2\rvert}{\sin kx} + 10^4\sqrt[5]{\lvert\sin kx - bc\rvert} - \frac{k^2+tg3k}{e^{kx}}.$	$c = 0.83; x = 1.5;$ $k = 2.$ $c = 1.52;$
5	$y = \frac{\sqrt[3]{\ln x + a^2}}{0.47x^2} - \left\lvert 0.47x^2 - \frac{10^4}{7}\cos^2 k\right\rvert - \frac{c}{x}$	$a = -2.4; x = 0.29; k = 3.$

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.

Практическое занятие № 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*×*n*
zeros(m,n)	Нулевая матрица размерности *m*×*n*
ones(m,n)	Матрица, состоящая из одних единиц размерности *m*×*n*
rand(m,n)	Возвращает матрицу случайных чисел равномерно распределенных в диапазоне от 0 до 1, размерность *m*×*n*
randn(m, n)	Возвращает матрицу размерности *m*×*n*, состоящих из случайных чисел, имеющих гаусовское 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 полных 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

№	Задание
1	*M=v*w+m+mr*me*
2	*M=m+mr*me*
3	*M=(v/m)*(mr+me)*
4	*M=w*v+mr*me*
5	*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 служит команды zeros, ones, rand, eye?
5. Как определяется число строк и столбцов матрицы?
6. Какие операции служат для определения минимального и максимального элемента матрицы?

Практическое занятие № 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**)**.

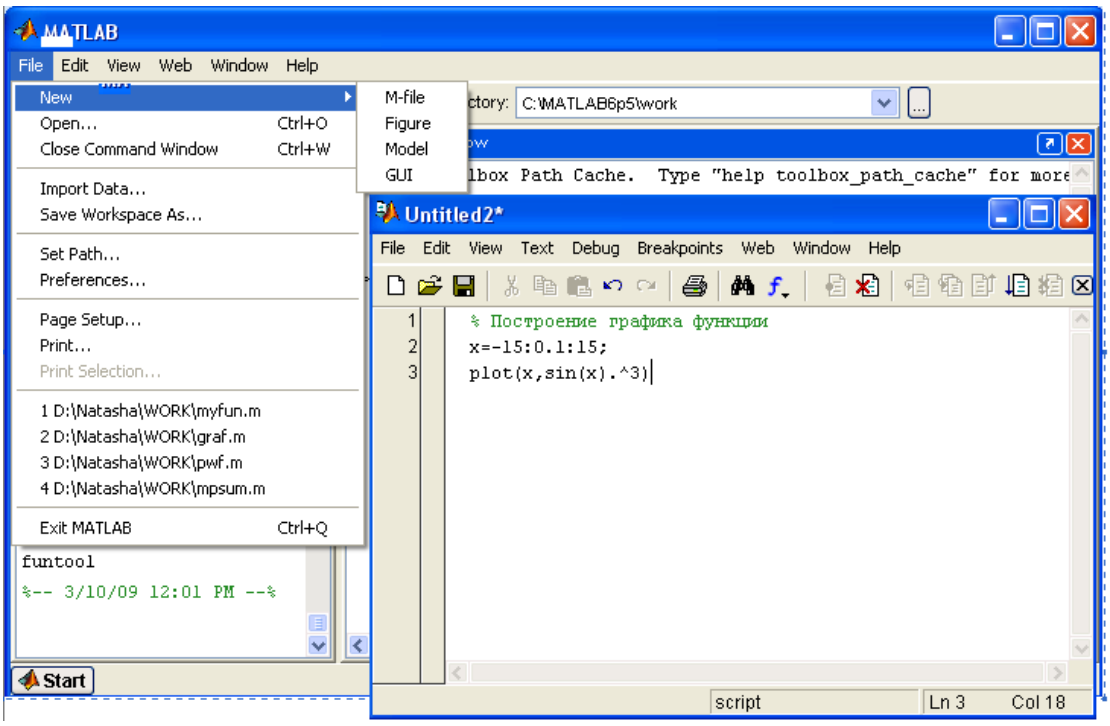

Рисунок 3.1– Многооконный редактор–отладчик

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
%Дополнительный 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):

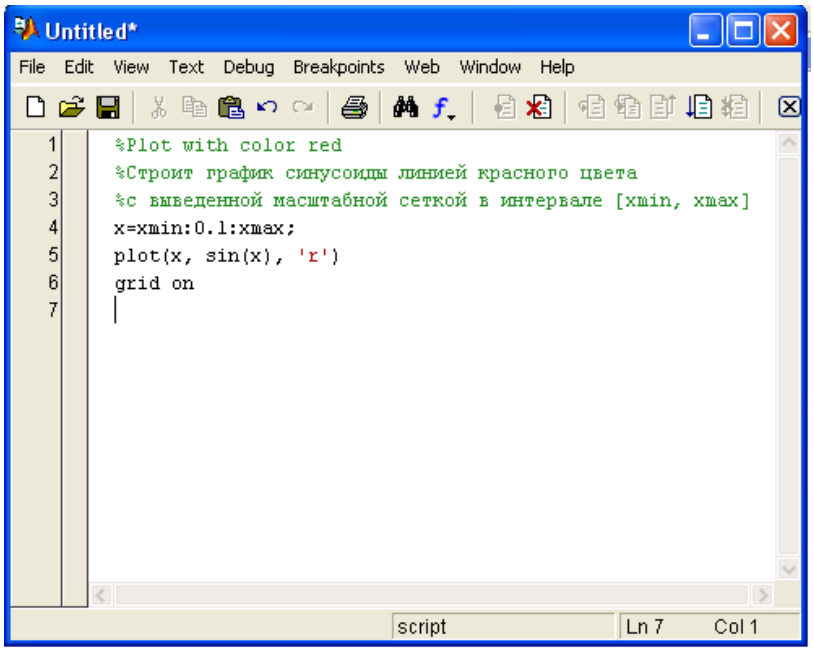

Рисунок 3.2 – Создание файла-сценария в MATLAB

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 [varl,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 [varl, var2] = f_name (*Список_параметров*). После его применения переменные выхода varl, 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	*x* = input(‘<приглашение>’)	Для ввода данных с клавиатуры
DISP	disp (<переменная или текст в апострофах>)	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).

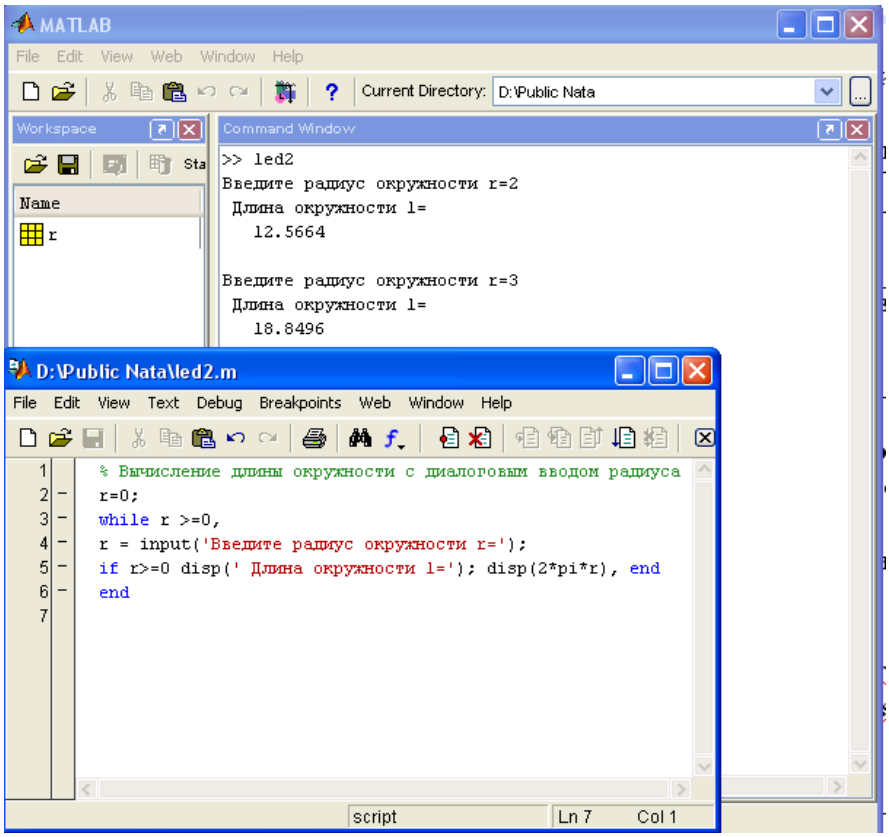

Рисунок 3.3 – Пример диалоговой программы

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:

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

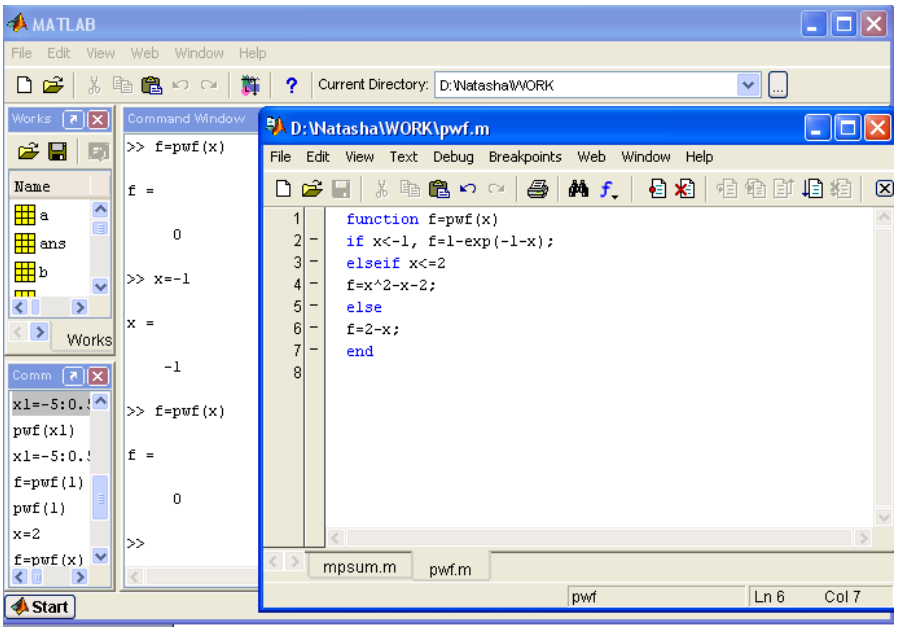

Рисунок 3.4 – Листинг программы для вычисления значения функции

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).

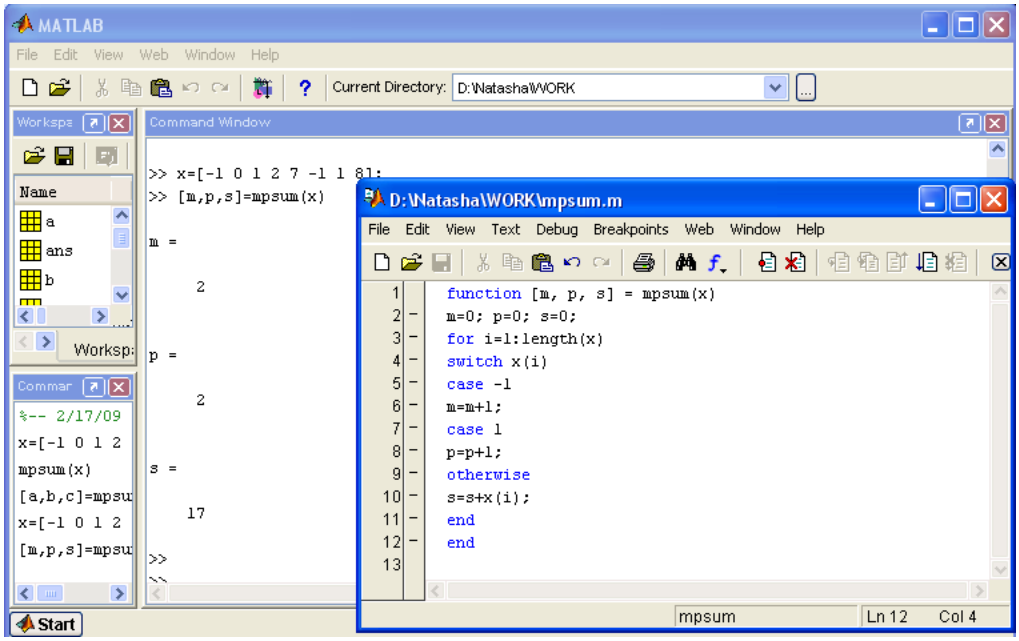

Рисунок 3.5 – Листинг программы

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	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.

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	$A_{3\times3}$	$B_{3\times3}, b_{ij} = \begin{cases} a_{ij}, i < j \\ a_{ji}^2, i \geq j \end{cases}$	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	$B_3, b_i = \sin(i^2), i = 1\ldots3$ $B_{3\times3}, b_{ij} = \sin(i) * \sin(j)$	Сформировать массив C – сумму элементов массивов A и B. Найти максимальное значение массивов A, B, C.
3	$A_{3\times3}$	$i = 1\ldots3, j = 1\ldots3.$	Определить минимальные элементы в матрицах A и B (mA и mB). Вычислить $C=A*B*mA*mB$.
4	$A_{3\times3}$	$B_{3\times3}, b_{ij} = \begin{cases} 2*i+3*j, i=j, \\ 5*i+2*j, \textit{иначе.} \end{cases}$	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.

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?

Практическое занятие № 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
:	Пурктирная (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*,...) 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 удается. 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] = meshgrid(x, y, z) — возвращает трехмерные массивы, используемые для вычисления функций трех переменных и построения трехмерных графиков;

– [X,Y] = **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*,...)— 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)
```

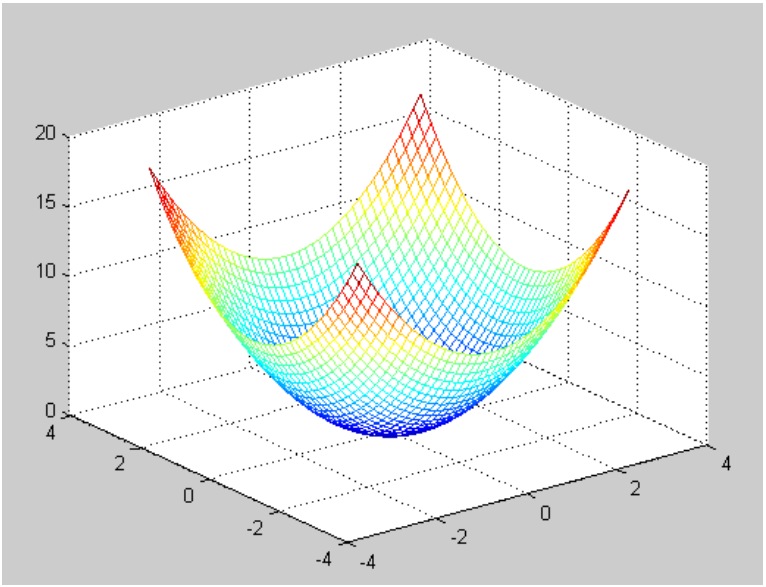

Рисунок 4.1 – График поверхности, созданный командой mesh(X,Y,Z)

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').

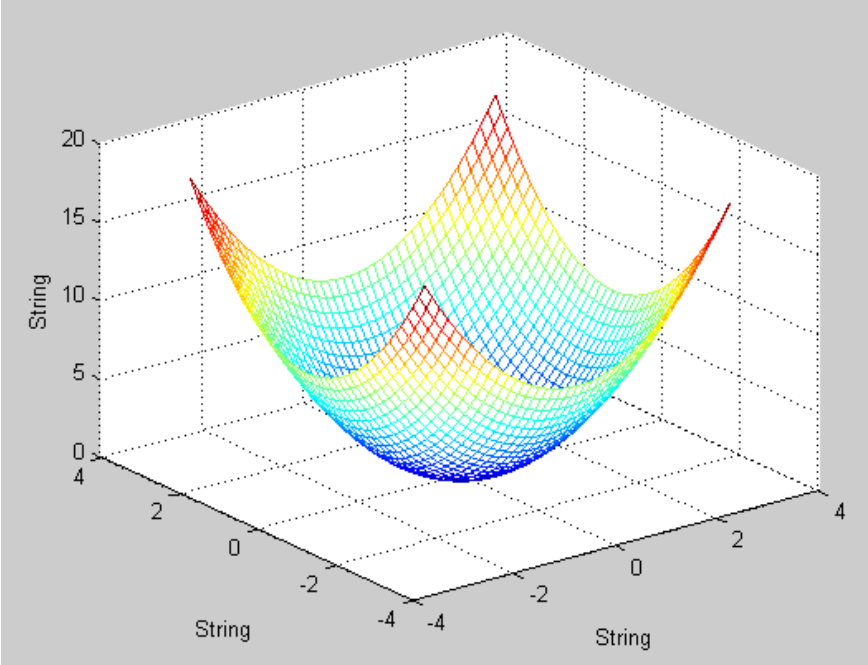

Рисунок 4.2 – Установка надписей с использованием команды: xlabel('String'), ylabel('String'), zlabel('String')

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(stringl, string2,…, strings) — добавляет к текущему графику легенду в виде строк, указанных в списке параметров;

```
>> legend('график')
```

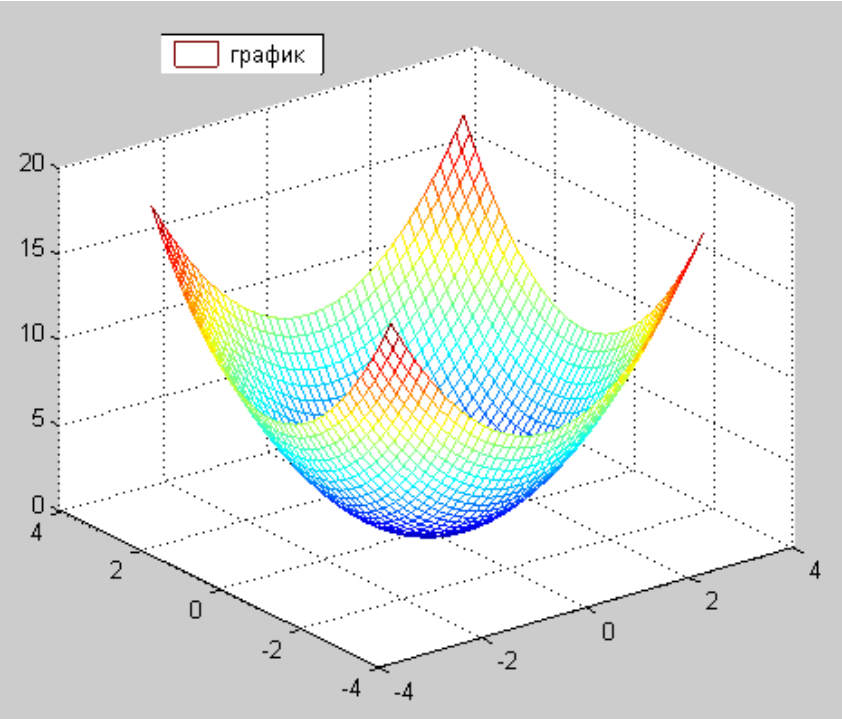

Рисунок 4.3 – График с пояснениями

l**egend** (Pos) — помещает легенду в точно определенное место, специфицированное параметром Pos:

Pos=0 — лучшее место, выбираемое автоматически;

Pos=l — верхний правый угол;

Pos=2 — верхний левый угол;

Pos=3 — нижний левый угол;

Pos=4 — нижний правый угол;

Pos=-l — 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*×*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);
```

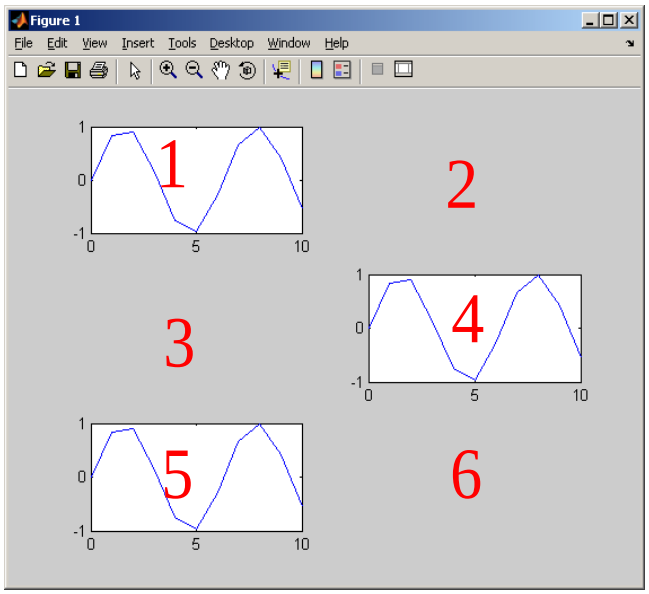

Рисунок 4.4 – Работа функции subplot

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 *f1*, *f2*, *f3* 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	*f*1	*f*2	*f*3	*f*4
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	$(\cos(r)/r)^2$

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

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

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 системах 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?

Практическое занятие № 5

РЕШЕНИЕ ТИПОВЫХ ЗАДАЧ АЛГЕБРЫ И АНАЛИЗА

Цель занятия: ознакомиться с возможностями системы MATLAB в решении типовых задача 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

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

где A – заданная квадратная матрица размером NxN,

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

Операция \ называется левым делением матриц и, будучи примененная к матрицам A и B в виде $A \backslash B$, примерно эквивалентна вычислению выражения **inv**(A)*B. Здесь под **inv**(A) понимается вычисление матрицы, обратной к матрице A. Операцию / называют правым делением матрицы. Выражение A/B примерно соответствует вычислению выражения B***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 pi. 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 xmin =**fmin**(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**: xmins = **fmins**(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 xmin 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**: Integ = **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 $\int_0^1 \int_1^2 (x\sin(y) + y\sin(x))dxdy$.

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**:

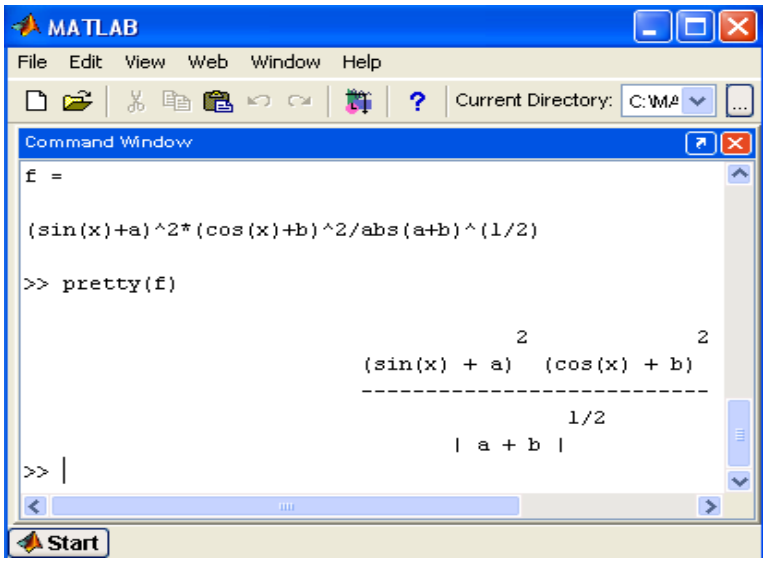

Рисунок 5.1– Демонстрация работы функции **pretty**

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

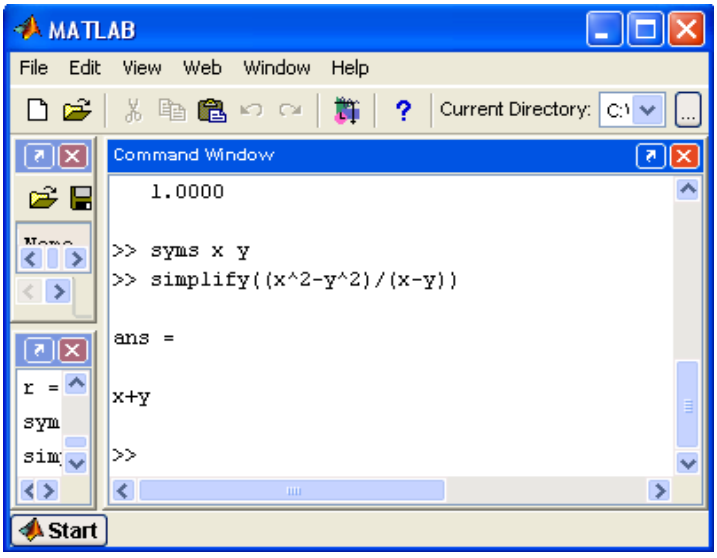

Рисунок 5.2– Демонстрация работы функции **simplify**

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

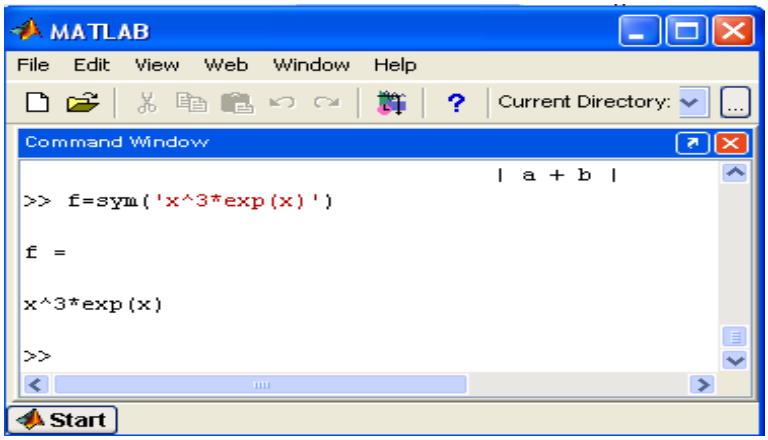

Рисунок 5.3 – Демонстрация создания символьной переменной без предварительного объявления

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:

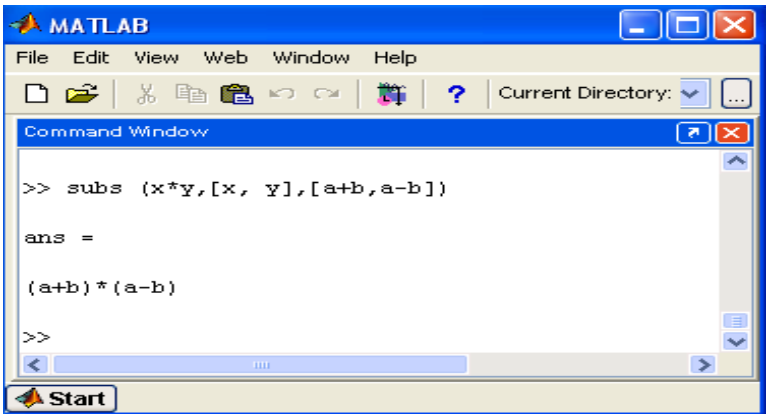

Рисунок 5.4– Демонстрация работы функции **subs**

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).

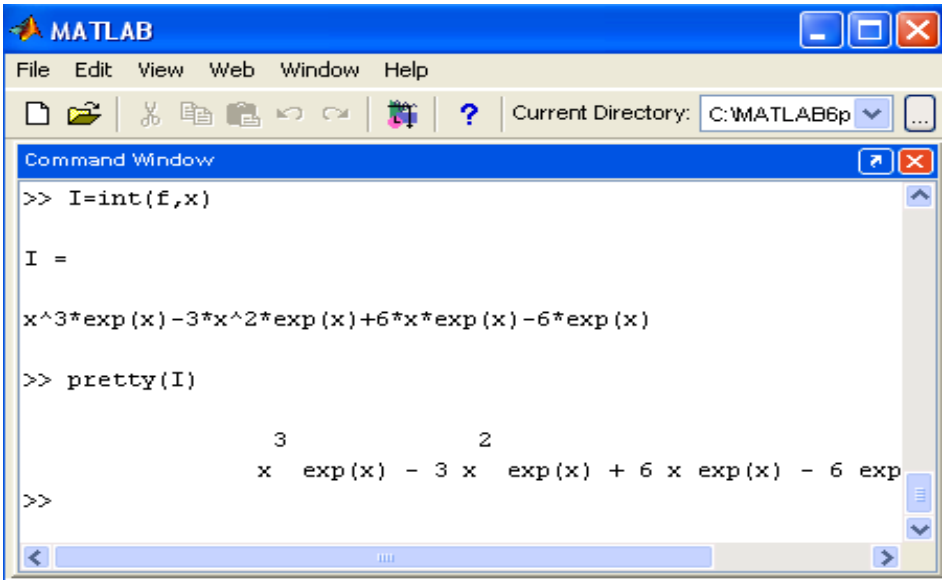

Рисунок 5.5– Демонстрация работы функции **int**

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

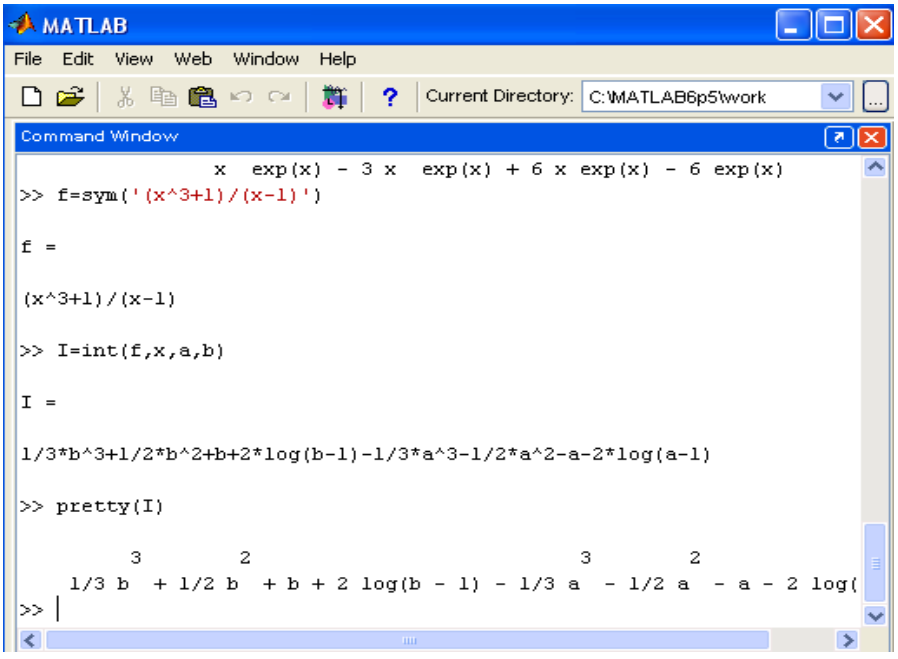

Рисунок 5.6– Демонстрация работы функции **int** для нахождения определенного интеграла

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.

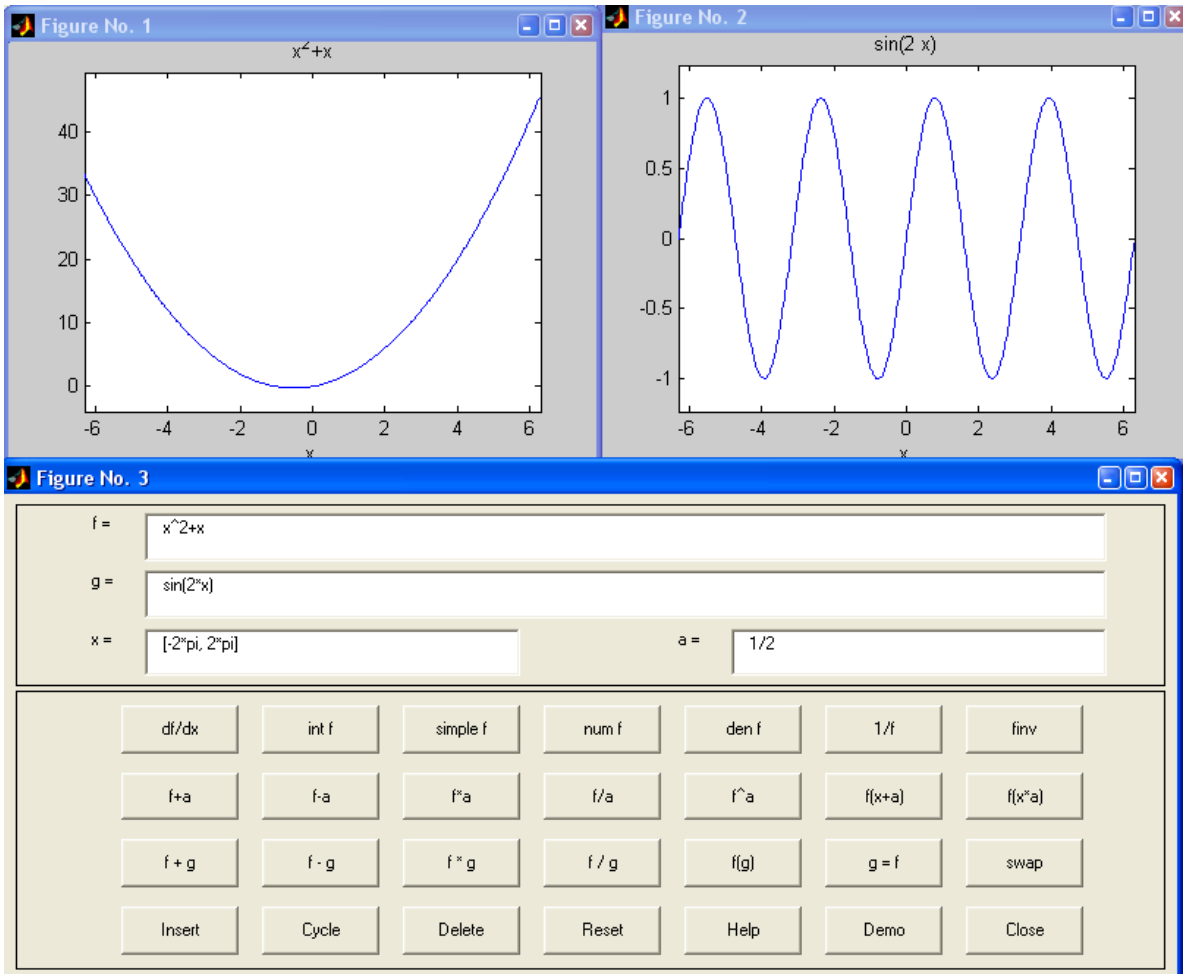

Рисунок 5.7 – Интерфейс приложения **funtool**

5.2 Порядок выполнения

1. Составить и отладить программы для нахождения корней уравнения *f1*(*x*) = 0 и *f2*(*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

№ варианта	$f_1(x)$ **– полином 3-ей степени с коэффициентами *a***				$f_2(x)$
1	0	-1	4	-1	$0.2e^x - 20$
2	0	2	-2	-15	$40 \cdot \lvert\cos(x)\rvert$
3	0	1	4	-1	$10 \cdot \log(x + 5.5)$
4	0	9	-8	-70	$100 \cdot \lvert\sin(x)\rvert$
5	0	-4	4	50	$70 \cdot \cos(x)$

Таблица 5.2 – Варианты функций для нахождения значения интеграла

№ варианта	**Функция**	**Интервал интегрирования**	
		начало интервала	**конец интервала**
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

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.

ЛИТЕРАТУРА

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.