Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра систем управления

**Л. В. Русак, С. В. Снисаренко, Н. А. Стасевич**

# *ОСНОВЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ В ПАКЕТЕ MATLAB*

Методическое пособие
к практическим занятиям
по дисциплине «Учебная практика»
для студентов специальности
«Информационные технологии и управление в технических системах»
дневной формы обучения

Минск БГУИР 2010

УДК 004.42(076)
ББК 32.973.26-018.2я73
Р88

Рецензент :
кандидат физико-математических наук,
доцент кафедры вычислительных методов и программирования БГУИР
А. В. Щербаков



**Русак, Л. В.**

Р88 Основы математического моделирования в пакете MATLAB : метод. пособие к практ. занятиям по дисц. «Учебная практика» для студ. спец. «Информационные технологии и управление в технических системах» днев. формы обуч. / Л. В. Русак, С. В. Снисаренко, Н. А. Стасевич. – Минск : БГУИР, 2010. – 64 с.: ил.

ISBN 978-985-488-485-1

Методическое пособие включает семь практических занятий для выполнения на ЭВМ с использованием пакета MATLAB v6.5, содержащих краткие теоретические сведения, варианты заданий и вопросы для самопроверки.

Приводятся основные сведения о базовых возможностях пакета MATLAB и его расширения – пакета моделирования динамических систем Simulink. Подробно описаны особенности организации графического интерфейса, объекты MATLAB и операции над ними, процедуры создания, редактирования и отладки программ, работа с двумерной графикой, а также блоки основной библиотеки Simulink и рекомендации по настройке их параметров.

**УДК 004.42 (076)**
**ББК 32.973.26-018.2я73**

**ISBN 978-985-488-485-1**

СОДЕРЖАНИЕ

Практическое занятие №1

# ОСНОВЫ РАБОТЫ С MATLAB

*Цель занятия* – изучение пользовательского интерфейса системы MATLAB и основ работы с ней в режиме прямых вычислений.

## 1.1 Основные теоретические сведения

Первоначально система MATLAB разрабатывалась как диалоговая среда для матричных вычислений (MATrix LABoratory). Впоследствии 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 (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.

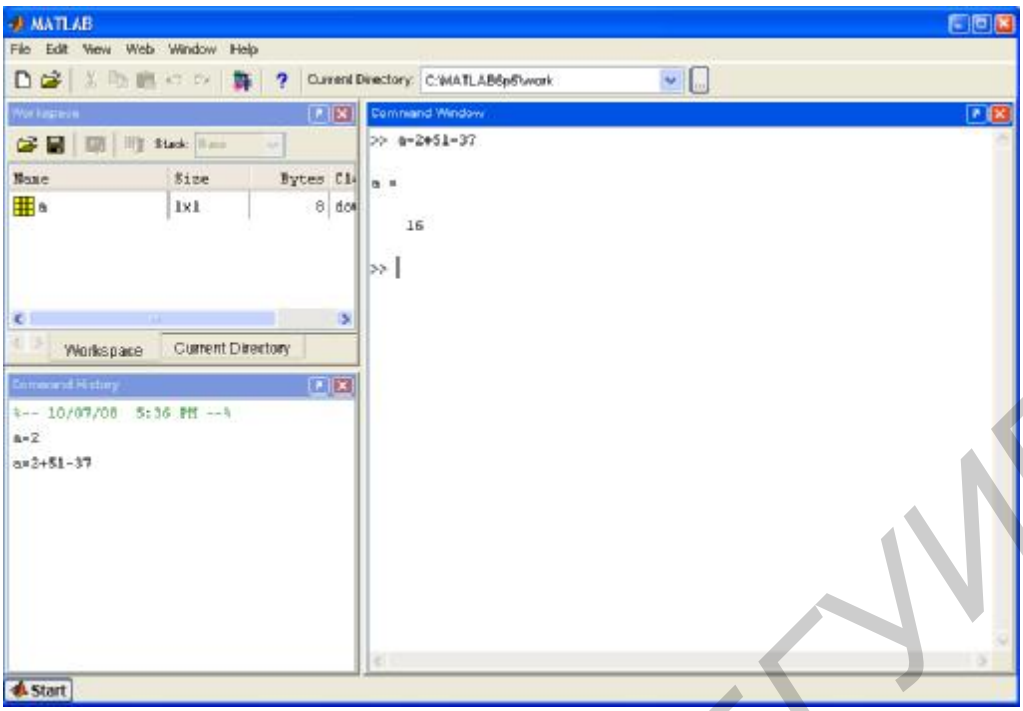



Рисунок 1.1 – Интерфейс программы MATLAB

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 (таблица 1.1).

Таблица 1.1 – Форматы вывода на экран

| **Формат** | **Представление** |
|---|---|
| short | Число отображается с четырьмя цифрами после десятичной точки или в формате short e |
| short e | Число в экспоненциальной форме с мантиссой из пяти цифр и показателем из трех 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 трех 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 размером 1×1. 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 символам – 31 из общего числа – и различает большие и малые буквы. В MATLAB имеется ряд констант (таблица 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* (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.

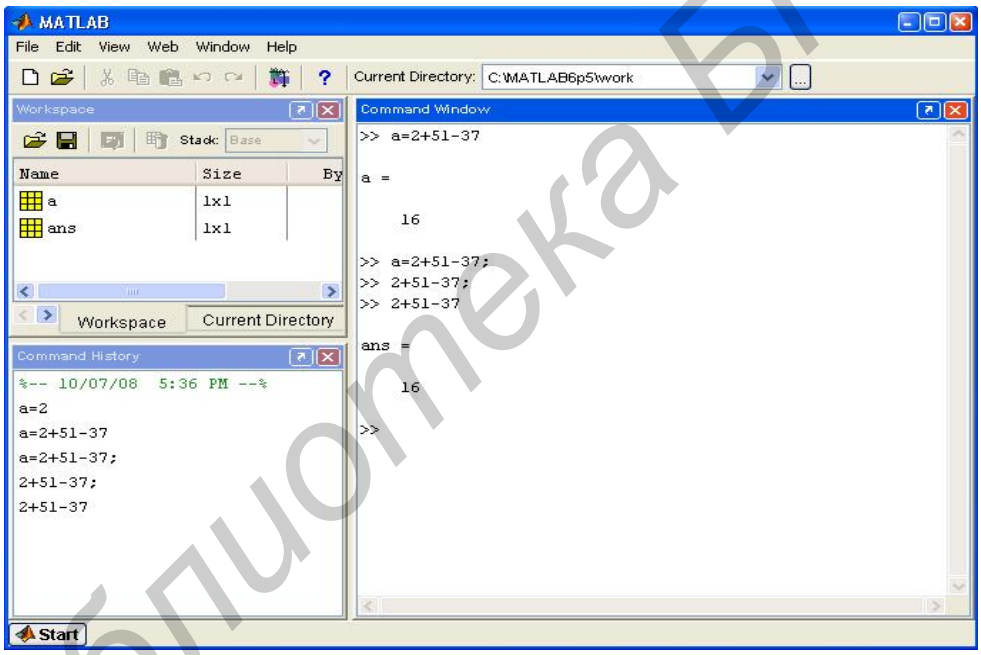



Рисунок 1.2 – Демонстрация выполнения команды присваивания

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 (таблица 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(**$x$) | Вычисление $e$ в степени $x$ |
| **log**($x$) | Вычисление натурального логарифма числа $x$ |
| **log10**($x$) | Вычисление десятичного логарифма числа $x$ |

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

Таблица 1.5 – Тригонометрические функции

| 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

| Вариант | Выражение | Переменные |
|---|---|---|
| 1 | 2 | 3 |
| 1 | $y=\sin\left(\frac{a-x}{c}\right)+10^4\cdot\sqrt[3]{\frac{a-kx^2}{2b}}+\frac{\cos(k)\cdot x^2}{tg3}-\frac{bc}{ax}.$ | $a=-1.3;\ b=0.91;$ $c=0.75;\ x=2.32;$ $k=8$ |
| 2 | $y=-\frac{(x-d)(x^2+b^2)}{\sqrt[3]{x^2+b^2-cd}}+10^{-3}tg(k\cdot n)-\frac{\cos(k\cdot x)}{\sin(5)}.$ | $d=1.25;\ b=0.75;$ $n=4;\ c=2.2;$ $x=0.32;\ k=2$ |
| 3 | $y=tg(i\cdot k)+10^3e^{-5}+\sqrt[3]{\frac{10^2\lvert xk\rvert}{(a+b)^2}}-\frac{ax^3-b}{(a+b)^2}.$ | $i=5;\ b=2.35;$ $a=25.2;\ x=0.1;$ $k=-2$ |
| 4 | $y=\frac{\sqrt{\lvert c-d\rvert+(a+c)^2}}{\sin(2i)}+10^{-3}e^{ix}-\frac{\lvert c-d\rvert+a^2}{\sqrt[3]{(a+c)^2}}.$ | $a=-1.25;\ d=2.5;$ $i=5;\ c=0.05;$ $x=1.35$ |
| 5 | $y=\frac{\ln\lvert kx\rvert}{\sin(7)}-\sqrt{\lvert x-a^2\rvert}-\frac{10^4a-b}{\cos(k\cdot x)}+\sqrt[3]{x-a^2}+c^3x.$ | $a=0.93;\ b=5.61;$ $c=0.31;\ x=-2.5;$ $k=2$ |
| 6 | $y=10^4\frac{ax}{b^2}-\left\lvert\frac{a-b}{kx}\right\rvert+\frac{\ln 3}{\sqrt[3]{ax^2+b^2}}-e^{-kx}.$ | $b=0.35;\ a=3.5;$ $x=1.523;\ k=-2$ |
| 7 | $y=-\frac{\lvert b-a\rvert}{kx}+10^4\cdot\sqrt[5]{\lvert\cos(k\cdot x)\rvert}+\sqrt{\frac{abc}{2.4}}-\frac{0.7abc}{\sin(7)}.$ | $a=1.7;\ b=-1.25;$ $c=-0.3;\ x=2.5;$ $k=3$ |
| 8 | $y=\frac{\lvert a^2-b^2\rvert}{\sin(k\cdot x)}+10^4\cdot\sqrt[5]{\lvert\sin(k\cdot x)-bc\rvert}-\frac{k^2+tg(3\cdot k)}{e^{kx}}.$ | $a=1.3;\ b=2.42;$ $c=0.83;\ x=1.5;$ $k=2$ |
| 9 | $y=\frac{\sqrt[3]{\ln x+a^2}}{0.47x^2}-\left\lvert 0.47x^2-\frac{10^4}{7}\cdot\cos^2 k\right\rvert-\frac{c}{x}.$ | $c=1.52;\ a=-2.4;$ $x=0.29;\ k=3$ |
| 10 | $y=\frac{1.5(a-b)^2}{\lvert a-b\rvert c}+\frac{i}{5}+10^3\cdot\sqrt{\lvert a-b\rvert}-\frac{(a+x^2)\cos(7)}{ix^2+a^2bc}.$ | $a=-2.5;\ b=1.35;$ $i=3;\ c=-0.72;$ $x=2.75$ |
| 11 | $y=10^4\cdot\sin^2 i-\frac{0.32x^3+4x+b}{\cos(i\cdot a)}\cdot\sqrt[6]{0.32x^3-b}+\lvert b\rvert.$ | $a=3.5;\ b=-0.7;$ $i=2;\ x=0.8$ |

Продолжение таблицы 1.6

| 1 | 2 | 3 |
|---|---|---|
| 12 | $y = -\frac{\cos(i)}{\sin(k \cdot x)} + \frac{ax^2 + \lvert d \rvert}{(a+b)^2} - 10^4 \cdot \sqrt[6]{\frac{kx}{(a+b)^2}}.$ | $d = -0.01;\ b = 1.25;$ $a = 4.72;\ i = 2;$ $x = 2.25;\ k = 3$ |
| 13 | $y = \cos(k) \cdot (x-a) + 10^{-4} \cdot \frac{(x+a)^3 + x^4 d}{k(x-a)^3} + \frac{\sqrt[5]{\lvert x+a \rvert}}{2.4b}.$ | $d = 0.95;\ b = 0.05;$ $a = -3.25;\ x = 8.2;$ $k = 4$ |
| 14 | $y = \sqrt[5]{\lvert ax^2 - b^3 \rvert} + \ln(k \cdot x) - \frac{e^{kx} + c^2}{\sin(k \cdot x)} - 10^{-3} \cdot \sqrt{2157}.$ | $c = 1.72;\ b = -0.31;$ $a = 2.01;\ x = 0.48;$ $k = 3$ |
| 15 | $y = \frac{1}{9} - 10^{-4} \cdot e^{kx} + \cos\left(\sqrt{(x^2+b)}\right) + \frac{\sqrt{x^2+b}}{0.4x} + \frac{\sin(3)}{(x^2+b) \cdot n}.$ | $x = 2.5;\ b = 0.04;$ $k = 3;\ n = 5$ |

### 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.

Практическое занятие №2

# ОПЕРАЦИИ С ВЕКТОРАМИ И МАТРИЦАМИ В MATLAB

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

## 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 (т.е. матрица, вторая размерность которой равна единице) может быть присвоен 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 задания диапазона $n{:}m{:}k$ будет сформирован вектор, последний элемент которого не больше $k$ для положительного шага $m$, и не меньше – для отрицательного: $[n, n + m, n + m + m,\ldots]$

Например:

```
>> a = 1:2:5
a =
    1    3    5
```

Задание диапазона используется также при организации цикла.

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

Таблица 2.1 – Функции описания матриц

| **Функция** | **Описание** |
|---|---|
| **eye**(m,n) | Единичная матрица размерностью *m*×*n* |
| **zeros**(m,n) | Нулевая матрица размерностью *m*×*n* |
| **ones**(m,n) | Матрица, состоящая из одних единиц размерностью *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 размерностью *m*×*n*, состоящую 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 согласно следующему правилу: в круглых скобках после имени матрицы даются индексы, которые должны быть положительными целыми числами, указывающими номер строки, и через запятую – номер столбца. Например, запись A(2,1) означает элемент из второй строки первого столбца матрицы A.

Для примера введем матрицу 2×2:

```
>> 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 при помощи команды **size**, а результат 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 выделить 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 дополнен специальными матричными операциями (таблица 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 комбинированным знаком **.'** |
| / | Левое деление |
| \ | Правое деление |
| ^ | Оператор возведения в степень, для поэлементного возведения в степень применяется комбинированный знак **.^** |

Покажем различие обычного и поэлементного умножения при помощи следующего примера.

Введём матрицу *H* размером 2×2 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, представленных 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 среднеквадратичное 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
```

```
>> h = [v + ' от MATLAB']
v =
Привет от MATLAB
```

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

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

Для перевода численных данных в строковые переменные имеется ряд команд преобразования. В таблице 2.4 приведены некоторые функции для этих и обратных им 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*) размерностью 2;

– произвольный вектор-столбец (*w*) размерностью 2;

– произвольную матрицу (*m*) размерностью 2×2.

2 Генерация матриц специального вида.

Создать:

– матрицу с нулевыми элементами (*m0*) размерностью 2×2;

– матрицу с единичными элементами (*m1*) размерностью 2×2;

– матрицу с элементами, имеющими случайные значения (*mr*) размерностью 2×2;

– матрицу с единичными диагональными элементами (*me*), размерностью 2×2.

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

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 формул для вычисления матрицы *M*

| Вариант | Формула | Вариант | 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 служит команды **zeros**, **ones**, **rand**, **eye**?
5 Как определяется число строк и столбцов матрицы?
6 Какие операции служат для определения минимального и максимального элементов матрицы?

Практическое занятие №3

## ПРОГРАММИРОВАНИЕ В СРЕДЕ MATLAB

*Цель занятия* – ознакомление с операциями отношения, логическими операциями и условными операторами, приобретение навыков 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 (*m*-файлы) представляют собой текстовые файлы, содержащие внешние программы, определения команд и функций системы. Именно ко второму типу относится большая часть команд и функций, в том числе задаваемых пользователем для решения специфических задач.

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

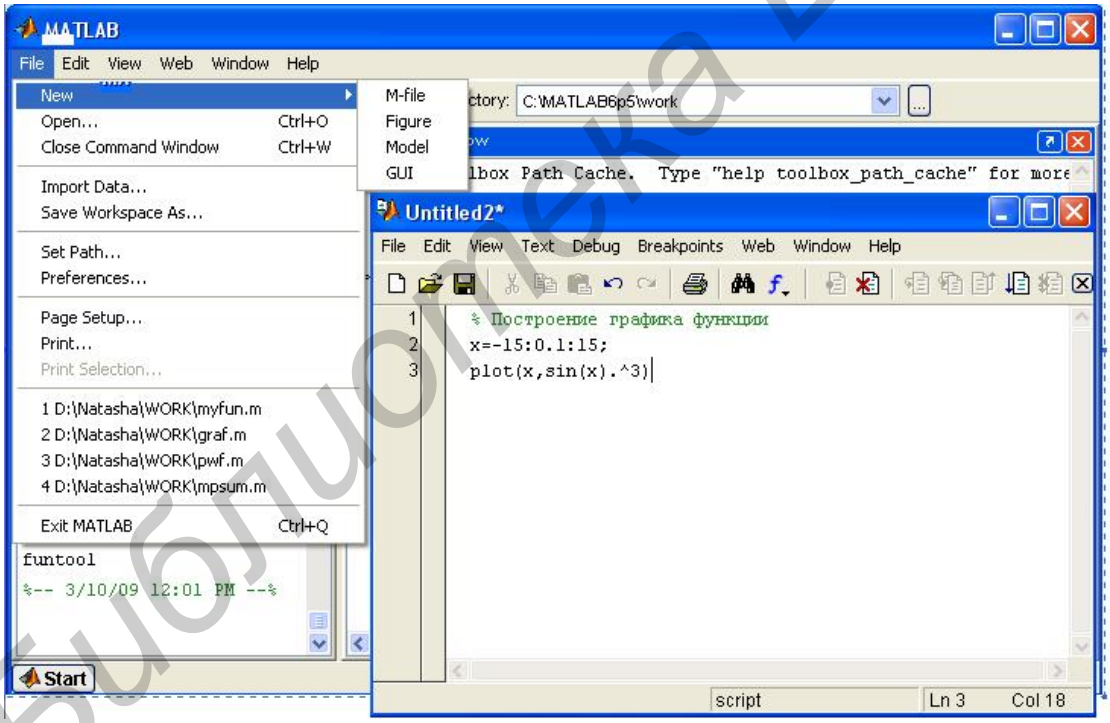



Рисунок 3.1 – Многооконный редактор-отладчик

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

– ключевые слова языка программирования выделяются синим цветом;

– операторы, константы и переменные – черным;

- комментарии после знака % – зеленым;

- символьные переменные (в апострофах) – коричневым;

- синтаксические ошибки – красным.

Благодаря цветовому выделению вероятность синтаксических ошибок резко снижается.

Создаваемые редактором-отладчиком *m*-файлы делятся на два класса: файлы-сценарии, не имеющие входных параметров, и файлы-функции, имеющие входные параметры. Файл-сценарий, именуемый также *script*-файлом, представляет собой простую запись серии команд без входных и выходных параметров и имеет следующую структуру:

%Основной комментарий
%Дополнительный комментарий
*Тело файла с любыми выражениями*

Важны следующие свойства:

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

Рассмотрим следующий файл-сценарий (рисунок 3.2).

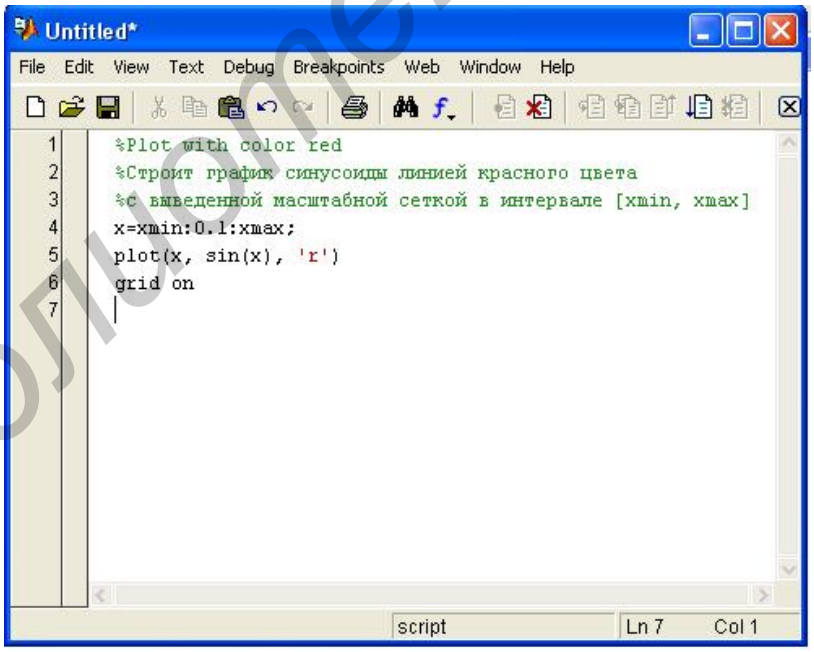



Рисунок 3.2 – Создание файла-сценария в MATLAB

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.

*M*-файл-функция является типичным объектом языка программирования системы MATLAB и одновременно – 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

*M*-файл-функция имеет следующие свойства:

1) начинается с объявления function, после которого указывается имя переменной var – выходного параметра, имя самой функции f_name и список ее входных параметров;

2) возвращает свое значение и может использоваться в математических выражениях;

3) имеет те же правила вывода комментариев, что и для файлов-сценариев;

4) файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;

5) все переменные, имеющиеся в теле файла-функции, являются локальными, т. е. действуют только в пределах тела функции;

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 выше 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 часто приводит к ошибкам в математических вычислениях. Поэтому, как отмечалось, данная функция используется как отдельный элемент программ вида [varl, var2] = f_name (*Список_параметров*). После ее 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 операторы, представленные в таблице 3.1.

Таблица 3.1– Операторы диалогового ввода/вывода

| **Оператор** | **Синтаксис** | **Назначение** |
|---|---|---|
| INPUT | *x* = input(‘<приглашение>’) | Ввод данных с клавиатуры |
| DISP | disp (<переменная или текст в апострофах>) | Вывод на дисплей |

Приведем пример простой диалоговой программы для многократного вычисления длины окружности по вводимому пользователем значению радиуса *r* (рисунок 3.3).

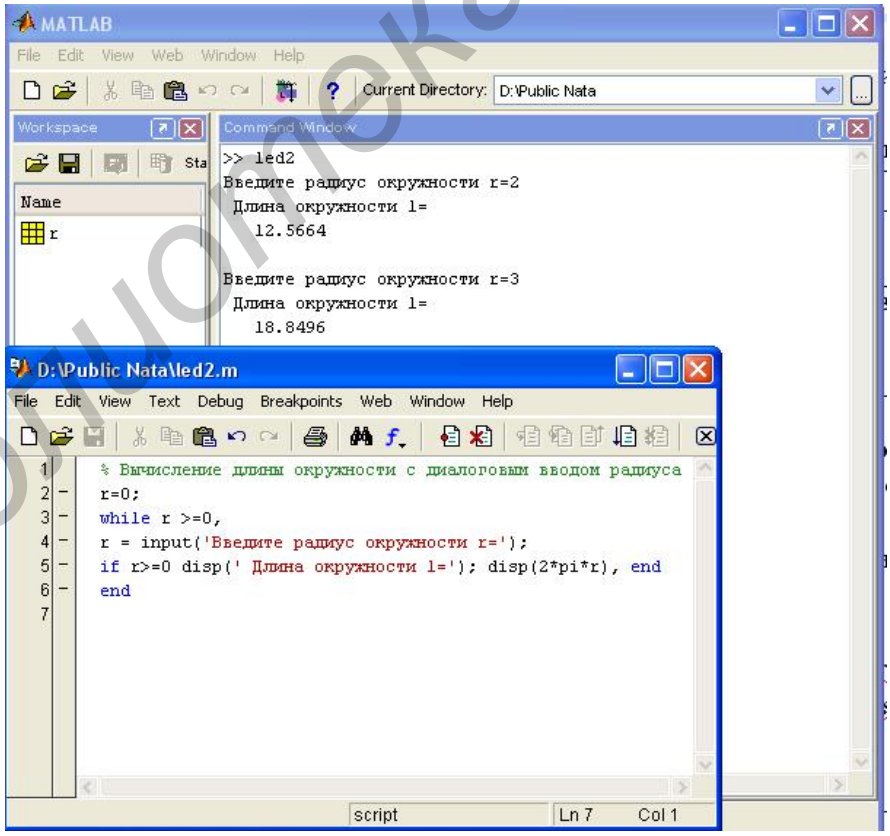



Рисунок 3.3 – Пример диалоговой программы

Для организации ветвлений служат условные операторы.

Конструкции условных операторов:

1) **if** *<условие>*
*<операторы>*
**end**

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

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

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.

Пусть например, требуется написать файл-функцию для вычисления кусочно-заданной функции

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

На рисунке 3.4 приведен листинг программы, вычисляющей значение данной функции.

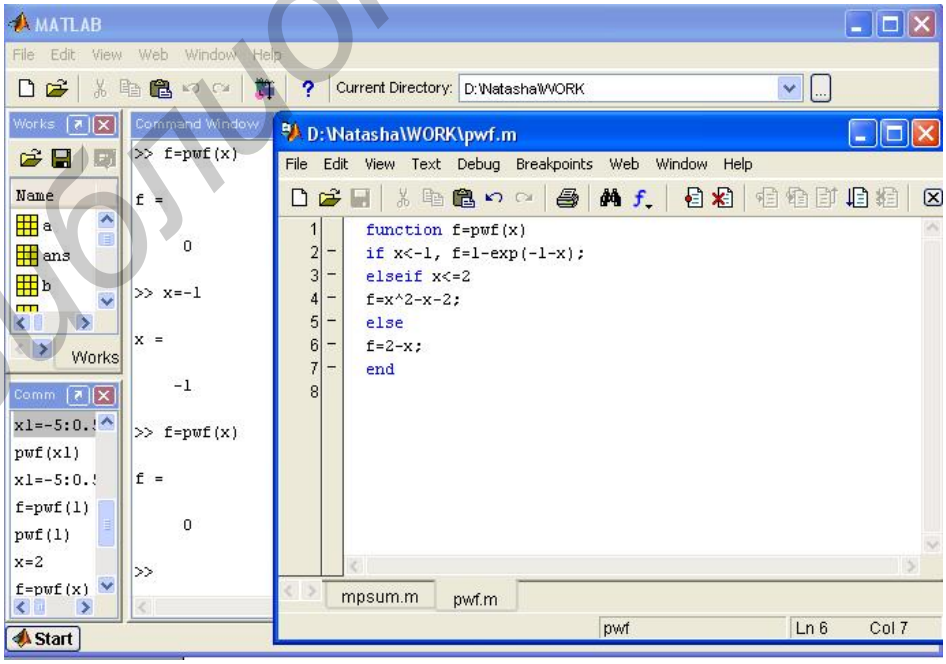



Рисунок 3.4 – Листинг программы для вычисления значения функции

В системе MATLAB могут применяться следующие операторы сравнения (таблица 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 (таблице 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 следующему после конца цикла оператору.

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 (рисунок 3.5).

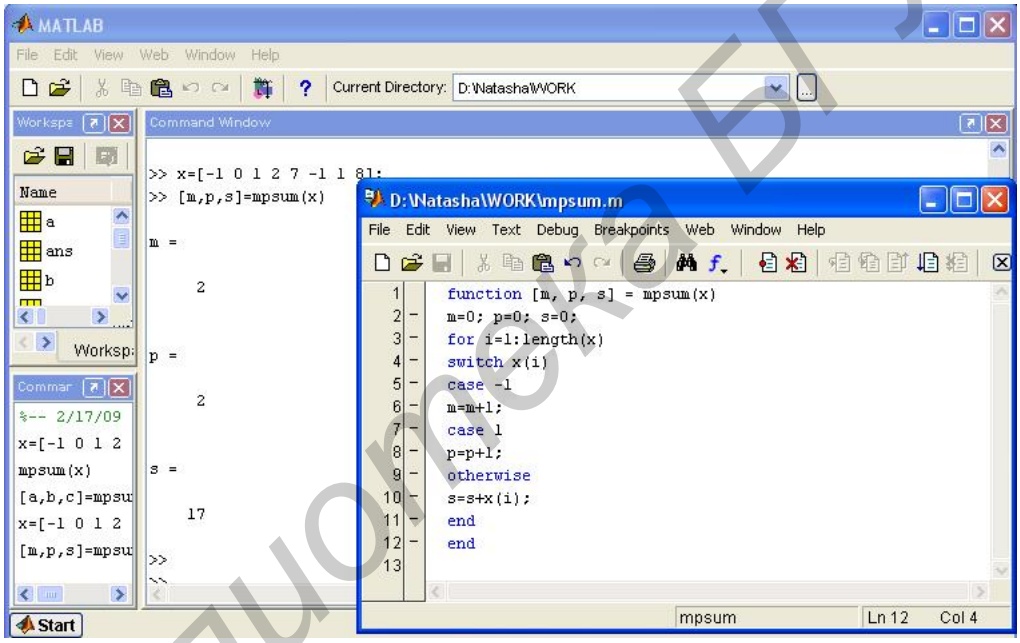



Рисунок 3.5 – Листинг программы

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

а) **pause** – останавливает вычисления до нажатия любой клавиши;
б) **pause(N)** – останавливает вычисления на N секунд;
в) **pause on** – включает режим отработки пауз;
г) **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

| **Вариант** | **Задание** |
|---|---|
| 1 | Найти сумму положительных из четырех заданных переменных. |
| 2 | Из четырех заданных переменных найти переменную с максимальным значением и вывести ее. |
| 3 | Заданы четыре переменные. Заменить наименьшую из них суммой остальных. |
| 4 | Заданы четыре переменные. Подсчитать количество отрицательных и количество нулевых из них. |
| 5 | Найти произведение только отрицательных переменных из четырех заданных. |
| 6 | Заданы две фигуры: квадрат – длиной стороны, круг – длиной радиуса. Определить, какая из фигур имеет большую площадь и во сколько раз. |
| 7 | Заданы четыре переменные. Каждую отрицательную из них заменить абсолютным значением и увеличить в два раза. |
| 8 | Заданы четыре переменные. Подсчитать количество равных нулю, положительных и отрицательных из них. |
| 9 | Заданы четыре переменные. Из них найти наиболее близкие по значению к переменной $x$. |
| 10 | Заданы четыре переменные. Все положительные из них заменить отрицательными значениями, умноженными на пять. |
| 11 | Найти минимальное и максимальное значения четырех заданных переменных. |
| 12 | Заданы четыре значения. Определить, какие из них представляют собой целые числа. |
| 13 | Заданы четыре переменные. Подсчитать количество и произведение значений, попавших в интервал [1; 5]. |
| 14 | Заданы четыре переменные. Подсчитать количество переменных с отрицательным и с нулевым значением. |
| 15 | Найти произведение только отрицательных переменных из четырех заданных. |

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

Таблица 3.5 – Варианты заданий

| Вариант | Вх. массив | Формируемый массив | Задача |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| 1 | $A_{3\times3}$ | $B_{3\times3},\ b_{ij}=\begin{cases}a_{ij},\ i<j;\\ a_{ji}^2,\ i\ge j.\end{cases}$ | 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 | $A_3$ | $B_3,\ b_i=\sin(i^2),\ i=1\mathbf{K}3$ | Сформировать массив $C$ – сумму элементов массивов $A$ и $B$. Найти максимальное значение массивов $A$, $B$, $C$ |
| 3 | $A_{3\times3}$ | $B_{3\times3},\ b_{ij}=\sin(i)*\sin(j),$ $i=1\mathbf{K}3,\ j=1\mathbf{K}3.$ | Определить минимальные элементы в матрицах $A$ и $B$ ($mA$ и $mB$). Вычислить $C=A*B*mA*mB$ |
| 4 | $A_{3\times3}$ | $B_{3\times3},\ b_{ij}=\begin{cases}2*i+3*j,\ i=j,\\ 5*i+2*j,\ i<j,\ i>j.\end{cases}$ | 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 | $A_{3\times3}$ | $B_3,\ b_i=\sin(i)+\cos(i),$ $i=1\mathbf{K}3.$ | Определить максимальные элементы в матрице $A$ и массиве $B$ ($mA$ и $mB$). Вычислить $C = A*B*mA*mB$ |
| 6 | $A_{3\times3}$ | $B_3,\ b_i=\log(2i+\cos(i)),$ $i=1\mathbf{K}3.$ | 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 | $A_{3\times3}$ | $B_3,\ b_i=\sin(\ln(i)+\cos(i)),$ $i=1\mathbf{K}3.$ | Заменить строку матрицы $A$, содержащую максимальный элемент, массивом $B$. Вычислить сумму элементов полученной матрицы |
| 8 | $A_3$ | $B_{3\times3},\ b_{ij}=\begin{cases}1+\cos(i-j),\ i<j,\\ 1-\sin(i+j),\ i\ge j.\end{cases}$ | Массив $A$ упорядочить по возрастанию и заменить им последнюю строку матрицы $B$ |

Продолжение таблицы 3.5

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 9 | $A_3$ | $B_3,\ b_i = i * \log(i^2) + \sin(i),$ $i = 1\mathbf{K}3.$ | Упорядочить по возрастанию массивы $A$ и $B$. Осуществить поэлементное деление упорядоченных массивов. Определить произведение элементов результирующего массива |
| 10 | $A_{3\times3}$ | $B_3,\ b_i = i * \sin(j) * \log(i),$ $i = 1\mathbf{K}3$ | Вычислить произведение элементов матрицы $A$ ($pA$) и сумму элементов матрицы $B$ ($cB$). Вычислить матрицу $C = pA*cB*A*B'$ |
| 11 | $A_{3\times3}$ | $B_3,\ b_i = i * \sin(i) + j * \cos(i),$ $i = 1\mathbf{K}3,\ j = 1\mathbf{K}3.$ | Определить минимальные элементы в матрицах $A$ и $B$ ($mA$ и $mB$). Вычислить $C = A*B*mA*mB$ |
| 12 | $A_{3\times3}$ | $B_3,\ b_i = \sin(2i) + \cos(3i),$ $i = 1\mathbf{K}3.$ | Сформировать массив $A1$ из максимальных элементов строк матрицы $A$. Осуществить поэлементное умножение $A1*B$. Упорядочить массив $A1$ по возрастанию |
| 13 | $A_{3\times3}$ | $B_3,\ b_i = \sin(\ln(i) + i * \cos(i)),$ $i = 1\mathbf{K}3.$ | Заменить строку матрицы $A$, содержащую минимальный элемент, массивом $B$. Вычислить произведение элементов полученной матрицы |
| 14 | $A_3$ | $B_{3\times3},\ b_{ij} = \begin{cases} i + \cos(i+j), & i > j, \\ j - \sin(i-j), & i \le j, \end{cases}$ $i = 1\mathbf{K}3,\ j = 1\mathbf{K}3.$ | Массив $A$ упорядочить по убыванию и заменить им первую строку матрицы $B$ |
| 15 | $A_3$ | $B_3,\ b_i = \cos(i^2),$ $i = 1\mathbf{K}3.$ | 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?

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

Несомненным достоинством системы MATLAB является 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 (Graphic User Interface – 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 $Y$, 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 символы, которые представлены в таблицах 4.1– 4.3.

Таблица 4.1 – Задание типа линии

| **Тип линии** | Непрерывная | Пунктирная (точками) | Штриховая | Штрих-пунктирная |
|---|---|---|---|---|
| **Маркер** | - | : | -- | -. |

Таблица 4.2 – Задание цвета графика

| **Цвет графика** | Голубой | Фиолетовый | Желтый | Красный | Зеленый | Синий | Белый | Черный |
|---|---|---|---|---|---|---|---|---|
| **Маркер** | c | m | y | r | g | b | w | k |

Таблица 4.3 – Задание типа точки

| **Тип точки** | Точка | Плюс | Звездочка | Кружок | Крестик |
|---|---|---|---|---|---|
| **Маркер** | . | + | * | o | 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 цвет 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* (т.е. векторов *x* и *y*), но 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 функции 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*). Данная команда 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 наглядными являются сетчатые графики поверхностей с заданной или функциональной окраской. В названии команд для их построения

присутствует слово **mesh** (англ. «ячейка сети»). 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)
```

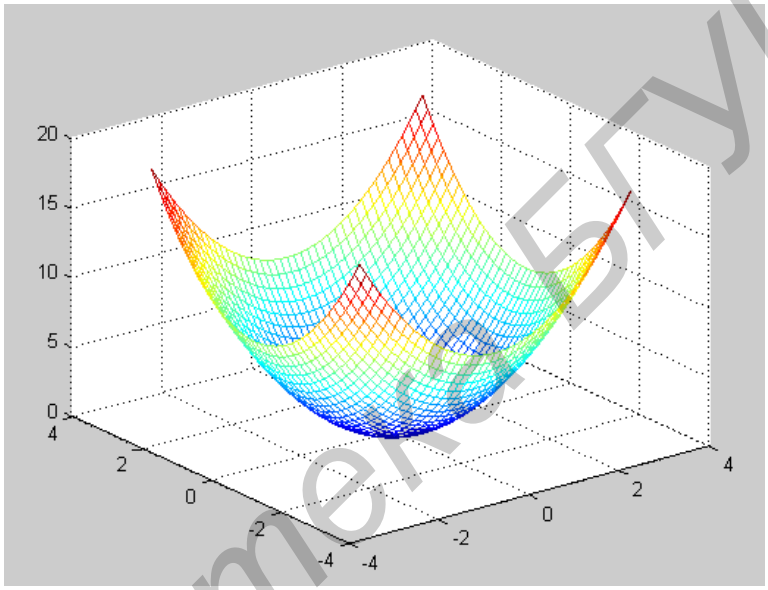



Рисунок 4.1 – График поверхности, созданный командой mesh(X,Y,Z)

После того как график уже построен, MATLAB позволяет выполнить его форматирование или оформить в нужном виде. Так, для размещения 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').

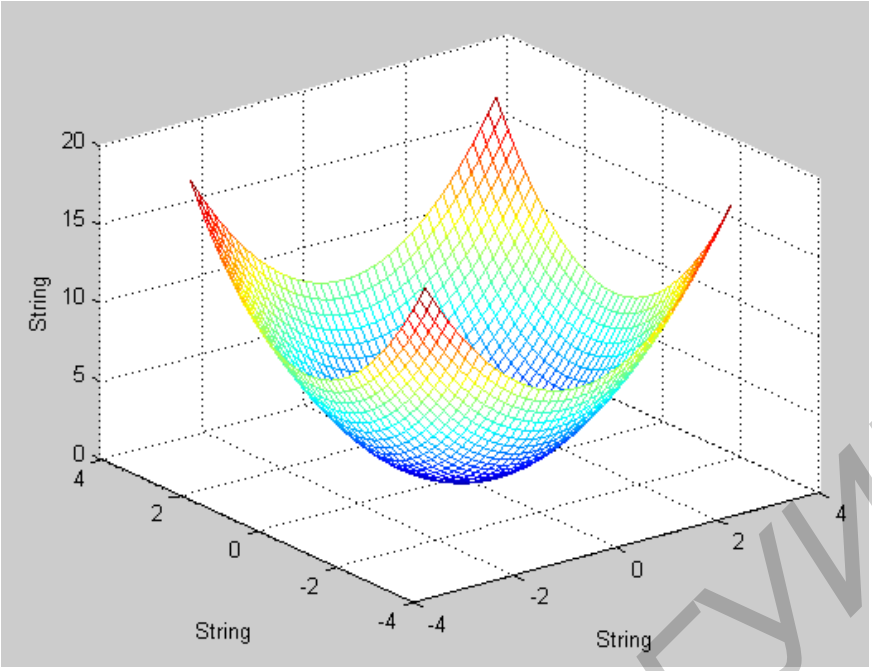



Рисунок 4.2 – Установка надписей с использованием команды: xlabel('String'), ylabel('String'), zlabel('String')

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

– **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')
```

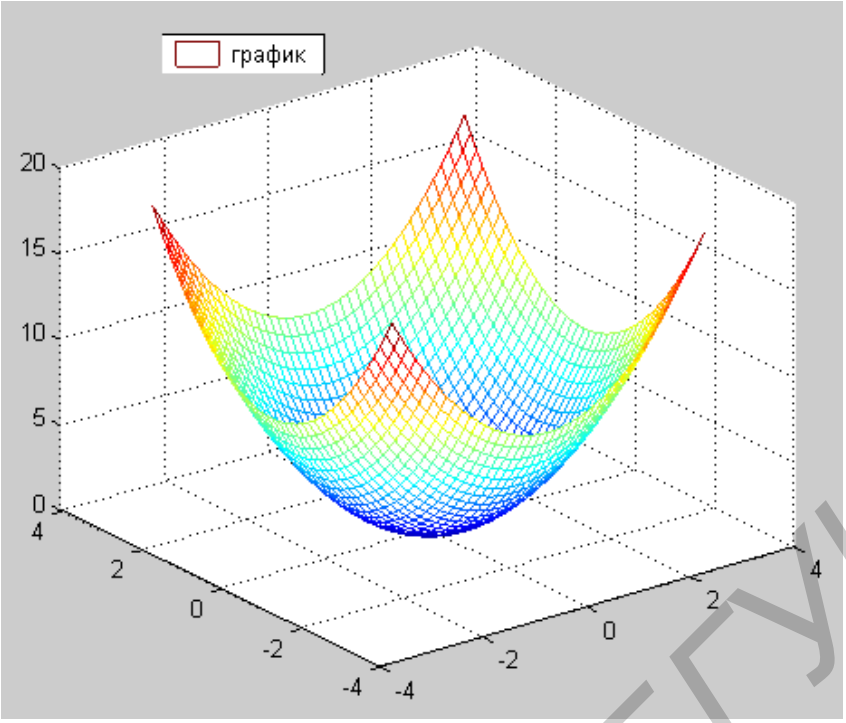



Рисунок 4.3 – График с пояснениями

**legend** (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** предназначены для ручного ввода диапазонов координат по осям:

– **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 класса **grid** позволяют задавать построение сетки или отменять его:

– **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 предназначена команда про-

должения 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.

Иногда в одном окне надо расположить несколько координатных осей с различными графиками без наложения их друг на друга. Тогда используются команды класса **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 для вывода текущего графика (подокна отсчитываются последовательно по строкам).

Проиллюстрируем работу функции **subplot** (рисунок 4.4):

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

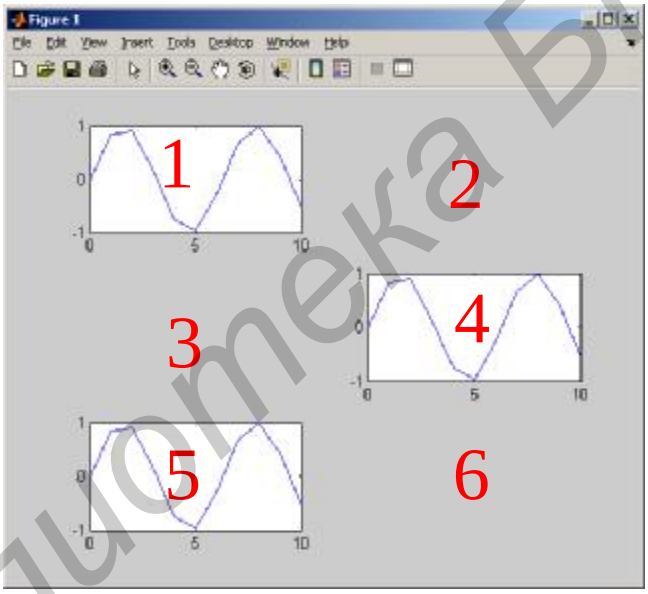



Рисунок 4.4 – Работа функции **subplot**

Было сформировано три строки и два столбца полей для вывода графиков. Обращение к каждому конкретному полю происходит с указанием номера данного поля. 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 варианта задания из таблицы 4.4. Вывод графиков должен быть осуществлен в одном окне, все графики должны быть подписаны и отмасштабированы.

Таблица 4.4 – Варианты заданий

| **Вариант** | ***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 | $x*\sin(x)$ | $x*\cos(x)$ | $x^2$ | $(\cos(r)/r)^2$ |
| 6 | $x*e^x$ | $\sin(x)+x^2$ | $\sin(x)+x$ | $\sin^2(r)/r$ |
| 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 функции *f4* из таблицы 4.4, где $r=\sqrt{x^2+y^2}$ .

3 Написать файл-функцию для вычисления кусочно-заданной функции (таблица 4.5) и построить ее график.

Таблица 4.5 – Варианты заданий кусочно-заданной функции

| Вариант | Функция | Вариант | Функция |
|---|---|---|---|
| **1** | **2** | **3** | **4** |
| 1 | $y=\begin{cases}\dfrac{1+x^2}{\sqrt{1+x^4}}, x\le 0\\ 2x+\dfrac{\sin^2 x}{3+x}, x>0\end{cases}$ | 3 | $y=\begin{cases}3\sin x-\cos^2 x, x\le 0\\ \dfrac{3\sqrt{1+x^4}}{\ln(x+5)}, x>0\end{cases}$ |
| 2 | $y=\begin{cases}\dfrac{3+\sin^2(2x)}{1+\cos^2 x}, x\le 0\\ 2x+\dfrac{\sin^2 x}{3+x}, x>0\end{cases}$ | 4 | $y=\begin{cases}\dfrac{3x^2}{1+x^2}, x\le 0\\ \sqrt{1+\dfrac{2x}{e^{0.5x}+x^2}}, x>0\end{cases}$ |

Продолжение таблицы 4.5

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 5 | $y=\begin{cases}\dfrac{3+\sin^2 x}{1+x^2}, x\le 0\\ 2x^2\cos^2 x, x>0\end{cases}$ | 11 | $y=\begin{cases}\sqrt{1+2x^2+\sin^2 x}, & x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0.5x}}}, & x>0\end{cases}$ |
| 6 | $y=\begin{cases}\sqrt{1+x^2}, x\le 0\\ \dfrac{1+x}{\sqrt[3]{1+e^{-0.2x}}+1}, x>0\end{cases}$ | 12 | $y=\begin{cases}\sqrt{1+\lvert x\rvert}, & x\le 0\\ \dfrac{1+3x}{\sqrt[3]{1+x}+2}, & x>0\end{cases}$ |
| 7 | $y=\begin{cases}\dfrac{\sqrt{1+\lvert x\rvert}}{2+\lvert x\rvert}, x\le 0\\ \dfrac{1+x}{2+\cos^3 x}, x>0\end{cases}$ | 13 | $y=\begin{cases}\sqrt[3]{1+x^2}, & x\le 0\\ \sin^2 x+\dfrac{1+x}{1+e^x}, & x>0\end{cases}$ |
| 8 | $y=\begin{cases}\dfrac{1+\lvert x\rvert}{\sqrt[3]{1+x+x^2}}, x\le -1\\ \dfrac{1+\cos^4 x}{3+x}, x>-1\end{cases}$ | 14 | $y=\begin{cases}2\ln(1+x^2), & x\le -1\\ (1+\cos^2 x)^{3/5}, & x>-1\end{cases}$ |
| 9 | $y=\begin{cases}\dfrac{1+x}{\sqrt[3]{1+x^2}}, & x\le 0\\ -x+2e^{-2x}, & x>0\end{cases}$ | 15 | $y=\begin{cases}3x+\sqrt{1+x^2}, & x\le 0\\ 2\cos x e^{-2x}, x>0\end{cases}$ |
| 10 | $y=\begin{cases}\sqrt{1+x^2}, & x\le 0\\ \dfrac{1+x}{\sqrt[3]{1+e^{-0.2x}}+1}, x>0\end{cases}$ | 16 | $y=\begin{cases}\sqrt{1+2x^2+\sin^2 x}, x\le 0\\ \dfrac{2+x}{\sqrt[3]{2+e^{-0.5x}}}, x>0\end{cases}$ |

### 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 в декартовой системе 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* – ознакомление 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

$$A \cdot y = B, \qquad (5.1)$$

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

$B$ – 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 матрице $A$. Операцию / называют правым делением матрицы. Выражение $A/B$ примерно соответствует выражению $B$***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. Для примера определим нули функции cos($x$) на отрезке от 0 до π (pi). В качестве начального приближения примем $x_0 = 1$.

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

Если требуется найти корень функции, отличной от стандартной (т.е. 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 их в порядке 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 **fmin:** 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 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 $\int_0^p \cos(x)dx$ методом трапеций с шагом интегрирования π/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 над достигнутой точностью осуществлять путем сравнения последовательных результатов. При одном и том же шаге интегрирования методы более высоких порядков точности позволяют достигать более точных результатов. В системе MATLAB методы интегрирования более высоких порядков точности реализуются функциями **quad** (метод Симпсона) и **quad8** (метод Ньютона–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, поэтому 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 выражения в одну строку не всегда удобна, более естественный вид выражения выводит в командное окно функция **pretty** (рисунок 5.1).

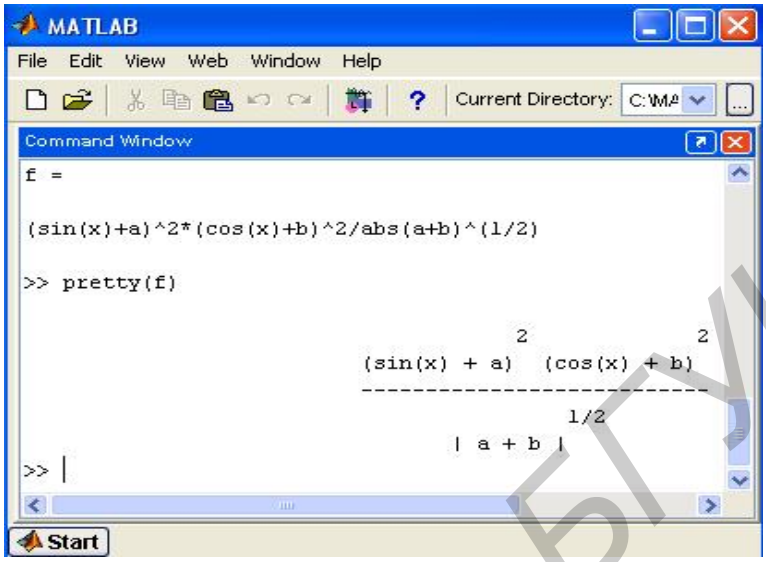



Рисунок 5.1 – Демонстрация работы функции **pretty**

Упростим выражение $\frac{x^2 - y^2}{x - y}$, используя функцию **simplify** (рисунок 5.2).

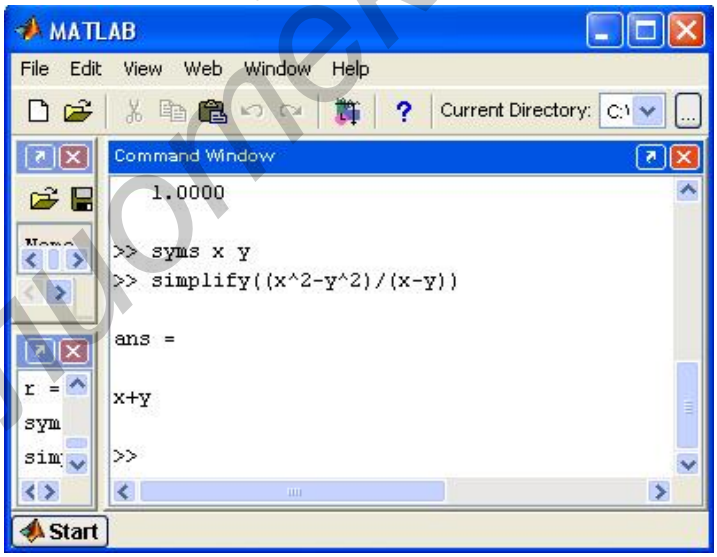



Рисунок 5.2 – Демонстрация работы функции **simplify**

Символьную функцию можно создать без предварительного объявления переменных при помощи функции **sym**, входным аргументом которой является строка с выражением, заключенная в апострофы (рисунок 5.3).

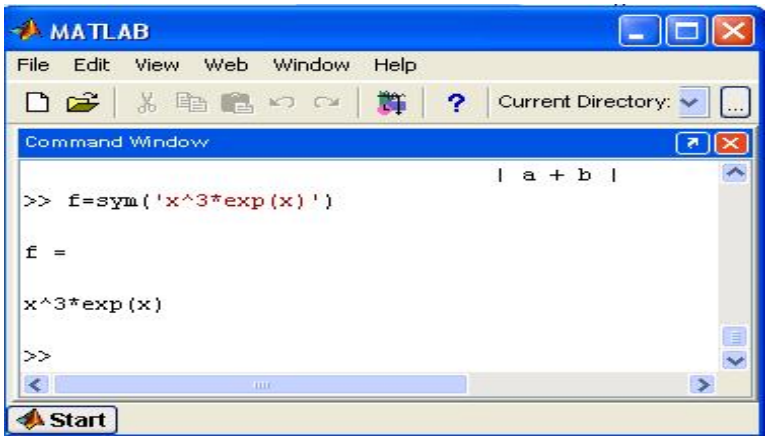



Рисунок 5.3 – Демонстрация создания символьной переменной без предварительного объявления

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).

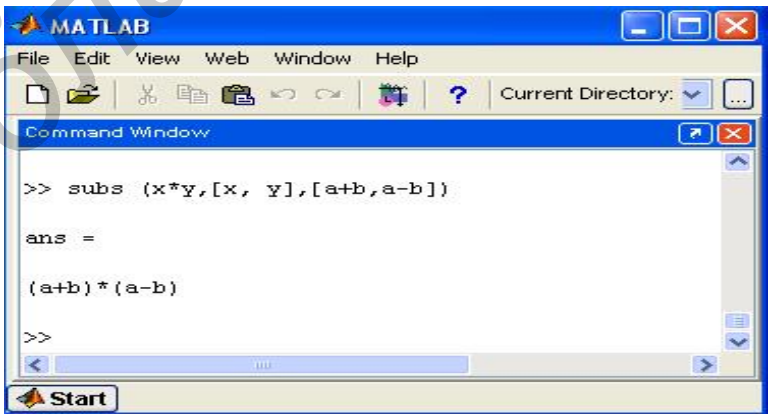



Рисунок 5.4 – Демонстрация работы функции **subs**

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

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

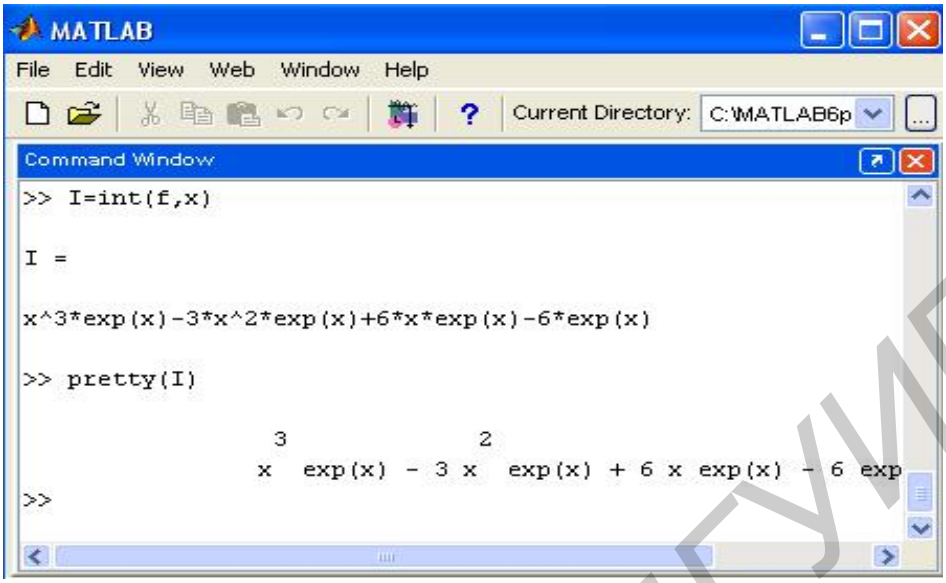



Рисунок 5.5 – Демонстрация работы функции **int**

Для нахождения определенного интеграла в символьном виде следует задать нижний и верхний пределы интегрирования соответственно в третьем и четвертом аргументах **int** (рисунок 5.6).

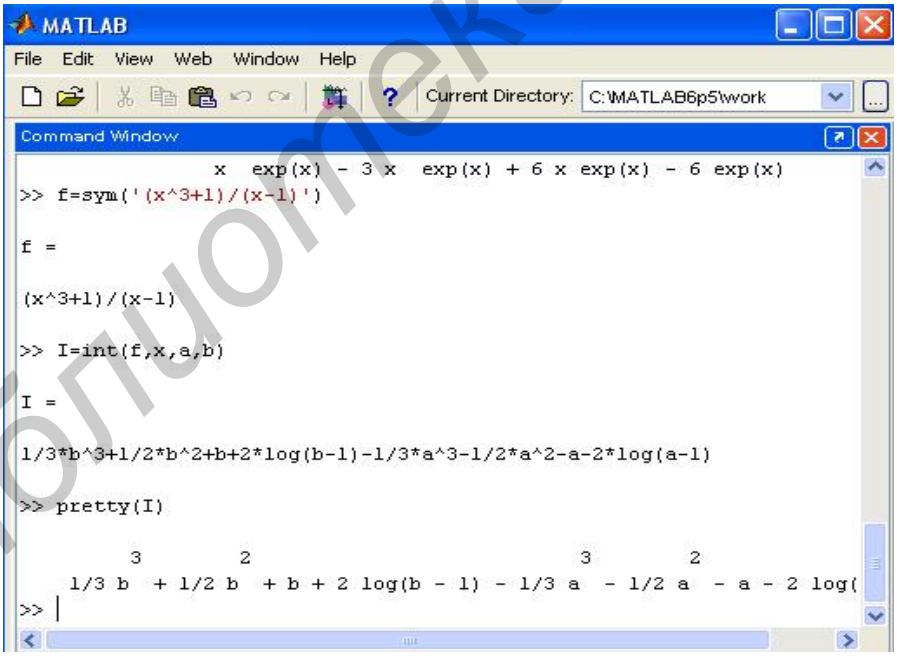



Рисунок 5.6 – Демонстрация работы функции **int** для нахождения определенного интеграла

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 приложения выводятся три автономных окна: два графических и управляющее. Управляющее окно содержит два поля для ввода функций $f(x)$ и $g(x)$, поле ввода пределов изменения переменной *x*, поле ввода масштабирующего коэффициента.

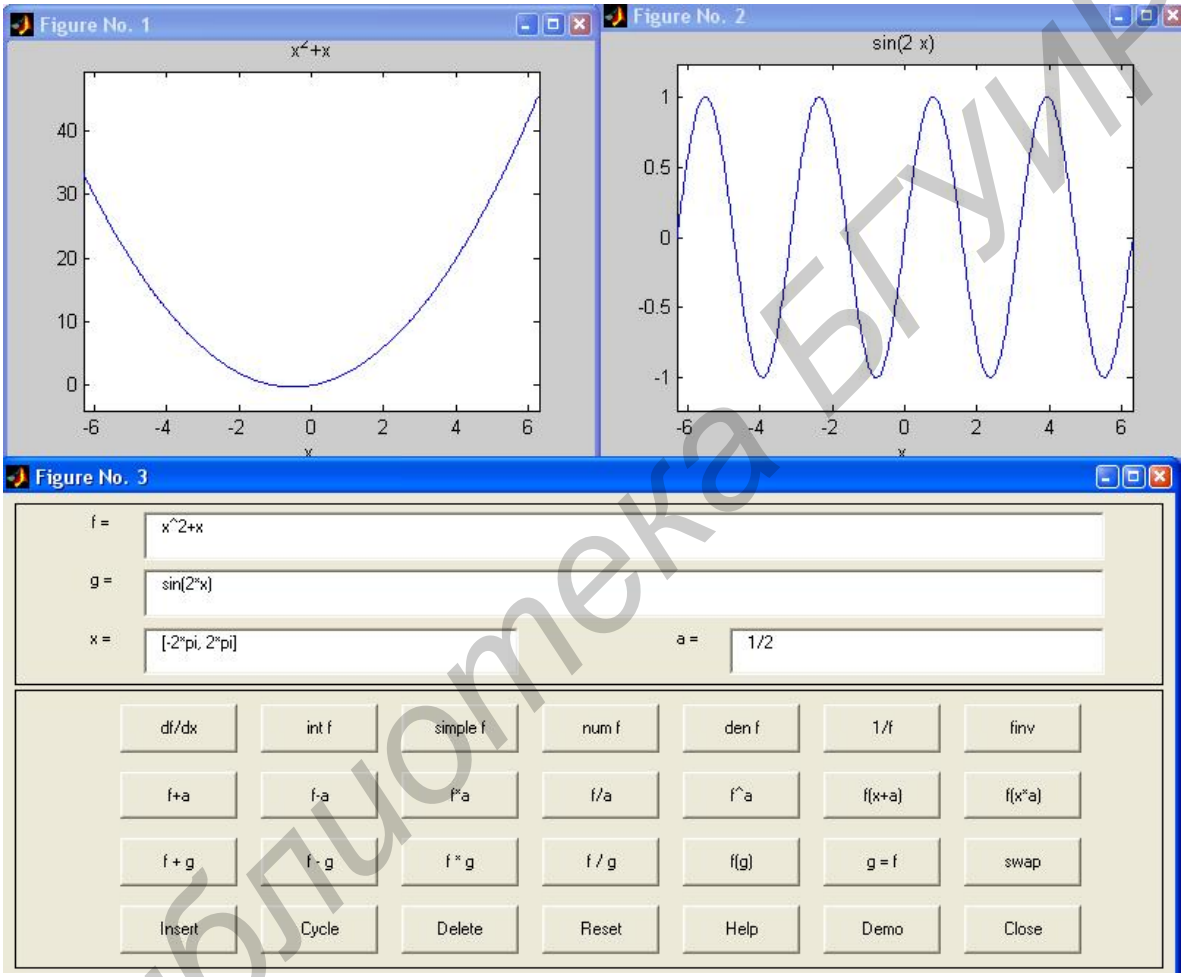



Рисунок 5.7 – Интерфейс приложения **funtool**

## 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

| **Вариант** | **$f_1(x)$ – полином 3-й 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 | $70 \cdot \cos(x)$ |
| 6 | 1 | –5 | 4 | 40 | $60e^{\cdot\lvert 0.1x\rvert} - 100$ |
| 7 | 2 | –3 | 2 | 30 | $20 \cdot \sin(2x)$ |
| 8 | 3 | –6 | 1 | 50 | $e^{\cdot\lvert x\rvert} \sin(2x)$ |
| 9 | 4 | –9 | 1 | 70 | $e^{\cdot\lvert x\rvert} \cos(3x)$ |
| 10 | 5 | –7 | 5 | 60 | $-60 \cdot \lvert\cos(x)\rvert$ |
| 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 | $-100 \cdot \lvert\cos(x)\rvert$ |
| 14 | –4 | 7 | 8 | –75 | $20 \cdot \sin(x/2)$ |
| 15 | –5 | 1 | 4 | –1 | $40 \cdot \cos(x/2)$ |

Таблица 5.2 – Варианты заданий для нахождения значения интеграла функции

| **Вариант** | **Функция** | **Интервал интегрирования** | |
|---|---|---|---|
| | | **начало интервала** | **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 | $f(x) = 2x - tg(x)$ | –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 |
| 12 | $f(x) = x^2 - 20\cos(x + 1)$ | –2 | 5 |

Продолжение таблицы 5.2

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| 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**?
3 Перечислите основные внешние расширения системы MATLAB для поиска экстремумов функций.
4 Какая функция служит для вычисления определенного интеграла?
5 Назовите функции работы с символьными переменными.

Практическое занятие №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 подразумевают описание некоторой, порой не заданной явно зависимости или совокупности представляющих ее данных с помощью другой, обычно более простой или более единообразной зависимости. Часто данные задаются в виде отдельных узловых точек, координаты которых приводятся в таблице 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 **polyfit**(*x*, *y*, *n*). Она 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, когда промежуточные точки ищутся по отрезкам полиномов третьей степени – это *кубическая сплайновая интерполяция* (сплайн-интерполяция). При этом обычно данные 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) этого вида интерполяции (аппроксимации).

Для одномерной табличной интерполяции используется функция **interpl** в следующих вариантах:

– $y_i$ = **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*;

– $y_i$ = **interp1**(*x*, y, $x_i$, 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:

$y_i$ = **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 воспроизведении функции по табличным данным) и в построении функции аппроксимации или интерполяции (рисунок 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, выполняемая прямо в окне редактора графики **Property Editor**. Для этого в позиции **Tools** графического окна имеются две 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) (рисунок 6.2).

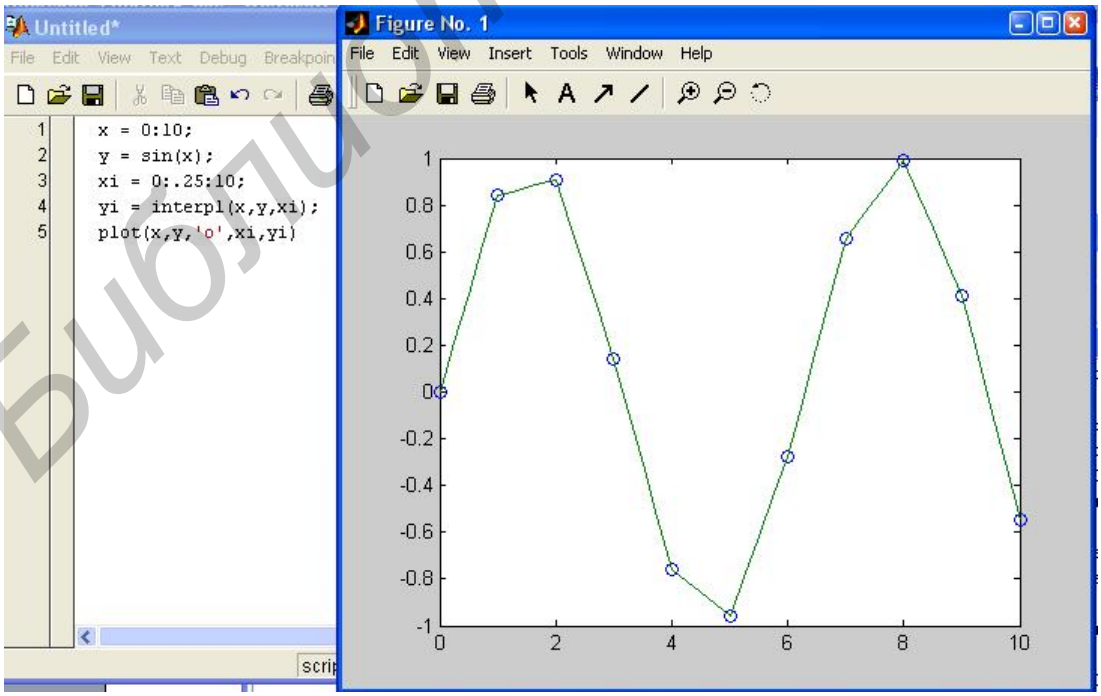



Рисунок 6.1 – Пример визуализации процесса интерполяции

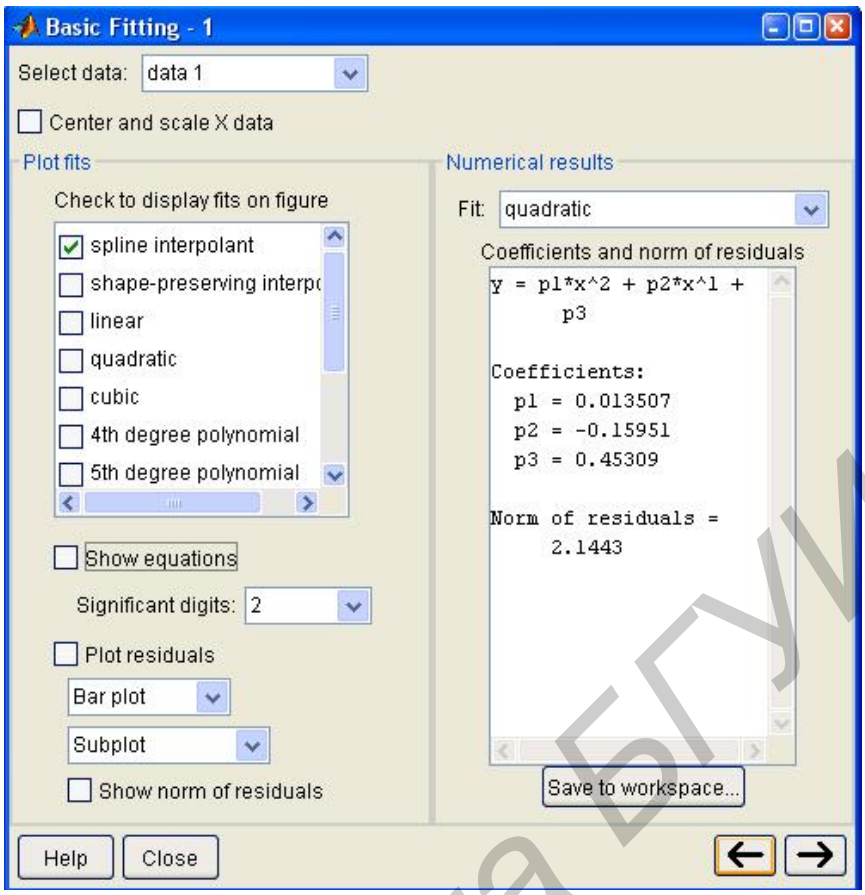



Рисунок 6.2 – Окно доступа к видам аппроксимации и регрессии

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

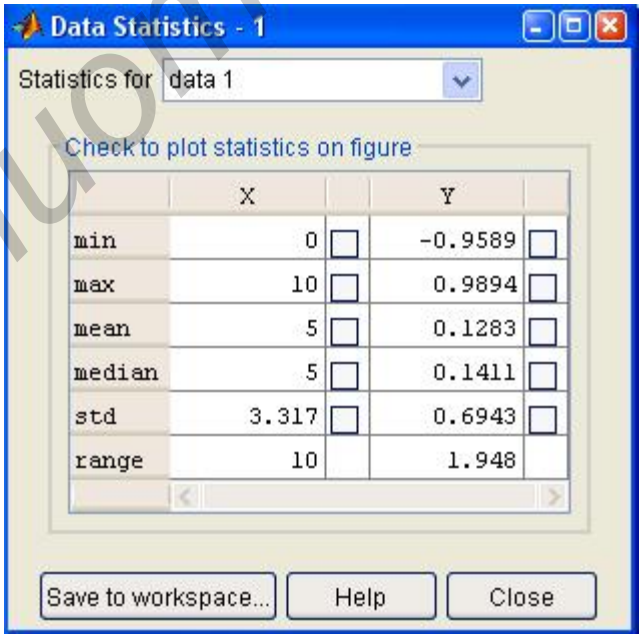



Рисунок 6.3 – Окно результатов статистической обработки

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

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

$$\frac{dy}{dt} = y';$$
$$y' = f(x, y),\ y(t_0, t_{end}) = b. \qquad (6.1)$$

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

Параметр $t$ необязательно означает время, хотя чаще всего поиск решения дифференциальных уравнений происходит во временной области. Вектор $b$ задает начальные и конечные условия.

Для решения систем ОДУ в MATLAB имеются различные методы, реализации которых названы решателями ОДУ.

Опишем функцию для решения систем дифференциальных уравнений **solver**:

– [*T*, *Y*] = (@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.

В описанных функциях приняты следующие обозначения:

– options – аргумент, создаваемый функцией **odeset** (данная функция также позволяет вывести параметры, установленные по умолчанию);

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

– *y0* – вектор начальных условий;

– $p_i$, $p_2$,… – произвольные параметры, передаваемые в функцию F;

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

Для решения жестких систем уравнений рекомендуется использовать только специальные решатели **ode45**, **ode23**:

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

– **ode23** – одношаговые явные методы Рунге–Кутта 2-го и 4-го порядка.

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 сопровождающего его графика:

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

Пусть, например, требуется решить дифференциальное уравнение

$$y''' - 2y'' - y' + 2y = 0 \qquad (6.2)$$

с единичными начальными условиями.

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

$$\begin{cases} \dfrac{dy_1}{dt} = y_2; \\ \dfrac{dy_2}{dt} = y_3; \\ \dfrac{dy_3}{dt} = 2y_3 + y_2 - 2y_1 \end{cases} \qquad (6.3)$$

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):

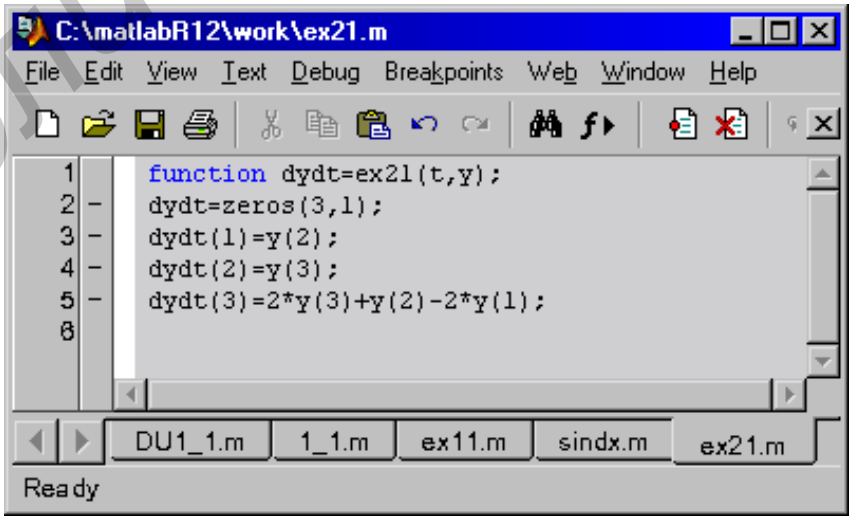



Рисунок 6.4 – Пример создания функции для решения системы ОДУ

Теперь можно вызывать функцию **ode45**, находящую решение 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)
```

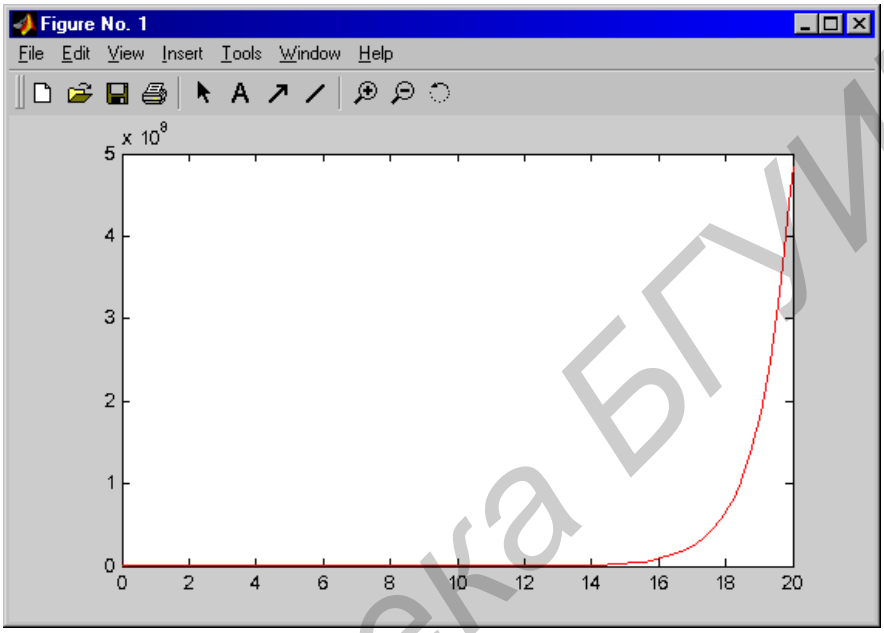



Рисунок 6.5 – Результат работы программы

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

– y = **dsolve**( 'Dy(x)' ), где Dy(x) – уравнение, y – возвращаемые функцией **dsolve** решения.

– y = **dsolve** ('Dy(x)', 'НУ'), где Dy(x) – уравнение, НУ – начальные условия.

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

– [f,g] = **dsolve**('Df(x),Dg(x)', 'НУ'), где Df(x), Dg(x) – система уравнений, НУ – начальные условия.

Решить дифференциальное уравнение (6.2) с использованием функции **dsolve** (рисунок 6.6).

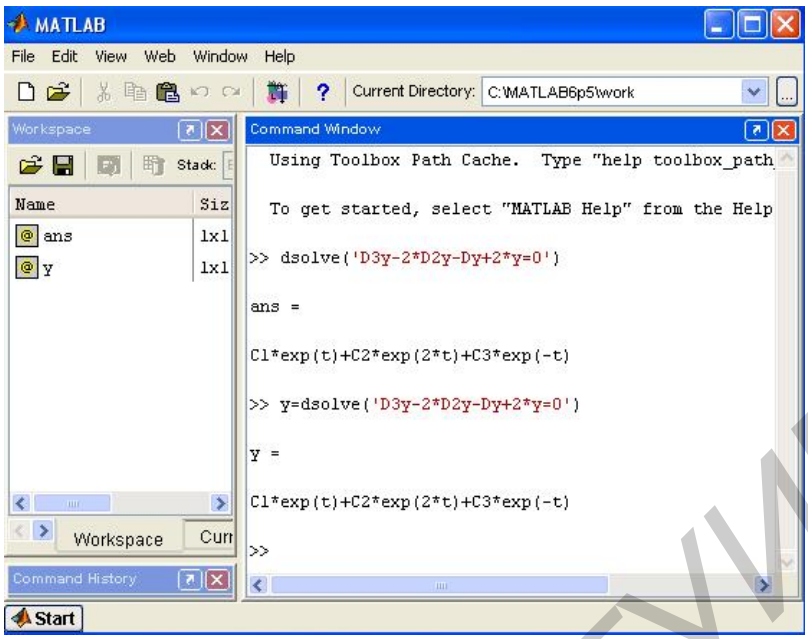



Рисунок 6.6 – Пример использования команды **dsolve**

### 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 (см. таблицу 6.1).

Таблица 6.1 – Варианты заданий

| **Вариант** | **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 |
| 9 | *X* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| | *Y* | –1 | 0 | 3 | 8 | 15 | 24 | 35 | 48 |

Продолжение таблицы 6.1

| 1 | 2 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 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 $\frac{d^2y}{dx^2} = f(x, y, y')$ с начальными условиями $y(0)$, $y'(0)$. Варианты заданий представлены в таблице 6.2. Построить график функции.

Таблица 6.2 – Варианты заданий

| Вариант | $f(x, y, y')$ | 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 |
| 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

| Вариант | 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 |
| 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 |

Продолжение таблицы 6.3

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 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?

Практическое занятие №7

# ОСНОВНЫЕ ВОЗМОЖНОСТИ ПАКЕТА МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ SIMULINK 4.0

*Цель занятия* – ознакомление с возможностями пакета математического моделирования Simulink.

## 7.1 Основные теоретические сведения

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

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

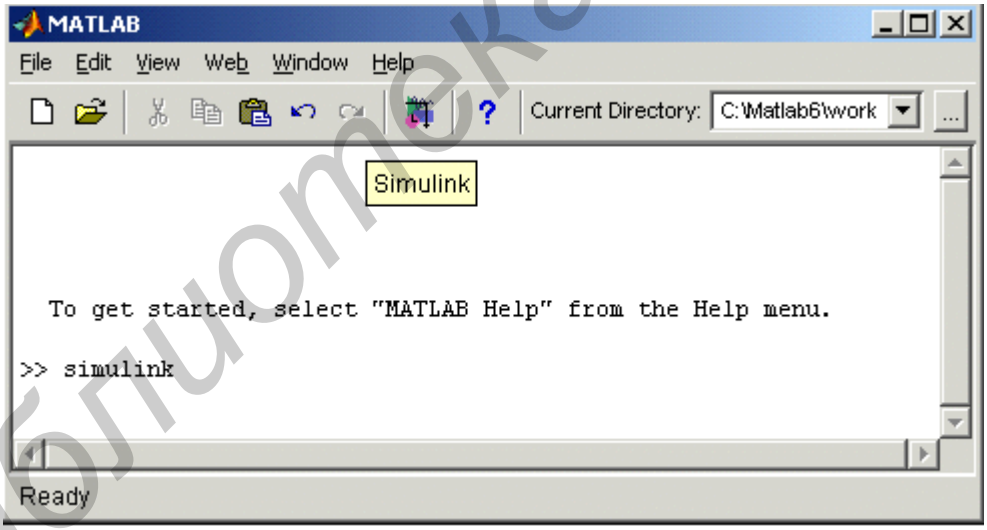



Рисунок 7.1 – Основное окно пакета MATLAB

После открытия основного окна системы MATLAB нужно запустить приложение Simulink одним из трех способов:

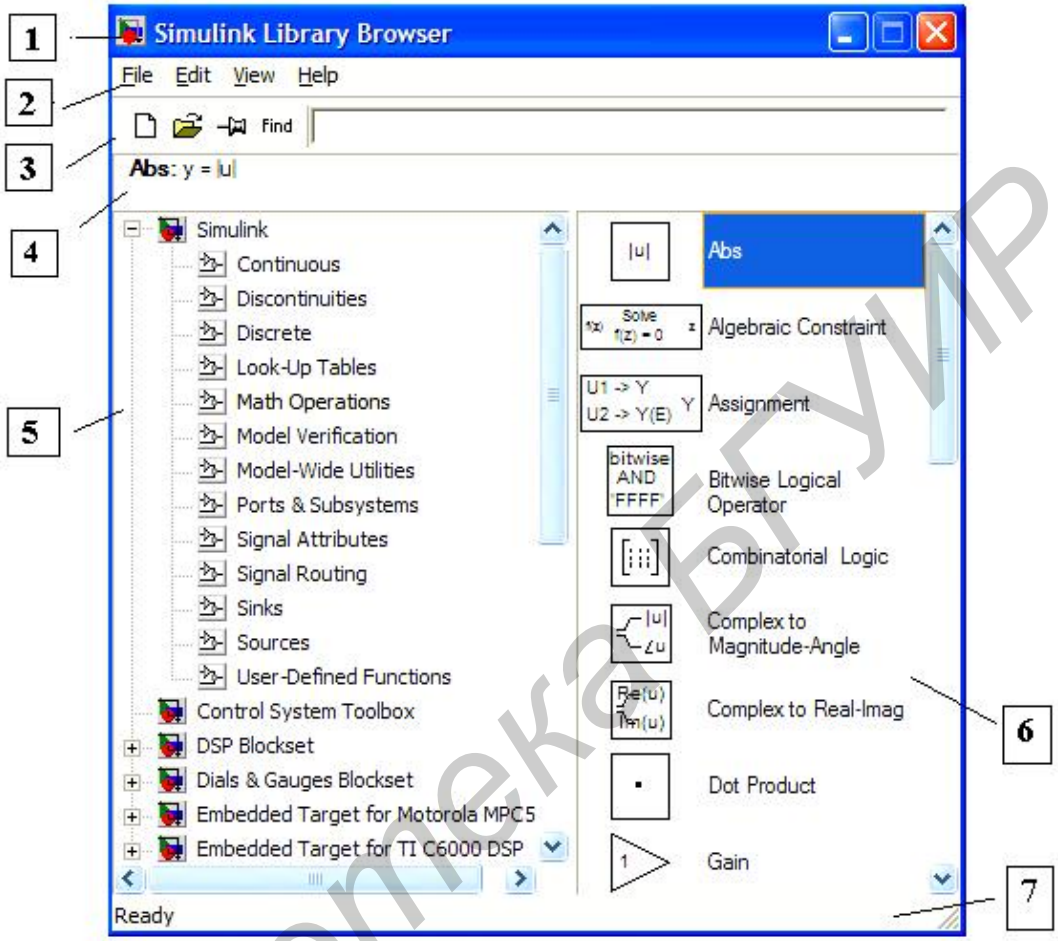
1 Нажать кнопку **Simulink** на панели инструментов командного окна MATLAB.

2 В командной строке главного окна MATLAB напечатать *Simulink* и нажать клавишу Enter на клавиатуре.

3 Выбрать команду **Open**… в меню **File** и открыть файл модели (mdl-файл). Данный способ удобно использовать для запуска уже готовой и отлаженной

модели, когда требуется лишь провести расчеты и не нужно добавлять новые блоки в модель.

Запуск приложения при помощи первого и второго способов приводит к открытию окна обозревателя разделов библиотеки Simulink (рисунок 7.2).




1 – заголовок; 2 – меню; 3 – панель инструментов;
4 – окно комментария; 5 – список разделов библиотеки;
6 – окно содержимого разделов библиотеки; 7 – строка состояния

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.

Библиотека Simulink содержит следующие основные разделы:

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

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 по его пиктограмме левой клавишей мыши.

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

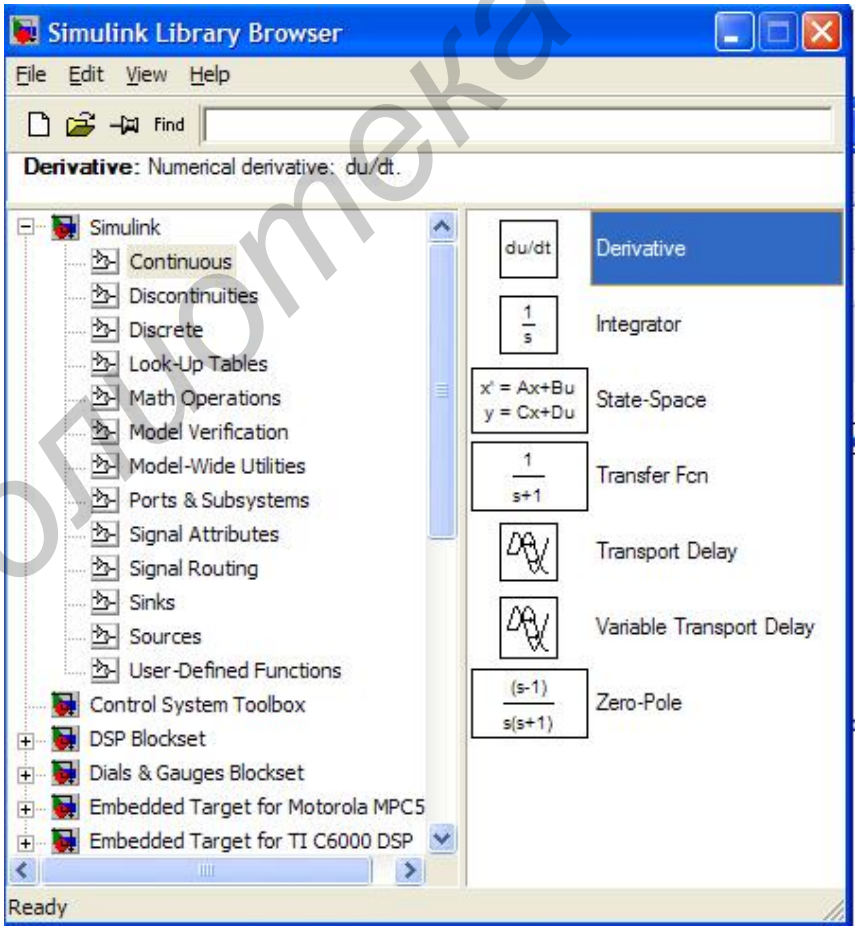



Рисунок 7.3 – Содержимое раздела **Continuous**

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.

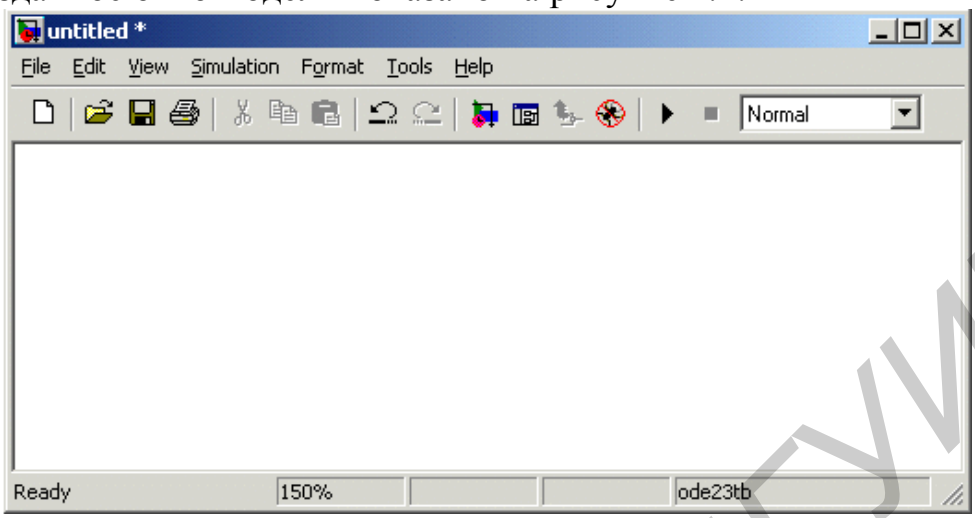



Рисунок 7.4 – Пустое окно модели

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, удерживая клавишу мыши нажатой. На рисунке 7.5 показано окно модели, содержащее блоки.

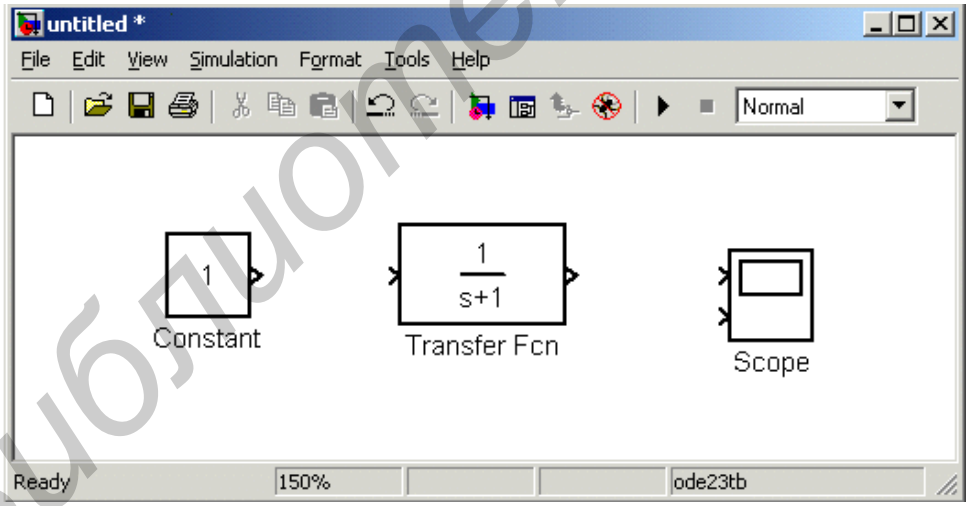
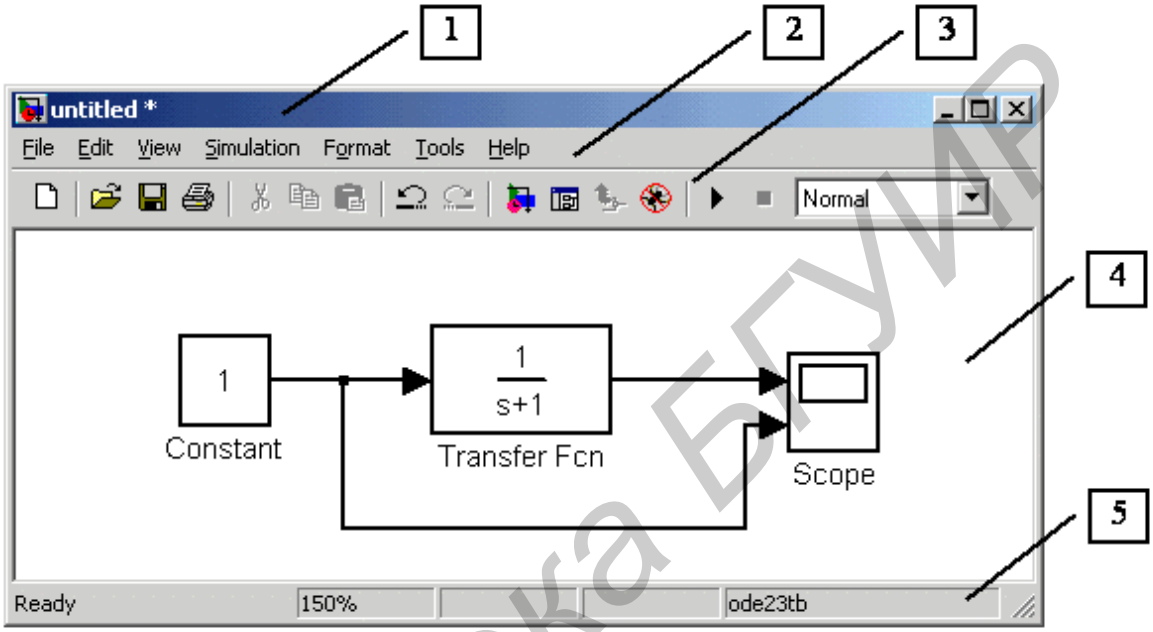



Рисунок 7.5 – Окно модели, содержащее блоки

Для изменения параметров блока необходимо дважды щелкнуть левой клавишей мыши, указав курсором на изображение блока. Откроется окно редактирования параметров данного блока. Следует иметь в виду, что при задании численных параметров в качестве десятичного разделителя должна использоваться точка, а не запятая. После внесения изменений нужно закрыть окно нажатием кнопки **OK**.

После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы. Для соединения блоков необходимо

подвести курсор к выходу блока, затем нажать левую клавишу мыши и, не отпуская, 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 к входу блока. Схема модели, в которой выполнены соединения между блоками, показана на рисунке 7.6.




1 – заголовок; 2 – меню; 3 – панель инструментов;
4 – окно для создания схемы модели; 5 – строка состояния

Рисунок 7.6 – Схема модели

Окно модели содержит следующие элементы (см. рисунок 7.6).

1 Заголовок, с названием модели. Вновь созданной модели 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 модели содержит следующие команды для ее редактирования, настройки и управления процессом расчета, работы файлами и т.п.:

1 **File** (Файл) – работа с файлами моделей.

2 **Edit** (Редактирование) – изменение модели и поиск блоков.

3 **View** (Вид) – управление показом элементов интерфейса.

4 **Simulation** (Моделирование) – задание настроек для моделирования и управления процессом расчета.

5 **Format** (Форматирование) – изменение внешнего вида блоков и модели в целом.

6 **Tools** (Инструментальные средства) – применение специальных средств для работы с моделью (отладчик, линейный анализ и т.п.)

7 **Help** (Справка) – вывод окон справочной системы.

Для большей наглядности модели удобно использовать текстовые надписи. Для создания надписи нужно подвести курсор к желаемому месту надписи и дважды щелкнуть левой клавишей мыши, после чего появится прямоугольная рамка с курсором ввода. Аналогичным образом можно изменить и подписи к блокам моделей. Следует иметь в виду, что поскольку рассматриваемая версия программы (*Simulink* 4) не адаптирована к использованию кириллических шрифтов, их применение 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 4-й (текущей) версии *Simulink* крайне нежелательно.

Перед выполнением расчетов необходимо предварительно задать параметры расчета. Задание параметров расчета осуществляется в панели управления меню **Simulation/Parameters**. Вид панели управления приведен на рисунке 7.7.

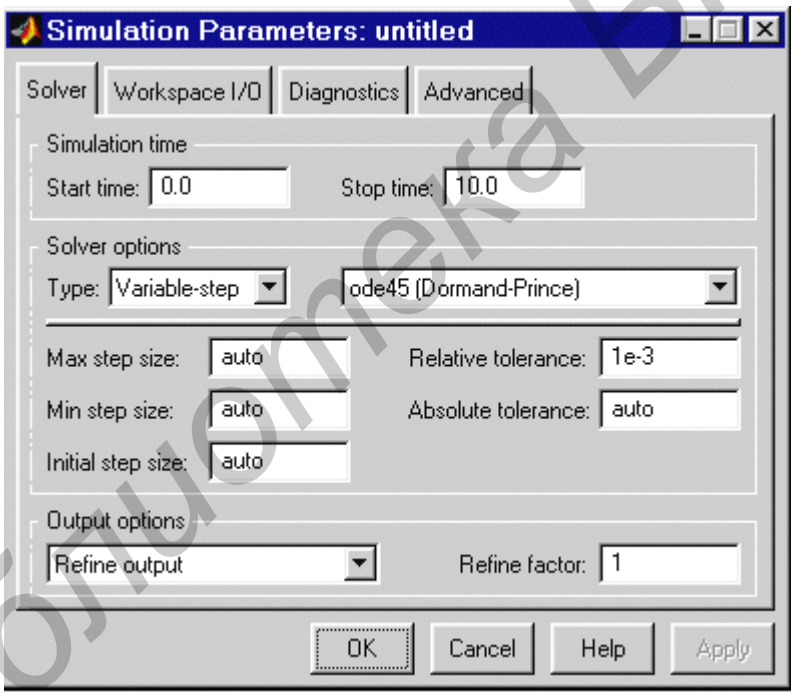



Рисунок 7.7 – Панель управления

Окно настройки параметров расчета имеет четыре 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 7.8).

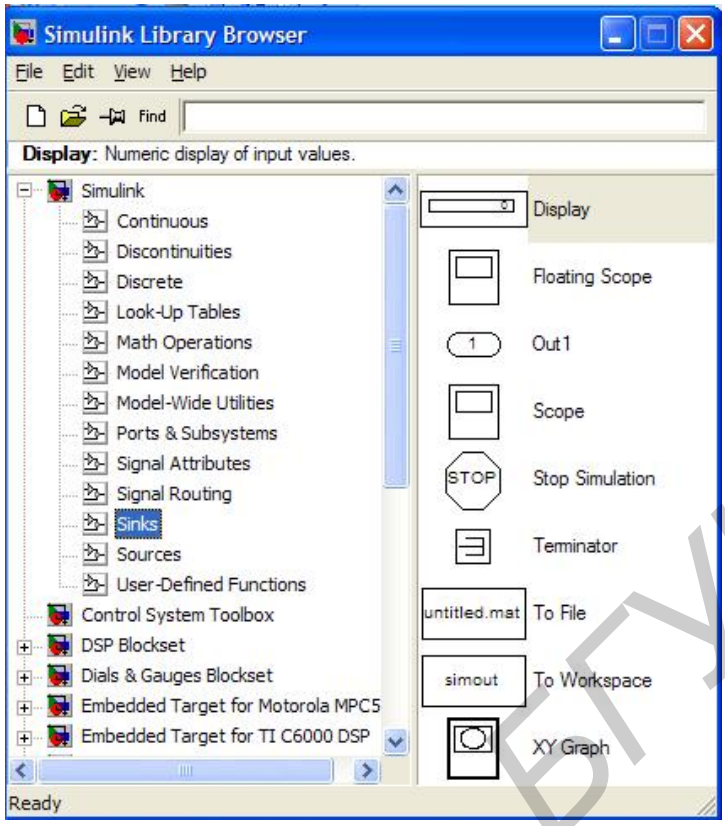



Рисунок 7.8 – Содержимое раздела **Sinks**

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 7.9).

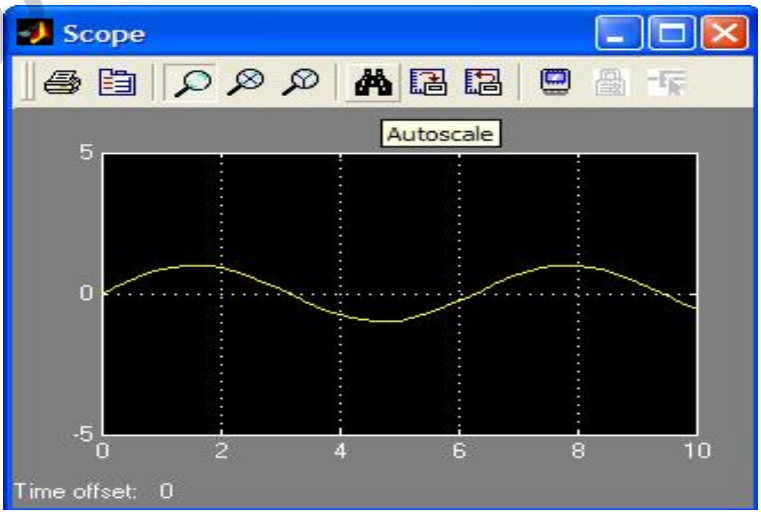



Рисунок 7.9 – Окно виртуального осциллографа

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 7.10).

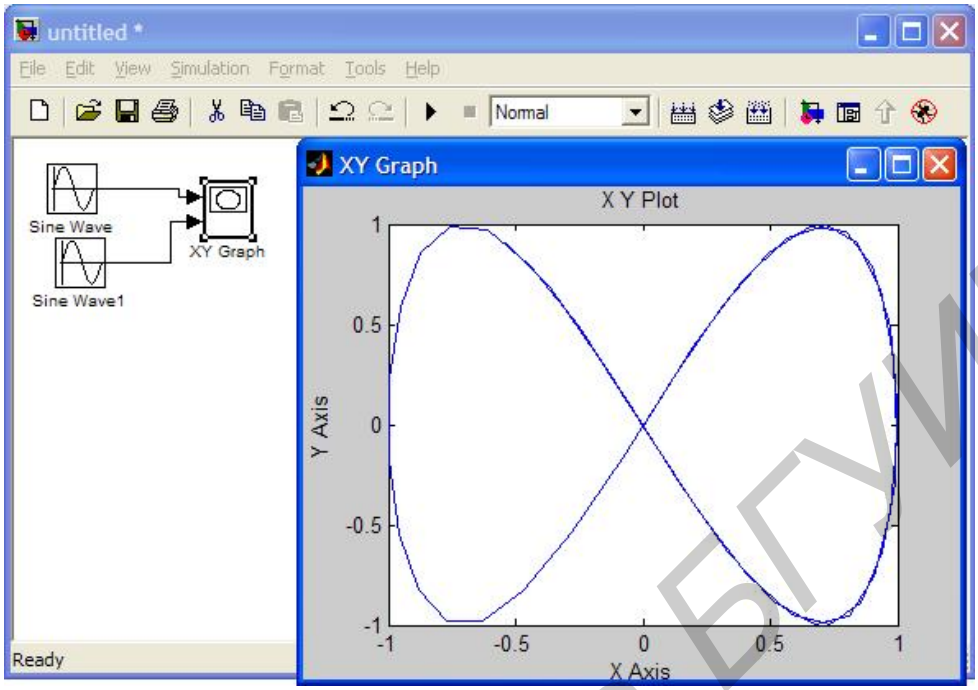



Рисунок 7.10 – График фигуры Лиссажу

Для выполнения арифметических операций используется вкладка **Math Operations** (рисунок 7.11). 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 7.12).

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.

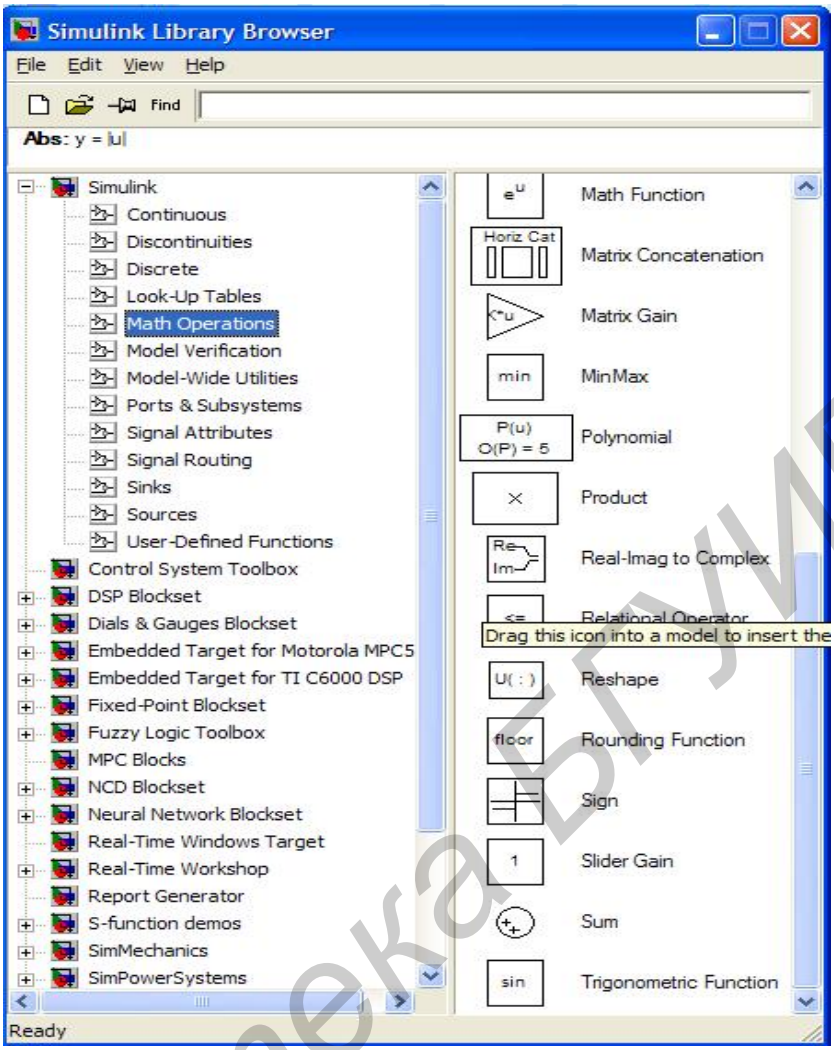



Рисунок 7.11 – Содержимое раздела **Math Operations**

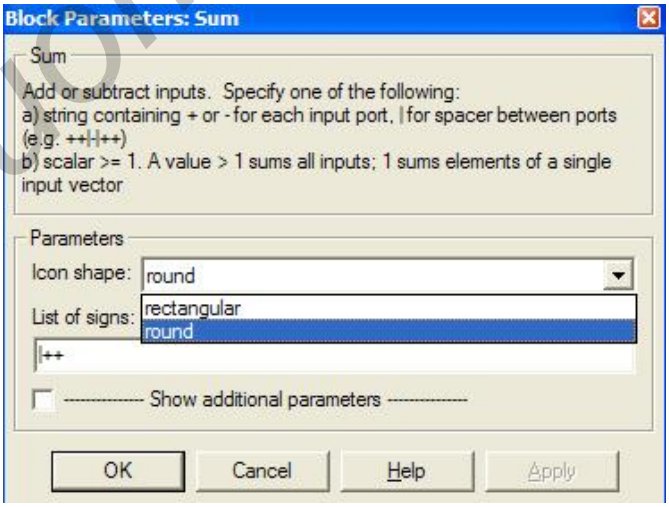



Рисунок 7.12 – Окно настройки блока сложения/вычитания

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

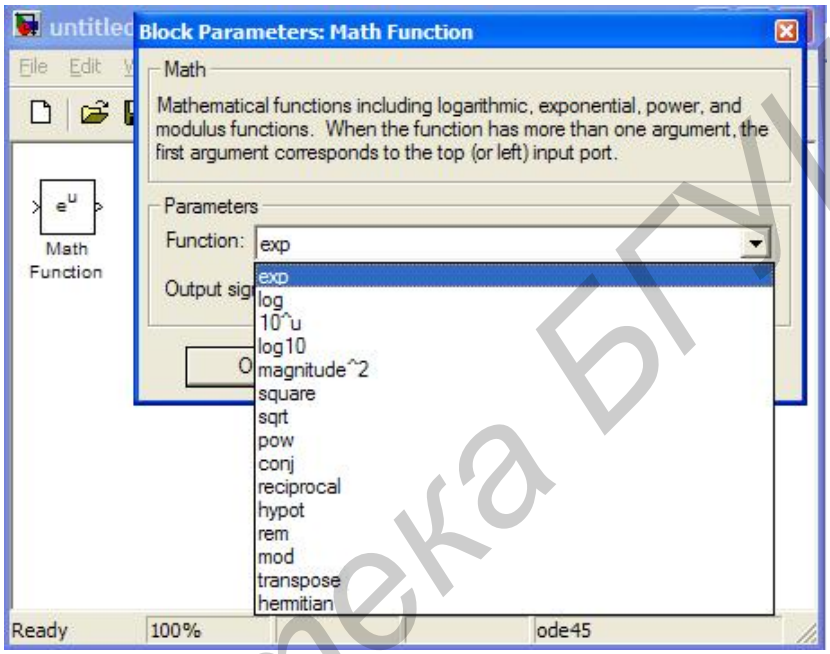



Рисунок 7.13 – Окно параметров блока **Math Function**

## 7.2 Порядок выполнения

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

## 7.3 Содержание отчета

10 Схема модели.
11Результаты моделирования при различных параметрах.

## 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?

# 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-е изд. – СПб. : ГААПСПб., 1994. – 76 с.

6 Потемкин, В. Г. Система MATLAB : справ. пособие / В. Г. Потемкин. – М. : ДИАЛОГ-МИФИ, 1997. – 350 с.

Св. план 2008, резерв

*Учебное издание*

**Русак** Леонид Владимирович
**Снисаренко** Светлана Валерьевна
**Стасевич** Наталья Александровна

# *ОСНОВЫ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ В ПАКЕТЕ MATLAB*

Методическое пособие
к практическим занятиям
по дисциплине «Учебная практика»
для студентов специальности
«Информационные технологии и управление в технических системах»
дневной формы обучения



Редактор Г. С. Корбут
Корректор Е. Н. Батурчик

Подписано в печать 11.01.2010. Формат 60x84 1/16. Бумага офсетная.
Гарнитура «Таймс». Отпечатано на ризографе. Усл. печ. л. 3,95.
Уч.-изд. л. 4,0. Тираж 100 экз. Заказ 266.

Издатель и полиграфическое исполнение: Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6