Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра системы управления аэрокосмической техникой

МАТЕМАТИЧЕСКАЯ ОБРАБОТКА ИНФОРМАЦИИ

Методические указания к выполнению лабораторных работ
для студентов специальности
5В074600 – Космическая техника и технологии

Алматы 2016

СОСТАВИТЕЛИ: М.К. Шимырбаев, Д.Т. Тулекенова. Математическая обработка информации. Методические указания к выполнению лабораторных работ для студентов специальности 5В074600 – Космическая техника и технологии. – Алматы: АУЭС, 2016. – 41 с.

Данная работа представляет собой методические указания для студентов, изучающих дисциплину «Математическая обработка информации». Методические указания включают теоретический материал и задания для лабораторных и контрольных работ, позволяющие освоить основные принципы работы с пакетом MATLAB: арифметические вычисления, работа с массивами, построение графиков и поверхностей, решение уравнений и систем уравнений, вычисление интегралов, пределов, производных функций, решение дифференциальных уравнений.

Ил.24, табл.5, библиогр. – 6 назв.

Рецензент канд.экон.наук, доцент каф. ИС Ж.Г. Аренбаева

Печатается по плану издания некоммерческого акционерного общества «Алматинский университет энергетики и связи» на 2016 г.

©НАО “Алматинский университет энергетики и связи”, 2016 г.

Содержание

Введение 4
1 Лабораторная работа №1. Построение двумерных графиков. 5
2 Лабораторная работа №2. Построение трёхмерных графиков 9
3 Лабораторная работа №3. Решение системы линейных алгебраических уравнений 12
4 Лабораторная работа №4. Решение задач алгебры и анализа.....14
5 Лабораторная работа №5. Решение обыкновенных дифференциальных уравнений 24
6 Лабораторная работа №6. Решение ОДУ с помощью m-file 28
7 Лабораторная работа №7. Моделирование в Simulink 35
Список литературы 40

Введение

MATLAB – одна из старейших, тщательно проработанных и проверенных временем систем автоматизации математических расчетов, построенная на расширенном представлении и применении матричных операций. Однако синтаксис языка программирования системы настолько продуман, что эта ориентация почти не ощущается теми пользователями, которых не интересуют непосредственно матричные вычисления. Матрицы широко применяются в сложных математических расчетах, например, при решении задач линейной алгебры и математического моделирования статических и динамических систем и объектов. Они являются основой автоматического составления и решения уравнений состояния динамических объектов и систем.

В MATLAB реализованы современные численные методы компьютерной математики, используются мощные средства графической визуализации и анимационной графики. Возможности системы весьма обширны, а по скорости выполнения задач она нередко превосходит своих конкурентов и применима для расчетов практически в любой области науки и техники.

1 Лабораторная работа №1. Построение двумерных графиков

1.1 Основные теоретические сведения

Функции одной переменной *y(x)* находят широкое применение в практике математических и других расчетов, а также в технике компьютерного математического моделирования. Для отображения таких функций используются графики в декартовой (прямоугольной) системе координат. При этом обычно строятся две оси – горизонтальная *X* и вертикальная *Y,* и задаются координаты *x* и *y,* определяющие узловые точки функции *y(x).* Эти точки соединяются друг с другом отрезками прямых, т.е. при построении графика осуществляется линейная интерполяция для промежуточных точек. Поскольку MATLAB–матричная система, совокупность точек *y(x)* задается векторами *X* и *Y* одинакового размера.

При построении графиков появляется графическое окно. Иногда оно бывает скрыто ранее имеющимися окнами как системы MATLAB, так и других приложений. Если вы не увидели окна графика, то поищите его в списке открытых окон (приложений) на панели задач или с помощью клавиш [Alt]+[Tab].

Порядок работы при построении графика функции следующий:

1) Задать значения аргумента функции.
2) Задать функцию.
3) Построить график.
4) Отформатировать график.
5) Добавить на график дополнительные элементы.

Для построения графиков функций в MATLAB служит команда plot, имеющая несколько вариантов записи (*x* – аргумент функции, *y* – функция):

- plot(x,y) – строит график одной функции;
- plot(x,y,s) – строит график функции с заданным типом и цветом линии и точек (s – строковая константа);
- plot(x,y1,x,y2,…) – строит графики нескольких функций в одной системе координат;
- plot(x,y1,s1,x,y2,s2,…) – строит графики нескольких функций в одной системе координат с заданным типом и цветом линии и точек.

С помощью строковой константы *s* можно изменять цвет линии, представлять узловые точки различными отметками (точка, окружность, крест и т. д.) и менять тип линии графика. Значения строковой константы представлены в таблице 1-3.

Таблица 1 - Цвет линии

Код	Описание	Код	Описание
Y	Желтый	G	Зеленый
M	Фиолетовый	B	Синий
C	Голубой	W	Белый
R	Красный	K	Черный

Таблица 2 - Тип точки

Код	Описание	Код	Описание
.	Точка	D	Ромб
0	Окружность	V	Треугольник
X	Крест	<	Треугольник
+	Плюс	>	Треугольник
*	Звездочка	P	Пятиугольник
S	Квадрат	H	Шестиугольник

Таблица 3 - Тип линии

Код	Описание
-	Сплошная
-.	Штрихпунктир
--	Штриховая

Пример. Построить график функции $y=\sin x$ на отрезке [–4; 4], шаг 0,2.
Порядок ввода:

```
>> x=–4:0.2:4;
>> y=sin(x);
>> plot(x,y).
```

Пример. Построить график этой же функции штриховой линией фиолетового цвета, отметив точки ромбами.
Порядок ввода:

```
>> x=–4:0.2:4;
>> y=sin(x);
>> plot(x,y,'dm--').
```

В результате каждого построения получатся графики, рисунок 1.

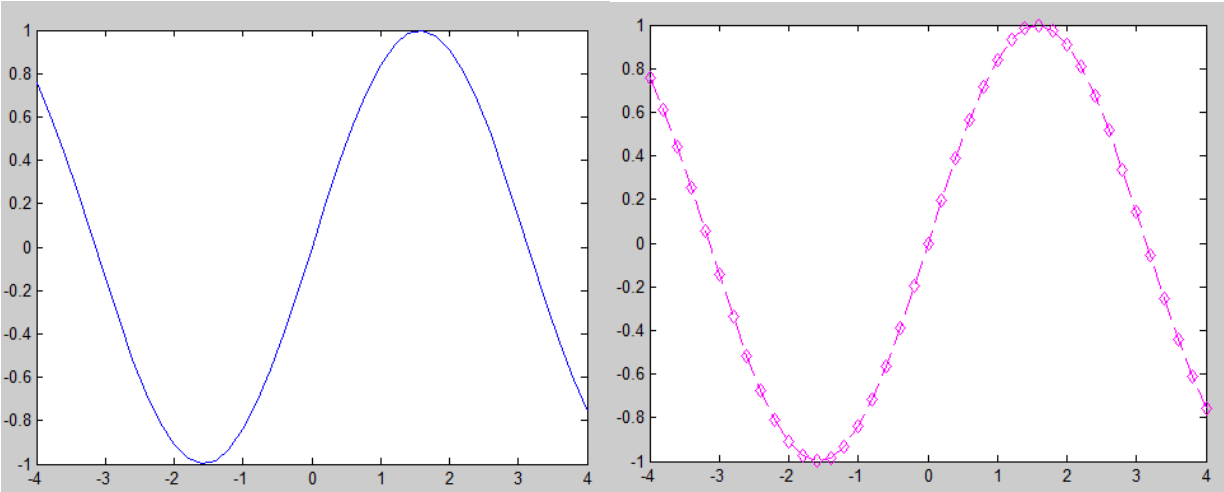

Рисунок 1 – Графики функции *y*=sin*x* стандартного вида и с заданными параметрами

Пример. Построить в одной системе координат графики функций *y*=sin*x* и *z*=cos*x* на отрезке [–5; 5] с шагом 0,2.

Порядок ввода:

```
>> x=–5:0.2:5;
>> y=sin(x);
>> z=cos(x);
>> plot(x,y,'-.+r',x,z,'--ok').
```

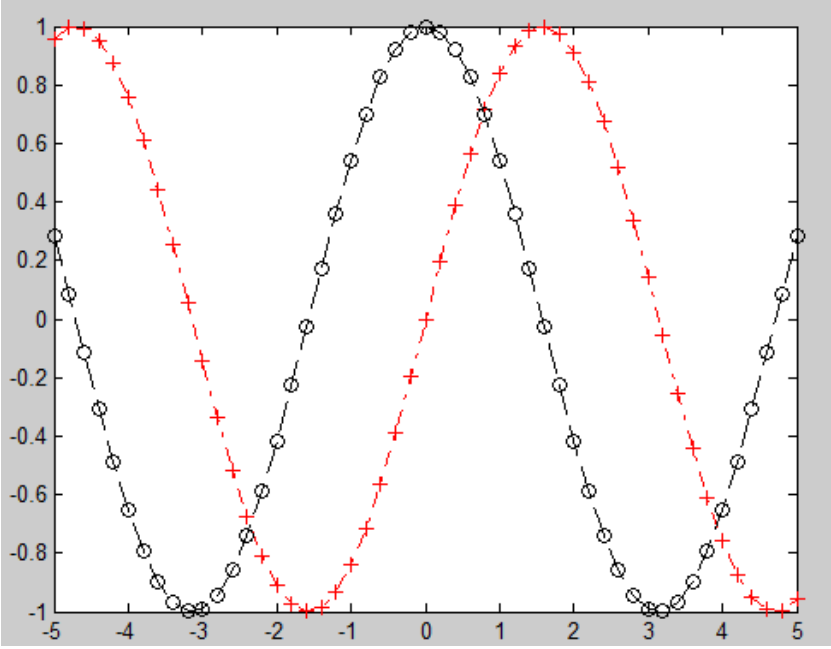

Рисунок 2 – Графики функций *y*=sin*x* и *z*=cos*x*

1.2 Добавление объектов на график

Дополнительно на график можно нанести различные объекты с помощью соответствующих команд пункта меню Insert:

X Label – подпись оси *X*;
Y Label – подпись оси *Y*;
Title – заголовок графика;
Legend – легенда;
Line – линия;
Arrow – стрелка;
Text Arrow – стрелка с текстом;
Double Arrow – двойная стрелка;
TextBox – текстовое поле;
Rectangle – прямоугольник;
Ellipse – эллипс;
Axes – оси.

Пример. Построить и отформатировать графики функций $y=2x^2+3x^2$ и $z=3|3x-5|$ на отрезке [–5; 5] с шагом 0,1.
Порядок ввода:

```
>> x=–5:0.1:5;
>> y=2*x.^3+3*x.^2;
>> z=3*abs(3*x–5);
>> g=plot(x,y,x,z)
>> set(g,'linewidth',3,'linestyle','--')
>> title('Plots')
>> xlabel('x')
>> ylabel('y, z')
>> grid
>> legend('y','z')
>> text(–2.5,80,'z')
>> line([–2.4 –2.2],[70 50])
>> text(–1.5,–50,'y')
>> line([–1.4 –1.2],[–40 5])
```

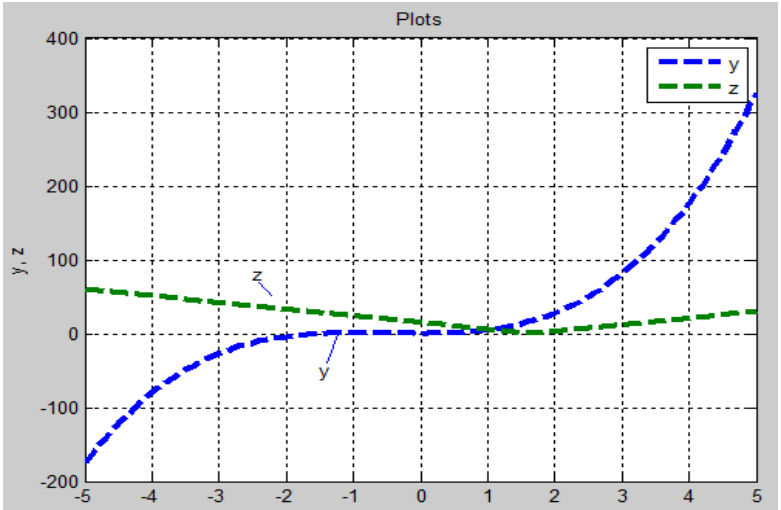

Рисунок 3 – Графики функций $y=2x^2+3x^2$ и $z=3|3x-5|$

1.3 Содержание отчета

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

1.4 Контрольные вопросы

1. Перечислите основные команды MATLAB для построения графиков.
2. Напишите команды пункта меню Insert.

1.5 Варианты заданий

Построить графики функций в одной системе координат, отформатировав их с помощью окна свойств графика по образцу:

1) $f = \ln x + x^2$, $x \in [1; 7]$, шаг 0,4;
2) $f = x^2$, $y = \sin x$, $x \in [-5; 5]$, шаг 0,5;
3) $f = \sin x^2 - \cos x$, $y = x^2 - 3$, $x \in [-4; 4]$, шаг 0,3;
4) $f = \sin x^2 + \cos x$, $y = x^2 - 4$, $z = x \in 0{,}8$, $x \in [-4; 4]$, шаг 0,4.
5) $x(t) = t \cdot \cos t$, $y(t) = t \cdot \sin t$, $t \in [0; 10]$, шаг 1/10;
6) $b = 3$, $x(t) = b \cdot \cos 3t$, $y(t) = b \cdot \sin 3t$, $t \in [0; 20]$, шаг 5/12;
7) $a = 4$, $x(t) = a \cdot (t^2 - 1)/(t^2 + 1)$, $y(t) = a \cdot t \cdot (t^2 - 1)$, $t \in [-10; 10]$, шаг 0,5
8) $x(t) = t \cdot \cos t$, $y(t) = t \cdot \sin t$, $t \in [0; 10]$, шаг 1/10;
9) $f = \ln|x + 2{,}5| + 1$, $x \in [-5; 5]$, шаг 1;
10) $f = \sin x^3 + \cos x$, $y = x^2 - 6$, $z = x \in 0{,}1$, $x \in [-5; 5]$, шаг 0,01.

2 Лабораторная работа №2. Построение трёхмерных графиков

2.1 Основные теоретические сведения

Построение трехмерных графиков (поверхностей) во многом похоже на построение двумерных графиков. Для этого используется команда plot3, которая имеет несколько вариантов записи:

- plot3(X,Y,Z) – строит поверхность по точкам, координаты которых берутся из матриц *X, Y, Z*;
- plot3(X1,Y1,Z1,X2,Y2,Z2,...) – строит несколько поверхностей *Z*1, *Z*2 и т. д.;
- plot3 (X,Y,Z,S) – строит поверхность заданным типом и цветом линии и точек (*S* – строковая константа, задающая тип и цвет линии и точек);
- plot3(X1,Y1,Z1,S1,X2,Y2,Z2,S2,X3,Y3,Z3,S3,...) – строит несколько поверхностей *Z*1, *Z*2 и т.д. заданным типом и цветом линии и точек (*S*1, *S*2 и т.д.).

Значения строковой константы *S* приведены ранее (таблицы 4-6).

Особенно наглядное представление о поверхностях дают сетчатые графики, использующие функциональную закраску ячеек. Например, цвет окраски поверхности *z(x, y)* может быть поставлен в соответствие с высотой *z*

поверхности с выбором для малых высот темных тонов, а для больших – светлых. Для построения таких поверхностей используются команды класса surf:

- surf(X,Y,Z) – строит цветную параметрическую поверхность по данным матриц *X, Y* и *Z*;
- surfc(X,Y,Z) – строит цветную параметрическую поверхность по данным матриц *X, Y, Z* и проекцию фигуры на опорную плоскость.

Иногда бывают полезны графики трехмерных слоеных поверхностей, как бы состоящие из тонких пластинок – слоев.

Такие поверхности строит функция waterfall:

waterfall(X,Y,Z) – строит поверхность, состоящую из тонких пластинок – слоев.

Порядок построения трехмерных графиков следующий:

1) Задать матрицы *X* и *Y* на основе диапазонов значений переменных *x* и *y* с помощью команды преобразования диапазонов значений переменных в соответствующие матрицы:

[X,Y]=meshgrid(диапазон1, дипазон2);

если диапазоны одинаковые, то

[X,Y]=meshgrid(диапазон);

2) Задать функцию *Z*(*X*,*Y*).

3) Построить поверхность нужного вида с помощью соответствующей команды.

4) Отформатировать график.

Отформатировать трехмерный график можно с помощью окна свойств графика или специальных команд (см. построение двумерных графиков). Кроме того, для форматирования цветных поверхностей есть дополнительные команды:

- colormap(gray) – задает окраску тонами серого цвета;
- shading interp – устраняет изображения линий и задает интерполяцию для оттенков цвета поверхности;
- colorbar – выводит на экран цветовую шкалу.

Пример. Построить график функции $z(x,y)=x^2+y^2$ на отрезке [–3; 3] с шагом 0,15.

Порядок ввода:

```
>> [X,Y]=meshgrid([–3:0.15:3]);
>> Z=X.^ 2+Y.^2;
>> plot3(X,Y,Z);
>> grid.
```

Пример. Построить цветной график функции $g(x,y)=5x\cdot\sin x-3{,}5y^2$ и его проекцию на отрезке [–3; 3] с шагом 0,15.

Порядок ввода:

```
>> [X,Y]=meshgrid([–3:0.15:3]);
>> G=5·X.·sin(Y)–3.5·Y.^2;
```

>> surfc(X,Y,G).

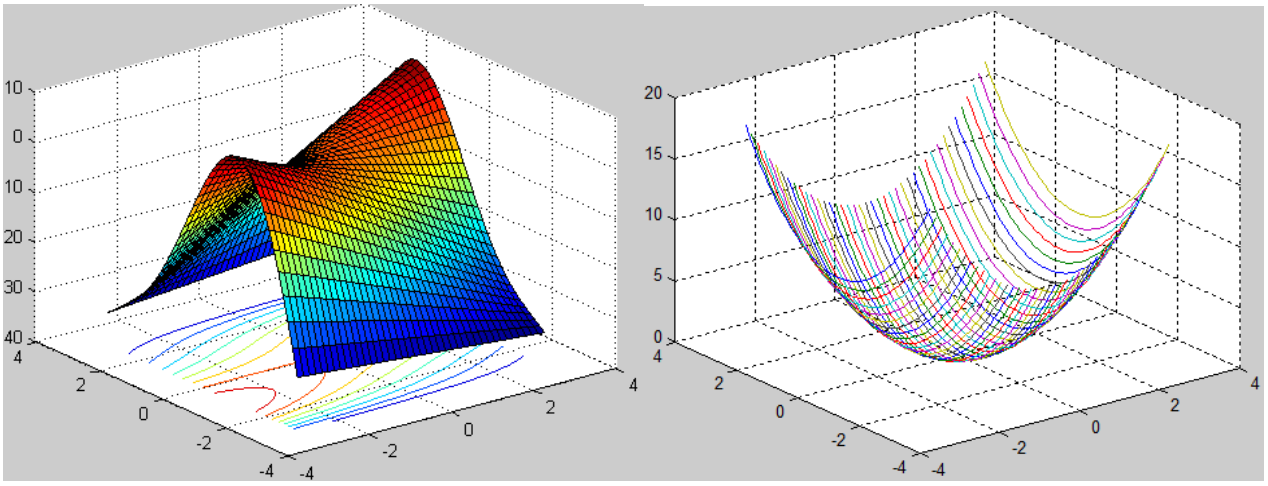

Рисунок 4 – Графики функций $z(x,y)= x^2 + y^2$ и $g(x,y)=5x\cdot\sin x–3,5\ y^2$

2.2 Содержание отчета

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

2.3 Контрольные вопросы

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

2.4 Варианты заданий

Построить цветные поверхности функции на заданных отрезках и отформатировать их по образцу:

1) $z=2x\sin x+3y\cos y$ на отрезке [–2; 2], шагом 0,2;
2) $z=2x\sin x+3y\cos y$ на отрезке [–5; 5], шагом 0,5;
3) $z=x^2+y^2$ на отрезке [–2; 2] с шагом 0,2;
4) $g = \frac{2x^2+y}{3} \cdot (3\cos 3y+\sin y)$ на отрезке [–4; 4] с шагом 0,2;
5) $\frac{x^2}{10} + \frac{y^2}{5} = 2z$ на отрезке [–4; 4] с шагом 0,2;
6) $\frac{x^2}{8} - \frac{y^2}{8} = 2z$ на отрезке [–4; 4] с шагом 0,2;
7) $\frac{x^2}{50} + \frac{y^2}{50} + \frac{z^2}{50} = 1$ на отрезке [–5; 5], шагом 0,5;

8) $f = \frac{2y^2+x}{5} \cdot (5\cos 5y + \sin y)$ на отрезке [–5; 5] с шагом 0,5;

9) $z = 5x\sin x + 2y\cos y$ на отрезке [–3; 3], шагом 0,3;

10) $\frac{x^2}{8} + \frac{y^2}{4} = 4z$ на отрезке [–5; 5] с шагом 0,5.

3 Лабораторная работа №3. Решение системы линейных алгебраических уравнений

3.1 Основные теоретические сведения. Матричный способ решения систем линейных уравнений

Решение систем линейных уравнений относится к самой массовой области применения матричных методов. Как известно, обычная система линейных уравнений имеет вид:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1; \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2; \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \quad (1)$$

Рассмотрим различные способы решения систем уравнений.

Пусть *A* – матрица коэффициентов при неизвестных;
B – вектор-столбец свободных членов;
X – вектор-столбец неизвестных, т.е.

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad B = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}. \quad (2)$$

Тогда систему уравнений можно записать в матричном виде *A*·*X*=*B*. Решение системы уравнений имеет вид *X*=*A*^–1·*B*, где *A*^–1 – матрица, обратная матрице *A*.

Пример. Решить систему уравнений

$$\begin{cases} 2x_1 - x_2 = 4; \\ 5x_1 + 2x_2 = 3; \end{cases}$$

Порядок ввода:

```
>> A=[2 –1;5 2]
>> B=[4; 3]
>> X=A^–1·B
```

В результате получим *x*1=1,222, *x*2=–1,556.

3.2 Решение систем уравнений с помощью функции solve

Для решения систем линейных или нелинейных уравнений в MATLAB существует специальная функция solve. Чтобы решить систему уравнений с помощью этой функции, нужно:

1) Определить символьные переменные (неизвестные системы уравнений).

2) Вычислить неизвестные по формуле [x1,x2,…]= solve('уравнение1','уравнение2',…).

3) Вывести найденное решение с заданной точностью, используя функцию vpa(переменная, число знаков).

Пример. Решить систему уравнений

$$\begin{cases} 2x_1 - x_2 = 4; \\ 5x_1 + 2x_2 = 3. \end{cases}$$

Порядок ввода:

```
>> syms x1 x2
>> [x1,x2]=solve('2*x1–x2=4','5*x1+2*x2=3');
>> vpa(x1,4)
>> vpa(x2,4)
```

В результате получим *x*1=1,222, *x*2=–1,556.

3.3 Содержание отчета

1. Перечислите способы решения СЛАУ.
2. Листинг программы для вывода графиков функций.

3.4 Контрольные вопросы

1. С помощью какой команды осуществляется построение графиков декартовой системах координат?
2. Для чего служит функция vpa?
3. Для чего используется команда solve?

3.5 Варианты заданий

1. Решить системы линейных уравнений матричным способом, методом Крамера и с помощью функции solve:

1) $\begin{cases} 2x_1 + x_2 + x_4 = 8; \\ x_1 - 3x_2 + 2x_3 + 4x_4 = 9; \\ -5x_1 - x_3 - 7x_4 = -5; \\ x_1 - 6x_2 + 2x_3 + 6x_4 = 0. \end{cases}$

3) $\begin{cases} 2x_1 + 3x_2 - 4x_3 + x_4 = 3{,}1; \\ 0{,}1x_1 - 2x_2 - 5x_3 + x_4 = 2; \\ 0{,}15x_1 - 3x_2 + x_3 - 4x_4 = 1; \\ 10x_1 + 2x_2 - x_3 + 2{,}1x_4 = -4{,}7. \end{cases}$

2) $\begin{cases} 3x_1 - x_2 = 5; \\ -2x_1 + x_2 - x_3 = 0; \\ 2x_1 - x_2 + 4x_3 = 15. \end{cases}$

4) $\begin{cases} 4x_1 + 0{,}24x_2 - 0{,}08x_3 = 8; \\ 0{,}09x_1 + 3x_2 - 0{,}15x_3 = 9; \\ 0{,}04x_1 - 0{,}08x_2 + 4x_3 = 20. \end{cases}$

5) $\begin{cases} 2xy = 16; \\ x^2 + y = 8. \end{cases}$

6) $\begin{cases} x^2 - y = 3; \\ x + 2y = 4. \end{cases}$

7) $\begin{cases} x^3 - 2y = 1; \\ x^2 + 3y = 2. \end{cases}$

8) $\begin{cases} 3x^3 + 2x^2 + y = 5; \\ 2x^2 - 3x + y = 4. \end{cases}$

2. Решить системы нелинейных уравнений с помощью функции solve:

1) $\begin{cases} x^3 - 2y = 1; \\ 3x + 3y^2 = 2. \end{cases}$

3) $\begin{cases} \sqrt{x+y} + \sqrt[4]{x-y} = 8; \\ \sqrt[4]{x^3 + x^2y - xy^2 - y^3} = 12. \end{cases}$

2) $\begin{cases} uvx^2 = 8; \\ vx^2w = 24; \\ x^3wu = 12; \\ u + v + w = x + 4. \end{cases}$

4) $\begin{cases} x^3 - 2x^2 + 3y + z = 2; \\ 2x^2 - x + y^2 - z^2 = 1; \\ y^3 - 2z^2 + 3z = 1. \end{cases}$

4 Лабораторная работа №4. Решение задач алгебры и анализа

4.1 Основные теоретические сведения

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

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

где A – заданная квадратная матрица размером NxN;

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

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

Решение уравнения $F(x)=0$, или нахождение нулей функции, осуществляется с помощью функции fzero(name, x_0) . В качестве первого аргумента ей передается имя функции, задающей исходное уравнение, вторым

аргументом служит начальное приближение к корню. Определим, в качестве примера, нули функции cos(x) на отрезке от 0 до pi. В качестве начального приближения примем x_0=1.

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

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

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

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

После этого можно воспользоваться функцией fzero:

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

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

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

В задаче о нахождении нулей функции сложным моментом является нахождение начального приближения к нулю функции, а также априорная оценка их количества. Поэтому важно параллельно с применением функций

типа roots или fzero визуализировать поведение искомых функций на том или ином отрезке значений аргумента.

В системе MATLAB имеются специальные функции для поиска минимума заданных функций. При этом возможен поиск минимума как для функции одной вещественной переменной, так и для функций многих переменных. Для функций одной переменной их минимумы разыскивает функция xmin =fmin(name, x_0, x_1).

Здесь name представляет имя функции, у которой находятся минимумы, а x_0 и x_1 задают отрезок поиска. Для поиска минимума функции нескольких переменных применяется функция fmins: xmins = fmins(name, x_0). Здесь name является именем функции нескольких переменных, для которой ищется минимум, а x_0 – это вектор ее аргументов, с которого начинается поиск. Для иллюстративного примера создадим простую функцию двух переменных имеющую минимумом точку (0,0).

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

После этого можно вызвать функцию fmins, которая приближенно находит вектор xmin координат точки минимума:

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

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

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

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

Обычно для достижения высокой точности требуется выполнять интегрирование с очень малыми шагами, а контроль достигнутой точности осуществлять путем сравнения последовательных результатов. При одном и том же шаге интегрирования методы более высоких порядков точности достигают более точных результатов. В системе MATLAB методы интегрирования более высоких порядков точности реализуются функциями:

quad (метод Симпсона) и quad8 (метод Ньюто-на-Котеса 8-го порядка точности).

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

Вычислим интеграл вида $\int_0^1 \int_1^2 (x\sin(y) + y\sin(x))dxdy$.

Оформим подынтегральную mat–функцию и вызовем функцию dblquad:

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

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

» syms x a b.

Конструирование символьных функций от переменных класса sym object производится с использованием обычных арифметических операций и обозначений для встроенных математических функций. Запись формулы для выражения в одну строку не всегда удобна, более естественный вид выражения выводит в командное окно функция pretty:

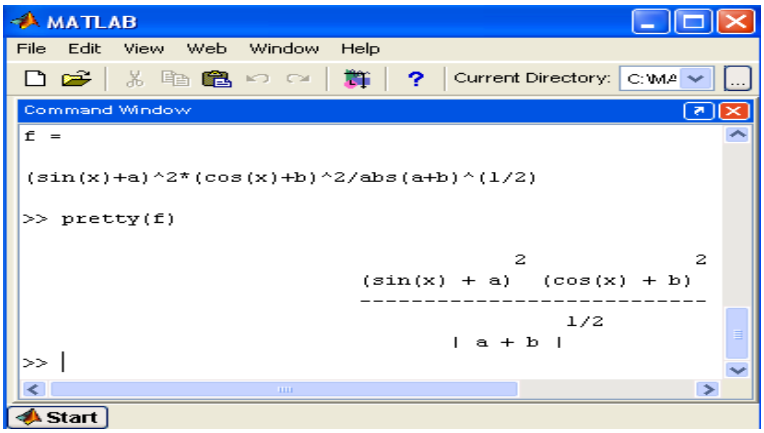

Рисунок 5 – Демонстрация работы функции pretty

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

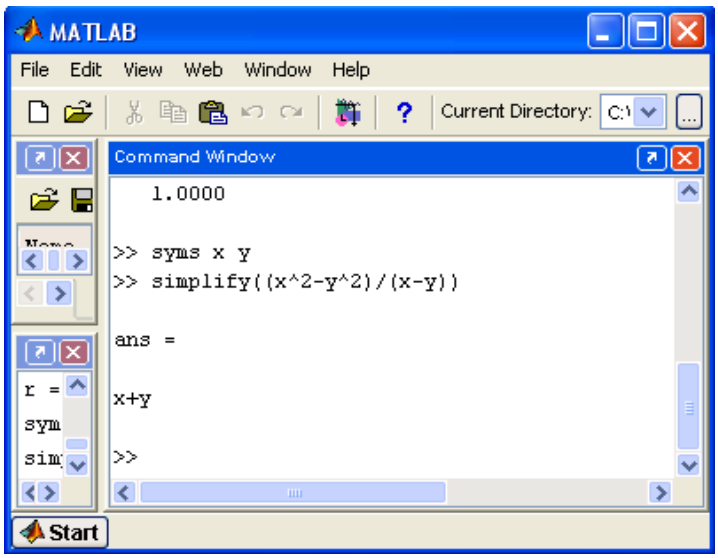

Рисунок 6 – Демонстрация работы функции simplify

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

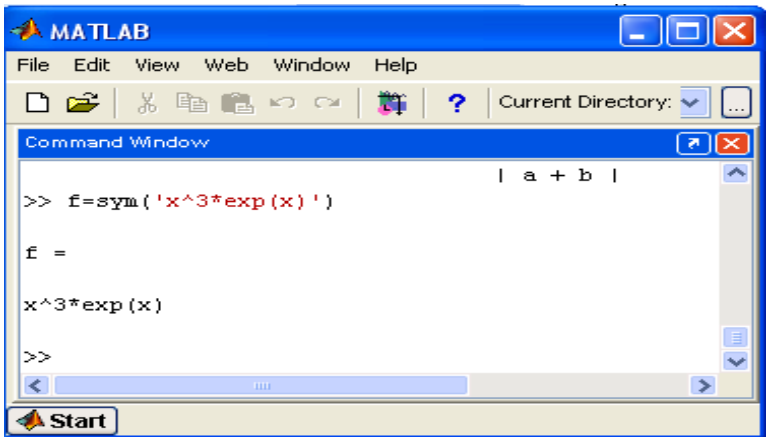

Рисунок 7 – Демонстрация создания символьной переменной без предварительного объявления

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:

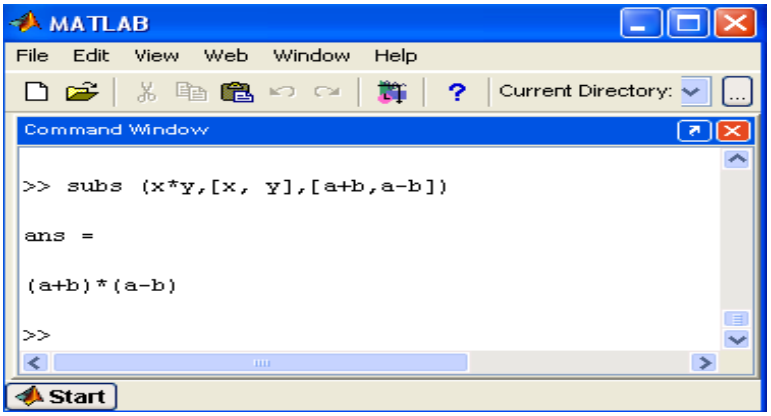

Рисунок 8 – Демонстрация работы функции subs

Symbolic Math Toolbox позволяет работать как с неопределенными интегралами, так и с определенными. Неопределенные интегралы от символьных функций вычисляются при помощи функции int, в качестве входных аргументов указываются символьная функция и переменная, по которой происходит интегрирование (рисунок 5).

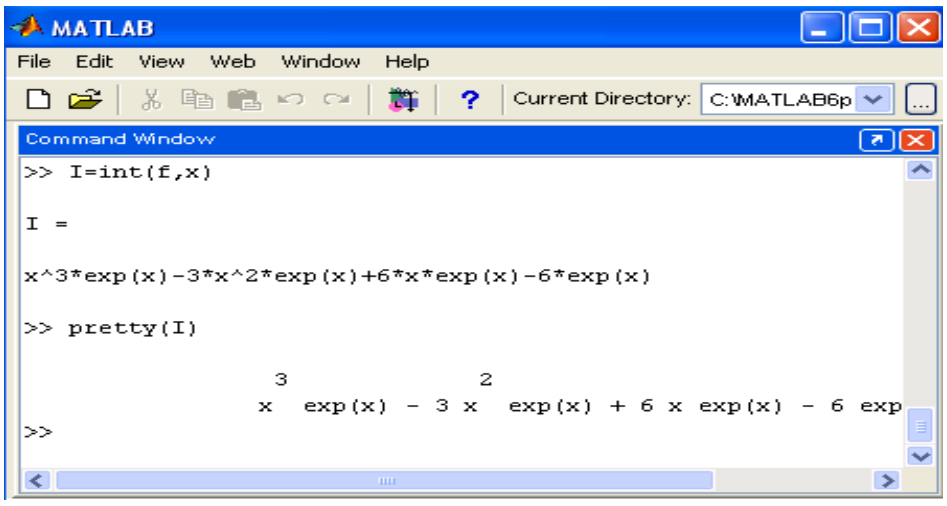

Рисунок 9 – Демонстрация работы функции int

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

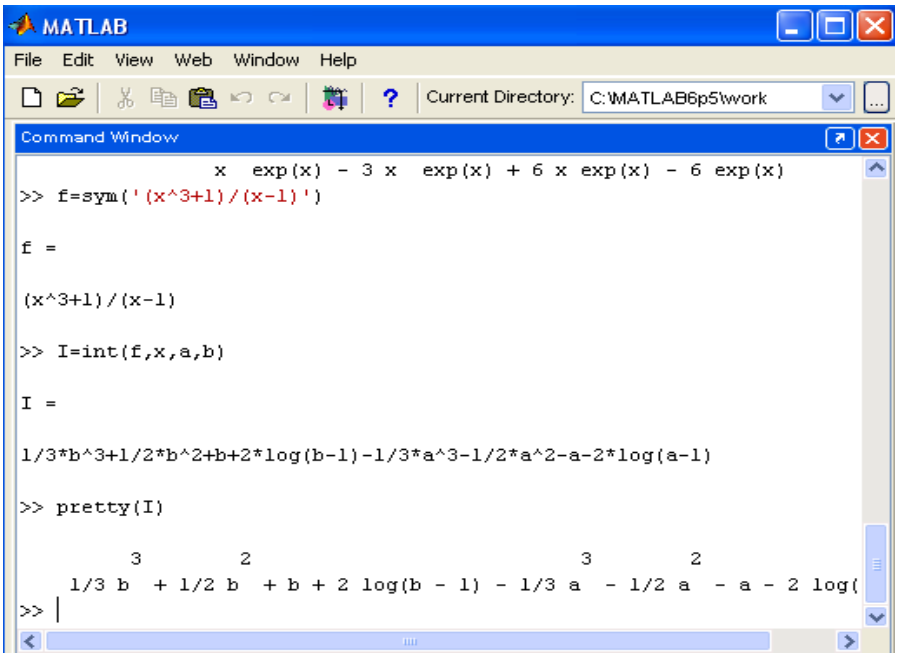

Рисунок 10 – Демонстрация работы функции int для нахождения определенного интеграла

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 10. При запуске выводятся три автономных окна: два графически и управляющее. Управляющее окно содержит два поля для ввода функций, поле ввода пределом изменения переменной x, поле ввода масштабирующего коэффициента.

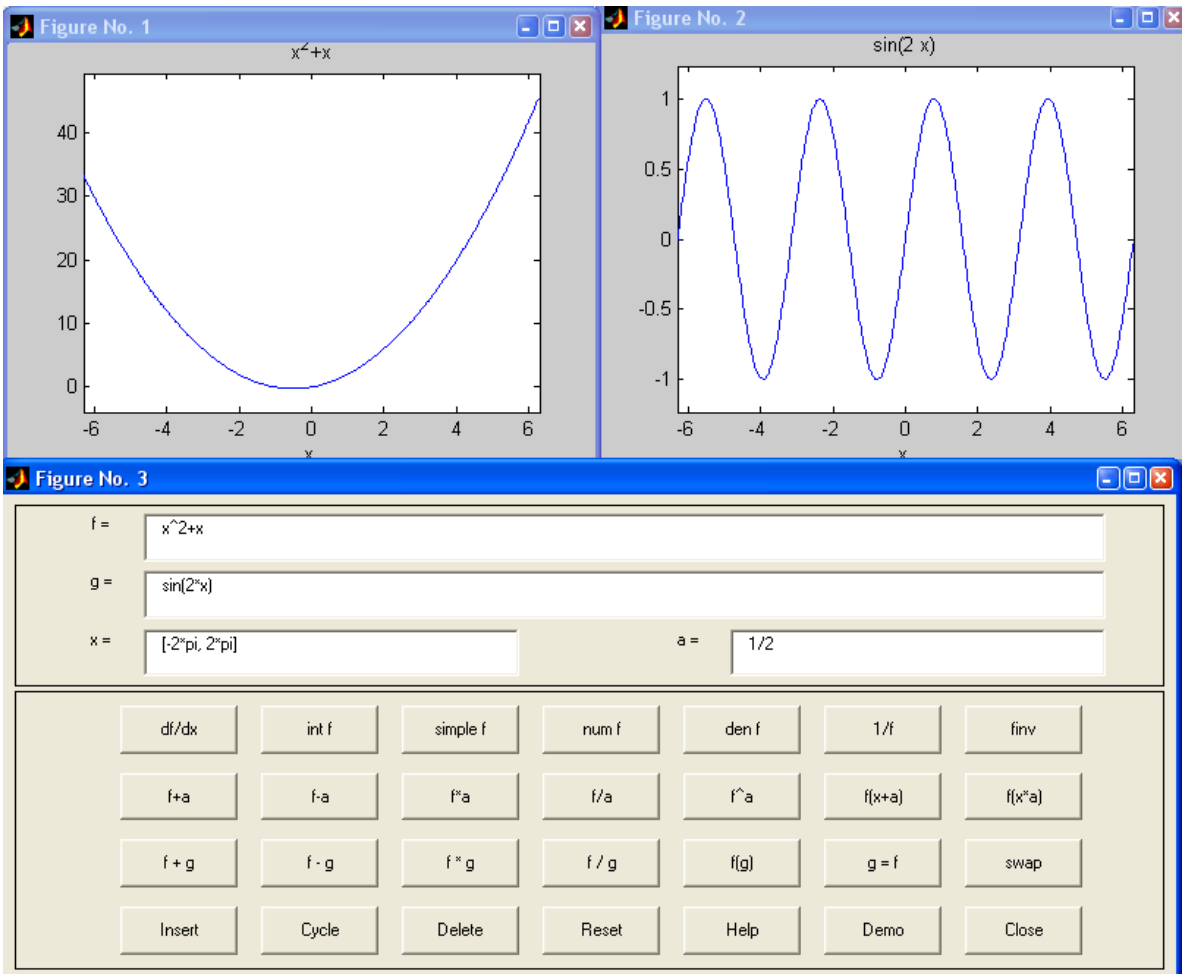

Рисунок 11 – Интерфейс приложения funtool

4.2 Содержание отчета

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

4.3 Контрольные вопросы

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

4.4 Варианты задании

1. Составить и отладить программы для нахождения корней уравнения $f_1(x) = 0$ и $f_2\ (x) = 0$ и вывести графики функции на основании задания из таблицы 1.

2. Найти определенный интеграл для подынтегральной функции, заданной в таблице 2.

3. Найти определенный интеграл для подынтегральной функции, заданной в таблице 2 с использованием пакета символьных вычислений.

Таблица 1 – Варианты заданий для нахождения корней уравнения

№ варианта	$f_1(x)$ – полином 3-ей степени с коэффициентами a				$f_2(x)$
1	2	3	4	5	6
1	0	-1	4	-1	$0.2e^{x} - 20$
2	0	2	-2	-15	$40 \cdot \lvert\cos(x)\rvert$
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 2 – Варианты функций для нахождения значения интеграла

№ варианта	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
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

5 Лабораторная работа №5. Решение обыкновенных дифференциальных уравнений

5.1 Основные теоретические сведения. Решение ОДУ с помощью команды dsolve

Для решения обыкновенных дифференциальных уравнений в символьном виде в MATLAB существует команда dsolve. Она может быть использована, если решение существует в аналитическом виде.

Практически это означает, что командой dsolve можно пользоваться только при поиске решения линейного дифференциального уравнения (или системы линейных уравнений).

Пример. Решить дифференциальное уравнение $x\ dx/dt$=-0,5 x с начальным условием $x(0)$=10. Построить график решения в интервале [–0,5; 7].

Порядок ввода:

```
>> x=dsolve('Dx=–0.5*x','x(0)=10');
>> ezplot(x,[–0.5,7]);
>> grid.
```

В результате получим функцию x=10e–1/2t и график

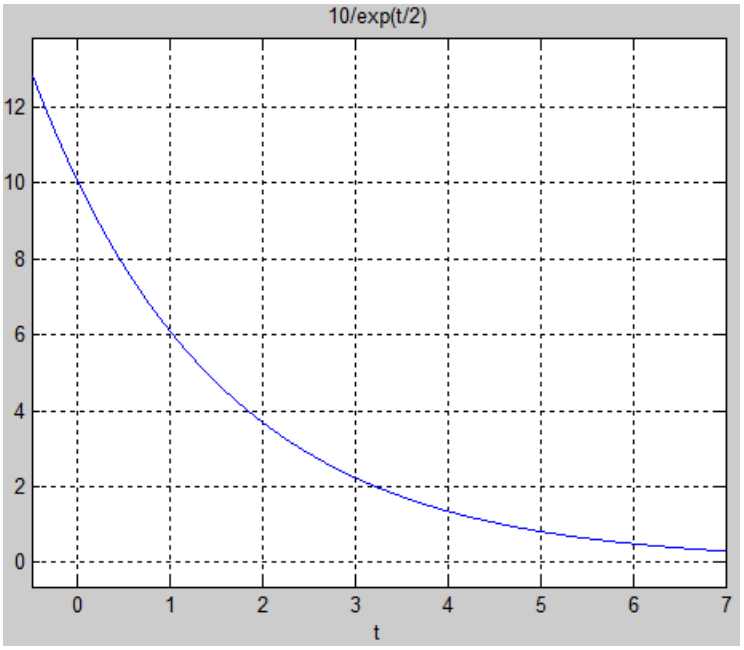

Рисунок 12 – График функции-решения уравнения

Пример. Решить дифференциальное уравнение 2-го порядка с нулевыми начальными

$$2{,}5\frac{d^2x}{dt^2} + 3\frac{dx}{dt} + 5x = 12$$

условиями и построить график решения в интервале [–0,2; 9].

Порядок ввода:

```
>> x=dsolve('2.5D2x+3Dx+5x=12','Dx(0)=0','x(0)=0');
>> ezplot(x,[–0.2 9]);
>> grid.
```

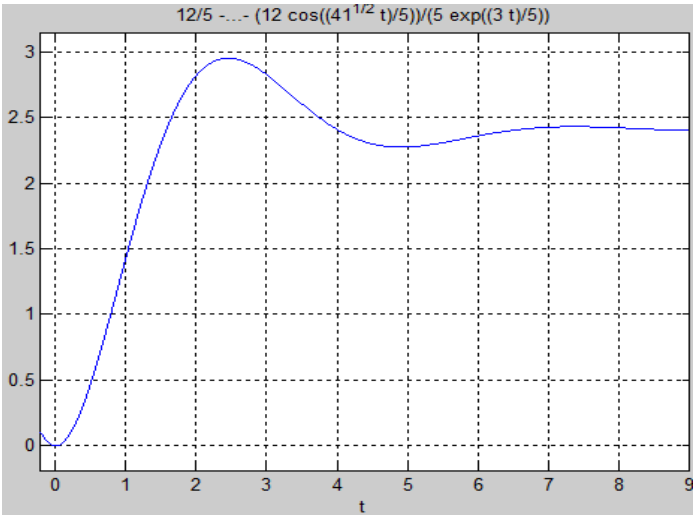

Рисунок 13 – График функции-решения уравнения

Пример. Построить график решения дифференциального уравнения 3-го порядка с нулевыми начальными условиями в интервале [–0,2; 21]:

$$1.5\frac{d^3x}{dt^3} + 4\frac{d^2x}{dt^2} + 3\frac{dx}{dt} = 12.$$

Порядок ввода:

```
>> x=dsolve('1.5D3x+4D2x+3Dx+5x=12','D2x(0)=0','Dx(0)=0','x(0)=0');
>> ezplot(x,[–0.2 21]);
>> grid.
```

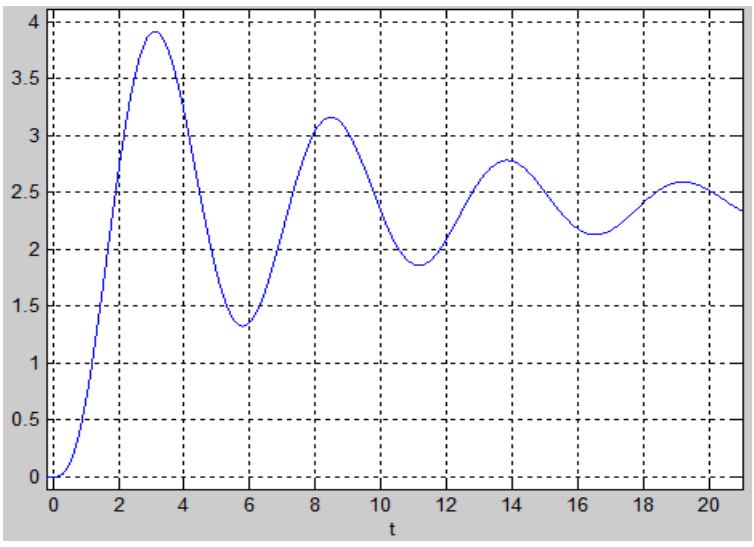

Рисунок 14 – График решения уравнения

Пример. Решить неоднородную систему дифференциальных уравнений третьего порядка с нулевыми начальными условиями:

$$\begin{cases} \dfrac{dx_1}{dt} = -x1 + 10; \\ \dfrac{dx_2}{dt} = 2 \cdot x1 - x3; \\ \dfrac{dx_3}{dt} = 2{,}5 \cdot x1 + 3 \cdot x2 - 2 \cdot x3. \end{cases}$$

Порядок ввода:

>> [x1,x2,x3]=dsolve('Dx1= –x1+10', 'Dx2=2·x1–x3', 'Dx3=2.5·x1–3·x2–2·x3','x1(0)=0','x2(0)=0','x3(0)=0').

5.2 Содержание отчета

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

5.3 Контрольные вопросы

1. Какие существуют методы решения систем дифференциальных уравнений?
2. Для чего служит функция ode?

5.4 Вариант задании

1. Решить дифференциальные уравнения старшего порядка при заданных начальных условиях:

1) $\frac{d^2x}{dt^2} + 4x = 0, x(0) = 0, \frac{dx}{dt}(0) = 2.$

2) $\frac{d^2x}{dt^2} + 3\frac{dx}{dt} + 2x = 0,\ x(0) = 1. \frac{dx}{dt}(0) = -1.$

3) $\frac{d^2x}{dt^2} + 2\frac{dx}{dt} = 0, x(0) = 1, \frac{dx}{dt}(0) = 0.$

4) $t\frac{d^2x}{dt^2} = \frac{dx}{dt}, x(0) = 0, \frac{dx}{dt} = 0.$

5) $(1 - t^2)\frac{d^2x}{dt^2} - 2t\frac{dx}{dt} = 0, x(0) = 0,\ \frac{dx}{dt}(0) = 3.$

6) $\frac{d^{2x}}{dt^2}(1 - 3lnt) + \frac{1dx}{tdt} = 3 + lnt,\ x(1) = 0.8,\ \frac{dx}{dt} = -1.$

7) $\frac{d^2x}{dt^2} - 6x = 0, x(1) = 0,\ \frac{dx}{dt}(0) = 1.$

8) $5t\frac{d^2x}{dt^2} = \frac{dx}{dt}, x(0) = 0,\ \frac{dx}{dt} = -1.$

9) $(1 - t^2)\frac{d^2x}{dt^2} - 6t\frac{dx}{dt} = 0, x(1) = 0, \frac{dx}{dt}(0) = -1.$

10) $\frac{d^{2x}}{dt^2}(1 - lnt) + \frac{1dx}{tdt} = 3 + lnt,\ x(0) = 0.5,\ \frac{dx}{dt} = -1.$

2. Решить системы дифференциальных уравнений при заданных начальных условиях и построить графики решения:

1) $\begin{cases} \frac{dx1}{dt} = 2 \cdot x1 - 1; \\ \frac{dx2}{dt} = 3 \cdot x1 + 2 \cdot x2 + 1; \end{cases} \quad x1(0) = 1; \quad x(2) = 1.$

2) $\begin{cases} \frac{dx1}{dt} = x1 + x2; \\ \frac{dx2}{dt} = 2 \cdot x1 + 2; \end{cases} \quad x1(1) = 0; \quad x2(1) = 1.$

6 Лабораторная работа №6. Решение ОДУ с помощью m-file

6.1 Основные теоретические сведения. Методы решения ОДУ

Для решения систем ОДУ в MatLAB реализованы различные методы. Их реализации названы решателями ОДУ. Решатели реализуют следующие методы решения систем дифференциальных уравнений:

- ode45 – одношаговые явные методы Рунге-Кутта 4-го и 5-го порядка. Это классический метод, рекомендуемый для начальной пробы решения. Во многих случаях он дает хорошие результаты;
- ode23 – одношаговые явные методы Рунге-Кутта 2-го и 4-го порядка. При умеренной жесткости системы ОДУ и низких требованиях к точности этот метод может дать выигрыш в скорости решения;
- ode133 – многошаговый метод Адамса-Башворта-Мултона переменного порядка. Это адаптивный метод, который может обеспечить высокую точность решения;
- ode15s – многошаговый метод переменного порядка (от 1-го до 5-го, по умолчанию 5), использующий формулы численного дифференцирования. Это адаптивный метод, его стоит применять, если решатель ode45 не обеспечивает решения;
- ode23s – одношаговый метод, использующий модифицированную формулу Розенброка 2-го порядка. Может обеспечить высокую скорость вычислений при низкой точности;

- ode23t – метод трапеций с интерполяцией. Этот метод дает хорошие результаты при решении задач, описывающих осцилляторы с почти гармоническим выходным сигналом;

- ode23tb – неявный метод Рунге-Кутта в начале решения и метод, использующий формулы обратного дифференцирования 2-го порядка в последующем. При низкой точности этот метод может оказаться более эффективным, чем ode15s.

Все решатели могут решать системы уравнений явного вида: $y' = F(t, y)$. Решатели ode15s, ode23s, ode23t, ode23tb могут решать уравнения неявного вида $My' = F(t, y)$.

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

- F – название ODE-файла, то есть функции от *t* и *y*, которая возвращает вектор-столбец;
- tspan – вектор, определяющий интервал интегрирования [*to tfinal*]. Для получения решений в конкретные моменты *времени to, t1, …, tfinal* (расположенные в порядке уменьшения или увеличения), нужно использовать *tspan* = [*t0 t1 … tfinal*];
- y0 –вектор начальных условий;
- options – аргумент, создаваемый функцией *odeset*;
- p1, p2, … - произвольные параметры, передаваемые в функцию F;
- T, Y – матрица решений Y, где каждая строка соответствует времени, возвращенном в векторе-столбце T.

Перейдем к описанию функций для решения систем дифференциальных уравнений (будем обозначать понятием solver один из возможных численных методов решения ОДУ: ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb):

- [T, Y] = solver(‘F’, tspan, y0) интегрирует систему дифференциальных уравнений вида $y' = F(t, y)$ на интервале *tspan* с начальными условиями *y0*. ‘F’ – строка, содержащая имя ODE-файла. Функция $F(t, y)$ должна возвращать вектор-столбец. Каждая строка в массиве решений Y соответствует времени, возвращаемом 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. Обычно используемые параметры включают допустимое значение относительной погрешности RelTol (по умолчанию 1e-3) и вектор допустимых значений абсолютной погрешности AbsTol (все компоненты по умолчанию равны 1e-6);
- [T, Y] = solver(‘F’, tspan, y0, options, p1, p2, …) дает решение, подобное описанному выше, передавая дополнительные параметры p1, p2, … в m-файл F всякий раз, когда он вызывается. Используйте options = [], если никакие опции не задаются;
- [T, Y, TE, YE, IE] = solver(‘F’, tspan, y0, options) в дополнение к описанному решению содержит свойства Events, установленные в структуре options в положение on. ODE-файл должен быть создан так, чтобы вызов $F(t,$

y, 'events') возвращал соответствующую информацию. Выходной аргумент TE – вектор-столбец времен, в которые происходят события, строки YE являются соответствующими решениями, а индексы в векторе IE определяют, какое событие произошло;

- [T, X, Y] = solver('model', tspan, y0, options, ut, p1, p2, …) использует модель Simulink, вызывая соответствующий решатель из нее: [T, X, Y] = sim(solver, 'model', …).

Решатель систем ОДУ дает возможность получать решения систем из n уравнений. Система ОДУ может быть как однородной, так и неоднородной.

Решение сводится к следующему:

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

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

2) Получение решения и сопровождающий его график:

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

6.2 Линейные однородные уравнения с постоянными коэффициентами

Дифференциальное уравнение

$$a_n y^{(n)} + a_{n-1} y^{(n-1)} + \ldots + a_2 y'' + a_1 y' + a_0 y = 0\ , \qquad (1)$$

где $a_n, a_{n-1}, \ldots, a_2, a_1, a_0$ - постоянные величины, называется линейным однородным уравнением n - го порядка с постоянными коэффициентами.

$$y = C_1 y_2 + C_2 y_2 + C_3 y_3 + \ldots + C_n y_n, \qquad (2)$$

где $y_1, \ldots, y_n$ - его линейно независимые частные решения. Последние находятся с помощью характеристического уравнения

$$a_n r^n + a_{n-1} r^{n-1} + \ldots + a_2 r^2 + a_1 r + a_0 = 0. \qquad (3)$$

Если характеристическое уравнение имеет *n* действительных различных корней $r_m (m = 1,2, \ldots, n)$, то каждому из них соответствует частное решение

$$y_m = e^{r_n x} \qquad (4)$$

и общее решение уравнения (3) принимает вид:

$$y = C_1 e^{r_1 x} + C_2 e^{r_1 x} + \ldots + C_n e^{r_k x}\ . \qquad (5)$$

Пример. Решить уравнение

$$y''' - 2y'' - y' + 2y = 0.$$

Решение (аналитическое).

Для данного уравнения с постоянными коэффициентами составим характеристическое уравнение (при этом нужно сохранить коэффициенты, вместо y поставить 1, вместо ее производной k -того порядка поставить r^k):

$$r^3 - 2r^2 - r + 2 = 0.$$

Преобразуя правую часть уравнения, получим

$$r^2(r-2) - (r-2) = 0, (r-2)(r^2-1) = 0,$$

откуда

$$r_1 = -1 \quad r_2 = 1 \quad r_3 = 2.$$

В соответствии с формулой (5) получаем общее решение

$$y = C_1 e^{-x} + C_2 e^{x} + C_3 e^{2x}.$$

Решение в *MATLAB*

Аналитическое:

```
>> dsolve('D3y-2D2y-Dy+2y=0');
ans =C1exp(t)+C2exp(2t)+C3exp(-t).
```

Численное:

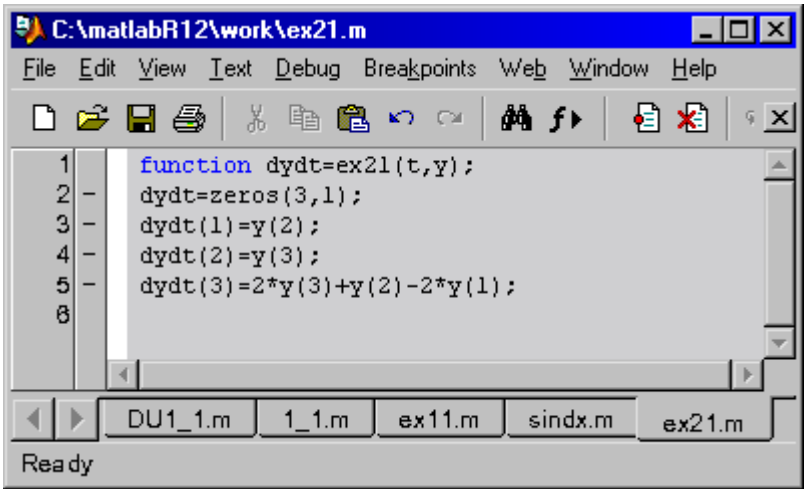

Рисунок 15

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

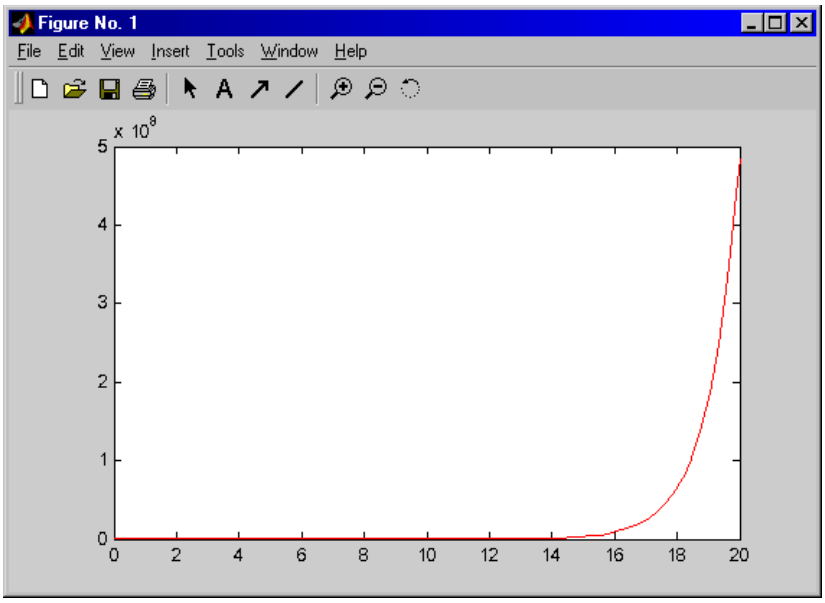

Рисунок 16

Пример. Решить уравнение

$$y^{IV} - 4y''' - 8y'' - 8y' + 4y = 0.$$

Составим характеристическое уравнение

$$r^4 - 4r^3 + 8r^2 - 8r + 4y = 0.$$

Преобразуя левую часть этого уравнения, получим

$$r^4 - 4r^3 + 8r^2 - 8r + 4 = r^4 - 4r^3 - 4r^2 - 8r + 4 = (r^2 - 2r + 2)^2 = (r^2 - 2r + 2)(r^2 - 2r + 2) = 0.$$

откуда

$$r^2 - 2r + 2 = 0, r^2 - 2r + 2 = 0.$$

Следовательно, комплексно сопряженные корни характеристического уравнения $r_{1,2} = 1 \pm i$ имеют кратность $k = 2$. Так как в данном случае $\alpha = 1, \beta = 1$, то в соответствии с формулами (1) и (2) общее решение примет вид:

$$y = C_1 e^2 \cos x + C_2 e^x \sin x + C_3 x e^x \cos x + C_4 \sin x$$

или

$$y = e^x[(C_1 + C_3x)\cos x + (C_2 + C_4x)\sin x].$$

Решение в *MATLAB*

```
>> dsolve('D4y-4·D3y+8·D2y-8·Dy+4·y=0');
ans =
C1·exp(t) ·sin(t)+C2·exp(t) ·cos(t)+C3·exp(t) ·cos(t) ·t+C4·exp(t) ·sin(t) ·t.
```

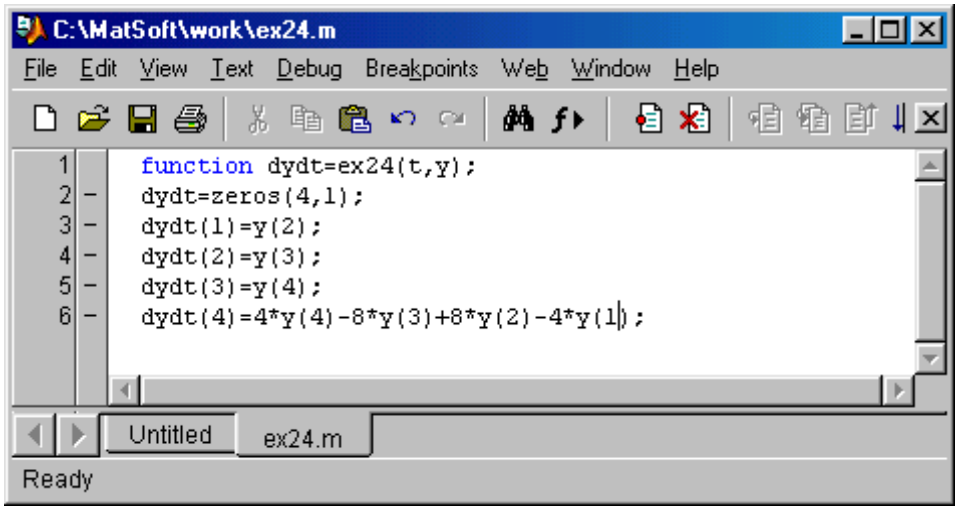

Рисунок 17

```
>> tspan=[0 10];
>> y0=[-1 1 0 -5];
>> [t,y]=ode45('ex24',tspan,y0);
>> plot(t,y(:,1)).
```

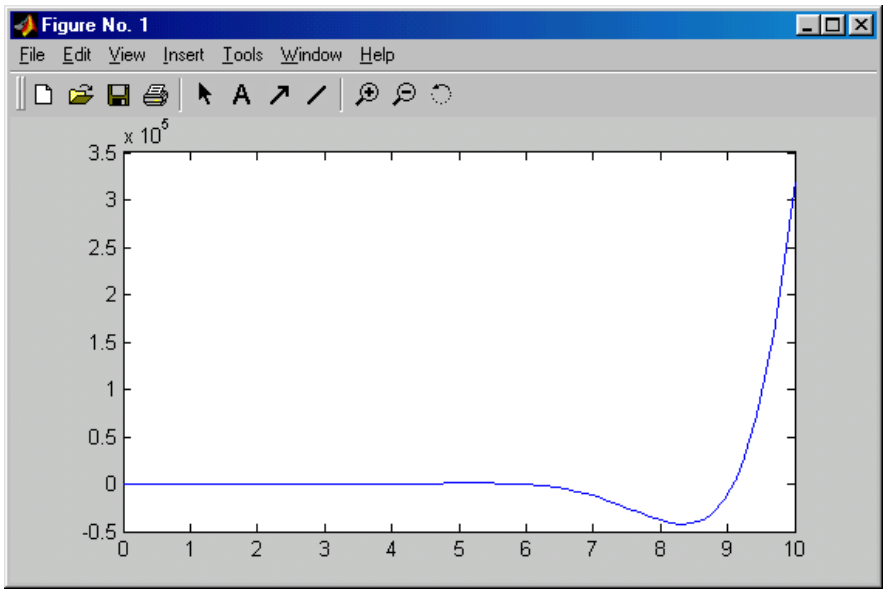

Рисунок 18

Пример. Исследуем задачу о движении планеты вокруг Солнца под действием тяготения. Она записывается в виде системы ОДУ 2-го порядка

$$\ddot{x} = -\frac{kx}{(x^2+y^2)^{3/2}}, \qquad \ddot{y} = -\frac{ky}{(x^2+y^2)^{3/2}},$$

где x$(t), y(t)$ координаты движущейся планеты. Преобразуем исходную систему к системе ОДУ 1-го порядка. Обозначим

$$z_1 = x, z_2 = \dot{x}, \qquad z_3 = \dot{y}, \qquad z_4 = \dot{y}.$$

Тогда

$$\begin{cases} z_1' = z_2; \\ z_2' = -\dfrac{kz_1}{(z_1^2+z_3^2)^{3/2}}; \\ z_3' = z_4; \\ z_4' = -\dfrac{kz_3}{(z_1^2+z_3^2)^{3/2}}. \end{cases}$$

Для модельной задачи выберем *k=1* и зададим начальные условия

$$z_1(0) = 1,\ z_2(0) = 0,\ z_3(0) = 0,\ z_4(0) = 1.$$

Решение удобно оформить в виде одной функции с подфункцией для вычисления правой части системы

```
function planet
% решение системы
Y0=[1; 0; 0; 0.4]; % вектор начальных условий
[T,Y]=ode45(@fun,[0 20],Y0); % решаем систему
plot(Y(:,1),Y(:,3)); % график траектории движения
grid on;
axis equal;
pause;
comet(Y(:,1),Y(:,3)); % анимация движения точки
function F=fun(x,y) % подфункция правой части системы
F=[y(2);-y(1)/(y(1).^2+y(3).^2).^(3/2); y(4);-y(3)/(y(1).^2+y(3).^2).^(3/2)];
```

На следующем рисунке показана получаемая траектория движения планеты (кривая с параметрическим уравнением $x(t), y(t)$).

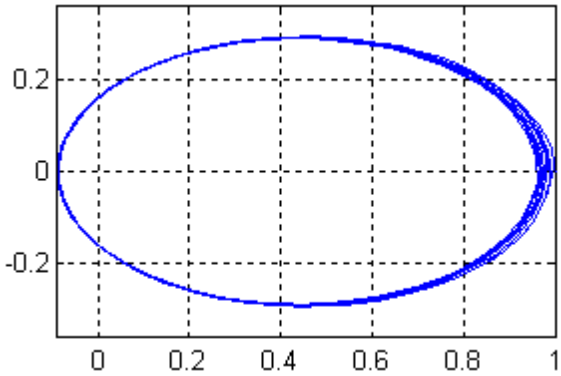

Рисунок 19 – Траектория движения планеты

Для того, чтобы следить за движением точки по траектории, мы использовали функцию comet. Она позволяет получить анимированный график, на котором кружок, обозначающий точку, перемещается на плоскости, оставляя за собой след в виде линии – траектории движения. Следите за тем, чтобы окно с графиком было поверх остальных окон.

6.3 Содержание отчета

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

6.4 Контрольные вопросы

1. Для чего служит команда dsolve?
2. В чем отличие command window от m-file?
3. Какие существуют методы решения систем дифференциальных уравнений?

6.5 Вариант задании

1. Решить уравнение $y^{IV} - 4y''' - 8y'' - 8y' + 4y = 0.$
2. Решить уравнение $y''' - 2y'' - y' + 2y = 0.$
3. Решить уравнение $y''' - 7y'' + 15y' - 9y = 0.$
4. Решить уравнение $y^{(IV)} - 16y = 0.$
5. Решить уравнение $y^{IV} - 4y''' - 8y'' - 8y' + y = 0.$
6. Решить уравнение $y^{VI} + 2y^{IV} + y'' = 0.$
7. Решить уравнение $y^{VIII} + 2y^{VI} - 2y' - y = 0.$
8. Решить уравнение $y''' - 8y'' + 6y' - 2y = 0.$
9. Решить уравнение $y^{(IV)} - 16y = 0.$
10. Решить уравнение $y^{V} - 4y'' - 8y' - 8y = 0.$

7 Лабораторная работа №7. Моделирование в SIMULINK

7.1 Основные теоретические сведения

Наиболее простым и удобным средством для моделирования динамических систем в пакете MATLAB является SIMULINK. Он поставляется в виде отдельного приложения и содержит библиотеки различных блоков, из которых в рабочем окне строится структурная схема модели. Процесс программирования в явном виде отсутствует, поэтому иногда говорят о визуальном моделировании. Особенно удобен такой подход для моделирования систем автоматического управления, заданных структурными схемами, дифференциальных уравнений (схемы для них строятся по методу понижения производных), механических систем и электрических схем.

SIMULINK можно запустить, набрав команду simulink или нажимая на кнопку в линейке меню. При этом открывается окно браузера библиотек. Для создания новой модели выбираем пункт меню File/New/Model. То же самое можно сделать при помощи кнопки на панели инструментов. После выделения одной из библиотек (например, непрерывные блоки – Continuous) на правой панели появится список доступных блоков. Выбранный блок следует перетащить мышью в окно модели. Если щелкнуть мышью по блоку в модели, то откроется окно параметров блока.

Использование первого и второго способов приводит к открытию окна обозревателя разделов библиотеки Simulink.

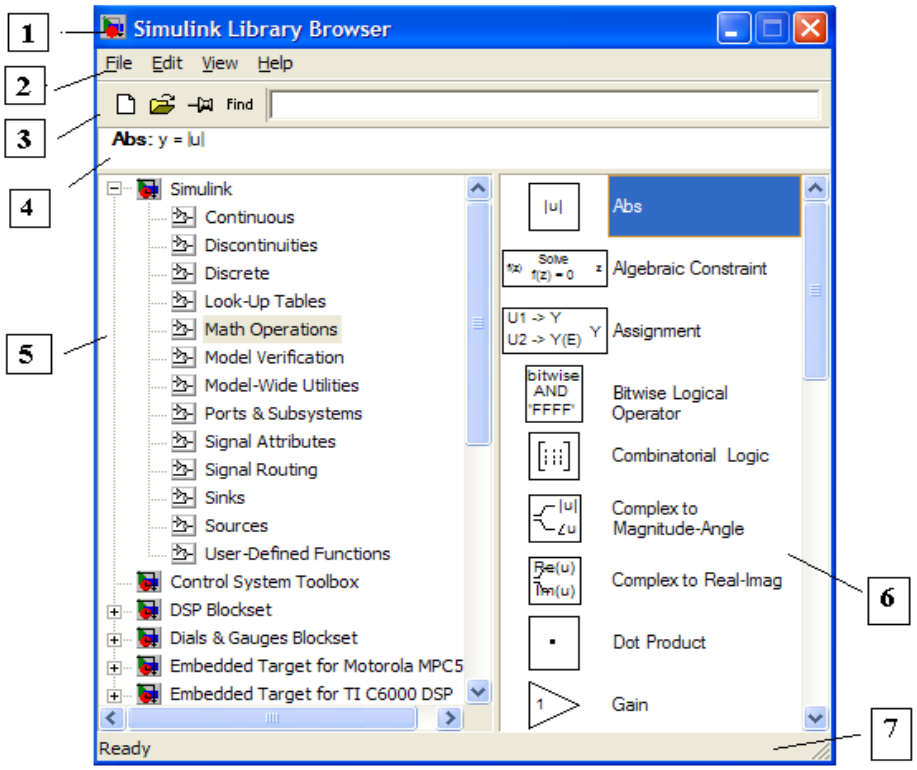

Рисунок 20 – Окно обозревателя разделов библиотеки Simulink

Окно обозревателя библиотеки блоков содержит следующие элементы:

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

На рисунке 20 выделена основная библиотека Simulink (в левой части окна) и показаны ее разделы (в правой части окна).

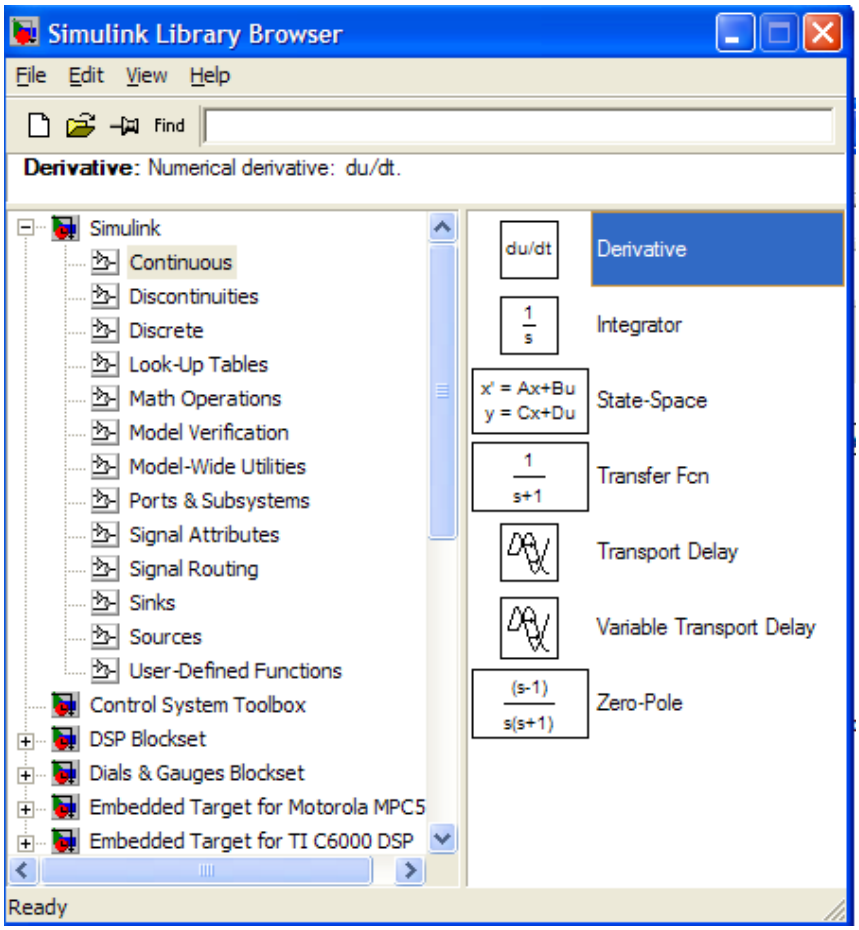

Рисунок 21 – Содержимое раздела Continuous

Библиотека Simulink содержит следующие основные разделы:

1) Continuous – линейные блоки.
2) Discrete – дискретные блоки.
3) User–Defined Functions – функции и таблицы.

4) Math Operations – блоки математических операций.
5) Discontinuities – нелинейные блоки.
6) Signals Attribute, Signals Routing – сигналы и системы.
7) Sinks - регистрирующие устройства.
8) Sources — источники сигналов и воздействий.
9) Ports & Subsystems – порты и блоки подсистем.

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

– пиктограмма свернутого узла дерева содержит символ "+", а пиктограмма развернутого содержит символ "-";

– для того чтобы развернуть или свернуть узел дерева, достаточно щелкнуть на его пиктограмме левой клавишей мыши.

При выборе соответствующего раздела библиотеки в правой части окна отображается его содержимое.

Пример моделирования в SIMULINK.

Основной способ исследования систем в SIMULINK – составление структурной схемы на усилителях, сумматорах, интеграторах и других блоках, последующее моделирование и наблюдение результатов на имитаторах осциллографов и других регистрирующих приборов.

Рассмотрим процедуру моделирования на примере дифференциального уравнения:

$$\ddot{y} + \sin y = 0;\ \ \dot{y}(0) = 0;\ \ y(0) = 1/2.$$

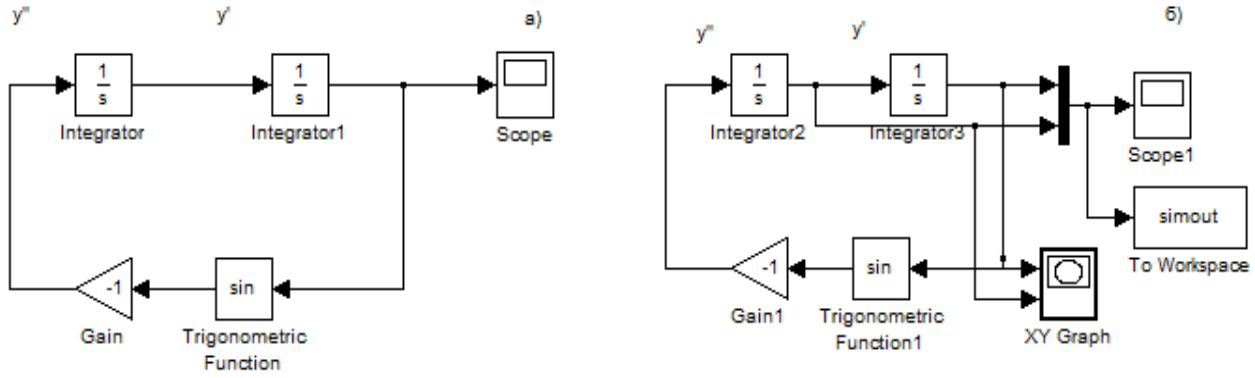

Рисунок 22

Пример. Моделирование двойного маятника. Консервативная колебательная система из двух взаимосвязанных маятников описывается уравнениями

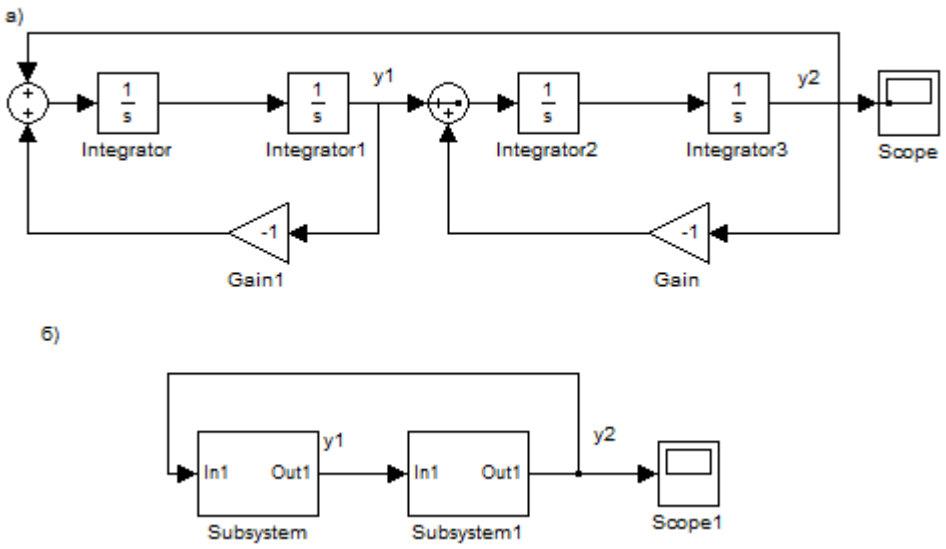

Рисунок 23 – Результат моделирования для начальных условий y1(0)=0, y2(0)=1 и значения k=1,2

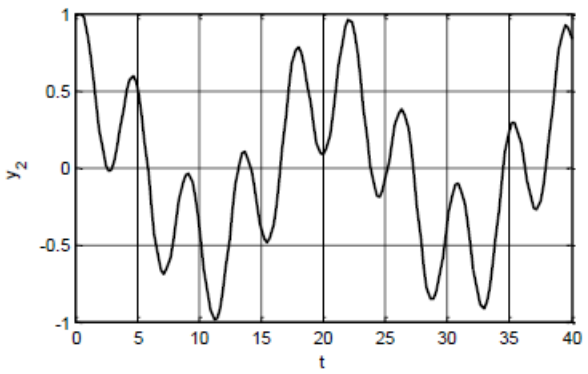

Рисунок 24

7.2 Содержание отчета

1. Схема модели.
2. Результаты моделирования.

7.3 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?

7.4 Варианты задании

1. Составить схему и выполнить моделирование в SIMULINK следующих линейных дифференциальных уравнений с переменными коэффициентами:

1) Уравнения Матье $\ddot{y} + (1 - 0{,}5\cos t)\dot{y} + 4y = 0.$

2) Уравнения Бесселя $t^2\ddot{y} + t\dot{y} + (t^2 - n^2)y = 0.$

3) Уравнения Лежандра $(1 - t^2)\ddot{y} - 2t\dot{y} + n(n+1)y = 0.$

4) Уравнения Чебышева $(1 - t^2)\ddot{y} - ty\dot{} + n^2y = 0.$

5) Уравнения Лагерра $t\ddot{y} + (1 - t)y + n\dot{y} = 0.$

6) Свободные колебания маятника описываются дифференциальным уравнением

$$\ddot{y} + \frac{g}{l}\sin y = 0,$$

где y – угол отклонения;
g – ускорение свободного падения;
l – длина маятника.

Список литературы

1 Горбаченко В.И. Вычислительная линейная алгебра с примерами на MATLAB. – СПб.: БХВ – Петербург, 2011. – 320 с. (http://www.bhv.ru/books/full_contents.php?id=189162)

2 Курбатова Е.А. MATLAB 7. Самоучитель. – М.: Вильямс, 2006 – 256 с. ISBN: 5–8459–0904–X.

3 Иглин С.П. Математические расчеты на базе Matlab. – М.: BHV–Санкт-Петербург, 2005. – 649 с. ISBN: 5–94157–290–5.

4 Ануфриев И. Е. Самоучитель MatLab 5.3/6.x. -СПб.: БХВ – Петербург, 2003.

5 Дьяконов В.П. MATLAB 6/6.1/6.5 + SIMULINK 4/5 в математике и моделировании. – М.: Солон – Пресс, 2003.

6 Мироновский Л.А., Петрова К. Ю. Введение в MATLAB: Учебное пособие. – СПб: ГУАП – СПб., 2005.

Сводный план 2016г., поз. 147

Марат Катубаевич Шимырбаев
Дана Толеубековна Тулекенова

МАТЕМАТИЧЕСКАЯ ОБРАБОТКА ИНФОРМАЦИИ

Методические указания к выполнению лабораторных работ
для студентов специальности
5В074600 – Космическая техника и технологии

Редактор Н.М. Голева
Специалист по стандартизации Н.К. Молдабекова

Подписано в печать__. __. __. Формат 60х84 1/ 16
Тираж 50 экз. Бумага типографская №
Объем 2,5 уч.-изд.л. Заказ ___ Цена 1250 тенге

Копировально-множительное бюро
некоммерческого акционерного общества
«Алматинский университет энергетики и связи»
050013, Алматы, ул. Байтұрсынұлы, 126