Тема 3.
Возможности и использование системы компьютерной математики Scilab

УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

1. **Алексеев, Е.Р. Scilab:** Решение инженерных и математических задач / Е.Р.Алексеев, О.В.Чеснокова, Е.А.Рудченко. — М. : ALT Linux ; БИНОМ. Лаборатория знаний, 2008. — 269с.

2. **Введение в Scilab :** Практикум по курсу «Информатика» для студентов технических специальностей дневной и заочной форм обучения / Т.А.Трохова, Т.Л.Романькова. – Гомель : ГГТУ им. П.О. Сухого, 2016. – 56 с. , **№4257**

1. Основные функции системы

Наиболее известными системами компьютерной математики (**СКМ**) являются:

- **Mathematica** численные и
- **Maple** аналитические вычисления

- **Matlab** в основном численные
- **MathCAD** вычисления

Scilab – это свободно распространяемая система компьютерной математики (аналог **Matlab)**, которая предназначена для выполнения научно-технических расчетов, графической интерпретации полученных результатов и визуального моделирования. Эта система имеет удобный пользовательский интерфейс и развитый язык программирования.

Все функции системы можно классифицировать следующим образом:

- математические
- использования численных методов
- программирование
- графические
- имитационного моделирования
- сервисные

К основным вычислительным функциям относятся:

- вычисление арифметических и логических выражений
- вычисление стандартных математических функций
- операции с векторами и матрицами
- матричные операции линейной алгебры

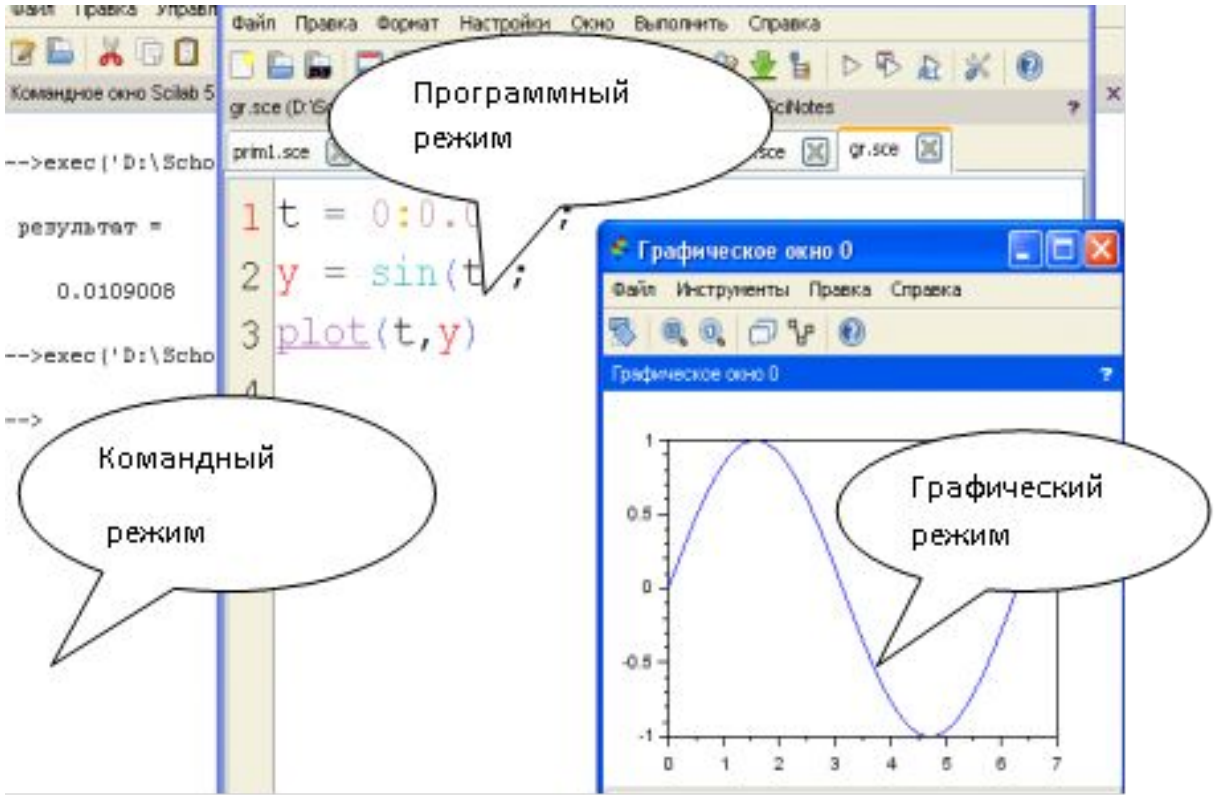
К численным методам относятся:

- численные методы решения алгебраических уравнений и систем
- методы работы с полиномами
- методы решения обыкновенных дифференциальных уравнений и систем
- методы аппроксимации и интерполяции

2. Интерфейс системы

Система имеет несколько режимов работы, каждый из которых поддерживается собственным диалоговым окном:

- командный режим – командное окно
- программный режим – окно создания и редактирования программных файлов (SCE-файлов)
- графический режим – окно редактирования графиков
- режим помощи – окно помощи
- режим демонстрации – окна демонстрационных примеров


Программный режим
Командный режим
Графический режим
Графическое окно 0
Файл Правка Формат Настройки Окно Выполнить Справка
Файл Инструменты Правка Справка
Командное окно Scilab 5
-->exec('D:\Scho
результат =
0.0109008
-->exec('D:\Scho
-->
prim1.sce
gr.sce
1 t = 0:0.
2 y = sin(t);
3 plot(t,y)

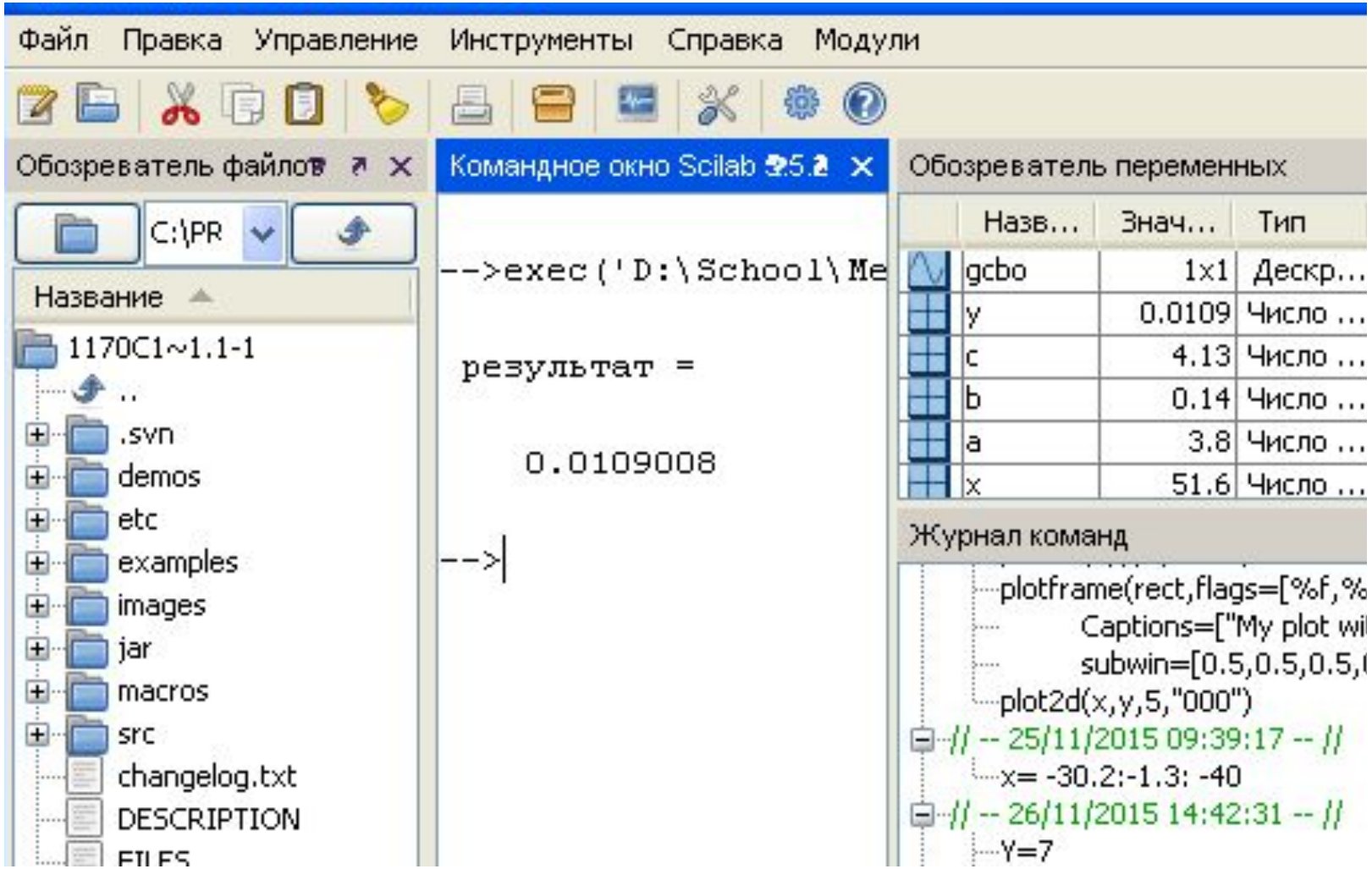
При работе в любом из перечисленных режимов могут быть использованы дополнительные информационные окна.

Окно рабочей области (**обозреватель переменных**) – предназначено для просмотра содержимого рабочей области памяти, в нем указывается имя переменной (массива или структуры), ее размерность и тип.

Окно журнала команд содержит перечень команд, введенных пользователем в командном режиме за текущий и предыдущий сеансы работы с системой.

Окно управления файлами (**обозреватель файлов**) служит для доступа к файлам при работе с системой.

Управлять информационными окнами можно с помощью меню ***Инструменты***. Очистить журнал команд и командное окно можно с помощью меню ***Правка***.


Файл Правка Управление Инструменты Справка Модули
Обозреватель файлов
C:\PR
Название
1170C1~1.1-1
..
.svn
demos
etc
examples
images
jar
macros
src
changelog.txt
DESCRIPTION
Командное окно Scilab 5.2
-->exec('D:\School\Me
результат =
0.0109008
-->
Обозреватель переменных
Назв... Знач... Тип
gcbo 1x1 Дескр...
y 0.0109 Число ...
c 4.13 Число ...
b 0.14 Число ...
a 3.8 Число ...
x 51.6 Число ...
Журнал команд
plotframe(rect,flags=[%f,%
Captions=["My plot wi
subwin=[0.5,0.5,0.5,
plot2d(x,y,5,"000")
// -- 25/11/2015 09:39:17 -- //
x= -30.2:-1.3: -40
// -- 26/11/2015 14:42:31 -- //
Y=7

3. Работа с документом

Создание документа в системе Scilab может выполняться в командном и программном режимах.

Документ при работе **в командном режиме** представляет собой последовательность команд пользователя и ответов системы, расположенных в командном окне.

Символ **-->** в окне команд показывает, что система готова к диалогу с пользователем.

Командная строка может содержать одну или несколько команд и завершается нажатием клавиши **Enter**.

Строка реакции системы называется строкой вывода, она показывает результаты выполнения команды либо в стандартной переменной ответа ***ans***, либо в переменной, заданной пользователем. Переменная ***ans*** хранится в оперативной памяти и может использоваться в дальнейших вычислениях.

--> **4+3**

ans=

7

--> **b=4+3**

b=

7

Выражения или команды разделяются символами «,» или «;». Результат вычисления выражения или команды, за которой следует **запятая или пробел**, выводится на экран в строку вывода.

Результат вычисления выражения или команды, за которой следует **точка с запятой**, на экран не выводится, но сохраняется в памяти

```
--> x=5, y=x+7;
x=
5
```

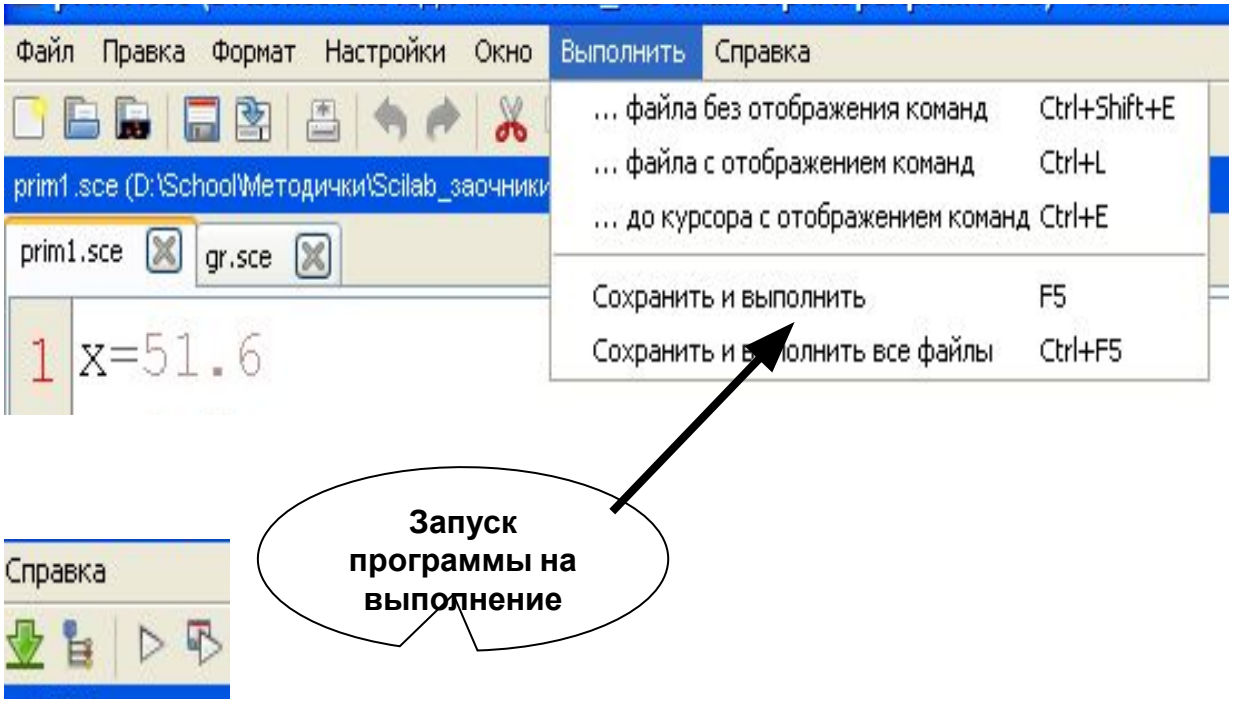
Отработанная командная строка не может быть выполнена повторно путем возвращения в нее курсора мыши в командном окне. Для нового выполнения команды она должна быть вызвана из стека команд нажатием клавиш ↑ или ↓.

Если выражение не помещается в одной командной строке, то его можно перенести на следующую строку, а предыдущую закончить тремя точками.

В программном режиме пользователь создает программу, которая состоит из команд системы Scilab и хранится на диске в виде файла с типом ***sce***.

Последовательность создания и выполнения программы:

1. открыть окно редактора программ **SciNotes** командой ***Инструменты – Текстовый редактор SciNotes*** или кнопкой **Открыть SciNotes** на панели инструментов командного окна
2. набрать текст программы
3. сохранить программу в файле с типом **sce** с помощью команды ***Файл – Сохранить как*** программного окна
4. запустить программу на выполнение с помощью команды ***Сохранить и выполнить*** меню ***Выполнить*** программного окна, кнопки **F5** или кнопки ***Сохранить и выполнить*** на панели инструментов программного окна



Если компиляция и выполнение программы прошли успешно, то результаты выполнения будут отражены в командном окне.

Если в результате компиляции или выполнения найдены ошибки, то в командном окне выводятся сообщения об ошибках.

В этом случае необходимо открыть программу в окне редактора **SciNotes**, исправить ошибки и повторить выполнение программы.

4. Базовые вычисления в Scilab

Входной язык системы представлен данными, выражениями, операторами.

Данные – константы и переменные делятся на пользовательские и системные.

Основные системные переменные и константы, применяемые в Scilab:

%i или **%j** – мнимая единица (корень квадратный из –1);
%pi – число π=3,1415926…;
%e – число e=2.7182818;
ans – результат выполнения последней операции.

Действительные константы могут быть целыми, вещественными, с фиксированной и плавающей точкой.

Возможно представление чисел в научном формате с указанием мантиссы и порядка числа. **Дробная часть отделяется от целой точкой.**

10 -23 – целые константы.

5.3 -57.88 – вещественные константы с фиксир. точкой

34.35e-5 4.3e+4 4.3e4 – вещественные константы с указанием мантиссы и порядка числа (с плав. точкой).

Символьная константа представляет собой набор символов, заключенных в двойные или одинарные кавычки (апострофы).

"Заочный факультет" 'ГГТУ им. П.О. Сухого'

Символы // используются для ввода комментариев к вычислениям.

Правила образования имен данных (идентификаторов):

- имя может содержать буквы латинского алфавита, цифры и символ подчеркивания
- имя должно начинаться с буквы
- заглавные и строчные буквы различаются

Примеры имен: **SUM, sr1, prim_1**

Выражения в Scilab делятся на **арифметические, логические и строковые.**

Арифметические выражения используются для вычисления целых или вещественных значений. Они состоят из констант, переменных, стандартных функций, знаков арифметических операций. Круглые скобки используются для указания порядка выполнения операций и аргументов функций.

Основные арифметические операции

Название	Знак операции	Пример
Сложение	+	x+y
Вычитание	-	x-y
Умножение	*	x*y
Возведение в степень	^	x^5
Деление	/	x/y

Основные математические функции

Описание	Имя	Описание	Имя
Абсолютная величина	**abs(x)**	Синус	**sin(x)**
Экспонента	**exp(x)**	Тангенс	**tan(x)**
Натуральный логарифм	**log(x)**	Котангенс	**cotg(x)**
Знак числа	**sign(x)**	Арккосинус	**acos(x)**
Десятичный логарифм	**log10(x)**	Арксинус	**asin(x)**
Корень квадратный	**sqrt(x)**	Арктангенс	**atan(x)**
Косинус	**cos(x)**	Арккотангенс	**acot(x)**

Примеры записи арифметических выражений в Scilab

$$\frac{\cos x^2}{x+\sin^3 x}+e^{-2.1}$$

cos(x^2) / (x+sin(x)^3)+exp (-2.1)

$$\sqrt{\frac{tg^2 x}{\ln x^5-\sqrt[3]{x}}}$$

sqrt (tan(x)^ 2 / (log (x^5)–x^(1 / 3)))

$$\frac{\cos^3 x+\sqrt[5]{|x+arctg\ 2x|}}{3e^{\sqrt{x}}}$$

(cos(x)^3+abs(x+atan(2*x))^(1/5)) / (3*exp(sqr(x)))

Оператор присваивания присваивает переменной, стоящей слева от знака присваивания (=) значение выражения, стоящего справа от знака присваивания.

имя_переменной = выражение

Выражение может быть арифметическим, логическим или символьным.

a = cos(x)+c-d^2*p^2+4.92

Все переменные, используемые в выражении, должны иметь значения.

Для ввода данных используется функция **input**, которая имеет вид:

имя = input (символьная константа)

Здесь **имя –** это имя простой переменной, значение которой нужно ввести.

Символьная константа – набор любых символов, заключенный в двойные или одинарные кавычки. Символьная константа, как правило, уточняет действия пользователя.

n=input ("Введите значение n")

В командном окне выводится текст – значение символьной константы, выполнение программы приостанавливается, пользователь вводит необходимое значение, нажимает клавишу **Enter**, и введенное значение присваивается переменной с именем **имя**, т.е. помещается в то место оперативной памяти, которое отведено для этой переменной.

Для вывода данных в командное окно используется функция (оператор) **disp**.

disp (выражение)

Выражение – это арифметическое, логическое или символьное выражение, частным случаем которого являются константы или переменные любого типа.

Каждый новый оператор **disp** выполняет вывод с новой строки командного окна.

Если переменным уже присвоены числовые значения **a=11, b=3, c=10** , то в результате выполнения программы

```
d = a-b+2*c;
disp ('d='), disp(d);
```

в командном окне будет выведен результат

```
d=
28
```

5. Обработка структурированных данных

В Scilab можно использовать различные типы структурированных данных, т.е. данных, содержащих несколько элементов. К основным структурированным данным относятся массивы чисел (вектора, матрицы, многомерные массивы).

Массив – это последовательность величин одного типа, имеющих одно имя, но различающихся индексами (порядковыми номерами).

Одномерный массив называют вектором, двумерный – матрицей.

Векторы в Scilab делятся на векторы-строки и векторы-столбцы.

Для создания векторов есть несколько способов:

1. *Непосредственный ввод*

Для того, чтобы создать вектор-строку, значения его элементов необходимо перечислить в квадратных скобках, разделяя их пробелами или запятыми.

V=[2 3 8 0]

Для того, чтобы создать вектор-столбец, значения его элементов необходимо перечислить в квадратных скобках, используя для разделения строк точку с запятой.

V=[2;3;8;0]

Обращение к элементу вектора выполняется указанием имени вектора и номера элемента в векторе в круглых скобках.

V(2) → 3 V(3) → 8

2) Ввод с использованием диапазона

x = xn : dx : xk , где

xn – начальное значение переменной x
dx – шаг изменения значений переменной x
xk – конечное значение переменной x

В результате будет сформирован вектор, первый элемент которого равен **xn**, второй – **xn+dx**, третий – **xn+2*dx** и т.д. Последний элемент будет не больше **xk** для положительного шага **dx** , и не меньше **xk** – для отрицательного.

Если величина шага отсутствует, то по умолчанию его значение равно **1** или **-1**:

x=xn : xk

Фрагмент программы создания векторов и их вывод в командном окне:

```
a=5:2:15
disp(a)          7    9    11    13    15
v=7:3
disp(v)          7    6    5     4     3
```

3. Для того, чтобы **создать матрицу**, значения ее элементов следует перечислить в квадратных скобках, разделяя элементы в строках пробелами или запятыми, а для разделения строк использовать точку с запятой.

```
A=[ 3  1   -3  ;  2   0    1.5]
disp(A)
3.    1.    -3.
2.    0     1.5
```

Для указания отдельного элемента матрицы после имени матрицы в круглых скобках указываются два индекса: номер строки и номер столбца.

A(2,3) – это элемент, расположенный во второй строке и третьем столбце матрицы **A**.

Система имеет обширный набор стандартных функций и операций по обработке матриц, который позволяет:

- формировать новые матрицы стандартного вида
- выполнять матричные арифметические операции
- вычислять матричные характеристики и математические функции

Для формирования новых матриц стандартного вида применяются следующие системные **функции**:

- **rand(m,n)** – формирует прямоугольную матрицу размерностью **m×n**, элементами которой являются случайные числа в интервале от **0** до **1**, функция rand без параметров формирует одно случайное число в том же интервале.
- **ones(m,n)** формирует матрицу размерностью **m×n**, состоящую из единиц.
- **zeros(m,n)** формирует матрицу размерностью **m×n**, состоящую из нулей.
- **diag(V)** создает диагональную матрицу, в которой элементы вектора V являются элементами главной диагонали.

Стандартные функции, позволяющие вычислять различные характеристики матриц:

- **det(A)** – вычисляет определитель матрицы
- **trace(A)** – вычисление следа матрицы
- **rank(A)** – вычисление ранга матрицы
- **inv(A)** – вычисление обратной матрицы

Матричные арифметические операции:

- **A+B A-B** – матричное сложение и вычитание. Оба операнда этой операции должны иметь одинаковую размерность, если они являются матрицами. Один из операндов может выть скалярной величиной.

- **A*B** – матричное умножение. Операция выполняется по правилам умножения матриц, число столбцов матрицы A должно быть равно числу строк матрицы B.

- **A^P** – возведение матрицы в степень. Эта операция при скалярном значении P возводит квадратную матрицу **A** в степень **P**. Если A – скалярная величина, а P – квадратная матрица, то **A^P** возводит **A** в матричную степень P. Эта операция является ошибочной, если оба операнда – матрицы.

1. Умножение матриц

```
x=[2 3 4; 5 6 7]
y=[9 8; 7 6; 5 4]
z=x*y
disp(z)
```

```
x=
   2   3   4
   5   6   7

y=
   9   8
  76
   5   4

z=
  59   50
 122  104
```

2. Умножение матрицы на вектор

```
x=[2 3 4; 5 6 7]
t=[3; 6; 9]
z=x*t
disp(z)
```

```
  60
 114
```

3. Умножение вектора-строки на вектор-столбец

```
a=[2 3 4]
b=[5;6;7]
z=a*b
disp(z)
```

```
2   3   4

5
6
7

  56
```

В Scilab существуют матричные операции, которые выполняются над каждым элементом матрицы:

.* – поэлементное матричное умножение

./ – поэлементное деление матриц

.^ – поэлементное возведение матрицы в степень

Оба операнда этих операций должны иметь одинаковую размерность, или один из них должен являться скалярной величиной.

Операция ′ (апостроф) выполняет транспонирование матрицы.

Поэлементное умножение матриц:

```
x=[2 3 4; 5 6 7]
y=[9 8 7 ; 6 5 4]
z=x .*y
disp(z)
```

```
 2   3   4          9   8   7
 5   6   7          6   5   4

 18  24  28
 30  30  28
```

Транспонирование матрицы:

```
y=[9 8 7 ; 6 5 4]
w=y'
disp(w)
```

```
 9   6
 8   5
 7   4
```

Вычисление определителя матрицы:

```
x=[2 3 4; 5 6 7; 1 2 2];
d=det(x)
disp (d)
```

```
 3
```

Над массивами можно выполнять различные операции, заданные системными функциями:

- **max(A)** – вычисление максимального элемента массива
- **min(A)** – вычисление минимального элемента массива
- **sum(A)** – вычисление суммы элементов массива
- **prod(A)** – вычисление произведения элементов массива
- **mean(A)** – вычисление среднего значения элементов массива
- **[Amax, Nmax] = max(A)** – вычисление максимального элемента в массиве и его номера (номера строки и столбца)
- **Amax = max(A)** – вычисление только максимального элемента массива.

6. Программирование в Scilab

Для программирования разветвляющихся алгоритмов в Scilab существуют условный оператор и оператор выбора.

Условный оператор представлен в нескольких формах и имеет следующий общий вид.

Полная форма 1

```
if   <условие>
     оператор1
   else
     оператор2
end
```

Краткая форма

```
if <условие>
     оператор1
 end
```

Полная форма 2

```
 if <условие1>
        оператор1
   elseif  <условие2>  оператор2
   elseif  < условие3> оператор3
   ……..
   else    оператор
end
```

В условиях используются логические выражения, операторы – любые операторы или группы операторов Scilab.

Пример:

```
if    b^2 - 4*a*c < 0
disp ("корни мнимые")
        else
                disp ("корни вещественные")
end
```

В простых логических выражениях используется операции отношения (<, <=, <, >=)

a > b c <= 5.8

Для образования сложных логических выражений используются логические операции ***& (И), | (ИЛИ)***, например:

(a>b) | (b<0) ***(a > 0) & (b>0)***

Операторы цикла предназначены для программирования циклических алгоритмов.

В Scilab два оператора цикла:

оператор цикла с предусловием **while**
оператор цикла с параметрами **for**

Оператор while предназначен для программирования любых циклических алгоритмов, где проверка условия повторения цикла выполняется перед выполнением операторов цикла.

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

Оператор цикла for

```
for   x= xn : dx : xk
          операторы
End
```

x – параметр цикла
xn – начальное значение параметра цикла
dx – шаг изменения параметра цикла
xk – конечное значение параметра цикла

Пример: Вывести числа от 1 до 4 с шагом 0.5

```
k=1
while k<=4
disp(k)
k=k+0.5
end

for k=1 : 0.5 : 4
    disp(k)
end
```

7. Построение графиков

Графические объекты в Scilab строятся в специальном графическом окне (**figure**). Одновременно может быть открыто несколько таких окон, каждому из которых присваивается порядковый номер, **(начиная с нуля)**. Для перехода к имеющемуся окну с номером **N** или открытия нового графического окна необходимо ввести команду **figure(N).**

Первое обращение к графической команде автоматически вызывает появление графического окна, которому присваивается первый номер.

Для построения графиков функций одной переменной в декартовой системе координат используются различные формы команды **plot**, которая рисует графики функций по ряду точек, соединяя их отрезками прямых.

Команда **plot(x,y)** – строит график функции, координаты точек которой берутся **из векторов одинаковой размерности x** и **y**.

Для построения графика функции **y=sin(t)** нужно задать следующий фрагмент программы.

```
x = 0:0.01:7
y = sin(x)
plot(x,y)
```

или

```
t = 0:0.01:7
plot(t,sin(t))
```

Если используется команда **plot(y),** то значениями первого вектора (аргумента) являются индексы (номера) элементов вектора **y**.

```
x = 0:0.01:7;
y = sin(x);
plot(y)
```

Команда **plot (x,y1,x,y2….)** – строит графики нескольких функций. Можно также использовать несколько команд **plot(x,y1), plot(x,y2),** ….

Для построения графиков двух функций — ***sin(x)*** **и** ***cos(x)*** нужно задать фрагмент программы:

1. x=0:0.01:6
y1=sin(x)
y2=cos(x)
plot(x,y1,x,y2)

2. x=0:0.01:6
plot(x,sin(x),x,cos(x))

3. x=0:0.01:6
plot(x,sin(x))
plot(x,cos(x))

Для форматирования линий графиков используется команда **plot(x,y,s),** где **s** - строковая константа, символы которой определяют формат графика.

Символы, которые могут использоваться в параметре **s**, приведены в таблице:

Тип линии		**Тип маркера точки линии**		**Цвет линии**	
-	Сплошная	**.**	Точка	**y**	Желтый
:	Двойной пунктир	**o**	Окружность	**m**	Фиолетовый
-.	Штрих-пунктир	**x**	Крест	**c**	Голубой
--	Штриховая	+	Плюс	**r**	Красный
		*	Звездочка	**g**	Зеленый
		s	Квадрат	**b**	Синий
		d	Ромб	**w**	Белый
		v	Треугольник	**k**	Черный

Для отображения графика функции $y=0.02x^3$ штриховой линией красного цвета с узловыми точками в виде ромбов:

```
x=-5:0.5:2;
plot(x,0.02*x.^3,'--dr')
```

Команда **plot(x1,y1,s1,x2,y2,s2,…)** – строит графики нескольких функций на одном поле графика.

Дополнительные параметры **s1, s2** и т.д. позволяют задать стиль линий графиков.

```
x=-6:0.1:6
y1=sin(x)
y2=cos(x)
plot(x,y1,'-xb',x,y2,'-+r')
```

Команда **xgrid ()** позволяет задавать построение сетки на поле графика.

Заголовок графика и надписи осей графика можно вывести с помощью команды

xtitle(title,xstr,ystr)

где **title** — символьная константа, содержащая название графика

xstr — символьная константа, содержащая название оси X

ystr — символьная константа, содержащая название оси Y

Создание легенды можно выполнить с помощью команды **legend**.

legend ("График функции y1(x)","График функции y2(x)")

Для очистки графического окна от старых результатов перед построением графика используется команда

newaxes

Пример. Построить графики двух функций, у каждой из которых задать тип линии, тип маркера точки графика и цвет линии. На график нанести координатную сетку, подписать оси, вывести заголовок графика и легенду.

```
figure(1)

x1=2:0.1:5
y1=atan(x1)./(1+sin(x1).^2)

x2=2:0.1:4
y2=(1+sqrt(0.5*x2))./(0.5+sin(x2).^2)

plot(x1,y1,'--rx',x2,y2,'-.go')

xgrid ()

xtitle("Графики двух функций",…
        "Значение аргумента","Значения функций")

legend("График функции y1(x1)","График функции y2(x2)")
```

8. Вычисление интегралов

В Scilab вычисление определенного интеграла методом трапеций реализовано функцией

inttrap(x,y),

где **x** – вектор значений аргумента подынтегральной функции на отрезке интегрирования, **y** – вектор значений, полученных при вычислении подынтегральной функции для элементов вектора **x.**

Пример. Вычислить интеграл $\int_{2}^{5.3} \frac{2xdx}{\sin x + 1.5}$

Фрагмент программы:

```
x =2:0.01:5.3
y =2*x./(sin(x)+1.5)
int = inttrap(x,y)
disp(int)
```

Можно вычислить интеграл с помощью функции

intg(a, b, f_name),

где **a, b –** пределы интегрирования, **f_name** – имя подынтегральной функции (должна быть задана с помощью внешней функции). Функция **intg** возвращает значение интеграла.

Фрагмент программы:

```
function y=f(x)
   y=2*x/(sin(x)+1.5)
endfunction
z=intg(2,5.3,f)
disp(z)
```

Внешнюю функцию можно задать командой

deff ('переменная=имя функции(параметр)', 'символьное представление функции')

deff('y=f(x)', 'y=2*x/(sin(x)+1.5)')

или

function переменная = имя функции (аргумент функции)

операторы, вычисляющие значение функции

Endfunction

```
function    y=f(t)
        y=2*t/(sin(t)+1.5)
endfunction
```

9. Решение нелинейных и полиномиальных уравнений

Для решения нелинейных уравнений в Scilab используется функция

fsolve(x0,f)

где **x0** – начальное приближение корня, **f** – функция, описывающая левую часть уравнения **f(x)=0**.

Пример. Решить уравнение $\sin(2x) = \cos(3x^2) + \sin(3x)$ с начальным приближением $x \approx 7$

```
deff('y=f(x)', 'y=sin(2*x)-cos(3*x.^2)-sin(3*x)')
root=fsolve(7,f)
disp(root)
```

Для выполнения графической интерпретации результатов решения уравнения **f(x)=0** нужно:

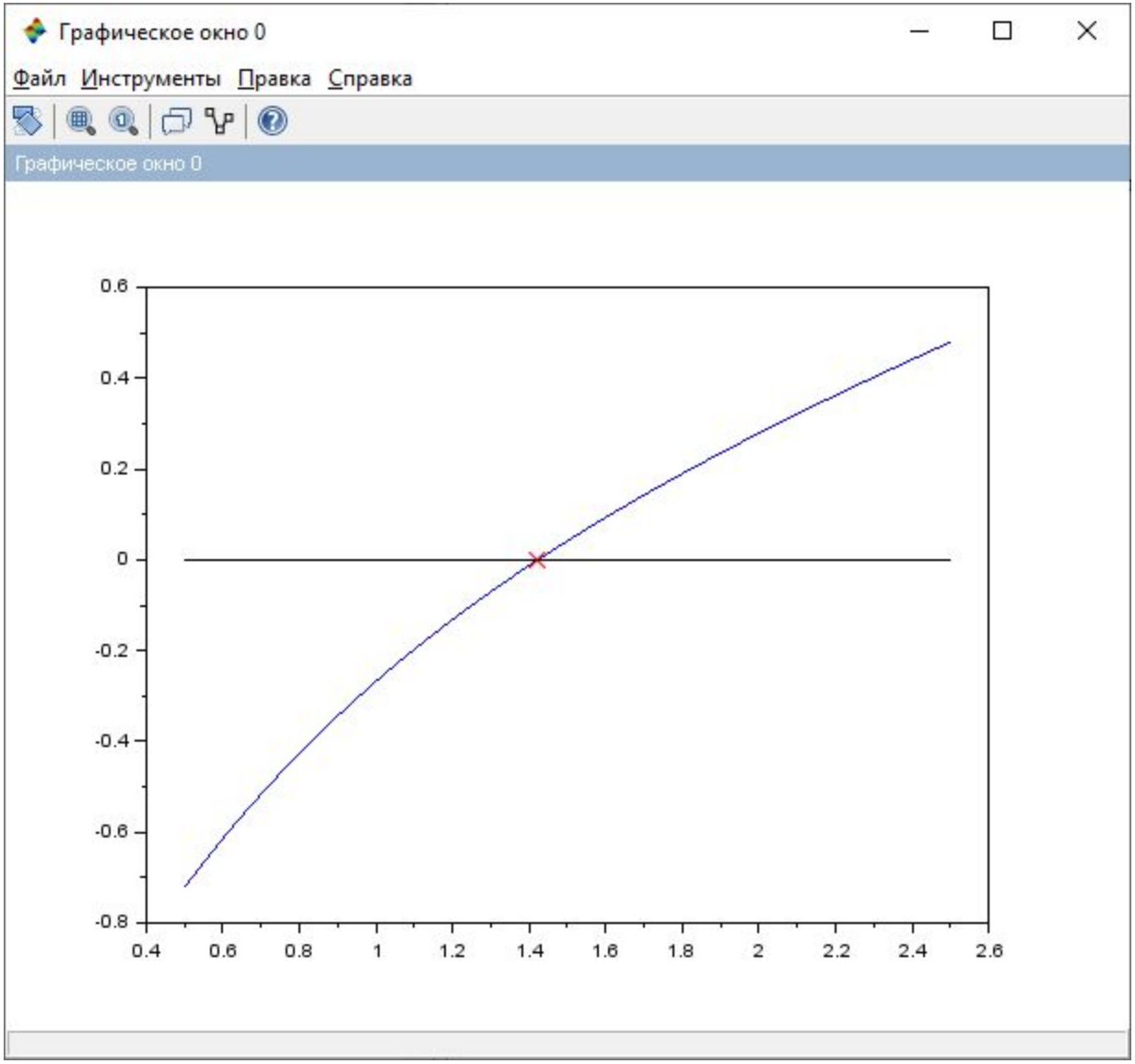
- задать аргумент функции **x** (вектор в виде **x = xn : dx : xk**) так, чтобы найденный корень попадал в диапазон между первым и последним элементом вектора
- построить график функции **f(x)**
- провести линию **y=0**
- отметить точку с абсциссой равной корню уравнения **(root)** и ординатой, равной значению функции для аргумента, равного корню уравнения **(f(root))**

Если корень найден правильно, отмеченная точка должна находиться на пересечении графика функции с линией **y=0**.

Решить уравнение и выполнить графическую интерпретацию решения:

```
deff('z=f
(x)','z=sqrt(3*x.^2)+sqrt(x.^2+3)-sqrt(6*x.^2+10)')

rk=fsolve(1,f )
disp(rk)                          Результат rk = 1.4229815

x=0.5:0.01:2.5          Функция
y=f(x)

x1=[0.5,2.5]          Прямая линия y=0
y1=[0,0]

plot ( x,y,       rk, f(rk), 'rx',      x1, y1, '-k')
    функция   корень ур-я       прямая y=0
```

Графическая интерпретация решения уравнения:



Для решения полиномиальных уравнений вида

$$v_n \cdot x^n + v_{n-1} \cdot x^{n-1} + \ldots + v_1 \cdot x + v_0 = 0$$

используется функция

roots(v),

где **v** – вектор коэффициентов перед неизвестными полинома размерности **n+1** (**n** – степень полинома).

Функция возвращает вектор корней полинома размерностью **n.**

Пример. Решить уравнение $3x^3 + x^2 - 10x - 8 = 0$

Фрагмент программы:

```
v = [3   1   -10   -8]
rt = roots(v)
disp(rt)
```

Результат:

```
2.
 - 1.3333333
 - 1.
```

10. Решение систем линейных и нелинейных уравнений

Для решения систем линейных уравнений в Scilab есть следующие способы:

- применение операции левого матричного деления
- использование обратной матрицы

Задана система линейных алгебраических уравнений вида

$$\mathbf{Ax=B}$$

где **A** – матрица коэффициентов перед неизвестными системы, **B** – вектор свободных членов.

Решение системы может быть найдено в виде:

```
x=A \ B
```

Или с помощью обратной матрицы, например:

```
x=inv(A)*B
```

Пример. Решить систему уравнений $\begin{cases} 3x_1 + x_2 = -4 \\ -3x_1 + 5x_2 = 36 \end{cases}$

Фрагмент программы:

```
A=[3 1;-3 5]
B=[-4 ;36]
x = inv(A)*B
disp (X)
```

или

```
x=A\B
disp (x)
```

Результат

```
- 3.1111111
  5.3333333
```

Для решения **систем нелинейных уравнений** можно использовать функцию

fsolve(x0,f)

где **x0** – вектор начальных приближений для неизвестных системы

f – функция, определяющая систему, в которой каждый элемент вектора результата определяется, как функция от вектора неизвестных системы.

Пример. Решить систему уравнений

$$\begin{cases} 2 \cdot x + y = 6 \\ x^2 + y^2 = 14 \end{cases}$$

Преобразуем систему уравнений:

$2*x + y - 6 = 0$ **Обозначим x как s(1)**

$x^2 + y^2 – 14 = 0$ **y как s(2)**

Фрагмент программы

```
function [z]=fun(s)
  z(1)=2*s(1)+s(2)-6
  z(2)=s(1)^2+s(2)^2-14
endfunction

x0=[1;1] – начальное приближение

xx=fsolve (x0, fun)

disp(xx)
```

Результат

```
1.2338096
3.5323808
```

10. Решение обыкновенных дифференциальных уравнений и систем уравнений

Дифференциальное уравнение — это уравнение, в которое входят производные функции, может входить сама функция, независимая переменная и параметры. Порядок входящих в уравнение производных может быть различен.

Порядок дифференциального уравнения – это наивысший порядок производных, входящих в него.

Производные, функции, независимые переменные и параметры могут входить в уравнение в различных комбинациях или могут отсутствовать, кроме хотя бы одной производной.

В отличие от алгебраических уравнений, результатом решения которых является одно или несколько чисел, **результатом решения дифференциального уравнения** является **функция** (семейство функций).

Дифференциальное уравнение порядка выше первого можно преобразовать в систему уравнений первого порядка, в которой число уравнений равно порядку исходного дифференциального уравнения.

Обыкновенные дифференциальные уравнения (**ОДУ**) — это уравнения, зависящие от одной независимой переменной, они имеют вид

$$F(x,y,y',y'',...,y^{(n)}) = 0$$

Дифференциальное уравнение первого порядка, разрешенное относительно первой производной имеет вид:

$$y' = f(x, y) \quad (1)$$

Система дифференциальных уравнений, разрешенных относительно первой производной имеет вид:

$$y1' = f1(x, y1, ..., ym),$$
$$...$$
$$ym' = fm(x, y1, ..., ym),$$

Решением уравнения **(1)** на интервале **[*a*, *b*]** называется **функция φ(x)**, обращающая уравнение в тождество на интервале **[*a*, *b*]**

$$\varphi'(x) \equiv f(x, \varphi(x))$$

График решения называется ***интегральной кривой Iφ*.**

Решение системы **ОДУ** — вектор функций $\mathbf{y_1(x)}$,… $\mathbf{y_m(x)}$, которые обращают уравнения системы в тождества.

Дифференциальные уравнения имеют множество решений, отличающихся константами. Для того, чтобы выделить одно из них, нужны дополнительные условия.

Количество таких условий должно совпадать с порядком дифференциального уравнения или системы

Такими условиями для уравнения (1) может быть начальное условие – требование, чтобы решение (функция) в заданной точке (обычно начале интервала **x0=a**) принимала заданное значение:

$$y(x0) = y0 \quad (2)$$

Задача о нахождении решения уравнения **(1)**, удовлетворяющего начальному условию **(2)**, называется **задачей Коши**.

Для решения обыкновенного дифференциального уравнение вида $\mathbf{dx/dt = f(t,x)}$, $\mathbf{x(t_0) = x_0}$ (задача Коши) используется функция

x=ode(x0,t0,t,f)

где

x0 – начальное значение $\mathbf{x_0}$

t0 – начальная точка интервала интегрирования $\mathbf{t_0}$

t – значения аргумента функции, в которых происходит поиск решения

f – внешняя функция, определяющая правую часть уравнения

x – решение (вектор значений функции **x(t)**)

Пример. Решить задачу Коши для уравнения

x' + x = sin(xt) на интервале [0, 35] при x(0) = 1.5

Преобразуем уравнение следующим образом:

$$x' = \sin(xt) - x, \quad x(0) = 1.5$$

Для решения используем функцию **x=ode (x0,t0,t,f)**
где

f – имя внешней функции **f(t,x)**

t – независимая переменная

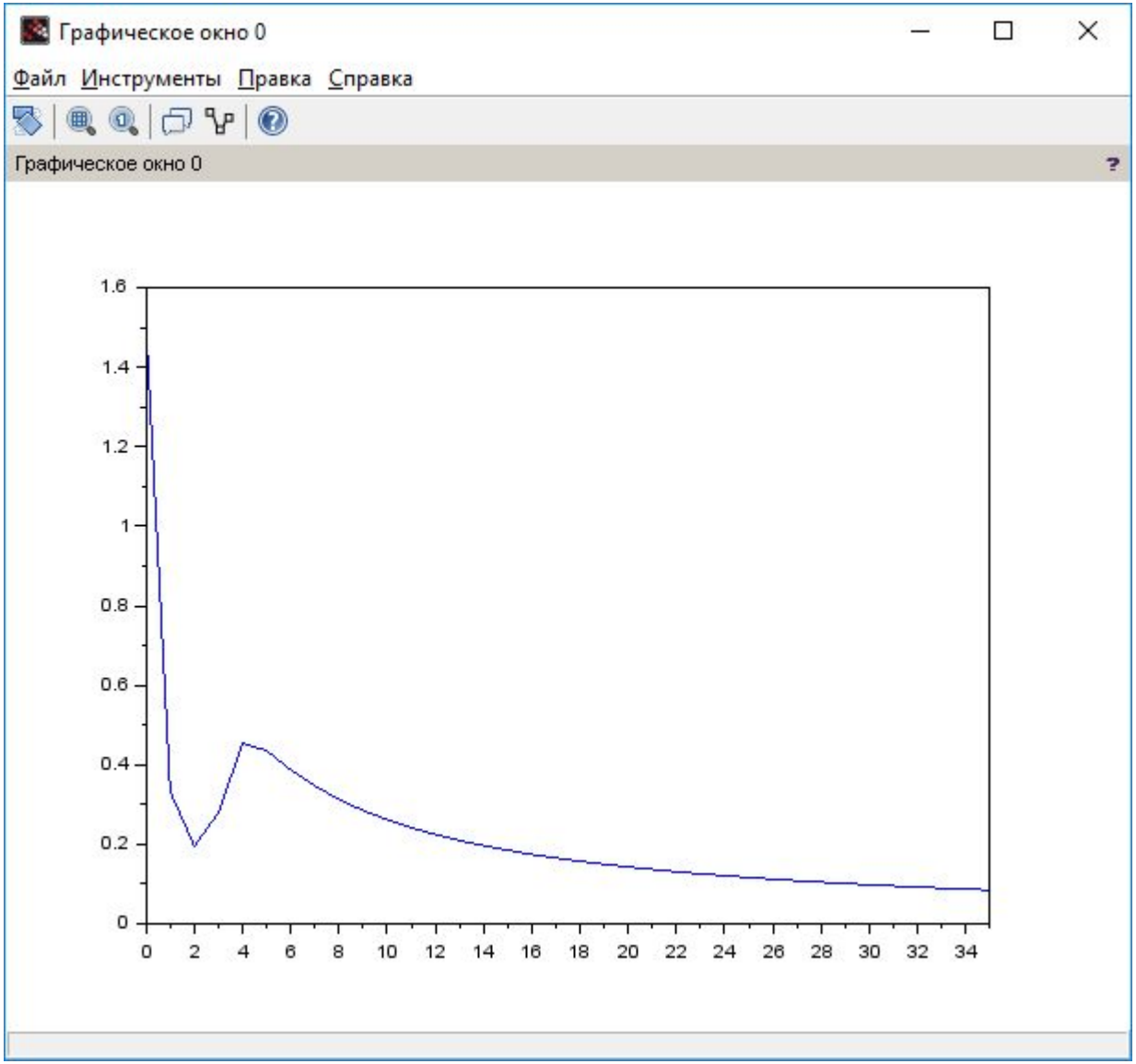
x0, t0 —начальные условия

x – результат работы функции

Фрагмент программы:

```
function y=f(t,x)
y=sin(x*t)-x
endfunction

x0=1.5
t0=0

t=0:0.5:35

y=ode(x0,t0,t,f)

plot(t,y)
```

Графическая интерпретация решения:



Решить задачу Коши для системы дифференциальных уравнений вида:

x' = cos(xy)

y' = sin(x + ty)

x(0) = 0 y(0) = 0 на интервале **[0; 10]**

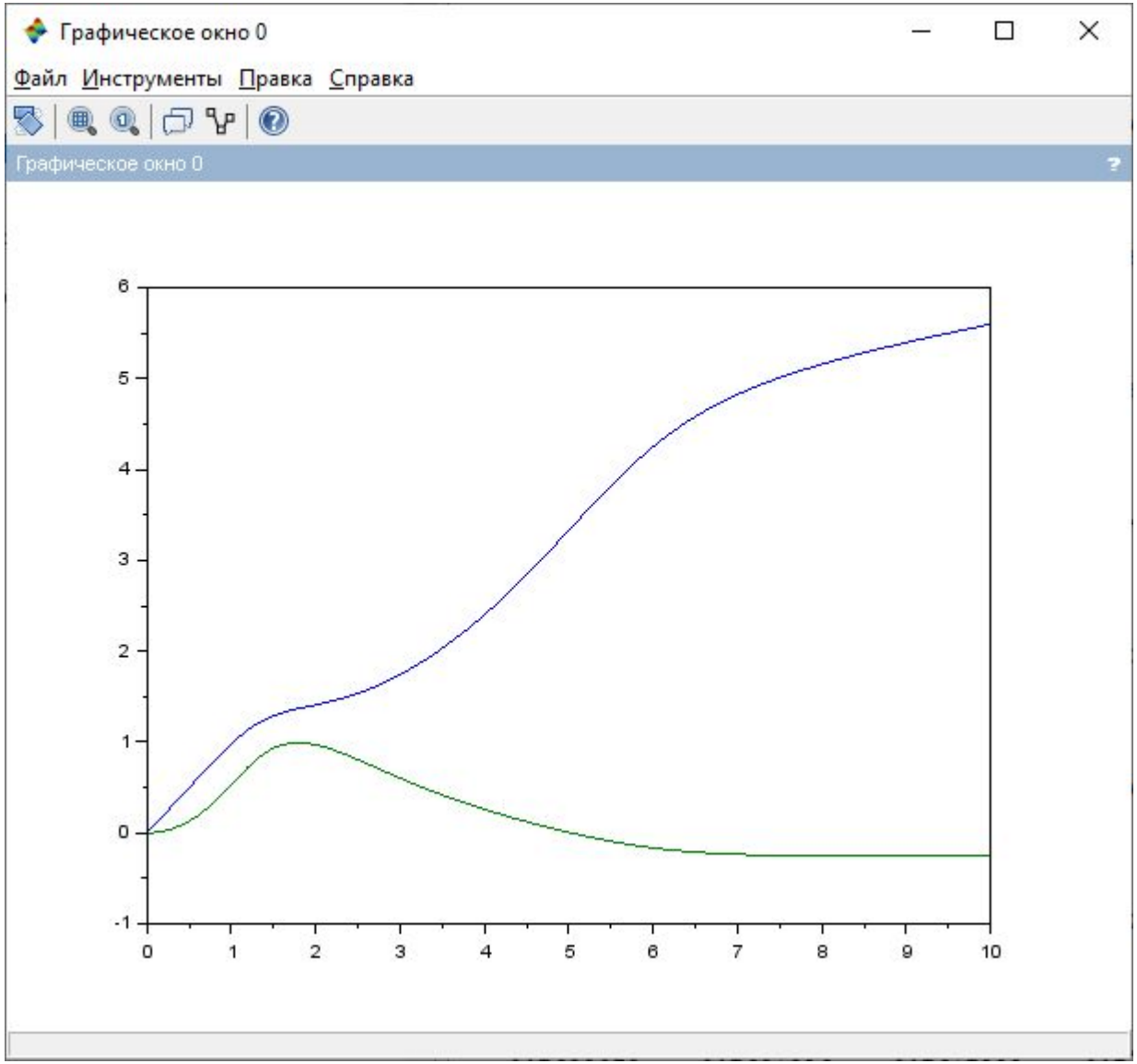
Фрагмент программы:

```
function [q]=ff(t,x)
   q(1)=cos(s(1)*s(2))
   q(2)=sin(s(1)+s(2)*t)
endfunction
x0=[0;0]
t0=0
t=0:0.1:10
y=ode(x0,t0,t,ff);
plot(t,y)   или  plot(t, y(1,:), t, y(2,:))
                  x(t)        y(t)
```

Обозначим x как s(1), y как s(2)

q – вектор результата

Графическая интерпретация решения:



Пример задачи для экзамена

Построить график функции:

$$Z(\varphi) = b\sqrt{f} + a\frac{k}{m} - c\,w\varphi^2$$

где k=$sin^2 f + cos\ a$

$$m = \frac{\sum_{i=1}^{5}(i-0{,}5)}{\sqrt{|\cos 7{,}86|}} \qquad w = \int_0^{2\pi} \sin\left(\frac{t}{2}\right)dt$$

f – определитель матрицы $\begin{vmatrix} -1 & 2 & 2 \\ 2 & -1 & 3 \\ 0 & 2 & 1 \end{vmatrix}$

$a,\ b,\ c$ – корни системы

$$\begin{cases} 4a+5b+c=17 \\ 2a-3b+c=-1 \\ 4a-2b=0 \end{cases}$$