Направление подготовки: 13.03.02 Электроэнергетика и электротехника

Наименование образовательной программы: Электрический транспорт

Уровень образования: бакалавриат

Форма обучения: очная

Дисциплина:

"Моделирование устройств электрической тяги"

Комаров В.Г. Практическое занятие 1 09.09.2026 г.

Освоение современных инструментов моделирования

Для эффективного моделирования современных электрических транспортных средств и систем нам необходимо освоить современные программные продукты и языки, используемые в качестве инструмента для создания компьютерных моделей. Сегодня мы с вами рассмотрим мощную интегрированную среду математических вычислений и моделирования (ИСМ) Scilab, позволяющую за счёт различных пакетов программ (инструментов — Toolboxes) существенно ускорить и облегчить построение различных компьютерных моделей.

Краткий обзор системы Scilab

Scilab (сокращение от Science Laboratory — Научная Лаборатория) это встроенный базовый язык и пакет прикладных математических программ, предоставляющий мощное открытое окружение для научных, инженерных и технических расчётов, а также моделирования. Встроенный язык Scilang – это интерпретируемый язык структурного программирования (скриптовый язык), весь выполняемый код которого размещается в функциях. В одном файле может быть несколько функций. Однако при разработке пакетов расширений принято хранить каждую функцию в отдельном файле. Он распространяется под лицензией GNU General Public License 2.0 — это лицензия на свободное программное обеспечение. Сайт разработчика: www.scilab.org

Возможности

Scilab содержит сотни математических функций, и есть возможность добавления новых, написанных на различных языках (C, C++ и т. д.). Также имеются разнообразные структуры данных (списки, полиномы, рациональные функции, линейные системы), интерпретатор и язык высокого уровня. Scilab был спроектирован как открытая система, и пользователи могут добавлять в него свои типы данных и операции путём перегрузки.

В системе доступно множество инструментов:

2D и 3D графики, анимация,

Линейная алгебра, разреженные матрицы (sparse matrices),

Полиномиальные и рациональные функции,

Интерполяция, аппроксимация,

Симуляция: решение ОДУ и ДУ,

Xcos (Scicos) - система визуального моделирования динамических систем и симуляции,

Дифференциальные и не дифференциальные оптимизации,

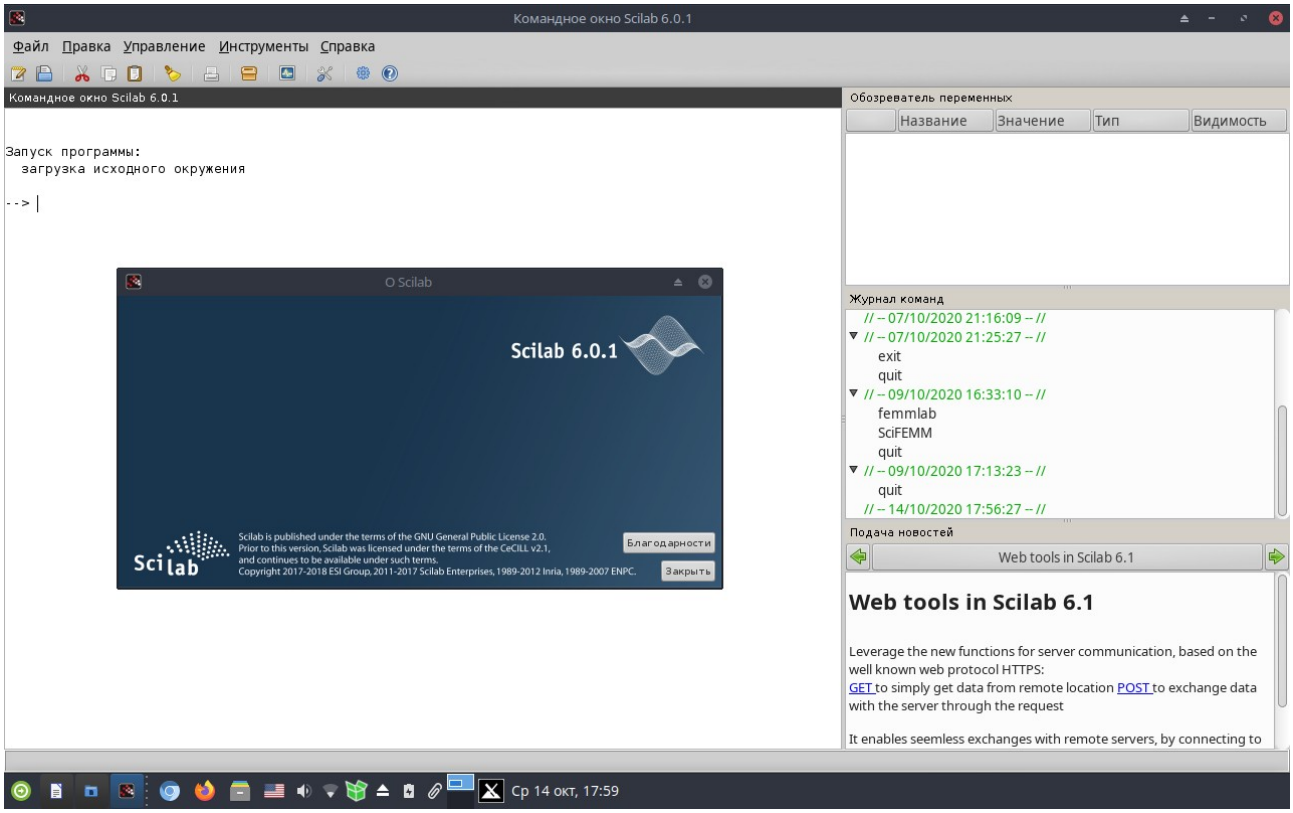
Обработка сигналов,
Параллельная работа,
Статистика,
Работа с компьютерной алгеброй,
Интерфейс к C, C++, Java, Tcl/Tk, LabVIEW.

Scilab имеет схожий с MATLAB язык программирования Scilang. В состав пакета входит утилита, позволяющая конвертировать документы Matlab в Scilab.

Scilab позволяет работать с элементарными и большим числом специальных функций (Бесселя, Неймана, интегральных функций), имеет мощные средства работы с матрицами, полиномами (в том числе и символьно), производить численные вычисления (например, численное интегрирование) и решение задач линейной алгебры, оптимизации и симуляции, мощные статистические функции, а также средства для построения и работы с графиками, структурными и символьными схемами, графическим интерфейсом и анимацией.
Для численных расчётов используются библиотеки Lapack, LINPACK, ODEPACK, Atlas и другие.
В состав пакета также входит Xcos (Scicos)— инструмент для редактирования блочных диаграмм и симуляции. Имеется возможность совместной работы Scilab с программой LabVIEW.

Основы работы в Scilab

При запуске открывается командное окно.



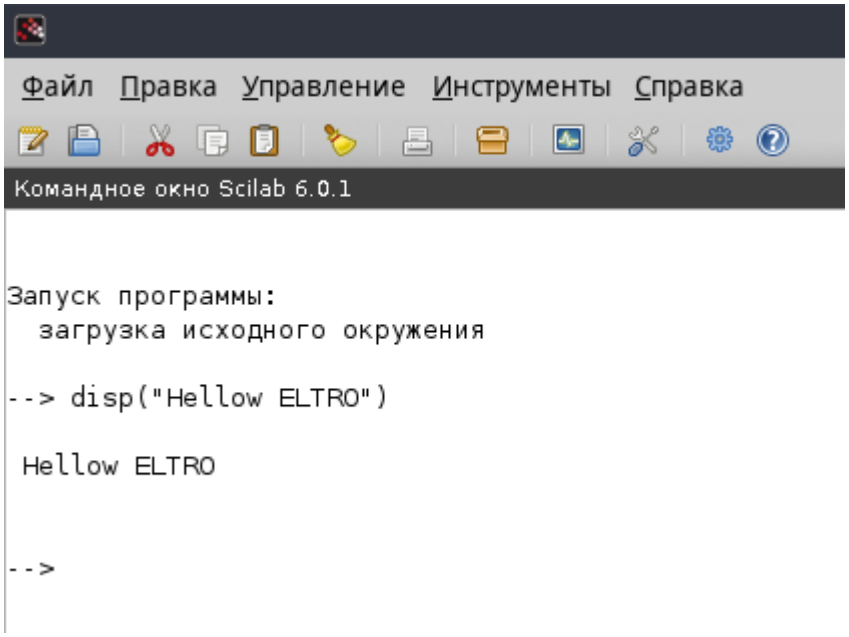
Есть 2 варианта работы:

1 — это работа в режиме командного интерпретатора,

2 — это работа в скриптовом режиме с помощью редактора текста программ SciNotes, где можно написать код, который позднее запустить, результат выполнения появится в командном окне.

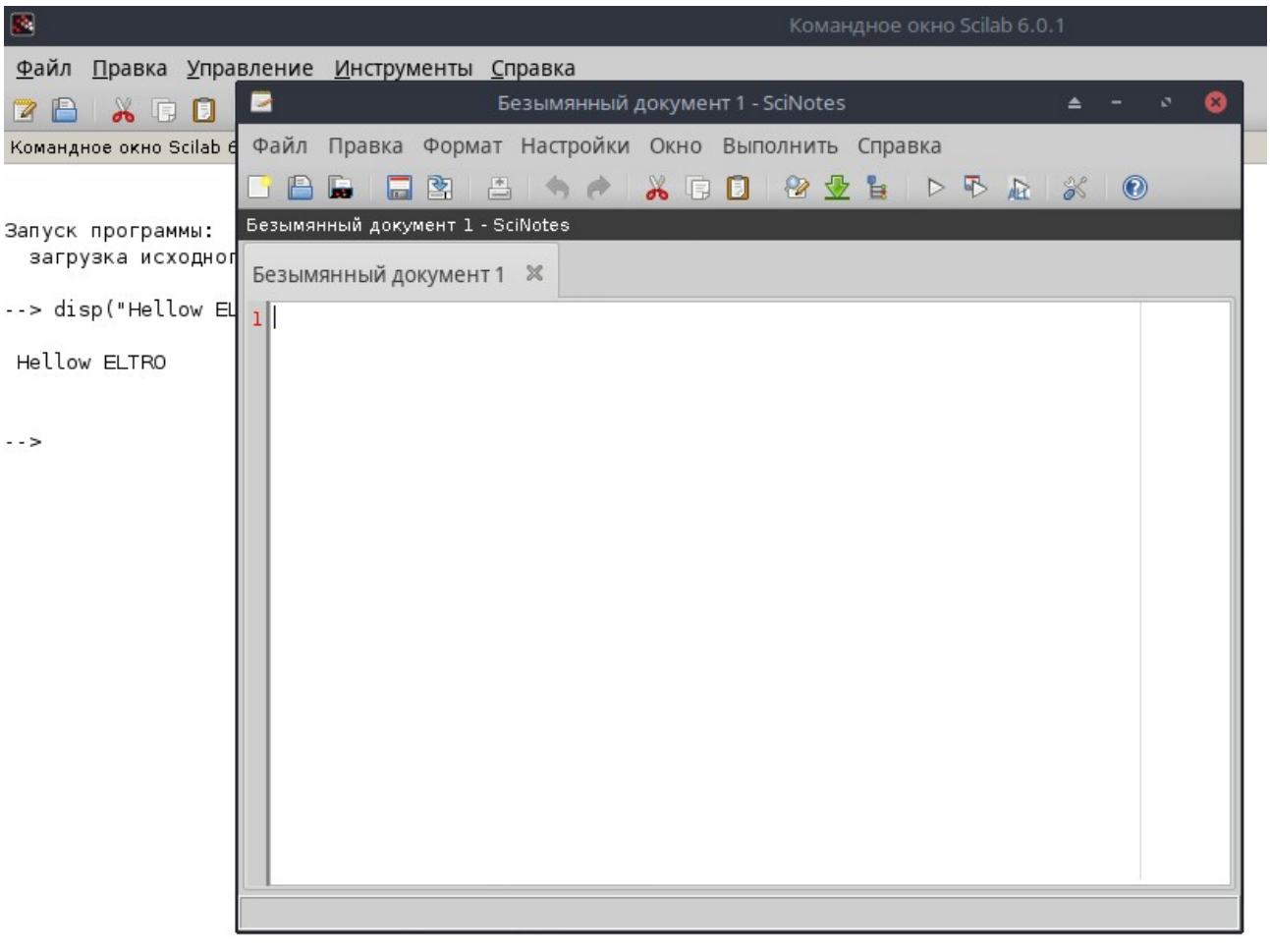
Для примера рассмотрим вывод на дисплей строки Hellow ELTRO.
Используем функцию вывода на дисплей disp().

Работа в режиме командного интерпретатора (командная строка).

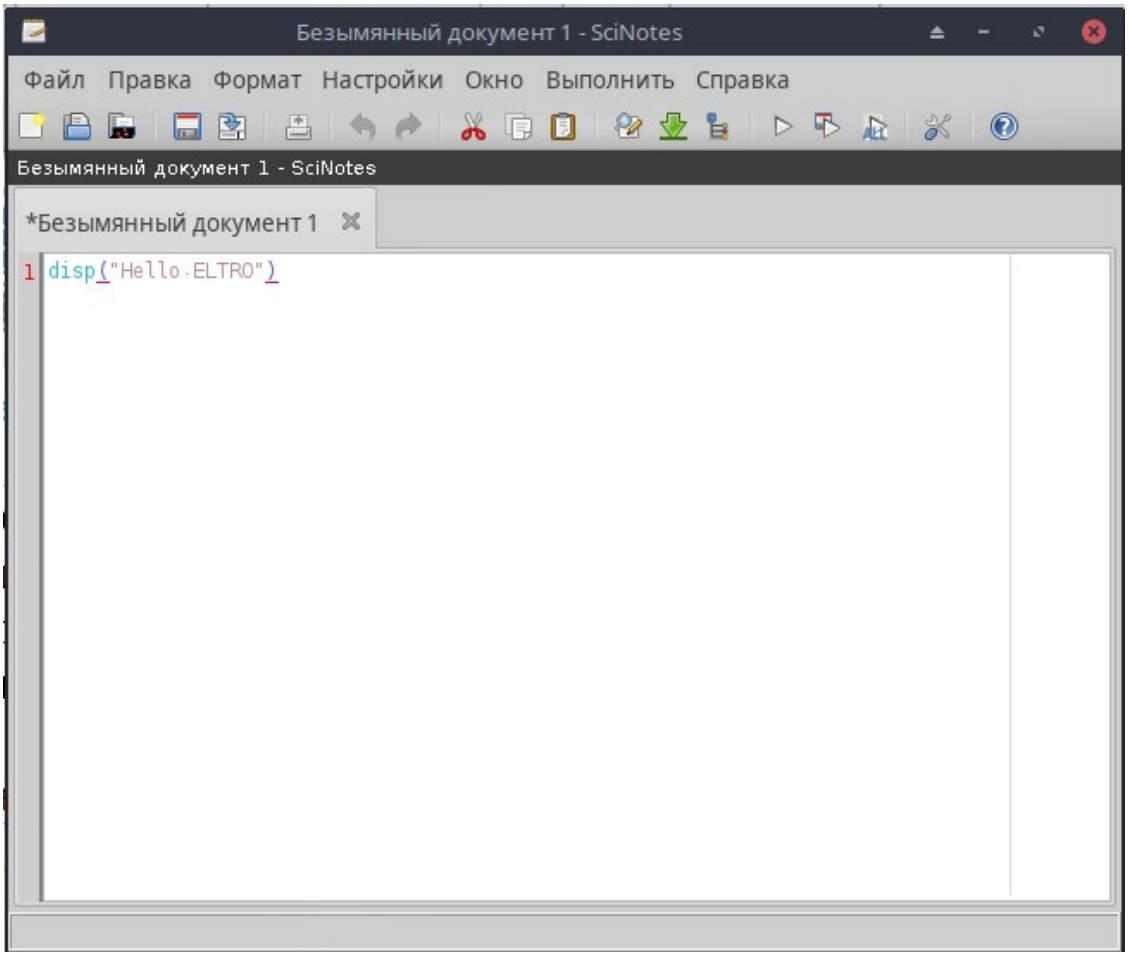


Режим работы в скриптовом режиме (ввод программы в редакторе SciNotes).

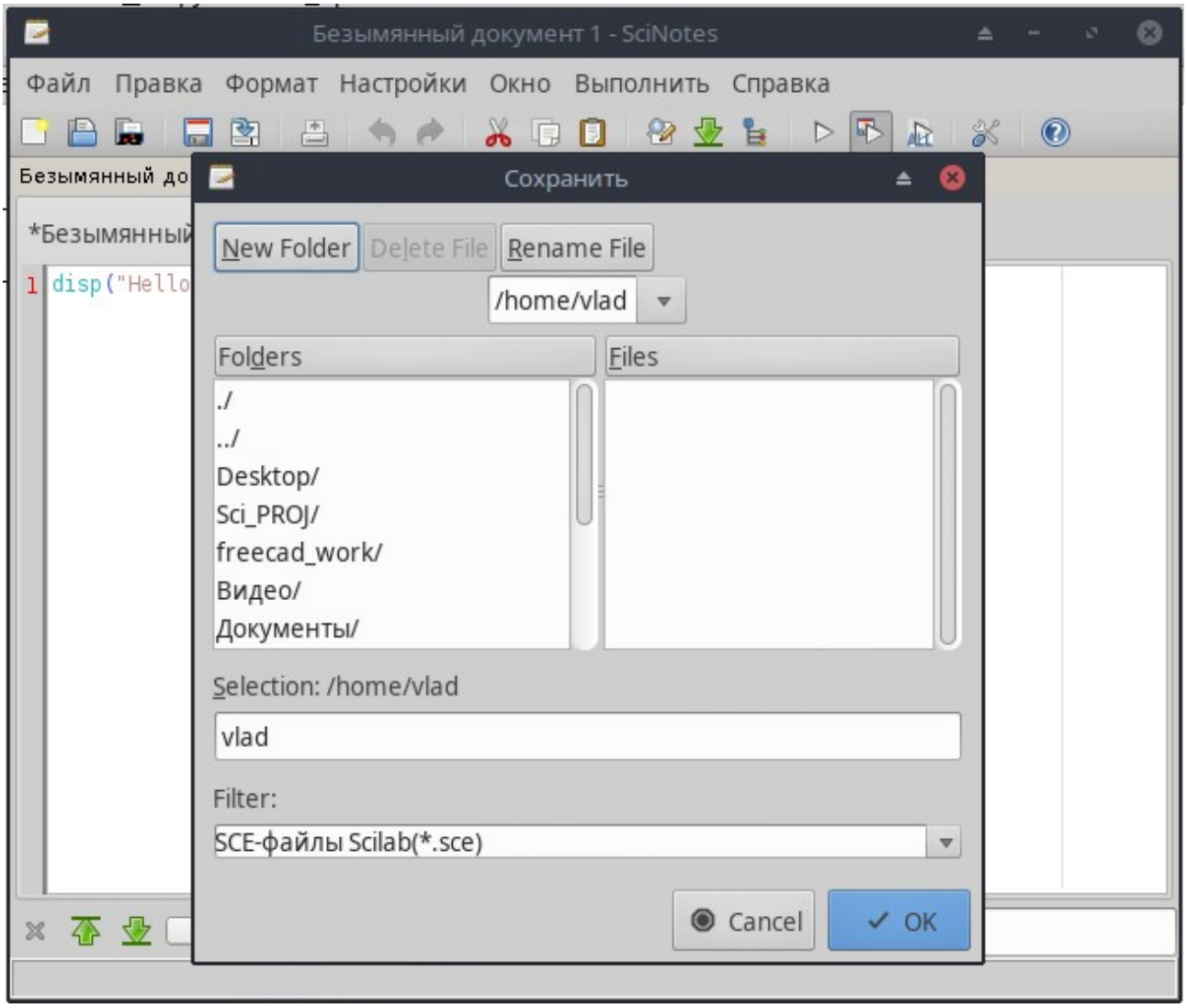
Для этого откроем редактор, кликнув первую иконку главного меню в левом углу.



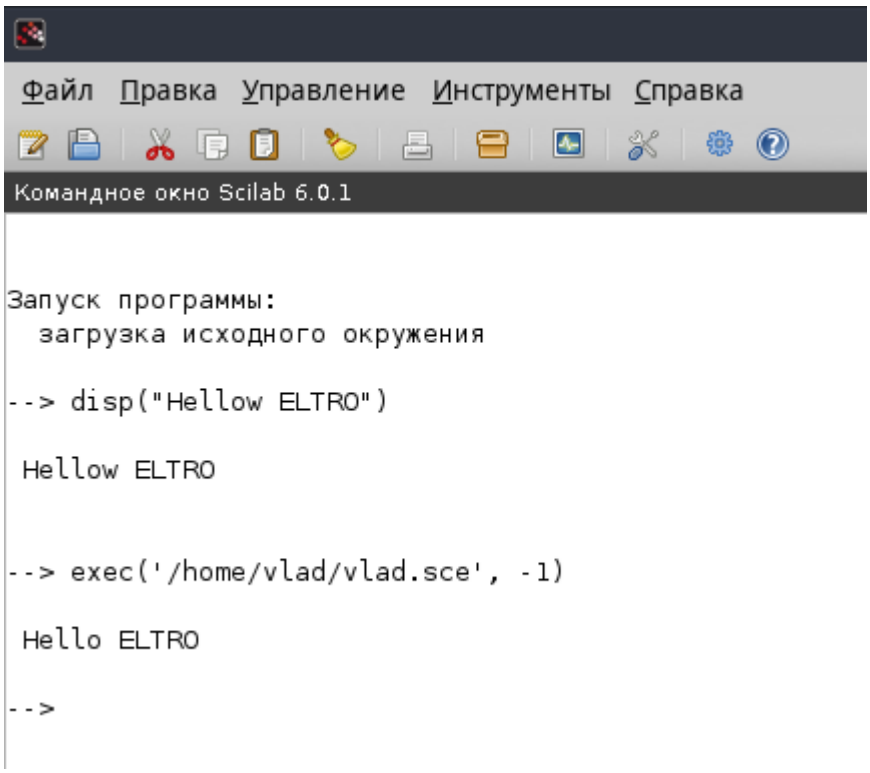
В открывшемся окне редактора кода необходимо ввести команду или текст программы.



Для выполнения кода, надо в главном меню выбрать команду Сохранить и выполнить. Откроется окно Сохранить, в котором необходимо указать место сохранения файла и нажать Ok.



В командном окне после сохранения файла программы будет выведен результат её выполнения.



Можно сказать что интерфейс изучен, далее будем приводить просто код и результат выполнения.

Переменные в языке Scilab не описываются, а создаются путем присвоения им начального значения, например так:

```
a = 1
b='Hello'
c= %t
```

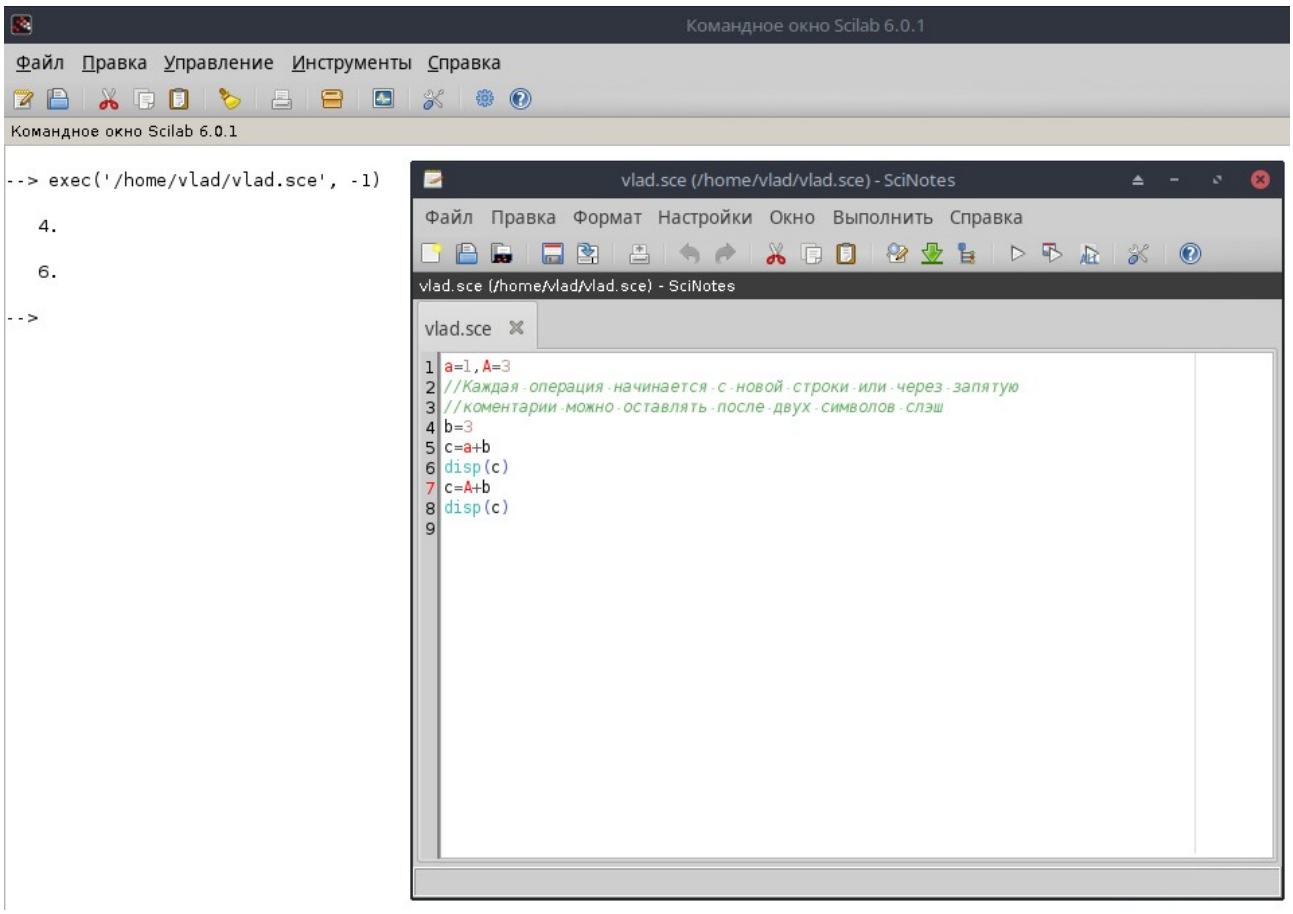
Переменные в Scilab не имеют строгой типизации, т. е. если в переменной хранился текст, то можно на следующем шаге записать в нее число, а затем логическое значение. Scilab следит за соответствием типов только при вычислении значений выражений. Переменные, созданные внутри функции, являются локальными и действуют только в пределах этой функции. Переменные, созданные в пространстве до начала функции, являются глобальными и доступны во всех функциях данного файла или текущей рабочей сессии.

```
def_base=2 //глобальная переменная

function rez=log_b(num, base)
   chk_log=%f //локальная переменная

   rez=log(num)/log(base)
endfunction
```

Язык SciLab различает регистр, т.е. A и a — разные переменные. Например:



Основные операции в Scilab

\+ сложение

— вычитание

* умножение

/ деление справа, т.е. x/y = xy^(-1)

\ деление слева, т.е. x\y = x^(-1)y

^ возведение в степень, т.е. x^y

** возведение в степень (эквивалентно ^)

’ эрмитово сопряжение (комплексное сопряжение и транспонирование)

Изначально эти операции служат для выполнения матричных действий по правилам матричной алгебры. Например:

```
1 -->a=[1 2 3],b=[3 2 1]
2 a =
3   1.  2.  3.
4 b =
5   3.  2.  1.
6
7 -->a*b
8   !--error 10
9 Некорректное умножение.
```

Здесь сделана попытка перемножить две строки, но по правилам матричной алгебры это нельзя сделать. Одну из строк необходимо транспонировать, чтобы получился столбец. Кроме того, согласно правилам матричной алгебры, важен порядок множителей:

```
1  -->a*b'
2  ans =
3     10.
4
5  -->b' * a
6  ans =
7     3.  6.  9.
8     2.  4.  6.
9     1.  2.  3.
```

Для выполнения поэлементного умножения двух массивов необходимо использовать признак поэлементного действия, т. е. поставить перед знаком действия точку (точка и знак действия пишутся слитно, без пробела):

```
1  -->a .* b
2  ans =
3     3.  4.  3.
```

То же самое относится и ко всем остальным действиям кроме операции транспонирования.

Типы чисел в SciLab

y = int8(x) 8-битовое число со знаком [-2^7; (2^7)-1] = [-128; 127]
y = uint8(x) 8-битовое число без знака [0; (2^8)-1] = [0; 255]
y = int16(x) 16-битовое число со знаком [-2^15; (2^15)-1] = [-32768; 32767]
y = uint16(x) 16-битовое число без знака [0; (2^16)-1] = [0; 65535]
y = int32(x) 32-битовое число со знаком [-2^31; (2^31)-1] = [-2147483648; 2147483647]
y = uint32(x) 32-битовое число без знака [0; (2^32)-1] = [0; 4294967295]

`iconvert` преобразование к целочисленному представлению
`inttype` определение типа целого числа

Функции в Scilab.

Элементарные математические функции.

```
acos acosd acosh acoshm acosm acot acotd acoth
acsc acscd acsch asec asecd asech asin asind
asinh asinhm asinm atan atand atanh atanhm atanm
cos cosd cosh coshm cosm cotd cotg coth
cothm csc cscd csch sec secd sech sin
sinc sind sinh sinhm sinm tan tand tanh
tanhm tanm
exp expm log log10 log1p log2 logm max
maxi min mini modulo pmodulo sign signm sqrt
sqrtm
```

Способы объявления пользовательских функций.

В Scilab, как и в языке Си всё есть функция. Пользовательские функции создаются с помощью редактора кода в виде файлов-сценариев.

Простейший способ объявления пользовательской функции:

```
outvar = myfunction ( invar )
```

В общем случае описание функции выглядит следующим образом:

```
function [выходные параметры]=имя_функции(входные параметры)
   …
   тело функции
   …
   [выходные параметры]=…
endfunction
```

Если у функции всего один выходной параметр, то его можно не заключать в квадратные скобки, если же их больше одного, то они заключаются в скобки и перечисляются через запятую. Создавая функцию, всегда помните, что входной параметр может быть массивом, и в критических случаях предусматривайте выполнение проверки на размер массива.

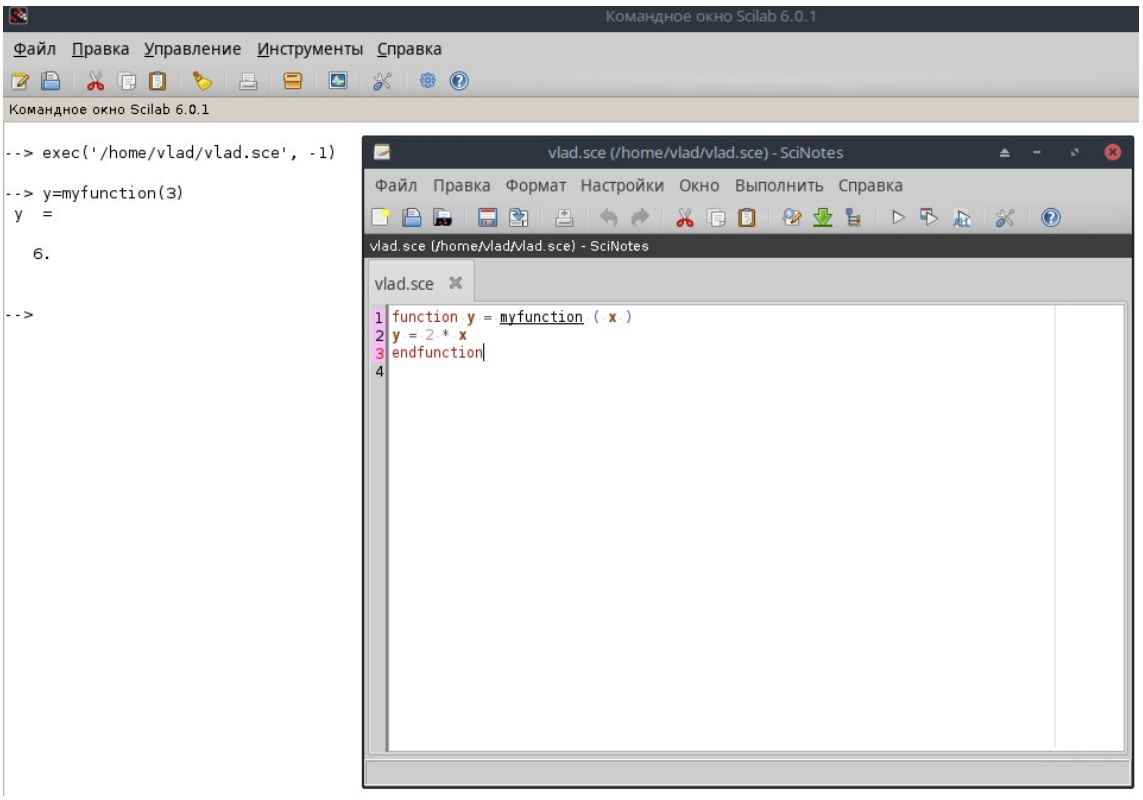
Пример объявления и создания пользовательской фунции:

Открываем редактор кода и вводим текст (код), описывающий функцию на языке Scilab

```
function y = myfunction ( x )
y = 2 * x
endfunction
```

Сохраняем её.

Далее приведён пример вызова данной функции



Массивы (матрицы)

Пример задания одномерного массива:

```
-->A = [1, 2, 3, 4, 5, 6]
A =
1. 2. 3. 4. 5. 6.
```

Задание двухмерного массива:

```
-->A = [1, 2, 3; 4, 5, 6]
A =
1. 2. 3.
4. 5. 6.
```

Квадратные скобки обозначают начало и конец перечисления элементов матрицы, запятой отделяются элементы матрицы, находящиеся в одной строке, точка с запятой разделяет строки матрицы.
Команды:
`size` определить размер матрицы
`matrix` изменить размер матрицы
`resize_matrix` создать новую матрицу заданного размера и скопировать в нее элементы из исходной матрицы.

Операции над матрицами:

Обращение к элементам матрицы

`i = 1; 2`, - это номера строк, а `j = 3; 4` — номера столбцов
Для примера возьмём уже готовую матрицу

```
-->A = testmatrix (" hilb ", 5)
A =
25. – 300. 1050. – 1400. 630.
– 300. 4800. – 18900. 26880. – 12600.
1050. – 18900. 79380. – 117600. 56700.
– 1400. 26880. – 117600. 179200. – 88200.
630. – 12600. 56700. – 88200. 44100.
-->A(1: 2, 3: 4)
ans =
1050. – 1400.
– 18900. 26880
```

`A` матрица целиком
`A(:,:)` матрица целиком
`A(i:j,k)` элементы матрицы в k-ом столбце с i-ой по j-ую строку
`A(i,j:k)` элементы матрицы в i-ой строке с j-ого по k-ый столбец
`A(i,:)` i-ая строка матрицы
`A(:,j)` j-ый столбец матрицы

Генерация единичной матрицы

```
-->A = ones (3, 3)
A =
1. 1. 1.
1. 1. 1.
1. 1. 1.
```

Операции над матрицами

+ сложение .+ поэлементное сложение

— вычитание .- поэлементное вычитание

* умножение .* поэлементное умножение

/ деление справа ./ поэлементное деление справа

\ деление слева .\ поэлементное деление слева

^ или * возведение в степень :^ поэлементное возведение в степень

’ эрмитово сопряжение (комплексное сопряжение и транспонирование)

.’ транспонирование без сопряжения

Пример умножения числа на единичную матрицу 2 на 2

```
-->B = 2 * ones (2, 2)
B =
2. 2.
2. 2.
```

Численное интегрирование

Численное интегрирование (историческое название: (численная) квадратура) — вычисление значения определённого интеграла (как правило, приближённое). Под численным интегрированием понимают набор численных методов отыскания значения определённого интеграла.

Интегрирование по методу трапеций

Проинтегрируем функцию, корень из *2*x-1* на отрезке от 1 до 10 с разбиением в 1 шаг

```
-->x=1:10
x =
1. 2. 3. 4. 5. 6. 7. 8. 9. 10.
-->y=sqrt(2*x-1)
y =
column 1 to 6
1. 1.7320508 2.236068 2.6457513 3. 3.3166248
column 7 to 10
3.6055513 3.8729833 4.1231056 4.3588989
-->inttrap(x,y)
ans =
27.211585
```

Квадратурные формулы Ньютона Котеса

```
-->integrate('(2*x-1)^0.5','x',5,13)
ans =
32.666667
```

Численное дифференцирование

Численное дифференцирование — совокупность методов вычисления значения производной дискретно заданной функции. В основе численного дифференцирования лежит аппроксимация функции, от которой берется производная, интерполяционным многочленом. Все основные формулы численного дифференцирования могут быть получены при помощи первого интерполяционного многочлена Ньютона.

В точке

```
-->function f=myf(x), f=(x+2)^3+5*x, endfunction;
-->numdiff(myf,1)
ans =
32.
-->x=1;3*(x+2)^2+5
ans =
32.
```

на отрезке

```
v=0:3;
-->numdiff(my,v)
ans =
17. 0. 0. 0.
0. 32. 0. 0.
0. 0. 52.999999 0.
0. 0. 0. 80.000002
```

и проверка

```
-->function f1=my1(x), f1=3*(x+2)^2+5, endfunction;
-->my1(v)
ans =
17. 32. 53. 80.
```

Решение обыкновенных дифференциальных уравнений (ОДУ) средствами SciLab.

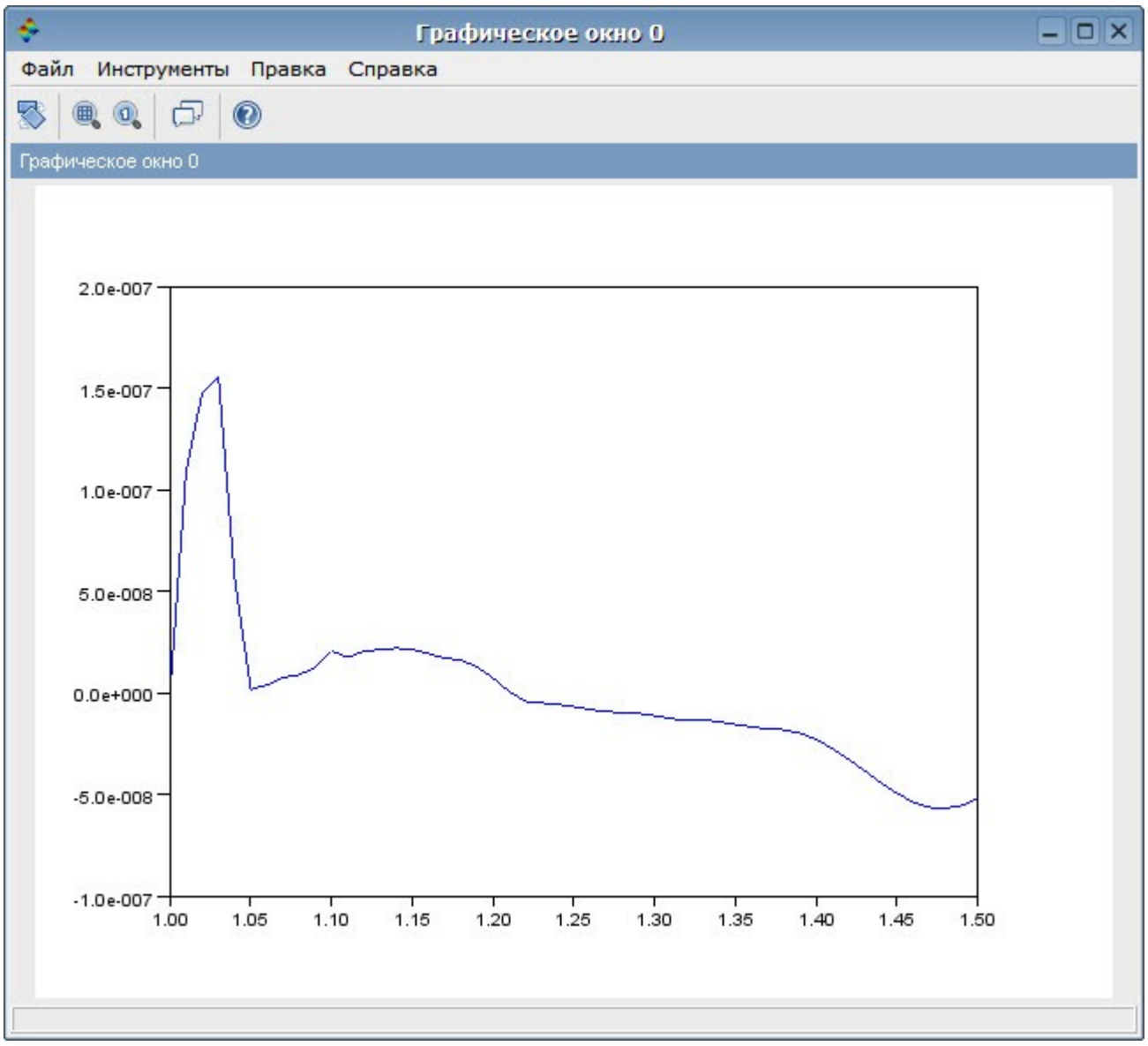
Существует 4 способа для решения ОДУ:

1. С помощью команды ode, которая является решателем (солвером) обыкновенного дифференциального уравнения.
2. С помощью команды odedc, которая вычисляет решение смешанной дискретно-непрерывной системы.
3. Команда dassl, которая дает решение неявно выраженного дифференциального уравнения.
4. С помощью команды impl, которая дает решение неявно выраженного линейного дифференциального уравнения.

Пример:

```
-->y0=1;
-->t0=1;
-->t=1:0.01:1.5;
-->deff("[ydot]=f(t,y)",«ydot=y^(1/3)*t»)
-->y=ode(y0,t0,t,f);
-->y_exact=((t^2+2)/3)^(1.5);// это функция точного решения для
сравнения
-->my_er=y-y_exact;
-->plot(t,y-y_exact) // это график ошибки вычисления от аргумента t
```

результатом является такой график



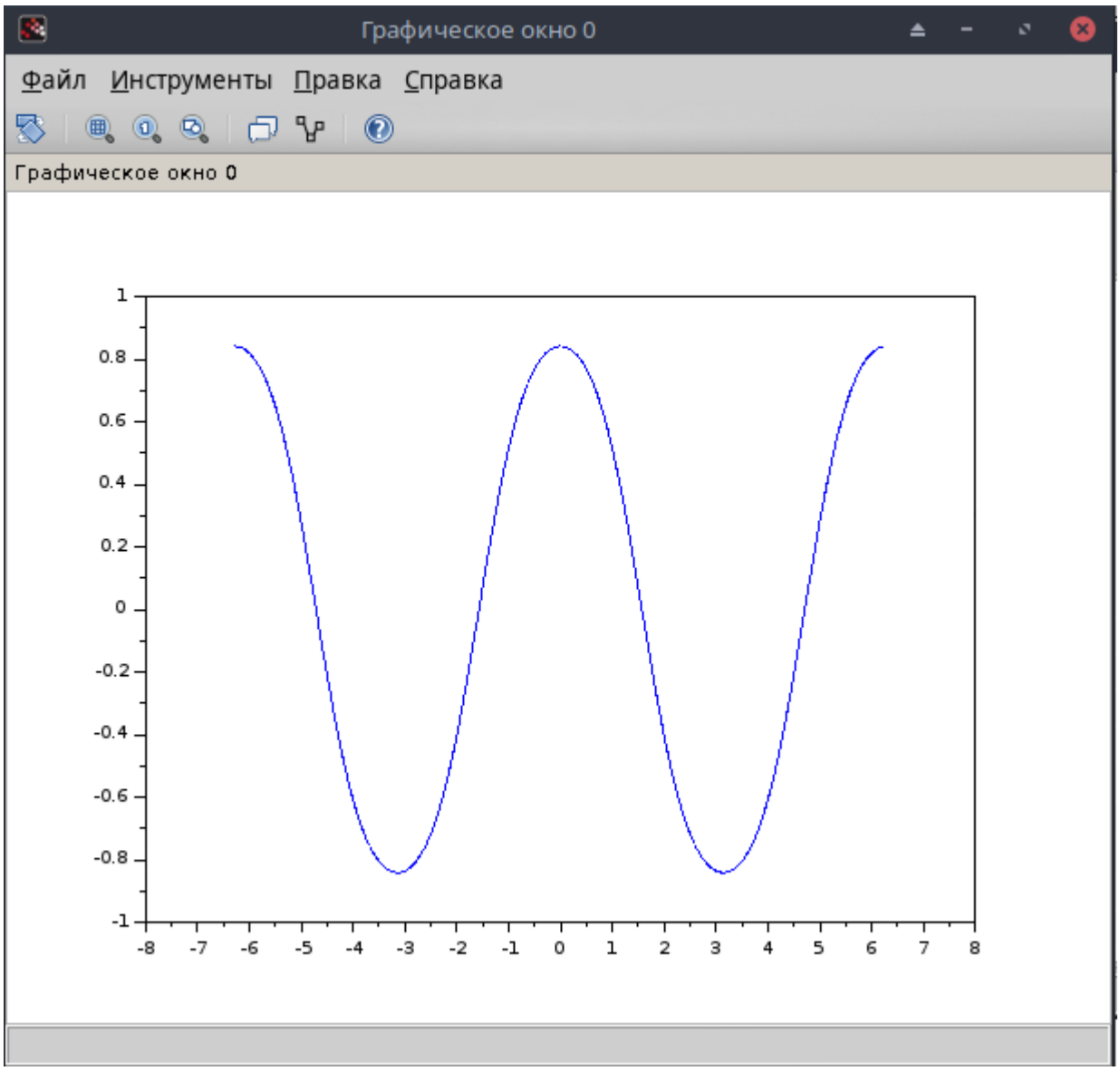
Построение двухмерных графиков в системе SciLab. Основные функции и типы графиков.

Функция plot

Рассмотрим пример:

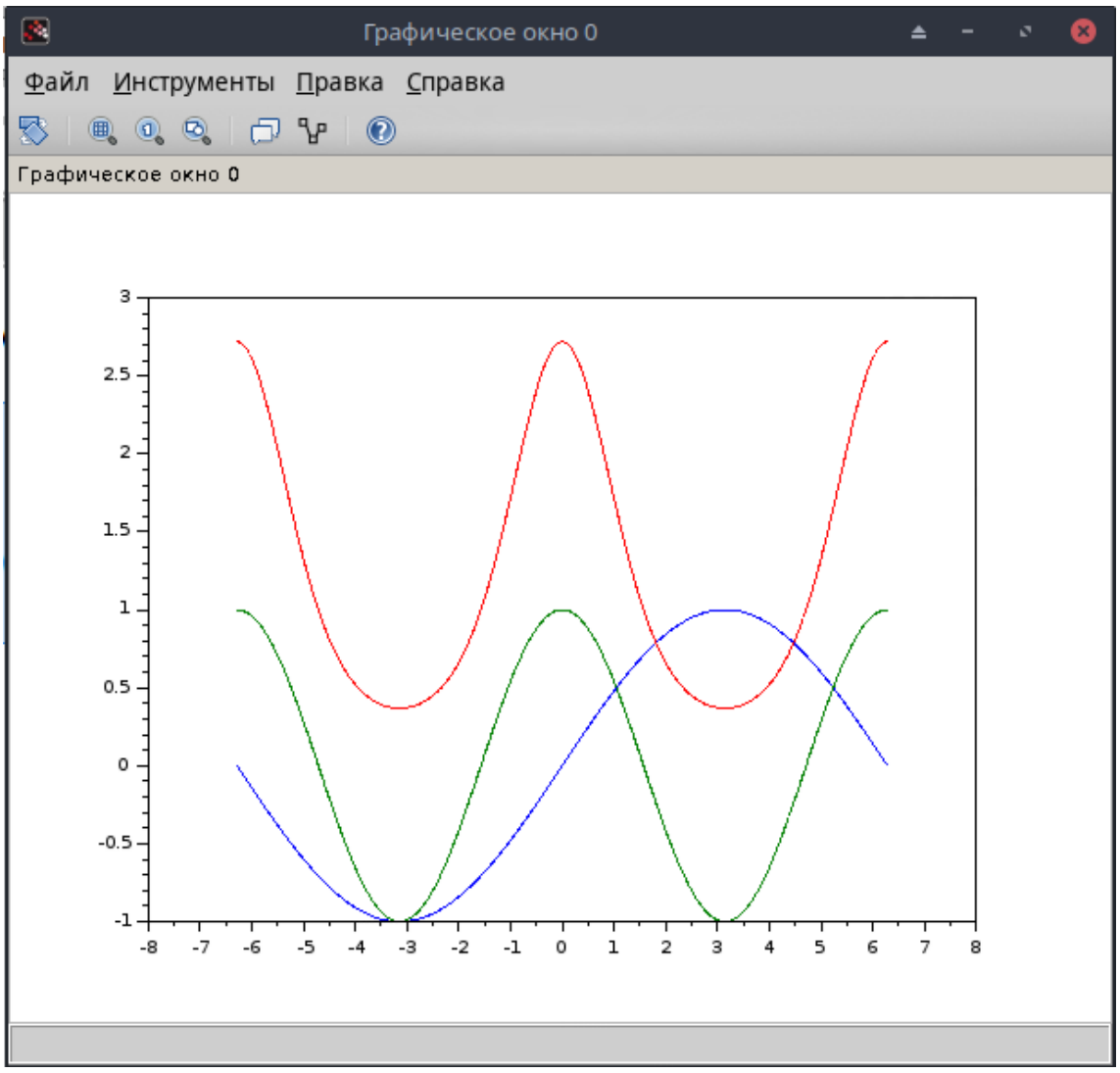
```
x=-2*%pi:0.1:2*%pi;
y=sin(cos(x));
plot(x,y);
```

как можно заметить первый параметр функции — это заданный с шагом интервал аргумента, а второй значения функции для этого интервала. В графическом окне есть возможность экспорта данных, т.е. сохранить картинку.



Также существует возможность нарисовать сразу несколько функций, если их перечислить:

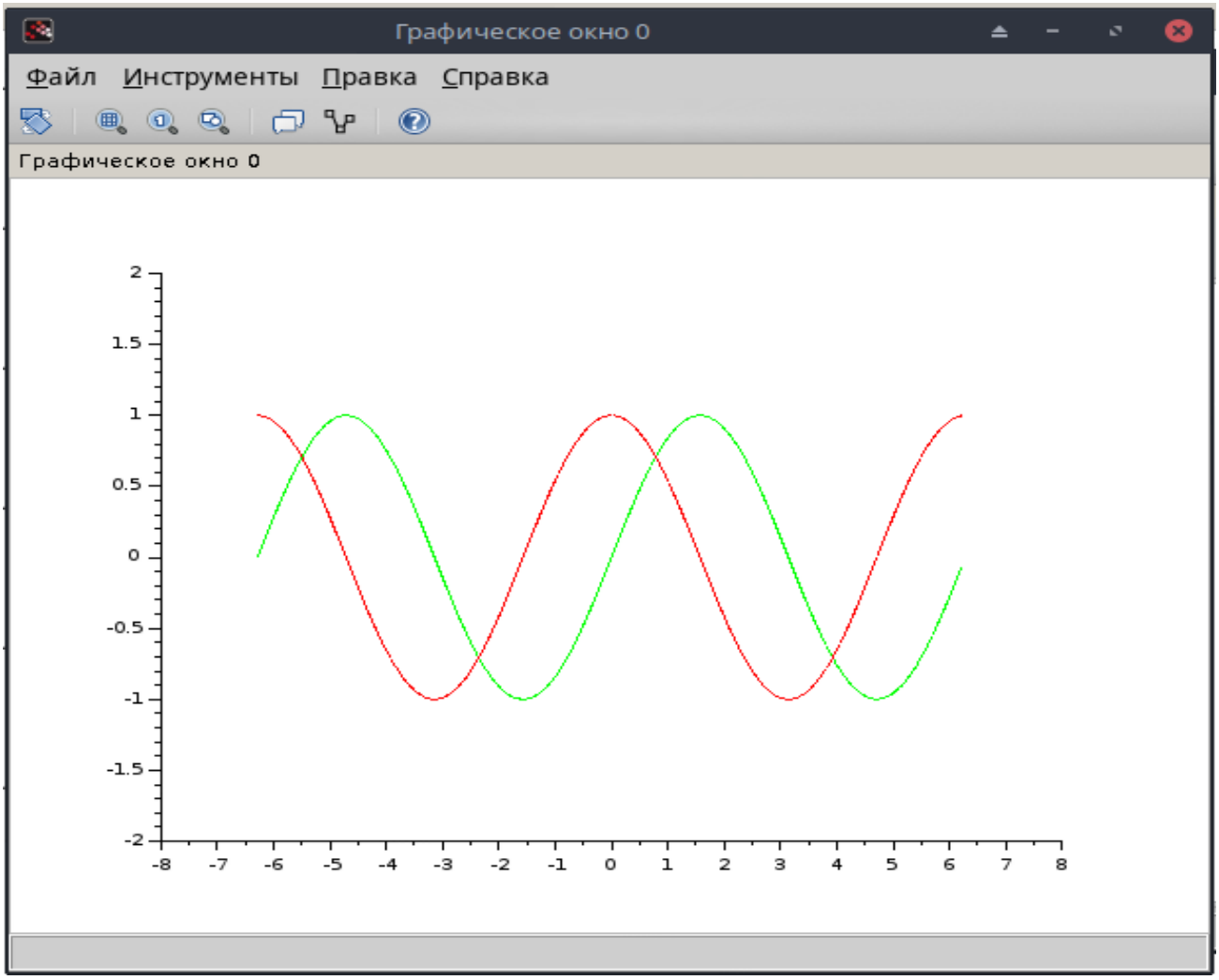
```
x=-6.28:0.02:6.28;
y=sin(x/2);
z=cos(x);
v=exp(cos(x));
plot(x,y,x,z,x,v);
```



Функция plot2d

Рассмотрим функцию опять на примере:

```
x=[-2*%pi:0.1:2*%pi];
y=[sin(x); cos(x)];
plot2d(x,y',style=[3,5],rect=[-8,-2,8,2])
```

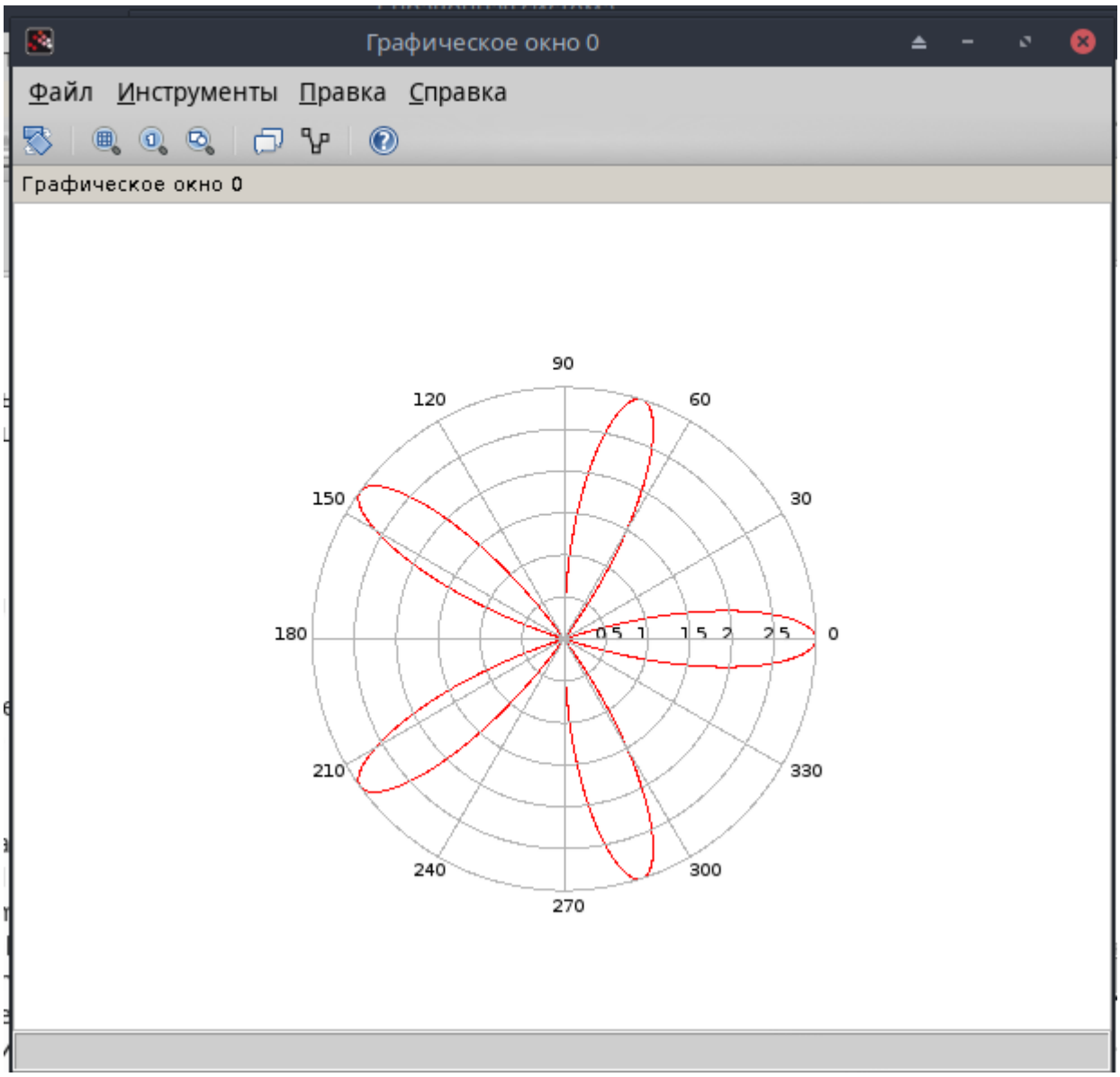


Как мы можем увидеть, у этой функции намного больше функционала. Фунции передаются сразу массивом, так же можно указать цвет линий и интервал вывода.

Функция polarplot

Служит для построения графика в полярных координатах

```
fi=0:0.01:2*%pi;
ro=3*cos(5*fi);
ro1=3*cos(3*fi);
polarplot(fi,ro,style=5));
```



Параметры похожи как и в случае с plot2d.

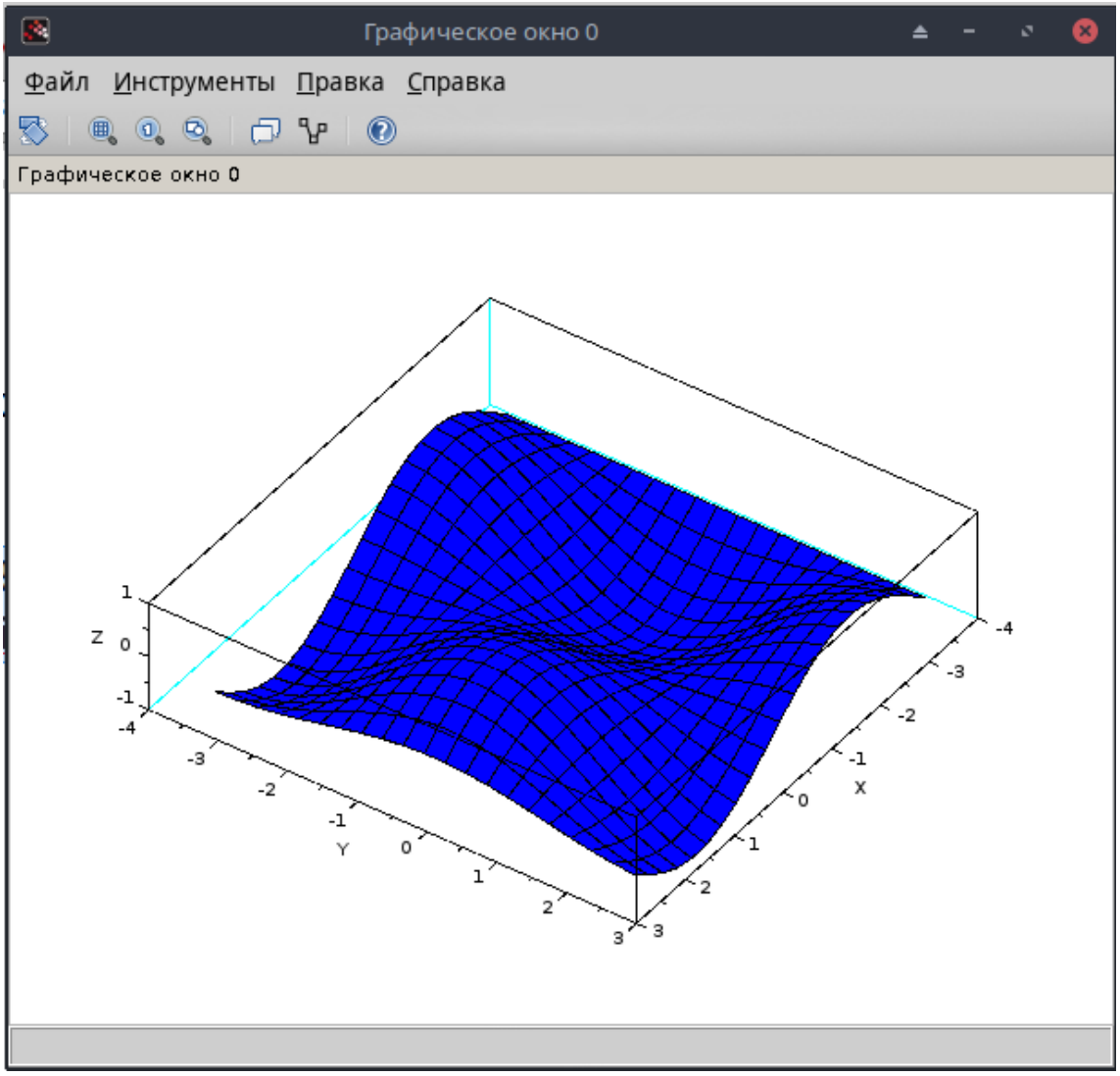
Построение трёхмерных графиков в системе SciLab.

Существует 4 способа построение графика:
Способ 1.
С помощью команды plot3d. Команда создает 3D график по точкам, заданным матрицами x, y и z.
Способ 2.
С помощью команды plot3d1. Команда создает 3D график по точкам, заданным матрицами x, y и z с помощью уровней цвета. Вещь в общем избыточная: величина координаты z дополнительно еще и покрашена, в зависимости от принимаемого значения z.
Способ 3.
С помощью команды fplot3d. Это аналог команды plot3d, но изображаемая поверхность задана с помощью внешней функции.
Способ 4.
С помощью команды fplot3d1. Это аналог команды plot3d1, но изображаемая поверхность задана с помощью внешней функции.

Синтаксис этих команд смотри с помощью help.

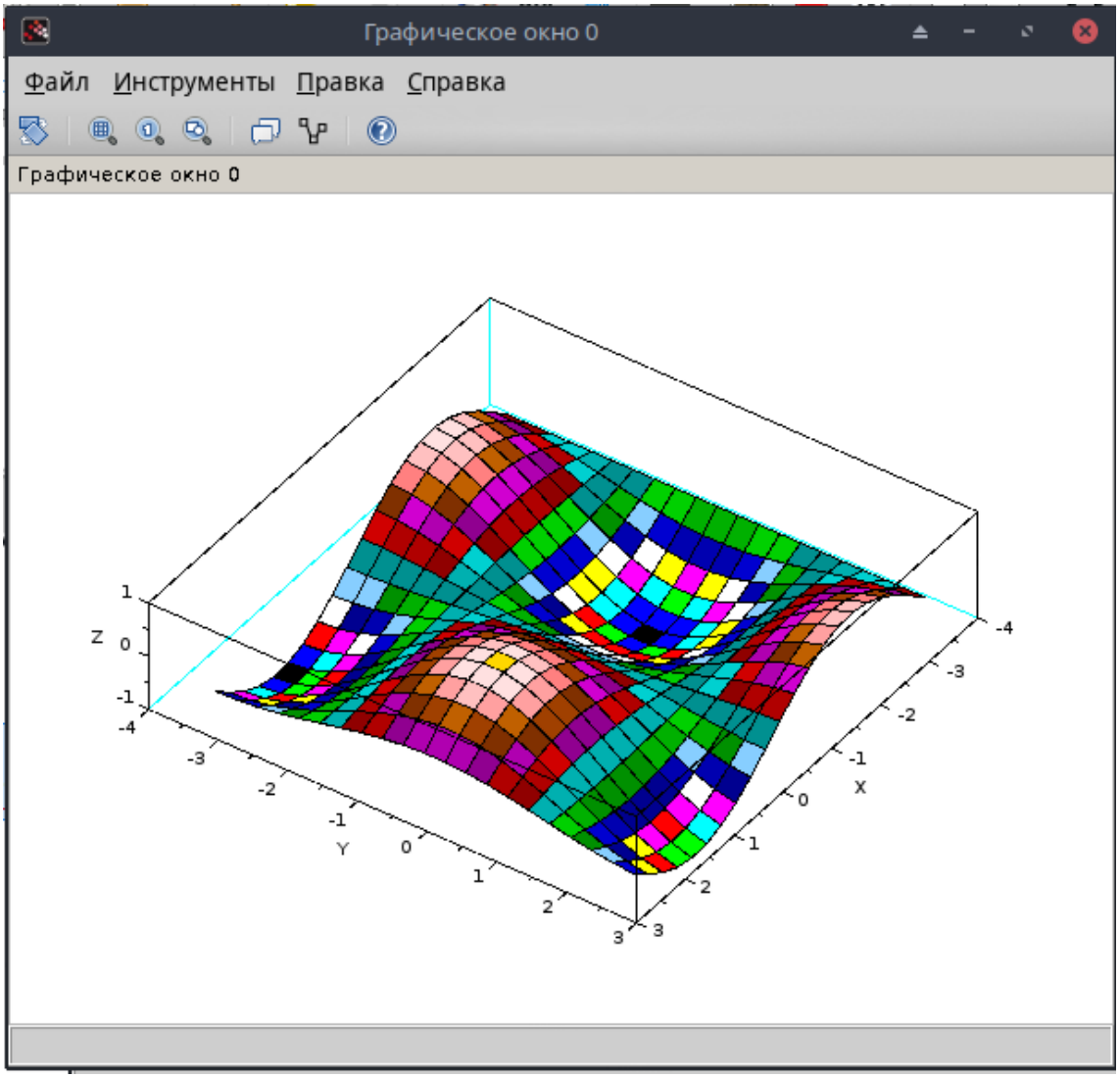
Пример 1.

```
t=-%pi:0.3:%pi;
plot3d(t,t,sin(t)'*cos(t),35,45,'X@Y@Z',[2,2,4]);
```



Пример2.

```
t=-%pi:0.3:%pi;
plot3d1(t,t,sin(t)'*cos(t),35,45,'X@Y@Z',[2,2,4]);
```



Пример 3.

```
deff('[z]=surf(x,y)','z=sin(x)*cos(y)');
t=-%pi:0.3:%pi;
fplot3d(t,t,surf,35,45,«X@Y@Z»);
```

Результат такой же как в примере 1.

Пример 4.

```
deff('[z]=surf(x,y)','z=sin(x)*cos(y)');
t=-%pi:0.3:%pi;
fplot3d1(t,t,surf,35,45,«X@Y@Z»);
```

результат такой же, как в примере 2.

Вызов функции, созданной пользователем.

В Scilab нужно явно загрузить скрипт, содержащий пользовательскую функцию. Предположим, что функция *func* находится в файле *func.sce*. Для загрузки скрипта необходимо выполнить команду

```
exec("путь/к/файлу/func.sce")
```

Отметим, что в *func.sce* может содержаться несколько функций и все они будут загружены после выполнения указанной команды.

Подключение дополнительных модулей

Выполняется с помощью системы управления пакетами (модулями) Scilab — ATOMS. Для ее запуска в командном окне Scilab набирают:

```
atomsGui();
```

или пользуются меню

а затем находят и устанавливают нужный пакет, следуя приведенным там инструкциям.

Список используемых источников:

1. http://ru.wikipedia.org/wiki/Scilab

Задание для самостоятельного выполнения и составления отчёта.

Разберитесь детально с примерами на движение транспортного средства из лабораторной работы №1 и постройте графики *v(t)* и *x(t)* для вашего варианта с аналитическим и численным решением дифференциального уравнения движения транспортного средства, используя возможности ИСМ Scilab.

Шаблон оформления отчёта

Национальный исследовательский университет "МЭИ"

ИЭТЭ

Кафедра ЭКАОиЭТ.

Отчет по практической работе на тему:

«Интегрированная среда компьютерного моделирования Scilab»

Вариант задания №12

Выполнил:	Иксов А. Б.
Группа:	ЭЛ-05-23
Проверил:	Комаров В. Г.

Москва 2026

Дано.

Транспортное средство движущееся на подъём i(уклон)=0.05, начальная координата X0=10 м, начальная скорость V0=10 м/с.

Задание.

Смоделировать движение транспортного средства при движении на заданном подъёме с построением и анализом кривых движения x(t) [м], V(t) [м/с]. Определить скорость сообщения Vs [км/ч]. (Скорость сообщения- это средняя скорость движения ТС от начальной до конечной точки маршрута с учетом промежуточных остановок).

Построение модели

…

```
//Программа расчёта кривых движения транспортного средства на основе аналитического решения уравнения движения *LR1a_MUET.sce*
t=[0:1:30];
V0=10 //начальная скорость движения, м/с
X0=10 //начальная координата пути, м
g=9.81 //ускорение свободного падения (напряжённость гравитации),м/с^2
i=0.05 //уклон
alfa=atan(i) //угол уклона,рад
v=V0-g*t*sin(alfa) //текущая скорость движения ТС, м/с
x=X0+V0*t-((g*t^2)/2)*sin(alfa) //текущая координата пути, м
plot(t,x,t,v)
xgrid();
xtitle('Скорость движения','v, м/с','Путь','x, м');
legend('Путь','Скорость',2)
```

Результаты моделирования

…

Анализ результатов моделирования

…