Саратовский государственный университет
имени Н.Г. Чернышевского

И.А. Панкратов

Scilab. Первые шаги

*Учебное пособие для студентов
механико-математического факультета*

Саратов, 2012

Содержание

Введение

Scilab – это система компьютерной математики, использующаяся для выполнения таких инженерных и научных вычислений, как

– решение нелинейных уравнений и систем;

– решение задач линейной алгебры;

– решение задач оптимизации;

– дифференцирование и интегрирование;

– решение обыкновенных дифференциальных уравнений и систем и т.д.

Кроме того Scilab предоставляет широкие возможности по созданию и редактированию различных видов графиков и поверхностей.

Несмотря на то, что система Scilab содержит множество встроенных команд, операторов и функций; пользователь может создать новую команду или функцию и использовать её наравне со встроенными. К тому же, система имеет свой достаточно мощный язык программирования высокого уровня, что позволяет решать новые задачи.

По своим возможностям Scilab сопоставим с известным математическим пакетом Mathcad, а по интерфейсу похож на пакет MATLAB. Однако при этом пакет Scilab – свободно распространяемая программа. Существуют версии Scilab для различных операционных систем: Linux, Windows, Mac OS. Последнюю версию пакета всегда можно скачать на официальном сайте программы www.scilab.org. Отметим, что пособие написано на базе версии 5.3.3.

Первый раздел пособия посвящён основам работы в Scilab. Описана работа с переменными, основные операции и функции.

Во втором разделе обсуждаются графические возможности Scilab.

В третьем разделе описана работа с массивами и матрицами в Scilab.

В четвёртом разделе речь идёт о решении обыкновенных дифференциальных уравнений.

В пятом разделе читатель найдёт сведения о программировании в системе Scilab.

Шестой раздел познакомит читателя с визуальным программированием в системе Scilab.

Основы

Примеры программ, приведённые в данном пособии, выполнены в виде файлов-сценариев. Файл-сценарий – это список команд Scilab, сохранённый на диске. Для подготовки, редактирования и отладки файлов-сценариев служит служит специальный редактор SciNotes (в более ранних версиях он назывался SciPad). Окно редактора файлов-сценариев выглядит стандартно, т.е. имеет заголовок, меню, строку состояния. Ввод текста в окно редактора файла-сценария осуществляется по правилам, принятым для команд Scilab.

Для выполнения простейших арифметических операций в Scilab применяются следующие операторы: + – сложение; - – вычитание; * – умножение; / – деление слева направо; \ – деление справа налево; ^ – возведение в степень.

В Scilab можно определять переменные и затем использовать их в выражениях. Любая переменная до использования в формулах и выражениях должна быть определена. В общем виде оператор присваивания выглядит так:

$$\textit{имя\_переменной} = \textit{значение\_выражения}$$

Имя переменной не должно совпадать с именами встроенных функций и переменных системы и может содержать до 24 символов. Отметим, что Scilab различает заглавные и строчнвые буквы в именах переменных. Выражение в правой части оператора присваивания может быть числом, арифметическим выражением, строкой символов или строковым выражением. Если речь идёт о символьной или строковой переменной, то выражение в правой части оператора присваивания следует брать в кавычки. Также в Scilab существуют следующие системные переменные:

%i – мнимая единица;

%pi– число π;

%e– число e;

%inf– машинный символ бесконечности;

%NaN – неопределённый результат;

%eps– условный ноль, %eps $= 2.220 \cdot 10^{-16}$.

Пакет Scilab снабжён достаточным количеством встроенных функций. В табл. 1 приведены элементарные математические функции, используемые чаще всего

Таблица 1 Элементарные математические функции

Функция	Описание функции	Функция	Описание функции
sin(x)	синус числа x	atan(x)	арктангенс числа x
cos(x)	косинус числа x	exp(x)	экспонента числа x
tan(x)	тангенс числа x	log(x)	натуральный логарифм x
cotg(x)	котангенс числа x	sqrt(x)	корень квадратный из x
asin(x)	арксинус числа x	abs(x)	модуль числа x
acos(x)	арккосинус числа x	log10(x)	десятичный логарифм x

Также в Scilab можно создать функцию, определённую пользователем. Сделать это можно, воспользовавшись следующей конструкцией:

```
function [имя1,...,имяN] = имя_функции(переменная1, ...,
 переменнаяM)
тело функции
endfunction
```

где имя1,..., имяN – список выходных параметров; имя_функции – имя, с которым эта функция будет вызываться; переменная1, ..., переменнаяM – выходные параметры.

Все имена переменных внутри функции, а также имена из списка входных и выходных параметров воспринимаются системой как локальные, т.е. считаются определёнными только внутри функции.

Целесообразно сохранять тексты функций в виде отдельных файлов. При этом имя файла должно совпадать с именем функции. Обращение к пользовательской функции осуществляется также, как и к встроенной. Необходимо помнить, что функции, хранящиеся в отдельных файлах,

должны быть предварительно загружены в систему при помощи оператора exec(полный_путь_к_файлу).

Построение графиков

Для построения двумерных графиков можно использовать функцию plot2d. В общем случае обращение к функции имеет вид:

```
plot2d( [logflag,] x, y', [key1=value1,..., keyN=valueN])
```

где logflag – строка из двух символов, каждый из которых определяет тип осей (n – нормальная ось, l – логарифмическая ось), по умолчанию "nn";

x - массив абсцисс;

y – массив ординат или матрица, каждый столбец которой содержит массив ординат очередного графика;

keyi=valuei – последовательностьзначений свойств графика, определяющие его внешний вид.

Возможны следующие значения свойств графика:

style – определяет массив числовых значений цветов графиков. Количество элементов массива совпадает с количеством изображаемых графиков.

rect – это вектор [xmin, ymin, xmax, ymax], определяющий размер окна вокруг графика. Здесь xmin, ymin – положение нижней левой вершины окна; xmax, ymax – положение верхней правой вершины окна.

axesflag – определяет наличие рамки вокруг графика. Необходимо выделить следующие щначения этого параметра:

0 – нет рамки;

1 – рамка есть, ось ординат слева (по умолчанию);

3 – рамка есть, ось ординат справа;

5 – рамка есть, оси координат проходят через центр графика.

Чтобы график проще «читался», удобно выводить сетку. В Scilab это можно сделать с помощью команды xgrid(color), где color определяет цвет линий сетки (по умолчанию рисуется сетка чёрного цвета). Можно

воспользоваться функцией color, которая по названию (color("имя_цвета")) или по коду rgb (color(r, g, b)) формирует нужный цвет.

Заголовок графика можно вывести командой xtitle:

```
xtitle( title , xstr , ystr )
```

где

title – название графика;

xstr – название оси абсцисс;

ystr – название оси ординат.

Для изображения нескольких графиков в одном окне можно воспользоваться функцией subplot. Она разделяет окно на несколько областей. Обращение к ней имеет вид:

```
subplot( m, n, p )
```

Выполнение функции приводит к тому, что графическое окно разбивается на m окон по вертикали и n окон по горизонтали, текущим становится окно с номером p.

Задача 1:

Построить траекторию движения точки, получающегося при сложении взаимно перпендикулярных колебаний разной частоты: $x = a \sin 2\omega t$ м, $y = a \sin \omega t$ м.

Решение:

Вычертим траекторию точки при следующих значениях параметров: $a = 1$ м, $\omega = \pi$ рад / сек. на отрезке времени $[0; \pi]$

Листинг 1: Построение двумерного графика

```
a = 1;
omega = %pi;
h=0.01;
t = 0:h:%pi;

x = a * sin( omega * t );
y = a * sin( 2 * omega * t );

plot2d("nn", x, y', style=color("blue"),
 rect = [-1.25, -1.25, 1.25, 1.25], axesflag=1)
xgrid()
```

```
xtitle("Trajectory", "X", "Y")
```

Результат работы программы:

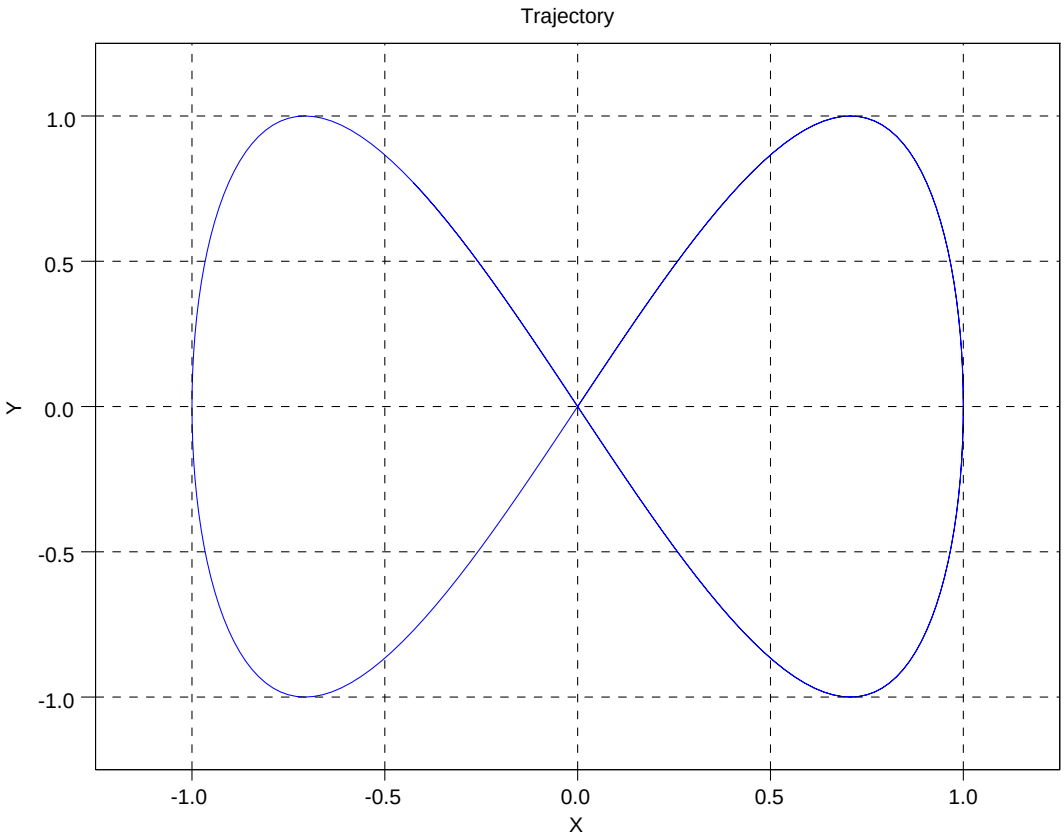

Рис. 1. Траектория движения точки

Массивы

Задать одномерный массив в Scilab можно следующим образом:

$$name = X0 : dX : XN,$$

где $name$ – имя переменной, в которую будет записан сформированный массив; $X0$ – значение первого элемента массива; dX – шаг, с помощью которого формируется очередной элемент массива, т.е. значение второго элемента составит $X0 + dX$, третьего – $X0 + 2 \cdot dX$ и т.д. (по умолчанию $dX = 1$); последний элемент принимает максимально возможное значение (при указанном способе формирования элементов массива), не превосходящее XN.

Обратиться к элементу массива можно, указав его имя и порядковый номер элемента в круглых скобках (нумерация элементов начинается с единицы):

$$name(\text{индекс}).$$

Также для задания массива можно воспользоваться поэлементным вводом. Для определения вектора-строки следует ввести имя массива, а затем после знака присваивания в квадратных скобках через пробел или запятую перечислить элементы массива:

$$name = [X1,\ X2,\ ...,\ XN] \text{ или } name = [X1 \quad X2 \quad ... \quad XN].$$

Элементы вектора-столбца вводятся через точку с запятой:

$$name = [X11;\ X12;\ ...,\ X1N].$$

Ввод элементов матрицы также осуществляется в квадратных скобках, при этом элементы строки отделяются друг от друга пробелом или запятой, а строки разделяются между собой точкой с запятой:

$$name = [X11,\ X12,\ ...,\ X1N;\ X21,\ X22,\ ...,\ X2N;\ ...,\\ XM1,\ XM2,\ ...,\ XMN;].$$

Обратиться к элементу матрицы можно, указав после имени матрицы, в круглых скобках через запятую, номер строки и номер столбца, на пересечении которых расположен элемент:

$$name(\text{индекс1, индекс2}).$$

Важную роль при работе с матрицами играет знак двоеточия: «:». Указывая его вместо индекса при обращении к массиву, можно получать доступ к группам его элементов:

Листинг 2: Примеры использования операции «:»

```
//Пусть задана матрица A
A = [1, 2; 3, 4]

disp( A, "A = ")

//Выделить из матрицы A второй столбец
disp( A(:, 2 ) )

//Выделить из матрицы A первую строку
disp( A(1, : ) )
```

Результат работы программы:

```
A =

  1.    2.
  3.    4.

  2.
  4.

  1.    2.
```

Для работы с матрицами и векторами в Scilab предусмотрены следующие операции:

+ – сложение;

- – вычитание;

* – матричное умножение и умножение на число;

’ – транспонирование;

$\hat{}$ – возведение в степень;

\ – левое деление; ($A\backslash B \Rightarrow A^{-1}\,B$);

/ – правое деление; ($B/A \Rightarrow B\,A^{-1}$).

Полезными также являются следующие специальные функции Scilab:

– ones(m, n) – создаёт матрицу единиц из m строк и n столбцов.

– zeros(m, n) – создаёт нулевую матрицу из m строк и n столбцов.

– eye(m, n) – создаёт единичную матрицу из m строк и n столбцов.

– rand(n1, n2, ..., nm [, p]) – создаёт многомерную матрицу случайных чисел размерности $n1 \times n2 \times \cdots \times nm$. Необязательный параметр p – символьная переменная, с помощью которой можно задать тип распределения случайной величины("uniform" – равномерное, "normal" – гауссовское); результат функции rand() – случайный скаляр. (Также можно использовать функцию grand, позволяющую получить случайные числа, имеющие бета-распределение; биномиальное, пуассоновское распределение; распредение «хи-квадрат» и др.)

– sort(X) – выполняет упорядочивание массива X по убыванию; если X – матрица, то сортировка выполняется по столбцам.

– size(V [, fl]) – определяет размер массива V; если V – двумерный массив, то size(V, 1) или size(V, "r") определяют число строк матрицы V, а size(V, 2) или size(V, "c") определяют число столбцов.

– max(V [, fl]) – если параметр fl отсутствует, то функция возвращает наибольший элемент в массиве V; если fl=1 или fl="r", то функция вернёт строку максимальных элементов столбцов матрицы V; если fl=2 или fl="c", то функция вернёт столбец максимальных элементов строк матрицы V.

– min(V [,fl]) – возвращает наименьший элемент в массиве V, работает аналогично функции max(V [,fl]).

– norm(V [, fl]) – возвращает норму вектора (матрицы) V. Тип нормы определяется параметром fl.

– det(V) – возвращает определитель квадратной матрицы V.

– inv(V) – возвращает матрицу, обратную к квадратной матрице V.

– linsolve(A, b) – решает систему линейных уравнений $A \cdot x - b = 0$.

Задача 2:

При сборке моста пришлось поднимать часть мостовой фермы ABC тремя канатами, расположенными, как указано на рис. 2. Вес этой части фермы $P = 42$ кН, центр тяжести находится в точке D. Расстояния соответственно равны: $AD = 4$ м., $DB = 2$ м., $BF = 1$ м. При этом углы наклона канатов равны $\alpha = 60°$, $\beta = 45°$. Найти натяжения канатов, если прямая AC горизонтальна.

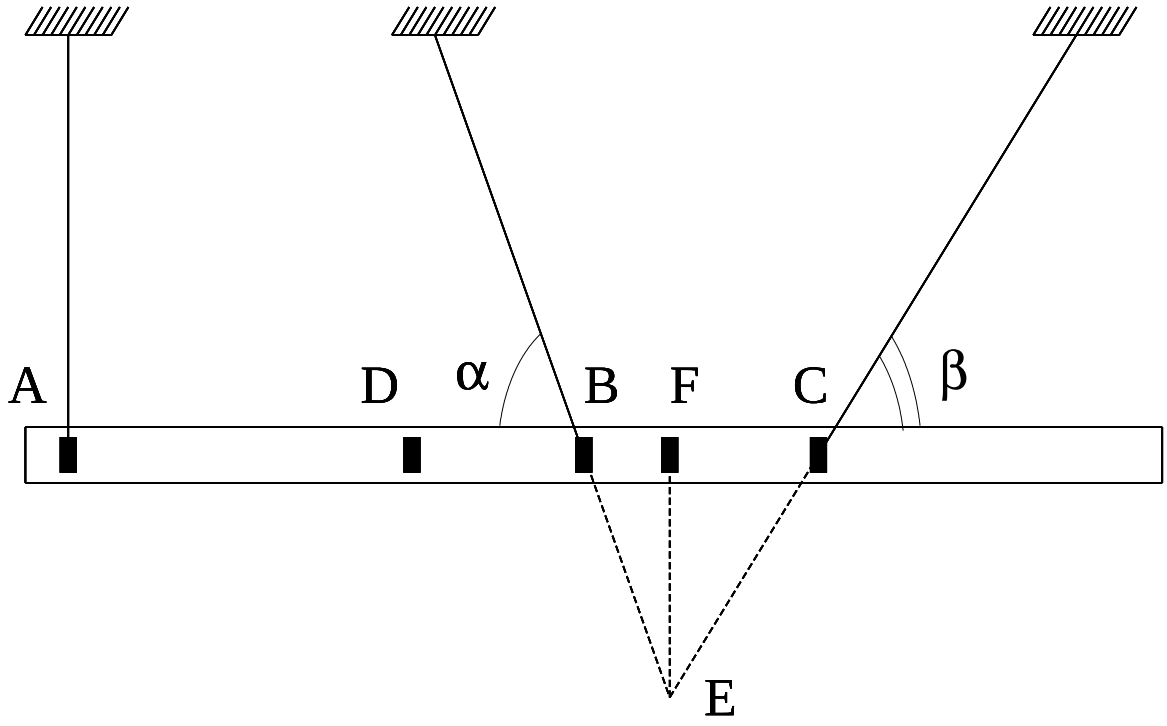

Рис. 2. Часть мостовой фермы ABC

Решение:

Введем правую систему координат Axy, ось абсцисс которой параллельна AC, а ось ординат направлена вертикально вверх. На рис. 3 показаны система координат и силы, приложенные к части фермы ABC.

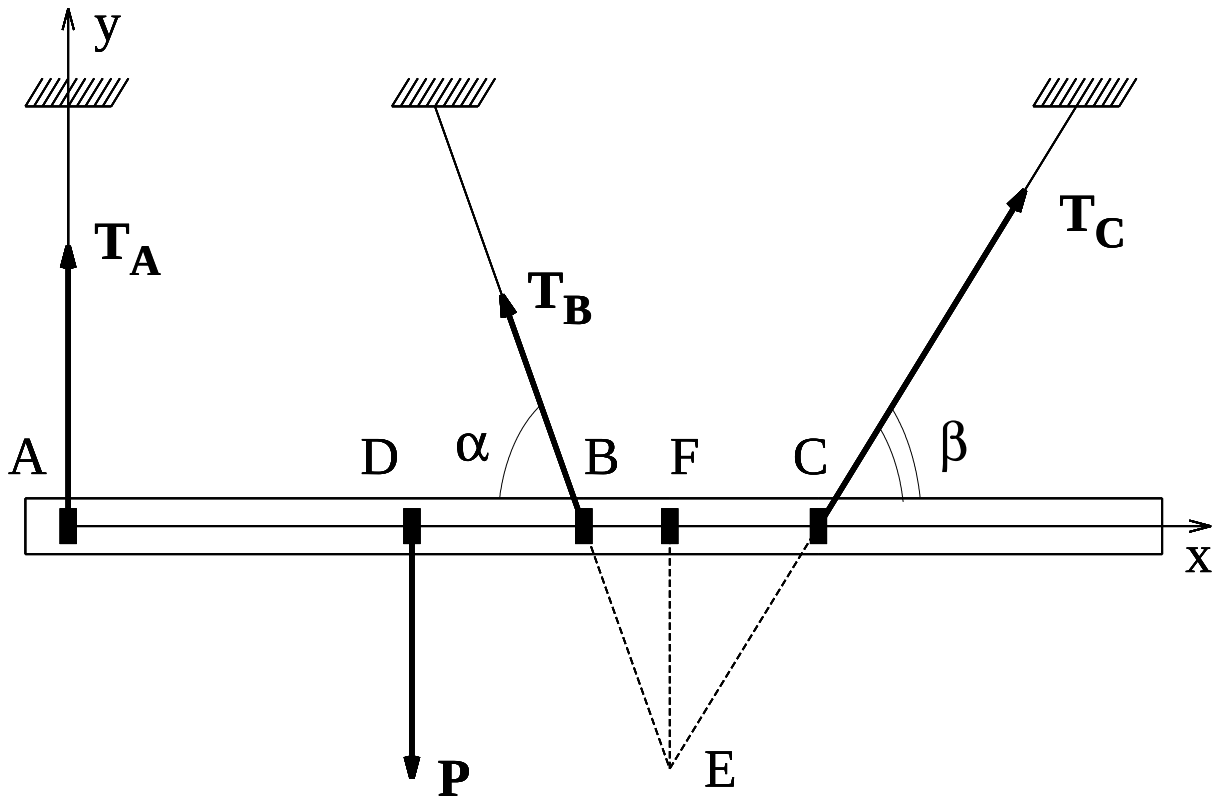

Рис. 3. Силы, приложенные к части фермы ABC

Приравняем нулю суммы проекций всех сил на оси Ax, Ay и сумму моментов всех сил относительно точки A. Получим следующую систему трёх линейных алгебраических уравнений:

$$\begin{aligned} Ax: &\ -T_B\cos\alpha + T_c\cos\beta = 0,\\ Ay: &\ T_A + T_B\sin\alpha + T_c\sin\beta - P = 0,\\ \Sigma mom_A: &\ T_B\cdot AB\sin\alpha + T_C\cdot AF\sin\beta - P\cdot AD = 0. \end{aligned}$$

Здесь $AB = AD + DB + BF + FC$, $AB = AD + DB$. Из рассмотрения $\triangle BEF$ и $\triangle CEF$ следует, что $BF\tan\alpha = EF = CF\tan\beta$.

Таким образом, необходимо решить систему линейных алгебраических уравнений вида $Ax = B$, где

$$A = \begin{pmatrix} 0 & -\cos\alpha & \cos\beta \\ 1 & \sin\alpha & \sin\beta \\ 0 & AB\sin\alpha & AF\sin\beta \end{pmatrix};\ \ x = \begin{pmatrix} T_A \\ T_B \\ T_C \end{pmatrix};\ \ B = \begin{pmatrix} 0 \\ P \\ P\cdot AD \end{pmatrix}.$$

Листинг 3: Решение системы линейных алгебраических уравнений

```
P = 42 // Вес части фермы в кН
// Углы наклона канатов в рад.
Alpha = %pi / 3;
Beta = %pi / 4;
// Расстояния в м.
AD = 4;
DB = 2;
BF = 1;
CF = BF * tan(Alpha) / tan(Beta);
AB = AD + DB;
AF = AD + DB + BF + CF;
//Матрица коэффициентов
A = [ [0, -cos(Alpha), cos(Beta)];
[1, sin(Alpha), sin(Beta)];
[0,  AB * sin(Alpha), AF * sin(Beta) ]];

disp( A, "A =" );

B = [0; P; P * AD]; // Вектор свободных членов

disp( B, "B =" );

// Система A * x = B
x = inv(A) * B // Вектор неизвестных

//или так
// x = linsolve(A, -B);

disp( x, "x =" );
disp( A * x - B, "Ошибка = "  )  // Проверка
```

Результат работы программы:

```
A =

   0.  - 0.5          0.7071068
   1.    0.8660254    0.7071068
   0.    5.1961524    6.1744923

B =

   0.
   42.
```

```
    168.

x =

    18.
    17.569219
    12.423314

Ошибка =

10^(-14) *

    0.1776357
    0.
    0.
```

Итак, $T_A = 18$ кН, $T_B = 17.569219$ кН, $T_C = 12.423314$ кН,

Дифференциальные уравнения

Для решения обыкновенных дифференциальных уравнений (ОДУ) и систем, записанных в нормальной форме Коши, в Scilab предусмотрена функция

$$y = ode([type,]\, y0,\, t0,\, t,\, f)$$

для которой обязательными входными параметрами являются: $y0$ – вектор начальных условий; $t0$ – начальная точка отрезка интегрирования; t – вектор, содержащий координаты узлов сетки, в которых происходит поиск решения; f – внешняя функция, определяющая правую часть уравнения или системы уравнений; y – вектор решений.

Таким образом, для того, чтобы решить ОДУ вида

$$\frac{dy}{dt} = f(t,\, y), \;\; y(t_0) = y_0;$$

необходимо вызвать функцию $y = ode(y0,\, t0,\, t, f)$.

С помощью необязательного параметра *type* можно выбрать метод решения ОДУ (если этот параметр не задать, то метод будет выбран автоматически.) Параметр *type* может принимать следующие значения:

"adams" – метод прогноза-коррекции Адамса;

"rk" – метод Рунге-Кутта четвёртого порядка точности;

"stiff" – указывают при решении жёстких задач и др.

Задача 3:

Груз массы $m = 24.5$ кг. висит на пружине жёсткости $c = 392$ Н/м. Определить движение груза, если на него начинает действовать сила $F(t) = 39.2\cos 6t$ Н.

Решение:

Введём координатную. ось Ox, направленную вертикально вниз. Точка O совпадает с начальным положением груза. По второму закону Ньютона

дифференциальные уравнения движения груза имеют вид:

$$\begin{aligned} &m\ddot{x} = -cx + F(t), \\ &x(0) = 0, \\ &\dot{x}(0) = 0. \end{aligned} \tag{1}$$

Запишем (1) в нормальной форме Коши:

$$\begin{aligned} &\dot{x_1} = x_2, \\ &\dot{x_2} = (-cx_1 + F(t))/m, \\ &x_1(0) = 0, \\ &x_2(0) = 0. \end{aligned}$$

Здесь $x_1 = x$, а $x_2 = \dot{x}$.

Листинг 4: Интегрирование системы ОДУ

```
// Сила, действующая на груз, в Н
function  f = F( t )
  f = 39.2 * cos( 6 * t );
endfunction
m = 24.5;  // Масса груза в кг.
c = 392;  // Жёсткость пружины в Н/м
// Правая часть системы ОДУ
function dy = dy( t, y )
  dy = zeros( 2, 1  );
  dy( 1 ) = y( 2 );
  dy( 2 ) = ( - c * y( 1 ) + F( t ) ) / m;
endfunction

// Начальные условия
y0 = [0; 0];
t0 = 0;
// Отрезок интегрирования
t = 0:0.01:%pi;

y = ode( "rk", y0, t0, t, dy )

subplot(2, 1, 1)
plot2d( t, y(1,:), style = color("red"), axesflag=5 );
xgrid(1);
xtitle( 'x(t)', boxed = 0 );
subplot(2, 1, 2)
plot2d( t, y(2,:), style = color("green"), axesflag=5 );
```

```
xgrid(1);
xtitle( 'dx(t) / dt', boxed = 0 );
```

Результат работы программы:

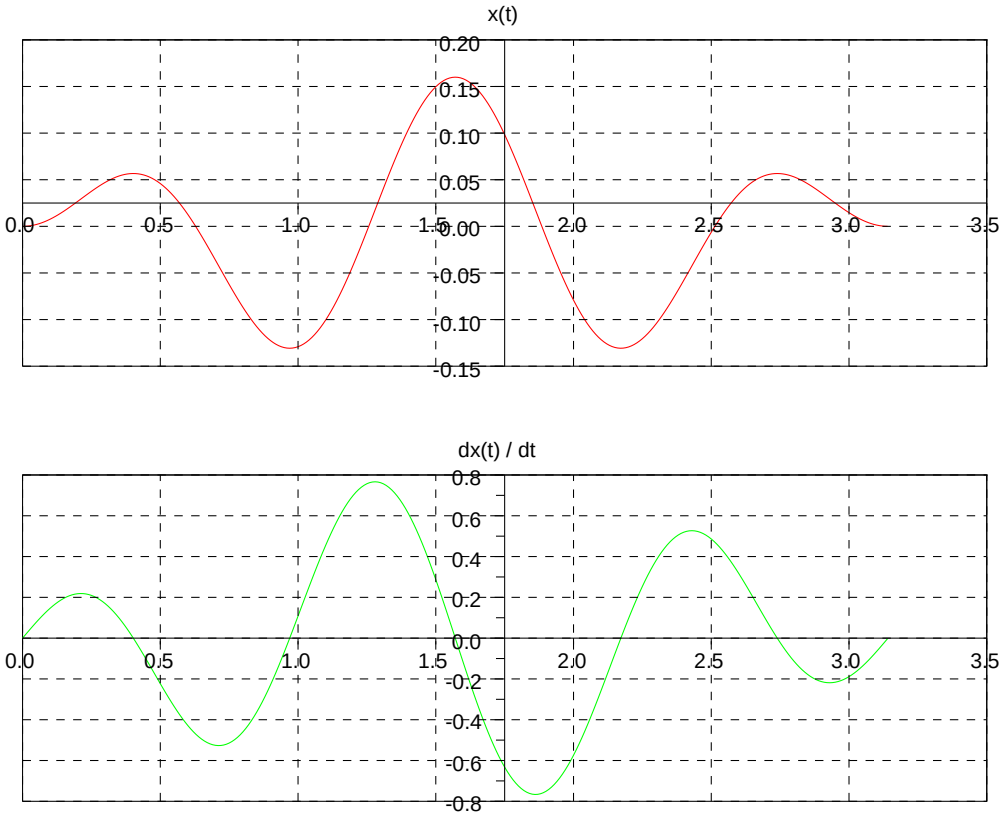

Рис. 4. Положение и скорость груза

Программирование

Рассмотрим основные операторы языка программирования Scilab.

Функции ввода-вывода

Для организации простейшего ввода данных в Scilab можно воспользоваться функциями

$$x = input(\text{"}title\text{"})$$

или

$$x = x\_dialog(\text{"}title\text{"}value\text{"}).$$

Функция *input* выводит в командной строке Scilab подсказку title и ждёт, пока пользователь введёт значение, которое в качестве результата возвращается в переменную x. Функция x_dialog выводит на экран диалоговое окно с именем *title*, после чего пользователь может щёлкнуть OK, и тогда строка *value* вернётся в качестве результата в переменную x; либо ввести новое значение вместо *value*, которое и вернётся в качестве результата в переменную x.

Функция *input* преобразовывает введённое значение к числовому типу данных, а функция x_dialog возвращает строку, которую можно преобразовать в число с помощью функции *evstr* :

$$x = evstr(x\_dialog(\text{"}title\text{"}value\text{"})).$$

Для вывода информации можно использовать функцию *disp* :

$$disp(x).$$

Здесь x – имя переменной или заключённый в кавычки текст.

Оператор присваивания

Оператор присваивания имеет следующую структуру:

$$a = b.$$

В результате выполнения присваивания переменной a присваивается значение выражения b.

Условный оператор

Одним из основных операторов является условный оператор if. Существует обычная и расширенная формы оператора if в Scilab. Обычная форма имеет вид

```
if  условие then
операторы1
else
операторы2
end;
```

Здесь условие – логическое выражение; операторы1 и операторы2 – операторы языка Scilab или встроенные функции. Для построения логических выражений могут быть использованы операторы: &, and (логическое «и»); |, or (логическое «или»); not (логическое отрицание); < (меньше); > (больше); == (равно); <> (не равно); >= (больше или равно); <= (меньше или равно).

Расширенная форма оператора if имеет вид:

```
if  условие then
операторы1
elseif условие2
операторы2
elseif условие3
операторы3
...
elseif условиеn
операторыn
else
операторы
end;
```

В этом случае оператор if работает так: если условие1 истинно, то выполняются операторы1, иначе проверяется условие2, если оно истинно, то выполняются операторы2, иначе проверяется условие3 и т.д. Если все условия по веткам elseif ложны, то выполняются операторы по ветке else.

Оператор while

Оператор цикла while имеет вид:

```
while  условие
операторы
end;
```

Операторы будут выполняться пока истинно условие.

Оператор for

Оператор цикла for имеет вид:

```
for  x=x0:dx:xn
операторы
end;
```

Выполнение цикла начинается с присвоения параметру x стартового значения ($x = x0$). Затем следует проверка, не превосходит ли параметр заданное значение ($x > xn$). Если $x > xn$, то цикл считается завершённым и управление передаётся следующему за телом цикла оператору. Если $x \leq xn$, то выполняются операторы в цикле. Далее параметр цикла увеличивается на dx ($x = x + dx$). После этого снова производится проверка значения параметра цикла и алгоритм повторяется.

Задача 4:

Движение управляемой системы описывается системой уравнений

$$\frac{dx_1}{dt} = x_2, \quad \frac{dx_2}{dt} = u. \tag{2}$$

В начальный момент времени

$$\text{при } t = 0 \quad x_1 = 5, \quad x_2 = 0. \tag{3}$$

В конечный момент времени

$$\text{при } t = 4 \quad x_1 + 2x_2 = 0. \tag{4}$$

Качество процесса управления определяется функционалом

$$J = \int_0^4 u^2 \, dt. \tag{5}$$

Требуется найти оптимальное управление $u = u(t)$, которое переводит управляемую систему (2) из начального состояния (3) на многообразие (4)

за заданный промежуток времени и сообщает минимальное значение функционалу (5).

Решение:

Поставленную задачу будем решать с помощью принципа максимума Понтрягина. Для этого введём дополнительные переменные ψ_1, ψ_2, сопряжённые по отношению к фазовым переменным x_1, x_2.

Функция Гамильтона-Понтрягина имеет вид

$$H = -u^2 + \psi_1 x_2 + \psi_2 u.$$

Сопряжённые дифференциальные уравнения имеют вид:

$$\frac{d\psi_1}{dt} = -\frac{\partial H}{\partial x_1} = 0, \quad \frac{d\psi_2}{dt} = -\frac{\partial H}{\partial x_2} = -\psi_1. \tag{6}$$

Оптимальное управление u_{opt} находится из условия максимума функции Гамильтона-Понтрягина по переменной u. Так как на управленние не налагаются ограничения, то

$$\left.\frac{\partial H}{\partial u}\right|_{u=u_{opt}} = 0 \;\Rightarrow\; -2u_{opt} + \psi_2 = 0 \;\Rightarrow\; u_{opt} = \frac{\psi_2}{2}. \tag{7}$$

После подстановки (7) в (2) получим

$$\frac{dx_1}{dt} = x_2, \quad \frac{dx_2}{dt} = \frac{\psi_2}{2}. \tag{8}$$

Таким образом, задача оптимального управления прямолинейным движением точки сведена к решению двухточечной краевой задачи для системы дифференциальных уравнений (6), (8), с начальными условиями (3) при $t = 0$ и конечными условиями (4) при $t = 4$, которые необходимо дополнить условием

$$H(t,\, x_1,\, x_2,\, \psi_1,\, \psi_2,\, u_{opt}(t))|_{t=4} = 0.$$

Для решения полученной краевой задачи составим программу, являющуюся комбинацией метода Рунге-Кутта и модифицированного метода Ньютона. (Можно показать, что задача имеет два решения. Одним из них является вектор $(x_1,\, x_2,\, \psi_1, \psi_2) = (5,\, 0,\, 0.375, 0)$. Читателю предлагается самостоятельно найти второе решение задачи и выбрать из двух решений то, которое доставляет минимальное значение функционалу качества (5).)

Листинг 5: Решение краевой задачи

```
 // Правая часть системы ОДУ
function dy = dy(t, x)
    dy = zeros(4, 1);
    dy(1, 1) = x(2);
    dy(2, 1) = x(4) / 2;
    dy(3, 1) = 0;
    dy(4, 1) = -x(3);
endfunction

// Функция Гамильтона-Понтрягина
function H = Hamiltonian( x )
    H = x(3, 1) * x(2, 1) + 0.25 * (x(4, 1))^2;
endfunction

// Невязки
function N = Nev(t0, tk, h, x0 )
    time = t0:h:tk;
    res = ode("rk", x0, t0, time, dy);
    cnt = size(res, "c");
    N = zeros(2, 1);
    N(1) = res(1, cnt) + 2 * res(2, cnt);
    N(2) = Hamiltonian( res(:, cnt) );
endfunction

t0 = 0;
tk = 4;
h = 10^-3;
eps = 10^-9
delta = 10^-7
x = [5; 0; 3/8-10^-3; 0 + 10^-3 ];
disp(x, "x = ");
module = Nev(t0, tk, h, x)

// Номер итерации
iter = 0;

while norm( module, 2 ) > eps,
      J = zeros(2, 2);
      B = module;

      // Нахождение матрицы частных производных
```

```
        for i=1:2
            x_new = x;
            x_new(2+i) = x_new(2+i) + delta;
            time = t0:h:tk;
            module_new = Nev(t0, tk, h, x_new);

            for j = 1:2
              J(j, i) = ( module_new( j ) - module( j ) );
              J(j, i) = J(j, i) / delta;
            end
        end
      // Вектор поправок
      gamma = inv(J) * B;
      kappa = 1;
      norm_old = norm( module );
      while ( 1 )
          x_new = x;
          x_new(3) = x_new(3) - kappa * gamma(1);
          x_new(4) = x_new(4) - kappa * gamma(2);
          norm_new = norm( Nev(t0, tk, h, x_new ) );
          if ( norm_new < norm_old ) then
                    x = x_new;
                    break
          end;
          kappa = kappa / 2;
          disp( kappa, "kappa = " );
      end;

      module = Nev(t0, tk, h, x);
      iter = iter + 1;
      disp(iter, "iter = ");
      disp( module, "module = " );
      disp(norm(module,2), "norm");
      disp( x, "x = " );
      disp( "---" );
end

time = t0:h/10:tk;
res = ode("rk", x, t0, time, dy);
// Значение функционала
disp( inttrap(time, res(4,:)^2 / 4), "int = " );
arg = 2:h:6;
```

```
plot2d( res(1,:), res(2,:), axesflag=5 );
plot2d( arg, -0.5*arg, axesflag=5 );
disp("done");
```

Результат работы программы:

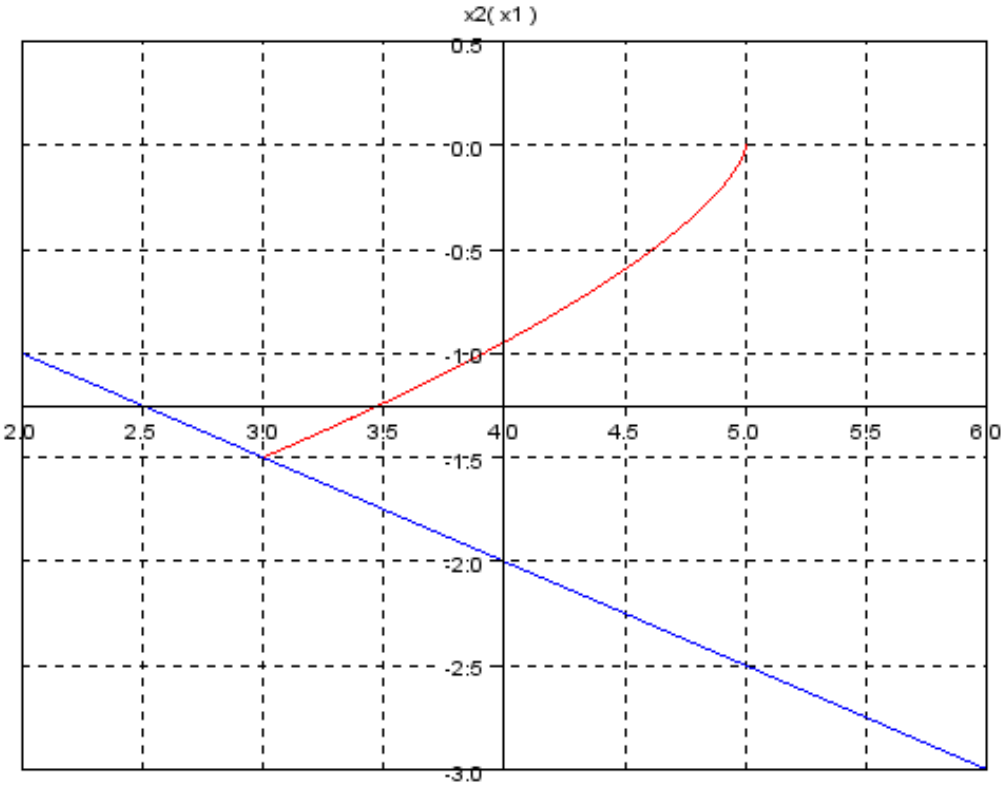

Рис. 5. Траектория движения (красный); конечное многообразие (синий)

Номер итерации	Невязка	Номер итерации	Невязка
0	0.0213333	3	3.899^{-9}
1	6.247^{-8}	4	9.814^{-10}
2	1.563^{-8}		

Значение функционала = 0.7500563.

Создание графических приложений

Основным объектом при работе с визуальными приложениями в среде Scilab является графическое окно. Для создания графического окна служит функция figure:

```
F = figure("Свойство1", "Значение1", ..., "СвойствоN",
  "ЗначениеN").
```

Здесь F – указатель на графическое окно; "Свойствоk" – название k – го параметра, "Значениеk" – значение k – го параметра. Пара «параметр – значение» может иметь следующий вид:

```
"position" -- [x, y, dx, dy];
"figure_name" -- "Name Of Figure";
"BackgroundColor" -- [1, 0, 0] и т.д.
```

В данном примере "position" – положение окна, x, y – координаты левого верхнего угла (ось абсцисс направлена слева направо, ось ординат – сверху вниз), dx, dy – ширина и высота окна; "figure_name"– имя окна; "BackgroundColor" – фоновый цвет в формате RGB.

После создания окна можно задать значение параметра с помощью функции

```
set("Свойство", "Значение").
```

В Scilab можно создавать (и удалять) те или иные компоненты управления (кнопки, метки и т.д.) на стадии выполнения программы. Для этого используется функция uicontrol, возвращающая указатель на сформированный компонент:

```
C = uicontrol(F, "Style", "Тип_компонента", "Свойство1",
  "Значение1", ..., "СвойствоN", "ЗначениеN").
```

Здесь C – указатель на создаваемый компонент;

F – указатель на объект, внутри которого будет создаваться компонент;

"Style" – указывает на тип создаваемого компонента;

"Тип_компонента" – определяет класс, к которому будет принадлежать создаваемый компонент (см. табл. 2).

Таблица 2 Типы визуальных компонентов

Значение свойства "Style"	Тип компонента
"pushbutton"	Командная кнопка
"text"	Метка
"radiobutton"	Радио-кнопка
"checkbox"	Переключатель
"edit"	Окно редактирования

Пары "Свойствок", "Значениек" определяют значения свойств компонента. В табл. 3 приведены некоторые свойства компонентов из табл. 2.

Важный параметр функции uicontrol – это событие Callback, которое генерируется при щелчке по кнопке, переключателю и т.д. Значением параметра Callback является строка с именем функции, вызываемой при щелчке по кнопке. Отметим, что при работе с компонентом «окно редактирования» событие Callback генерируется при нажатии клавиши Enter, а при переводе содержимого свойства "String" в число нужно воспользоваться описанной ранее функцией evstr.

У существующего компонента можно изменить те или иные свойства с помощью функции set:

```
set(C, "Свойство1", "Значение1", ..., "СвойствоN",
 "ЗначениеN").
```

Получить значение параметра компонента можно с помощью функции

```
get(C, "Свойство").
```

Задача 5:

Груз массы m кг., подвешенный к концу пружины, движется в жидкости. Коэффициент жёсткости пружины c Н/м. Сила сопротивления дви-

Таблица 3 Свойства визуальных компонентов

Свойство	Компоненты с этим свойством
"String" – Надпись	Все (для окна редактирования это строка ввода)
"HorizontalAlignment" – – Горизонтальное выравнивание ("left"("right"), "center" – по левому (правому) краю, по центру)	Все (для окна редактирования речь идёт о строке ввода)
"Position" – Положение (ось абсцисс идёт слева направо, ось ординат – снизу вверх)	Все
"Enable" / "Visible" – Доступность и видимость компонента ("on" – есть, "off" – нет)	Все
"Value" – Состояние переключателя (1 – включено, 0 – выключено)	Радио-кнопка, переключатель

жению пропорциональна первой степени скорости груза $\boldsymbol{R} = -\alpha \boldsymbol{v}$, $\alpha > 0$, $[\alpha] = \text{Н} \cdot \text{с} / \text{м}$. Составить визуальную программу, позволяющую построить графики движения груза для различных значений параметров задачи при условии, что в начальный момент груз был смещён из положения равновесия на x_0 м. и отпущен без начальной скорости.

Решение:

Введём координатную. ось Ox, направленную вертикально вниз. Точка O совпадает с положением равновесия груза. По второму закону Ньютона дифференциальные уравнения движения груза имеют вид:

$$\begin{aligned} &m\ddot{x} = -cx - \alpha\dot{x}, \\ &x(0) = x_0, \\ &\dot{x}(0) = 0. \end{aligned} \tag{9}$$

Запишем (9) в нормальной форме Коши:

$$\begin{aligned}
&\dot{x_1} = x_2,\\
&\dot{x_2} = -(cx_1 + \alpha x_2)/m,\\
&x_1(0) = x_0,\\
&x_2(0) = 0.
\end{aligned}$$

Здесь $x_1 = x$, а $x_2 = \dot{x}$.

Листинг 6: Визуальное программирование

```
// Главное окно
MainWindow = figure("figure_name",
"Oscillations + Resistance",
 "BackgroundColor", [1, 1, 1] )
set(MainWindow, "Position", [500, 300, 700, 500])

// Масса груза
mLabel = uicontrol("Style", "text",
 "Position", [50, 450, 50, 20], "String", "m = ",
 "BackgroundColor", [1, 1, 1])
mEdit = uicontrol("Style", "edit",
 "Position", [80, 450, 50, 20], "String", "1",
 "HorizontalAlignment", "right")

// Жёсткость пружины
cLabel = uicontrol("Style", "text",
 "Position", [150, 450, 50, 20], "String", "c = ",
"BackgroundColor", [1, 1, 1])
cEdit = uicontrol("Style", "edit",
 "Position", [180, 450, 50, 20], "String", "19.6",
  "HorizontalAlignment", "right")

// Альфа
alphaLabel = uicontrol("Style", "text",
"Position", [250, 450, 50, 20], "String", "$\alpha = $",
 "BackgroundColor", [1, 1, 1])
alphaEdit = uicontrol("Style", "edit",
"Position", [280, 450, 50, 20], "String", "3.5",
 "HorizontalAlignment", "right")

// Начальное положение груза
x0Label = uicontrol("Style", "text",
 "Position", [50, 400, 50, 20], "String", "x0 = ",
```

```
  "BackgroundColor", [1, 1, 1])
x0Edit = uicontrol("Style", "edit",
 "Position", [80, 400, 50, 20], "String", "0.01",
  "HorizontalAlignment", "right")

// Шаг интегрирования
hLabel = uicontrol("Style", "text",
 "Position", [150, 400, 50, 20], "String", "h = ",
  "BackgroundColor", [1, 1, 1])
hEdit = uicontrol("Style", "edit",
 "Position", [180, 400, 50, 20], "String", "0.01",
  "HorizontalAlignment", "right")

// Конец отрезка интегрирования
TLabel = uicontrol("Style", "text",
 "Position", [250, 400, 50, 20], "String", "T =",
  "BackgroundColor", [1, 1, 1])
TEdit = uicontrol("Style", "edit",
 "Position", [280, 400, 50, 20], "String", "10",
  "HorizontalAlignment", "right")

// Начать расчёт
Button = uicontrol("Style", "pushbutton",
"Position", [180, 350, 50, 20], "String", "Start ",
 "Callback", "Draw")

// Правая часть системы ОДУ
function dy = F( t, y )
    dy = zeros(2, 1)
    dy( 1 ) = y( 2 )
    dy( 2 ) = - ( c * y( 1 ) + alpha * y( 2 ) ) / m;
endfunction


function Draw()
// Получение параметров задачи
m = evstr( get( mEdit, "String" ));

c = evstr( get( cEdit, "String" ));

alpha = evstr( get( alphaEdit, "String" ));
```

Результат работы программы:

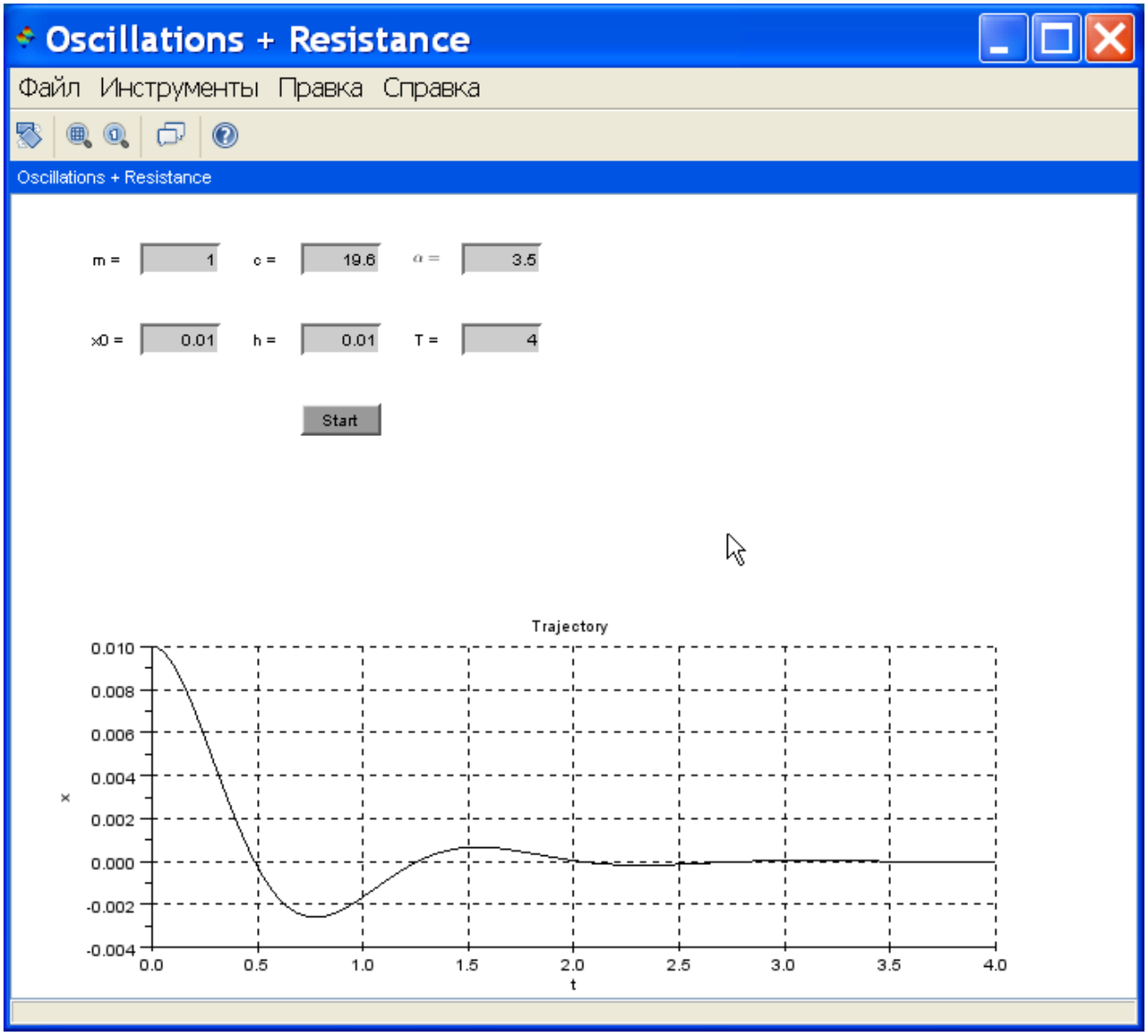

Рис. 6. Траектория движения груза

```
x0 = evstr( get( x0Edit, "String" ));

h = evstr( get( hEdit, "String" ));

T = evstr( get( TEdit, "String" ));

t = 0:h:T
y0 = [x0; 0]
// Интегрирование системы ОДУ
y = ode( "rk", y0 , 0, t, F )

// Удаление текущих графических осей
```

```
delete(gca())

subplot(2, 1, 2)

//Положение груза
plot2d( t, y(1, :)  )
xgrid()
xtitle("Trajectory", "t", "x"");
endfunction
```

Отметим, что для того, чтобы в надписи на метке появилась греческая буква, были использованы команды Latex'a.

Список использованных источников

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, 269 с.
2. Мещерский И.В. Задачи по теоретической механике: Учебное пособие. 45-е изд., стер. / Под ред. В.А. Пальмова, Д.Р. Меркина. СПб.: Лань, 2006, 448 с.
3. Сапунков Я.Г. Численное исследование систем автоматического управления: Учеб. пособие для стдентов мех-мат. фак. / Я.Г. Сапунков. Саратов: Изд-во Сарат. ун-та, 2001, 24 с.
4. Официальный сайт программы Scilab. URL: http://www.scilab.org.