Об опыте использования свободных математических программ на кафедре «Вычислительная математика и программирование» Донецкого национального технического университета

Алексеев Е.Р., Чеснокова О.В.

Аннотация

Рассмотрен опыт использования свободно распространяемых математических программ Scilab, Maxima, Octave. Описаны сновные возможности приложений. Приведена методическая литература, посвященная свободным математическим программам.

Abstract

Experience of using open source mathematical programs of Scilab, Maxima, Octave is considered. Describes the main features of applications. Methodological literature on the free mathematical software is shown.

Введение

Преподавание информационных дисциплин в техническом университете имеет ряд особенностей. Будущего инженера очень важно научить решать практические технические задачи с использованием современных компьютерных технологий. Поэтому курс информатики в техническом университете тесно связан с курсами высшей и вычислительной математики, а также с такими дисциплинами, как сопромат, теоретические основы электротехники и др. Важную роль в организации учебного процесса играет выбор программного обеспечения (ПО).

Для решения математических задач, обработки, моделирования и визуализации данных можно использовать, такие мощные проприетарные программные продуты, как MathCAD, MATLAB, Maple, Mathematica [1].

MATLAB представляет собой язык программирования высокого уровня для научно-технических вычислений. В MATLAB можно выполнять математические расчеты, разрабатывать алгоритмы, моделировать, проводить анализ данных, визуализировать различные процессы, создавать визуальные приложения различной сложности. MATLAB стал де-факто стандартом для проведения инженерных расчётов [1].

Mathcad - это интегрированная среда для выполнения различного рода вычислений. Matcad предоставляет широкие возможности по созданию и редактированию различных графиков. Удобен Mathcad и для документирования, так как в нем математические выражения отображаются с помощью принятых в математике обозначений. Благодаря простому интерфейсу пакет пользуется популярностью у студентов и школьников [1]. Однако, Mathcad не лишён недостатков: пакет не

справляется с достаточно сложными инженерными и математическими задачами, ограничен язык программирования, файлы Mathcad настолько закрыты, что полученные результаты сложно обрабатывать какой-либо другой программой.

Maple называют системой символьных вычислений или системой компьютерной математики аналитических преобразований. Пакет выполняет как численные, так и аналитические расчеты, оснащен мощной графикой и анимацией, может быть использован как научный редактор, имеет встроенный язык программирования [1].

Однако многие университеты не в состоянии купить эти проприетарные программы. Значительный интерес представляет свободно распространяемое кроссплатформенное ПО, которое бурно развиваются в последние годы и составляет реальную конкуренцию проприетарным программам. Среди свободных математических пакетов можно выделить систему компьютерной математики Scilab [2, 11, 15-17], предназначенную для выполнения инженерных и научных вычислений, математическую программу символьных и численных вычислений Maxima [4-10, 12], высокоуровневый язык программирования Octave [3, 13, 14] и другие.

В Донецком национальном техническом университете на кафедре «Вычислительная математика и программирования» авторы используют свободные математические программы, как в учебном процессе, так и в научных исследованиях.

1. Использование Scilab и Octave при решении инженерных и математических задач

Синтаксис и правила работы в Scilab и Octave достаточно понятны и схожи с MATLAB. Оба пакета обладают широким спектром встроенных элементарных математических функций и поддерживают работу с комплексными числами. Кроме того в пакетах можно создавать свои собственные функции.

В Scilab для создания функции можно воспользоваться оператором

```
deff('[имя_1,...,имя_N]=
имя_функции(переменная_1,...,переменная_M)',
'имя_1=выражение_1;...;имя_N=выражение_N');
```

Еще один способ создания функции пользователя в Scilab - это применение конструкции вида:

```
function[имя_1,...имя_N]=
имя_функции(переменная_1,...переменная_M)
тело функции
endfunction
```

В листинге 1 приведены примеры создания функций в Scilab.

```
//Функция вычисления площади треугольника по формуле Герона
```

```
-->deff('S=G(a,b,c)','p=(a+b+c)/2;S=sqrt((p-a)*(p-b)*(p-c))');
//Вычисление площади треугольника со сторонами 2,3,3.
-->G(2,3,3)
 ans  =  1.4142136
//Функция для решения кубического уравнения
function [x1,x2,x3]=cub(a,b,c,d)
r=b/a;s=c/a;t=d/a;
p=(3*s-r^2)/3;q=2*r^3/27-r*s/3+t;
D=(p/3)^3+(q/2)^2;
u=(-q/2+sqrt(D))^(1/3);
v=(-q/2-sqrt(D))^(1/3);
y1=u+v;
y2=-(u+v)/2+(u-v)/2*%i*sqrt(3);
y3=-(u+v)/2-(u-v)/2*%i*sqrt(3);
x1=y1-r/3;
x2=y2-r/3;
x3=y3-r/3;
endfunction
//Вызов функции и вывод результатов ее работы:
-->[x1,x2,x3]=cub(3,-20,-3,4)
 x3  =    0.3880206
 x2  =  - 0.5064407
 x1  =    6.7850868
```

Листинг 1

Функции пользователя в Octave также создаются при помощи конструкции **function...endfunction**. Функция, описанная в листинге 1, с успехом отработает и в Octave (листинг 2).

```
>>>%Вызов функции и вывод результатов ее работы:
>>> [x1,x2,x3]=cub(3,-2,-1,-4)
x1 =  1.4905
x2 = -0.41191 + 0.85141i
x3 = -0.41191 - 0.85141i
```

Листинг 2

В описании и обработке множественных типов данных в Scilab и Octave практически нет разницы.

Для работы с матрицами и векторами в пакетах предусмотрены следующие операции: «+» (сложение), «-» (вычитание), «'» (транспонирование), «*» (матричное умножение, умножение на число), « ^ » (возведение в степень),«/»(левое деление), «\» (правое деление), «.*» (поэлементное умножение матриц), « .^ » (поэлементное возведение в степень), «./» (поэлементное левое деление), «.\» (поэле-

ментное правое деление) [2, 3].

Кроме того, если к некоторому заданному вектору или матрице применить математическую функцию, то результатом будет новый вектор или матрица той же размерности, но элементы будут преобразованы в соответствии с заданной функцией. Примеры применения матричных операций показаны в листинге 3.

```
>>>%Действия над матрицами
>>> A=[-3 2 0;0 1 2;5 3 1];
B=[0 -2 1;3 -1 1;0 1 1];(2*A+1/3*B')^2-A*B^(-1)
ans =
    32.667   -20.667    20.667
    47.333    26.889    15.667
   -40.333    75.333    31.778
>>>%Решение СЛАУ Ax=B
>>> A=[1 2;1 1]; b=[7;6];  x=A\b
x =
    5
    1
--> //Решение матричных уравнений A*X=B и X*A=B
-->A=[3 2;4 3]; B=[-1 7;3 5]; X=A\B
 X  =
  - 9.      11.
    13.   - 13.
-->X=B/A
 X  =
  - 31.     23.
 11.      9.
--> //Поэлементные операции с векторами
-->a=[2 4 6]; b=[1 3 5]; 0.5*(a./b).^2+sin(a.\b)-sqrt(a.*b)
 ans  =      1.065212  - 1.893574  - 4.0170487
```

Листинг 3

В Scilab и Octave предусмотрено огромное количество специальных функции, предназначенных для работы с матрицами и векторами [2, 3]. для К наиболее часто используемым функциям можно отнести функции определения матриц. Так, функции **eye** и **zeros** возвращают единичную и нулевую матрицы, соответственно. Функция **ones** формирует матрицу, состоящую из единиц. За формирование диагональной матрицы, заданного вида отвечает функция **diag**. Есть несколько функций, которые возвращают матрицу случайных чисел, распределенную по разным законам. Одна из них это функция **rand**, которая возвращает матрицу случайных чисел с элементами распределенными по равномерному закону. Функция **cat** объединяет все матрицы, заданные в виде аргументов, а **rot90** – осуществ-

ляет поворот матрицы на 90 градусов или на величину 90k, где k – целое число. Сформировать нижнюю или верхнюю треугольную матрицу можно с помощью функций **tril** и **triu**. С помощью функции **sort** можно выполнить упорядочивание элементов столбцов, строк или всей матрицы [2, 3].

Не менее часто используют функции определения различных числовых характеристик матрицы. Функция **size** определяет число строк и столбцов матрицы. Функции **prod**, **sum**, **min**, **max**, **mean** применяют для вычисления произведения, суммы, наименьшего, наибольшего и среднего значения элементов матрицы. В зависимости от синтаксиса эти функции могут вычислять соответствующие значения по строкам или столбцам матрицы [2, 3].

Особый интерес представляют функции, реализующие численные алгоритмы решения задач линейной алгебры. Например, вычислить определитель можно с помощью функции **det**. След матрицы определяет функция **trace**. Для вычисления различных норм матрицы существует функция **norm**, а число обусловленности вычисляет функция **cond**. Функция **inv** возвращает обратную матрицу. Кроме того, в пакетах определены функции, которые выполняют различные преобразования матриц. Преобразовать матрицу к треугольной форме по методу Гаусса можно с помощью функции **rref**. Функции **lu** и **qr** выполняет соответственно LU- и QR-разложения, а **svd** — сингулярное разложение матрицы. Собственные значения и собственные векторы матрицы в Scilab вычисляет функция **spec**, а в Octave — функция **eig**. Для решения систем линейных алгебраических уравнений в Scilab предусмотрена функция **linsolve** [2,3]. Некоторые задачи линейной алгебры, которые могут быть решены с помощью описанных выше функций представлены в листинге 4.

```
disp('Решение СЛАУ методом Гаусса');
disp('Введите матрицу системы:');
A=input('A=');
disp('Введите вектор свободных коэффициентов:');
b=input('b=');
disp('Расширенная матрица системы:');
C=rref([A b])
disp('Размерность матрицы C:');
n=size(C)
disp('Вектор решений СЛАУ Ax=b');
x=C(:,n(2))
disp('Проверка Ax–b');
A*x–b
disp('Исследование системы на совместность');
disp('Введите матрицу системы:');
A=input('A=');
```

```
disp('Введите вектор свободных коэффициентов:');
b=input('b=');
disp('Размерность системы:');
[n,m]=size(A)
disp('Ранг матрицы системы:');
r=rank(A)
disp('Ранг расширенной матрицы:');
R=rank([A b])
if r==R
        disp('Система совместна.');
        if r==m
        disp('Система имеет единственное решение.');
        disp('Решение системы методом обратной матрицы:');
                        x=inv(A)*b
                disp('Проверка Ax-b=0:');
                A*x-b
        else
        disp('Система имеет бесконечно много решений.');
        end;
else
         disp('Система не совместна');
end;
disp('Собственные значения и собственные векторы матрицы');
disp('Введите матрицу:');
A=input('A=');
[n,m]=size(A);
disp('Вектор собственных значений матрицы A:');
d=eig(A)
[L, D]=eig(A);
disp('L - Матрица собственных векторов:');
L
disp('D - Диагональная матрица собственных значений:');
D
disp('Проверка:');
for i=1:n
          (A-D(i,i)*eye(n))*L(:,i)
end;
```

Листинг 4

Размер обрабатываемой матрицы в Octave, в отличии от Scilab, ограничен только доступной физической памятью компьютера.

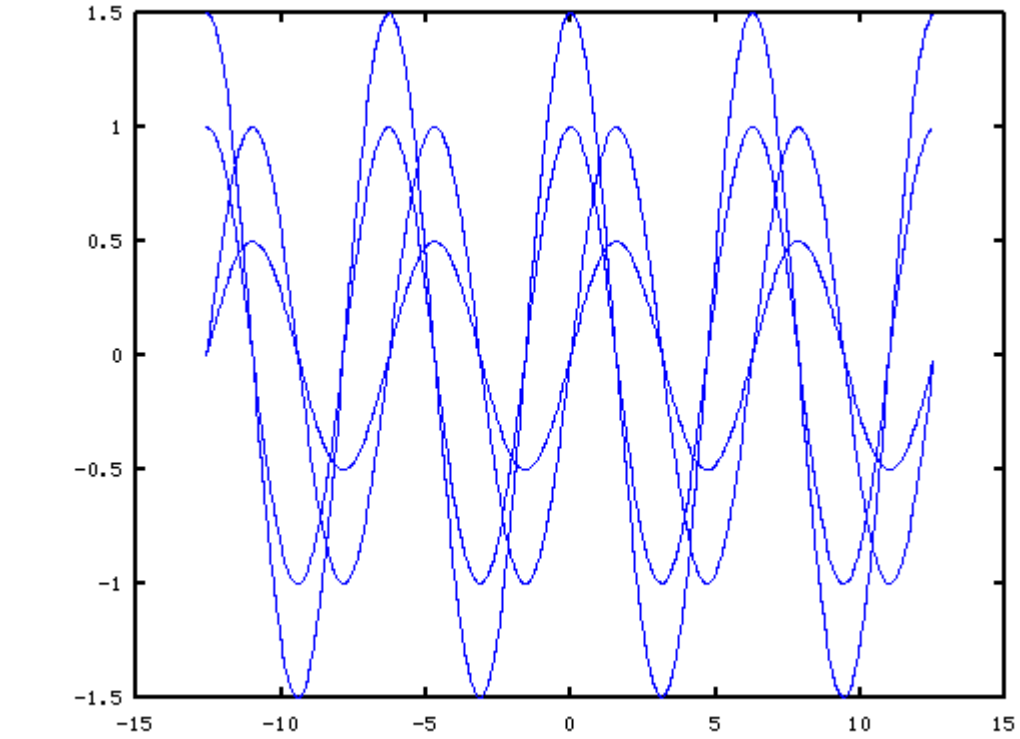

Рис. 1. Графики четырёх функций

Scilab и Octave обладают мощной графической базой. Оба пакета поддерживают двух-, трехмерную графику и анимацию [2, 3].

Двумерные графики наиболее часто строят в декартовой системе координат. Для этого Scilab и Octave оснащены функцией **plot**. У этой функции достаточно параметров [2, 3], чтобы управлять видом изображаемой линии и графического окна. Листинг 5 содержит примеры команд, которые можно использовать для построения графиков в декартовой системе координат.

```
%Построение графиков 4-х функций на заданном
% интервале (рис. 1)
x=-4*pi:0.1:4*pi;
v=sin(x); w=cos(x);   r=sin(x)/2; p=1.5*cos(x);
plot(x,v,x,w,x,r,x,p);
```

Листинг 5

Построить график в полярной системе координат можно в Scilab с помощью функции **polarplot**, а в Octave – **polar**. В листинге 6 приведен пример построения полярных графиков.

```
//Полярный график в Scilab (рис. 2)
fi=0:0.01:2*%pi; ro=3*cos(5*fi); ro1=3*cos(3*fi);
```

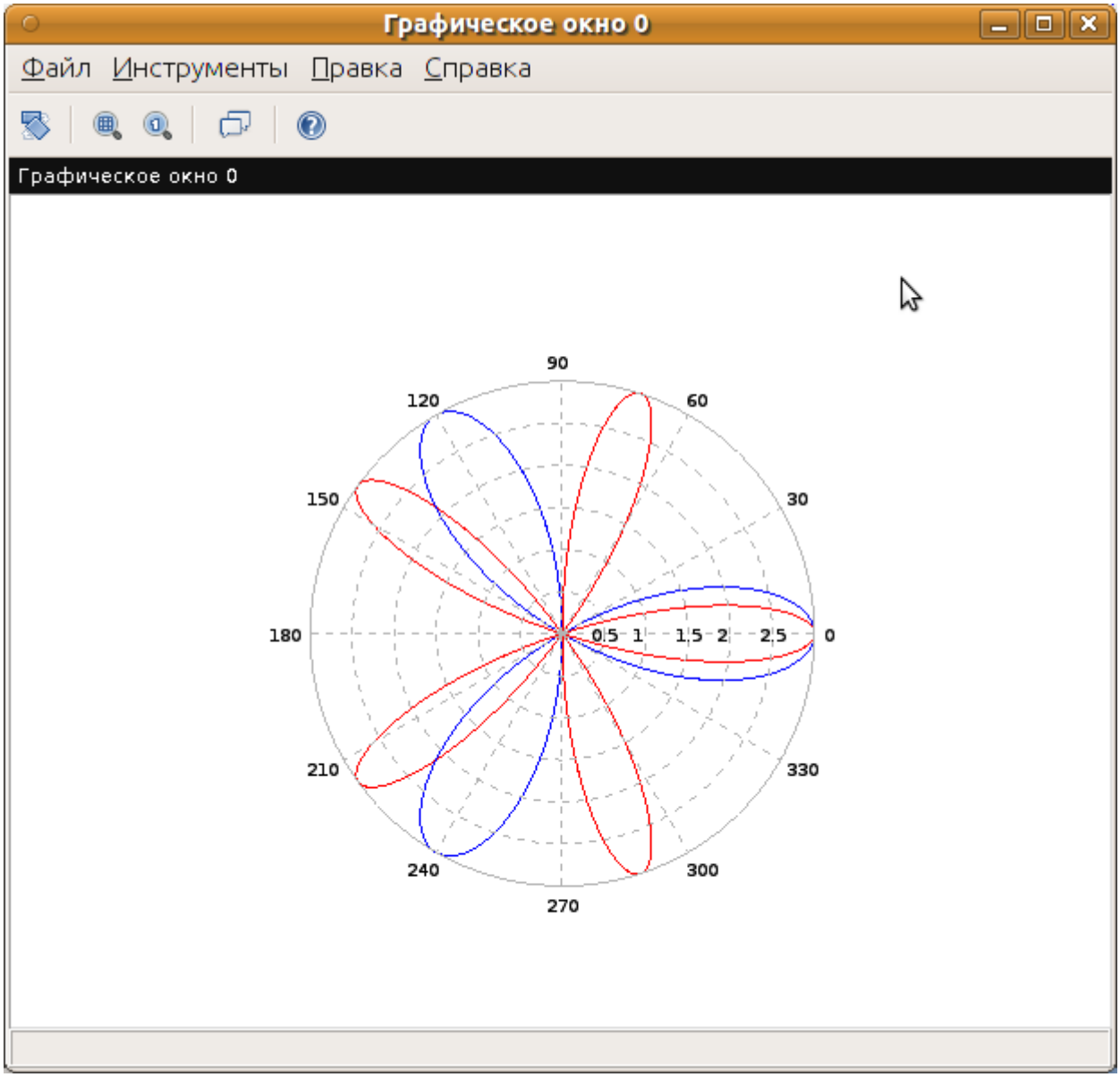

Рис. 2. Полярный график

```
polarplot(fi,ro,style=color("red"));
polarplot(fi,ro1,style=color("blue"));
```

Листинг 6

Трехмерные графики (поверхности) в Scilab и Octave строят функции **mesh** и **surf** [2, 3]. Отличие функций состоит в том, что **mesh** строит каркасный график, а **mesh** - каркасную поверхность, заливая ее каждую клетку цветом, который зависит от значения функции в узлах сетки (рис. 3). В Octave можно построить анимационный ролик движущейся точки. Увидеть движение точки вдоль кривой на плоскости можно с помощью функции **comet**. Например, для движения точки на плоскости вдоль синусоиды достаточно ввести команды

```
x=0:pi/30:6*pi;   y=sin(x);   comet(x,y);
```

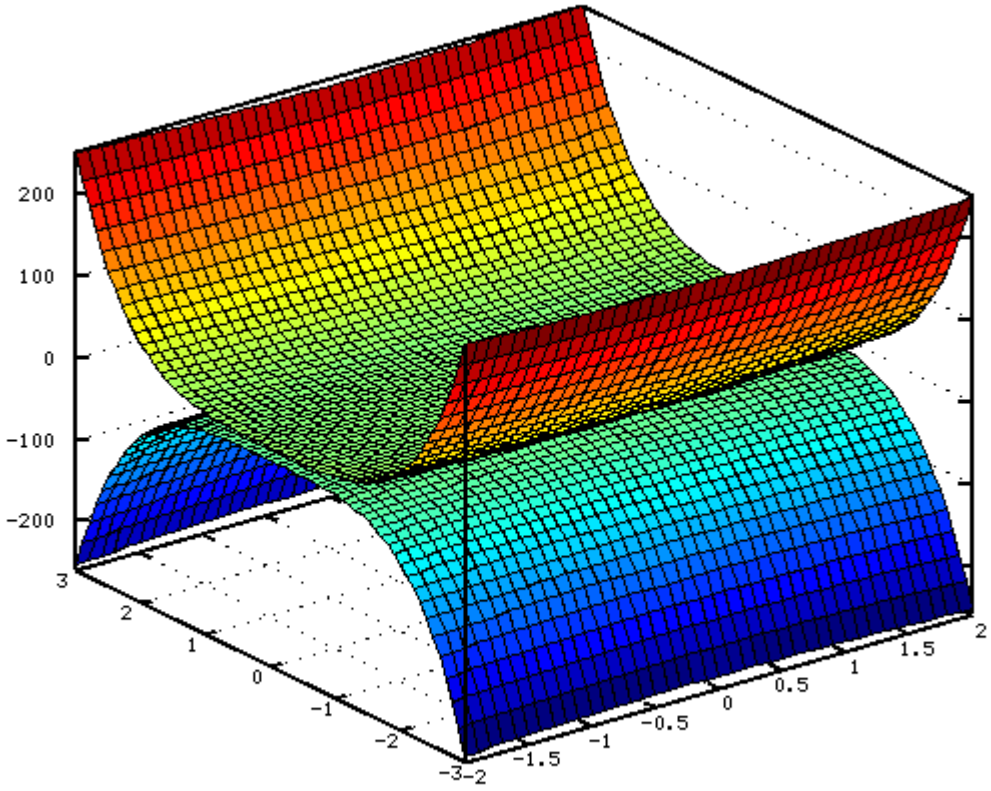

Рис. 3. Изображение двух поверхностей в одной системе координат с использованием функции surf

Начало движения точки по синусоиде представлено на рис. 4.

Благодаря мощным графическим средствам Scilab и Octave становятся более понятными и наглядными задачи линейной алгебры и аналитической геометрии. Так на рис.5 представлена геометрическая интерпретация действий над векторами, а рис. 6 содержит изображение плоскости заданной отрезками. Эти рисунки получены с помощью команд Octave. Кроме описанных выше графических функций здесь были использованы функции **line** (построение линии) и **patch** (заливка цветом заданной области) [3].

Scilab и Octave хорошо справляются с решением сложных нелинейных уравнений и систем. В Scilab для решения алгебраического уравнения нужно определить полином, задающий левую часть алгебраического уравнения с помощью функции **poly**, а затем найти его корни, используя функцию **roots**. В Octave корни полинома так же вычисляет функция **roots**, однако сам многочлен можно определить в виде вектора его коэффициентов [2,3]. Листинг 7 содержит примеры решения алгебраических уравнений.

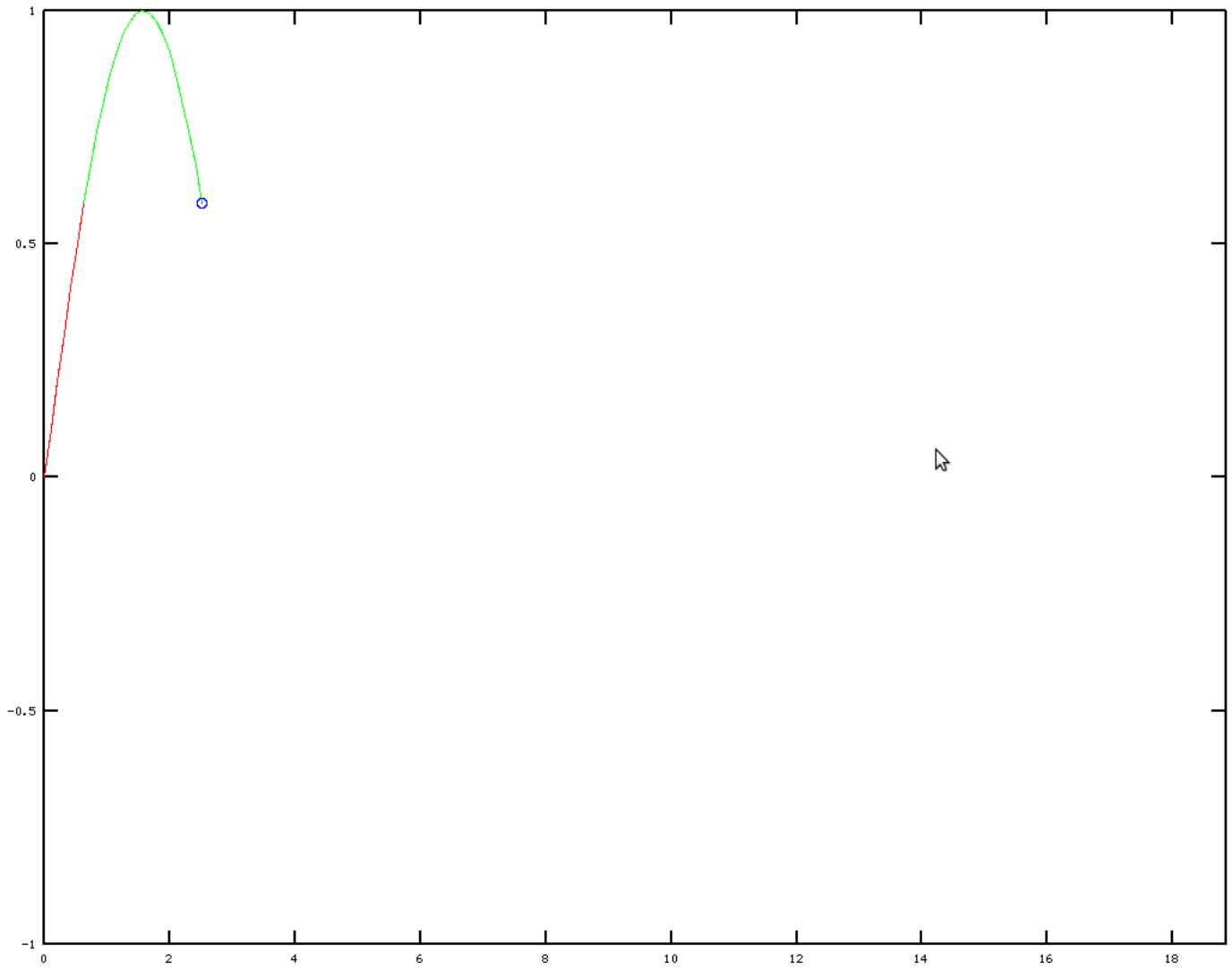

Рис. 4. Движение точки вдоль синусоиды

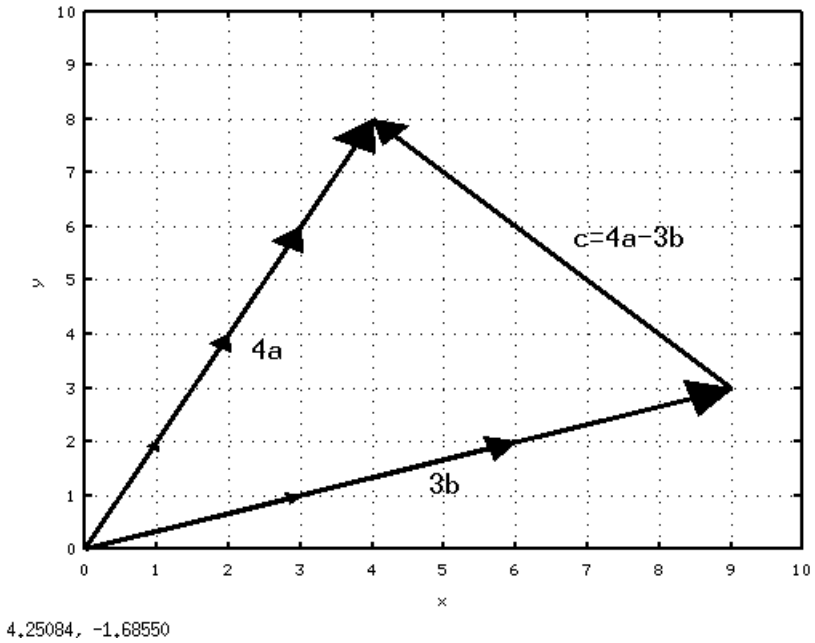

Рис. 5. Геометрическое представление действий над векторами

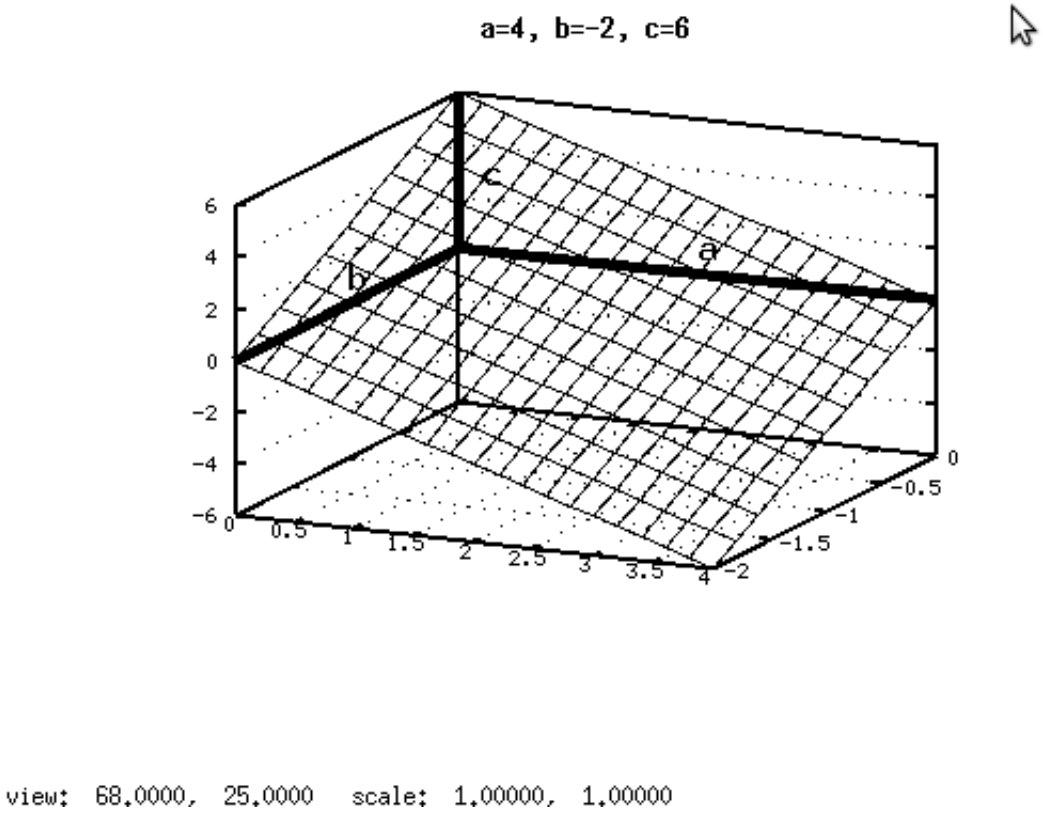

Рис. 6. Плоскость, заданная отрезками

```
-->//Вычисление корней полинома 2x^4+8x^2-8x+2=0
-->V=[-1 0 8 -8 2];
-->p=poly(V,'x','c')
 p  =
              2     3     4
      1 + 8x - 8x + 2x
-->X=roots(p)
 X  =
!   0.4588039  !
! - 0.3065630  !
!   1.5411961  !
!   2.306563   !
>>>%Вычисление корней полинома 2x^3-3x^2-12x-5=0
>>> p=[2 -3 -12 -5];
x=roots(p)
>>>x =
    3.44949
   -1.44949
   -0.50000
```

Листинг 7

Для решения трансцендентных уравнений и систем в Scilab применяют функцию **fsolve**, а в Octave функцию **fzero**. При обращении к этим функциям в качестве аргументов указывают начальное приближение корня, которое нужно найти заранее, например, графическим методом, и имя функции, описывающей левую часть нелинейного уравнения f(x)=0. Для функции **fzero** аргументом так же может быть и интервал изоляции корня нелинейного уравнения [2, 3]. Примеры решения нелинейных уравнений и систем с помощью этих функций показаны в листингах 8, 9.

```
//Решение уравнения f(x)=exp(x)/5-2(x-1)^2 в Scilab
deff('[y]=f(x)','y=exp(x)/5-2*(x-1)^2')
fsolve([0;2;5],f)
 ans  =          !   0.5778406  !
                 !   1.7638701  !
                 !  5.1476865  !
```

Листинг 8

```
%Решение системы уравнений {cos(x)+2y=2;(x^2)/3-(y^2)/3=1}
%в Octave
function [y]=fun(x)
                 y(1)=cos(x(1))+2*x(2)-2;
                 y(2)=x(1)^2/3-x(2)^2/3-1;
```

```
end;
[X1_Y1]=fsolve('fun', [-3 -1])
[X2_Y2]=fsolve('fun', [1 3])
>>>X1_Y1 =
   -2.1499    1.2736
>>>X2_Y2 =
    2.1499    1.2736
```

Листинг 9

В функциях интегрирования в Scilab и Octave реализованы различные численные алгоритмы. Так, в Scilab [2] численное интегрирование по методу трапеций реализовано с помощью функции **inttrap**. Вычислительный алгоритм квадратурных формул Ньютона-Котеса представлен в Scilab функцией **integrate**. В Octave метод трапеций представлен функцией **trapz**. Частный случай квадратурных формул, метод Симпсона, представлен в Octave функцией **quadv**. Интегрирование по квадратуре Гаусса в Octave [3] выполняет функция **quad**. Функции **quadl** и **quadgk** также выполняют интегрирование по квадратуре Гаусса. В этих функциях специальным образом подбирается шаг. В первом случае по методу Гаусса-Лобатто, во втором Гауса-Конрада. Листинг 10 содержит примеры численного интегрирования в Scilab, листинг 11 - в Octave.

```
-->//Интегрирование по методу трапеций в Scilab
-->h=0.1; x=5:h:13; y=sqrt(2*x-1);
-->inttrap(x,y)
 ans  = 32.666556
-->//Интегрирование по методу Ньютона–Котеса в Scilab
-->integrate('(2*x-1)^0.5','x',5,13)
 ans  = 32.666667
```

Листинг 10

```
>>>%Интегрирование по методу Симпсона в Octave
%Подынтегральная функция
function y=G(x)
y=(4-x^2).^(1/2);
end;
[F1,K1]=quadv('G',0,1)
>>>F1 =  1.91322288999134
K1 =  17
>>>%-------------------------------------------------------
>>>%Интегрирование по квадратуре Гаусса в Octave
function y=f(x)
y=(x.^2).*sqrt(3+sin(1./x));
end;
```

```
format long
[F,kod, K, err]=quad('f', 0, 1)
>>>F =  0.6543437191498 02
kod = 0
K =  1323
err =  1.37557012147481e-08
```

Листинг 11

Численное дифференцирование в Scilab представлено функцией **g=numdiff(f, x0)** [2]. Результат работы функции – матрица $g_{i,j} = \frac{df_i}{dx_j}$ (листинг 12).

Octave умеет отыскивать производные в символьном виде. Дифференцирование осуществляется с помощью функции **differentiate** [3] (листинг 13).

```
-->//Вычисление производной f(x)=(x+2)^3+5x в т.1 в Scilab.
-->function f=my(x), f=(x+2)^3+5*x, endfunction;
-->numdiff(my,1)
 ans  =  32.
-->x=1;3*(x+2)^2+5
 ans  =  32.
```

Листинг 12

```
>>>%Вычисление производной в символьном виде в Octave
symbols
x = sym ("x");
f=Tan(Log(x)^(1/3)); f1=differentiate(f,x)
>>>f1 =
(0.333)*(1+tan(log(x)^(0.333))^2)*x^(-1)*log(x)^(-0.666)
f=Log(Cos(x)); differentiate(f,x,2)
>>>ans = 16*(1+tan(x)^2)^2*tan(x)+8*(1+tan(x)^2)*tan(x)^3
```

Листинг 13

Для решения дифференциальных уравнений и систем в Sciab предусмотрена функция **ode** [2], для которой, обязательными входными параметрами являются вектор начальных условий, начальная точка интервала интегрирования, координаты узлов сетки, в которых происходит поиск решения и внешняя функция, определяющая правую часть уравнения или системы уравнений. Таким образом, для того чтобы решить обыкновенное дифференциальное уравнение вида $\frac{dy}{dt} = f(t, y), y(t_0) = y_0$, необходимо вызвать функцию **y=ode(y0,t0,t,f)**. Среди необязательных параметров функции есть параметр с помощью которого можно выбрать метод решения (Адамса, Рунге-Кутта с автоматическим или с фиксированным шагом) или тип решаемой задачи (жесткие уравнения или системы) [2]. Так же можно указать погрешность вычислений.

Octave предоставляет достаточное количество функций [3, 14] для решения

дифференциальных уравнений различного вида. Они подробно описаны в справке. Чаще всего используют следующие функции: **ode23** и **ode45** - решают обыкновенные нежёсткие дифференциальные уравнения (или системы) методом Рунге-Кутта 2-3-го и 4-5-го порядка точности; **ode5r** и **ode2r** - решают обыкновенные жёсткие дифференциальные уравнения (или системы) [3]. Пример решения системы дифференциальных уравнений в Scilab представлен на листинге 14 и рис. 7. Листинг 15 и рис.8 демонстрируют решение жёсткой системы дифференциальных уравнений в Octave. Сециализированные функции для решения дифференциальных уравнений находятся в пакетах расширений odebvp и odepkg [14].

```
//Решение задачи Коши x'=cos(xy), y'=sin(x+ty), x(0)=0, y(0)=0
// на интервале [0;10]
//Функция, описывающая систему дифференциальных уравнений
function dy=syst(t,y)
dy=zeros(2,1);
dy(1)=cos(y(1)*y(2));
dy(2)=sin(y(1)+y(2)*t);
endfunction
//Решение системы дифференциальных уравнений
x0=[0;0]; t0=0; t=0:1:10; y=ode(x0,t0,t,syst);
//Формирование графического решения (рис. 7)
plot(t,y)
```

Листинг 14

```
%Решение задачи Коши для жесткой системы
%дифференциальных уравнений dx/dt=Bx, при x(0)=(1,1,1,1)
%Функция правой части жёсткой системы  уравнений.
function dx=syst1(t,x)
B=[119.46 185.38 126.88 121.03;-10.395 -10.136 -3.636...
  8.577; -53.302 -85.932 -63.182 -54.211;-115.58 -181.75...
  -112.8 -199];
dx=B*x;
end
%Определение параметров управления ходом решения жёсткой
% системы дифференциальных уравнений.
%RelTol - относительная точность решения 1E-8,
%AbsTol - абсолютная точность решения 1E-8,
%InitialStep - начальное значение шага изменения
%независимой переменной 0.02,
%MaxStep - максимальное значение шага изменения
%независимой переменной 0.1.
par=odeset ("RelTol", 1e-8, "AbsTol", 1e-8,...
```

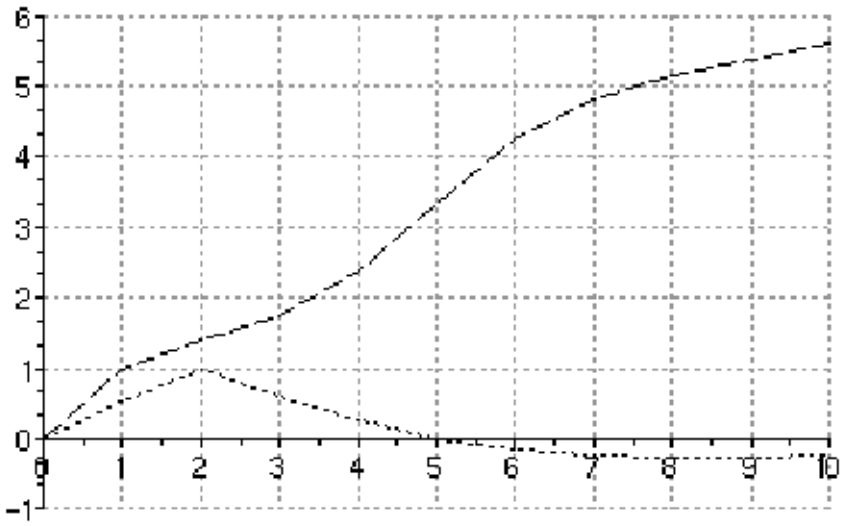

Рис. 7. Решение задачи задачи Коши
$x' = cos(xy), y' = sin(x + ty), x(0) = 0, y(0) = 0$ **на интервале [0;10]**

```
'InitialStep',0.02,'MaxStep',0.1);
%Решение жёсткой системы дифференциальных уравнений.
%Построение графика решения (рис. 8).
[A,B]=ode2r(@syst1,[0  5],[1;1;1;1]);
plot(A,B,'-k'); grid on;
```

Листинг 15

Задачи оптимизации встречаются во многих отраслях знаний. Алгоритмы их решения реализованы во многих математических пакетах. Не являются исключением Scilab и Octave. Одним из классов оптимизационных задач являются задачи поиска минимума функций. В Scilab минимум функции одной или нескольких переменных ищет функция **optim** [2]. Как известно функция Розеброка является тестовой для алгоритмов минимизации, ее минимум в точке (1, 1) равен 0. Листинг 16 содержит пример вычисления минимума функции Розенброка с помощью функции optim.

```
// Начальное приближение x0
x0=[-2;2]
//Функция Розенброка
function y=gg(x)
y=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
endfunction
//Формирование функции, возвращающей функцию Розенброка
// и ее градиент.
```

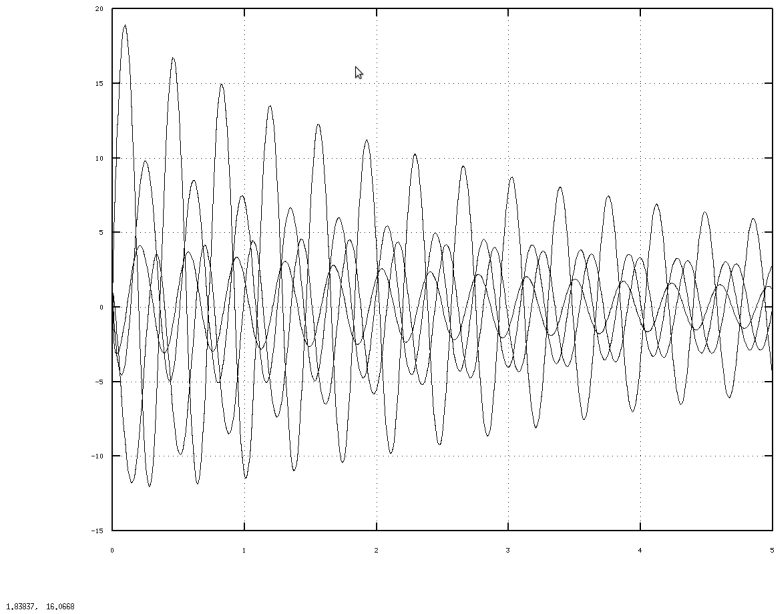

Рис. 8. Решение жесткой системы дифференциальных уравнений

```
function [f,g,ind]=cst(x,ind)
f=gg(x);
g=numdiff(gg,x);
endfunction
// Вызов функции optim
[f,xopt]=optim(cst,x0)
// Результат поиска минимума функции Розенброка
x0 =
          -2.
           2.
xopt =
           0.9999955
           0.9999910
f =        2.010D-11
```

Листинг 16

Функция **sqp** в Octave ищет минимум функции при заданных ограничениях [3]. Пример работы с этой функцией показан в листинге 17.

```
%Поиск максимума функции F=(x-3)^2-(y-4)^2
%при ограничениях: 3x+2y>=7, 10x-y<=8, -18x+4y<=12, x>=0, y>=0
function y=f1(x)
                    y=-f(-x);
endfunction
function y=f(x)
```

```
                y=(x(1)-3)^2+(x(2)-4)^2;
endfunction
function r = g (x)
                r=      [3*x(1)+2*x(2)-7;
                        -10*x(1)+x(2)+8;
                        18*x(1)-4*x(2)+12;
                        x(1);
                        x(2)];
endfunction
x0=[0;0];
[x, obj, info, iter]=sqp(x0,@f1,[],@g)
maximum=f(x)
%Результаты поиска максимума
>>>x =
2.0000
12.0000
obj = -281.00
info = 101
iter = 3
>>>maximum = 65.000
```

Листинг 17

Еще один класс задач оптимизации это задачи линейного программирования. Для решения этих задач в Scilab предназначена функция **linpro** [2]. Обязательными входными параметрами функции являются массив коэффициентов функции цели, матрица коэффициентов и вектор свободных коэффициентов системы ограничений. Кроме того можно задать нижнюю и (или) верхнюю границу ограничений переменных, указать тип ограничений (неравенства, равенства) и вектор начальных приблизительных значений неизвестных. Результатом работы функции будут массив неизвестных, минимальное значение функции и массив множителей Лагранжа [2]. Листинг 18 содержит пример использования этой функции.

```
//Поиск значений x1, x2, x3, x4,
//при которых функция L = -x2 - 2 x3 + x4
//достигает своего минимального значения и
//при этом выполняются ограничения:
//3 x1 - x2 <= 2,
//x2 - 2 x3 <= -1,
//4 x3 - x4 <= 3,
//5 x1 + x4 >= 6,
//x1>=0, x2>=0, x3>=0,   x4>=0.
c=[0;-1;-2;1];
```

```
A=[3 -1 0 0;0 1 -2 0; 0 0 4 -1; -5 0 0 -1];
b=[2;-1;3;-6]; ci=[0;0;0;0];
[x,kl,f]=linpro(p,A,b,ai,[])
//Результат поиска минимума
```

Полученные значения представлены в листинге 18.

```
f=
    2.
kl=
    0.
    0.
    0.
    0.
    0.0909091
    1.0909091
    0.5454545
    0.4545455
x=
    1.
    1.
    1.
    1.
```

Листинг 18

В Octave подобные задачи решает функция **qlpk** [3]. Это достаточно мощная и гибкая функция. В качестве ее аргументов выступают не только коэффициенты функции цели, матрица коэффициентов и вектор свободных коэффициентов системы ограничений, но и параметры, позволяющие задавать верхнюю и (или) нижнюю границы переменных, тип ограничений, тип переменной (вещественная или целочисленная), тип задачи оптимизации (минимум или максимум) и оптимизационный алгоритм [3]. Результатом работы функции является массив неизвестных, при которых функция цели достигает своего оптимального значения, оптимальное значение функции цели, параметр, определяющий решена задача оптимизации или нет. Кроме того, функция возвращает множители Лагранжа, время решения задачи, объём памяти, которая был использован при решении задачи [3]. Пример функции представлен в листинге 19.

```
%Поиск значений x1, x2, x3, при которых функция
%L = -5 + x1 - x2 - 3x3
%достигает своего минимального значения и
%при этом выполняются ограничения:
% x1+ x2 >= 2,
% x1 - x2 <= 0,
% x1 + x3 >=2,
```

```
% x1 + x2 - x3 <= 3,
% x1 >= 0, x2 >= 0, x3 >= 0,
% x1, x2, x3 - целые
c=[1;-1;-3]; a=[1 1 0; 1 -1 0;1 0 1;1 1 -1]; b=[2; 0;2; 3];
ctype="LULU"; vartype="III"; sense=-1;
[xmax,fmax,status]=glpk(c, a, b, [],[], ctype, vartype, sense)
%Результаты решения задачи целочисленного программирования:
>>>xmin =
                    2
                    2
                    1
fmin = -3
status = 171
```

Листинг 19

На практике часто встречаются задачи по обработке реальных количественных экспериментальных данных, полученных в результате всевозможных научных опытов и технических испытаний. Одним из способов решения таких задач является метод наименьших квадратов [1]. Он позволяет по экспериментальным данным подобрать такую аналитическую функцию, которая проходит настолько близко к экспериментальным точкам, насколько это возможно. Для реализации этой задачи в Scilab предусмотрена функция **datafit**. Для работы этой функции нужно указать имя внешней функции, представляющей собой разность между экспериментальными и теоретическими значениями зависимости, матрицу исходных данных и вектор начальных приблизительных значений коэффициентов искомой аналитической зависимости. В результате функция **datafit** выдаст вектор коэффициентов искомой аналитической зависимости и значение суммы квадратов отклонений измеренных значений от расчетных. Функция **datafit** с успехом подбирает коэффициенты многих аналитических зависимостей [2]. Пример использования функции представлен в листинге 20. График подобранной в Scilab зависимости представлен на рис. 9.

```
//Подбор параметров зависимость вида y = a1 x ^ a2 + a3.
function [zr]=F(c,z)
zr=z(2)-c(1)*z(1)^c(2)-c(3);
endfunction
x=[10.1,10.2,10.3,10.8,10.9,11,11.1,11.4,12.2,13.3,13.8,...
14,14.4,14.5,15,15.6,15.8,17,18.1,19];
y=[24,36,26,45,34,37,55,51,75,84,74,91,85,87,...
94,92,96,97,98,99];
z=[x;y];   c=[0;0;0];  [a,S]=datafit(F,z,c);
//График экспериментальных данных и
```

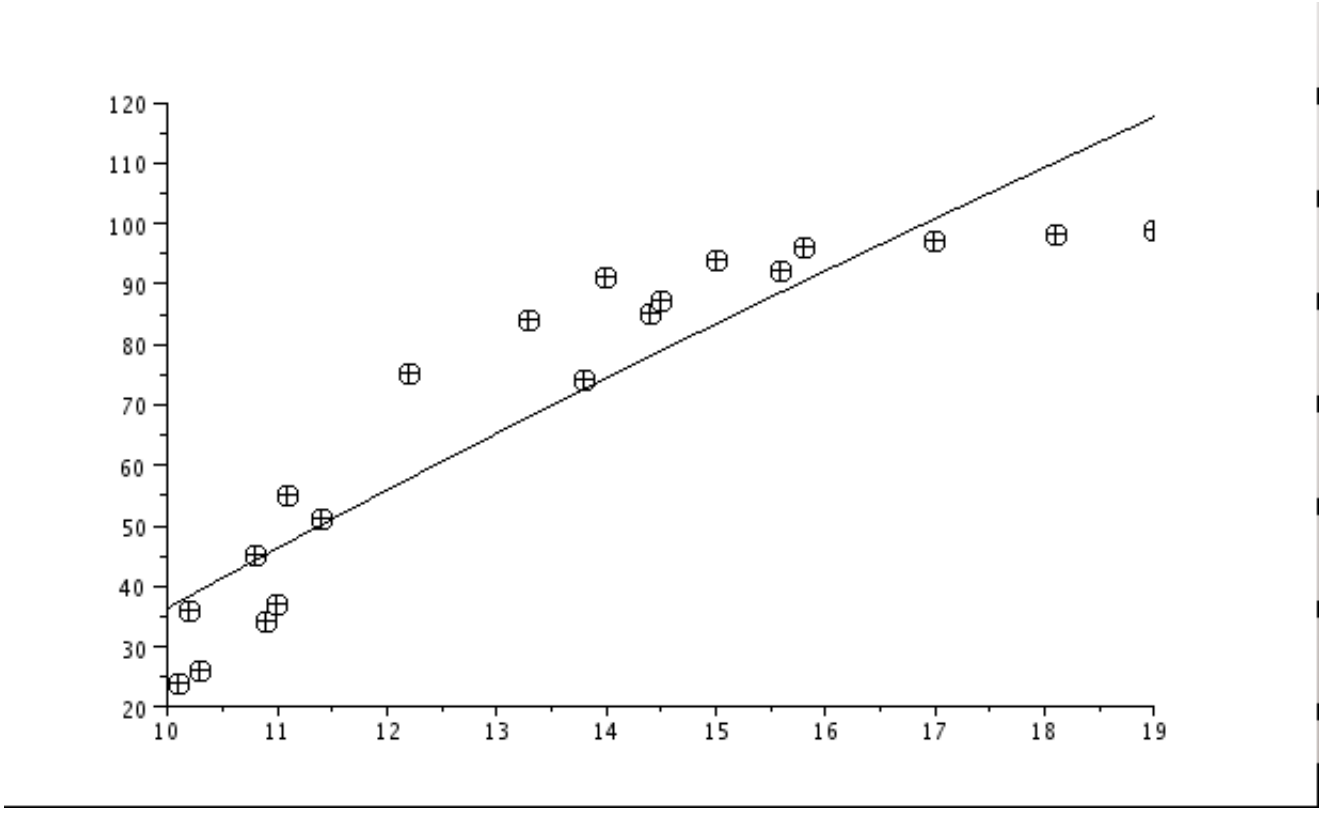

Рис. 9. Графическое решение задачи обработки эксперимента методом наименьших квадратов

```
//подобранной аналитической зависимости (рис. 9)
t=10:0.01:19; Yt=a(1)*t^a(2)+a(3); plot2d(x,y,-3); plot2d(t,Yt);
```

Листинг 20

В Octave для решения задач подбора аналитических зависимостей по экспериментальным данным можно использовать функции **polyfit** и **sqp** [3]. Функция **polyfit** возвращает коэффициенты полинома указанной степени, используя в качестве аргументов массивы абсцисс и ординат экспериментальных точек, а так же целое число, указывающее полином какой степени следует подобрать. Функцию **sqp** можно использовать для поиска минимума суммы квадратов отклонений измеренных значений от расчетных. В листинге 21 представлены примеры решения задачи обработки результатов эксперимента методом наименьших квадратов в Octave.

```
%Подбор параметров полинома первой степени (уравнение регресии)
%Экспериментальные данные
X=[0 4 10 15 21 29 36 51 68];
Y=[66.7 71.0 76.3 80.6 85.7 92.9 99.4 113.6 125.1];
%Вычисление коэффициентов полинома y=a1*x+a2
[a]=polyfit(X,Y,1)
```

```
%Результаты подбора коэффициентов полинома:
>>>a =    0.87064    67.50779

%Подбор параметров функции вида y=ax^b e^{cx}
function s=f_mnk(c)
        global x; global y;
        s=0;
        for i=1:length(x)
        s=s+(log(y(i))-c(1)-c(2)*log(x(i))-c(3)*x(i))^2;
        end
end
global x; global y;
%Задание начального значения коэффициентов.
%При неправильном определении начального значения
%экстремум функции может быть найден неправильно.
c=[2;1;3];
%Определение координат экспериментальных точек
x=[1 1.4 1.8 2.2 2.6 3 3.4 3.8 4.2 4.6 5 5.4 5.8];
y=[0.7 0.75 0.67 0.62 0.51 0.45 0.4 0.32 0.28 0.25 ...
0.22 0.16 0.1];
%Решение задачи оптимизации
c=sqp(c,@f_mnk)
%Результат решения задачи оптимизации.
>>>c =
    0.33503
    0.90183
   -0.69337
```

Листинг 21

Другой способ решения задач обработки эксперимента — интерполяция. Один из наиболее распространенных вариантов интерполяции это интерполяция сплайнами. В Scilab линейная интерполяция реализована функцией **interpln** [2], пример использования которой представлен на листинге 22.

```
x=[132 140 150 162 170 180 190 200 211 220 232 240 251];
y=[330 350 385 425 450 485 540 600 660 730 920 1020 1350];
z=[x;y]; t=132:5:252; ptd=interpln(z,t);
//График экспериментальных точек и интерполяционного
// полинома (рис. 10)
plot2d(x,y,-4); plot2d(t,ptd);
```

Листинг 22

Построение кубического сплайна в Scilab состоит из двух этапов. В начале

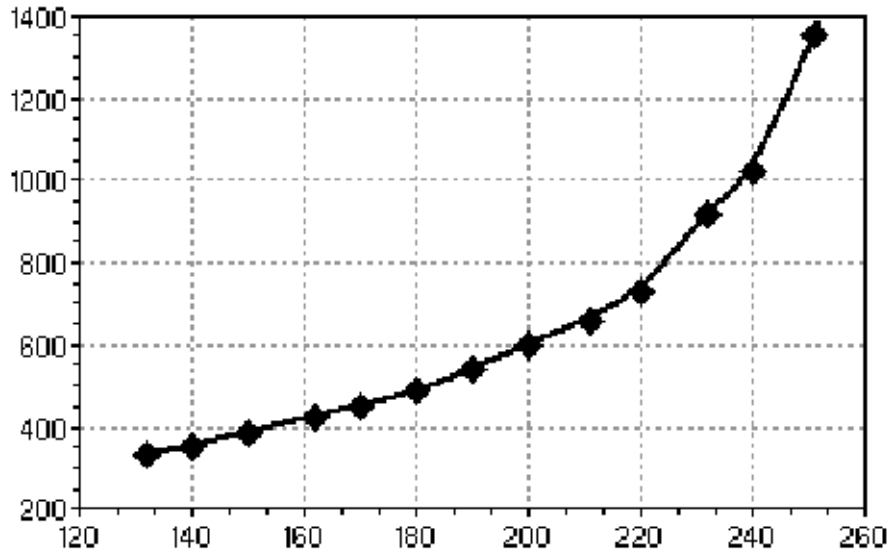

Рис. 10. Графическое представление экспериментальных значений и линейного сплайна

вычисляются коэффициенты сплайна с помощью функции **splin**, а затем рассчитываются значения интерполяционного полинома в точке с помощью функции **interp** [2] (листинг 23).

```
//Вычисление приближенного значения функции
//при заданном значении аргумента
//с помощью интерполяции кубическими сплайнами
//в точках x1 = 0,702, x2 = 0,512, x3 = 0,608.
x=[0.43 0.48 0.55 0.62 0.7 0.75];
y=[1.63597 1.73234 1.87686 2.03345 2.22846 2.35973];
koeff=splin(x,y);
X=[0.702 0.512 0.608];
//Значение функции в заданных точках
Y=interp(X,x,y,koeff)
//Результат вычислений:
 Y = 2.2335678    1.7969698    2.0057073
```

Листинг 23

Интерполяция сплайнами в Octave реализована функцией **interp1** [3]. Аргументами этой функции являются значения экспериментальных точек, точки, в которых необходимо вычислить значение с помощью сплайна и параметр, определяющий метод построения сплайна (линейная интерполяция или кубический сплайн). Пример применения функции **interp1** приведен в листинге 24.

```
%В результате опыта холостого хода определена зависимость.
```

```
%Построить график интерполяционной зависимости.
%Вычислить ожидаемое значение мощности
%при x=0.308, 0.312, 0.325.
%Экспериментальные точки
x=[0.298 0.303 0.31 0.317 0.323 0.33];
u=[3.25578 3.17639 3.1218 3.04819 2.98755 2.9195];
%Точки, в которых надо посчитать ожидаемое значение.
X1=[0.308 0.312 0.325];
%Расчёт значений в точках 0.308, 0.312, 0.325 с помощью
% кубического сплайна.
u1s=interp1(x,u,x1,'spline')
%Расчёт значений в точках 0.308, 0.312, 0.325 с помощью
% линейного сплайна.
u1l=interp1(x,u,x1,'linear')
%Графики линейного и кубического сплайнов (рис. 11).
Xi=0.298:0.002:0.33;
uxis=interp1(x,u,xi,'spline'); uxil=interp1(x,u,xi,'linear');
plot(x,u,'*b;experimentl;',xi,uxil,'-r;linear_spline;',...
xi,uxis,'-b;cubic_spline;',x1,u1s,...
'pr;points_(cubic_spline);',x1,u1l,...
'<b;points_(linear_spline);');
axis([0.29,0.34,2.8,3.3]);
grid on;
%Решение задачи
>>>u1s = 3.1370 3.1031 2.9685
>>>u1l = 3.1374 3.1008 2.9681
```

Листинг 24

Если сравнивать возможности Scilab и Octave. то следует отметить следущее. Scilab - это мощный язык программирования. Он позволяет не только запрограммировать алгоритм численных расчётов любой сложности, но и создать приложение с графическим интерфейсом. Scilab - мощная IDE, его легко освоить и использовать. что позволяет быстро освоить пакет. В состав Scilab система моделирования Xcos [16, 17] — аналог Simulink. Немаловажную роль играет достаточно большого количества литературы Scilab и Xcos [2, 11, 16, 17], в том числе и на русском языке [2, 11, 17]. Имеенно Scilab авторы рекомендуют использовать в учебном процессе. На базе этого пакета можно построить курс информатики для студентов инженерного и физико-математическогонаправления.

Octave по существу не имеет стабильной графической оболочки, что вызывает определённые сложности в освоении. Но эти недостатки компенсируется мощным языком программирования, большим количеством специальных функций и мно-

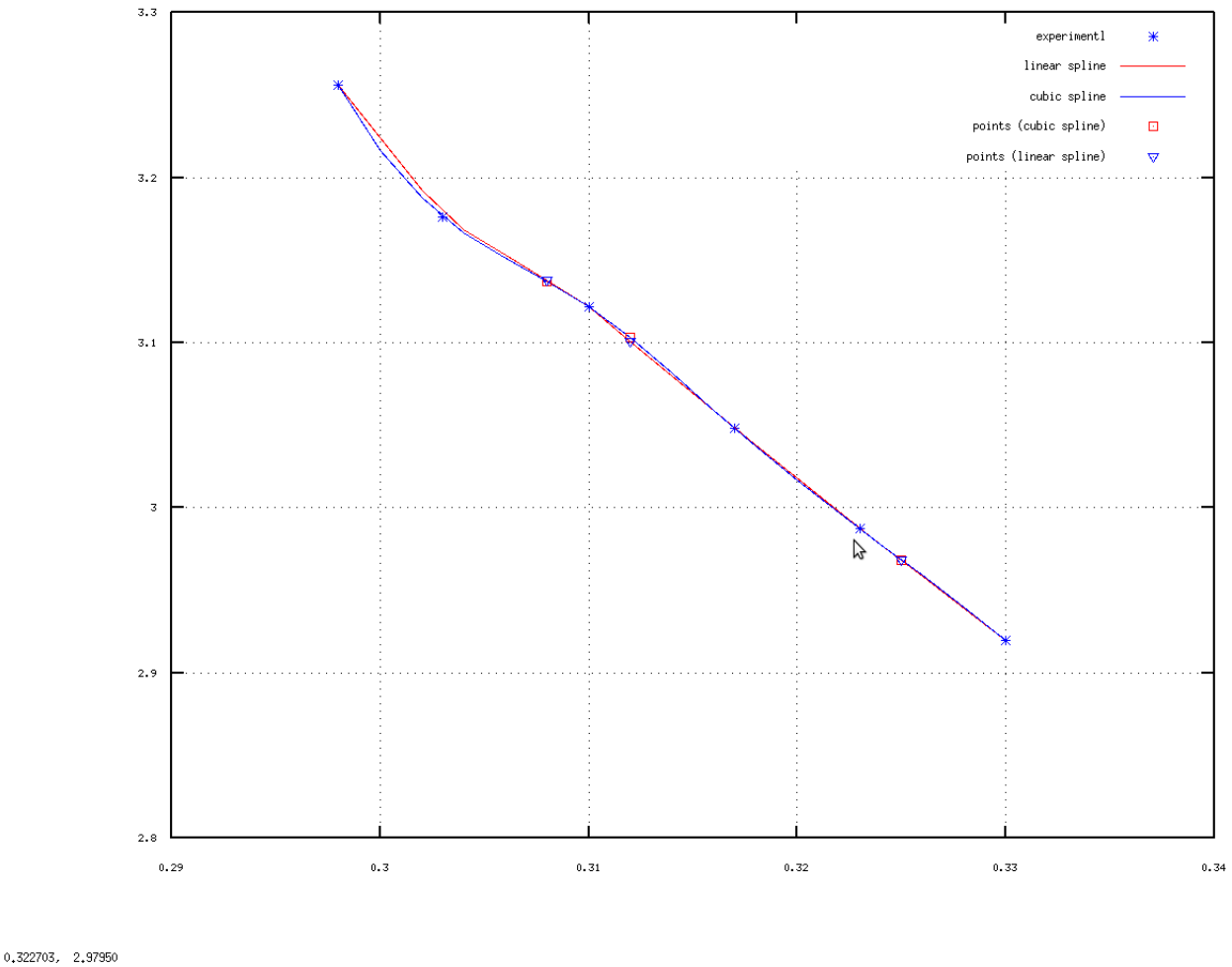

Рис. 11. Графическое представление линейного и кубического сплайнов

жеством пакетов расширений [14], которые позволяют решать задачи из разных отраслей знаний.

Если Scilab и GNU Octave предназначены для численных расчётов, то для проведения аналитических вычислений серьёзным конкурентом проприетарным пакетам Mathematica и Maple является свободная программа Maxima.

2. Об опыте использования программы Maxima в курсе «Высшая и прикладная математика»

В 2010/11 учебном году один из авторов вёл практические занятия по курсу «Высшая и прикладная математика» для студентов специальности «Экономика предприятий» с использованием свободных математических пакетов. Студенты изучали возможности Maxima для решения разных классов математических задач. С помощью этого пакета можно решать задачи линейной алгебры, аналитической геометрии, нелинейные уравнения и системы, обыкновенные дифференциальные уравнения и системы, вычислять интегралы и производные. Maxima позволяет проводить вычисления в численном и символьном виде. Графическая база пакетов позволяет иллюстрировать решение задачи.

Интерфейс программы достаточно прост. Пользователь вводит команды в специальный объект, называемый входной ячейкой. В ячейке может находиться одна или несколько команд. Команда завершается символом «$» или «;». Ввод команд в ячейку завершается комбинацией клавиш Shit+Enter. Если команда заканчивается «$», то результат на экран не выводится. Команда, заканчивающаяся «;», выводит результат на экран. Правила ввода чисел в Maxima такие же, как и в многих других подобных программ. Целая и дробная часть десятичных дробей разделяются символом "точка". Числитель и знаменатель обыкновенных дробей разделяются при помощи символа «/» (прямой слэш). Для перехода от обыкновенной дроби к десятичной служат функции **numer** и **float** [6, 12] (см. рис. 12).

Maxima можно использовать для построения различных графиков. С помощью функции **plot2d** возможно построить график f(x) на отрезке [a, b]. Функцию можно записать двумя способами: **plot2d (f(x), [x,a,b], опции)**, где x – имя независимой переменной; a, b – определяют интервал, на котором строится функция; **plot2d(f(x), [x,a,b], [y,c,d])**, где x – имя независимой переменной; a,b – определяют интервал, на котором строится функция, c,d – интервал изменения по оси ординат [12]. Последняя форма очень удобная для построения графиков разрывных функций. В листинге 25 представлен код Maxima для построения разрывной функции $f(x) = \frac{x^2-3}{\sqrt{3x^2-2}}$ (рис. 13).

```
(%i25) (1/2+1/3)/(1/13-1/17);
                              1105
(%o25)                        ----
                               24

(%i26) %,numer;
(%o26)                46.04166666666666

(%i28) float(1/3+1/17);
(%o28)                0.3921568627451

(%i29) 1/3-1/17;
                              14
(%o29)                        --
                              51

(%i30) %,float;
(%o30)                0.27450980392157
```

Рис. 12. Использование функций numer и float

```
(%i1)  f(x):=(x^2-3)/((3*x^2-2)^(1/2));
                                  2
                                 x  - 3
(%o1)                  f(x) := ----------
                                  2     1/2
                               (3 x  - 2)
(%i2)  plot2d(f,[x,-15,15],[y,-7,7]);
```

Листинг 25

Для построения трёхмерного графика используется функция **plot3d(f(x,y), [x,a,b], [y,c,d])**, где x, y – имена независимых переменных, a, b – интервал изменения переменной x, c, d – интервал изменения переменной y [6, 9, 12]. На рис. 14 представлен график функции $f(x,y) = (xy)^2 e^{-\frac{x^2+y^2}{2}}$.

Матрица вводится функцией matrix ([...],[...],[...],...), где с помощью [....] вводится каждая строка матрицы, элементы которой разделяются между собой запятыми; новая строка отделяется от предыдущей также запятой [12].

Над матрицами определены поэлементные операции: «+»(сложение), «-»(вычитание), «*»(умножение), «\» (деление), « ^ » (возведение в степень). Кроме того, определены матричные операции: умножение (.), « ^^-1» (вычисление обратной матрицы) [6, 12]. Для вычисления обратной матрицы также можно воспользоваться функцией **invert(a)**. Вычисление определителя матрицы реализуется при помощи функции **determinant(a)**. На рис. 15 приведен пример вычисления

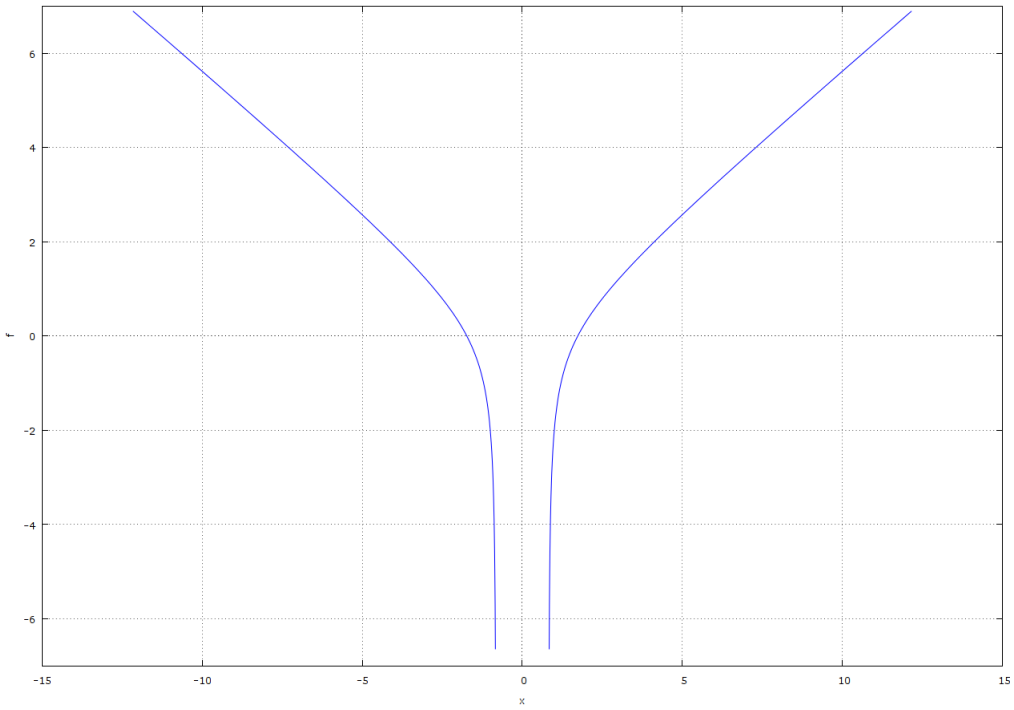

Рис. 13. График разрывной функции $f(x) = \frac{x^2-3}{\sqrt{3x^2-2}}$

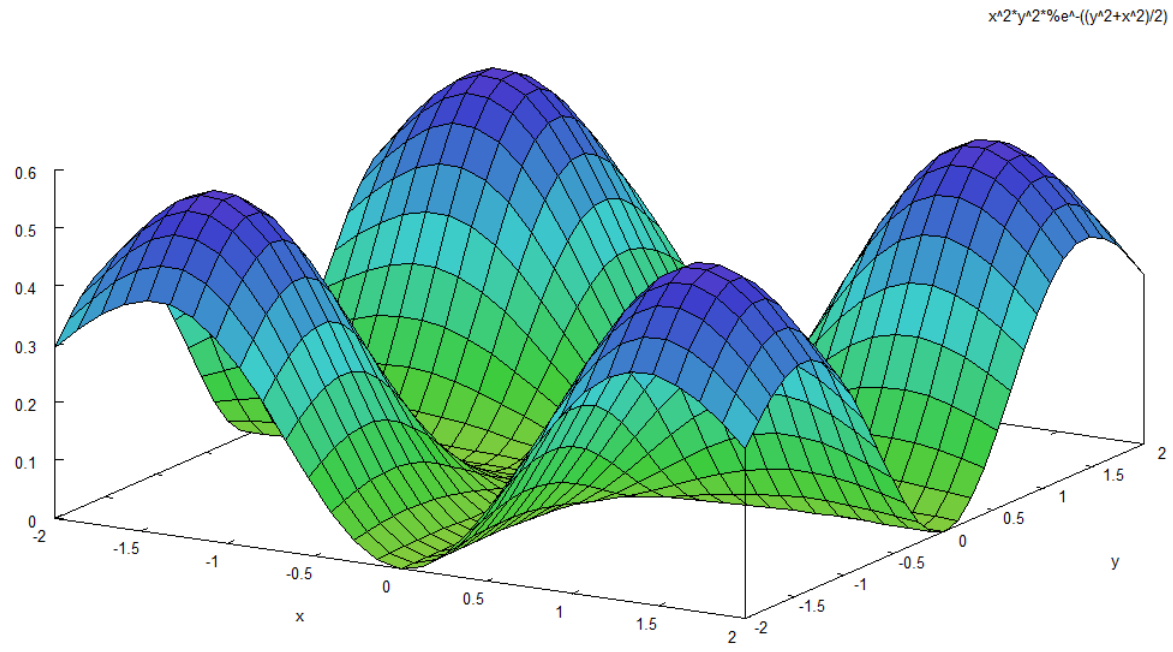

Рис. 14. График поверхности $f(x,y) = (xy)^2 e^{-\frac{x^2+y^2}{2}}$

```
(%i3) a:matrix([3/4,5/7, 1,3],[2/3, -1, 1 ,2],[6,7,2/3,1/5],[1,2, -1,3]);
```

$$(\%o3)\quad \begin{bmatrix} \frac{3}{4} & \frac{5}{7} & 1 & 3 \\ \frac{2}{3} & -1 & 1 & 2 \\ 6 & 7 & \frac{2}{3} & \frac{1}{5} \\ 1 & 2 & -1 & 3 \end{bmatrix}$$

```
(%i5) invert(a);
```

$$(\%o5)\quad \begin{bmatrix} -\frac{4144}{5753} & \frac{4623}{5753} & \frac{795}{5753} & \frac{1009}{5753} \\ \frac{3115}{5753} & -\frac{15183}{23012} & \frac{525}{23012} & -\frac{2373}{23012} \\ \frac{4361}{5753} & -\frac{7449}{23012} & \frac{735}{23012} & -\frac{12527}{23012} \\ \frac{2275}{17259} & \frac{1475}{23012} & -\frac{1165}{23012} & \frac{11195}{69036} \end{bmatrix}$$

```
(%i6) determinant(a);
```

$$(\%o6)\quad \frac{5753}{105}$$

Рис. 15. Вычисление обратной матрицы и определителя матрицы

обратной матрицы и определителя.

Для численного и аналитического решения системы линейных уравнений предназанчена функция **linsolve([eq1,eq2,...,eqn],[x1,x2,...,xk])**, где eq1, eq2, ..., eqn – линейные уравнения, x1,x2, . . . , xk – неизвестные переменные [12]. На рис. 16 представлено решение системы линейных алгебраических уравнений с помощью функции **linsolve**.

Нелинейные уравнения и системы уравнений решаются в Maxima одной функцией **solve([eq1,eq2,...,eqn],[x1,x2,...,xk])**. Здесь eq1, eq2, ..., eqn – уравнения, x1,x2, . . . , xk – неизвестные переменные [12].

Maxima можно использовать для решения классических задач математического анализа.

Предел функции в Maxima можно найти с помощью функции **limit (выражение, переменная, точка)** [12]. Пример вычисления предела функции представлен на рис. 17.

Для вычисления односторонних пределов функцию следует вызывать в виде **limit(выражение, переменная, точка, направление)**. Причем для предела справа в качестве «направления» указывается plus, для предела слева — minus

```
ex1:x+y+z-3*t=6; ex2:2*x-11*y+z+3*t=2; ex3:3*x-y+2*z-t=8; ex4:x-t=14;
```

$$z+y+x-3\,t=6$$
$$z-11\,y+2\,x+3\,t=2$$
$$2\,z-y+3\,x-t=8$$
$$x-t=14$$

```
linsolve([ex1,ex2,ex3,ex4],[x,y,z,t]);
```

$$\left[x=\frac{184}{17},\ y=-\frac{6}{17},\ z=-14,\ t=-\frac{54}{17}\right]$$

Рис. 16. Решение системы линейных алгебраических уравнений с помощью функции linsolve

```
limit((2^x-2)/log(x),x,1);
2 log(2)

limit((1-sin(x))/(%pi/2-x),x,%pi/2);
0
```

Рис. 17. Нахождение предела функции в Maxima

(рис.18) [12].

Функция **diff(f,x,n)** возвращает производную n-го порядка функции по переменной x (рис. 19), а **diff(f)** – возвращает полный дифференциал. Функция может также принимать любое нечетное число аргументов **diff(выражение, переменная, порядок, переменная, порядок, ...)** и возвращать при этом частную производную заданного порядка по заданным переменным [12].

Рассмотренные ранее функции **solve** и **diff** позволяют найти эстремум функции нескольких переменных.

Функция интегрирования имеет два варианта вызова: **integrate(выражение, переменная)** – вычисление неопределенного интеграла; **integrate(выражение, переменная, нижний-предел, верхний-предел)** – вычисление определенного

```
limit(cos(x)/(1-2*(x^2)),x,0,plus);
1

limit((cos(x)-76)/(1-2*(x^2)+64*(x^3)),x,0,minus);
-75
```

Рис. 18. Нахождение одностороннего предела функции в Maxima

```
f:((12*(6^(1/3))*((x-1)^(2/3)))/(((x+1)^2)+8));
```

$$\frac{2\,6^{4/3}(x-1)^{2/3}}{(x+1)^2+8}$$

```
u:diff(f,x,1);
```

$$\frac{4\,6^{4/3}}{3(x-1)^{1/3}((x+1)^2+8)}-\frac{4\,6^{4/3}(x-1)^{2/3}(x+1)}{((x+1)^2+8)^2}$$

Рис. 19. Аналитическое вычисление производной в Maxima

```
f(x):=cos(x)/(1+sin(x)-cos(x));
```

$$f(x):=\frac{\cos(x)}{1+\sin(x)-\cos(x)}$$

```
define(F(x),integrate(f(x),x));
```

$$F(x):=2\left(-\frac{\log\left(\frac{\sin(x)^2}{(\cos(x)+1)^2}+1\right)}{4}+\frac{\log\left(\frac{\sin(x)}{\cos(x)+1}\right)}{2}-\frac{\operatorname{atan}\left(\frac{\sin(x)}{\cos(x)+1}\right)}{2}\right)$$

Рис. 20. Вычисление неопределённого интеграла

интеграла [12]. В качестве примера расмотрим вычисление $\int \frac{cos(x)}{1-sin(x)-cos(x)}dx$ (рис. 20).

В Maxima достаточно мощные средства для аналитического решения обыкновенных дифференциальных уравнений. Программа позволяет найти общее решение обыкновенного дифференциального уравнения первого и второго порядков с помощью функции **ode2(уравнение, зависимая переменная, независимая переменная)** [4]. На рис. 21 представлены общие решения обыкновенных дифференциальных уравнений $x^2y' + 3xy = \frac{sin(x)}{x}$ и $y'' + yy'^3 = 0$.

В дополнении к функции **ode2** существуют три функции для поиска частных решений на основе полученных общих.

Функция **ic1(solution, xval, yval)** [4] предназначена для получения частного решения дифференциального уравнения первого порядка, здесь solution — общее решение, найденное функцией **ode2**; xval — значение независимой переменной, yval — значение функции при заданном значении независимой переменной. Функция **ic1** возвращает частное решение, проходящее через точку с заданными координатами (xval, yval) [3, 6].

Функция **ic2(solution, xval, yval, dval)** [4] предназначена для получения

(%i1) $x^2\,'\mathrm{diff}(y,x)+3\,y\,x=\dfrac{\sin(x)}{x}$

(%o1) $x^2\left(\dfrac{d}{d\,x}\,y\right)+3\,x\,y=\dfrac{\sin x}{x}$

(%i2) $\mathrm{ode2}(\%,y,x)$

(%o2) $y=\dfrac{\%\mathrm{c}-\cos x}{x^3}$

(%i3) $'\mathrm{diff}(y,x,2)+y\,'\mathrm{diff}(y,x)^3=0$

(%o3) $\dfrac{d^2}{d\,x^2}\,y+y\left(\dfrac{d}{d\,x}\,y\right)^3=0$

(%i4) $\mathrm{ode2}(\%,y,x)$

(%o4) $\dfrac{y^3+6\,\%\mathrm{k1}\,y}{6}=x+\%\mathrm{k2}$

Рис. 21. Общие решения обыкновенных дифференциальных уравнений $x^2y'+3xy=\frac{sin(x)}{x}$ и $y''+yy'^3=0$

частного решения дифференциального уравнения второго порядка с начальными условиями, здесь solution – общее решение уравнения, найденное с помощью функции **ode2**, xval – начальное значение независимой переменной в форме x=x0, yval — начальное значение зависимой переменной в форме y=y0, dval — начальное значение для первой производной зависимой переменной.

Функция **bc2(solution, xval1, yval1, xval2, yval2)** [4] предназначена для нахождения решения граничной задачи дифференциального уравнения второго порядка, solution – общее решение уравнения, найденное с помощью функции **ode2**; xval1 – значение независимой переменной в первой граничной точке, задается в виде x=x1, yval1 – соответственно значение зависимой переменной в точке x, задается также в виде y=y1, xval2, yval2 – вторая граничная точка, задается в той же форме, что и первая точка.

В качестве примера рассмотрим решение задачи Коши $y'=-\frac{2e^x}{e^x+2}, y(0)=\frac{1}{9}$ (см. рис. 22) с помощью функций **ode2** и **ic1**.

В результате изучения курса высшей математики с использованием пакета Maxima студенты получили опыт решения классических и прикладных математических задач на персональном компьютере. Использование Maxima в учебном процессе снимает психологический барьер при изучении курса высшей математики, делает его более интересным. Грамотное применение Maxima в учебном процессе обеспечивает повышение фундаментальности математического и техни-

```
diffur1:'diff(y,x)=-(2*exp(x))/(exp(x)+2);
```

$$\frac{d}{dx}y=-\frac{2\,\%e^{x}}{\%e^{x}+2}$$

```
solv:ode2(diffur1,y,x);
```

$$y=\%c-2\,\log(\%e^{x}+2)$$

```
F:ic1(solv,x=0,y=1/9);
```

$$y=-\frac{18\,\log(\%e^{x}+2)-18\,\log(3)-1}{9}$$

Рис. 22. Решение задачи Коши $y' = -\frac{2e^x}{e^x+2}, y(0) = \frac{1}{9}$

ческого образования, содействует подлинной интеграции процесса образования.

Заключение

Современные свободные математические программы яавляются реальными конкурентами проприетарных приложений. Отсутствие лицензионной платы позволяет широко использовать свободные математические пакеты в образовании и научных исследованиях. Открытость кода свободных математических программ позволяет любому пользователю расширять функциональность системы. Конечно, освоение подобных систем сложнее, чем таких проприетерных приложений, как MathCAD и MS Excel. Однако, потратив некоторое время на освоение системы, учёный получает мощный инструмент на многие годы. Кроме того, пользователей свободных приложений не волнуют проблемы, с которыми постоянно сталкиваются пользователи проприетарных приложений: изменится или нет в последующих версиях программы внутренний формат данных, на сколько дороже станет новая версия программы. Авторы рекомендуют широко использовать свободные математические программы в университетах и исследовательских лабораториях. Стоит хотя бы попробовать.

Список литературы

1. Алексеев Е. Р., Чеснокова О. В. Решение задач вычислительной математики в пакетах MathCad 12, MATLAB 7, Maple 9. Самоучитель. — М.: НТ Пресс, 2005. — 496 с.
2. Алексеев Е. Р., Чеснокова Е. А., Рудченко Е. А. Решение инженерных и математических задач в пакете Scilab. — М.: ALT Linux, 2008. — 257 с.
3. Алексеев Е.Р., Чеснокова О.В. Черновик книги "GNU OCTAVE для преподавателя и студента"// http://gnu-octave.narod2.ru/

4. Губина Т.Н., Андропова Е.В. Решение дифференциальных уравнений в системе компьютерной математики Maxima: учебное пособие. — Елец ЕГУ им.И.А.Бунина, 2009. — 99 с.
5. Система компьютерной алгебры Maxima. // http://maxima.sourceforge.net/ru
6. Ильина В. А., Силаев.П.К. Система аналитических вычислений MAXIMA для физиков-теоретиков. — М.: МГУ, 2007. — 112 с.
7. Тарнавский Т. Maxima — максимум свободы символьных вычислений. // http://maxima.sourceforge.net/ru/maxima-tarnavsky-1.html
8. Тарнавский Т. Maxima. Функции и операторы. // http://maxima.sourceforge.net/ru/maxima-tarnavsky-2.html
9. Тарнавский Т. Maxima. Графики и управляющие конструкции.// http://maxima.sourceforge.net/ru/maxima-tarnavsky-5.html
10. Тарнавский Т. Maxima. Алгебра и начала анализа. // http://maxima.sourceforge.net/ru/maxima-tarnavsky-4.html
11. Тропинин И.С., Михайлова О.И., Михайлов В.И. Численные и технические расчеты в среде Scilab (ПО для решения задач численных и технических вычислений): Учебное пособие. — Москва: 2008. — 65 с.
12. Чичкарёв Е.А. Компьютерная математика с Maxima. Руководство для школьников и студентов. // http://www.altlinux.org/Books:Maxima
13. Octave. // http://www.gnu.org/software/octave
14. Octave-Forge. // http://octave.sourceforge.net/
15. Scilab Home Page. // http://www.scilab.org/
16. Stephen L. Campbell, Jean-Philippe Chancelier, Ramine Nikoukhah. Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4. — Springer, 2010. — 330 p.
17. Xcos hints.// http://cyxtp.narod.ru/xcos/xcos_hints.html