МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ УЧРЕЖДЕНИЕ
КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

М.А.Тынкевич

Система MATLAB

Справочное пособие к курсу
“ЧИСЛЕННЫЕ МЕТОДЫ АНАЛИЗА”

для студентов специальности
«Прикладная информатика в экономике»

Кемерово 2002

Введение в MatLab (происхождение и возможности)

Электронные вычислительные машины (ЭВМ) первого поколения, в основном, были востребованы именно для выполнения вычислений при расчете баллистических таблиц, оболочек ядерных реакторов, траекторий вывода на орбиту космических аппаратов и в других, более скромных, научных и инженерных расчетах. Программист того времени должен был искать взаимопонимание с заказчиком в математической постановке задачи, быть специалистом в области вычислительной математики, способным найти или создать численный метод решения задачи и, зная систему команд ЭВМ и основные приемы программирования, составить машинную программу.

Система команд первых ЭВМ (для каждого типа машин своя) включала достаточно ограниченный набор элементарных операций (арифметических и логических, пересылки между ячейками памяти, перехода по условию) и лишь избранные ЭВМ (например, первая наша серийная машина “Стрела” с памятью в 2048 43-разрядных ячеек и быстродействием 2000-4000 операций/сек) имели в этом наборе более сложные операции (перевод числа из двоично-десятичной системы в двоичную и обратно, вычисления синуса, натурального логарифма, экспоненты, обратной величины и квадратного корня). Построение из этих “кирпичиков” программ для реальных задач в условиях ограниченной емкости памяти и невысокого (по современным меркам) быстродействия было достаточно трудоемким процессом и не случайно возникло понятие *искусства программирования*.

Естественно, что уже на первых этапах практического программирования появилась идея *подпрограммы* как программного блока, допускающего многократное обращение к себе из различных участков программы. Так появились сборники текстов подпрограмм для вычисления элементарных функций и основных численных методов (интегрирования, решения обыкновенных дифференциальных уравнений и пр.), позднее эти сборники стали хранить во внешней памяти машины (как правило, на магнитных лентах или барабанах). Наконец, были созданы системы, позволяющие по номеру подпрограммы вызывать ее из внешней памяти в оперативную с настройкой по месту вызова и при наличии определенных договоренностей передавать входную и выходную информацию. В отечественной практике наиболее совершенной была интерпретирующая система ИС-2 (позднее ИС-22) для семейства машин типа М-20 с уникальной по качеству и разнообразию *библиотекой стандартных подпрограмм.*

Переход от программирования в кодах ЭВМ к универсальным

языкам программирования – в первую очередь, к Алголу-60 - породил *библиотеки алгоритмов*, которые и составили базу для построения 30 лет спустя систем для решения математических задач, возникающих в разнообразных научных исследованиях и технических разработках – Maple, Mathematica, MathCad, MatLab и др. Для этих систем характерны простота подготовки данных, удобные формы вывода результатов вычислений, встроенные средства помощи, диагностики ошибок – т.н. *дружественный интерфейс*.

Рассматриваемая ниже система MatLab (Matrix Laboratory) является интерактивной системой для выполнения инженерных и научных расчетов, ориентированная на ***работу с массивами*** данных. Она допускает написание на специальном языке программ, оформляемых в виде т.н. ***М-файл***ов, поддерживает работу в программном и интерактивном режиме с векторами и матрицами, позволяет решать системы уравнений, выполнять численное интегрирование, строить графики и пр. Система допускает использование пакетов прикладных программ (ППП) символьной математики, статистики, оптимизации, анализа и синтеза систем управления, обработки сигналов и изображений, финансов, картографии и др. Система позволяет с легкостью обмениваться информацией с текстовым редактором Microsoft Word, в частности переносить любые тексты и рисунки в буфер или читать текстовые строки из буфера как исполняемые команды.

Опыт показывает, что студенты, владеющие программированием в Pascal'е или Visual Basic'е, улавливают технологию программирования и работы в MatLab'е в течение 2-3 часов (с фантастическими возможностями системы можно знакомиться месяцами).

Ниже мы излагаем информацию лишь о небольшой части этой системы, полезную при изучении численных методов решения основных задач вычислительной математики студентами, основные интересы которых ограничены областью экономико-математического моделирования.

1. Режим командной строки. Форматы данных

При вызове MatLab на дисплей выводится заставка, которая сменяется *командным окном*, в верхней части которого размещено окно управления - меню с пунктами *Файл*, *Правка*, *Окно* и *Помощь* и панель инструментов. Ниже выводится ***командная строка*** (начинается символом "»") с предварительными предложениями вызвать перечень разделов, войти в справочник, открыть окно помощи, приступить к демонстрации и др.

В командной строке в режиме диалога можно *набрать команду (оператор)* или *выражение* и, нажав Enter, получить ответ (*answer*). Например, после набора команды (оператора присваивания) **a=3.2** в последующих строках появится **a = 3.20000000000000** (переменной **a** присвоено значение 3.2), после набора выражения **sin(a)/a** увидим его значение **ans = -0.01824191982112**.

Сразу же учтите, что в именах переменных (последовательностях латинских букв и цифр, начинающихся с буквы; знак _ относится к буквам) строчные и заглавные буквы отнюдь не тождественны !

Если вы хотите выполнить команду без вывода результата, в конце команды ставьте символ точки с запятой.

Кстати, если команда не помещается полностью в видимой части одной строки экрана, поставьте многоточие (хотя бы две точки), нажмите Enter и продолжайте в следующей строке.

Для *очистки командного окна* достаточно выполнить команду **clr**.

Заметим, что всегда можно обратиться к помощи, выбрать интересующий раздел (например, **matlab/elfun** – элементарные математические функции) и воспользоваться полученной информацией. Имейте в виду, что любой фрагмент окна командной строки можно выделить и копировать в буфер, например, для переноса в Word. Возможен перенос в командную строку текстовых фрагментов из других систем.

Полезно сразу обратить внимание на подпункт *Свойства* (*Preferences*) пункта *Файл* (*File*) окна управления.

В первом его окне (*General*) предусматривается установка Numeric Format формата представления чисел : **Short** – короткое 5-значное , **Long** – длинное 15-значное, **Hex** – шестнадцатеричное, **Bank** – доллары и центы, **ShortE** и **LongE** – экспоненциальное, **Rational** – отношение целых чисел (обратите внимание на эту форму – такой нет ни в одной универсальной среде программирования), межстрочного интервала (с пробелом между строками **Loose** или без такового **Compact**), а также вывод на экран панели инструментов и поддержка возможности отладки графики.

Во втором окне (*Command Window Font*) имеются 6 полей: Font (шрифт), Style (Light-светлый, Regular - нормальный, Bold – жирный), Size (размер 10, 12 или 15), BackGround Color (цвет фона) , Color (цвет символа) и др.

Как и в любой системе, в MatLab’е присутствует понятие *переменной* величины, но в роли ее значения выступает *массив* (*array*).

В системе определены 6 встроенных типов данных (массивов):

- числа удвоенной точности (**double**);
- массивы символов – строки (**char**), при задании строковой константы ее символы заключают в апострофы;
- двумерные разреженные матрицы (**sparse**), массивы ячеек (**cell**), массивы записей (**struct**) и специальные массивы целых чисел от 0 до 255 (uint8).

Здесь мы ограничимся рассмотрением лишь обычных числовых массивов и строк.

Для задания массива (в частности, скалярной величины) используется *команда присваивания*. Например, командой **» a=[1 2 3; 4 5 6]** формируется матрица размерности 2×3 с соответствующими элементами; командой **» b=[1 2 3]** – вектор-строка; командой **» b=[1; 2; 3]** - вектор-столбец; **d=zeros(4,7)** - матрица размерности 4×7 с нулевыми элементами. Для выборки отдельных элементов массивов можно пользоваться индексами, например, a(k,3) определяет третий элемент k-ой строки, a(:,3) – весь третий столбец. Встроенная система контроля отлавливает типичные ошибки при задании массивов: например, при попытке выполнения команды **» a=[1 2 3; 4 5]** получаем:

Number of elements in each row must be the same.

(Число элементов в каждой строке должно быть тем же)

Обратите внимание на то, что следует :

- при задании массива значениями заключать их в квадратные скобки;
- элементы в строке массива разделять пробелами или запятыми;
- при указании списка индексов использовать круглые скобки и разделительные запятые (указание индекса символом двоеточия соответствует заданию всех значений по соответствующему индексу).

При работе с массивами можно пользоваться списками **i:k** и **i:j:k** : в первом варианте понимаем “от **i** до **k** с шагом **1**” и во втором – то же с шагом **j**, например t=-pi:0.01:pi или p=0:8 (некоторые сочетания дают пустое множество, например q=3:1).

В библиотеке предусмотрен ряд функций для формирования массивов простейшей структуры, например :

- нулей zeros(n), zeros(m,n), zeros(m,n,p,..), zeros(size(A)) (одномерный, двумерный, многомерный, соразмерный с массивом A);
- единиц ones(n), ones (m,n), ones (size(A)) и др.

Естественно, что к числовым переменным применимы все арифметические операции, но при выполнении ряда операций приходится различать *поэлементные операции* с массивами и *операции над матрицами* по правилам линейной алгебры (для массивов перед знаком операции ставят точку):

+A -A		A\B	Решение системы m уравнений AX=B с несколькими правыми частями: B –матрица m×k {тождественно (B'/A')' }
A+B A-B A .*B	Предполагается одинаковая размерность или один из операндов – скаляр		
A.\ B	Левое деление (B на A)		
A ./B	Правое деление (A на B)		
A .^B	Поэлементное возведение в степень	A^k	Степень матрицы (при k=0 – единичная матрица, при целом k<0 – умножение обратной и при целом k>0 исходной матриц; для других k вычисляются собственные числа R и векторы D и A^k= R^TD.^k/R
A.'	Транспонирование массива	A'	Транспонирование матрицы (для комплексных дополняется комплексным сопряжением)

Система работает не только с действительными, но и **с комплексными числами** и роль мнимой единицы играют символы **i, j:**

» a=1+2i a = 1.0000 + 2.0000i
» b=1-3i b = 1.0000 - 3.0000i
» a*b ans = 7.0000 - 1.0000i
» exp(ans) ans = 592.51 -922.78 i

Над массивами можно выполнять *операции поэлементного отношения* : A < B , A <= B , A > B , A>= B (только для действительных частей), A == B, A~=B (равно/не равно - для действительных и мнимых частей), которые порождают массив с единицами (*истина*) и нулями (*ложь*) той же размерности. Аналогично реализуются и *логические операции*: отрицания ~A, конъюнкции (логического умножения - И) A**&**B, дизъюнкции (логического сложения – ИЛИ) A|B.

Все переменные системы размещаются в т.н. **рабочей области**, содержимое которой (имена, размерность, тип) можно просмотреть командами **who** и **whos**:

```
» who
Your variables are:
i       r       t
» whos
  Name      Size          Bytes  Class
  i         1x1               8  double array
  r         4x629         20128  double array
  t         1x629          5032  double array
Grand total is 3146 elements using 25168 bytes
```

Рабочую область можно сохранять (**save**) как MAT-файл и вызывать (**load**) . Так командой **save myfile v1 v2** сохраняем в файле с именем **myfile** переменные **v1**и **v2**, а командой **save myfile du*rak** – переменные, имена которых начинаются на **du** и заканчиваются на **rak**.

Можно очищать рабочую область полностью командой **clear** или частично - **clear** <список имен>.

2. Элементарные математические функции

Из многообразия т.н. элементарных встроенных математических функций отметим лишь некоторые наиболее известные или оригинальные. Аргументами большинства из приведенных ниже функций являются не только скаляры, но и массивы.

pi = 4*atan(1)=imag(log(-1))=3.1415926535897..;

abs(X) – абсолютная величина: для комплексного числа $a+bi$ его модуль равен $\sqrt{a^2+b}$ / abs(3-4i)=5 , abs(-13)=13;

angle(X) – аргумент комплексного числа (из диапазона [-π,π]): комплексное $X=a+bi$ представимо как $r\cdot e^{i\varphi}$, где $a = r\cos\varphi$, $b = r\sin\varphi$:

» angle(3+4i) ans = 0.9273 ; » angle(1) ans = 0 ;

» angle(4+3i) ans = 0.6435 ;

real(X), **imag**(X) – действительная и мнимая часть числа;

conj(X) – комплексно-сопряженное:

» conj(2+3i) ans = 2.0000 - 3.0000i ;

ceil(X), **fix**(X), **floor**(X), **round**(X) - округления (до ближайшего целого, не меньшего X; отбрасывание дробной части; до ближайшего целого, не большего X; до ближайшего целого);

mod(X,Y) - остаток от деления X на Y;

sign(X) – знак числа (для комплексных X / |X|);

gcd(m,n) –наибольший общий делитель для целых чисел; если использовать оператор [g,c,d]=gcd(m,n), то дает указанный делитель и множители c,d такие , что g==m*c+n*d :

» f=gcd(18,27) f = 9

» [g,c,d]=gcd(18,27) g = 9 c = -1 d = 1 ;

lcm(m,n) – наименьшее общее кратное:

» lcm(34,51) ans = 102 ;

rat(X) , **rat** (X,k) – представление цепной дробью с точностью $|X|\cdot 10^{-k/2}$ (по умолчанию $|X|\cdot 10^{-6}$):

» rat(12.5) ans =13 + 1/(-2)

» rat(12.546) ans =13 + 1/(-2 + 1/(-5 + 1/(15))) ;

rats(X), **rats**(X,k) – представление отношением целых чисел :

» rats(12.546) ans = 2045/163 ;

sqrt(X) – квадратный корень :

» sqrt(5) ans = 2.2361

» sqrt(3+4i) ans = 2.0000 + 1.0000i;

exp(X) – экспонента e^x ($e^{x+iy}= e^x(\cos y+i \sin y)$) :

» exp(1) ans = 2.7183

» exp(2+i) ans = 3.9923 + 6.2177i ;

pow2(X) – двоичная экспонента 2^x;

log(X) – натуральный логарифм;

log2(X), **log10**(X) –логарифм по основанию 2 и основанию 10;

sin(X) **cos**(X) **tan**(X) **cot**(X) **csc**(X) **sec**(X) – тригонометрические функции (синус, косинус, тангенс, котангенс, косеканс, секанс):

sin(x+iy)=sin(x) ch(y) +i cos(x) sh(y); cos(x+iy)=cos(x) ch(y) -i sin(x) sh(y),

tg (X)= sin(X)/ cos(X) ; ctg(X)=cos(X)/sin(X);

cosec(X)=1/sin(X); sec(X)=1/ cos(X) :

» sin(pi/2) ans = 1 ; » sin(3+4i) ans = 3.8537 -27.0168i ;

asin(X) **acos**(X) **atan**(X) **acot**(X) **acsc**(X) **asec**(X) – обратные тригонометрические функции (арксинус, арккосинус и т.д.):

» asin(1/sqrt(2)) ans = 0.7854 ; » asin(3+4i) ans = 0.6340 + 2.3055i ;

atan2(Y,X) - круговой арктангенс Arctg (только для действительных частей аргументов), берется в интервале [-π,π];

sinh(X) **cosh**(X) **tanh**(X) **coth**(X) **csch**(X) **sech**(X) – гиперболические функции (синус, косинус, тангенс, котангенс, косеканс, секанс): $sh(X)=(e^X-e^{-X})/2$, $ch(X)=(e^X+e^{-X})/2$ и др.;

asinh(X) **acosh**(X) **atanh**(X) **acoth**(X) **acsch**(X) **asech**(X) – обратные гиперболические функции:

$$arsh(X)=ln(X+\sqrt{X^2+1}),\ arch(X)=ln(X+\sqrt{X^2-1})$$

$$arth(X)=\frac{1}{2}ln\frac{1+X}{1-X},\qquad arcth(X)=\frac{1}{2}ln\frac{1-X}{1+X},$$

arcsch(X)=arsh(1/X), arsech(X)=arch(1/X);

erf (X)- интеграл вероятностей (функция Гаусса, функция ошибок)

$$erf(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt$$

и родственные функции :

erfc(x)=*1-erf(x)* (дополнительный интеграл вероятностей) ;

erfcx(x)=$exp(x^2)\cdot erfc(x)$ (нормированный дополнительный интеграл вероятностей);

erfinv(x) (аргумент, для которого интеграл вероятностей равен x);

gamma(x) -гамма-функция $\Gamma(x)=\int_0^\infty e^{-t}t^{x-1}dt$ (при целочисленных x $\Gamma(1+x)=x!$)

» gamma(5) ans = 24

» gamma(0) Warning: Divide by zero (деление на нуль).

ans = Inf (неопределенное значение)

» gamma(0.5) ans = 1.7725

» gamma(-0.5) ans = -3.5449

» gamma(0.1) ans = 9.5135

и родственные функции :

gammainc(x,a)= $P(x,a)=\frac{1}{\Gamma(a)}\int_0^x e^{-t}t^{a-1}dt$ (неполная гамма-функция);

gammaln(x)=$ln\ \Gamma(x)$ (логарифмическая гамма-функция);

beta(x,y) - бета-функция $B(x,y)=\int_0^1 t^{x-1}(1-t)^{y-1}dt=\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$

и родственные ей неполная и логарифмическая бета-функции;

функции преобразования координат:

из декартовых (X,Y) в полярные (r,φ): $r=(X^2+Y^2)^{1/2}$, $\varphi=Arctg(Y/X)$ – **[φ,r]=cart2pol(X,Y)**;

из декартовой системы (X,Y,Z) в цилиндрическую (r,φ,Z) - **[φ,r,Z]=cart2pol(X,Y,Z)** ;

из декартовой системы в сферическую (r,φ,θ) : $r=(X^2+Y^2+Z^2)^{1/2}$, $\varphi=Arctg(Z/(X^2+Y^2)^{1/2})$, $\theta=Arctg(Y,X)$ - **[θ,φ,r]=cart2sph(X,Y,Z)**;

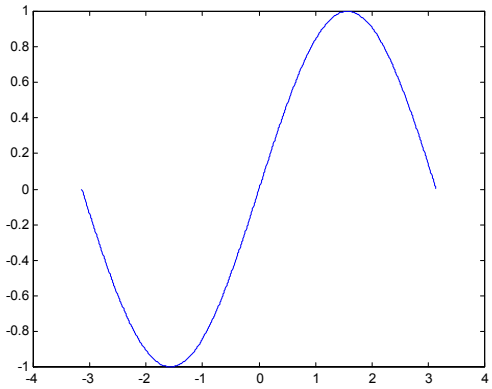
из полярной и цилиндрической в декартову (**pol2cart)**: X=r·cos(φ), Y=r·sin(φ) ; из сферической в декартову(**sph2cart)**: Z=r·sin(φ), X=r·cos(φ)·cos(θ), X=r·cos(φ)·sin (θ) (эти функции незаменимы при графических отображениях результатов анализа, хотя многое из их графики уже предлагается среди готовых библиотечных средств);

специальные функции (цилиндрические функции Бесселя, Неймана, Ханкеля; функции Эйри, эллиптические функции Якоби и эллиптические интегралы, интегральная показательная функция, присоединенные функции Лежандра и много других функций, полезных при изучении физических процессов),

функции линейной алгебры, аппроксимации данных, численного интегрирования, поиска корней уравнений, обслуживания графики, обработки дат, множеств и др.

Если вы имеете намерение познакомиться с поведением какой-то из функций, поступите по аналогии с примитивным примером:

```
»  t=-pi:0.01:pi;  % значения аргумента
                    от  -π до π с шагом 0.01
                   (без вывода на экран);
» e=sin(t);         %  массива значений
                    функции;
» plot(t,e)         % построение графика
                    функции .
```



3. Режим программирования

Для перехода в режим программирования в окне управления выбираем пункт File и входим в редактор MatLab'a: *New* (создать новый М-файл) или *Open*(открыть существующий файл с расширением **.m**). В дальнейшем может производиться обычный набор текста программы или его корректура и действия в соответствии с меню (сохранение под текущим или другим именем, запуск на исполнение в обычном и отладочном режимах и др.).

Различают два вида М-файлов: *М-сценарии* и *М-функции*.

М-сценарий – это файл, содержащий последовательность команд и комментариев (строк, начинающихся символом %) и пользующийся данными из рабочей области. Заголовок его начинается командой script или может отсутствовать.

Для М-функции допускаются входные и выходные аргументы, локализация внутренних ее переменных и возможность обращения к ней из других программ. М-функции включаются в библиотеку функций системы в виде текстовых файлов.

Заголовок М-функции имеет вид:

function [<список выходных переменных>**]=**
<имя функции> **(**<список входных переменных>**)**

например, функция вычисления факториала положительного числа и его обратной величины может быть описана файлом fact.m:

```
function [f, g]=fact(n)   %  факториал и обратная величина
f=prod (1:n);
g=1/f;
```

Кроме использованных выше операторов присваивания, программирование в MatLab'е допускает и ряд других традиционных для прогрммных сред операторов.

Условный оператор выступает в одной из следующих форм:

```
if <условие>          if <условие>          if <условие>
    <команды>             <команды>             <команды>
end                   else                  elseif <условие>
                          <команды>             <команды>
                      end                     else
                                                  <команды>
                                              end
```

В роли условия может использоваться любое логическое выражение, построенное на основе операций отношения и логических. Если значение этого выражения является массивом, то условие считается истинным, если все его элементы истинны (истина – 1, ложь – 0).

Оператор цикла с заданным числом повторений, в основном используемый в форме:

```
for V=A :H:B          for V=A:B
    <команды>             <команды>
end                   end
```

(V –переменная/параметр цикла, A,B – начальное и конечные значения; H – приращение, по умолчанию 1). Допускаются и вложенные циклы, например:

```
for i=1:n                     for i=1:n-1
    for j=1:m                     for k=i+1:n
        a(i,j)=x(i)^j;                if a(i)<a(k)
    end                                   m=a(i)
end                                       a(i)=a(k)
                                          a(k)=m
                                  end
                              end
                          end
```

В заголовке цикла можно использовать одномерный массив. Так цикл

```
k=1;
for i=[0 5 7]
    x(k)=2^i;
    k=k+1;
    end
```

формирует массив X=[1 32 128].

Оператор цикла с предусловием имеет традиционную конструкцию:

```
while <условие>
    <команды>
end
```

и обеспечивает выполнение команд тела цикла, пока истинно проверяемое условие. Заметим, что работа цикла может быть прервана (вы-

ход из внутреннего цикла) оператором **break**:

```
while a<1
        n=n+1
         if   n>250
                break
end ...
```

Оператор переключения обобщает условный оператор на случай более двух условий и имеет конструкцию:

```
switch  <выражение>
        case <значение 1>
                <команды>
        case <значение 2>
                <команды>
            . . . . .
        otherwise                % может отсутствовать
                <команды>
        end
```

Контрольные значения проверяются на равенство и могут задаваться и списком:

```
swith  k
        case 0
                t=1
        case (1, 2,5)
                t=2
        otherwise
                t=0
end
```

Выход из функции в вызывающую программу обеспечивается выполнением последнего ее оператора или командой **return**.

Кроме упомянутых основных операторов, традиционных для любой системы программирования, остановимся на ряде операторов обеспечения пользовательского интерфейса.

Ввод с клавиатуры реализуется командой вида

<переменная>= input (‘подсказка’)

Например,

```
» x=input(' степень полинома ');
степень полинома 3
```

Приостановка выполнения программы может быть предусмотрена включением в текст команды **pause** (приостановка до нажатия любой клавиши), **pause (n)** (приостановка на n сек), **keyboard** (приостановка с возможностью выполнять практически любые команды и последующим возвратом в программу командой **return**).

Можно построить выбор варианта с клавиатуры созданием **меню**:

<переменная>=**menu**(‘заголовок’,’выбор1’,’выбор2’,...)

Например, команда:

k=**menu**(‘Использовать метод’,’Гаусса’,’Краута’,’простой итерации’)

создаст на экране всплывающее меню с указанными пунктами-клавишами и щелчок по клавише задаст значение переменной k, равное 1, 2 или 3.

Мы не останавливаемся на многообразии операторов, связанных с выводом на экран (вывод значения **disp,** форматированный вывод **fprint**), отладкой и сигнализацией об ошибках, анализом списка аргументов и др.

Приведем несколько простейших примеров использования программного режима.

Пример 1. Анализ скорости убывания элементов числовой последовательности

$$y_n = {(-1)^{n+1}}/{n^n}\ .$$

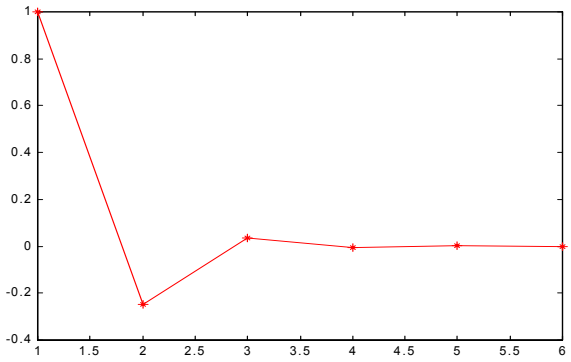
до значений, меньших 0.0001)

Чтобы с легкостью отыскивать значения элементов последовательности, опишем *функцию* (файл *y.m*):

```
function f=y(n)
    if mod(n,2)==0
        f=-1;
    else
        f=1;
    end
    f=f/n^n;
end
```

и *сценарий* (файл *limit1.m*):

```
n=1 ;
while abs(y(n))>1e-4
    n=n+1 ;
end
disp('Число элементов последовательности равно')
k=n
for x=1:k
    Y(x)=y(x);
end
disp('Значения элементов последовательности')
Y
plot(1:k,Y, 'r-*' )   %Линия -сплошная(-)красная ®, маркеры(*)
```



Теперь можно в командной строке набрать вызов ***limit1***, получая на экране число элементов последовательности ***k***, значения элементов последовательности ***Y (n=1÷k)*** и "график" функции:

```
Число элементов последовательности равно
k =6
Значения элементов последовательности
Y =    1.0000   -0.2500    0.0370   -0.0039    0.0003   -0.0000
```

Пример 2. Поиск оценки суммы ряда.

$$\sum_{m=1}^{\infty}(-1)^{m-1}\frac{1}{(2m-1)\cdot(2m-1)!}$$

с точностью 10^{-6}.

*Файл **limit2.m** (сценарий):*

```
y=1;
s=y ;
m=1;
while abs(y)>1e-6
      y=-y*(2*m-1)/(2*m+1).^2;
      s=s+y;
      m=m+1 ;
 end
disp('Число слагаемых' )
disp(  m-1)
disp('Оценка суммы')
disp(s)
```

В командной строке набираем ***limit2*** , получая на экране число элементов отрезка суммы, превышающих 10^{-6} и саму оценку суммы:

```
Число слагаемых   6
Оценка суммы       0.90097107966794
```

4. Операции над массивами

Формирование массива, как было показано выше, осуществляется прямым (построчным) перечислением его элементов подобно A=[1 3 5 7; 4 5 6 7] (2 строки и 4 столбца), B=[1; 3; 5; 7] (столбец с 4

элементами) или заданием диапазона значений с заданным (или умалчиваемым единичным) шагом [1:2:7], [4:7], [[1:2:7]; [4:7]] и т.п.

Доступ к элементам или блокам элементов массива производится указанием индексов или массива индексов:

A(2,k) – элемент второй строки и k-го столбца;

A(:,k) - k-й столбец;

A(1:3; 1:4) – подматрица из первых 3 строк и 4 столбцов матрицы;

C(:, :, 12) –12-я страница трехмерного массива.

Следует учесть, что хранение массивов в памяти ведется по столбцам. Поэтому возможна работа с созданным многомерным массивом как с одномерным, например, A(:) – вектор-столбец из всех элементов массива A, A(13:17) – столбец из элементов с номерами от 13 до 17.

Имеется возможность **объединять массивы "по горизонтали"**-[A, B, C] или [A B C] (массивы с одинаковым числом строк) и **"по вертикали"** -[A; B;C] (массивы с одинаковым числом столбцов).

Из вектора можно **удалить одинаковые элементы** функцией **unique**(X). Существует возможность **объединения** множеств - **union**(X,Y), **пересечения - intersect**(X,Y), **разности – setdiff**(X,Y):

» a=[1 2 3 6]; » b=[1 3 7];	» union(a,b) ans = 1 2 3 6 7	»intersect(a,b) ans = 1 3	» setdiff(a,b) ans = 2 6

Функция **find** дает поиск по условию элементов одно- или двухмерного массива в формате команд k=find(X<условие>), [i,j]=find(A<условие>) (если условия нет, отыскиваются ненулевые элементы):

» X=[1 0 -3 6 7 13] X = 1 0 -3 6 7 13 » A=[1 4 7 ; 2 0 -2] A = 1 4 7 2 0 -2	» k=find(X==0 | X<0) k = 2 3 » k=find(X) k = 1 3 4 5 6	»[i,j]=find(A>0& A<5) i = 1 2 1 j = 1 1 2

Для определение **длины вектора** используется функция **length** :

» k=length('Это строка') k = 10	» X=[1 0 -3 6 7 13]; » k=length(X) k = 6	» k=length([1 4 7 ; 2 0 -2]) k = 3

и для **размеров массива** – функцию **size :**

» X =[1 0 –3 ; 6 7 13] » k=size(X) k = 2 3	» [m, n]=size(X) m = 1 n = 6	» size([2 4 7]) ans = 1 3

Суммирование и умножение элементов массива можно реализовать функциями **sum**(A) и **prod**(A) (для двумерного массива выполняется поиск сумм и произведений по столбцам). С помощью функций sum(A,dim) и prod(A,dim) можно выполнить операции по измерению dim. Функцию sum часто используют для поиска **скалярного произведения** векторов $(A \cdot B) = \sum_{i=1}^{n} A_i B_i$ в форме sum(A.*B):

» a=[1 2 3; 4 7 -1]; » sum(a,2) ans = 6 10	» t=sum(a) t = 5 9 2 » tt=sum(t) tt = 16	» p=prod(a) p = 4 14 -3 » p=prod([1:5]) p = 120	» a=[1 2 3]; » b=[3 5 7]; » sum(a.*b) ans = 34

Сортировку элементов массива по возрастанию можно выполнить функцией **sort(A,dim)** , причем команда **[B,I]= sort(A)** выдает и список индексов. **Сортировку по убыванию** можно выполнить аналогичной функцией **sortrows.**

Среди других следует отметить и ряд функций комбинаторики:

perms(V) –перестановки всех элементов вектора V размерности n (массив размерности n!×n) :

» perms (3:2:7)

ans =[7 5 3; 5 7 3; 7 3 5; 3 7 5; 5 3 7; 3 5 7]

» perms([3 2 7])

ans =[7 2 3; 2 7 3; 7 3 2; 3 7 2; 2 3 7; 3 2 7] ;

nchoosek (n,k) – число сочетаний из n по k =n!/ (k! (n-k)!):

» nchoosek(7,2) ans = 21;

nchoosek (V,k) – массив всех сочетаний элементов вектора V:

» nchoosek([3 2 7],2) ans = [3 2; 3 7; 2 7];

Иногда могут быть полезными функции начального задания:

zeros(n), zeros(m,n), zeros(size(A)) – формирование массива нулей (одномерного, двумерного, соразмерного с массивом A); допустимо формирование массива и большей размерности zeros(m,n,p,...) ;

ones(n), ones (m,n), ones (size(A)) - формирование массива единиц ;

rand(n), rand (m,n), rand (size(A)) - формирование массива чисел с равномерным законом распределения в (0,1);

randn(n), randn (m,n), randn (size(A)) -формирование массива чисел с нормальным законом распределения (Mx=0, Dx=1);

eye(n), eye(m,n), eye(size(A)) - формирование единичной матрицы (n×n, m×n, соразмерной с матрицей A):

» eye(2,3) » eye(2)

ans = 1 0 0 ans = 1 0

0 1 0 0 1

Отметим также и некоторые полезные конструкции:

cross(X,Y) – **векторное произведение** (X,Y-трехмерные векторы):

$$X \times Y = [\,(X_2Y_3 - X_3Y_2), (X_3Y_1 - X_1Y_3), (X_1Y_2 - X_2Y_1)\,];$$

kron(X,Y) – **тензорное произведение** (произведение Кронекера):

$$\begin{vmatrix} X_{11}Y & X_{12}Y & \dots & X_{1n}Y \\ X_{21}Y & X_{22}Y & \dots & X_{2n}Y \\ \dots & \dots & \dots & \dots \\ X_{m1}Y & X_{m2}Y & \dots & X_{mn}Y \end{vmatrix}$$

meshgrid(X,Y), **meshgrid**(X,Y,Z) –**формирование двумерной (трехмерной) сетки** (обычно используется при реализации графики):

```
[x,y]=meshgrid(-2:0.1:2,   -10:0.5:10)
```

Ряд других функций, связанных с обработкой массивов, рассмотрен ниже.

5. Решение основных задач линейной алгебры

При реализации многих задач, связанных с матричной алгеброй, полезными могут оказаться функции для оценок основных характеристик:

det(A) – определитель квадратной матрицы;

rank(A) – ранг матрицы;

trace(A) – след матрицы (сумма элементов главной диагонали);

```
» A=[1 2 3; 5 4 3; 3 4 3]
A =     1    2    3
        5    4    3
        3    4    3
```

```
» det(A)
  ans =12
```

```
» rank(A)
  ans =  3
```

```
» trace(A)
ans =  8
```

Выше среди операций системы мы уже упоминали операцию **транспонирования** матрицы :

```
» A=[ 1 2 3 ; 23 11 0]
A =
        1    2    3
       23   11    0
```

```
» B=A'
B =     1   23
        2   11
        3    0
```

и возведения в степень (матричного умножения на себя или инверсии):

```
» A=[1 2 3; 5 4 3; 3 4 3]
A =
    1    2    3
    5    4    3
    3    4    3
```

```
» D=A^(-1)
D =
  -0.0000   0.5000   -0.5000
  -0.5000  -0.5000    1.0000
   0.6667   0.1667   -0.5000
```

```
» A^0
ans =
     1     0     0
     0     1     0
     0     0     1
```

При решении многих задач (например, при оценке сходимости методов) используется понятие **нормы** вектора (матрицы). В рассматриваемой системе для поиска нормы предлагается функции **norm(A)** и **norm(A, k)** .

Если A – вектор, то норма определяется (по умолчанию k=2)

$$\|A\| = \sqrt[k]{\sum_{i=1}^{k} |A_i|^k} \; ;$$

при k=inf и k=-inf соответственно $\|A\| = \max(|A_i|)$ и $\|A\| = \min(|A_i|)$;

» v=[3 4 -10] v = 3 4 -10	» norm(v) ans = 11.1803	» norm(v,2) ans = 11.1803	» norm(v,inf) ans = 10	» norm(v,-inf) ans = 3
	» norm(v,1) ans = 17	» norm(v,-1) ans = 1.4634	» norm(v,3) ans = 10.2946	» norm(v,'fro') ans = 11.1803

Если A – матрица, то норма определяется только для k=1, 2, inf и fro (по умолчанию k=2):

k=1 - $\|A\| = \max_j \sum_{i=1}^{m} |A_{ij}|$;

k=2 - $\|A\| = \max(\text{svd}(A))$ - максимальное из сингулярных чисел матрицы (значений квадратных корней из собственных чисел матрицы A′A);

k=inf - $\|A\| = \max_i \sum_{j=1}^{n} |A_{ij}|$; k='fro' - $\|A\| = \sqrt{\sum_{i=1}^{n} B_{ii}}$, $B=A'*A$:

A = 1 2 3 5 4 3 3 4 3	» norm(A,1) ans = 10	» norm(A) ans = 9.6871	» norm(A,inf) ans = 12	» norm(A,'fro') ans = 9.8995

Для задачи ***решения системы линейных алгебраических уравнений,*** одной из популярнейших в вычислительной математике, предусмотрены даже "элементарные" операции для подобной задачи.

Так для решения системы AX=B (A –матрица коэффициентов размерности m×n, B- матрица правых частей размерности n×k, X – матрица из k векторов-столбцов решений) можно использовать *команду обратного деления* "\". Например, для решения системы

$x_1+2\,x_2+3\,x_3 = 3$ (или 3*)*

$5\,x_1+4\,x_2+3\,x_3 = 9$ (или *9*)

$3\,x_1+4\,x_2+3\,x_3 = 6$ (или *7*)

задаем (построчно) матрицу коэффициентов и векторов правой части

» A=[1 2 3; 5 4 3; 3 4 3] A = 1 2 3 5 4 3 3 4 3	» B=[3 3; 9 9; 6 7] B = 3 3 9 9 6 7

и выполнить

```
» X=A\B        X =1.5000    1.0000
                  0.0000    1.0000
                  0.5000         0
```

При решении системы XA=B можно воспользоваться *операцией обычного деления*. Так решение той же системы

```
»X=B'/A'          X = 1.5000   -0.0000    0.5000
                      1.0000    1.0000         0
```

(обратите внимание на строчное представление решений).

Под кажущейся простотой решения скрывается достаточно серьезный анализ структуры матрицы и использование лучшего по точности и быстродействию алгоритма (метод Гаусса, разложение Холецкого и др.)

Для прямоугольной матрицы A (m≠n) решение строится по минимуму квадрата ошибки (используется QR-разложение на основе преобразований Хаусхолдера) и не сопровождается сообщениями о множественности решений или переопределенности системы:

одно уравнение с 3 неизвестными :	три уравнения с 2 неизвестными:
» a=[1 2 3]; » b=6; » x=a\b x = 0 0 2	» c=[1 2; 3 7; 2 5]; » d=[3; 10 ; 9]; » x=c\d x = -5.00000000000002 3.66666666666667

Естественно, что квадратная матрица коэффициентов должна быть невырожденной (определитель отличен от нуля) и в противном случае выдается сообщение *Matrix is singular to working precision* и элементы решения принимают значения *inf* (не определено).

Особого упоминания заслуживает **обращение (инверсия) матрицы**, для которого предусмотрена операция возведения в степень –1 и функция **inv(A)**:

» A=[1 2 3; 5 4 3; 3 4 3]	» C=inv(A)	» D=A^(-1)
A =	C =	D =
1 2 3	-0.0000 0.5000 -0.5000	-0.0000 0.5000 -0.5000
5 4 3	-0.5000 -0.5000 1.0000	-0.5000 -0.5000 1.0000
3 4 3	0.6667 0.1667 -0.5000	0.6667 0.1667 -0.5000

Напомним, что обращение матрицы может оказаться полезным при решении системы AX=B в виде $X=A^{-1}B$:

» B=[3 3; 9 9; 6 7]	» X=inv(A)*B
B = 3 3	X = 1.5000 1.0000
9 9	0 1.0000
6 7	0.5000 0.0000

При решении линейных систем и других задач интересно представление матрицы разложением на матрицы упрощенной структуры.

[L,U]=lu(A) – дает т.н. **LU-разложение** произвольной квадратной матицы в виде *произведения нижней и верхней треугольных матриц* A=LU (в матрице L возможны перестановки); такое представление позволяет, в частности, решение системы AX=B свести к двум простым системам LZ=B, UX=Z .

[L,U,P]=lu(A) – дает LU-разложение с выводом матрицы перестановок P такой, что PA=LU.

```
A =                  » [L,U]=lu(A)
 1   2   3           L =                          U =
 5   4   3             0.2000 0.7500   1.0000       5.0000  4.0000  3.0000
 3   4   3             1.0000      0        0            0  1.6000  1.2000
                       0.6000 1.0000        0            0       0  1.5000

                     » [L,U,P]=lu(A)
                     L =                          U =                      P =
                       1.0000      0        0       5.0000  4.0000  3.0000   0   1   0
                       0.6000 1.0000        0            0  1.6000  1.2000   0   0   1
                       0.2000 0.7500   1.0000            0       0  1.5000   1   0   0
```

R=chol(A), [R,p] =chol(A) - дает разложение Холецкого для положительно определенной симметрической матрицы A=R′R, где R – верхняя треугольная матрица. Если матрица A не является положительно определенной, то в первом варианте возникает сообщение об ошибке и во втором R – матрица порядка q=p-1:

```
C =                  » R=chol(C)
   1   2   3         ??? Error using ==> chol
   2   5   5         Matrix must be positive definite
   3   5   7

                     » [R,p]=chol(C)
                       R =   1   2
                             0   1
                        p =  3
```

В приведенном примере лишь первые два главных минора положительны (det(C)= -3) и, соответственно, q=2 и R′R дает второй главный минор матрицы C.

[Q,R]=qr(A), [Q,R,P]=qr(A), [Q,R]=qr(A,0) находит **QR-разложение** для прямоугольной матрицы размерности m×n:

[Q,R]=qr(A) – в виде A=QR произведения унитарной матрицы Q (Q*Q'=E) и верхней треугольной матрицы R :

```
C =                  » [Q,R]=qr(C)
   1   2   3         Q = -0.2672 -0.0514  -0.9622      R =  -3.7416 -7.2160 -9.0868
   2   5   5            -0.5345 -0.8229   0.1924            0 -1.3887 -0.3086
   3   5   7            -0.8017  0.5657   0.1924            0       0 -0.5773

t =                  » [Q,R]=qr(t)                     R =
   1   3   5           Q = -0.1961  -0.9805               -5.0990 -3.5301 -1.9611
   5   3   1              -0.9805   0.1961                     0 -2.3534 -4.7068
```

[Q,R,P]=qr(A) отличается от предыдущего упорядочением по убыванию модулей диагональных элементов R и наличием соответствующей матрицы перестановок P (A*P'=Q*R);

[Q,R]=qr(A,0) при m>n отличается тем, что вычисляются лишь n столбцов матрицы Q.

Если после выполнения QR-разложения выполнить команду **[Q1,R1]=qrdelete(Q,R,k),** то будет выполнен пересчет матриц для варианта, когда в матрице A удален k-й столбец. Если после QR-разложения выполнить команду **[Q1,R1]=qrdelete(Q,R,k,X),** то будут пересчитаны матрицы для варианта, когда в матрице A перед столбцом k вставлен столбец X.

X=nnis(A,B), X=nnis(A,B,t) позволяют искать **решение системы** AX=B **методом наименьших квадратов**, где отыскиваются *неотрицательные решения* X, минимизирующие norm(A*X-B) или гарантирующие точность ε при задании t=max(m,n)*norm(A,1)* ε.

Особое место в библиотеке занимают средства для **вычисления собственных чисел и векторов**.

В простейшем варианте отыскиваются ненулевые решения системы AX=λX командами **d=eig(A)** или **[X,d]=eig(A)** (d- диагональная матрица собственных чисел, X –матрица из нормированных собственных векторов):

```
a =                  » d=eig(a)      » [R,d]=eig(a)
   1     2     3     d =             R =                                  d =
   1     4     9       0.2179          0.8484   0.7163  -0.1198          0.2179       0       0
   1     8    27       1.8393         -0.5150   0.6563  -0.3295               0  1.8393       0
                       29.9428          0.1222  -0.2371  -0.9365               0       0  29.9428
```

Для проверки качества поиска (при значительных размерностях и многочисленных особых случаях такая проверка весьма желательна) достаточно проверить на близость к нулю значений A*X=R*D:

```
» a*R-R*d
ans =   1.0e-014 *
0.0583   0.0666  -0.7549
0.0166  -0.0444  -0.5329
0.0902  -0.3886  -0.7105
```

Функции **d=eig(A,B)** и **[V,D]= eig(A,B)** позволяют решать полную (обобщенную) проблему собственных значений AX=λBX.

Решение задачи осуществляется на основе QR-алгоритма и его модификаций и при числе итераций, превышающем 30·n, может быть прервано с сообщением *Solution will not converge* (решение не сходится).

Проблему собственных значений можно решать и для матрич-

ного полинома $(A_0+\lambda A_1+\lambda^2 A_2+...+\lambda^p A_p)X=0$ командой **[R,d]=polyeig(A_0,A_1,.., A_p),** где R- матрица размера n×(n×p) собственных векторов. При p=0 эта функция тождественна eig(A_0) , при p=1 – eig(A_0,-A_1) и в случае матриц с n=1 (скаляров) – **roots**(A_p,...,A_1,A_0), то есть ищет корни уравнения $A_p\lambda^p+...+A_2\lambda^2+A_1\lambda+A_0=0$.

Для решения ряда задач используется функция **сингулярного разложения матрицы** в формах **s=svd(A), [U,S,V]=svd(A), [U,S,V]=svd(A,0)** - матрица A размерности m×n (m≥n) представляется в виде A=U*S*V′, где U′*U=V*V′=E, S=diag(s_1,s_2,...,s_n). Здесь U состоит из n собственных векторов для n наибольших собственных значений матрицы AA′, а V – из ортонормированных собственных векторов матрицы A′A; на диагонали матрицы S - значения квадратных корней из собственных чисел матрицы A′A (*сингулярные числа*).

6. Операции над полиномами

Рассмотрим полином вида $P_n(x)=p_1x^n+p_2x^{n-1}+...+p_nx+a_{n+1}$. Соответственно будем обозначать P – $n+1$-мерный вектор коэффициентов, X – массив значений аргумента.

При **вычислении значений полинома** для элементов массива можно использовать функцию **polyval(P,X):**

» polyval([1 2 5],[0 3 1]) ans = 5 20 8	» polyval([1 2 5],[0 3 1; 1 1 1]) ans = 5 20 8 8 8 8

С помощью функции **polyvalm(P,X)** можно вычислять **значения матричного полинома** для квадратной матрицы X :

```
» polyvalm([1 2 5],[0 3 1; 1 1 1; 0 0 2])
        ans =    8    9    7
                 3   11    6
                 0    0   13
```

Умножение полиномов $C_{m+n}(x)=P_m(x)\times Q_n(x)$ выполняется командой **C=conv(P,Q) –**

$$C_k=\sum_{j=max(1,k+1-n)}^{min(k,m)} A_j\ B_{k+1-j}\ .$$

Деление полиномов можно реализовать командой **[C,R]=deconv(A,B)** , где C –частное и R – остаток от деления A на B.

» conv([1 2 3],[5 6]) ans = 5 16 27 18	» [c,r]=deconv([1 2 3],[5 6]) c = 0.2000 0.1600 r = 0 0 2.0400

Вычисление производных от полинома, произведения и отношения полиномов производится соответственно командами **dp=polyder(P), dc=polyder(A,B)** и **[f,g]=polyder(A,B)**:

» polyder([1 -2 3 4 5])	» polyder([1 2 3],[5 6])	» [f,g]=polyder([1 2 3],[5 6])
ans =	ans =	f = 5 12 -3
4 -6 6 4	15 32 27	g = 25 60 36

Вычисление корней полинома реализуется функцией **roots(P)**, а **построение полинома по его корням** – функцией **poly(R)**.

» r=roots([1 3 5 7])	» poly(r)
r = -2.1795	ans =
-0.4102 + 1.7445 i	1.0000 3.0000 5.0000 7.0000
-0.4102 - 1.7445 i	

Функция **poly(A)** обеспечивает **построение характеристического полинома** |λE-A|=0 (см. проблему собственных значений):

```
» A= magic(3)                » P=poly(A)
A =   8   1   6              P =
      3   5   7                1.0e+002 *
      4   9   2                 0.0100  -0.1500   -0.2400   3.6000
```

В приложениях, особенно связанных с преобразованием Лапласа при решении дифференциальных уравнений, оперируют с отношениями полиномов и представлениями их в виде простых дробей:

$$\frac{P_m(s)}{Q_n(s)} = \sum_{k=1}^{n} \frac{r_k}{s - s_k} + f(s),$$

где s_k - простые корни полинома $Q_n(s)$; если некоторый корень s_j имеет кратность m, то соответствующее слагаемое представляется в виде

$$\sum_{i=1}^{m} \frac{r_{j+i-1}}{(s - s_j)^i}$$

Команда **[r,s,f] =residue(P,Q)** дает **разложение отношения полиномов на простые дроби** (в случае близких корней возможна значительная погрешность). В случае кратного корня пользуются функцией **rj=resi2(P,Q,sj,m,j)**, где j –номер вычисляемого коэффициента (по умолчанию j=m); по умолчанию m=1 (простой корень).

Команда **[P,Q] =residue(r,s,f)** выполняет обратное действие свертки разложения в отношение полиномов.

Выполнив действия

[r,s,f]=residue([1 -6 11 -6],[1 -5 4 0])	» [A,B]=residue([0.5 0 -1.5],[4 1 0],[1])
r = 0.5000 0 -1.5000	A = 1 -6 11 -6
s = 4 1 0	B = 1 -5 4 0
f = 1	

видим, что

$$\frac{s^3-6s^2+11s-6}{s^3-5s^2+4s}=\frac{0.5}{s-4}+\frac{0}{s-1}-\frac{1.5}{s}+1$$

В случае кратного корня

```
» P=poly([1 2 3 ])
P =     1    -6    11    -6
» Q=poly([0 0 0 6])
Q =     1    -6     0     0     0
» [r,s,f]=residue(P,Q)
r =   0.2778   0.7222  -1.6667   1.0000
s =     6     0     0     0
f =     []
```

```
» r1=resi2(P,Q,0,3,1)
r1 =   0.7222
» r2=resi2(P,Q,0,3,2)
r2 =  -1.6667
» r3=resi2(P,Q,0,3,3)
r3 =     1
```

видим разложение:

$$\frac{s^3-6s^2+11s-6}{s^4-6s^3}=\frac{0.2778}{s-6}+\frac{0.7222}{s}-\frac{1.667}{s^2}+\frac{1}{s^3}$$

7. Коллекция тестовых матриц

Предлагаемая ниже небольшая часть коллекции тестовых матриц интересна как с позиций хотя бы дилетантского знакомства с итогами многовекового математического творчества, так и тестирования элементов собственных программных разработок.

hadamard (n) – матрица Адамара (ортогональная матрица из 1 и –1); n должно быть целым и n, n/12 или n/20 должны быть степенью 2:

```
» hadamard (4)
ans =1     1     1     1
      1    -1     1    -1
      1     1    -1    -1
      1    -1    -1     1
```

hilb (n), invhilb(n) – матрица Гильберта $h_{ij}=1/(i+j-1)$ и ей обратная (пример матрицы, плохо обусловленной :к обращению):

```
»A=hilb(3)
A = 1.0000   0.5000   0.3333
    0.5000   0.3333   0.2500
    0.3333   0.2500   0.2000
```

```
» B=invhilb(3)
B =     9   -36    30
      -36   192  -180
       30  -180   180
```

```
» det(A)
ans =
   4.6296e-004
```

magic(n) – магический квадрат (квадратная матрица с элементами от 1 до n^2 с равными суммами элементов по строкам и столбцам):

```
» magic(2)
ans =     1     3
          4     2
```

```
» magic(3)
ans =     8     1     6
          3     5     7
          4     9     2
```

pascal(n) – матрица Паскаля - симметрическая матрица из коэффициентов разложения бинома $(1+x)^j$ (треугольника Паскаля)

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

» pascal(3)	» pascal(5)
ans = 1 1 1 1 2 3 1 3 6	ans = 1 1 1 1 1 1 2 3 4 5 1 3 6 10 15 1 4 10 20 35 1 5 15 35 70

rosser - матрица Рессера (матрица 8-го порядка, служащая тестом для алгоритмов решения симметричной проблемы собственных значений):

611 196 -192 407 -8 -52 -49 29
196 899 113 -192 -71 -43 -8 -44
192 113 899 196 61 49 8 52
407 -192 196 611 8 44 59 -23
8 -71 61 8 411 -599 208 208
52 -43 49 44 -599 411 208 208
49 -8 8 59 208 208 99 -911
29 -44 52 -23 208 208 -911 99

имеет два кратных значения, три близких, нулевое и малое ненулевое):

1000 1000 1020.049 1020.000 1019.902 0.098 0 -1020.049

toepliz(X) – симметрическая матрица Теплица, определяющая перестановки элементов вектора X ;

toepliz(X,Y) – несимметрическая матрица Теплица, первый столбец которой совпадает с вектором X и первая строка с вектором Y (если $x_1 \neq y_1$, возникает конфликт на главной диагонали с предпочтением для X):

» x=[1 3 5 7]; » toeplitz(x)	» x=[1 3 5 7]; » y=[1 11 21 31]; » toeplitz(x,y)	» x=[1 3 5 7]; » y=[-1 -11 -21 -31]; » toeplitz(x,y) Column wins diagonal conflict.
ans = 1 3 5 7 3 1 3 5 5 3 1 3 7 5 3 1	ans = 1 11 21 31 3 1 11 21 5 3 1 11 7 5 3 1	ans = 1 -11 -21 -31 3 1 -11 -21 5 3 1 -11 7 5 3 1

wilkinson(n) - матрица Уилкинсона (трехдиагональная симметрическая матрица *n*-го порядка, служащая тестом для алгоритмов решения проблемы собственных значений ; обычно берут *n*=21, где возникают кратные и близкие значения);

vander(X) – матрица Вандермонда (размерность совпадает с числом *n* элементов вектора *X* $V_{ij}=x_i^{n-j}$):

```
» wilkinson(4)
ans =
  1.5000   1.0000        0        0
  1.0000   0.5000   1.0000        0
       0   1.0000   0.5000   1.0000
       0        0   1.0000   1.5000
```

```
x= [1 2 3 5]
» vander(x)
ans =     1     1     1     1
          8     4     2     1
         27     9     3     1
        125    25     5     1
```

Ряд тестовых матриц упакован в специальный подкаталог, вызываемый командой :

[<выходные параметры>]= **gallery**(‘имя матрицы’, <входные параметры>)

К их числу относятся -

cauchy(X,Y), cauchy(X) – матрица Коши с элементами $C_{ij}=1/(x_i+y_j)$: при монотонно возрастающих последовательностях X и Y матрица положительно определенная;

circul(X) – циркулянтная матрица, получается из вектора X построчно циклической перестановкой его элементов; если X скаляр, то берется вектор [1:X]; ее собственное значение равно скалярному произведениюX и вектора [$1\ t\ t^2\ \ldots\ t^{n-1}$] , где t – корень *n*-й степени из –1:

```
» x=[1 2 3 5];
»  y=[1 2 100 200];
» gallery(‘cauchy’,x,y)
ans =   0.5000   0.3333   0.0099   0.0050
        0.3333   0.2500   0.0098   0.0050
        0.2500   0.2000   0.0097   0.0049
        0.1667   0.1429   0.0095   0.0049
```

```
» gallery(‘circul’,[1 3 6 9])
ans =
     1     3     6     9
     9     1     3     6
     6     9     1     3
     3     6     9     1
```

clement(n,sym) – трехдиагональная n-мерная матрица Клемента с нулями на главной диагонали с собственными значениями из последовательности [n1, n-3 ,n-5...] с плюсом и минусом (при нечетном n добавляется 0 или 1); sym=1 определяет симметричность матрицы:

```
»gallery(‘clement’,4)
ans =
     0     1     0     0
     3     0     2     0
     0     2     0     3
     0     0     1     0
» eig(ans)
ans =   3    -3     1    -1
```

```
» gallery(‘clement’,4,1)
ans =
0         1.7320    0         0
1.7320    0         2.0000    0
0         2.0000    0         1.7320
0         0         1.7320    0
» eig(ans)
  ans =   3    -3     1    -1
```

```
» gallery(‘clement’,3,1)
ans =
     0    1.4142         0
1.4142         0    1.4142
     0    1.4142         0
» eig(ans)
ans =   2   0   -2
```

Кроме упомянутых в подкаталог входят еще свыше 40 тестовых матриц, связанных с проблемой собственных значений, обращением, полиномами Чебышева и др.

8. Анализ и обработка данных

8.1. Обработка статистических данных

Никакой анализ статистических данных не может обойтись без предварительной их обработки:

max (A) , min(A) – поиск экстремальных элементов по столбцам массива A;

max(A,B) , min(A,B) – формирование массива с элементами, равными экстремальным из соответствующих элементов массивов;

max(A,[],dim) , min(A,[],dim) – вектор экстремальных элементов по измерению dim;

[C,I]=max(...) , [C,I]=min(...) - дополнительно выводится строка индексов экстремальных элементов;

median(X) , median (X,dim) – медианы массива;

mean(X) , mean (X,dim) –средние значения;

std(X) , std(X,flag) , std(X,flag,dim) – стандартное отклонение (flag=0 –несмещенная оценка σ; flag=1 – смещенная оценка s):

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}\ ;\ s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}\ ;$$

cov(X, Y) , cov(X,Y, flag) – ковариация для массивов X и Y (каждый столбец – переменная, строка – наблюдение)

$cov_{ij}(X,Y) = \frac{1}{f} X_i^T Y_j$, *f=n или n-1;*

cov(X) , cov(X,flag) – ковариация для столбцов массива X ;

corrcoef (X) , corrcoef (X,Y) –коэффициенты корреляции:

$$R_{ij} = \frac{cov_{ij}}{\sqrt{cov_{ii}\ cov_{jj}}}\ .$$

8.2. Численное дифференцирование

Как известно, численное дифференцирование строится на использовании аппарата конечных разностей и соответствующего многообразия аппроксимаций. Здесь полезны функции:

diff(X), diff(X, n), diff(X, n, dim) – вычисление конечных разностей (первых, n-го порядка или по указанному измерению); если X

–массив, берутся разности между столбцами:

» F= [0 0.0998 0.1987 0.2955 0.3894 0.4794]
» D=diff(F)
D = 0.0998 0.0988 0.0969 0.0939 0.0900
» D2=diff(F,2)
D2 = -0.0010 -0.0020 -0.0030 -0.0039

Для задач оптимизации градиентными методами полезны функции:

gradient(F), gradient(F,h), gradient(F,h1,h2,...) –приближенная оценка градиента функции *n* переменных с автоматическим выбором шага или с указанным шагом (одинаковым или разным по переменным):

```
» [x,y]=meshgrid(-2:0.2:2, -2:0.2:2);  %   выбор сетки узлов
» z=x.*exp(-x.^2-y.^2);                %   вычисление значений на сетке
» [px,py]=gradient(z,.2,.2);           %  поиск градиента в узлах сетки
» contour(z), hold on,quiver(px,py), hold off   % рис.8.1
```

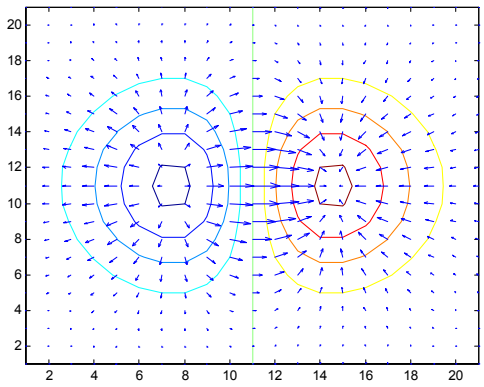


Рис. 8.1

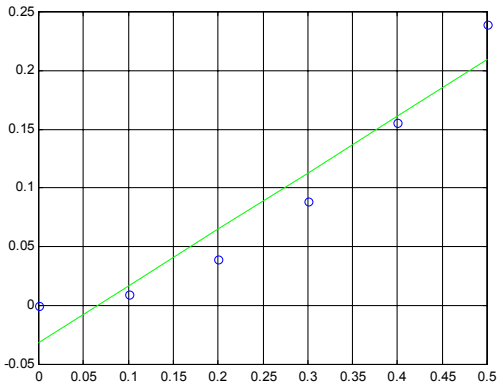


Рис. 8.2

8.3. Аппроксимация и интерполяция

polifit(X,Y,n) – аппроксимация функции Y=Y(X) полиномом n-й степени:

```
» X=0:0.1:0.5;
» F=X.*sin(X);
» P=polyfit(X,F,1)
P =   0.4814   -0.0314                    % P(x)= 0.4814 x  -0.0314
» FF=polyval(P,X)
FF =  -0.0314   0.0168   0.0649   0.1130   0.1612   0.2093
» plot(X,F,'ob', X,FF,'-g'),grid,axis([0 0.5 -0.05 0.25])      %  рис.8.2
```

interpft(Y,n,dim) -аппроксимация периодической функции на основе быстрого преобразования Фурье (Y – одномерный массив

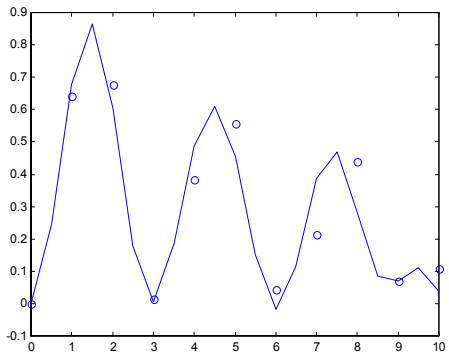


Рис. 8.3

значений функции; n – число узлов в массиве значений):

```
» X=0:10;
» Y=sin(X).^2.*exp(-0.1.*X);
» YP=interpft(Y,21);
» xp=0:0.5:10;
» plot(X,Y,'ob', xp,YP) % Рис. 8.3
```

Заметим, что в библиотеке имеется богатый ассортимент средств для преобразования Фурье.

spline(X,Y,Z) – интерполяция Y=Y(X) кубическим сплайном и вывод соответствующих значений в точках Z. Для получения большей информации используется конструкция **pp=spline(X,Y):** здесь командой **V=ppval(pp,Z)** можно найти значения в точках Z, а командой **[Xs, Coef, m,L]=unmkpp(pp)** получить данные о векторе разбиений аргумента Xs, коэффициентах Coef, m=length(Xs), L=length(Coef)/m .

interp1(X,Y,Z), interp1(X,Y,Z,'method') – одномерная табличная интерполяция (если Y двумерный массив, интерполяция ведется по каждому столбцу; значения Z должны входить в диапазон значений X). Можно указать метод интерполяции – кусочно-линейной (linear, по умолчанию), ступенчатой (*nearest*), кубической (*cubic*), кубическими сплайнами (*spline*). Функция **interp1q(X,Y,Z)** реализует быструю линейную интерполяцию на неравномерной сетке.

interp2(X1,X2,Y,Z1,Z2), interp1(X1,X2,Y,Z1,Z2,'method') – двумерная табличная интерполяция Y=Y(X1,X2), аргументы должны

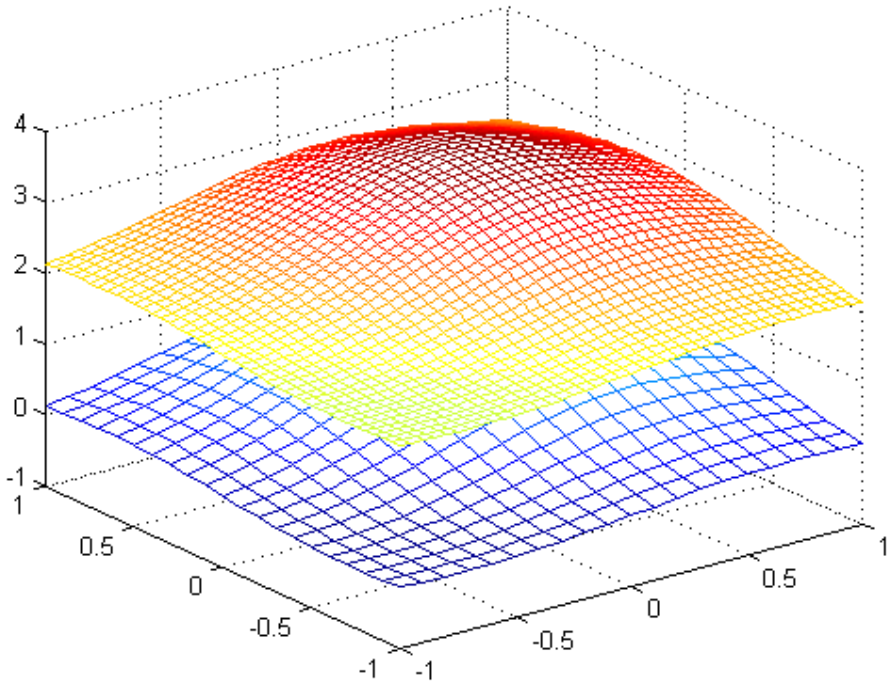


Рис. 8.4

меняться монотонно и заданы в формате функции meshgrid.

```
» [X1,X2]=meshgrid(-1:0.1:1);
» Y=exp(-X1.^2-X2.^2).*(1+X1+X2);
» [Z1,Z2]=meshgrid(-1:0.05:1);
» Y2=interp2(X1,X2,Y,Z1,Z2);
» mesh(X1,X2,Y),hold on,mesh(Z1,Z2,Y2+2),hold off      % Рис. 8.4
```

interp3(X1,X2,X3,Y,Z1,Z2,Z3), interp3(..., 'method') – трехмерная табличная интерполяция Y=Y(X1,X2,X3);

interpn(X1,X2,...,Y,Z1,Z2,...), interp3(..., 'method') – многомерная табличная интерполяция Y=Y(X1,X2,...);

griddata(X1,X2,Y,Z1,Z2), griddata(X1,X2,Y,Z1,Z2, 'method') - двумерная табличная интерполяция на неравномерной сетке.

8.4. Численное интегрирование

polyarea (X,Y), polyarea (X,Y, dim) -площадь многоугольника с координатами вершин (X,Y):

```
» polyarea ([1 2 3 4 5],[0 3 6 3 0])
ans =    12
```

trapz(X,Y), trapz(X,Y, dim) – вычисление интеграла по формуле трапеций ; функции **cumtrapz(X,Y), cumtrapz(X,Y, dim)** вычисляют к тому же промежуточные результаты;

quad('имя'**, a,b), quad('**имя'**, a,b, eps), quad('**имя'**, a,b, eps, trace), quad('**имя'**, a,b, eps, trace,P1,P2,...), quad8**(...) - вычисление определенного интеграла: **a,b** – пределы интегрирования; **eps** – относительная погрешность (по умолчанию 10^{-3}); **trace** –построение точечного графика функции; **'**имя' – имя подинтегральной функции (встроенной или М-файла); **P1,P2,**... – передаваемые параметры функции; **quad** использует квадратуру Симпсона, **quad8** – Ньютона-Котеса 8-го порядка). Используется рекурсия до глубины 10 и может появиться сообщение – подозрение о сингулярности функции (наличии особенностей).

```
function f=integr(x)
        f=x.*exp(-x);
```

```
» quad('integr',0,2, 1e-5,1)
ans =    0.5940
```

dblquad('имя'**, a1,b1, a2,b2), dblquad('**имя'**, a1,b1, a2,b2, eps), dblquad('**имя'**, a1,b1, a2,b2, eps,** <метод>**)** - вычисление двойного интеграла от функции <имя>(X1,X2) ; <метод> **- quad** или **quad8.**

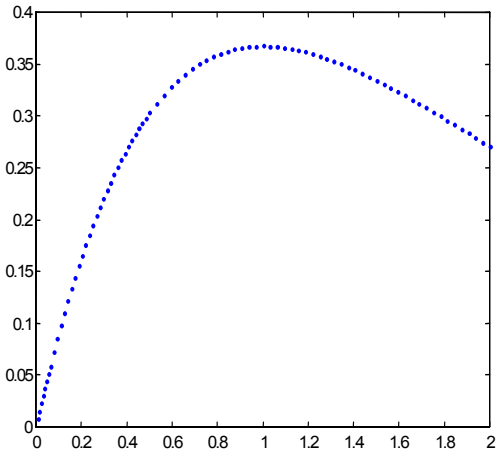


Рис. 8.5

8.5. Нули и экстремумы функций

Как известно, решение уравнений и поиск экстремумов функций – родственные задачи. Так решение системы $f_i(X)=0,\ i=1,...,n$ можно заменить поиском минимума $F(X)=\sum_{i=1}^{n} f_i^2(X)$, заведомо равного нулю, если система имеет решение. С другой стороны, поиск экстремумов функции можно свести к решению системы уравнений относительно нулевых значений ее производных.

Для некоторых классов функций имеются вполне универсальные методы решения: так для полиномов поиск корней реализуется без каких-то условий обобщенным методом Ньютона - функцией roots(...) . В общем случае при великом многообразии методов решение подобных задач отнюдь нетривиально. В системе реализованы функции лишь для простейших задач без гарантий получения решения.

xmin=fmin(‘имя**’,a,b), xmin=fmin(‘**имя**’,a,b,<**опции**>), xmin= fmin (‘**имя**’, a,b,<**опции**>, p1,...,p10), [xmin,options]=fmin(...)** – поиск минимума функции одной переменной на интервале [a,b]; в списке <опции> можно указать option(1)=1 (вывод промежуточных результатов), option(2)- погрешность итераций для аргумента (разница между смежными приближениями; по умолчанию 10^{-4}) и option(14)- максимальное число итераций (по умолчанию 500). Используется метод золотого сечения и параболической интерполяции.

```
function r=m1(x)
r=x^2-x-3;
» xm=fmin('m1',-2,3,[1, 1e-6])
      Func  evals     x         f(x)       Procedure
         1    -0.0901699    -2.9017       initial
         2       1.09017    -2.9017       golden
         3       1.81966    -1.5085       golden
         4           0.5        -3.25      parabolic
         5           0.5        -3.25      parabolic
         6           0.5        -3.25      parabolic
    xm =
       0.5000
```

xmin=fmins(‘имя**’,x0), xmin=fmins(‘**имя**’,x0,<**опции**>), xmin= fmins (‘**имя**’, x0,<**опции**>** или **[], p1,...,p10), [xmin,options]=fmins(...)** – поиск минимума функции нескольких переменных: x0 – начальное приближение; p1,...,p10 – дополнительные параметры; в списке <опции> можно указать option(1)=1 (вывод промежуточных результа-

тов), option(2)- погрешность итераций для аргумента (разница между смежными приближениями; по умолчанию 10^{-4}), option(3)- итерационная погрешность для функции и option(14)- максимальное число итераций (по умолчанию 200×n).

```
function r=m2(x)
r=(x(1)-1)^2+(x(2)-3)^2;
» [xmin,opt]=fmins('m2',[0.5 2.5])
xmin =
1.0000        3.0000
opt =
 0   0.0001   0.0001   0.0000        0        0        0        0.0000   0
        72         0         0         0      400         0   0.0000   0.1000   0
```

(возвращаемые здесь опции 8 и 10 определяют минимум функции и число итераций). Используется метод Нелдера-Мида (строится симплекс из n+1 вершины в n –мерном пространстве, берется новая точка внутри или вблизи симплекса и может заменить одну из вершин; процесс повторяется до малого диаметра симплекса).

fzero('имя',x0), fzero('имя',x0,eps), fzero ('имя', x0, eps , trace), fzero ('имя', x0, eps, trace,p1,...,p10) – поиск действительных корней функции одной переменной при начальном приближении **x0** (можно взять и в форме [a,b] при условии f(a)×f(b)<0) с заданной относительной погрешностью; trace=1 – вывод промежуточных результатов. Используется метод дихотомии, хорд и обратной квадратической интерполяции. При поиске корней полинома см. **roots**.

Процесс поиска корня виден из примера:

```
» fzero('sin',6,[],1)
    Func evals      x          f(x)          Procedure
            1           6        -0.279415        initial
            2     5.83029     -0.437567        search
            3     6.16971     -0.113236        search
            4        5.76     -0.499642        search
            5        6.24     -0.0431719       search
            6     5.66059      -0.583146       search
            7     6.33941       0.0561963      search
    Looking for a zero in the interval [5.6606, 6.3394]
            8     6.27974    -0.00344052       interpolation
            9     6.28319     1.70244e-006     interpolation
           10     6.28319    -3.35667e-012     interpolation
           11     6.28319    -2.44921e-016     interpolation
           12     6.28319     2.41961e-015     interpolation
    ans =   6.28318530717959
» 2*pi
    ans =   6.28318530717959
```

Заметим, что в случае двух переменных существенную помощь для выбора начального приближения может оказать функция gradient (см. 8.2) и contour(...), которая будет рассмотрена при ознакомлении с функциями графики.

8.6. Обыкновенные дифференциальные уравнения

В системе MatLab предусмотрены специальные средства решения задачи Коши для систем обыкновенных дифференциальных уравнений, заданных как в явной форме $\frac{dx}{dt} = F(t,x)$, так и в неявной $M\frac{dx}{dt} = F(t,x)$, где *M*- матрица, - т.н. *решатель ОДУ* (solver ODE), обеспечивающий пользователю возможность выбора метода, задания начальных условий и др.

В простейшем варианте достаточно воспользоваться командой [T,X]=solver('F', [DT], X0, ...), где DT - диапазон интегрирования, X0 – вектор начальных значений, F – имя функции вычисления правых частей системы, solver – имя используемой функции (**ode45** - метод Рунге-Кутта 4 и 5-го порядков, **ode23** – тот же метод 2 и 3-го порядков, **ode113** – метод Адамса - для нежестких систем, **ode23s, ode15s** – для жестких систем и др.). Версии решателя различаются используемыми методами (по умолчанию относительная погрешность 10^{-3} и абсолютная 10^{-6}) и соответственно временем и успешностью решения. Под жесткостью здесь понимается повышенное требование к точности – использование минимального шага во всей области интегрирования. При отсутствии информации о жесткости рекомендуется попытаться получить решение посредством ode45 и затем ode15s. Если диапазон DT задан начальным и конечным значением [t_o , t_k], то количество элементов в массиве T (и в массиве решений X) определяется необходимым для обеспечения точности шагом; при задании DT в виде [t_o , t_1, t_2, ... , t_k] или [t_o :Δt: t_k] - указанными значениями.

Например, в простейшем варианте решение уравнения $\frac{dx}{dt} = t \cdot e^{-t}$ в интервале t ∈[0, 0.5] с начальным условием x(t=0)=1 :

```
      function  f  =odu1(t,x)
                f=t*exp(-t);
      » [T,X]=ode45 ('odu1', [0, 0.5], 1)
T =    0        0.0125   0.0250   0.0375   0.0500   0.0625   0.0750   0.0875
       0.1000   0.1125   0.1250   0.1375   0.1500   0.1625   0.1750   …
X =    1.0000   1.0000   1.0003   1.0006   1.0012   1.0018   1.0027   1.0036
       1.0046   1.0058   1.0072   1.0086   1.0102   1.0118   1.0136   ….
```

Для иллюстрации решения системы и ряда нестандартных возможностей рассмотрим *задачу выравнивания цен* по уровню актива в следующей постановке.

Предположим, что изменение уровня актива y пропорционально разности между предложением s и спросом p, т.е. $y'=k(s-d),\ k>0$, и что изменение цены z пропорционально отклонению актива y от некоторого уровня y_0, т.е. $z'=-m(y-y_0),\ m>0$. Естественно, что предложение и спрос зависят от цены, например, $s(z)=az+s_0$, $d(z)= d_0 - cz$. Соответственно возникает система дифференциальных уравнений

$$y' = k\cdot(s(z)-d(z))$$
$$z' = - m\cdot (y-y_0)\ .$$

Вычисление правых частей оформляем функцией:

```
function f=odu2(t,X)
   y=X(1);     z=X(2);
   a=20; c=10;  s0=10; d0=50; k=0.3; m=0.1;
   s=a*z+s0; d= d0-c*z;     y0=19;
   f(1)=k*(s-d) ; f(2)=-m*(y-y0);
   f=f';                  % вектор-столбец
```

Если выполнить решение при $y_0= 19,\ z_0=2$

```
» [T,Y]=ode45('odu2', [0:0.3:9],[19  2]);
» [T Y]
» plot(T,Y)
```

будет выведена таблица значений искомых функций:

```
ans =         0   19.0000   2.0000
         0.3000   20.7757   1.9731
         0.6000   22.4101   1.8949
         0.9000   23.7686   1.7714
         1.2000   24.7427   1.6126
         1.5000   25.2569   1.4313 ...
```

и их графики (рис. 8.6).

Эта система уравнений относится к числу т.н. *автономных* (или *динамических*), ибо независимая переменная в нее явно не входит; соответственно может быть установлена связь между найденными решениями: в параметрическом задании линия $y=y(t),\ z=z(t)$ определяет *фазовую кривую* (*траекторию*) системы - гладкую кривую без самопересечений, замкнутую кривую или точку, которая позволяет судить об устойчивости системы.

Так, установив опции к построению двумерного фазового портрета (функция `odephas2`) и номера переменных состояния :

```
» opt=odeset('OutputSel',[1 2], 'OutputFcn','odephas2');
» [T,Y]=ode45('odu2', [0:0.3:9],[19  2],opt);
```

получаем фазовый портрет системы, свидетельствующий о ее устойчивости – гармонии между активом и ценами (рис.8.7).

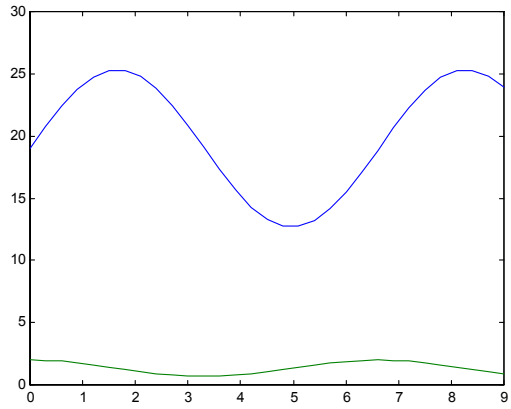


Рис.8.6

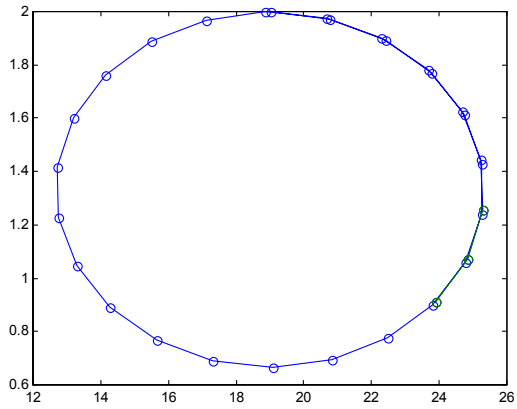


Рис.8.7

В качестве другого примера подобных задач рассмотрим известную *задачу динамики популяций*, где рассматривается модель взаимодействия “жертв” и “хищников”, в которой учитывается уменьшение численности представителей одной стороны с ростом численности другой. Модель была создана для биологических систем, но с определенными корректурами применима к конкуренции фирм, строительству финансовых пирамид, росту народонаселения, экологической проблематике и др.

Эта модель Вольтерра-Лотка с логистической поправкой описывается системой уравнений

$$\frac{dx_1}{dt} = (a - bx_2)x_1 - \alpha x_1^2$$

$$\frac{dx_2}{dt} = (-c + dx_1)x_2 - \alpha x_2^2$$

с условиями заданной численности “жертв” и “хищников” в начальный момент $t=0$.

Решая эту задачу при различных значениях α, получаем различные фазовые портреты (обычный колебательный процесс и постепенная гибель популяций) .

```
function f=VolterraLog(t,x)
a=4;
b=2.5;
c=2;
d=1;
alpha=0.1;
f(1)= (a-b*x(2))*x(1)-alpha*x(1)^2;
f(2)= (-c+d*x(1))*x(2)-alpha*x(2)^2;
f=f';
```

```
» opt=odeset('OutputSel',[1 2], 'OutputFcn', 'odephas2');
» [T,X]=ode45('VolterraLog', [0  10],[3  1],opt );
```

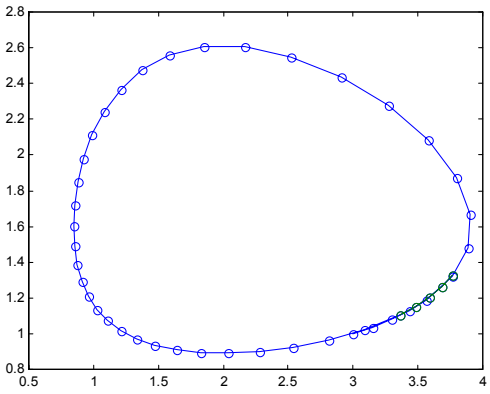


Рис.8.8 (α=0)

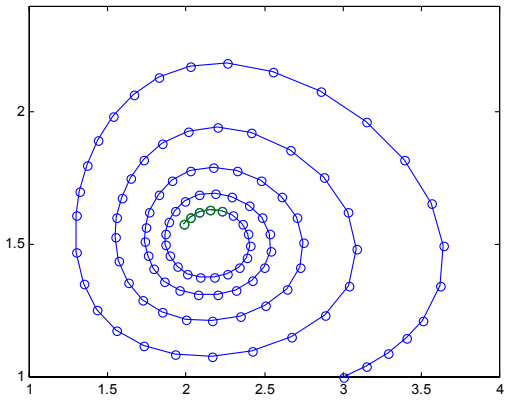


Рис.8.9 (α=0.1)

Имеется возможность построения и трехмерного фазового портрета с помощью функции odephas3. Например, решение задачи Эйлера свободного движения твердого тела:

$$\frac{dx_1}{dt} = x_2 x_3, \quad \frac{dx_2}{dt} = -x_1 x_3, \quad \frac{dx_3}{dt} = -0.51 \cdot x1 x_2;$$

$$x_1(0) = x_2(0) = x_3(0) = 0$$

выступает в виде:

```
function f=Eiler(t,x)
f(1)=   x(2)*x(3);
f(2)=   -x(1)*x(3)  ;
f(3)=   -0.51*x(1)*x(2)  ;
f=f';
```

```
» opt=odeset('OutputSel',[1 2 3], 'OutputFcn', 'odephas3');
» [T,X]=ode45('Eiler', [0  7.25], [0  0 1], opt ); % рис.8.10
» [T,X]=ode45('Eiler', [0:0.25:7.25],[0 1  1]);
» plot(T,X)                                          % рис.8.11
```

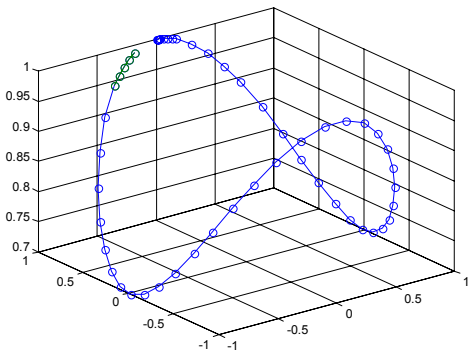


Рис.8.10

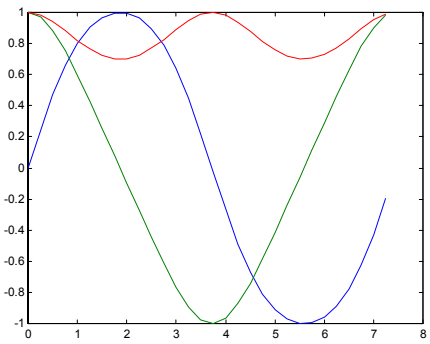


Рис.8.11

9. Элементарная графика

Здесь мы рассмотрим лишь небольшую часть графических команд высокого уровня, не затрагивая их базу - графические объекты (Axes, Line, Patch, Surface, Text).

9.1. Двумерная графика

Графика в линейном масштабе

plot (y) - построение графика одномерного массива в зависимости от номера элемента (для двумерного массива строятся графики для столбцов);

plot (x,y) - построение графика функции *y=y(x)* ; при двумерном *x* строятся графики *x=x(y)*; если оба массива двумерные, строятся зависимости для соответствующих столбцов;

plot (x,y, LineSpec) – заданием строки LineSpec (до 3 символов) определяет стиль линий , форму маркера точек и цвет линий и маркера:

Символ стиля линии		Цвет		Цвет	
Непрерывная	-	Желтый	y	Зеленый	g
Штриховая	--	Фиолетовый	m	Синий	b
Двойной пунктир	:	Голубой	c	Белый	w
Штрихпунктирная	-.	Красный	r	Черный	k

Маркер может определяться символами :

. **+** ***** **°** **×** **s** (квадрат) **d** (ромб) **p** (пятиугольник) **h**(шестиугольник)

v **^** **<** **>** (стрелки)

По умолчанию выбирается непрерывная линия с точечным маркером и чередованием цветов с желтого по синий.

plot (x1,y1, LineSpec1, x1,y1, LineSpec2,...) – строит на одном графике несколько линий (диапазон по аргументу - объединение x1 и x2;

plot (...,'PropertyName',PropertyValue,...) –задает значения свойств графического объекта Line (толщину линий LineWidth, размер маркера MarcerSize, цвет маркера MarcerFaceColor и др.) .

```
» x=0:0.3:6;
» y=besselj(0,x);                          % функция J0(x)
» x1=0:0.4:8;
» y1=besselj(1,x1);                        % функция J1(x)
» plot(x,y,'-sk', x1,y1,'-pk','LineWidth',1 )   % рис.9.1
```

Построение графиков функций

fplot(<имя функции**>,limits)** строит график функции (функций) в интервале limits=[xmin,xmax]. В качестве имени функции может использоваться М-файл или строка типа 'sin(x)', '[sin(x) cos(x)]', '[sin(x), myfun1(x), myfun2(x)]'. Можно установить размеры графика по оси значений функ-

ции limits=[xmin,xmax ymin ymax].

fplot(<имя функции>,limits, eps) строит график с относительной погрешностью eps (по умолчанию 0.002) и максимальное число шагов (1/eps)+1. Эту конструкцию можно дополнить четвертым параметром **n** (n+1 – минимальное число точек) и параметром LineSpec:

» fplot('[besselj(0,x) besselj(1,x) 0]',[0 10],[],20) & рис.9.2

ezplot('f(x)') строит график f(x), заданной символьным выражением (например, ezplot('x^2-2*x+1')), на интервале [-2π 2π] с выводом выражения в качестве заголовка графика.

ezplot('f(x)', limits) и **ezplot('f(x)', limits, fig)** строят график f(x) на указанном интервале и в заданном окне.

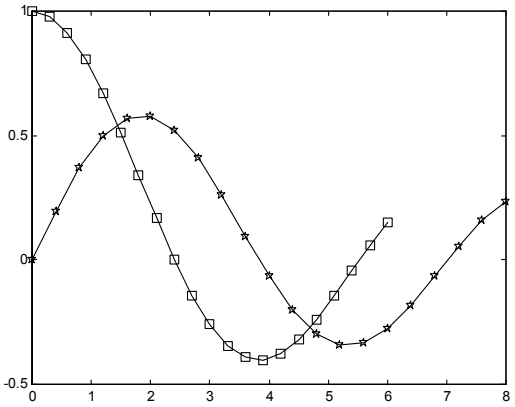


Рис.9.1

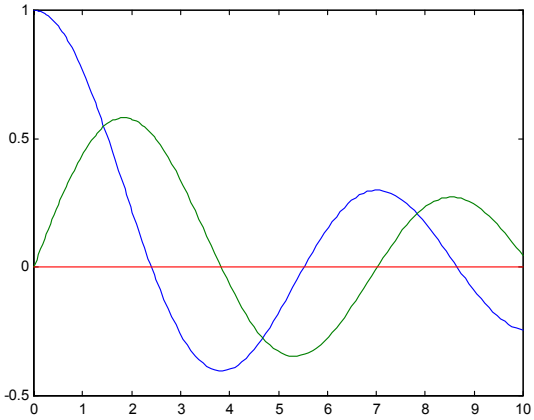


Рис.9.2

График в полярных координатах определяется функциями **polar(f,r)** и **polar(f,r, LineSpec)**, где **f** – массив значений угла и **r** – соответствующие значения радиуса : $x=r\cdot\cos(\varphi)$, $y=r\cdot\sin(\varphi)$:

```
» f=0:0.01:2*pi;                    » f=0:0.01:12*pi;
» r=sin(2.*f).*cos(2.*f);           » r=exp(-0.1*f);
» hp=polar(f,r),hold on             » hp=polar(f,r) ,hold on
» set(hp,'LineWidth',4) % рис.9.3   » set(hp,'LineWidth',2)  % рис.9.4
```

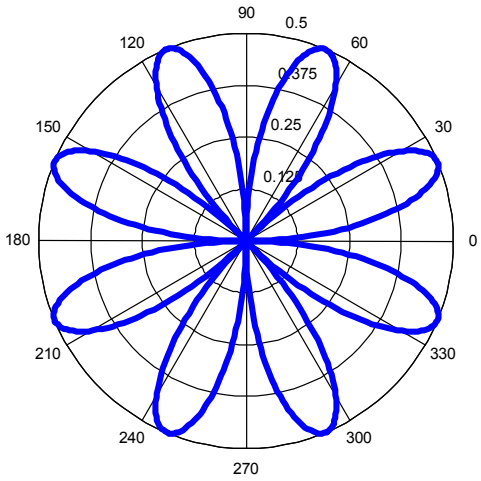


Рис.9.3

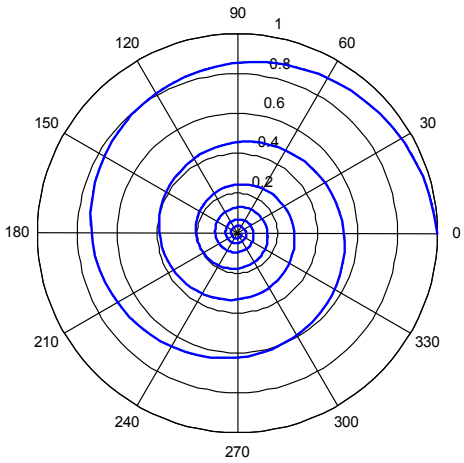


Рис.9.4

График в логарифмическом масштабе задается функцией **loglog** с тем же набором параметров, что и **plot**, с той лишь разницей, что проводится масштабирование десятичным логарифмированием по обеим координатам.

График в полулогарифмическом масштабе задается функциями **semilogx** и **semilogy** с тем же набором параметров, что и **plot** (проводится масштабирование логарифмированием по одной из координат).

График с двумя осями ординат (одна отображается слева, другая справа) реализуется функцией **plotyy(x1,y1,x2,y2)** и той же функцией с добавлением параметров масштабирования ‘f1’ или ‘f1’,’f2’, в роли которых могут выступать plot, semilogx, semilogx, loglog:

» x=0:0.01:12*pi;
» plotyy(x,sin(x).*exp(-0.1.*x),x, 10*exp(-0.1.*x)) % Рис.9.5

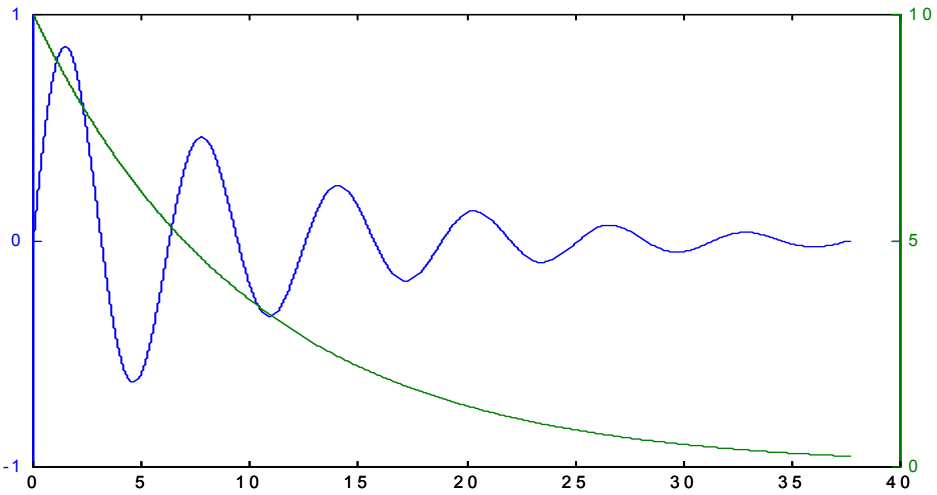


Рис.9.5

9.2. Трехмерная графика

В трехмерной графике выполняются представления функции z=z(x,y), отличающиеся способом соединения точек: линия, сечения, сетчатая или сплошная поверхность.

plot3(x,y,z) в тех же вариациях, что и plot, предполагает задание одномерных и двумерных массивов (строятся точки с координатами x(i,:),y(i,:),z(i,:) для каждого столбца и соединяются прямыми линиями. Если используется **[x,y]=meshgrid(...)**, то строятся сечения.

» t=0:pi/50:10*pi;
» plot3(sin(t),cos(t),t) %Рис.9.6

» [x,y]=meshgrid([-2:0.1:2],[-2:0.01:2]);
» z=exp(-x.^2-y.^2);
» plot3(x,y,z) %Рис.9.7

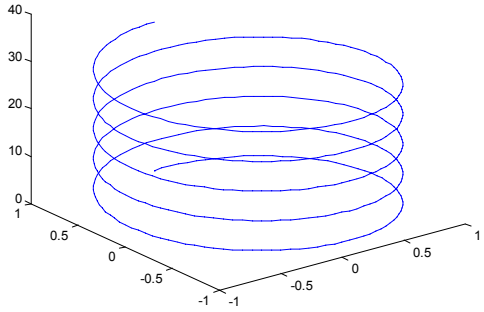


Рис.9.6

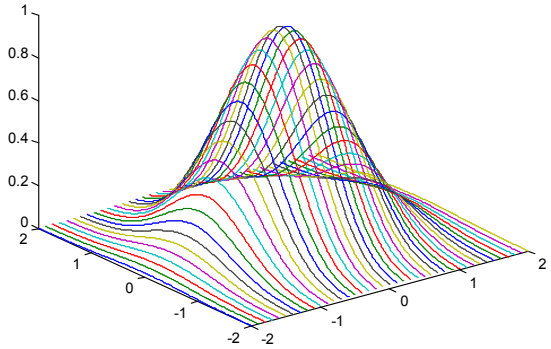


Рис.9.7

mesh(x,y,z,c), mesh(z,c), mesh(z) определяют задание сетчатой поверхности (массив **c** определяет цвета узлов поверхности; если **x,y** не указаны, то x=1:n, y=1:m, где [m,n]=size(z).

```
» [x,y]=meshgrid(-8:0.5:8);
» t=sqrt(x.^2+y.^2)+0.001;
» z=sin(t)./t;
» mesh(x,y,z)     %Рис.9.8
```

Аналогичная функция **meshc** в дополнение к поверхности строит проекции линий уровня, а **meshz** делает срез поверхности до нулевого уровня (своеобразный пьедестал).

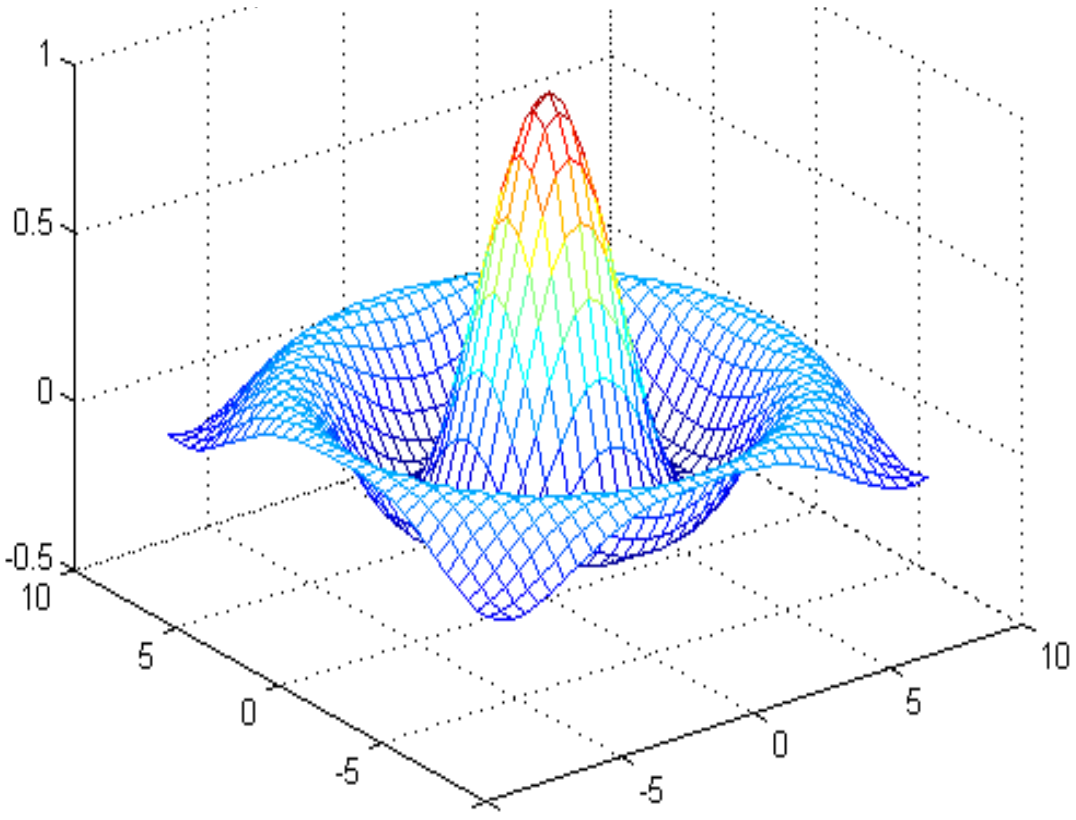


Рис.9.8

» meshc(x,y,z) %Рис.9.9

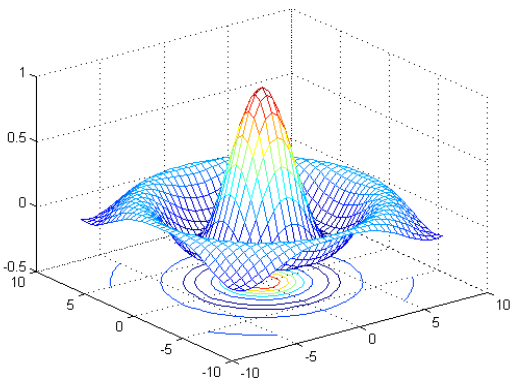


Рис.9.9

» meshz(x,y,z) %Рис.9.10

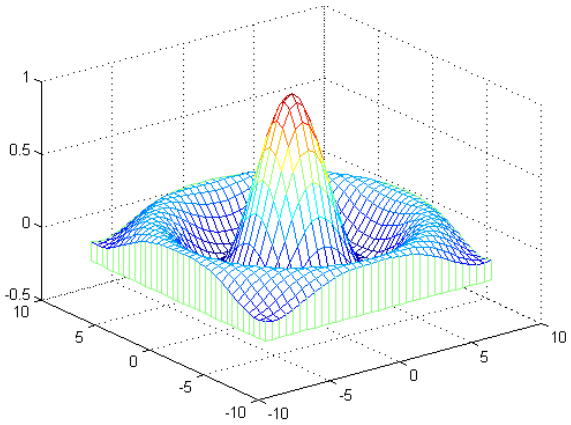


Рис.9.10

surf(x,y,z,c), surf(z,c), surf(z) определяют задание сплошной поверхности, отличаясь от mesh системой окраски; аналогичная функция **surfc(...)** задает проекции линий уровня.

Реализация трехмерной графики может сопровождаться множеством вспомогательных команд, например:

hidden on/off включает или выключает режим удаления невидимых линий (по умолчанию on);

shading faceted / **flat** / **interp** устанавливает затенение поверхностей (по умолчанию faceted дает равномерную окраску ячеек с черными гранями, flat – цветами узлов сетки, interp – интерполяцией цветов.

9.3. Задание осей координат

Создание графического объекта исходит автоматически при обращении к командам, порождающим объекты Line и Surface, но может выполняться и командой **axec**(‘<имя свойства>’,<значение>, ...). Есть и команды более высокого уровня:

axis([xmin xmax ymin ymax]), axis([xmin xmax ymin ymax zmin zmax]) устанавливает масштаб по осям;

axes off / on выключает (включает) вывод на координатные оси обозначений и маркеров;

grid on/off , grid включает (выключает) или переключает режим нанесения координатной сетки на осях;

box on/off, box включает (выключает) или переключает режим рисования контура параллелепипеда, трехмерный объект;

zoom on/off включает (выключает) режим интерактивного масштабирования графиков (левая мышь около точки увеличивает масштаб вдвое, правая – уменьшает; удержанием левой мыши можно выделить прямоугольную область для детального просмотра; **zoom out** восстанавливает исходный график.

9.4. Линии уровня

В отличие от **meshc (...)** и **surfc(...)** функция **contour** рисует только линии уровня соответствующих поверхностей и выступает в многообразии синтаксических форм: **contour(X,Y,Z) -** для массива Z =Z(X,Y), **contour(X,Y,Z,n) –** то же с указанием числа линий уровня (по умолчанию 10), **contour(X,Y,Z,v) –** то же для массива указанных значений ; **contour(Z), contour(Z,n), contour(Z,v)** – аналогичные функции без указания диапазонов для аргументов и **contour(...,LineSpec) -** аналогичные функции с указанием типа и цвета линий (см. plot); **[C,h]=contour (...)** возвращает массив С и вектор дескрипторов, позволяя тем самым продолжить работу с рисунком (давать оцифровку линий, заголовки и др.).

Функция **contourf(...)** закрашивает области между линиями уровня, аналогична contourf(...) с разницей в формате**[C,h, cf]=contour (...),** где cf определяет матрицу раскраски.

```
» [x,y]=meshgrid(-8:0.5:8);
» t=sqrt(x.^2+y.^2)+0.001;
» z=sin(t).^3./t;
» [c,h]=contour(x,y,z,20);
```

```
» [x,y]=meshgrid(-2:0.25:2);
» t=sqrt(x.^2+y.^2)+0.001;
» z=sin(t).^3./t;
» [c,h,cf]=contourf(x,y,z,4);
```

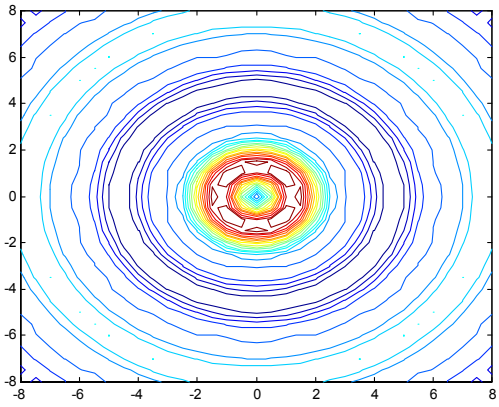


Рис.9.11

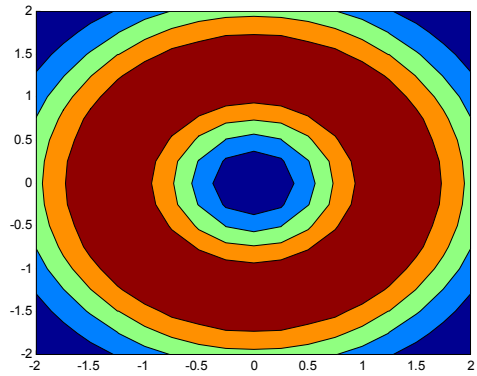


Рис.9.12

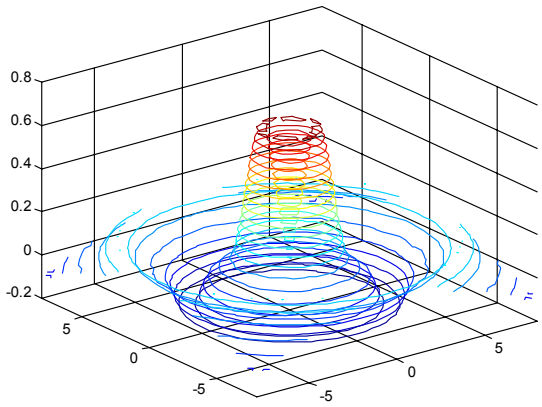


Рис.9.13

Функция **contour3(...)** по синтаксису полностью аналогична contour(...), но изображает не проекции линий уровня, а рисует их в пространственной интерпретации; так команда [c,h]=contour3(x,y,z,20); дает фигуру (рис.9.13).

9.5. Дополнительные возможности

Создание нового графического окна **figure** ; командой **figure(n)** можно выбирать некоторое из созданных окон в качестве текущего.

Включение (выключение) режима сохранения текущего графика : **hold on/off, hold** .

Вывод заголовков для графиков (в текущем окне):

title('текст'), title(<имя функции-строки>'), title(..., 'PropertyName','PropertyValue',...), h=title(...).

Вывод графиков в нескольких окнах рисунка: **subplot(m,n,k), subplot(mnk)** – m – число окон по горизонтали, n – по вертикали, k – номер окна:

```
» subplot(121)
» plot([1:0.3:4])
» subplot(122)
» plot([4:-0.3:1])
» title ('y=4-0.3(x-1)')
```

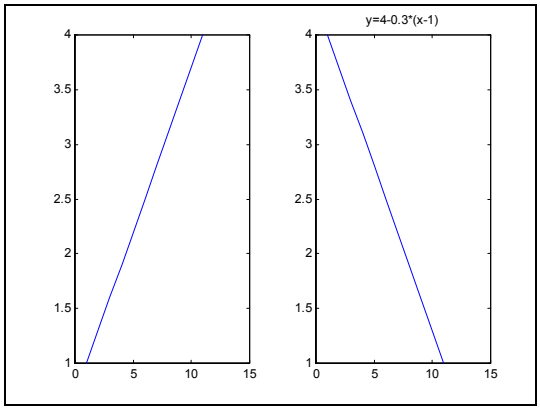


Рис.9.14

Вывод текста для обозначения координатной оси : **xlabel(...), ylabel(...), zlabel(...) –** синтаксис аналогичен title(...).

Вывод текста в указанной позиции графика: **text(x,y,'текст'), text(x,y,z,'текст'), text(...'PropertyName','PropertyValue',...), h=text(...).)** - x,y,z –координаты начала текста.

Вывод текста под управлением мыши: **gtext('текст'), h= gtext('текст') –** выведенный текст можно перемещать мышью.

Вывод легенды **legend('текст1','текст2',...), legend(M), legend(h,M), legend off, legend(...,pos), h= legend(...)** – здесь M – строковый массив (длина строк одинакова), off удаляет пояснения к графику, pos определяет позицию легенды (-1 - справа от графика, 0 – в одном из 4 углов с минимумом потерь точек графика, 1-4 – в указанном углу, [x y] – в указанном месте); можно перетаскивать легенду мышью.

```
» subplot(111)
» t=[0:pi/30:2*pi];
» a=sin(t); b=cos(t);
» x=0:60;
» plot(x,a+b),hold on
» hp=plot(x,a', 'g', x,b','r'); set(hp,'LineWidth',2)
» legend('a+b','a=sin(t)','b=cos(t)')
» title('y=sin(t)+cos(t)','FontSize',12,'FontWeight','bold')
```

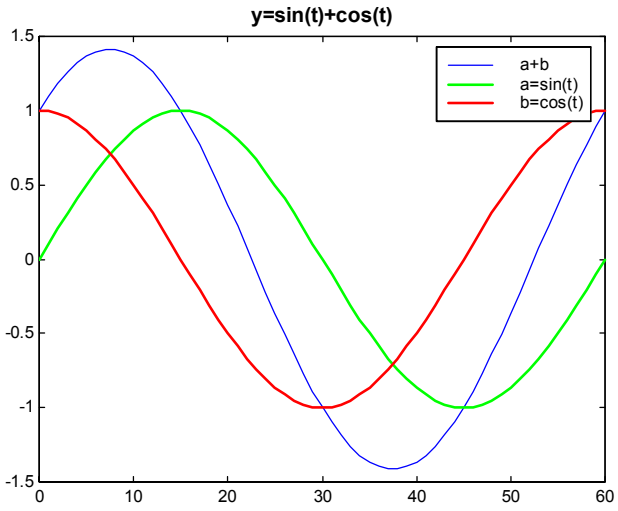


Рис.9.15

Маркировка линий уровня, создаваемых функциями contour, contour3, contourf **: clabel(C,h), clabel(C,h,v), clabel(C,h,'manual'), clabel(C), clabel(C,v), clabel(C,'manual')** – при наличии **h** маркировка на линиях, при наличии **'manual' -** принудительная маркировка нажатием левой мыши или пробела (правая мышь или Return завершает маркировку).

9.6. Специальная графика

Раздел специальной графики включает команды для построения диаграмм, гистограмм и прочих дискретных графиков.

Столбцовые диаграммы реализуются функциями **bar** и **barh**:

bar(y), bar(x,y) , h=bar(...) – здесь **y** –массив (одно- или двумерный), **x** – одномерный, упорядоченный по возрастанию массив (число смежных по горизонтали столбцов диаграммы равно числу столбцов массива **y**); можно указать параметры относительной ширины столбцов (1 – касание, >1 – перекрытие, <1 – с промежутками), или стиля ('group','stack') :

```
» x=0:0.1:6;
» y1=sin(x); y2=cos(x); y3=exp(-x./2);
» y=[y1;y2;y3];
```

```
» x=0:0.1:6;
» y1=sin(x); y2=cos(x); y3=exp(-x./2);
» y=[y1;y2;y3];
```

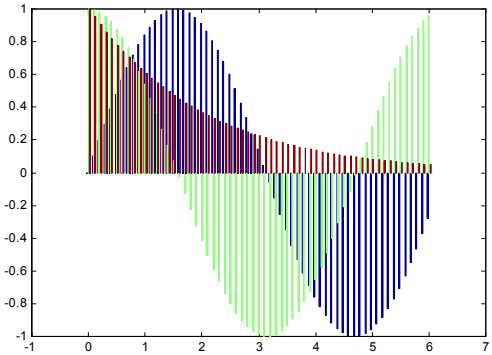


Рис.9.16

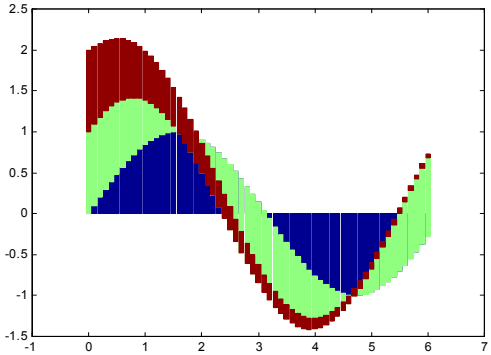


Рис.9.17

» bar(x,y') » bar(x,y',’stack’)

barh(...) отличается лишь размещением столбцов не по вертикали, а по горизонтали.

Секторная диаграмма реализуется функцией **pie(x), pi(x,v), h=pie(...)** – здесь v – вектор из 0 и 1 для отделения от диаграммы отдельных секторов:

» x=[1 4 0.5 5.5 2];
» pie(x,[0 1 0 0 0]) % Рис.9.18

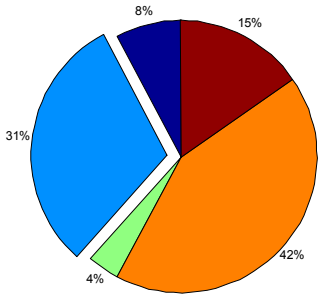


Рис.9.18

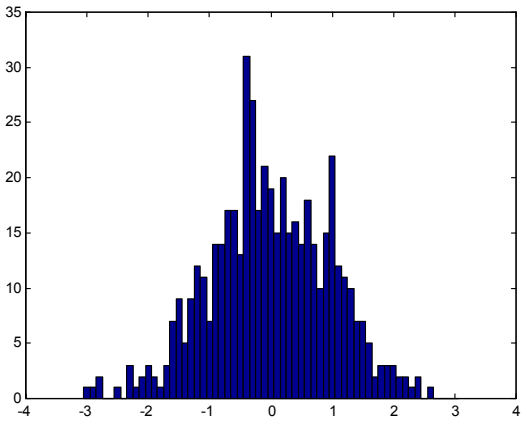


Рис.9.19

Построение гистограммы **hist(y), hist(y,x), hist(y,n), [p,x]=hist(y,...)** реализует подсчет числа элементов по столбцам массива **y** в **n** (по умолчанию 10) интервалах:

» x=-3:0.1:3;
» t=randn(500,1);
» hist(t,x) % Рис.9.19

Дискретный график **stem(y), stem(x,y), stem(...,’fill’), stem(...,LineSpec), h=stem(...)** аналогичен столбцовой диаграмме и выводит значения в виде отрезков с маркером (**‘fill’** –закраска маркера):

» x=-3:0.1:3;
» f=exp(-x.^2/2);
» stem(x,f) %Рис.9.20

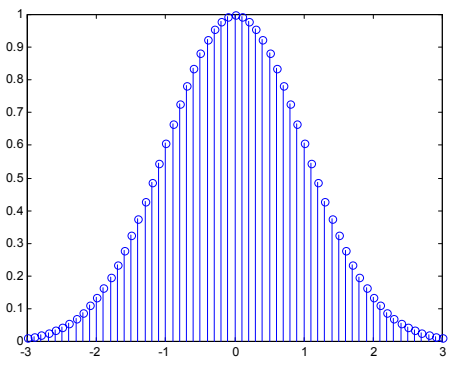


Рис.9.20

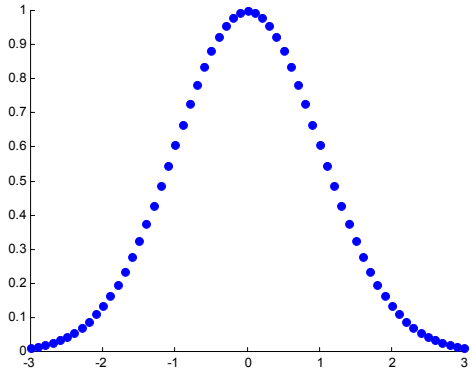


Рис.9.21

Вывод поля точек выполняется функцией **scatter(x,y,...)** с возможностью указывать размер, цвет и заполненность маркера:

```
»  x=-3:0.1:3;
» f=exp(-x.^2/2);
» scatter(x,f,'filled') % Рис.9.21
```

Среди многообразия функций специальной графики существенный интерес представляют функции поворота графического объекта **rotate** : например,

```
» h=surf(...);
» rotate (h,[1 0 0 ],90)        & поворот по оси x на 90°
```

и функции поворота графического объекта с помощью мыши **rotate3d on|ON|off** (on – режим включен, off-выключен, ON – подавляет информацию о текущих углах).

СОДЕРЖАНИЕ

ЦИТИРОВАННАЯ ЛИТЕРАТУРА

1. *В.Г.Потемкин*. Система инженерных и научных расчетов MATLAB 5.x. В 2-х т. –М.: ДИАЛОГ-МИФИ. 1999. – 670 с.

2. *М.А.Тынкевич.* Численные методы. – Кемерово: КузГТУ. 1997. – 122 с.

3. *А.И. Плис, Н.А. Сливина*. MATHCAD 2000. Практикум для экономистов и инженеров. -М.: Финансы и статистика. 2000. – 656 с.

Учебно-справочное издание

Тынкевич Моисей Аронович

Система MATLAB: справочное пособие к курсу
“Численные методы анализа”

ЛР № 020313 от 23.12.96
Подписано к печати 29.10.2001.
Формат 60×84 /16. Бумага офсетная.
Уч.-изд. л. 3.0 . Тираж 150 экз.
Заказ 438. Отпечатано на ризографе.

Кузбасский государственный технический университет.
650026, Кемерово, ул. Весенняя, 28.

Типография Кузбасского государственного технического университета.
650026, Кемерово, ул. Д.Бедного, 4а