Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Казанский (Приволжский) федеральный университет»

ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ ИМ. ЛОБАЧЕВСКОГО

КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ

Специальность: 050201.65: Математика с дополнительной специальностью информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Методическое пособие по FreeMat»

Работа завершена:

«__» _____ 2014 г._____ (Ризванова Г.Г.)

Работа допущена к защите:

Научный руководитель,

к.ф.-м.н., доцент

«___»_____ 2014 г._____(Попов А.А.)

Рецензент:
к.ф.-м.н., доцент

«___»_____ 2014 г._____ (Щербакова Н.К.)

Дата защиты:

« ____ » _____2014 г. Оценка _____

Заведующий кафедрой
Доктор физ.-мат. наук, профессор
«____» _____ 2014 г._____(Игнатьев Ю.Г.)

Казань – 2014 год

Оглавление

Введение 4

I. Интерфейс программы 6

1.1. Запуск программы 6

1.2. Арифметические операции с простыми переменными 8

1.3. Основные математические функции 11

1.4. Операторы и функции FreeMat 13

1.5. Операции отношения и логические операции 13

1.6. Переменные 14

1.7. Векторы и матрицы 14

1.8. Операции над матрицами и векторами 18

II. Работа с графиками 26

2.1. Двумерная графика 26

2.2. Графическая визуализация с помощью 2D в системе FreeMat 39

2.3. Трёхмерная графика 43

2.4. Отображение трехмерных графиков 48

2.5. Оформление графиков 57

III. Контрольные вопросы к теме «Основы работы в FreeMat» 63

Самостоятельные задания 64

Векторы 64

Матрицы 65

Двумерная графика 67

Трехмерная графика 68

Заключение 70

Литература 72

ВВЕДЕНИЕ

FreeMat - свободная матричная система для инженерных и научных расчетов, а также обработки данных. Система использует средства объектно-ориентированного программирования и имеет интерфейс к программным модулям, написанным на языках программирования C, C++ и Fortran. Freemat так же свободно распространяемая программа для математического моделирования и обработки данных.

Данная работа посвящена основам работы в FreeMat и охватывает только те возможности пакета, которые необходимы при дальнейшем изучении более специального круга вопросов. Описана работа из командной строки, вычисление арифметических выражений, использование одномерных и двумерных массивов.

Рассмотрены графические возможности FreeMat для визуализации данных и построения графиков функций.

Каждый параграф содержит варианты заданий для самостоятельной работы. Изучение материала рассчитано на семестр.

К целям данной работы можно отнести следующие аспекты:

- разработка методического пособия по использованию свободной алгебраической системы FreeMat при решении алгебраических задач и уравнений, а также для научных и инженерных вычислений, анализа данных;
- подбор заданий для представления практических возможностей программного продукта FreeMat и его применения в школьном образовательном процессе;
- подбор заданий для практического освоения программы.

Целевая аудитория данного пособия, в первую очередь, это студенты педагогических вузов и учителя математики, физики и близких им предметных областей, в которых часто применяются геометрические построения.

В работе описаны следующие возможности FreeMat

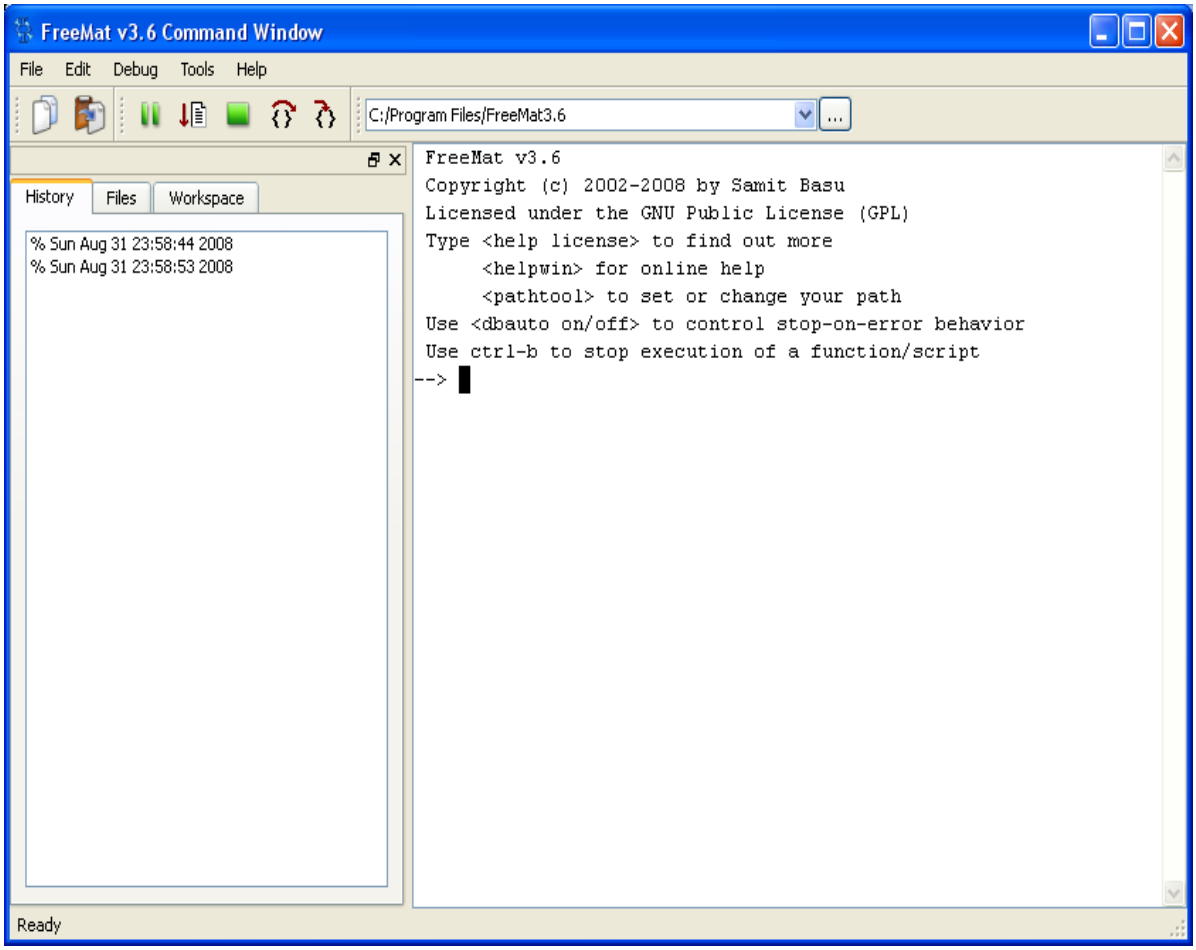
- арифметические действия над числами, векторами, матрицами;
- построение 2D-графиков;
- построение 3D-графиков.

К сожалению, имеющаяся сейчас в наличии версия FreeMat пока еще должным образом не русифицирована. В современных версиях языка FreeMat имеется достаточно подробная и удобная справочная информация, но она написана только на английском языке. Будем надеяться, что в следующих версиях FreeMat появится полноценная русифицированная справочная информация.

I. ИНТЕРФЕЙС ПРОГРАММЫ

1.1. Запуск программы

В командном окне: Запустите FreeMat. Когда вы делаете, вы должны увидеть окно, подобный показанному ниже.



Обратите внимание на мигающий курсор. Это показывает, где ваши нажатия клавиш пойдет, когда вы начинаете печатать.

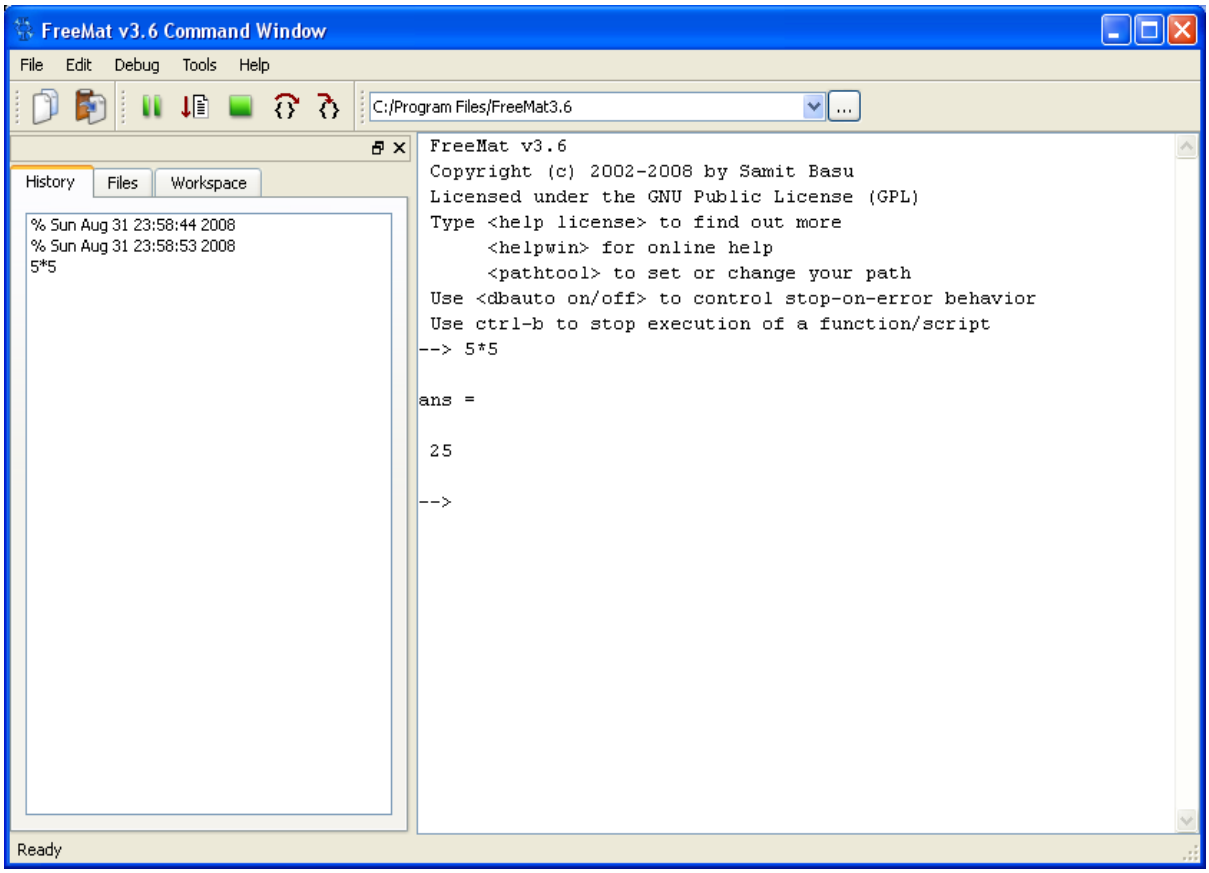
Сохранение и загрузка файлов:

--> X=[1 3]; Y=rand(10000,1);

-->save filename X Y; clear all;

-->loadfilename; whos.

Например, введите 5 * 5, а затем нажмите клавишу "Enter". Вы должны получить нечто подобное тому, что показано на рисунке ниже.



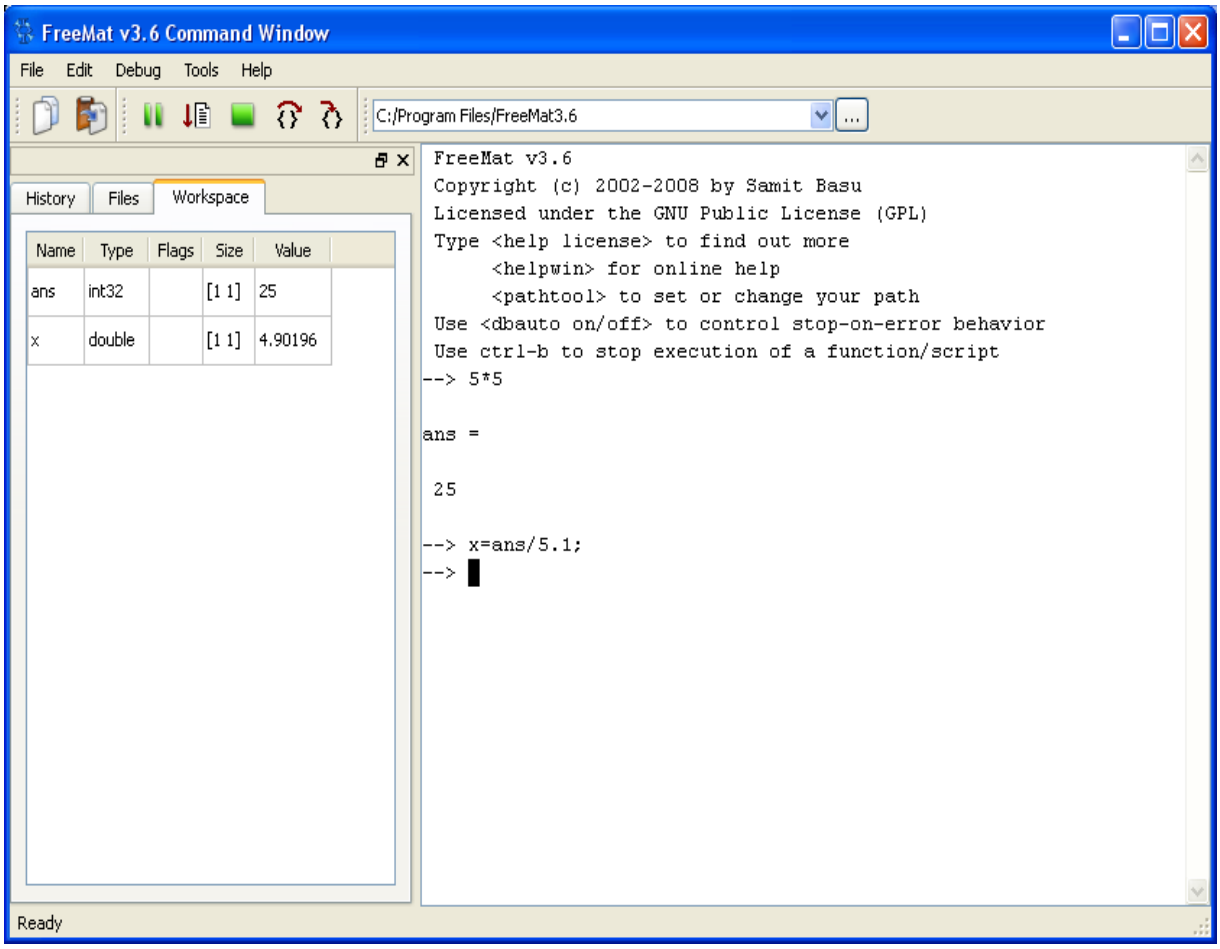
Ответ: Переменная

Обратите внимание на результат этой простой команды. Он отображает продукт как вошел и дает переменную, "ANS". Как вы уже догадались, это расшифровывается как "ответ". Вы можете использовать этой переменной. Например, введите следующую команду:

x = ans/5.1;

Помните: Для того, чтобы выполнить команду, нужно нажать "Enter".

Затем, щелкните на вкладке вблизи верхней, левом углу отмечены, "Рабочее пространство". Это дает вам список всех текущих переменных, которые FreeMat имеет значения, как показано на рисунке ниже.



Обратите внимание, как рабочего окна содержит имя каждой переменной в настоящее время в эксплуатации, его тип, размерность массива, и (в случае простых переменных) их значения.

1.2. Арифметические операции с простыми переменными

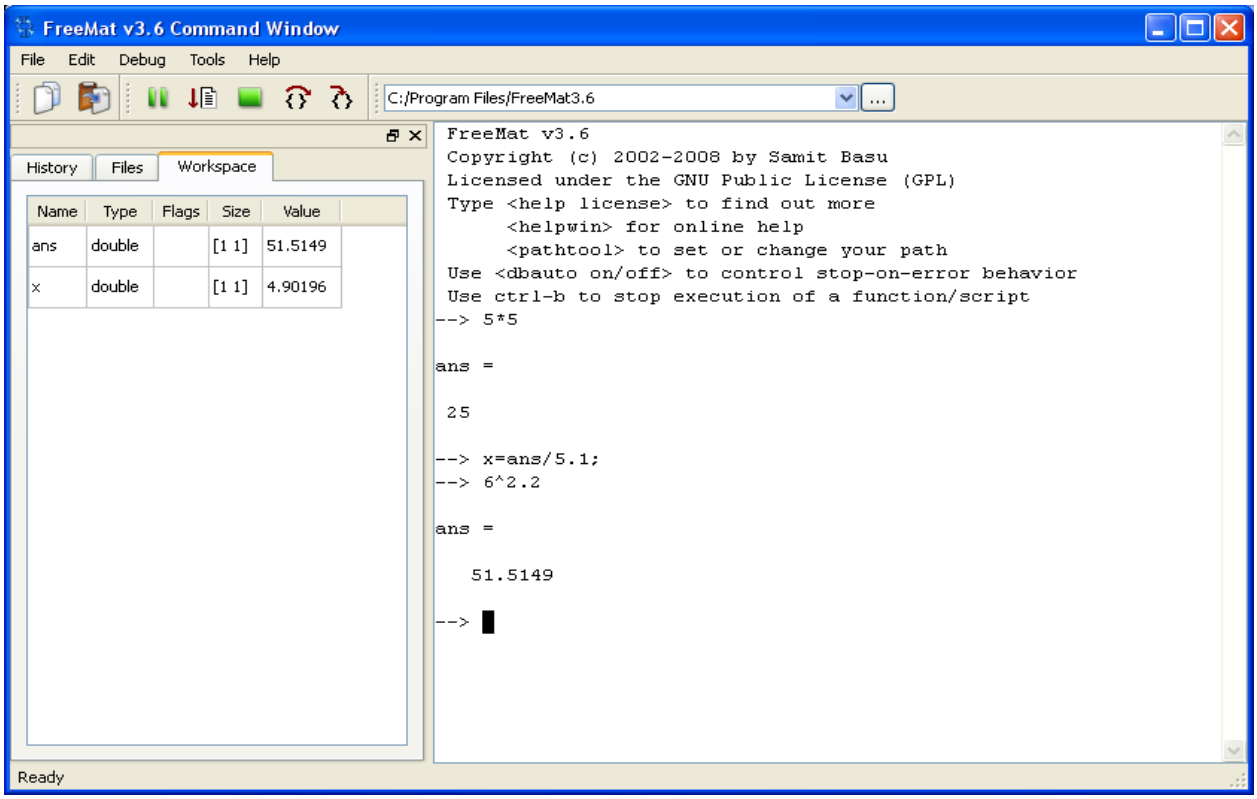
Рассмотрим базовые арифметические операции над простыми переменными в FreeMat. Пусть заданы две переменные a и b. Тогда операции сложения, вычитания, умножения и деления запишутся так:

c = a+b; % сложение

c = a-b; % вычитание

c = a*b; % умножение

c = a/b; % деление



Дополнительно, в FreeMat определена операция возведения в степень, которая записывается так:

c = a^2; % возведение переменной a в квадрат

c = a^0.5; % извлечение квадратного корня из переменной a

Приоритет арифметических операций * и / выше операций + и - , т.е. сначала выполняется умножение и деление, а затем, сложение и вычитание. Операция возведения в степень имеет наивысший приоритет. Для изменения приоритетов арифметических операций используются круглые скобки, как и в обычной математике, например,

c = 7+2*2; % значение c = 28

c = (7+2)*2; % значение c = 18

В практике программирования есть несколько устоявшихся подходов использования арифметических операций. Например, если необходимо увеличить значение переменной arg на 1, то этого можно добиться следующим образом:

arg = arg + 1;

При этом совершенно не важно чему равна переменная arg, в любом случае ее значение будет увеличено на 1. Аналогичным образом можно выполнять увеличение или уменьшение значения переменной на любую величину.

Другим устоявшимся приемом программирования является обмен значений между двумя переменными. Например, заданы переменные arg_a и arg_b и необходимо произвести обмен данными между ними. Это достигается следующим образом:

```
temp = arg_a;
arg_a = arg_b;
arg_b = temp;
```

Здесь temp – это временная переменная, необходимая для сохранения значения переменной arg_a, т.к. оно затирается во второй строчке значением arg_b. Поэтому в третьей строке переменной arg_b присваивается сохраненное в temp значение переменной arg_a.

Если в результате выполнения вычислений появляется **комплексное число**, то FreeMat автоматически будет оперировать с такими числами в соответствии с арифметикой комплексных чисел. Например, при извлечении квадратного корня из -1, получим следующий результат:

```
c = (-1)^0.5;       % c = 0.0000 + 1.0000i
```

Здесь i – зарезервированное имя мнимой единицы и представленная запись обозначает комплексное число с нулевой действительной частью и единичной мнимой. Во FreeMat в качестве зарезервированного имени мнимой единицы также используется буква j.

Для того, чтобы задать комплексную переменную достаточно указать значения их действительной и мнимой частей как показано ниже

```
c = 6 + 5i;         % комплексное число
```

и с заданными комплексными переменными также можно выполнять описанные выше арифметические операции.

При работе с комплексными числами существуют две специальные функции:

real(x) – взятие действительной части комплексного числа x;

imag(x) – взятие мнимой части комплексного числа x;

abs(x) – вычисление абсолютного значения комплексного числа x;

conj(x) – вычисление комплексно-сопряженного числа x;

angle(x) – вычисление аргумента комплексного числа x.

1.3. Основные математические функции

FreeMat содержит в себе все распространенные математические функции, которые доступны по их имени при реализации алгоритмов. Например, функция sqrt() позволяет вычислять квадрат числа и может быть использована в программе следующим образом:

```
x = 2;
y = 4;
d = sqrt(x^2+y^2);  %вычисление евклидового расстояния
```

Аналогичным образом вызываются и все другие математические функции, представленные в табл. 1.2.

Таблица 1.2. Основные математические функции FreeMat

sqrt(x)	вычисление квадратного корня
exp(x)	возведение в степень числа e
pow2(x)	возведение в степень числа 2
log(x)	вычисление натурального логарифма
log10(x)	вычисление десятичного логарифма
log2(x)	вычисление логарифма по

	основанию 2
sin(x)	синус угла x, заданного в радианах
cos(x)	косинус угла x, заданного в радианах
tan(x)	тангенс угла x, заданного в радианах
cot(x)	котангенс угла x, заданного в радианах
asin(x)	арксинус
acos(x)	арккосинус
atan(x)	арктангенс
Pi	число пи
round(x)	округление до ближайшего целого
fix(x)	усечение дробной части числа
floor(x)	округление до меньшего целого
ceil(x)	округление до большего целого
mod(x)	остаток от деления с учётом знака
sign(x)	знак числа
factor(x)	разложение числа на простые множители
isprime(x)	истинно, если число простое
rand	генерация псевдослучайного числа с равномерным законом распределения
Randn	генерация псевдослучайного числа с нормальным законом

	распределения
abs(x)	вычисление модуля числа

Почти все элементарные функции допускают вычисления и с комплексными аргументами. Например:

res = sin(2+3i)*atan(4i)/(1 — 6i); % res = -1.8009 — 1.9190i

или

exp(i*x) = cos(x)+i*sin(x).

1.4. Операторы и функции FreeMat

FreeMat имеет полный набор арифметических операций и математических функций, присущий современным системам компьютерной математики. Небольшие тонкости тут есть, например отсутствие функции натурального логарифма ln- она обозначена как log:

-->log(2)

ans = 0.6931

Зато есть функция логарифма по основанию 10 – log10 и по основанию 2 – log2.

1.5. Операции отношения и логические операции

- <меньше;
- <= меньше или равно;
- >больше;
- >= больше или равно;

- == равно;
- ~= не равно;
- **&**и;
- | или;
- ~ не.

1.6. Переменные

- Имя переменной должно начинаться с **буквы!**
- **F** и **f** – разные переменные.

Не используйте в качестве имѐн переменных имена функций и операторов (case, for, whileи т.д.).

-->which variable_name проверяет, используется ли уже variable_name в качестве имени функции.

1.7. Векторы и матрицы

Column vector	Row vector	General matrix
v=[1; 2; 3] v= 1 2 3	v=[1 2 3] v= 1 2 3	A =[1 2 3; 4 5 6] A = 1 2 3 4 5 6

-->**length(v)** длина вектора;

-->**size(A)** размер матрицы.

Выше были рассмотрены операции с простыми переменными. Однако с их помощью сложно описывать сложные данные, такие как

случайный сигнал, поступающий на вход фильтра или хранить кадр изображения и т.п. Поэтому в языках высокого уровня предусмотрена возможность хранить значения в виде массивов. В FreeMat эту роль выполняют векторы и матрицы.

Ниже показан пример задания вектора с именем a, и содержащий значения 1, 2, 3, 4:

```
a = [1 2 3 4];      % вектор-строка
```

Для доступа к тому или иному элементу вектора используется следующая конструкция языка:

```
disp( a(1) );  % отображение значения 1-го элемента вектора
disp( a(2) );  % отображение значения 2-го элемента вектора
disp( a(3) );  % отображение значения 3-го элемента вектора
disp( a(4) );  % отображение значения 4-го элемента вектора
```

т.е. нужно указать имя вектора и в круглых скобках написать номер индекса элемента, с которым предполагается работать. Например, для изменения значения 2-го элемента массива на 10 достаточно записать

```
a(2) = 10; % изменение значения 2-го элемента на 10
```

Часто возникает необходимость определения общего числа элементов в векторе, т.е. определения его размера. Это можно сделать, воспользовавшись функцией length() следующим образом:

```
N = length(a); % (N=4) число элементов массива a
```

Если требуется задать вектор-столбец, то это можно сделать так

```
a = [1; 2; 3; 4];   % вектор-столбец
```

или так

```
b = [1 2 3 4]';         % вектор-столбец
```

при этом доступ к элементам векторов осуществляется также как и для векторов-строк.

Следует отметить, что векторы можно составлять не только из отдельных чисел или переменных, но и из векторов. Например,

следующий фрагмент программы показывает, как можно создавать один вектор на основе другого:

a = [1 2 3 4]; % начальный вектор a = [1 2 3 4]

b = [a 5 6]; % второй вектор b = [1 2 3 4 5 6]

Здесь вектор b состоит из шести элементов и создан на основе вектора a. Используя этот прием, можно осуществлять увеличение размера векторов в процессе работы программы:

a = [a 5]; % увеличение вектора a на один элемент

Недостатком описанного способа задания (инициализации) векторов является сложность определения векторов больших размеров, состоящих, например, из 100 или 1000 элементов. Чтобы решить данную задачу, в FreeMat существуют функции инициализации векторов нулями, единицами или случайными значениями:

a1 = zeros(1, 100); % вектор-строка, 100 элементов с

% нулевыми значениями

a2 = zeros(100, 1); % вектор-столбец, 100 элементов с

% нулевыми значениями

a3 = ones(1, 1000); % вектор-строка, 1000 элементов с

% единичными значениями

a4 = ones(1000, 1); % вектор-столбец, 1000 элементов с

% единичными значениями

a5 = rand(1000, 1); % вектор-столбец, 1000 элементов со

% случайными значениями

Матрицы в FreeMat задаются аналогично векторам с той лишь разницей, что указываются обе размерности. Приведем пример инициализации единичной матрицы размером 3x3:

E = [1 0 0; 0 1 0; 0 01]; % единичная матрица 3x3

или

E = [1 0 0

0 1 0

0 0 1]; % единичная матрица 3x3

Аналогичным образом можно задавать любые другие матрицы, а также использовать приведенные выше функции zeros(), ones() и rand(), например:

A1 = zeros(10,10); % нулевая матрица 10x10 элементов

или

A2 = zeros(10); % нулевая матрица 10x10 элементов

A3 = ones(5); % матрица 5x5, состоящая из единиц

A4 = rand(100); % матрица 100x100, из случайных чисел

Для доступа к элементам матрицы применяется такой же синтаксис как и для векторов, но с указанием строки и столбца где находится требуемый элемент:

A = [1 2 3;4 5 6;7 8 9]; % матрица 3x3

disp(A(2,1)); % вывод на экран элемента, стоящего во

% второй строке первого столбца, т.е. 4

disp(A(1,2)); % вывод на экран элемента, стоящего в

% первой строке второго столбца, т.е. 2

Также возможны операции выделения указанной части матрицы, например:

B1 = A(:,1); % B1 = [1; 4; 7] – выделение первого столбца

B2 = A(2,:); % B2 = [1 2 3] – выделение первой строки

B3 = A(1:2,2:3); % B3 = [2 3; 5 6] – выделение первых двух

% строк и 2-го и 3-го столбцов матрицы A.

Размерность любой матрицы или вектора в FreeMat можно определить с помощью функции size(), которая возвращает число строк и столбцов переменной, указанной в качестве аргумента:

a = 5; % переменная a

A = [1 2 3]; % вектор-строка

B = [1 2 3; 4 5 6]; % матрица 2x3

size(a) % 1x1

size(A) % 1x3

size(B) % 2x3

1.8. Операции над матрицами и векторами

В системе FreeMat достаточно просто выполняются математические операции над матрицами и векторами. Рассмотрим сначала простые операции сложения и умножения матриц и векторов. Пусть даны два вектора

a = [1 2 3 4 5]; % вектор-строка

b = [1; 1; 1; 1; 1]; % вектор-столбец

тогда умножение этих двух векторов можно записать так

c = a*b; % c=1+2+3+4+5=16

d = b*a; % d – матрица 5x5 элементов

В соответствии с операциями над векторами, умножение вектор-строки на вектор-столбец дает число, а умножение вектор-столбца на вектор-строку дает двумерную матрицу, что и является результатом вычислений в приведенном примере, т.е.

$$c = \sum_{i=1}^{5} a_i b_i = 1+2+3+4+5+6 = 16,$$

$$d = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix}.$$

Сложение и вычитание двух векторов записывается так

a1 = [1 2 3 4 5];

a2 = [5 4 3 2 1];

c = a1+a2; % c = [1+5, 2+4, 3+3, 4+2, 5+1];

c = a2-a1; % c = [5-1, 4-2, 3-3, 2-4, 1-5];

Следует обратить внимание, что операции сложения и вычитания можно выполнять между двумя векторами-столбцами или двумя векторами-строками. Иначе FreeMat выдаст сообщение об ошибке, т.к. разнотипные векторы складывать нельзя. Так обстоит дело со всеми недопустимыми арифметическими операциями: в случае невозможности их вычисления система FreeMat сообщит об ошибке и выполнение программы будет завершено на соответствующей строке.

Аналогичным образом выполняются операции умножения и сложения между матрицами:

```
A = [1 2 3; 4 5 6; 7 8 9];
B = ones(3);
C = A+B;          % сложение двух матриц одинакового размера
D = A+5;          % сложение матрицы и числа
E = A*B;          % умножение матрицы A на B
F = B*A;          % умножение матрицы B на A
G = 5*A;          % умножение матрицы на число
```

Операции вычисления обратной матрицы, а также транспонирования матриц и векторов, записываются следующим образом:

```
a = [1 1 1];      % вектор-строка
b = a’;           % вектор-столбец, образованный
% транспонированием вектора-строки a.
A = [1 2 3; 4 5 6; 7 8 9]; % матрица 3x3 элемента
B = a*A;          % B = [12 15 18] – вектор-строка
C = A*b;          % C = [6; 15; 24] – вектор-столбец
D = a*A*a’;       % D = 45 – число, сумма эл-ов матрицы A
E = A’;           % E – транспонированная матрица A
F = inv(A);       % F – обратная матрица A
G = A^-1;         % G – обратная матрица A
```

Из приведенного примера видно, что операция транспонирования матриц и векторов обозначается символом ‘ (апостроф), который ставится после имени вектора или матрицы. Вычисление обратной матрицы можно делать путем вызова функции inv() или возводя матрицу в степень -1. Результат в обоих случаях будет одинаковым, а два способа вычисления сделано для удобства использования при реализации различных алгоритмов.

Скалярное произведение двух векторов возвращает функция dot, а векторное — cross:

```
>> s=dot(a,b)
>> c=cross(a,b)
```

Разумеется, векторное произведение определено только для векторов из трех элементов.

Для операции транспонирования зарезервирован апостроф '. Если вектор содержит комплексные числа, то операция ' приводит к комплексно-сопряженному вектору. При вычислении скалярного и векторного произведений функциями cross и dot не обязательно следить за тем, чтобы оба вектора были либо столбцами, либо строками.

Результат получается верный, например, при обращении c=cross(a,b'), только c становится вектор-строкой.

FreeMat поддерживает поэлементные операции с векторами. Наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения .*(точка со звездочкой). Данная операция применяется к векторам одинаковой длины и приводит к вектору той же длины, что исходные, элементы которого равны произведениям соответствующих элементов исходных векторов. Например, для векторов a и b, введенных выше, поэлементное умножение дает следующий результат:

```
>> c=a.*b
c =
```

1.5200

-0.3900

Аналогичным образом работает поэлементное деление ./ (точка с косой чертой). Кроме того, операция .\ (точка с обратной косой чертой) осуществляет обратное поэлементное деление, то есть выражения a./b и b.\a эквивалентны. Возведение элементов вектора a в степени, равные соответствующим элементам b, производится с использованием .^. Для транспонирования вектор-строк или вектор-столбцов предназначено сочетание .' (точка с апострофом). Операции ' и .' для вещественных векторов приводят к одинаковым результатам. Не обязательно применять поэлементные операции при умножении вектора на число и числа на вектор, делении вектора на число, сложении и вычитании вектора и числа. При выполнении, например, операции a*2, результат представляет собой вектор того же размера, что и a, с удвоенными элементами. Если в процессе вычислений требуется поэлементно умножить, разделить или возвести в степень элементы вектора или матрицы, то для этого используются операторы:

.* - поэлементное умножение;

./ и .\ - поэлементные деления;

.^ - поэлементное возведение в степень.

Рассмотрим работу данных операторов на следующем примере.

```
a = [1 2 3];        % вектор-строка
b = [3 2 1];        % вектор-строка
c = a.*b;           % c = [3 4 3]
A = ones(3);        % матрица 3x3, состоящая из единиц
B = [1 2 3;4 5 6; 7 8 9];        % матрица 3x3
C = A.*B;           % матрица 3x3
D = A./B;           % матрица 3x3
E = A.\B;           % матрица 3x3
F = A.^2;           % возведение элементов матрицы A в квадрат
```

В заключении данного параграфа рассмотрим несколько функций полезных при работе с векторами и матрицами.

Для поиска максимального значения элемента вектора используется стандартная функция max(), которая возвращает найденное максимальное значение элемента и его позицию (индекс):

```
a = [1 6 3 4];
[v, i] = max(a);          % v = 6, i = 2;
```

или

```
v = max(a);               % v = 6;
```

Приведенный пример показывает два разных способа вызова функции max(). В первом случае определяется и максимальное значение элемента и его индекс в векторе, а во втором – только максимальное значение элемента.

В случае с матрицами, данная функция определяет максимальные значения, стоящие в столбцах, как показано ниже в примере:

```
A = [4 3 5; 6 7 2; 3 1 8];
[V, I] = max(A);          % V=[6 7 8], I = [2 2 3]
V = max(A);               % V=[6 7 8]
```

Полный синтаксис функции max() можно узнать, набрав в командном окне freemat команду

help <название функции>

Аналогичным образом работает функция min(), которая определяет минимальное значение элемента вектора или матрицы и его индекс.

Другой полезной функцией работы с матрицами и векторами является функция sum(), которая вычисляет сумму значений элементов вектора или столбцов матрицы:

```
a = [3 5 4 2 1];
s = sum(a);               % s = 3+5+4+2+1=15
A = [4 3 5; 6 7 2; 3 1 8];
```

```
S1 = sum(A);                % S1=[13 11 15]
S2 = sum(sum(A));           % S2=39
```

При вычислении суммы S2 сначала вычисляется сумма значений элементов матрицы A по столбцам, а затем, по строкам. В результате, переменная S2 содержит сумму значений всех элементов матрицы A.

Для сортировки значений элементов вектора или матрицы по возрастанию или убыванию используется функция sort() следующим образом:

```
a = [3 5 4 2 1];
b1 = sort(a)            % b1=[1 2 3 4 5]
b2 = sort(a);     % b2=[5 4 3 2 1]
b3 = sort(a);     % b3=[1 2 3 4 5]
```

для матриц

```
A = [4 3 5; 6 7 2; 3 1 8];
B1 = sort(A);             % B1=[3 1 2
                     %     4 3 5
                      %     6 7 8]
B2 = sort(A);     % B2=[6 7 8
                      %     4 3 5
                      %     3 1 2]
```

Во многих практических задачах часто требуется найти определенный элемент в векторе или матрице. Это можно выполнить с помощью стандартной функции find(), которая в качестве аргумента принимает условие, в соответствии с которым и находятся требуемые элементы, например:

```
a = [3 5 4 2 1];
b1 = find(a == 2);          % b1 = 4 – индекс элемента 2
b2 = find(a ~= 2);          % b2 = [1 2 3 5] – индексы без 2
b3 = find(a > 3);           % b3 = [2 3]
```

В приведенном примере символ ‘==’ означает проверку на равенство, а символ ‘~=’ выполняет проверку на неравенство значений элементов вектора a. Более подробно об этих операторах будет описано в разделе условные операторы.

Еще одной полезной функцией работы с векторами и матрицами является функция mean() для вычисления среднего арифметического значения, которая работает следующим образом:

```
a = [3 5 4 2 1];
m = mean(a);                % m = 3
A = [4 3 5; 6 7 2; 3 1 8];
M1 = mean(A);               % M1 = [4.333 3.667 5.000]
M2 = mean(mean(A));         % M2 = 4.333
```

Векторы могут быть аргументами встроенных математических функций, таких, как sin, cos и т. д. В результате получается вектор с элементами, равными значению вызываемой функции от соответствующих элементов исходного вектора, например:

```
>> q=sin([0 pi/2 pi])
q =
0 1.0000 0.0000
```

Однако для вычисления более сложной функции от вектора значений, скажем, выражение f=(v*sin(v)+v^2)/(v+1) вызовет ошибку уже при попытке умножения v на sin(v). Дело в том, что v является вектор-строкой длиной четыре, т. е. хранится в двумерном массиве размером один на четыре. Точно также представлен и sin(v), следовательно, умножение при помощи звездочки (по правилу матричного умножения) лишено смысла. Аналогичная ситуация возникает и при возведении вектора v в квадрат, т. е., фактически, при вычислении v*v. Правильная запись выражения в FreeMat требует использования поэлементных операций:

```
>> f=(v.*sin(v)+v.^2)./(v+1)
```

Часто требуется вычислить функцию от вектора значений аргумента, отличающихся друг от друга на постоянный шаг. Для создания таких вектор-строк предусмотрено двоеточие. Последовательность команд

```
>> x=-1.2:0.5:1.8;
>> f=(x.*sin(x)+x.^2)./(x+1);
```

приводит к заполнению следующих векторов:

```
>> x
x =
-1.2000 -0.7000 -0.2000 0.3000 0.8000 1.3000 1.8000
>> f
f =
-12.7922 3.1365 0.0997 0.1374 0.6744 1.2794 1.7832
```

Шаг может быть отрицательным, в этом случае начальное значение должно быть больше, либо равно конечному для получения непустого вектора. Если шаг равен единице, то его можно не указывать, например:

```
>> n=-3:4
n =
-3 -2 -1 0 1 2 3 4
```

Ясно, что для заполнения вектор-столбца элементами с постоянным шагом следует транспонировать вектор-строку. Создание векторов при помощи двоеточия и умение производить поэлементные операции необходимо для визуализации массивов данных, о чем будет сказано в следующих разделах.

II. РАБОТА С ГРАФИКАМИ

Freemat предоставляет богатый инструментарий по визуализации данных. Используя внутренний язык, можно выводить двумерные и трехмерные графики в декартовых и полярных координатах, выполнять отображение изображений с разной глубиной цвета и разными цветовыми картами, создавать простую анимацию результатов моделирования в процессе вычислений и многое другое.

2.1. Двумерная графика

Рассмотрение возможностей FreeMat по визуализации данных начнем с двумерных графиков, которые обычно строятся с помощью функции plot(). Множество вариантов работы данной функции лучше всего рассмотреть на конкретных примерах.

Предположим, что требуется вывести график функции синуса в диапазоне от 0 до π. Для этого зададим вектор (множество) точек по оси Ox, в которых будут отображаться значения функции синуса:

```
x = 0:0.01:pi;
```

В результате получится вектор столбец со множеством значений от 0 до π и с шагом 0,01. Затем, вычислим множество значений функции синуса в этих точках:

```
y = sin(x);
```

и выведем результат на экран

```
plot(x,y);
```

В результате получим график, представленный на рис. 2.1.1.

Представленная запись функции plot() показывает, что сначала записывается аргумент со множеством точек оси Ox, а затем, аргумент со множеством точек оси Oy. Зная эти значения, функция plot() имеет

возможность построить точки на плоскости и линейно их интерполировать для придания непрерывного вида графика.

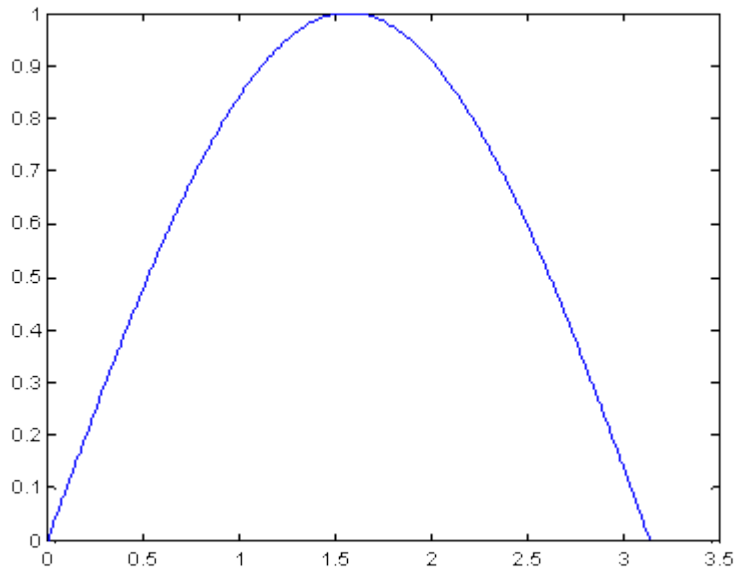


Рис. 2.1.1. Отображение функции синуса с помощью функции plot().

Функцию plot() можно записать и с одним аргументом x или y:

```
plot(x);
plot(y);
```

в результате получим два разных графика, представленные на рис. 2.1.2.

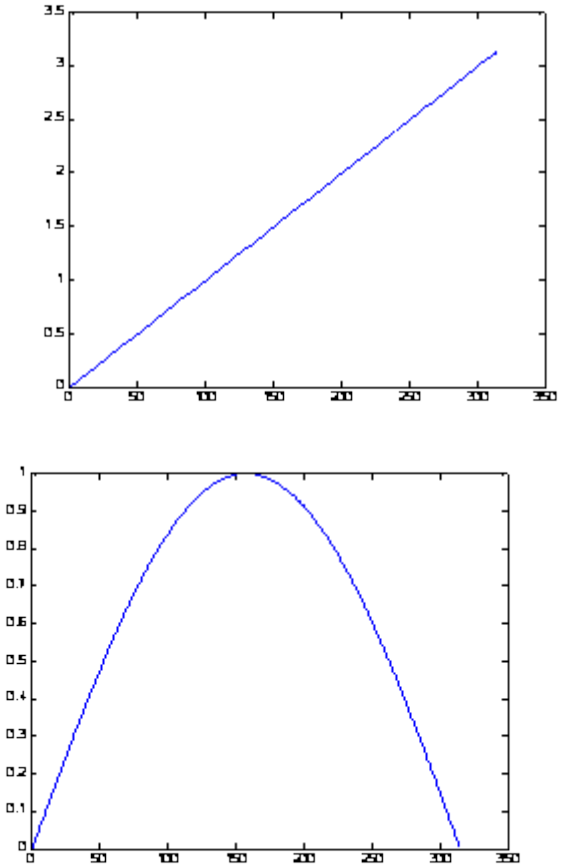
Анализ рис. 2.1.2 показывает, что в случае одного аргумента функция plot() отображает множество точек по оси Oy, а по оси Ox происходит автоматическая генерация множества точек с единичным шагом. Следовательно, для простой визуализации вектора в виде двумерного графика достаточно воспользоваться функцией plot() с одним аргументом.

Для построения нескольких графиков в одних и тех же координатных осях, функция plot() записывается следующим образом:

```
x=0:0.01:pi;
y1=sin(x);
```

```
y2=cos(x);
plot(x,y1,x,y2);
```

Результат работы данного фрагмента программы представлен на рис. 2.1.3.



а) б)

Рис. 2.1.2. Результаты работы функции plot() с одним аргументом: а – plot(x); б – plot(y).

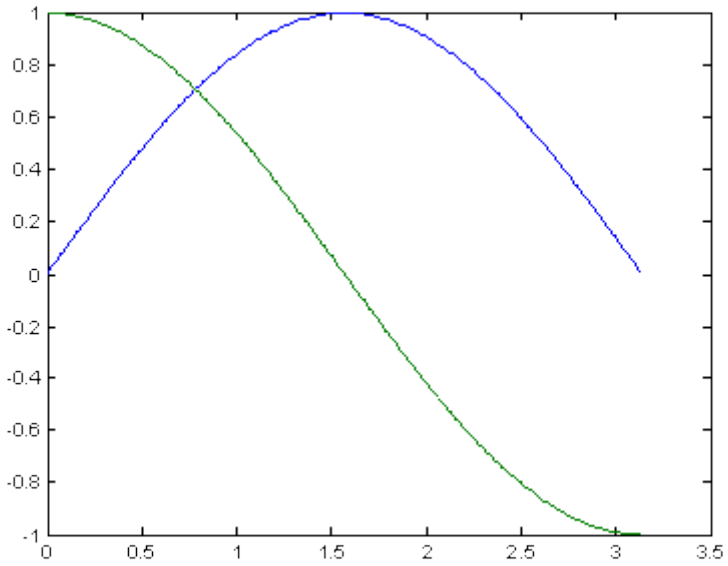


Рис. 2.1.3. Отображение двух графиков в одних координатных осях.

Аналогичным образом можно построить два графика, используя один аргумент функции plot(). Предположим, что есть два вектора значений

y1=sin(x);

y2 = cos(x);

которые требуется отобразить на экране. Для этого объединим их в двумерную матрицу

$$[y1'\,y2'] = \begin{vmatrix} y1_1 & y2_1 \\ y1_2 & y2_2 \\ \dots & \dots \\ y1_N & y2_N \end{vmatrix},$$

в которой столбцы составлены из векторов y1 и y2 соответственно. Такая матрица будет отображена функцией

plot([y1’y2’]); % апострофы переводят вектор-строку

% в вектор-столбец в виде двух графиков (рис. 2.1.4).

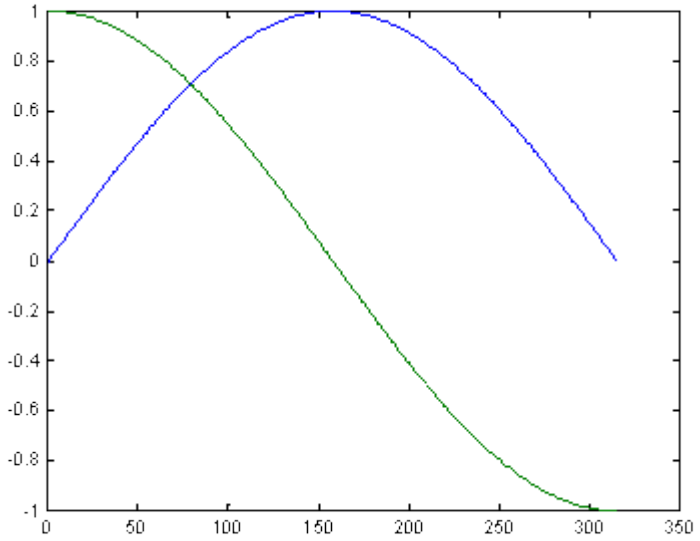


Рис. 2.1.4. Отображение двумерной матрицы в виде двух графиков.

Два вектора в одних осях можно отобразить только в том случае, если их размерности совпадают. Когда же выполняется работа с векторами разных размерностей, то они либо должны быть приведены друг к другу по числу элементов, либо отображены на разных графиках. Отобразить графики в разных координатных осях можно несколькими способами. В самом простом случае можно создать два графических окна и в них отобразить нужные графики. Это делается следующим образом:

x1=0:0.01:2*pi;

y1=sin(x1);

x2=0:0.01:pi;

y2=cos(x2);

plot(x1,y1); % рисование первого графика

figure; % создание 2-го графического окна

plot(x2, y2); % рисование 2-го графика во 2-м окне

Функция figure, используемая в данной программе, создает новое графическое окно и делает его активным. Функция plot(), вызываемая сразу после функции figure, отобразит график в текущем активном

графическом окне. В результате на экране будут показаны два окна с двумя графиками.

Неудобство работы приведенного фрагмента программы заключается в том, что повторный вызов функции figure отобразит на экране еще одно новое окно и если программа будет выполнена дважды, то на экране окажется три графических окна, но только в двух из них будут актуальные данные. В этом случае было бы лучше построить программу так, чтобы на экране всегда отображалось два окна с нужными графиками. Этого можно достичь, если при вызове функции figure в качестве аргумента указывать номер графического окна, которое необходимо создать или сделать активным, если оно уже создано. Таким образом, вышеприведенную программу можно записать так.

```
x1=0:0.01:2*pi;
y1=sin(x1);
x2=0:0.01:pi;
y2=cos(x2);
figure(1);        % создание окна с номером 1
plot(x1,y1);      % рисование первого графика
figure(2);        % создание графического окна с номером 2
plot(x2, y2);     % рисование 2-го графика во 2-м окне
```

При выполнении данной программы на экране всегда будут отображены только два графических окна с номерами 1 и 2, и в них показаны графики функций синуса и косинуса соответственно.

В некоторых случаях большего удобства представления информации можно достичь, отображая два графика в одном графическом окне. Это достигается путем использования функции subplot(), имеющая следующий синтаксис:

subplot(<число строк>, <число столбцов>, <номер координатной оси>)

Рассмотрим пример отображения двух графиков друг под другом вышеприведенных функций синуса и косинуса.

```
x1=0:0.01:2*pi;
y1=sin(x1);
x2=0:0.01:pi;
y2=cos(x2);
figure(1);
subplot(2,1,1);     % делим окно на 2 строки и один столбец
plot(x1,y1);        % отображение первого графика
subplot(2,1,2);     % строим 2-ю координатную ось
plot(x2,y2);        % отображаем 2-й график в новых осях
```

Результат работы программы показан на рис. 2.1.5.

Аналогичным образом можно выводить два и более графиков в столбец, в виде таблицы и т.п. Кроме того, можно указывать точные координаты расположения графика в графическом окне. Для этого используется параметр position в функции subplot():

subplot(‘position’, [left bottom width height]);

где left – смещение от левой стороны окна; bottom – смещение от нижней стороны окна; width, height – ширина и высота графика в окне. Все эти переменные изменяются в пределах от 0 до 1.

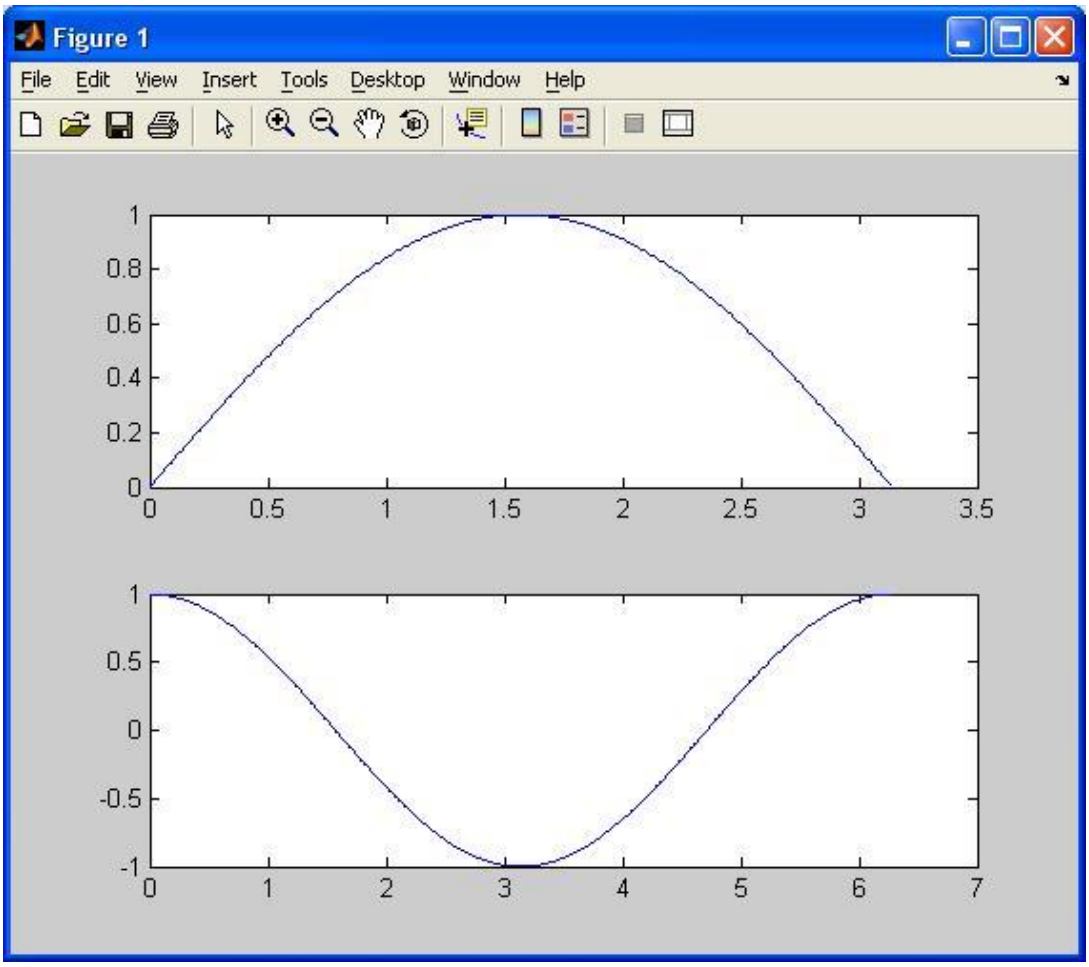


Рис. 2.1.5. Пример работы функции subplot.

Ниже представлен фрагмент программы отображения графика функции синуса в центре графического окна. Результат работы показан на рис. 2.1.6.

```
x1=0:0.01:2*pi;
y1=sin(x1);
subplot('position', [0.33 0.33 0.33 0.33]);
plot(x1,y1);
```

В данном примере функция subplot() смещает график на треть от левой и нижней границ окна и рисует график с шириной и высотой в треть графического окна. В результате, получается эффект рисования функции синуса по центру основного окна.

Таким образом, используя параметр position можно произвольно размещать графические элементы в плоскости окна.

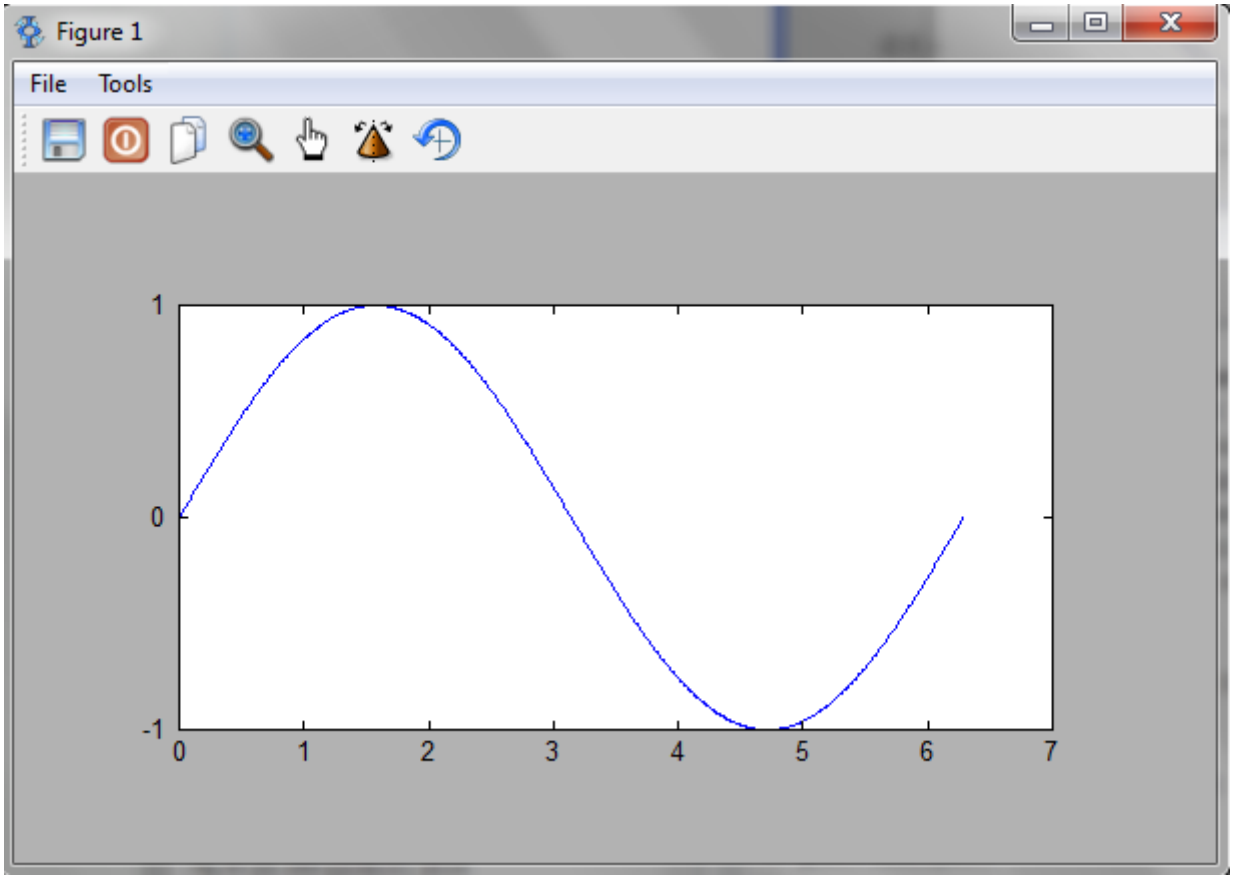


Рис. 2.1.6. Пример работы функции subplot с параметром position.

После написания программы для построения графика автоматически появляется окно уже с графиком. В этом окне имеются также свое меню с файлами и инструментами.

Итак,

 – может быть использовано, чтобы сохранить участок в графическом формате.

 – закрыть окно.

 – создаст копию окна графика.

 – позволит для увеличения сюжета с помощью мыши. Каждый щелчок мыши будет разделять обе горизонтальные и вертикальные оси в два раза.

 – позволит вам скользить область построения как вверх, так и вниз и / или влево и вправо. Выбрав этот значок, вы можете перетащить область построения с помощью мыши, чтобы переместить его.

 – позволяет вращать рисунок по оси. *ПРИМЕЧАНИЕ: Это действительно полезно только для трехмерных (3D) графики.*

 – позволяет вращать рисунок по оси x и по оси y. *ПРИМЕЧАНИЕ: Это действительно полезно только для трехмерных (3D) графики.*

```
x=0:0.1:pi;
y=sin(x); z=cos(x)
plot(x,y)
hold on
plot(x,z)
```

- Команды holdon и holdoff позволяют добавлять кривые на существующий график.
- Функция subplot позволяет выводить множество графиков в одном окне

subplot(row,col,num)

Число строк Число столбцов Порядковый номер графика

```
t=0:(2*pi/100):(2*pi);
x=cos(t);
y=sin(t*5);
z=cos(t/2).*3;
w=tan(t*4);
subplot(2,2,1);
plot(t,x);
subplot(2,2,2);
plot(t,y);
subplot(2,2,3);
plot(t,z);
subplot(2,2,4);
plot(t,w);
```

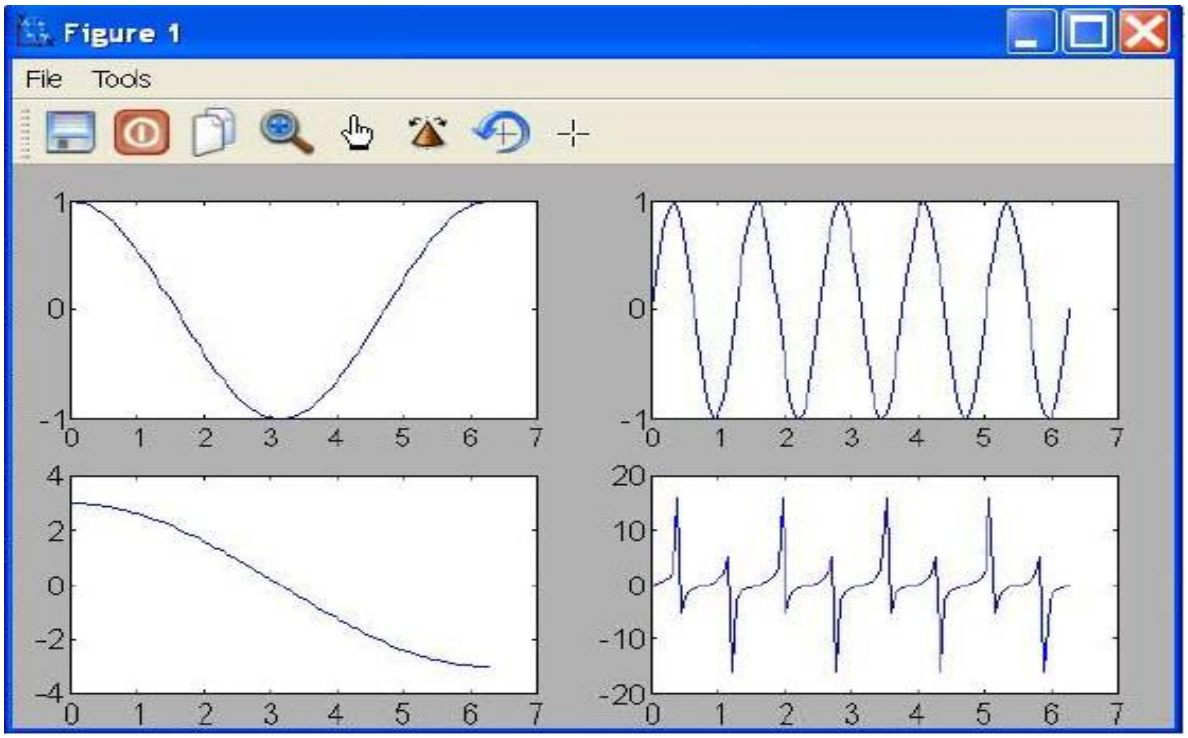


Рис. 2.1.7. Построение 4 функций на одном окне

FreeMat так же строит как простые (cos(x), sin(x), tan(x)), так и сложные тригонометрические функции (acos(x), acosh(x), acot()x,

acoth(x), csc(x), acsc(x)) и др. Например, построим графики функций y=cos(x) и y=acos(x), периоды в обоих случаях возьмем (-1, 1).

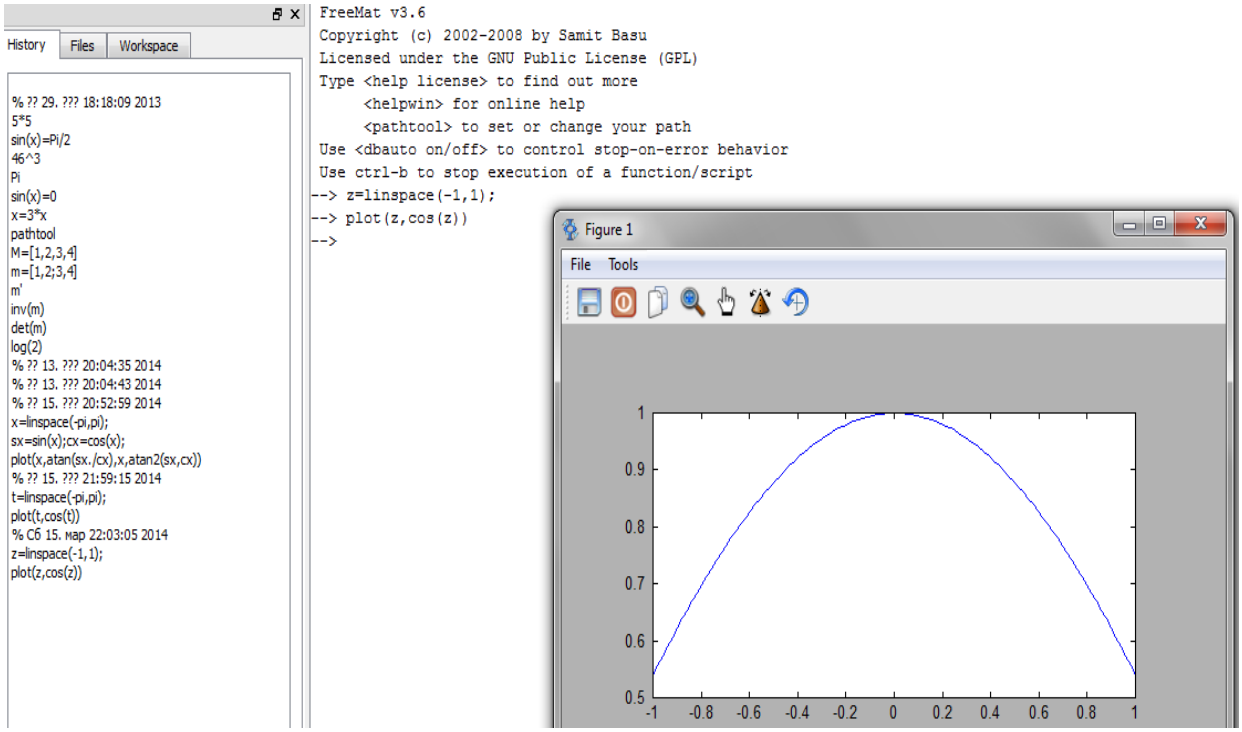


Рис. 2.1.8. График функции y=cos(x)

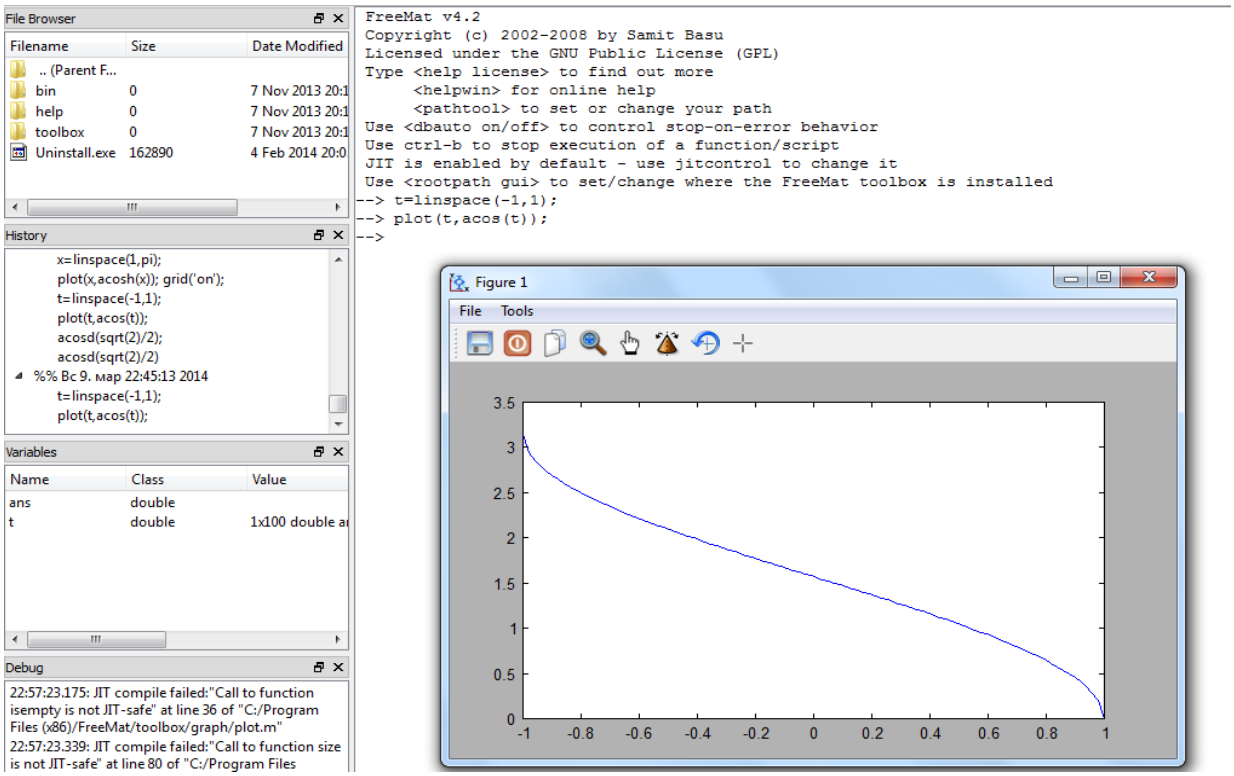


Рис. 2.1.9. График функции y=acos(x)

Так же одним из интересных и непростых графиков является график функции z = atan2(y,x).

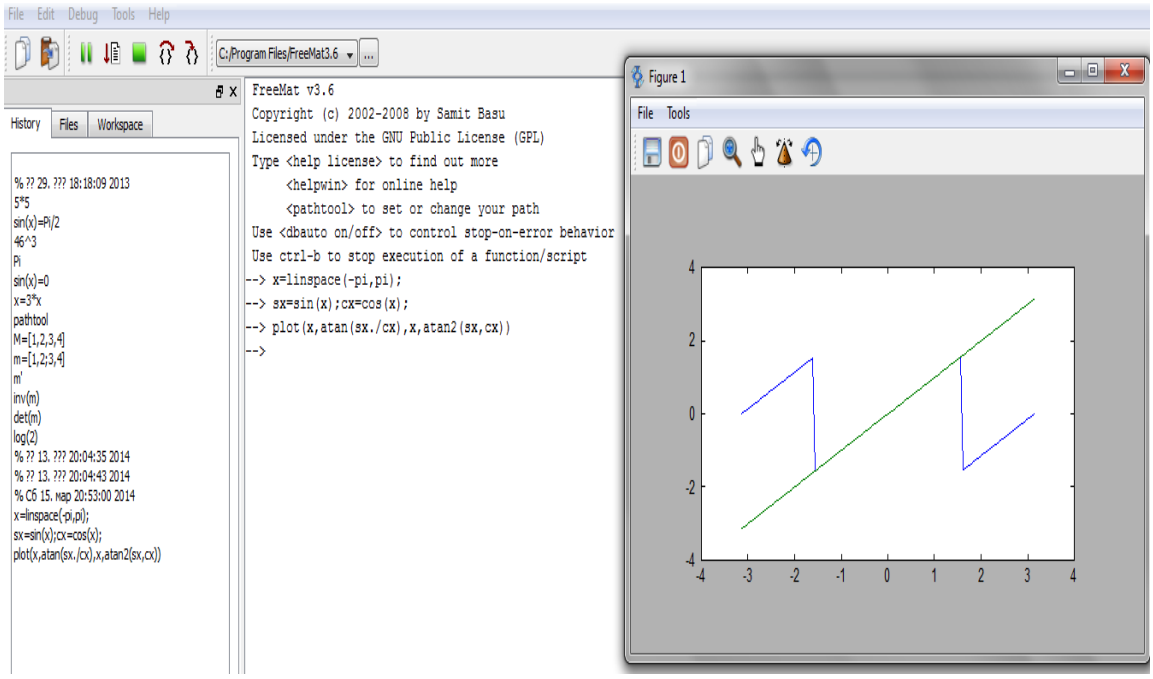


Рис. 2.1.10. График функции z = atan2(y,x)

Чтобы построить график надо:

1) указать в каком промежутке должна находиться функция;
2) написать саму функцию;
3) с помощью команды plot строим функцию.

Например,

```
x=linspace(-1,1);

t=cos(x);

plot(t);
```

2.2. Графическая визуализация с помощью 2D в системе FreeMat

Визуализация с помощью 2D графики широко используется в технике представления различных сигналов. Некоторые идеализированные периодические сигналы можно представить как комбинацию из тригонометрических и обратных тригонометрических функций - рис. 2.2.1. Здесь показано построение четырех графиков в одном окне.

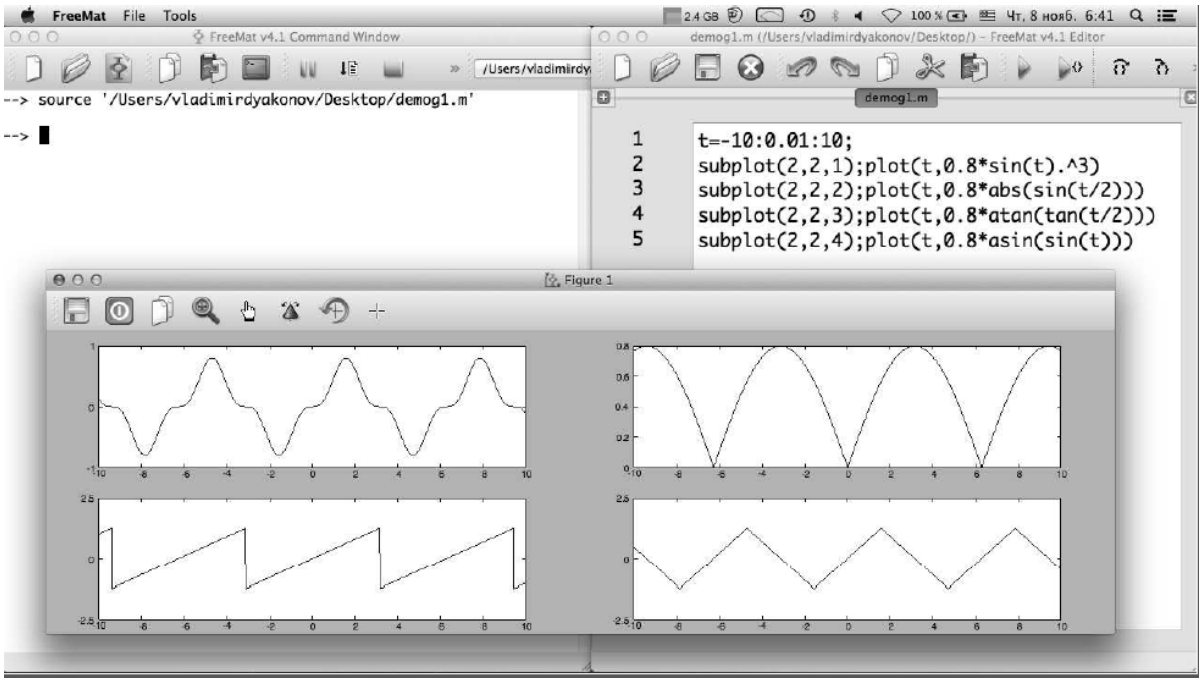


Рис. 2.2.1. Примеры моделирования 4 сигналов и построение их временных зависимостей.

Важным достоинством такого представления сигналов является однозначность и хорошее представление периодичности в пределах изменения времени от -∞ до +∞. Это очень важно при имитационном моделировании сигналов. Подобная техника применяется и при графической визуализации сложных сигналов, например с частотной (FM) и амплитудной (AM) модуляцией - рис. 2.2.2.

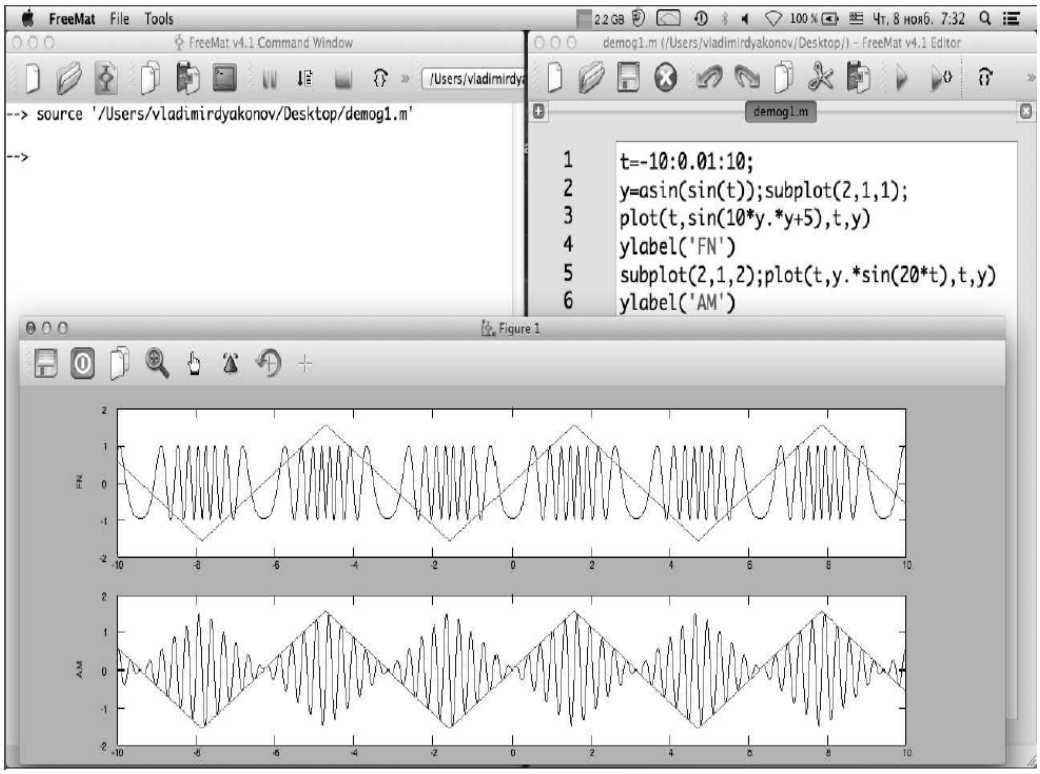


Рис. 2.2.2. Графики сигналов с частотной и амплитудной модуляцией

Графики функций комплексного переменного Z и случайных чисел (функция rand) на плоскости (рис. 2.2.2) строятся следующим script-файлом, представляющим четыре окна от figure1 до figure4:

```
figure(1)

x = linspace(-1,1,512)'*ones(1,512); y = x';

Z = exp(-(x.^2+y.^2)/0.3); image(Z);

colormap(copper);

figure(2)

x = rand(512);

x((-64:63)+256,(-128:127)+256) = 1.0;

image(x); colormap(gray)
```

```
figure(3)

x = linspace(-1,1,512)'*ones(1,512);

y = x'; Z = exp(-(x.^2+y.^2)/0.3);

image(Z);

figure(4)

x = linspace(-1,1,512)'*ones(1,512);

y = x'; Z = exp(-(x.^2+y.^2)/0.3);

image(Z); colormap(gray);
```

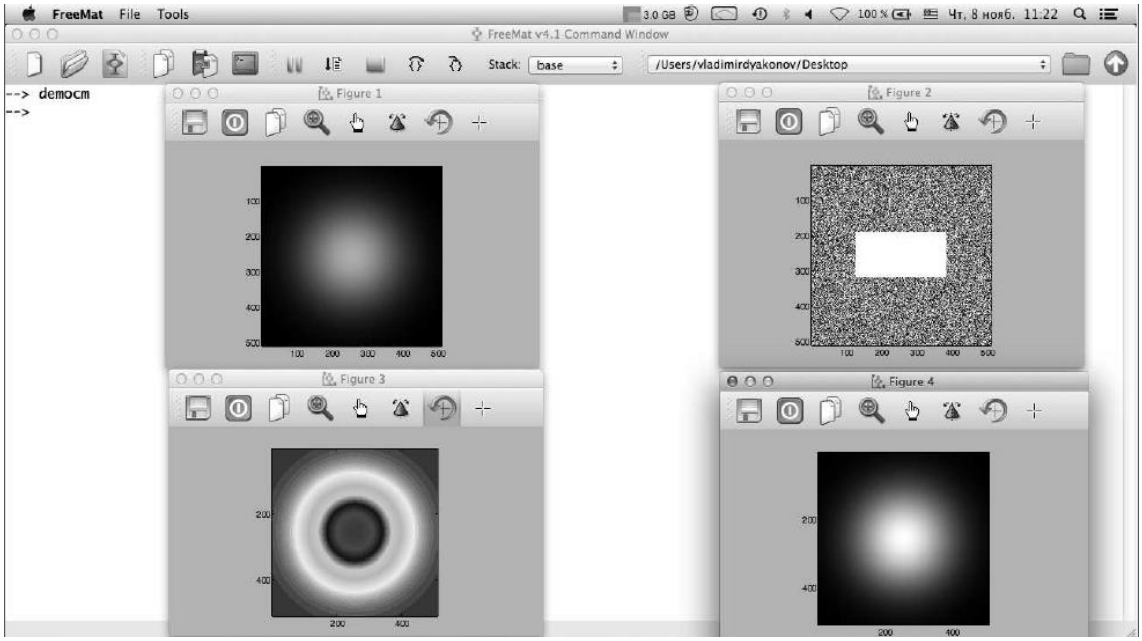


Рис. 2.2.3. Различные графики на плоскости

Такой способ позволяет строить разные графики с индивидуальной установкой цветов в каждом из них. В прямых апострофах можно в строковом виде задавать различные опции, меняя стиль и цвет линий, вводя титульные и иные надписи и т. д.

2.3. Трёхмерная графика

Графики функций двух переменных представляют из себя куски поверхностей, нависающие над областями определения функций. Отсюда ясно, что изображение графиков функций двух переменных требует реализации "трёхмерной графики" на плоском экране дисплея компьютера.

Высокоуровневая графическая подсистема FreeMat автоматически реализует трёхмерную графику без специальных усилий со стороны пользователя. Пусть в точке с координатами x1,y1 вычислено значение функции z=f(x,y) и оно равно z1. В некоторой другой точке (то есть при другом значении аргументов) x2,y2 вычисляют значение функции z2. Продолжая этот процесс, получают массив (набор) точек (x1,y1,z1), (x2,y2,z2), … (xN,yN,zN) в количестве N штук, расположенных в трёхмерном пространстве. Специальные функции системы FreeMat проводят через эти точки гладкие поверхности и отображают их проекции на плоский дисплей компьютера.

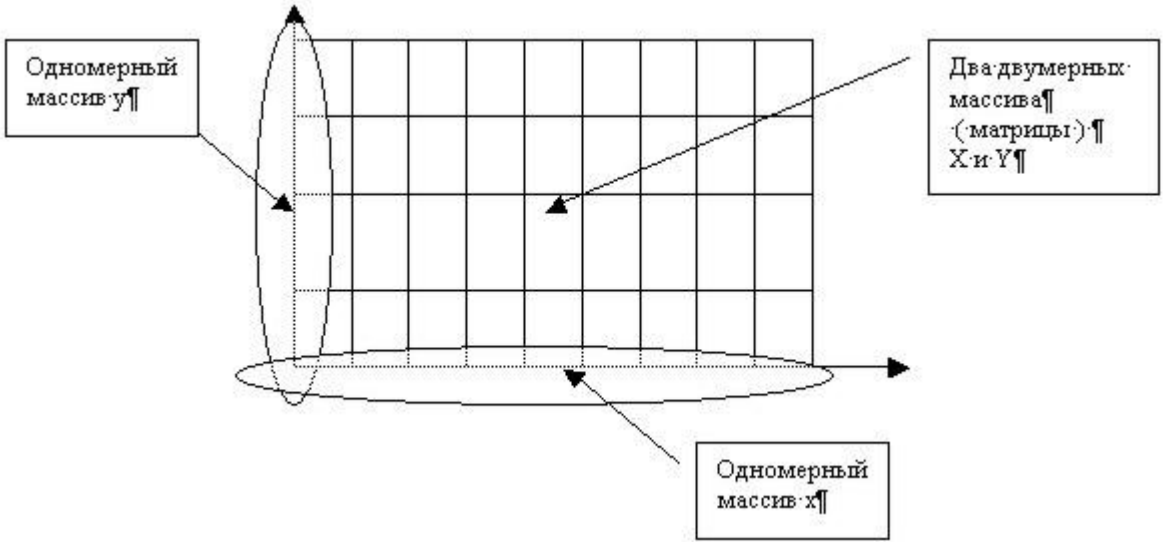
Чаще всего точки аргументов расположены в области определения функции регулярно в виде прямоугольной сетки (то есть матрицы). Такая сетка точек порождает две матрицы одной и той же структуры: первая матрица содержит значения первых координат этих точек (x - координат), а вторая матрица содержит значения вторых координат (y - координат). Обозначим первую матрицу как x, а вторую - как y. Есть ещё и третья матрица - матрица значений функции z=f(x,y) при этих аргументах. Эту матрицу обозначим буквой z.

Простейшей функцией построения графика функции двух переменных в системе FreeMat является функция

plot3 (x, y, z)

где x, y и z - матрицы одинаковых размеров.

В системе FreeMat имеется специальная функция для получения двумерных массивов x и y по одномерным массивам x, y.



Пусть по оси x задан диапазон значений в виде вектора

u = -2 : 0.1 : 2

а по оси y этот диапазон есть

v = -1 : 0.1 : 1

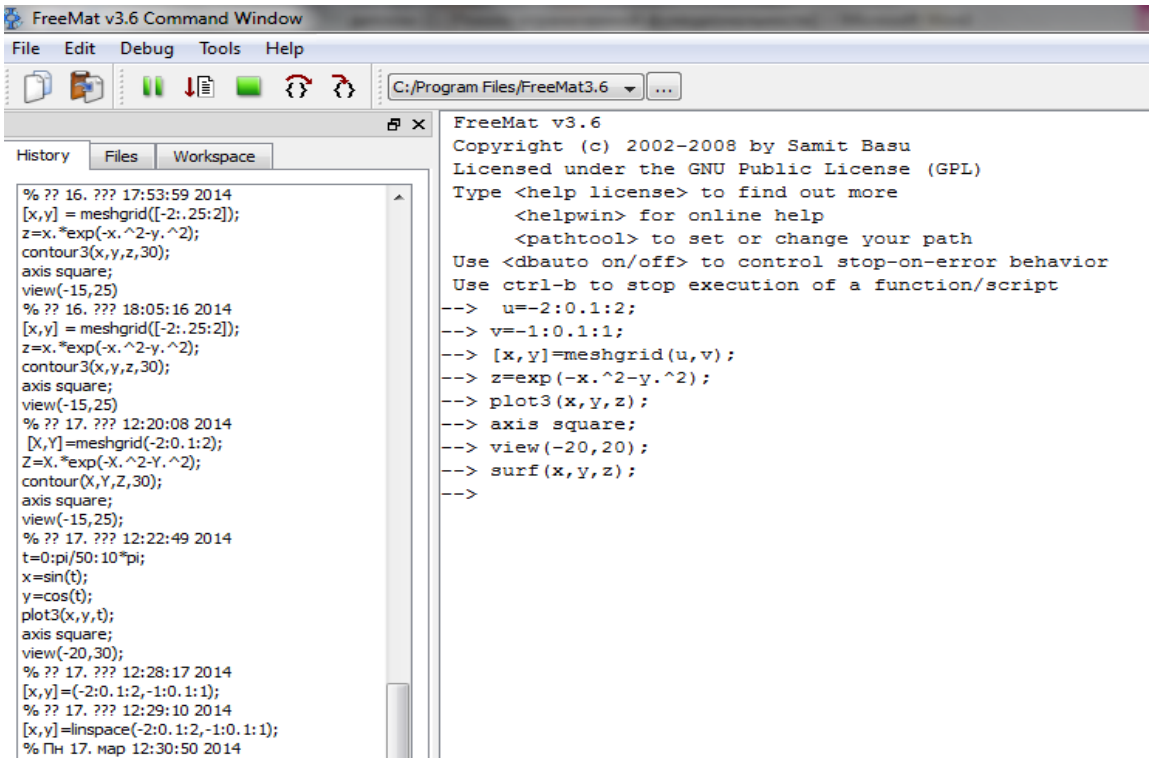
Для получения матриц X и Y, представляющих первые и вторые координаты получающейся прямоугольной сетки точек используют специальную функцию системы FreeMat:

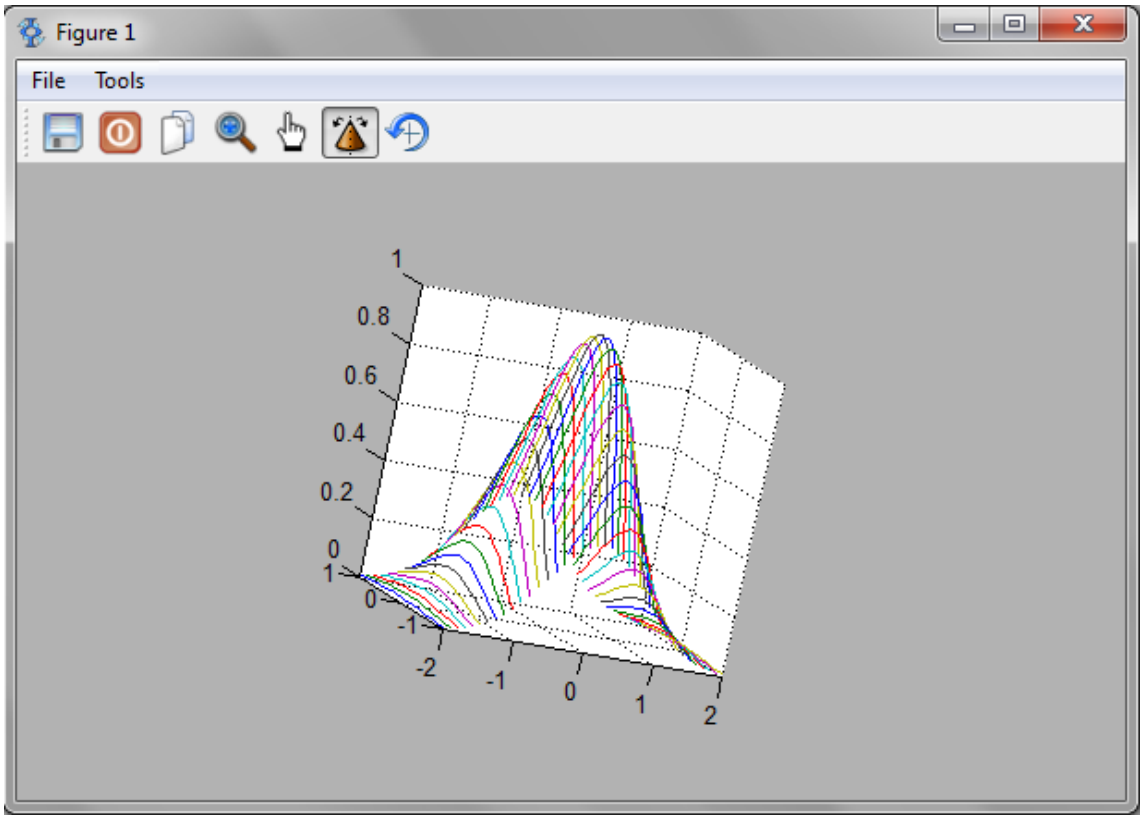
[x, y] = meshgrid(u, v)

Как мы видим, эта функция получает на входе два одномерных массива (вектора), представляющие массивы точек на осях координат, и возвращает сразу два искомых двумерных массива. На прямоугольной сетке точек вычисляем значения функции, например функции exp:

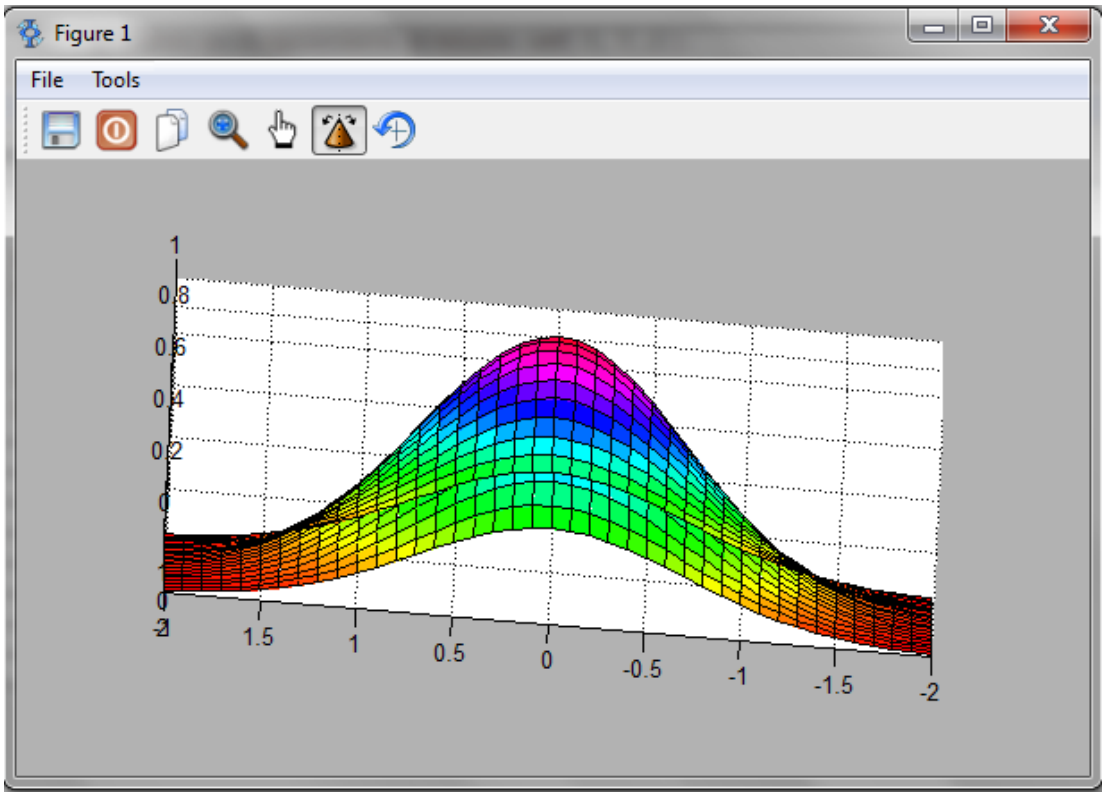
Z = exp(- x.^2 - y.^2)

Наконец, применяя описанную выше функцию plot3, получаем следующее изображение трёхмерного графика этой функции:

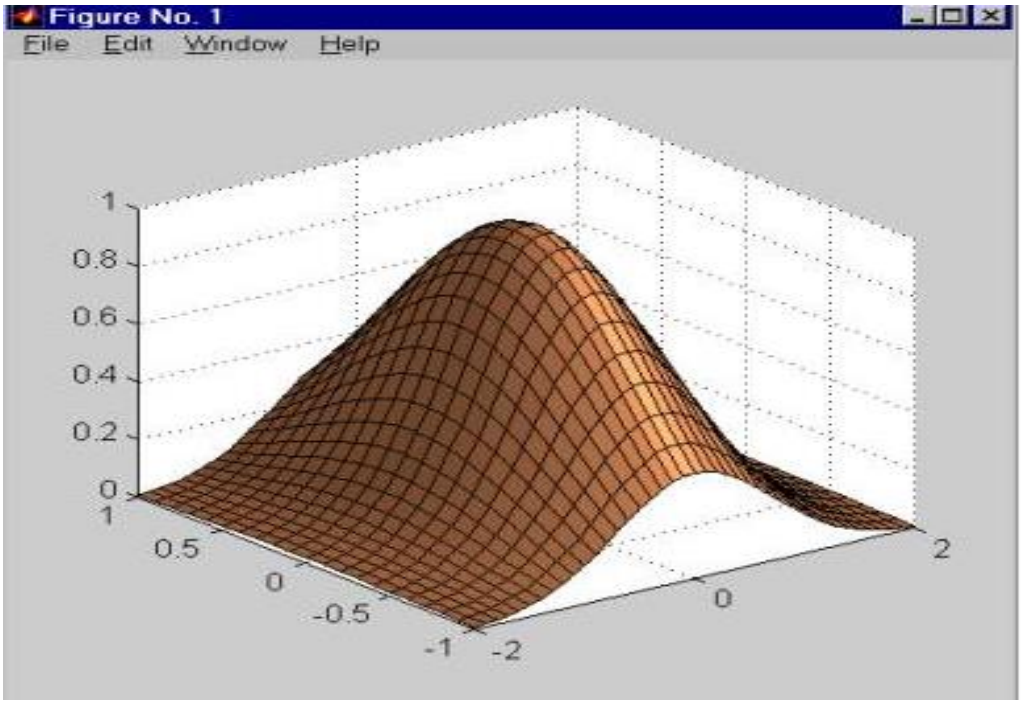




Помимо этой простейшей функции система FreeMat располагает ещё рядом функций, позволяющих добиваться большей реалистичности в изображении трёхмерных графиков. Это функции mesh, surf и surfl.



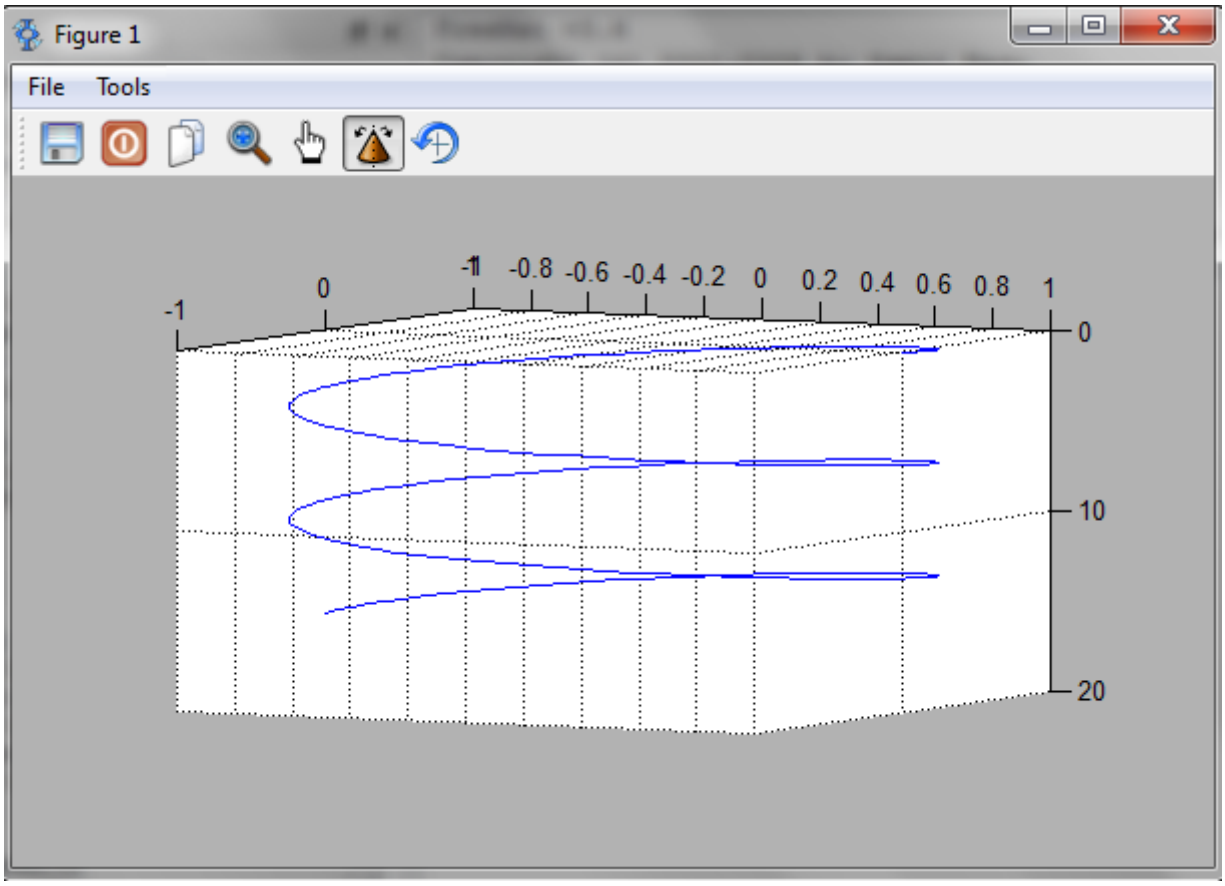
Функция mesh соединяет вычисленные соседние точки поверхности графика отрезками прямых и показывает в графическом окне системы FreeMat плоскую проекцию такого объёмного "каркасно-ребристого" (по-английски зовётся wireframe mesh) тела. Вместо ранее показанного при помощи функции plot3 графика функции.



Из этого рисунка видно, что функция plot3 строит график в виде набора линий в пространстве, каждая из которых является сечением трёхмерной поверхности плоскостями, параллельными плоскости yOz. По-другому можно сказать, что каждая линия получается из отрезков прямых, соединяющих набор точек, координаты которых берутся из одинаковых столбцов матриц X, Y и Z. То есть, первая линия соответствует первым столбцам матриц X, Y Z; вторая линия - вторым столбцам этих матриц и так далее.

Для построения трёхмерных линий, задаваемых параметрически применяется другая форма вызова функции plot3:

```
--> t = linspace (0,5*pi,200);
--> x = cos(t); y = sin(t); z = t;
--> plot3(x,y,z);
```



2.4. Отображение трехмерных графиков

Программа FreeMat обладает рядом инструментов для визуализации графиков в трехмерном пространстве. Такие задачи обычно возникают при отображении графиков функций типа $z = f(x, y)$.

В самом простом случае, для визуализации графика в трехмерных координатных осях, используется функция

plot3(X,Y,Z);

которая в качестве первых двух аргументов принимает матрицы с координатами точек по осям Ox и Oy соответственно, а в качестве третьего аргумента передается матрица значений точек по оси Oz. Рассмотрим работу данной функции на примере отображения графика функции

$z(x,y)=\exp\left(-x^2-y^2\right)$,

при $x=-1,-0.9,\ldots,1$ и $y=-2,-1.9,\ldots,2$.

Сформируем матрицы X и Y, содержащие координаты точек данного графика по осям Ox и Oy соответственно. Данные матрицы нужны для того, чтобы функция plot3() «знала» какие реальные координаты соответствуют точке Z(i,j) матрицы значений по оси Oz. Для этого достаточно взять i-ю и j-ю компоненту матриц

$x=X(i,j)$

$y=Y(i,j)$

Формирование матриц X и Y можно осуществить с помощью функции

```
[X,Y]=meshgrid(x,y);
```

языка MatLab. Здесь x и y – одномерные векторы значений координат по осям Ox и Oy соответственно, которые можно сформировать как

```
x= -1:0.1:1;             % координаты точек по оси Ox
y= -2:0.1:2;             % координаты точек по оси Oy
```

и, затем, вычислить матрицы

```
[X,Y]=meshgrid(x,y);    % матрицы координат точек по осям Ox и Oy
```

В результате, матрицы X и Y будут содержать следующие первые восемь значений по строкам и столбцам:

Матрица X:

-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3
-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3
-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3
-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3
-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3
-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3
-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3
-1	-0,9	-0,8	-0,7	-0,6	-0,5	-0,4	-0,3

Матрица Y:

-2	-2	-2	-2	-2	-2	-2	-2
-1,9	-1,9	-1,9	-1,9	-1,9	-1,9	-1,9	-1,9
-1,8	-1,8	-1,8	-1,8	-1,8	-1,8	-1,8	-1,8
-1,7	-1,7	-1,7	-1,7	-1,7	-1,7	-1,7	-1,7
-1,6	-1,6	-1,6	-1,6	-1,6	-1,6	-1,6	-1,6
-1,5	-1,5	-1,5	-1,5	-1,5	-1,5	-1,5	-1,5
-1,4	-1,4	-1,4	-1,4	-1,4	-1,4	-1,4	-1,4
-1,3	-1,3	-1,3	-1,3	-1,3	-1,3	-1,3	-1,3

Используя данные матрицы, можно вычислить значения матрицы Z, следующим образом:

Z=exp(-X.^2-Y.^2);

и отобразить результат на экране

plot3(X,Y,Z);

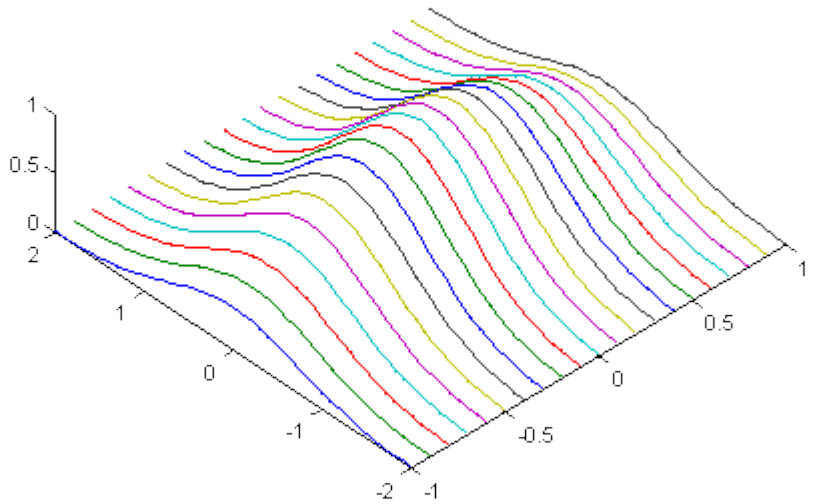


Рис. 2.4.1. Пример отображения графика с помощью функции plot3()

Из приведенного рисунка видно, что функция plot3() отображает график в виде набора линий, каждая из которых соответствует сечению графика функции $z(x,y) = \exp\left(-x^2 - y^2\right)$ вдоль оси Oy.

Такое представление графика не всегда удобно, т.к. набор одномерных не дает полное представление о характере двумерной плоскости. Более лучшей визуализации можно получить, используя функцию

mesh(X,Y,Z); % отображение графика в виде сетки

В результате получим следующий вид трехмерного графика (рис. 2.4.2).

Благодаря использованию функции mesh() получается график, образованный интерполяцией точек массивов X, Y и Z линиями по осям Ox и Oy. Кроме того, цветом указывается уровень точки по оси Oz: от самого малого значения (синего) до самого большого (красного) и производится удаление «невидимых» линий. Это позволяет лучше визуально оценивать структуру трехмерного графика по сравнению с функцией plot3(). Если же необходимо отобразить «прозрачный» график, то следует выключить режим удаления «невидимых» линий:

hidden off; % скрытые линии рисуются

В системе MatLab предусмотрена функция визуализации непрерывной поверхности в трехмерных осях

surf(X,Y,Z); % отображение непрерывной поверхности

В результате получается график, представленный на рис. 2.4.3.

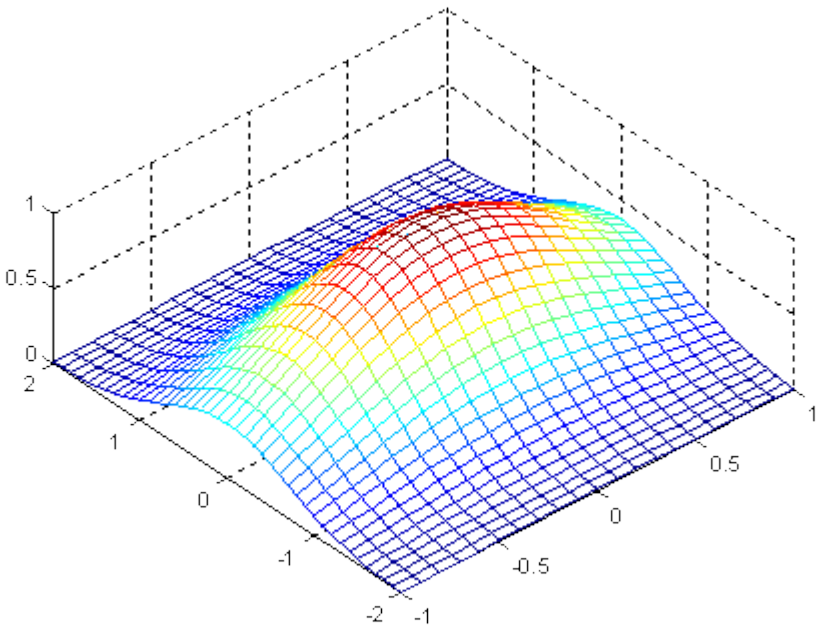


Рис. 2.4.2. Результат работы функции mesh()

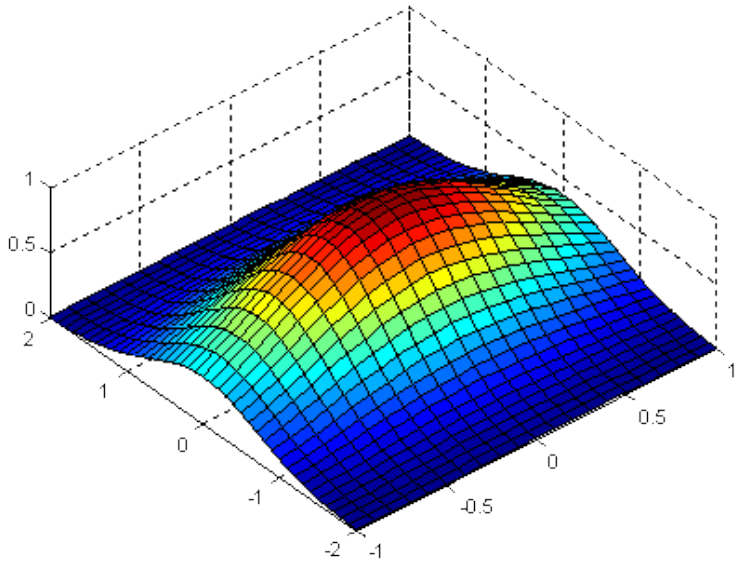


Рис. 2.4.3. Результат работы функции surf()

Наконец, для трёхмерных графиков существует возможность изменять точку их обзора, т.е. положение виртуальной камеры с помощью функции

view([az el]);

где az – угол азимута; el – угол возвышения. Изменение первого угла означает вращение плоскости xOy вокруг оси Oz против часовой стрелки. Угол возвышения есть угол между направлением на камеру и плоскостью xOy.

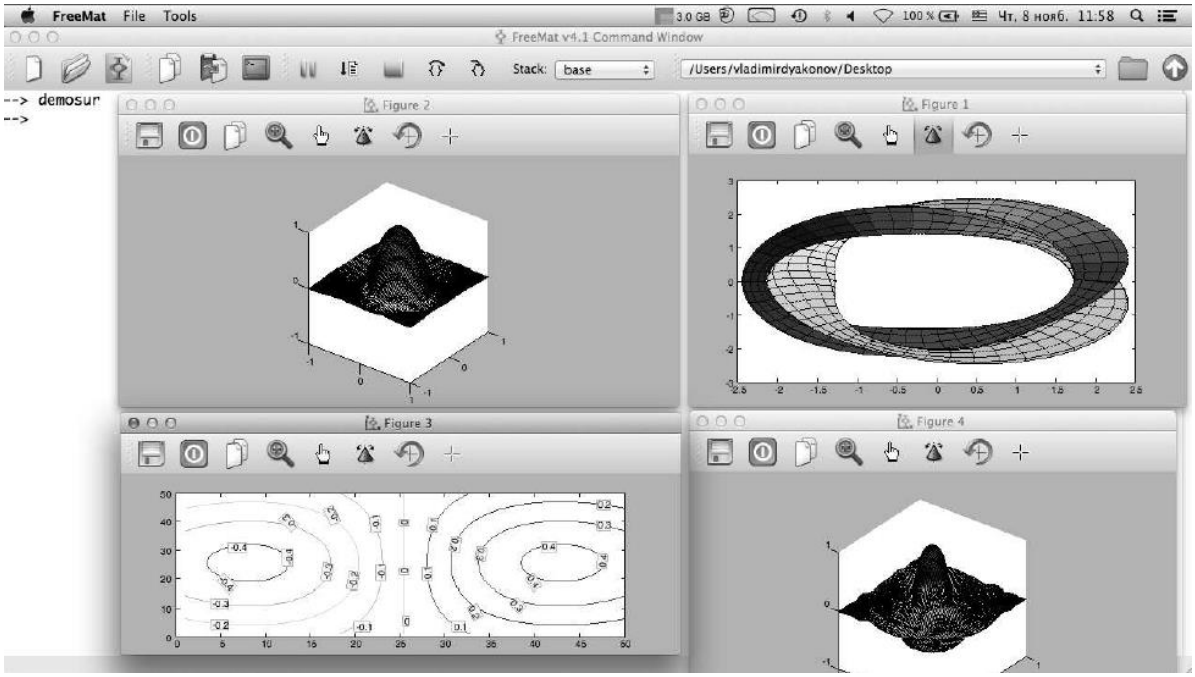


Рис. 2.4.4. Примеры построения 3D- графиков поверхностей и объемных фигур

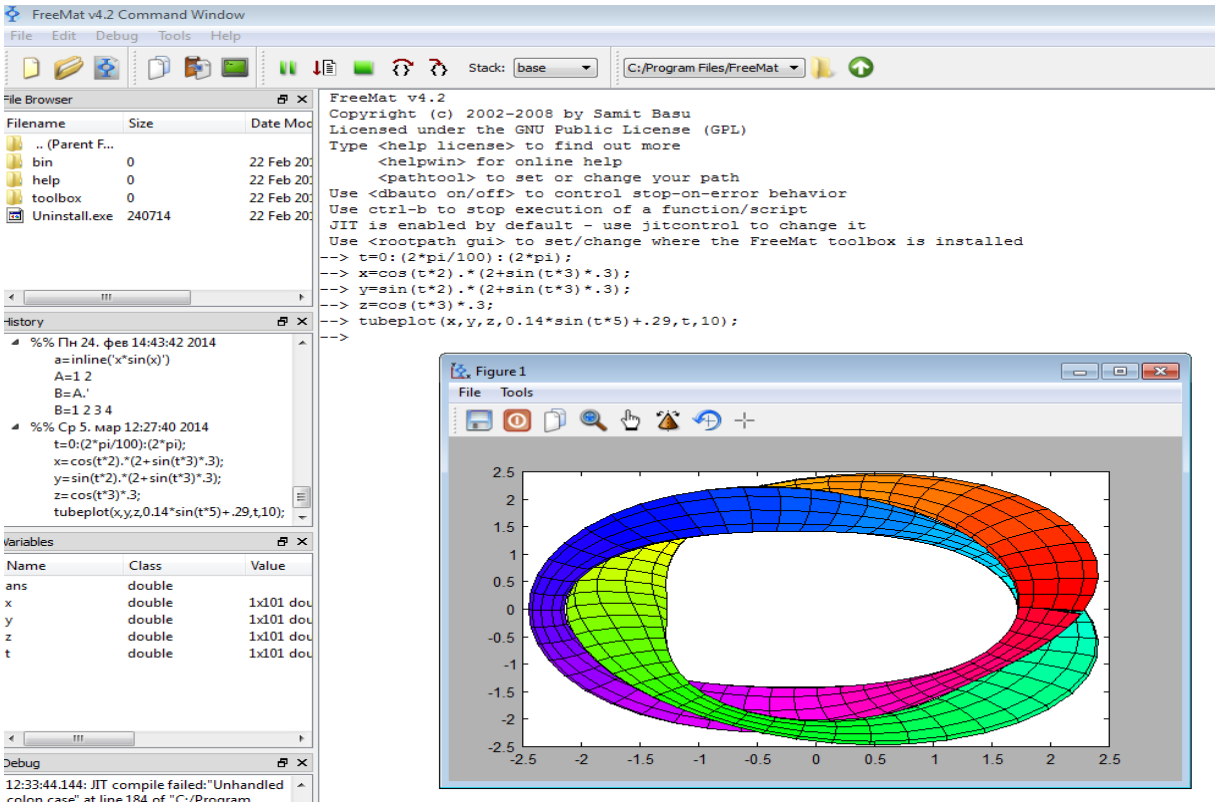
Построение еще четырех 3D графиков (рис. 2.4.4) обеспечивает следующая script-программа:

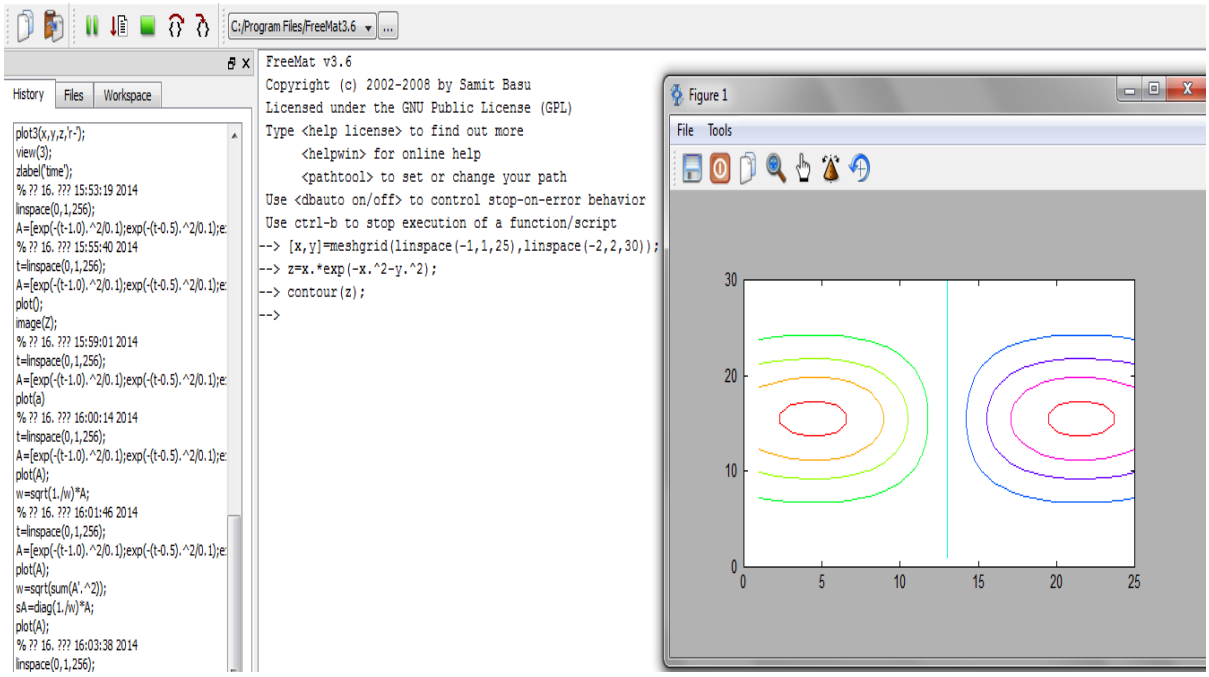
```
figure(1); t=0:(2*pi/100):(2*pi);

x=cos(t*2).*(2+sin(t*3)*.3); y=sin(t*2).*(2+sin(t*3)*.3);

z=cos(t*3)*.3; tubeplot(x,y,z,0.14*sin(t*5)+.29,t,10);

figure(2); x = repmat(linspace(-1,1),[100,1]);

y = x';  r = x.^2+y.^2; z = exp(-r*3).*cos(5*r);

c = r; surf(x,y,z,c); axis equal; view(3)

figure(3); [x,y] = meshgrid(linspace(-1,1,50));
```

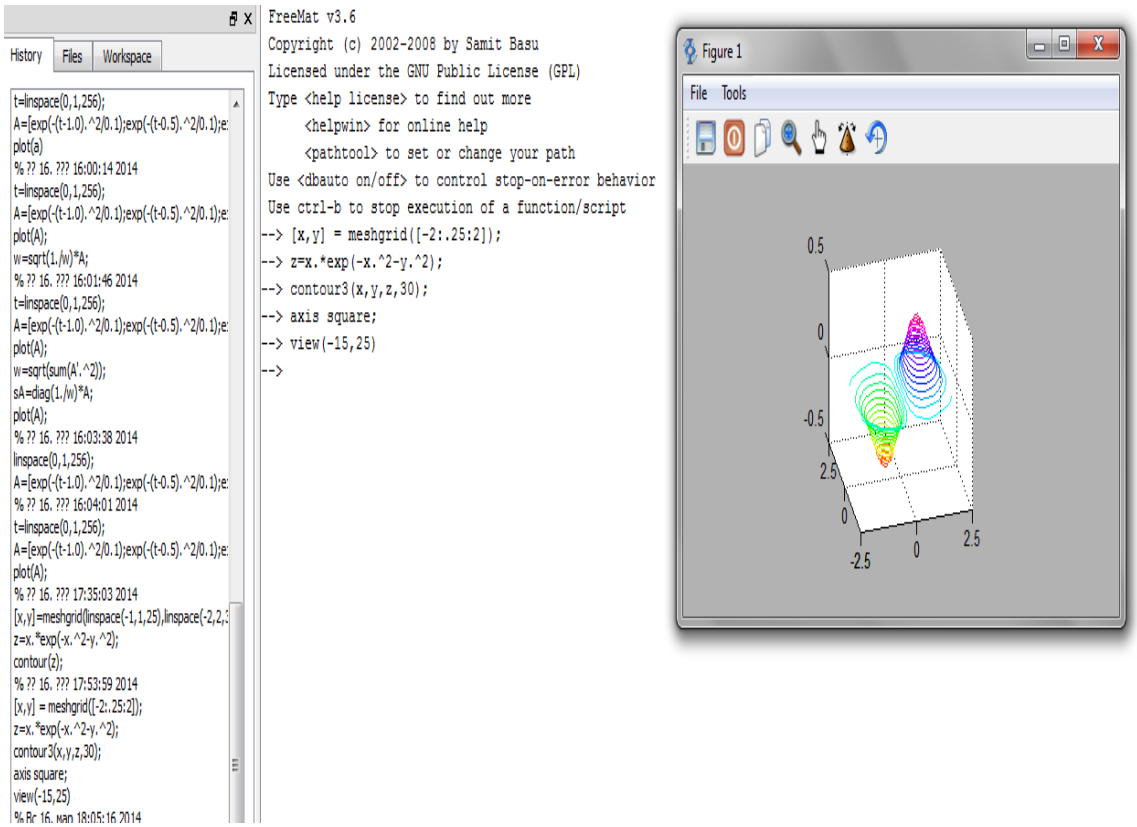
```
z = x.*exp(-(x.^2+y.^2)); h = contour(z);

clabel(h,'backgroundcolor',[1,1,.6],'edgecolor',[.7,.7,.7]);

figure(4); x = repmat(linspace(-1,1),[100,1]);

y = x'; r = x.^2+y.^2; z = exp(-r*3).*cos(5*pi*r);

surf(x,y,z); axis equal; view(3)
```

Из этих примеров видны обширные возможности 2D и 3D графики системы FreeMat.

Ниже приведены несколько примеров построения 3D- графиков. Обратите внимание, как определен диапазон и вместо plot3() можно использовать так же tubeplot() и contour3().


FreeMat v4.2 Command Window
File Edit Debug Tools Help
Stack: base
C:/Program Files/FreeMat
File Browser
Filename Size Date Mod
.. (Parent F...
bin 0 22 Feb 20
help 0 22 Feb 20
toolbox 0 22 Feb 20
Uninstall.exe 240714 22 Feb 20
History
%% Пн 24. фев 14:43:42 2014
a=inline('x*sin(x)')
A=1 2
B=A.'
B=1 2 3 4
%% Ср 5. мар 12:27:40 2014
t=0:(2*pi/100):(2*pi);
x=cos(t*2).*(2+sin(t*3)*.3);
y=sin(t*2).*(2+sin(t*3)*.3);
z=cos(t*3)*.3;
tubeplot(x,y,z,0.14*sin(t*5)+.29,t,10);
Variables
Name Class Value
ans double
x double 1x101 dou
y double 1x101 dou
z double 1x101 dou
t double 1x101 dou
Debug
12:33:44.144: JIT compile failed:"Unhandled
FreeMat v4.2
Copyright (c) 2002-2008 by Samit Basu
Licensed under the GNU Public License (GPL)
Type <help license> to find out more
<helpwin> for online help
<pathtool> to set or change your path
Use <dbauto on/off> to control stop-on-error behavior
Use ctrl-b to stop execution of a function/script
JIT is enabled by default - use jitcontrol to change it
Use <rootpath gui> to set/change where the FreeMat toolbox is installed
--> t=0:(2*pi/100):(2*pi);
--> x=cos(t*2).*(2+sin(t*3)*.3);
--> y=sin(t*2).*(2+sin(t*3)*.3);
--> z=cos(t*3)*.3;
--> tubeplot(x,y,z,0.14*sin(t*5)+.29,t,10);
-->
Figure 1
File Tools
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5


C:/Program Files/FreeMat3.6
History
Files
Workspace
plot3(x,y,z,'r-');
view(3);
zlabel('time');
% ?? 16. ??? 15:53:19 2014
linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
% ?? 16. ??? 15:55:40 2014
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot();
image(Z);
% ?? 16. ??? 15:59:01 2014
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot(a)
% ?? 16. ??? 16:00:14 2014
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot(A);
w=sqrt(1./w)*A;
% ?? 16. ??? 16:01:46 2014
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot(A);
w=sqrt(sum(A'.^2));
sA=diag(1./w)*A;
plot(A);
% ?? 16. ??? 16:03:38 2014
linspace(0,1,256);
FreeMat v3.6
Copyright (c) 2002-2008 by Samit Basu
Licensed under the GNU Public License (GPL)
Type <help license> to find out more
<helpwin> for online help
<pathtool> to set or change your path
Use <dbauto on/off> to control stop-on-error behavior
Use ctrl-b to stop execution of a function/script
--> [x,y]=meshgrid(linspace(-1,1,25),linspace(-2,2,30));
--> z=x.*exp(-x.^2-y.^2);
--> contour(z);
-->
Figure 1
File
Tools
30
20
10
0
0
5
10
15
20
25


History
Files
Workspace
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot(a)
% ?? 16. ??? 16:00:14 2014
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot(A);
w=sqrt(1./w)*A;
% ?? 16. ??? 16:01:46 2014
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot(A);
w=sqrt(sum(A'.^2));
sA=diag(1./w)*A;
plot(A);
% ?? 16. ??? 16:03:38 2014
linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
% ?? 16. ??? 16:04:01 2014
t=linspace(0,1,256);
A=[exp(-(t-1.0).^2/0.1);exp(-(t-0.5).^2/0.1);e
plot(A);
% ?? 16. ??? 17:35:03 2014
[x,y]=meshgrid(linspace(-1,1,25),linspace(-2,2,
z=x.*exp(-x.^2-y.^2);
contour(z);
% ?? 16. ??? 17:53:59 2014
[x,y] = meshgrid([-2:.25:2]);
z=x.*exp(-x.^2-y.^2);
contour3(x,y,z,30);
axis square;
view(-15,25)
FreeMat v3.6
Copyright (c) 2002-2008 by Samit Basu
Licensed under the GNU Public License (GPL)
Type <help license> to find out more
<helpwin> for online help
<pathtool> to set or change your path
Use <dbauto on/off> to control stop-on-error behavior
Use ctrl-b to stop execution of a function/script
--> [x,y] = meshgrid([-2:.25:2]);
--> z=x.*exp(-x.^2-y.^2);
--> contour3(x,y,z,30);
--> axis square;
--> view(-15,25)
-->
Figure 1
File
Tools
0.5
0
-0.5
2.5
0
-2.5
0
2.5

2.5. Оформление графиков

Пакет FreeMat позволяет отображать графики с разным цветом и типом линий, показывать или скрывать сетку на графике, выполнять подпись осей и графика в целом, создавать легенду и многое другое. В данном параграфе рассмотрим наиболее важные функции, позволяющие делать такие оформления на примере двумерных графиков.

Функция plot() позволяет менять цвет и тип отображаемой линии. Для этого, используются дополнительные параметры, которые записываются следующим образом:

plot(<x>, <y>, <’цвет линии, тип линии, маркер точек’>);

Обратите внимание, что третий параметр записывается в апострофах и имеет обозначения, приведенные в таблицах 3.1-3.3. Маркеры, указанные ниже записываются подряд друг за другом, например,

‘ko’ – на графике отображает черными кружками точки графика,

‘ko-‘ – рисует график черной линией и проставляет точки в виде кружков.

Табл. 2.5.1. Обозначение цвета линии графика

Маркер	Цвет линии
c	голубой
m	фиолетовый
y	желтый
r	красный
g	зеленый
b	синий
w	белый
k	черный

Табл. 2.5.2. Обозначение типа линии графика

Маркер	Цвет линии
-	непрерывная
--	штриховая
:	пунктирная
-.	штрих-пунктирная

Табл. 2.5.3. Обозначение типа точек графика

Маркер	Цвет линии
.	точка
+	плюс
*	звездочка
o	кружок
x	крестик

Ниже показаны примеры записи функции plot() с разным набором маркеров.

```
x = 0:0.1:2*pi;
y = sin(x);
subplot(2,2,1); plot(x,y,'r-');
subplot(2,2,2); plot(x,y,'r-',x,y,'ko');
subplot(2,2,3); plot(y,'b--');
subplot(2,2,4); plot(y,'b--+');
```

Результат работы фрагмента программы приведен на рис. 2.5.3. Представленный пример показывает, каким образом можно комбинировать маркеры для достижения требуемого результата. Так же наглядно видно к каким визуальным эффектам приводят разные маркеры, используемые в программе. Следует особо отметить, что в

четвертой строчке программы по сути отображаются два графика: первый рисуется красным цветом и непрерывной линией, а второй черными кружками заданных точек графика. Остальные варианты записи маркеров очевидны.

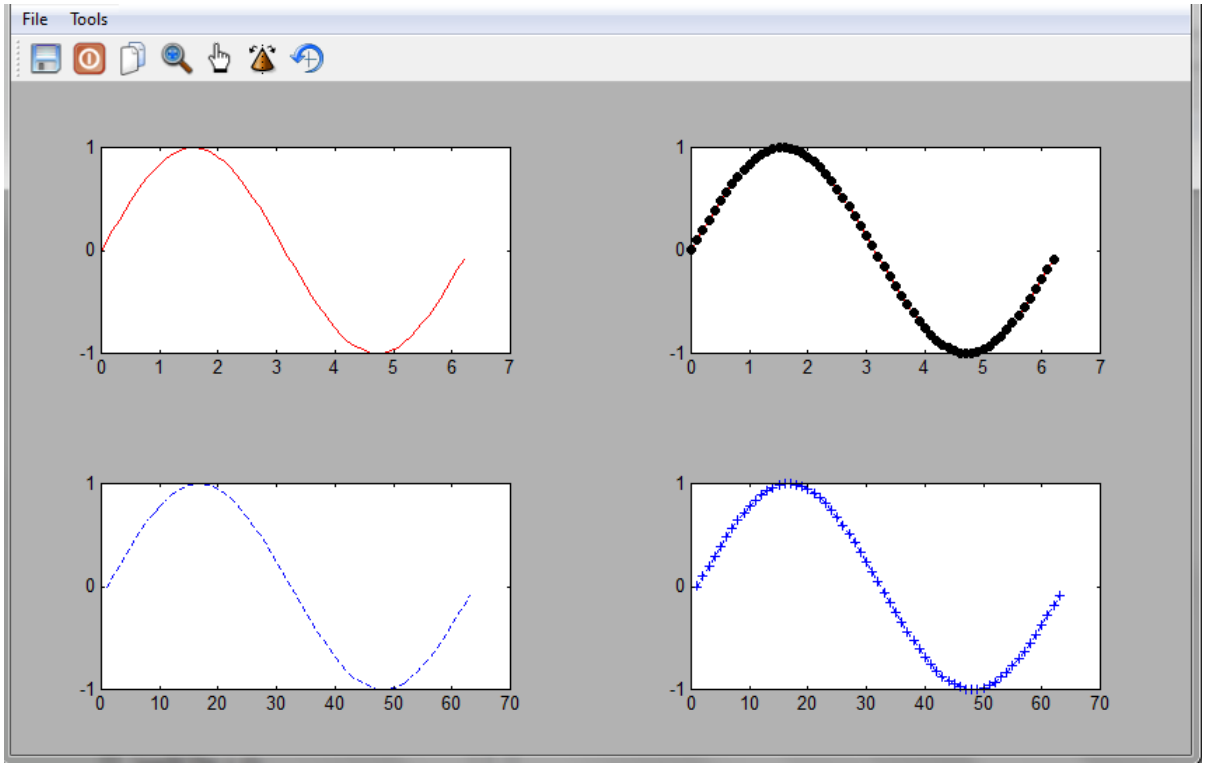


Рис. 2.5.3. Примеры отображения графиков с разными типами маркеров

Из примеров рис. 2.5.3 видно, что масштаб графиков по оси Ox несколько больше реальных значений. Дело в том, что система FreeMat автоматически масштабирует систему координат для полного представления данных. Однако такая автоматическая настройка не всегда может удовлетворять интересам пользователя. Иногда требуется выделить отдельный фрагмент графика и только его показать целиком. Для этого используется функция axis() языка FreeMat, которая имеет следующий синтаксис:

axis([xmin, xmax, ymin, ymax]), где название указанных параметров говорят сами за себя.

Воспользуемся данной функцией для отображения графика функции синуса в пределах от 0 до 2π:

```
x = 0:0.1:2*pi;
y = sin(x);
subplot(1,2,1);
plot(x,y);
axis([0 2*pi -1 1]);
subplot(1,2,2);
plot(x,y);
axis([0 pi 0 1]);
```

Из результата работы программы (рис. 2.5.4) видно, что несмотря на то, что функция синуса задана в диапазоне от 0 до 2π, с помощью функции axis() можно отобразить как весь график, так и его фрагмент в пределах от 0 до .

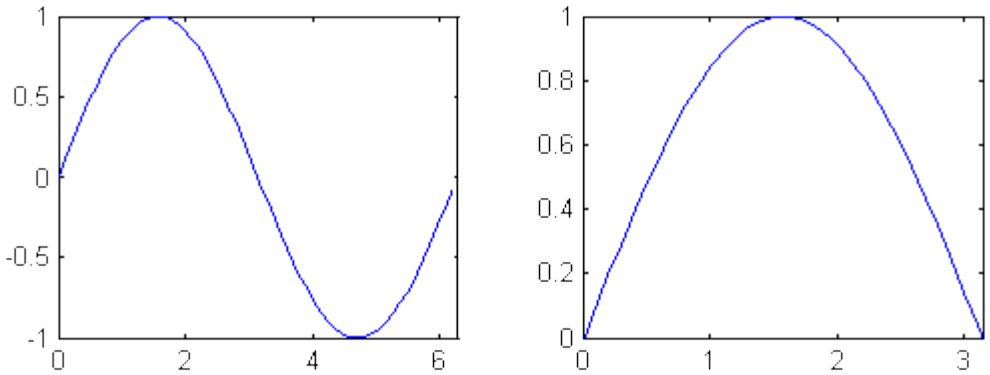


Рис. 2.5.4. Пример работы функции axis()

В заключении данного параграфа рассмотрим возможности создания подписей графиков, осей и отображения сетки на графике. Для этого используются функции языка FreeMat, перечисленные в табл. 2.5.4.

Таблица 2.5.4. Функции оформления графиков

Название	Описание
grid [on, off]	Включает/выключает сетку на графике
title(‘заголовок графика’)	Создает надпись заголовка графика
xlabel(‘подпись оси Ox’)	Создает подпись оси Ox
ylabel(‘подпись оси Oy’)	Создает подпись оси Oy
text(x,y,’текст’)	Создает текстовую надпись в координатах (x,y).

Рассмотрим работу данных функций в следующем примере:

```
x = 0:0.1:2*pi;
y = sin(x);
plot(x,y);
axis([0 2*pi -1 1]);
grid on;
title('The graphic of sin(x) function');
xlabel('The coordinate of Ox');
ylabel('The coordinate of Oy');
text(3.05,0.16,'\leftarrow sin(x)');
```

Из результата работы данной программы, представленного на рис. 2.5.5, видно каким образом работают функции создания подписей на графике, а также отображение сетки графика.

Таким образом, используя описанный набор функций и параметров, можно достичь желаемого способа оформления графиков в системе FreeMat.

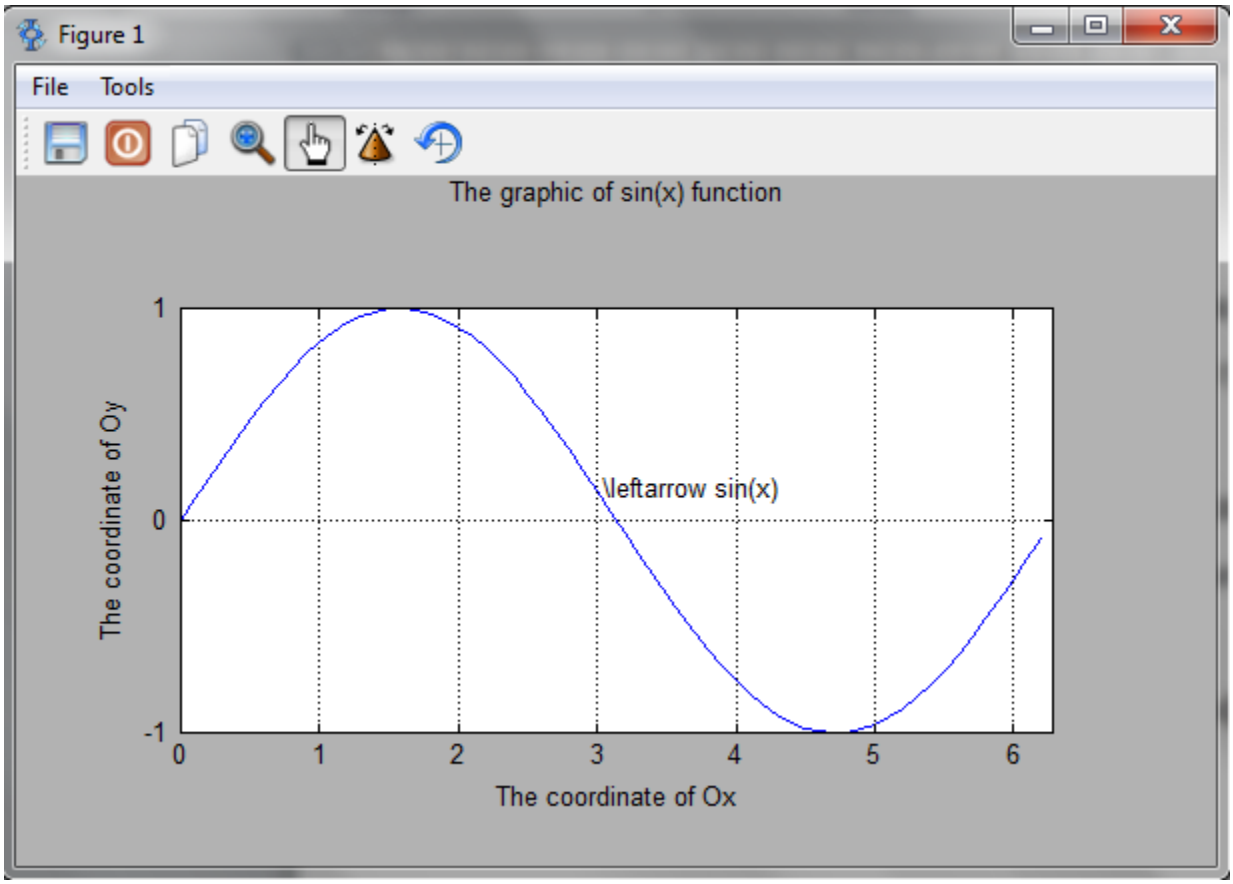


Рис. 2.5.5. Пример работы функций оформления графика

III. КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМЕ «ОСНОВЫ РАБОТЫ В FREEMAT»

1. С помощью каких команд можно ввести вектор, матрицу?
2. Какими двумя командами можно сложить два вектора одинаковой размерности (2 матрицы)?
3. Какими двумя командами можно вычислить произведение двух матриц (или матрицы на вектор)?
4. Какая матрица называется обратной и какими способами она вычисляется в *FreeMat*?
5. Перечислите и объясните действие операторов, используемых при вычислениях с массивами.
6. Опишите действие операций отношения.
7. Опишите действие логических операций.
8. Как построить несколько графиков в одной системе координат?
9. Как построить графики в разных подобластях одного графического окна?
10. Как изменить цвет и стиль линий на графиках?
11. Как сделать надписи на осях, на полученном рисунке? Как сделать заголовок для графика?
12. Как построить график функции?
13. Как построить график поверхности?

Самостоятельные задания

Векторы

Для заданных векторов a и b:

1) вычислить их сумму, разность и скалярное произведение;
2) вектор a, определить его максимальный и минимальный элементы и поменять их местами;
3) упорядочить вектор b по возрастанию и убыванию;
4) переставить элементы вектора a в обратном порядке и записать результат в новый вектор;
5) найти векторное произведение a и b.

№ варианта	**Вектор a**	**Вектор b**
I	[0.5; 3; 6; -4.3; 1.2]	[3; 7; 7; 5.4; -2]
II	[-4.8; -1; -1; 0.7; 4]	[-1; -1.9; 7.1; -2; 6]
III	[1; -3.9; -2; 3; 2]	[2.7; -2.7; 4; 0.4 -6;]
IV	[-2.4; 3.3; 0; 3; -7]	[6; 0.6; 4; -3; 7]
V	[8; -5.9; -6; 0; 6.8]	[0; 2; -1.5; 7.5; -4]
VI	[5.3; 6; -7.1; 6; -4]	[7; -1.5; -9; -4.6; -2]
VII	[1.2; -4; -0.8; -	[-1; 2.2; 1; -4; -

	0.7; -2]	1.8]
VIII	[6.6; -5; -2.7; 8; 3.8]	[-1; 3.2; 4.2; -6; 1]
IX	[-1.9; 0.4; 1; 4; -3.8]	[-8; -4; -1.4; 2.8; -2.2]
X	[9; 1.7; -3; -3.8; 7.3]	[0.6; -0.4; -6.9; -2; 1]
XI	[-1.7; 1.7; 0; -3; -6.2]	[-2.3; -4; -0.2; -5; 5.5]
XII	[1.7; 3.3; -6;-1.5; 2]	[1; 1; 2.7; -6.5; -6]
XIII	[-4; 4; -2.1; 7; 0]	[6; -0.2; 8.6; 3; -3.2]
XIV	[0; 2.3; -8.1; 0; 4]	[0; -1; 4.5; 9.9; 6]
XV	[-6; 7.4; 5; 0; -6.9]	[0.5; -1; 3; 5.61; 4.09]

Матрицы

Введите матрицы

I вариант. A=$\begin{bmatrix} 8 & 4 & -6 \\ -2 & -4 & -6 \\ 6 & 4 & 4 \end{bmatrix}$; B=$\begin{bmatrix} 7 & 3 & 4 \\ 5 & -9 & -4 \\ 9 & -9 & -9 \end{bmatrix}$; C=$\begin{bmatrix} 10 & -7 & -7 \\ -3 & -11 & -1 \\ -1 & -9 & 4 \end{bmatrix}$.

II вариант. A=$\begin{bmatrix} -5 & 1 & -7 \\ 9 & -3 & 4 \\ -3 & 7 & 5 \end{bmatrix}$; B=$\begin{bmatrix} 7 & 1 & 3 \\ -6 & -6 & 9 \\ 3 & 5 & -6 \end{bmatrix}$; C=$\begin{bmatrix} -8 & 2 & -2 \\ -4 & -4 & 24 \\ 8 & -4 & 36 \end{bmatrix}$.

III вариант. A=$\begin{bmatrix} 6 & 8 & 6 \\ 10 & -10 & -2 \\ -2 & 6 & -10 \end{bmatrix}$; B=$\begin{bmatrix} -4 & 6 & -4 \\ 10 & 8 & 2 \\ 2 & -6 & 6 \end{bmatrix}$; C=$\begin{bmatrix} 4 & 4 & 4 \\ -8 & -2 & 6 \\ -2 & 2 & -8 \end{bmatrix}$.

IV вариант. $A=\begin{bmatrix}-4 & -8 & -4\\ 6 & -2 & -6\\ 4 & 2 & -8\end{bmatrix}$; $B=\begin{bmatrix}-2 & -2 & 2\\ -8 & -6 & -8\\ -4 & -10 & -10\end{bmatrix}$; $C=\begin{bmatrix}-10 & 10 & -20\\ 6 & 2 & -6\\ 2 & 6 & 2\end{bmatrix}$.

V вариант. $A=\begin{bmatrix}-9 & -9 & -5\\ -4 & 7 & 5\\ 9 & -5 & 1\end{bmatrix}$; $B=\begin{bmatrix}-5 & 1 & -7\\ 9 & -3 & -4\\ -3 & 7 & 5\end{bmatrix}$; $C=\begin{bmatrix}3 & -11 & 5\\ -8 & -5 & -3\\ 3 & -1 & 5\end{bmatrix}$.

VI вариант. $A=\begin{bmatrix}5 & -7 & -11\\ -6 & -9 & -3\\ 3 & 5 & -5\end{bmatrix}$; $B=\begin{bmatrix}1 & 7 & 3\\ 5 & -9 & -4\\ 9 & -9 & -9\end{bmatrix}$; $C=\begin{bmatrix}7 & 1 & 3\\ 3 & 5 & -6\\ 1 & 9 & 5\end{bmatrix}$.

VII вариант. $A=\begin{bmatrix}-3 & -11 & -13\\ 7 & -9 & 6\\ 5 & -3 & -1\end{bmatrix}$; $B=\begin{bmatrix}-3 & 1 & -11\\ -4 & -3 & -3\\ 1 & -7 & -3\end{bmatrix}$; $C=\begin{bmatrix}-5 & -7 & -5\\ 5 & 7 & 0\\ -3 & 7 & 1\end{bmatrix}$.

VIII вариант. $A=\begin{bmatrix}-5 & -3 & 5\\ 4 & 7 & -3\\ 1 & -11 & -1\end{bmatrix}$; $B=\begin{bmatrix}9 & -5 & 5\\ -2 & 9 & -3\\ 1 & 3 & 3\end{bmatrix}$; $C=\begin{bmatrix}3 & -3 & 1\\ -9 & -5 & 8\\ -1 & -3 & -3\end{bmatrix}$.

IX вариант. $A=\begin{bmatrix}-3 & 9 & 7\\ -6 & 5 & -9\\ -7 & -9 & -7\end{bmatrix}$; $B=\begin{bmatrix}-9 & -1 & -5\\ 1 & -7 & -2\\ 7 & 3 & -5\end{bmatrix}$; $C=\begin{bmatrix}3 & 3 & -1\\ 2 & 7 & -5\\ 7 & -7 & -5\end{bmatrix}$.

X вариант. $A=\begin{bmatrix}-1 & 5 & -1\\ -2 & 3 & -11\\ -3 & 5 & -11\end{bmatrix}$; $B=\begin{bmatrix}-1 & 9 & 7\\ -4 & 9 & -1\\ 5 & -3 & 5\end{bmatrix}$; $C=\begin{bmatrix}7 & 5 & -9\\ -8 & -5 & -1\\ 3 & -5 & 3\end{bmatrix}$.

XI вариант. $A=\begin{bmatrix}5 & -9 & -9\\ -6 & -9 & -5\\ -5 & -3 & -3\end{bmatrix}$; $B=\begin{bmatrix}9 & 3 & -9\\ 2 & 3 & -7\\ -7 & -7 & -7\end{bmatrix}$; $C=\begin{bmatrix}1 & -5 & 5\\ -6 & -9 & -5\\ 7 & 5 & -1\end{bmatrix}$.

XII вариант. $A=\begin{bmatrix}3 & -5 & 5\\ 7 & -5 & -2\\ -9 & -7 & -13\end{bmatrix}$; $B=\begin{bmatrix}-5 & -9 & 3\\ -12 & -3 & -11\\ 5 & -3 & -5\end{bmatrix}$; $C=\begin{bmatrix}-7 & 1 & 5\\ 4 & -5 & -3\\ 3 & -1 & -6\end{bmatrix}$.

XIII вариант. $A=\begin{bmatrix}1 & 5 & 2\\ 3 & -1 & -6\\ -6 & -3 & -7\end{bmatrix}$; $B=\begin{bmatrix}-2 & -3 & -1\\ 9 & -11 & 2\\ -5 & -7 & 8\end{bmatrix}$; $C=\begin{bmatrix}-3 & -9 & -4\\ 7 & 1 & -8\\ 3 & -9 & -4\end{bmatrix}$.

XIV вариант. $A=\begin{bmatrix}7 & 0 & -7\\ 3 & 1 & -4\\ -14 & -3 & 3\end{bmatrix}$; $B=\begin{bmatrix}-12 & 5 & -11\\ 4 & -5 & 3\\ 5 & -3 & 10\end{bmatrix}$; $C=\begin{bmatrix}-10 & -7 & -7\\ -8 & 2 & -6\\ -6 & -2 & 4\end{bmatrix}$.

XV вариант. A=$\begin{bmatrix} -9 & 4 & 1 \\ -5 & 0 & 0 \\ 2 & 4 & 8 \end{bmatrix}$; B=$\begin{bmatrix} 1 & 0 & 2 \\ 3 & 0 & -1 \\ 5 & 2 & 2 \end{bmatrix}$; C=$\begin{bmatrix} 0 & 0 & -1 \\ -2 & 3 & 2 \\ 9 & 3 & 4 \end{bmatrix}$.

Найти:

1) A+B;
2) C*A;
3) B*5;
4) транспонированную матрицу. C;
5) обратную матрицу B;
6) возведите в квадрат матрицу A;
7) сумму значений элементов столбцов матрицы A;
8) максимальное и минимальное значения матрицы B;
9) сортируйте значения элементов матрицы C по возрастанию и убыванию.

Двумерная графика

Постройте два графика в одном окне:

I вариант. $y = e^{-x^2}$; $z = arctg(x)^{1/2}$; $x \in [0, \quad 4\pi]$.

II вариант. y = $\sqrt{1+x^2}$; z = $\log_{0.5}x$; xϵ [0, 1].

III вариант. y =sin(x) + 1; z = cos(x); xϵ [- $\frac{\pi}{2}$; $\frac{\pi}{2}$].

IV вариант. y = $(x+2)^2$; z = -$(x-2)^2$; xϵ (0, 5].

V вариант. y = $\frac{2}{x}$; z = 4x - x^2; xϵ [0, ∞).

VI вариант. y = x + sin(x); z = -cos(x); xϵ (0, π).

VII вариант. y = $\sqrt{x}$; z = x; xϵ [-4,5].

VIII вариант. y = 5x^2 + 4; z = 4.5x + 1; xϵ (-10, 29).

IX вариант. y = log_3x; z = ln($\frac{x}{2}$); xϵ [-3,9].

X вариант. y = sin(x); z = cos($\frac{4\pi}{3}$) + 2; xϵ (-$\frac{4\pi}{3}$, 0).

XI вариант. y = $\frac{\sin(x)}{\cos(x)}$; z = x; xϵ[0, 1].

XII вариант. y = x+2; z = $\sqrt{(x-2)^2}$; xϵ [0, 3]

XIII вариант. y = tg(x); z = $\frac{\cos(x)}{\sin(x)}$; xϵ [0, 5π].

XIV вариант. y = 1 – x; z = x^2 + 6x + 12; xϵ (-∞; 60].

XV вариант. y = 3; z = ctg($\frac{x}{2}$); xϵ (0, ∞).

Трехмерная графика

Постройте поверхность:

I вариант. z = ln($x^2 + y^2 - xy$), $x, y \in [1, \quad 2]$.

II вариант. z = $\frac{x}{x^2+y^2}$, x, y ϵ [-1,1].

III вариант. z = x^3 - $3xy^2$, x, y ϵ [-5, 5].

IV вариант. z = $\frac{\sin(x^2+y^2)}{x^2+y^2}$, x, y ϵ [-π, π].

V вариант. z = $4x^2$ + $9y^2$ – 72y, x, y ϵ [-10,10].

VI вариант. f(x, y, z) = $-2x^2$ + $3y^2$ +$4z^2$, x, y, z ϵ [-20, 20].

VII вариант. 2z = x^2 + y^2, x, y ϵ [-40, 0] .

VIII вариант. z = $\sqrt{xy}$, x, y ϵ [-35, 35].

IX вариант. z = cos(xy), x, y ϵ [-3, 3].

X вариант. z = $\frac{x^2-y}{1+x^2+y^2}$, x, y ϵ[-π, π].

XI вариант. z = x^3 + y^3 -5xy + $^1/_5$, x, y ϵ [-5, 5].

XII вариант. z = $(x^2 + y^2)^2$ - x^2 + y^2, x, y ϵ [-1, 1].

XIII вариант. z = $cosh^2x - sinh^2x$ – 1, x, y ϵ [0, 10].

XIV вариант. z = -ln($\sqrt{(x+1)^2+\ y^2}$) – ln($\sqrt{(x-1)^2+\ y^2}$), x, yϵ[-6, 6].

XV вариант. z = x^3 - 3xy^2 + y^2, x, y ϵ [-10, 10].

ЗАКЛЮЧЕНИЕ

Пакет FreeMat может использоваться во всех сферах вычислений начиная с самых простых, заканчивая самыми сложными. С его помощью можно работать с массивами чисел, выполнять операции над многочленами, решать различные уравнения, строить графики. Пакет FreeMat включает различные интерфейсы для получения доступа к внешним подпрограммам, написанным на других языках программирования. В системе есть полноценная IDE с возможностью создания и отладки кода, сохранения и выполнения истории команд, управления файлами. FreeMat чисто численная система.

При расчетах и проектировании радиоэлектронных компонентов и устройств необходимы матричные системы компьютерной математики с полной поддержкой аппарата комплексных вычислений. Лидером мирового рынка стала коммерческая система MATLAB. Однако ее применение сдерживается ее большим объемом (более 5 Гбайт) и высокой стоимостью. FreeMat имеет возможности, обеспечивающие примерно 95% функциональности среды Matlab. Недостаток - отсутствие поддержки создания графических интерфейсов в стиле Matlab (а важна ли эта функция?). Достоинства: статистический анализ больших наборов данных, совместимость с MatLab, инструментарий для проведения численных экспериментов, поддержка MPI-параллелизации и трехмерной визуализации, улучшенный графический интерфейс.

Таким образом, в квалификационной работе:

- приведено описание изучаемых команд FreeMat;
- приведены примеры решения практических заданий с подробным пошаговым описанием действия команд FreeMat; эти задания предназначены для выполнения студентами под руководством преподавателя;

- приведены контрольные вопросы;
- приведены задания для самостоятельного выполнения студентами.

ЛИТЕРАТУРА

1. http://www.butovo.com/~zss/matlab/2/1.htm
2. http://sernam.ru/lect_matlab.php?id=14
3. http://theor.mephi.ru/wiki/index.php?title=Вводные _задания _по_MATLAB
4. Наместников С.М. Основы программирования в MatLab / Сборник лекций: УлГТУ, Ульяновск. - 2011. - 55 с.
5. http://xreferat.ru/33/3656-3-matlab.html
6. http://ru.convdocs.org/docs/index-112390.html
7. http://ppt4web.ru/informatika/graficheskie-sposoby-predstavlenija-dannykh-v-ecel.html
8. FreeMat v4.0 Documentation. Samit BasuOctober. 4, 2009
9. Семченок М.С., Семченок Н.М. СистемаМ ATLAB. Часть 1.: Учебное пособие. — СПб.: изд. СПбГУКиТ, 2004.
10. MATLAB for beginners/ FreeMat environment
11. FreeMat. Generated by Doxygen 1.8.1.1. Thu Jul 25 2013 17:18:37