Лекция 1. Система MatLab.
Краткий обзор возможностей MatLab

Система MatLab относится к среднему уровню продуктов, предназначенных для символьной математики, но рассчитана на широкое применение в сфере CAE (то есть сильна и в других областях). MatLab – одна из старейших, тщательно проработанных и проверенных временем систем автоматизации математических расчетов, построенная на расширенном представлении и применении матричных операций. Это нашло отражение и в самом названии системы – MATrix LABoratory, то есть матричная лаборатория. Однако синтаксис языка программирования системы продуман настолько тщательно , что данная ориентация почти не ощущается теми пользователями, которых не интересуют непосренодственно матричные вычисления.

Несмотря на то что изначально MatLab предназначалась исключительно для вычислений, в процессе эволюции к ней была подключена библиотека Simulink, позволяющая построить логическую схему сложной системы управления из одних только стандартных блоков, не написав при этом ни строчки кода. После конструирования такой схемы можно детально проанализировать ее работу.

В системе MatLab также существуют широкие возможности для программирования. Ее библиотека C Math (компилятор MatLab) является объектной и содержит свыше 300 процедур обработки данных на языке C. Внутри пакета можно использовать как процедуры самой MatLab, так и стандартные процедуры языка C, что делает этот инструмент мощнейшим подспорьем при разработке приложений (используя компилятор C Math, можно встраивать любые процедуры MatLab в готовые приложения).

Основы работы в MATLAB

Рабочая среда

Рабочая среда MATLAB состоит из следующих основных окон:

- Command Window (интерфейс командной строки)
- Workspace (перечень всех созданных переменных)
- Command history (история команд)
- Editor (редактор m-файлов)
- Figures (вывод графики)
- Current Folder
- Variable Editor (редактор переменных)
- Help

Простейшие вычисления

Простейшие вычисления можно выполнять в окне Command Window в диалоговом текстовом режиме, например:

```
>> 1 + 2
```

```
ans =
     3
```

Результат вычислений помещается в специальную переменную ans, а командная строка переходит в режим ввода следующей команды.

Если требуется продолжить работу с предыдущим выражением, например, (1+2) / 4, то можно воспользоваться уже имеющимся результатом, хранящимся в переменной ans:

```
>> ans /
4 ans =
     0.7500
```

Форматы вывода результатов вычислений

При вычислениях в MATLAB используется режим двойной точности. При выводе результатов, по умолчанию выдаются числа с 4 цифрами после десятичной точки в действительной форме.

Для изменнения формата вывода, необходимо в командной строке перед выводимой величиной выполнить команду format name, где name - имя формата. Для числовых данных name может принимать следующие значения:

- short - короткое представление в фиксированном формате (5 знаков);
- shorte - короткое представление в экспоненциальной форме (5 знаков мантиссы и 3 знака порядка);
- long - длинное представление в фиксированном формате (15 знаков);
- longe - длинное представление в экспоненциальной форме (15 знаков мантиссы и 3 знака порядка).

Например, значения вектора x=[4/3 1.2345e-6] в различных форматах представления будут иметь следующий вид:

- format short 1.3333 0.0000
- format short e 1.3333E+000 1.2345E-006
- format long 1.333333333333338 0.000001234500000
- format long e 1.333333333333338E+000 1.234500000000000E-006
- format bank 1.33 0.00

Задание формата сказывается только на форме вывода чисел. Вычисления же всегда происходят в режиме двойной точности.

Алфавит языка

В MATLAB, как и в других системах, используются все буквы латинского алфавита от A до Z и арабские цифры от 0 до 9. Как и в C++, большие и малые буквы это разные переменные и константы.

Специальные символы. Типы данных

Кроме букв латинского алфавита используются все специальные символы клавиатуры компьютера.

Структура типов данных MATLAB представлена схемой на рис. 1.

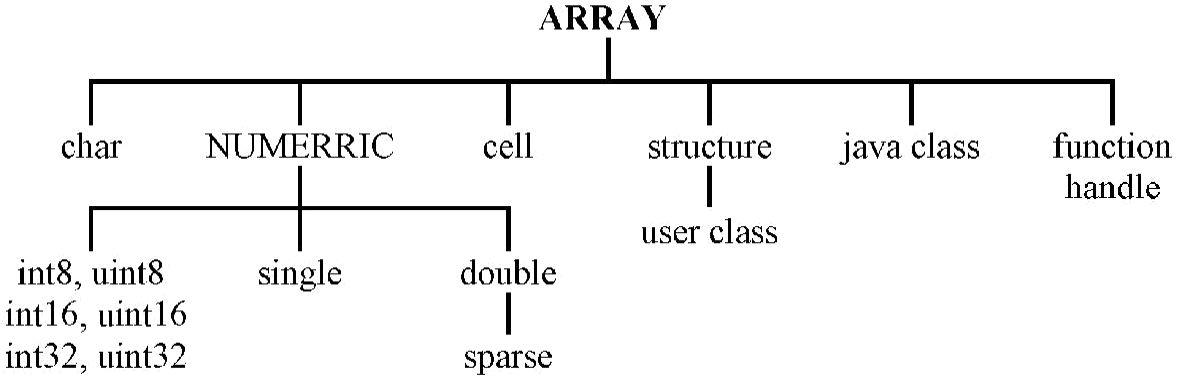

Рис. 1. Структура типов данных MATLAB

Типы данных *array* и *numerric* являются виртуальными ("кажущимися"), т. к. к ним нельзя отнести какие-либо переменные. Они служат для определения и комплектования некоторых типов данных. Таким образом, в MATLAB определены типы данных, представляющие собой многомерные массивы:

1. single - числовые массивы с числами одинарной точности;
2. double — числовые массивы с числами удвоенной точности;
3. char — строчные массивы с элементами - символами;
4. sparse - разреженные матрицы с элементами - числами двойной точности;
5. cell — массивы ячеек;
6. structure - массивы структур с полями;
7. function - handle - дескрипторы функций;
8. int8, ..., uint32 - массивы 8-, 16-, 32-разрядных целых чисел со знаками и без знаков.

Арифметические и логические операторы.

Таблица 1.

Список арифметических операторов.

Функция	Обозначение (синтаксис)
Сложение	+ (M1+M2)
Вычитание	- (M1-M2)
Матричное умножение	* (M1*M2)
Поэлементное умножение массивов	.* (M1.*M2)
Возведение матрицы в степень	^(M^X)
Поэлементное возведение массива в степень	.^ (M .^ X)
Деление матриц слева направо	/ (Ml / M2)
Поэлементное деление массивов слева направо	./ (Ml ./ M2)
Деление матриц справа налево	\ (Ml \ M2)
Поэлементное деление массивов справа налево	.\ (Ml .\ M2)

Таблица 2.

Список операторов отношения.

Функция	Обозначение (синтаксис)
Равно	== (x = =y)
Не равно	~= (x~=y)
Меньше	< (x<y)
Больше	>(x>y)
Меньше или равно	<= (x<=y)
Больше или равно	>= (x>=y)

Данные операторы выполняют поэлементное сравнение векторов или матриц одинакового размера.

Таблица 3.

Список логических операторов.

Функция	Обозначение (синтаксис)
Логическое И	& ; and (and (a, b))
Логическое ИЛИ	| ; or (or (a, b))
Логическое НЕ	~ ; not (not (a, b))
Исключающее ИЛИ	xor (xor (a, b))
Верно, если все элементы вектора равны нулю	any (any (a))
Верно, если все элементы вектора не равны нулю	all (all (a))

Логические операторы служат для реализации поэлементных логических операций над элементами одинаковых по размеру массивов.

Полный список операторов можно получить, используя команду help ops.

Приоритеты операций

В математических выражениях операторы имеют определенный приоритет исполнения:

- приоритет логических операторов выше, чем арифметических,
- приоритет возведения в степень выше приоритетов умножения и деления,
- приоритет умножения и деления выше сложения и вычитания.

Для повышения приоритета операций нужно использовать круглые скобки. Степень вложения скобок не ограничивается.

Константы и системные переменные

Константа – это предварительно определенное числовое или символьное значение, представленное уникальным именем (идентификатором).

Числа (например, 1, –2 и 1.23) являются безымянными числовыми константами.

Другие виды констант в MATLAB принято называть *системными переменными*, поскольку, с одной стороны, они задаются системой при ее загрузке, а с другой – могут переопределяться.

Основные системные переменные, применяемые:

- i или j – мнимая единица (корень квадратный из –1);
- pi – число p = 3,1415926…;
- eps – погрешность операций над числами с плавающей точкой (2–52);
- realmin – наименьшее число с плавающей точкой (2–1022);
- realmax – наибольшее число с плавающей точкой (21023)
- inf – значение машинной бесконечности;
- ans – переменная, хранящая результат последней операции;
- NaN – указание на нечисловой характер данных (Not a Number).

Примеры применения системных переменных:

```
>> 2*pi
ans = 6.2832
>> eps
ans = 2.2204e-016
>> realmin
ans = 2.2251e-308
>> realmax
ans = 1.7977e+308
>> 1/0
Warning: Divide by zero.
ans = Inf
>> 0/0
Warning: Divide by zero.
ans = NaN
```

Как отмечалось, системные переменные могут переопределяться. Можно задать системной переменной eps иное значение, например eps=0.0001. Однако важно то, что их значения по умолчанию задаются сразу после загрузки системы.

Поэтому неопределенными, в отличие от обычных переменных, системные переменные не могут быть никогда.

Символьная константа – это цепочка символов, заключенных в апострофы, например:

```
'Hello my friend!'
'2+3'
```

Если в апострофы помещено математическое выражение, то оно не вычисляется и рассматривается просто как цепочка символов. Так что '2+3' не будет возвращать число 5.

Переменные и присваивание им значений

Переменные – это имеющие имена объекты, способные хранить некоторые данные. В зависимости от этих данных переменные когут быть числовыми или символьными, векторными или матричными.

В MATLAB можно задавать переменным определенные значения. Для этого используется операция присваивания, вводимая знаком равенства:

```
Имя_переменной = Выражение
```

Типы переменных заранее не декларируются. Они определяются выражением, значение которого присваивается переменной. Так, если это выражение – вектор или матрица, то переменная будет векторной или матричной. Переменная, имеющая единственное значение, рассматривается как матрица размера 1×1.

Имя переменной (ее идентификатор) может содержать сколько угодно символов, но запоминается и идентифицируется только 31 начальный символ. Имя любой переменной не должно совпадать с именами других переменных , функций и процедур системы, то есть оно должно быть уникальным. Имя должно начинаться с буквы, может содержать буквы, цифры и символ подчеркивания _. Недопустимо включать в имена переменных пробелы и специальные знаки, например +, -, *, / и т. д., поскольку в этом случае правильная интерпретация выражений становится невозможной.

Могут использоваться и символьные переменные, причем символьные значения заключаются в апострофы, например s='Demo'.

Для вывода списка объявленных в Workspace переменных используется команда who. Команда whos используется для вывода подробной информации о переменных Workspace.

Сохранение в файл и чтение из файла переменных в Workspace можно выполнив команды save filename и load filename. Расширение файла можно не указывать, в этом случае файл получит расширение mat. Переменные в mat-файл записываются в двоичном формате.

Уничтожение определений переменных

В памяти компьютера переменные занимают определенное место, называемое рабочей областью (workspace). Для очистки рабочей области используется функция clear в разных формах, например:

• clear – уничтожение определений всех переменных;

• clear x – уничтожение определения переменной x;

• clear a, b, c – уничтожение определений нескольких переменных.

Уничтоженная (стертая в рабочей области) переменная становится неопределенной. Использовать неопределенные переменные нельзя, и такие попытки будут сопровождаться выдачей сообщений об ошибке. Приведем примеры задания и уничтожения переменных:

```
>> x=2*pi
x = 6.2832
>> V=[12345]
V=12345
>> clear V
>> V
??? Undefined function or variable 'V'.
>> clear
>> x
??? Undefined function or variable 'x'.
>> M
??? Undefined function or variable 'M'.
```

Элементарные функции

Набор элементарных функций представим их описанием. В тригонометрических функциях углы измеряются в радианах.

Таблица 4.

Список элементарных функций.

Функция	Обозначение (синтаксис)
$\|x\|$ - модуль	abs(x)
e^x - экспонента	exp(x)
In x - натуральный логарифм	log(x)
$\log_2 x$ - логарифм по основанию 2	log2(x)
lg x - десятичный логарифм	log10(x)
2^x - 2 в степени x	pow(x)
$x^{1/2}$ - квадратный корень	sqrt(x)
sin x - синус	sin(x)

cos x - косинус	cos(x)
tg x - тангенс	tan(x)
ctg x - котангенс	cot(x)
округление	round(x)

Следует помнить, что все элементарные функции должны записываться в программах малыми буквами.

С полным списком элементарных функций можно ознакомиться, выполнив команду help elfun, а со списком специальных функций – с помощью команды help specfun.

Лекция № 2. Работа с массивами в MATLAB

Основные соглашения

Массив – упорядоченная, пронумерованная совокупность однородных данных. Массивы различаются по числу размерностей (измерений): одномерные, двумерные, многомерные. Размером массива называют число элементов, вдоль каждого из изменений.

Доступ к элементам осуществляется при помощи индекса. В MATLAB нумерация элементов массивов начинается с единицы.

Вектор может быть записан в столбик (вектор-столбец) или в строку (вектор-строка).

Создание векторов и матриц

Если надо задать вектор из трех элементов, то их значения надо перечислить в квадратных скобках, разделяя пробелами.

```
>> V=[123]
V =
     1    2    3
```

В данном случае задан вектор-строка. Если разделить элементы точкой с запятой, то получим вектор-столбец.

```
>> V=[1;2;3]
V =
     1
     2
     3
```

Задание матрицы требует указания нескольких строк. Для разграничения строк используется символ ; (точка с запятой).

```
>> T=[123;456;789]
T =
      1    2    3
      4    5    6

      7    8    9
```

Узнать размерность и размер массива можно при помощи функции size.

Для указания отдельного элемента вектора или матрицы используются выражения вида V(i) или T(i, j). Например:

```
>> T(3,2)
ans =
     8
```

Если элементу T(i, j) нужно присвоить новое значение x, то используют оператор присваивания:

```
>>T(3,2)=x;
```

Выражение T(i) с одним индексом дает доступ к элементам матрицы, развернутым в один столбец. Такая матрица образуется из исходной, если подряд выписать ее столбцы. Например:

```
>> T(3)
 ans =
     7
>>T(8)
ans =

     6
```

Символ : (двоеточие) – формирование подвекторов и подматриц из векторов и матриц.

• A(:,j) – это j-й столбец из A;
• A(i,:) – это i-я строка из A;
• A(:,:) – эквивалент двумерного массива (для матриц это аналогично A);
• A(j:k) – это A(j), A(j+1),…,A(k);
• A(:,j:k) – это A(:,j),A(:,j+1),…,A(:,k);
• A(:,:,k) – это k-ая страница трехмерного массива A;
• A(:) – записывает все элементы массива À в виде столбца.

Имеются также ряд особых функций для задания векторов и матриц.

Для создания единичной матрицы, которая имеет единичные диагональные элементы и нулевые все остальные, служит функция eye:

• eye(n) – возвращает единичную матрицу размера nxn;
• eye(m,n) или eye([m n]) – возвращает матрицу размера mxn с единицами по диагонали и нулями в остальных ячейках;
• eye(size(A)) – возвращает единичную матрицу того же размера, что и A. Пример использования функции eye:

```
>> S=eye(4,5)
S =
     10000
     01000
     00100
     00010
```

Единичная матрица не определена для многомерных массивов. Так, функция y = eye([2,3,4]) при попытке ее вычисления приведет к выводу сообщения об ошибке.

Для создания матриц, все элементы которых – единицы, используется функция ones:

• ones(n) – возвращает матрицу размера nxn, все элементы которой – единицы. Если n – не скаляр, то появится сообщение об ошибке;

• ones(m,n) или ones([m n]) – возвращают матрицу размера mxn, состоящую из единиц;

• ones(d1,d2,d3,…) или ones([d1 d2 d3…]) – возвращают массив из единиц с размерностью d1xd2xd3x… ;

• ones(size(A)) – возвращает массив единиц той же размерности и размера, что и A. Матрица с единичными элементами, в отличие от единичной матрицы, в MATLAB определена и для многомерных массивов.

```
>> S=ones(3,4)
s =
     1111
     1111
     1111
```

Для создания матриц, все элементы которых – нули, используется функция zeros:

• zeros(n) – возвращает матрицу размера nxn, содержащую нули. Если n – не скаляр, то появится сообщение об ошибке;

• zeros(m,n) или zeros([m n]) – возвращают матрицу размера mxn, состоящую из нулей;

• zeros(d1,d2,d3,…) или zeros([d1 d2 d3…]) – возвращают массив из нулей размера d1xd2xd3x… ;

• zeros(size(A)) – возвращает массив нулей того же размера и размерности, что и A.

```
>> S=zeros(3,2)
S =
     0 0
     0 0
     0 0
```

Аналогично используются функции true(m,n) и false(m,n) для создания массивов размера mxn, содержащих соответственно логические 1 и 0.

Для создания линейного массива равноотстоящих точек используется функция linspace:

• linspace(a,b) – возвращает линейный массив из 100 точек, равномерно распределенных между a и b;

• linspace(a,b,n) – генерирует n точек, равномерно распределенных в нтервале от a до b.

```
>> M=linspace(4,20,14)
M =
     Columns 1 through  7
     4.0000  5.2308 6.4615 7.6923 8.9231 10.1538  11.3846
     Columns 8 through  14    16.3077 17.5385 18.7692 20.0000
```

```
     12.6154  13.8462 15.0769
```

Еще один вариант создания линейного массива равноотстоящих точек: a:b – создаст массив целых чисел начиная с a и заканчивая b. Для изменения шага значений массива можно использовать следующую апись: a: :b, где – шаг значений массива. Например:

```
>> x = 0 : 10;
>> y = -3 : 0.2 : 3
```

Для создания массива со случайными элементами используются следующие функции.

randperm(n) – возвращает *случайные перестановки* целых чисел 1:n в векторе_строке.

```
>> randperm(6)
ans =
     243651
```

rand генерирует массивы случайных чисел, значения элементов которых *равномерно* распределены в промежутке (0,1):

• rand(n) – возвращает матрицу размера nxn. Если n – не скаляр, то появится сообщение об ошибке;

• rand(m,n) или rand([m n]) – возвращают матрицу размера mxn;

• rand(m,n,p,…) или rand([m n p…]) – возвращает многомерный массив;

• rand(size(A)) – возвращает массив того же размера и размерности, что и A, с элементами, распределенными по равномерному закону;

• rand (без аргументов) – возвращает одно случайное число, которое изменяется при каждом последующем вызове и имеет равномерный закон распределения;

```
>> Y=rand(4,3)
Y =
     0.9501 0.8913 0.8214
     0.2311 0.7621 0.4447
     0.6068 0.4565 0.6154
     0.4860 0.0185 0.7919
```

randn генерирует массив со случайными элементами, распределенными по нормальному закону с нулевым математическим ожиданием и среднеквадратическим отклонением, равным 1:

• randn(n) – возвращает матрицу размера n n. Если n – не скаляр, то появится сообщение об ошибке;

• randn(m,n) или randn([m n]) – возвращают матрицу размера mxn;

• randn(m,n,p,…) или randn([m n p…]) – возвращает массив с элементами, значения которых распределены по нормальному закону;

• randn(size(A)) – возвращает массив того же размера, что и A, с элементами, распределенными по нормальному закону;

• randn (без аргументов) – возвращает одно случайное число, которое изменяется при каждом последующем вызове и имеет нормальное распределение;

```
>> Y=randn(4,3)
Y =
     -0.4326 -1.1465 0.3273
     -1.6656 1.1909 0.1746
     0.1253 1.1892 -0.1867
     0.2877 -0.0376 0.7258
```

Операции над векторами

Конкатенация матриц

Конкатенацией называют объединение массивов или матриц. Это реализует следующая функция:

• C = cat(dim,A,B) – объединяет массивы A и B в соответствии со спецификацией размерности dim и возвращает объединенный массив; dim=1 – горизонтальная конкатенация, dim=2 – вертикальная, dim=3 – многомерный массив размерности 3 и т. д.;

• C = cat(dim,A1,A2,A3,A4,…) объединяет все входные массивы (A1, A2, A3, A4 и т. д.) в соответствии со спецификацией размерности dim и возвращает объединенный массив;

```
>> A =[2,4;3,5];B=[8,7;9,0];C=cat(1,A,B)
     C =
     2 4
     3 5
     8 7
     9 0
```

Удаление строк и столбцов

Для удаления отдельных столбцов и строк матриц используются пустые квадратные скобки – [ ]. Проделаем это с матрицей M.

```
T(:,2)=[ ]
T =
     1    3
     4    6
     7    9
```

Транспонирование матрицы

Транспонирование вектора или матрицы производится при помощи ‘

```
T’
ans =
      1    4    7
      3    6    9
```

Определитель матрицы

Для нахождения *определителя* (*детерминанта*) матриц в MATLAB имеется функция:

• det(X) – возвращает определитель квадратной матрицы X. Если X содержит только целые элементы, то результат – тоже целое число.

```
>> A=[2,3,6;1,8,4;3,6,7]
A =
     2 3 6
     1 8 4
     3 6 7
>> det(A)

ans = -29
```

Обращение матриц

Обратной называют матрицу, получаемую в результате деления единичной матрицы E на исходную матрицу X. Таким образом, X–1=E/X. Следующая функция обеспечивает реализацию данной операции:

• inv(X) возвращает матрицу, обратную квадратной матрице X. Предупреждающее сообщение выдается, если X плохо масштабирована или близка к вырожденной.

```
>> inv(rand(4,4))
ans =
     2.2631  -2.3495 -0.4696  -0.6631
     -0.7620  1.2122 1.7041  -1.2146
     -2.0408  1.4228 1.5538  1.3730

     1.3075  -0.0183 -2.5483  0.6344
```

Статистические функции

Функция sum предназначена для суммирования элементов вектора.

Функция mean предназначена для вычисления среднего арифметического элементов вектора.

Для нахождения минимального и максимального элемента вектора используются функции min и max соответственно.

Решение систем линейных уравнений

Пусть задана система n линейных алгебраических уравнений с n неизвестными. Система уравнений в матричной форме представляется следующим образом:

AX = B (3.1.) где A - квадратная матрица коэффициентов, размером n x n строк и столбцов;

X - вектор-столбец неизвестных;

B - вектор-столбец правых частей.

Решение в матричном виде представляется как X=B/A

Пример: Решить систему 4-х линейных уравнений:

1.116x1 +0.1397x2 +0.1254x3 +0.1490x4 =1.5471;
0.1582x1 +0.1768x2 +1.1675x3 +0.1871x4 =1.6471;
0.1968x1 +1.2168x2 +0.2071x3 +0.2271x4 =1.7471;
0.2368x1 +0.2568x2 +0.2471x3 +1.2671x4 =1.8471.

```
a = [1.1161 0.1397 0.1254 0.1490;
0.1582 0.1768 1.1675 0.1871 ;
0.1968 1.2168 0.2071 0.2271 ;
0.2368 0.2568 0.2471 1.2671];
b = [1.5471 ; 1.6471 ; 1.7471 ; 1.8471];
X = a\b X =
     1.0406
     0.9351
     0.9870
     0.8813
```

Лекция № 3. Графика MATLAB

Графики функций в декартовой системе координат

Команда plot служит для построения графиков функций в декартовой системе координат. Эта команда имеет ряд параметров, рассматриваемых ниже.

• plot(X,Y) строит график функции $y(x)$, координаты точек (x,y) которой берутся из векторов одинакового размера Y и X. Если X или Y – матрица, то строится семейство графиков по данным, содержащимся в колонках матрицы.

```
>> x=[0 1 2 3 4 5]; Y=[sin(x);cos(x)]; plot(x,Y)
```

• plot(Y) строит график $y(i)$, где значения y берутся из вектора Y, а i представляет собой индекс соответствующего элемента. Если Y содержит комплексные элементы, то выполняется команда plot(real(Y), imag(Y)).

Во всех других случаях мнимая часть данных игнорируется.

```
>> x=-2*pi:0.02*pi:2*pi; y=sin(x)+i*cos(3*x); plot(y)
```

• plot(X,Y,S) аналогична команде plot(X,Y), но тип линии графика можно задавать с помощью строковой константы S.

Значениями константы S приведены в Таблице 5.

Таблица 5.

Цвет линии		Тип точки		Тип линии	
y	желтый	.	точка	-	сплошная
m	фиолетовый	o	окружность	:	двойной пунктир
c	голубой	x	крест	-.	штрихпунктир
r	красный	+	плюс	--	штриховая
g	зеленый	*	звездочка		

b	синий	s	квадрат
w	белый	d	ромб
k	черный	v	треугольник (вниз)
		^	треугольник (вверх)
		<	треугольник (влево)
		>	треугольник (вправо)
		p	пятиугольник
		h	шестиугольник

Таким образом, с помощью строковой константы S можно изменять цвет линии, представлять узловые точки различными отметками и менять тип линии графика.

plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,…) – эта команда строит на одном графике ряд линий, представленных данными вида (X*i*,Y*i*,S*i*), где X*i* и Y*i* – векторы или матрицы, а S*i* – строки. С помощью такой конструкции возможно построение, например, графика функции линией, цвет которой отличается от цвета узловых точек.

При отсутствии указания на цвет линий и точек он выбирается автоматически из таблицы цветов (белый исключается). Если линий больше шести , то выбор цветов повторяется. Для монохромных систем линии выделяются стилем.

```
% Программа построения графиков трех функций
x=-2*pi:0.1*pi:2*pi;
y1=sin(x); y2=sin(x).^2; y3=sin(x).^3;
plot(x,y1,'-m',x,y2,'-.+r',x,y3,'--ok')
```

Диаграммы и гистограммы

Команды для построения столбцовых диаграмм:

• bar(x,Y) строит столбцовый график элементов вектора Y (или группы столбцов для матрицы Y) со спецификацией положения столбцов, заданной значениями элементов вектора x, которые должны идти в монотонно возрастающем порядке;

• bar(Y) строит график значений элементов матрицы Y так же, как указано выше, но фактически для построения графика используется вектор x=1:m;

• bar(x,Y,WIDTH) или BAR(Y,WIDTH) – команда аналогична ранее рассмотренным, но со спецификацией ширины столбцов (при WIDTH > 1 столбцы перекрываются). По умолчанию задано WIDTH = 0.8.

Возможно применение этих команд и в следующем виде: bar(…,'Спецификация') для задания спецификации графиков, например типа линий, цвета и т. д., по аналогии с командой plot. Спецификация ‘stacked’ задает рисование всех *n* столбцов друг на друге.

Пример построения столбцовой диаграммы матрицы размером 12x3 приводится ниже:

```
>> bar(rand(12,3),'stacked')
```

Помимо команды bar(…), существует аналогичная ей по синтаксису команда barh(…), которая строит столбцовые диаграммы с горизонтальным расположением столбцов.

Классическая *гистограмма* характеризует в виде столбцовой диаграммы числа попаданий значений элементов вектора Y в M интервалов.

Для получения данных для гистограммы служит функция hist:

• N=hist(Y) возвращает вектор чисел попаданий для 10 интервалов, выбираемых автоматически. Если Y – матрица, то выдается массив данных о числе попаданий для каждого из ее столбцов;

• N=hist(Y,M) аналогична вышерассмотренной, но используется M интервалов (M – скаляр);

• N=hist(Y,X) возвращает числа попаданий элементов вектора Y в интервалы, центры которых заданы элементами вектора X;

• [N,X]=HIST(…)возвращает числа попаданий в интервалы и данные о центрах интервалов.

```
>> x=-3:0.2:3; y=randn(1000,1);
>> hist(y,x); h=hist(y,x)
h =
Columns 1  through  12
0  0  3  7  8  9  11  23  33  43  57  55
Columns 13 through 24
70  62  83  87  93  68  70  65  41  35  27  21
Columns 25 through 31
12  5  6  3  2  1  0
```

Команды для созданиякруговых диаграмм:

• pie(X) строит круговую диаграмму по данным нормализованного вектора X/SUM(X). SUM(X) – сумма элементов вектора. Если SUM(X)<=1.0, то значения в X непосредственно определяют площадь секторов;

• pie(X,EXPLODE) строит круговую диаграмму, у которой отрыв секторов от центра задается вектором EXPLODE, который должен иметь тот же размер, что и вектор данных X.

Следующий пример строит цветную круговую диаграмму с пятью секторами, причем последний сектор отделен от остальных:

```
>> X=[1 2 3 4 5]; pie(X,[0 0 0 0 2])
```

Создание массивов данных для трехмерной графики

Поверхности как объекты трехмерной графики обычно описываются функцией двух переменных z(x,y). Специфика построения трехмерных графиков

требует не просто задания ряда значений x и y, но и двумерных массивов – матриц. Для создания таких массивов служит функция meshgrid:

• [X,Y] = meshgrid(x,y) преобразует область, заданную векторами x и y, в массивы X и Y, которые могут быть использованы для вычисления функции двух переменных и построения трехмерных графиков. Строки выходного массива X являются копиями вектора x; а столбцы Y – копиями вектора y;

• [X,Y] = meshgrid(x) аналогична [X,Y] = meshgrid(x,x);

```
>> [X,Y] = meshgrid(1:4,13:17)
X =
1234
1234
1234
1234
1234 Y
=
13 13 13 13
14 14 14 14
15 15 15 15
16 16 16 16
```

Трёхмерная графика

Команда plot3(…) является аналогом команды plot(…), но относится к функции двух переменных z(x,y). Она строит аксонометрическое изображение трехмерных поверхностей и представлена следующими формами:

• plot3(x,y,z) строит массив точек, представленных векторами x, y и z, соединяя их отрезками прямых. Эта команда имеет ограниченное применение;

• plot3(X,Y,Z), где X, Y и Z – три матрицы одинакового размера, строит точки с координатами X(i,:), Y(i,:) и Z(i,:) и соединяет их отрезками прямых.

Ниже дан пример программы построения трехмерной поверхности, описываемой функцией z(x,y)=x^2+y^2:

```
% Программа построения поверхности линиями
[X,Y]=meshgrid([-3:0.15:3]);
Z=X.^2+Y.^2; plot3(X,Y,Z)
```

• plot3(X,Y,Z,S) обеспечивает построения, аналогичные рассмотренным ранее, но со спецификацией стиля линий и точек, соответствующей спецификации команды plot:

```
% Программа построения поверхности кружками
[X,Y]=meshgrid([-3:0.15:3]);
Z=X.^2+Y.^2;
plot3(X,Y,Z,'o')
```

• plot3(x1,y1,z1,s1,x2,y2,z2,s2,x3,y3,z3,s3,…) строит на одном рисунке графики нескольких функций z1(x1,y1), z2(x2,y2) и т. д. со спецификацией линий и маркеров каждой из них.

```
% Программа построения сетчатого графика функции
```

```
[X,Y]=meshgrid([-3:0.15:3]);
Z=X.^2+Y.^2; plot3(X,Y,Z,'-
k',Y,X,Z,'-k')
```

В данном случае строятся два графика одной и той же функции с взаимно перпендикулярными образующими линиями. Поэтому график имеет вид сетки без окраски ее ячеек (напоминает проволочный каркас фигуры).

Наиболее представительными и наглядными являются *сетчатые графики* поверхностей с заданной или функциональной окраской.

• mesh(X,Y,Z,C) выводит в графическое окно сетчатую поверхность Z(X,Y) с цветами узлов поверхности, заданными массивом C.

• mesh(X,Y,Z) – аналог предшествующей команды при C=Z. В данном случае используется функциональная окраска, при которой цвет задается высотой поверхности.

Возможны также формы команды mesh(x,y,Z), mesh(x,y,Z,C), mesh(Z) и mesh(Z,C). Функция mesh возвращает дескриптор для объекта класса surface. Ниже приводится пример программы с применением команды mesh:

```
% Программа построения графика поверхности с окраской
[X,Y]=meshgrid([-3:0.15:3]);
Z=X.^2+Y.^2;
mesh(X,Y,Z)
```

MATLAB имеет несколько графических функций, возвращающих *матричный образ поверхностей*. Например , функция peaks(N) возвращает матричный образ поверхности с рядом пиков и впадин. Такие функции удобно использовать для проверки работы графических команд трехмерной графики.

```
>> z=peaks(25); mesh(z);
```

Особенно наглядное представление о поверхностях дают сетчатые графики, использующие *функциональную закраску* ячеек. Например, цвет окраски поверхности $z(x,y)$ может быть поставлен в соответствии с высотой z поверхности с выбором для малых высот темных тонов, а для больших – светлых.

Для построения таких поверхностей используются команды класса surf(…):

• surf(X,Y,Z,C) строит цветную параметрическую поверхность по данным матриц X, Y и Z с цветом, задаваемым массивом C;

• surf(X,Y,Z) аналогична предшествующей команде, где C=Z, так что цвет задается высотой той или иной ячейки поверхности;

• surf(x,y,Z) и surf(x,y,Z,C) с двумя векторными аргументами x и y – векторы x и y заменяют два первых матричных аргумента и должны иметь

длины length(x)=n и length(y)=m, где [m,n] = size(Z). В этом случае вершины областей поверхности представлены тройками координат (x(j), y(i), Z(i,j)). Заметим, что x соответствует столбцам Z, а y соответствует строкам;

• surf(Z) и surf(Z,C) используют x = 1:n и y = 1:m. В этом случае высота Z – однозначно определенная функция, заданная геометрически прямоугольной сеткой;

• h=surf(…) строит поверхность и возвращает дескриптор объекта класса

surface.

```
% Программа построения графика параболоида с окраской
[X,Y]=meshgrid([-3:0.15:3]);
Z=X.^2+Y.^2;
surf(X,Y,Z)
```

Графики поверхностей, в которых имитируется освещение от точечного источника света, расположенного в заданном месте координатной системыстроятся при помощи команды surfl:

• surfl(…) аналогична команде surf(…), но строит график поверхности с подсветкой от источника света;

• surfl(Z,S) или surfl(X,Y,Z,S) строит графики поверхности с подсветкой от источника света, положение которого в системе декартовых координат задается вектором S=[Sx,Sy,Sz];

• surfl(…,'light') позволяет при построении задать цвет подсветки с помощью объекта Light;

• surfl(…,'cdata') при построении имитирует эффект отражения;

• surfl(X,Y,Z,S,K) задает построение поверхности с параметрами, заданными вектором K=[ka,kd,ks,spread], где ka – коэффициент фоновой подсветки, kd – коэффициент диффузного отражения, ks – коэффициент зеркального отражения и spread – коэффициент глянцевитости;

• H=surfl(…)строит поверхность и возвращает дескрипторы поверхности и источников света.

```
% Программа построения поверхности с имитацией
% ее освещенности от точечного источника
[X,Y]=meshgrid([-3:0.1:3]);
Z=sin(X)./(X.^2+Y.^2+0.3);
surfl(X,Y,Z);
```

Средства управления подсветкой и обзором фигур:

• diffuse – задание эффекта диффузионного рассеяния;

• lighting – управление подсветкой;

• material – имитация свойств рассеивания света различными материалами;

• specular – задание эффекта зеркального отражения.

Следующие три команды позволяют управлять углами просмотра, под которыми рассматривается видимая в графическом окне фигура:

• view – задание положения точки просмотра;

• viewmtx – задание и вычисление матрицы вращения;

• rotate3d – задание поворота трехмерной фигуры.

Цветные объемные круговые диаграммы

Иногда используются объемные круговые диаграммы. Для их построения служит команда pie3:

• pie3(…) аналогична команде pie(…), но дает построение объемных секторов.

В приведенном ниже примере строится объемная диаграмма с отделением двух секторов, показанная на рис. 6.52.

```
>> X=[1 2 3 4 5]; pie3(X,[0 0 1 0 1])
```

Лекция № 4. Редактирование графиков MATLAB

Текстовое оформление графиков

Установка титульной надписи. Для установки над графиком титульной надписи используется следующая команда:

• title('string') установка на двумерных и трехмерных графиках титульной надписи, заданной строковой константой 'string'.

Установка осевых надписей. Для установки надписей возле осей x, y и z используются следующие команды:

• xlabel('String')
• ylabel('String')
• zlabel('String')

Соответствующая надпись задается символьной константой или переменной 'String'.

```
% Программа построения графика поверхности
% с текстовым оформлением
[X,Y]=meshgrid([-3:0.1:3]);
Z=sin(X)./(X.^2+Y.^2+0.3); surfl(X,Y,Z);
colorbar colormap(gray); shading interp
xlabel('Axis X'); ylabel('Axis Y')
zlabel('Axis Z'); title('Surface
graphic')
```

Ввод текста в любое место графика. Часто возникает необходимость добавления текста в определенное место графика, например для обозначения той или иной кривой графика. Для этого используется команда text:

• text(X,Y,'string') добавляет в двумерный график текст, заданный строковой константой 'string', так что начало текста расположено в точке с координатами (X,Y). Если X и Y заданы как одномерные массивы, то надпись помещается во все позиции [x(i),y(i)];

• text(X,Y,Z,'string') добавляет в трехмерный график текст, заданный строковой константой 'string', так что начало текста расположено в позиции, заданной координатами X, Y и Z.

```
>> x=-10:0.1:10; plot(x,sin(x).^3)
>> text(-4,0.7,'Graphic sin(x)^3')
```

Позиционирование текста с помощью мыши. Очень удобный способ ввода текста предоставляет команда gtext:

• gtext('string') задает выводимый на график текст в виде строковой константы 'string' и выводит на график перемещаемый мышью маркер в виде крестика. Установив маркер в нужное место, достаточно щелкнуть любой кнопкой мыши для вывода текста;

• gtext(C) позволяет аналогичным образом разместить многострочную надпись из массива строковых переменных C.

```
>> x=-15:0.1:15; plot(x, sin(x).^3)
>> gtext('Function sin(x)^3')
```

Установив перекрестие в нужное место графика, достаточно нажать любую клавишу или любую кнопку мыши, и на этом месте появится надпись.

Высокая точность позиционирования надписи и быстрота процесса делают данный способ нанесения надписей на графики одним из наиболее удобных.

Форматирование графиков

Вывод пояснений и легенды. Пояснение в виде отрезков линий со справочными надписями, размещаемое внутри графика или около него, называется легендой. Для создания легенды используются различные варианты команды legend:

• legend(string1,string2,string3,…) добавляет к текущему графику легенду в виде строк, указанных в списке параметров;

• legend(H,string1,string2,string3,…) помещает легенду на график, содержащий объекты с дескрипторами H, используя заданные строки как метки для соответствующих дескрипторов;

• legend(AX,…) помещает легенду в осях (объект класса axes) с дескриптором AX;

• legend(M) размещает легенду, используя данные из строковой матрицы M;

• legend OFF устраняет ранее выведенную легенду;

• legend перерисовывает текущую легенду, если таковая имеется;

• legend(legendhandle) перерисовывает легенду, указанную дескриптором legendhandle;

• legend(…,Pos) помещает легенду в точно определенное место, специфицированное параметром Pos:

– Pos=0 – лучшее место, выбираемое автоматически;

– Pos=1 – верхний правый угол;

– Pos=2 – верхний левый угол;

– Pos=3 – нижний левый угол;

– Pos=4 – нижний правый угол;

– Pos=-1 – справа от графика.

Чтобы перенести легенду, установите на нее курсор, нажмите левую кнопку мыши и перетащите легенду в необходимую позицию.

Команда legend может использоваться с двумерной и трехмерной графикой и со специальной графикой – столбцовыми и круговыми диаграммами и т. д. Двойным щелчком можно вывести легенду на редактирование.

```
% Программа построения графика трех функций
% с выводом их обозначений - легендой
x=-2*pi:0.1*pi:2*pi;
y1=sin(x); y2=sin(x).^2; y3=sin(x).^3;
plot(x,y1,'-m',x,y2,'-.+r',x,y3,'—ok')
legend('Function 1','Function 2','Function 3');
```

Незначительная модификация команды legend (применение дополнительного параметра -1) позволяет построить график трех функций с легендой вне поля графика. Это иллюстрирует следующая программа:

```
% Программа построения графика трех функций
% с выводом легенды вне поля графика
x=-2*pi:0.1*pi:2*pi;
y1=sin(x); y2=sin(x).^2; y3=sin(x).^3; plot(x,y1,'-
m',x,y2,'-.+r',x,y3,'—ok') legend('Function
1','Function 2','Function 3',-1);
```

В данном случае недостатком можно считать сокращение полезной площади самого графика.

Управление свойствами осей графиков

Обычно графики выводятся в режиме автоматического масштабирования. Следующие команды класса axis меняют эту ситуацию:

• axis([XMIN XMAX YMIN YMAX]) – установка диапазонов координат по осям x и y для текущего двумерного графика;

• axis([XMIN XMAX YMIN YMAX ZMIN ZMAX]) – установка диапазонов координат по осям x, y и z текущего трехмерного графика;

• axis auto – установка параметров осей по умолчанию;

• axis manual «замораживает» масштабирование в текущем состоянии, чтобы при использовании команды hold on следующие графики использовали те же параметры осей;

• axis tight устанавливает диапазоны координат по осям в соответствии с диапазонами изменения данных;

• axis ij задает «матричную» прямоугольную систему координат с началом координат в левом верхнем углу, ось i – вертикальная, размечаемая сверху вниз, ось j – горизонтальная и размечается слева направо;

• axis xy устанавливает декартову систему координат с горизонтальной осью x, размечаемой слева направо, и вертикальной осью y, размечаемой снизу вверх (начало координат размещается в нижнем левом углу);

• axis equal включает масштаб с одинаковым расстоянием между метками по осям x, y и z;

• axis image устанавливает масштаб, при котором пиксели изображения становятся квадратами;

• axis square устанавливает текущие оси в виде квадрата (или куба в трехмерном случае) с одинаковым расстоянием между метками и одинаковой длиной осей;

• axis normal восстанавливает масштаб, отменяя установки axis equal и axis square;

• axis vis3d «замораживает» пропорции осей для возможности поворота трехмерных объектов;

• axis off убирает с осей их обозначения и маркеры;

• axis on восстанавливает ранее введенные обозначения осей и маркеры;

• V=axis возвращает вектор-строку, содержащую коэффициенты масштабирования для текущего графика. Если текущий график двумерный, то вектор имеет 4 компонента, если трехмерный – 6 компонентов.

Следующий пример иллюстрирует применение команды axis при построении двумерного графика функции одной переменной:

```
>> x=-5:0.1:5; plot(x,sin(x)); axis([-10 10 -1.5 1.5])
```

Теперь масштабы осей заданы командой axis, а не диапазоном изменения значений x и y.

Включение и выключение сетки. Команды grid позволяют задавать построение сетки или отменять это построение:

• grid on добавляет сетку к текущему графику;

• grid off отключает сетку;

• grid последовательно производит включение и отключение сетки.

Команды grid устанавливают свойства объектов XGrid, Ygrid и Zgrid для текущих осей.

```
>> x=-5:0.1:5; plot(x,sin(x));
>> axis([-10 10 -1.5 1.5]); grid on
```

Наложение графиков друг на друга. Во многих случаях желательно построение многих наложенных друг на друга графиков в одном и том же окне. Для этого служит команда продолжения графических построений hold. Она используется в следующих формах:

• hold on обеспечивает продолжение вывода графиков в текущее окно, что позволяет добавлять последующие графики к уже существующим;

• hold off отменяет режим продолжения графических построений;

• hold работает как переключатель, последовательно включая режим продолжения графических построений и отменяя его.

Команда hold on устанавливает значение add для свойства NextPlot объектов figure и axes, а hold off устанавливает для этого свойства значение replace. Рекомендуется также ознакомиться с командами ishold, newplot, figure и axes.

Приведенный ниже пример показывает, как с помощью команды hold on на график синусоиды накладываются еще три графика параметрически заданных функций:

```
>> x=-5:0.1:5; plot(x,sin(x)); hold on
>> plot(sin(x),cos(x)); plot(2*sin(x),cos(x))
>> plot(4*sin(x),cos(x)); hold off
```

Разбиение графического окна . Для расположения в одном окне нескольких координатных осей с различными графиками без наложения их друг на друга используются команды subplot, применяемые перед построением графиков:

• subplot создает новые объекты класса axes (подокна);

• subplot(m,n,p) или subplot(mnp) разбивает графическое окно на mxn подокон, при этом m – число подокон по горизонтали, n – число подокон по вертикали, а p – номер подокна, в которое будет выводиться текущий график (подокна отсчитываются последовательно по строкам);

• subplot(H), где H – дескриптор для объекта axes, дает альтернативный способ задания подокна для текущего графика;

• subplot('position',[left bottom width height]) создает подокно с заданными нормализованными координатами (в пределах от 0.0 до 1.0);

• subplot(111) и clf reset удаляют все подокна и возвращают графическое окно в обычное состояние.

Следующая программа иллюстрирует применение команды subplot:

```
% Программа построения четырех графиков в разных
подокнах x=-5:0.1:5;
subplot(2,2,1),plot(x,sin(x))
subplot(2,2,2),plot(sin(5*x),cos(2*x+0.2))
subplot(2,2,3),contour(peaks)
subplot(2,2,4),surf(peaks)
```

Для всех графиков возможна индивидуальная установка дополнительных объектов, например титульных надписей, надписей по осям и т. д.

Изменение масштаба графика. Для изменения масштаба двумерных графиков используются команды класса zoom:

• zoom переключает состояние режима интерактивного изменения масштаба для текущего графика;

• zoom(FACTOR) устанавливает масштаб в соответствии с коэффициентом FACTOR;

• zoom on включает режим интерактивного изменения масштаба для текущего графика;

• zoom off выключает режим интерактивного изменения масштаба для текущего графика;

• zoom out обеспечивает полный просмотр, то есть устанавливает стандартный масштаб графика;

• zoom xon или zoom yon включает режим изменения масштаба только по оси x или по оси y;

• zoom reset запоминает текущий масштаб в качестве масштаба по умолчанию для данного графика;

• zoom(FIG,OPTION) применяется к графику, заданному дескриптором FIG, при этом OPTION может быть любым из перечисленных выше аргументов.

Команда zoom позволяет управлять масштабированием графика с помощью мыши. Для этого надо подвести курсор мыши к интересующей вас области рисунка. Если команда zoom включена (on), то нажатие левой кнопки увеличивает масштаб вдвое, а правой – уменьшает вдвое. При нажатой левой кнопке мыши можно выделить пунктирным черным прямоугольником нужный участок графика – при отпускании кнопки он появится в увеличенном виде и в том масштабе, который соответствует выделяющему прямоугольнику.

```
>> x=-5:0.01:5; plot(x,sin(x.^5)./(x.^5+eps)); zoom on
```

Команда zoom, таким образом, выполняет функцию «лупы», позволяющей наблюдать в увеличенном виде отдельные фрагменты сложных графиков.

Цветовая окраска графиков

Установка палитры цветов. Палитра цветов RGB задается матрицей MAP из трех столбцов, определяющих значения интенсивности красного (red), зеленого (green) и синего (blue) цветов. Их интенсивность задается в относительных единицах от 0.0 до 1.0. Например, [0 0 0] задает черный цвет, [1 1 1] – белый цвет, [0 0 1] – синий цвет . При изменении интенсивности цветов в указанных пределах возможно задание любого цвета. Таким образом, цвет соответствует общепринятому формату RGB.

Для установки палитры цветов служит команда colormap, записываемая в следующих формах:

• colormap('default') устанавливает палитру по умолчанию, при которой распределение цветов соответствует радуге;

• colormap(MAP) устанавливает палитру RGB, заданную матрицей MAP;

• C=colormap – функция возвращает матрицу текущей палитры цветов C.

m_файл с именем **colormap** устанавливает свойства цветов для текущего графика.

Команда help graph3d наряду с прочим выводит полный список характерных палитр, используемых графической системой MATLAB:

• hsv – цвета радуги;
• hot – чередование черного, красного, желтого и белого цветов;
• gray – линейная палитра в оттенках серого цвета;
• bone – серые цвета с оттенком синего;
• copper – линейная палитра с оттенками меди;
• pink – розовые цвета с оттенками пастели;
• white – палитра белого цвета;
• flag – чередование красного, белого, синего и черного цветов;

• lines – палитра с чередованием цветов линий;
• colorcube – расширенная палитра RGB;
• jet – разновидность палитры HSV;
• prism – призматическая палитра цветов;
• cool – оттенки голубого и фиолетового цветов;
• autumn – оттенки красного и желтого цветов;
• spring – оттенки желтого и фиолетового цветов;
• winter – оттенки синего и зеленого цветов;
• summer – оттенки зеленого и желтого цветов.

Все эти палитры могут служить параметрами команды colormap, например colormap(hsv) фактически устанавливает то же, что и команда colormap('default').

Окраска поверхностей. Для окраски поверхностей используется команда shading, которая управляет объектами surface (поверхность) и patch (заплата), создаваемыми командами и функциями surf, mesh, pcolor, fill и fill3. Команда shading (затенение) работает с параметрами и имеет следующий вид:

• shading flat задает окраску ячеек или граней в зависимости от текущих данных;
• shading interp задает окраску с билинейной интерполяцией цветов;
• shading faceted – равномерная раскраска ячеек поверхности (принята по умолчанию).

Эти команды устанавливают свойства EdgeColor и FaceColor для графических объектов surface и patch в зависимости от того, какая из команд – mesh (сетчатая поверхность) или surf (затененная поверхность) – используется.

Другие команды управления световыми эффектами. Отметим некоторые дополнительные команды, связанные с управлением цветовыми палитрами:

• colstyle – выделение цвета и стиля графика из заданного массива;
• rgbplot – изображение палитры;
• hsv2rgb – преобразование палитры HSV в палитру RGB;
• rgb2hsv – преобразование палитры RGB в палитру HSV;
• brighten – управление яркостью;
• contrast – управление контрастом;
• hidden – управление показом невидимых линий;
• whitebg – управление цветом фона.

Ознакомиться с информацией об этих командах можно с помощью команды help.

Анимационная графика

Движение точки на плоскостию Для отображения движения точки по траектории используется команда comet. Используются следующие формы представления этой команды:

• comet(Y) отображает движение «кометы» по траектории, заданной вектором Y;

• comet(X,Y) отображает движение «кометы» по траектории, заданной парой векторов Y и X;

• comet(X,Y,p) аналогична предшествующей команде, но позволяет задавать длину хвоста кометы (отрезка траектории, выделенного цветом) как p*length(Y), где length(Y) – размер вектора Y, а p<1. По умолчанию p = 0.11.

```
>> X=0:0.01:15; comet(X,sin(X),0.15)
```

Движение точки в пространстве. Команда comet3 позволяет наблюдать движение точки в трехмерном пространстве:

• comet3(Z) отображает движение точки с цветным «хвостом» по трехмерной кривой, определенной массивом Z;

• comet3(X,Y,Z) отображает движение точки «кометы» по кривой в пространстве, заданной точками [X(i),Y(i),Z(i)];

• comet3(X,Y,Z,p) аналогична предшествующей команде с заданием длины «хвоста кометы» как p*length(Z). По умолчанию параметр p равен 0.1.

```
>> W=0:pi/500:10*pi; comet3(cos(W),sin(W)+W/10,W)
```

Основные средства анимации

Для более сложных случаев анимации возможно применение техники мульти_
пликации. Она сводится к построению ряда кадров изображения, причем каждый
кадр появляется на некоторое время, затем стирается и заменяется на новый кадр,
несколько отличающийся от предшествующего. Если это отличие незначительно,
то создается иллюзия плавного перемещения объекта.

Отметим кратко основные команды, реализующие анимацию в системе MATLAB:

• capture – захват видеоизображения;
• getframe – создание кадра для анимации;
• moviein – выполнение анимации;
• rotate – вращение фигуры;
• frame2im – преобразование кадра в графический образ;
• im2frame – преобразование графического образа в кадр.

Применение некоторых из этих команд мы рассмотрим далее на конкретных примерах.