СИСТЕМА ИНЖЕНЕРНЫХ И НАУЧНЫХ РАСЧЕТОВ MATLAB 5.X:
В 2-Х Т. ТОМ 2.

Потемкин В. Г.

В книге дано наиболее полное описание системы MATLAB, предназначенной для выполнения инженерных и научных расчетов и высококачественной визуализации получаемых результатов. Эта система применяется в математике, вычислительном эксперименте, имитационном моделировании.
Справочное пособие предназначено для инженеров, аспирантов и исследователей, выполняющих научные исследования и инженерные разработки, а также для студентов при выполнении исследовательских работ, курсовых и дипломных проектов.

ОГЛАВЛЕНИЕ

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

8. АНАЛИЗ И ОБРАБОТКА ДАННЫХ

В этой главе описаны функции системы MATLAB, которые предназначены для анализа и обработки данных, заданных в виде числовых массивов. Наряду с простейшими функциями вычисления среднего, медианы, коэффициентов корреляции элементов массива рассмотрены функции вычисления конечных разностей, градиента и аппроксимации Лапласиана. В отдельный раздел выделены функции аппроксимации и интерполяции, а также геометрического анализа данных. Представлены функции численного интегрирования, решения задачи Коши для систем ОДУ, а также минимизации функций одной и нескольких переменных. В состав функций обработки сигналов включены дискретное преобразование Фурье, функции свертки и фильтрации. В полном объеме функции обработки сигналов оформлены в виде специализированного пакета программ Signal Processing Toolbox.

Основные операции

SUM, CUMSUM — **Суммирование элементов массива**

Синтаксис:

```
sx = sum(X)                csx = cumsum(X)
sx = sum(X, dim)           csx = cumsum(X, dim)
```

Описание:

Функция sx = sum(X) в случае одномерного массива возвращает сумму элементов массива; в случае двумерного массива - это вектор-строка, содержащая суммы элементов каждого столбца; в случае многомерного массива - это суммы элементов в столбцах вдоль первой неединичной размерности.

Функция sx = sum(X, dim) возвращает вектор-строку, содержащую суммы элементов по столбцам вдоль размерности dim.

Функции csx = cumsum(X) и csx = cumsum(X, dim), кроме того, возвращают все промежуточные результаты суммирования.

Пример:

Рассмотрим трехмерный массив M размера 3×3×3, составленный из следующих матриц:

```
M(:, :, 1) = magic(3); M(:, :, 2) = pascal(3); M(:, :, 3) = hilb(3);
M = M
M(:, :, 1) =
     8     1     6
     3     5     7
     4     9     2
```

```
M(:, :, 2) =
     1     1     1
     1     2     3
     1     3     6
M(:, :, 3) =
     1    1/2   1/3
    1/2   1/3   1/4
    1/3   1/4   1/5
```

Вычислим частичные и полные суммы элементов вдоль третьей размерности:

cmx = cumsum(M, 3)	sx = sum(M, 3)
cmx(:, :, 1) = 8 1 6 3 5 7 4 9 2 cmx(:, :, 2) = 9 2 7 4 7 10 5 12 8 cmx(:, :, 3) = 10 5/2 22/3 9/2 22/3 41/4 16/3 9/4 41/5	sx = 10 5/2 22/3 9/2 22/3 41/4 16/3 49/4 41/5

Сопутствующие функции: CUMPROD, PROD.

PROD, CUMPROD **Произведение элементов массива**

Синтаксис:

px = prod(X) cpx = cumprod(X)
px = prod(X, dim) cpx = cumprod(X, dim)

Описание:

Функция px = prod(X) в случае одномерного массива возвращает произведение элементов массива; в случае двумерного массива - это вектор-строка, содержащая произведения элементов каждого столбца; в случае многомерного массива - это произведения элементов в столбцах вдоль первой неединичной размерности.

Функция px = prod(X, dim) возвращает вектор-строку, содержащую произведения элементов по столбцам вдоль размерности dim.

Функции cpx = cumprod(X) и cpx = cumprod(X, dim), кроме того, возвращают все промежуточные результаты умножения.

Пример:

Рассмотрим трехмерный массив M размера 3×3×3, составленный из следующих матриц

```
M(:, :, 1) = magic(3); M(:, :, 2) = pascal(3); M(:, :, 3) = hilb(3);
M = M
```

```
M(:, :, 1) =
     8     1     6
     3     5     7
     4     9     2
M(:, :, 2) =
     1     1     1
     1     2     3
     1     3     6
M(:, :, 3) =
    1     1/2   1/3
   1/2    1/3   1/4
   1/3    1/4   1/5
```

Вычислим частичные и полные суммы элементов вдоль третьей размерности:

cpx = cumprod(M, 3)	px = prod(M, 3)
cpx(:, :, 1) = 8 1 6 3 5 7 4 9 2 cpx(:, :, 2) = 8 1 6 3 10 21 4 27 12 cpx(:, :, 3) = 8 1/2 2 3/2 10/3 21/4 4/3 27/4 12/5	px = 8 1/2 2 3/2 10/3 21/4 4/3 27/4 12/5

Сопутствующие функции: CUMSUM, SUM.

SORT, SORTROWS **Сортировка элементов массива**

Синтаксис:

```
Y = sort(X)                Y = sortrows(X)
Y = sort(X, dim)           Y = sortrows(X, column)
[Y, I] = sort(X)           [ Y, I] = sortrows(X)
```

Описание:

Функция Y = sort(X) в случае одномерного массива упорядочивает элементы массива по возрастанию; в случае двумерного массива происходит упорядочение элементов каждого столбца; для многомерных массивов сортировка выполняется вдоль первой неединичной размерности. Если X массив строковых ячеек, то реализуется лексикографическое упорядочение элементов.

Функция Y = sort(X, dim) выполняет сортировку вдоль указанной размерности. Если dim вектор, то сортировка выполняется последовательно вдоль указанных размерностей, то есть sort(X, [1 2]) эквивалентно sort(sort(X, 2), 1).

Функция [Y, I] = sort(X) кроме массива упорядоченных элементов возвращает массив индексов, позволяющих восстановить структуру исходного массива.

Если анализируемый массив содержит комплексные элементы, то сначала элементы сортируются по модулю, а в случае равенства модулей - по значениям фазового угла в диапазоне [-π π]. Если массив содержит элементы NaN, они размещаются в конце сортируемого массива.

Функция Y = sortrows(X) сортирует строки матрицы в порядке возрастания элементов первого столбца; аргумент X может быть либо матрицей, либо вектором-столбцом. Если X матрица строк, то они сортируются лексикографически. Если массив содержит комплексные элементы, то сначала они сортируются по модулю, а в случае равенства модулей - по значениям фазового угла в диапазоне [-π π].

Функция Y = sortrows(X, column) выполняет сортировку элементов указанного столбца. Если column вектор, то сортировка выполняется последовательно по указанным столбцам, то есть sort(X, [1 2]) эквивалентно первоначальной сортировке по элементам первого столбца, а в случае равенства элементов реализуется сортировка по элементам второго столбца.

Функция [Y, I] = sortrows(X) кроме массива упорядоченных элементов возвращает массив индексов, позволяющих восстановить структуру исходного массива.

Пример:

Рассмотрим массив M = magic(3).

```
M = 8   1   6
    3   5   7
    4   9   2
```

[Y, I] = sort(M)

```
Y = 3   1   2          I = 2   1   3
    4   5   6              3   2   1
    8   9   7              1   3   2
```

[Y, I] = sortrows(M)

```
Y = 3   5   7          I = 2
    4   9   2              3
    8   1   6              1
```

Сопутствующие функции: MIN, MAX, MEAN, MEDIAN.

Переопределяемые методы:

help cell/sort.m

MAX — Определение максимальных элементов массива

Синтаксис:

```
C = max(A)
C = max(A, B)
C = max(A, [ ], dim)
[C, I] = max(...)
```

Описание:

Функция C = max(A) в случае одномерного массива возвращает наибольший элемент; в случае двумерного массива - это вектор-строка, содержащая максимальные элементы каждого столбца; в случае многомерного массива - это вектор максимальных элементов вдоль первой неединичной размерности.

Функция C = max(A, B) возвращает массив C тех же размеров, какие имеют массивы A и B, каждый элемент которого есть максимальный из соответствующих элементов этих массивов.

Функция C = max(A, [ ], dim) возвращает вектор максимальных элементов вдоль размерности dim.

Функция [Y, I] = max(...) кроме самих максимальных элементов возвращает вектор-строку индексов этих элементов в данном столбце.

Если массив содержит комплексные элементы, то максимальные элементы определяются из условия max(abs(A)).

Замечание:

Если массив содержит один или несколько элементов типа NaN, то они во внимание не принимаются; в версиях 4.x результатом было NaN.

Пример:

Рассмотрим массив M = magic(3).

```
M =
   8   1   6
   3   5   7
   4   9   2
```

y = max(M)	[y, I] = max(M)	max(max(M))
y = 8 9 7	y = 8 9 7 I = 1 3 2	9

Сопутствующие функции: MIN, SORT.

MIN **Определение минимальных элементов массива**

Синтаксис:

```
C = min(A)
C = min(A, B)
C = min(A, [ ], dim)
[C, I] = min(...)
```

Описание:

Функция C = min(A) в случае одномерного массива возвращает наименьший элемент; в случае двумерного массива - это вектор-строка, содержащая минимальные элементы каждого столбца; в случае многомерного массива - это вектор минимальных элементов вдоль первой неединичной размерности.

Функция C = min(A, B) возвращает массив C тех же размеров, какие имеют массивы A и B, каждый элемент которого есть минимальный из соответствующих элементов этих массивов.

Функция C = min(A, [ ], dim) возвращает вектор минимальных элементов вдоль размерности dim.

Функция [Y, I] = min(...) кроме самих минимальных элементов возвращает вектор-строку индексов этих элементов в данном столбце.

Если массив содержит комплексные элементы, то минимальные элементы определяются из условия min(abs(A)).

Замечание:

Если массив содержит один или несколько элементов типа NaN, то они во внимание не принимаются; в версиях 4.x результатом было NaN.

Пример:

Рассмотрим массив M = magic(3).

```
M =
   8   1   6
   3   5   7
   4   9   2
```

y = min(M)	[y, I] = min(M)	min(min(M))
y = 3 1 2	y = 3 1 2 I = 2 1 3	1

Сопутствующие функции: MAX, SORT.

MEDIAN

Определение срединных значений (медиан) элементов массива

Синтаксис:

```
mdx = median(X)
mdx = median(X, dim)
```

Описание:

Функция mdx = median(X) в случае одномерного массива возвращает значение срединного элемента; в случае двумерного массива - это вектор-строка, содержащая значения срединных элементов каждого столбца; в случае многомерного массива - это вектор срединных элементов вдоль первой неединичной размерности.

Функция mdx = median(X, dim) возвращает вектор срединных элементов вдоль размерности dim.

Пример:

Рассмотрим массив M = magic(4).

```
M =
      16     2     3    13
       5    11    10     8
       9     7     6    12
       4    14    15     1

mdx = median(M)                    median(median(M))
mdx =    7     9     8    10       ans = 8.5000
```

Сопутствующие функции: MEAN, STD, COV, CORRCOEF.

MEAN — Определение средних значений элементов массива

Синтаксис:

```
mx = mean(X)
mx = mean(X, dim)
```

Описание:

Функция mx = mean(X) в случае одномерного массива возвращает арифметическое среднее элементов массива; в случае двумерного массива - это вектор-строка, содержащая арифметические средние элементов каждого столбца; в случае многомерного массива - это вектор арифметических средних элементов вдоль первой неединичной размерности.

Функция mx = mean(X, dim) возвращает вектор арифметических средних вдоль размерности dim.

Пример:

Рассмотрим массив M = magic(4).

```
M = 16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1

mx = mean(M)                                    mean(mean(M))
mx = 8.5000   8.5000   8.5000   8.5000         ans = 8.5000
```

Сопутствующие функции: MEDIAN, STD, COV, CORRCOEF.

STD — Определение стандартных отклонений элементов массива

Синтаксис:

```
sx = std(X)
sx = std(X, flag)
sx = std(X, flag, dim)
```

Определения:

В статистике используется два определения для стандартного отклонения σ:

$$\sigma = \left(\frac{1}{n-1} \sum_{i=1}^{n} \left(x_i - \hat{x} \right)^2 \right)^{1/2}; \quad (1)$$

$$\sigma = \left(\frac{1}{n} \sum_{i=1}^{n} \left(x_i - \hat{x} \right)^2 \right)^{1/2}, \quad (2)$$

где $\hat{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$, n - количество элементов выборки.

Описание:

Функция sx = std(X) в случае одномерного массива возвращает стандартное отклонение элементов массива; если X - выборка из набора случайных данных, распределенных по нормальному закону, то $(sx)^2$ - наилучшая несмещенная оценка дисперсии. В случае двумерного массива - это вектор-строка, содержащая стандартные отклонения элементов каждого столбца. В случае многомерного массива - это вектор стандартных отклонений элементов вдоль первой неединичной размерности.

Функция sx = std(X, flag) для значения flag, равного 0, аналогична std(X); для значения flag, равного 1, функция std(X, 1) возвращает стандартное отклонение согласно определению (2), соответствующему центрированной оценке дисперсии.

Функция sx = std(X, flag, dim) возвращает вектор стандартных отклонений вдоль размерности dim.

Пример:

Рассмотрим массив

```
M = magic(4)
M =
    16    2    3   13
     5   11   10    8
     9    7    6   12
     4   14   15    1
sx = std(M, 0)
sx =  5.4467   5.1962   5.1962   5.4467
sx = std(M, 1)
sx =  4.7170   4.5000   4.5000   4.7170
```

Сопутствующие функции: CORRCOEF, COV, MEAN, MEDIAN.

COV — Определение ковариационной матрицы элементов массива

Синтаксис:

C = cov(X)
C = cov(X, flag)
C = cov(X, Y)
C = cov(X, Y, flag)

Определения:

Рассмотрим двумерный массив $X = [x_1\ x_2 \ldots x_n]$, где каждый столбец рассматривается как переменная, а каждая строка - как наблюдение. Тогда матрица ковариаций cov(X) определяется следующим образом:

$$C = \mathrm{cov}(X) = \frac{1}{n-1}\begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \ldots & x_1^T x_n \\ x_2^T x_1 & x_2^T x_2 & \ldots & x_2^T x_3 \\ \ldots & \ldots & \ldots & \ldots \\ x_n^T x_1 & x_n^T x_2 & \ldots & x_n^T x_n \end{bmatrix}; \qquad (1)$$

$$C = \mathrm{cov}(X) = \frac{1}{n}\begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \ldots & x_1^T x_n \\ x_2^T x_1 & x_2^T x_2 & \ldots & x_2^T x_3 \\ \ldots & \ldots & \ldots & \ldots \\ x_n^T x_1 & x_n^T x_2 & \ldots & x_n^T x_n \end{bmatrix}. \qquad (2)$$

Описание:

Функция C = cov(X) в случае одномерного массива возвращает дисперсию элементов массива; в случае двумерного массива, когда каждый столбец рассматривается как переменная, а каждая строка - как наблюдение, cov(X) - это матрица ковариаций, diag(cov(X)) - вектор дисперсий, sqrt(diag(cov(X))) - вектор стандартных отклонений для каждого столбца.

Функция C = cov(X, Y), где массивы X и Y имеют одинаковое количество строк, равносильна функции cov([X Y]).

Функции C = cov(X, flag) и C = cov(X, Y, flag) при значении параметра flag, равного 0, вычисляют ковариационные матрицы в соответствии с соотношением (1), а при значении параметра flag, равного 1, - в соответствии с соотношением (2).

Пример:

Рассмотрим массив

```
M = magic(4)/norm(magic(4))
M =
   0.4706   0.0588   0.0882   0.3824
   0.1471   0.3235   0.2941   0.2353
   0.2647   0.2059   0.1765   0.3529
   0.1176   0.4118   0.4412   0.0294
```

C = cov(M)				diag(cov(M))	sqrt(diag(cov(M)))
0.0257	-0.0239	-0.0222	0.0205	0.0257	0.1602
-0.0239	0.0234	0.0228	-0.0222	0.0234	0.1528
-0.0222	0.0228	0.0234	-0.0239	0.0234	0.1528
0.0205	-0.0222	-0.0239	0.0257	0.0257	0.1602

Алгоритм:

Вычисление матрицы ковариаций реализуется следующим алгоритмом:

```
[m, n] = size(X);
X = X - ones(m, 1) * mean(X);
C = X' * X/(n - 1);
```

Сопутствующие функции: CORRCOEF, MEAN, STD.

CORRCOEF — Определение коэффициентов корреляции элементов массива

Синтаксис:

```
S = corrcoef(X)
S = corrcoef(X, Y)
```

Описание:

Функция S = corrcoef(X) возвращает матрицу коэффициентов корреляции для двумерного массива, когда каждый столбец рассматривается как переменная, а каждая строка - как наблюдение.

Элементы матрицы S = corrcoef(X) связаны с элементами матрицы ковариаций C = cov(X) следующим соотношением:

$$S(i, j) = \frac{C(i, j)}{\sqrt{C(i, i)\,C(j, j)}}.$$

Функция S = corrcoef(X, Y), где массивы X и Y имеют одинаковое количество строк, равносильна функции corrcoef([X Y]).

Пример:

Рассмотрим массив

```
M = magic(4)/norm(magic(4))
M =
   0.4706   0.0588   0.0882   0.3824
   0.1471   0.3235   0.2941   0.2353
   0.2647   0.2059   0.1765   0.3529
   0.1176   0.4118   0.4412   0.0294
```

S = corrcoef(M)				C = cov(M)			
1.0000	-0.9776	-0.9069	0.7978	0.0257	-0.0239	-0.0222	0.0205
-0.9776	1.0000	0.9753	-0.9069	-0.0239	0.0234	0.0228	-0.0222
-0.9069	0.9753	1.0000	-0.9776	-0.0222	0.0228	0.0234	-0.0239
0.7978	-0.9069	-0.9776	1.0000	0.0205	-0.0222	-0.0239	0.0257

Сопутствующие функции: COV, MEAN, STD.

DIFF

Вычисление конечных разностей и приближенное дифференцирование

Синтаксис:

```
Y = diff(X)
Y = diff(X, n)
Y = diff(X, n, dim)
```

Описание:

Функция y = diff(x) вычисляет конечные разности. Если x - одномерный массив вида x = [x(1) x(2) ... x(n)], то diff(x) - это вектор разностей соседних элементов diff(x) = [x(2) - x(1) x(3) - x(2) ... x(n) - x(n -1)]. Количество элементов вектора x на единицу меньше количества элементов вектора diff(x). Если X - двумерный массив, то берутся разности столбцов diff(X) = X(2:m, :) - X(1:m-1, :).

Функция y = diff(x, n) вычисляет конечные разности порядка n, удовлетворяющие рекуррентному соотношению diff(x, n) = diff(x, n -1). Аппроксимацией производной n порядка является отношение diff(y, n)./diff(x, n).

Функция Y = diff(X, n, dim) вычисляет конечные разности порядка n вдоль диагонали dim. Если порядок равен или превышает размер вдоль указанной размерности, то функция diff возвращает пустой массив.

Замечание:

Поскольку каждая операция дифференцирования уменьшает размер массива X вдоль размерности dim, то может случиться так, что при достаточно большом n размер массива вдоль указанной размерности окажется единичным. Когда такое происходит, оператор дифференцирования начинает использовать данные, принадлежащие следующей неединичной размерности.

При наличии специализированного пакета Symbolic Math Toolbox [1] возможно реализовать точное дифференцирование в символьном виде, используя следующие функции пакета:

- diff(S) дифференцирует символьное выражение S по свободной переменной;
- diff(S, 'v') дифференцирует символьное выражение S по v;
- diff(S, n) и diff(S, 'v', n) дифференцируют n раз символьное выражение S;
- diff без аргументов дифференцирует предшествующее выражение.

Сопутствующие функции: GRADIENT, DEL2.

Ссылки:

1. Symbolic Mathematics Toolbox. User's Guide. Natick, MA: The MathWorks, Inc., 1997.

GRADIENT

Приближенное вычисление градиента функции

Синтаксис:

```
[px, py] = gradient(F)
[px, py, pz, ...] = gradient(F)
[...] = gradient(F, h)
[...] = gradient(F, h1, h2, ...)
```

Определения:

Градиент скалярной функции F(x, y), заданной на двумерной сетке, определяется следующим образом:

$$\mathbf{grad}(F) = \nabla F = \frac{\partial F(x,y)}{\partial x}\vec{\mathbf{i}} + \frac{\partial F(x,y)}{\partial y}\vec{\mathbf{j}}.$$

Градиент скалярной функции от n переменных F(x, y, z, ...) задается следующим образом:

$$\mathbf{grad}(F) = \nabla F = \frac{\partial F}{\partial x}\vec{\mathbf{i}} + \frac{\partial F}{\partial y}\vec{\mathbf{j}} + \frac{\partial F}{\partial z}\vec{\mathbf{k}} + \ldots.$$

Описание:

Функция [px, py] = gradient(F) вычисляет градиент функции F(x, y), заданной на двумерной сетке и представляющей собой массив чисел.

Функция [px, py, pz, ...] = gradient(F) вычисляет градиент функции от n переменных F(x, y, z, ...).

Функции вида [...] = gradient(F, h) вычисляют градиент функции, используя постоянный шаг h по всем переменным.

Функции вида [...] = gradient(F, h1, h2, ...) используют разный шаг разбиения сетки по переменным.

Пример:

Рассмотрим расчет и построение поля направлений для функции

$$F = xe^{-x^2-y^2}$$

с использованием функции gradient.

```
[x, y] = meshgrid(-2:.2:2, -2:.2:2);
z = x .* exp(-x.^2 - y.^2);
[px, py] = gradient(z, .2, .2);
contour(z), hold on, quiver(px, py), hold off
```

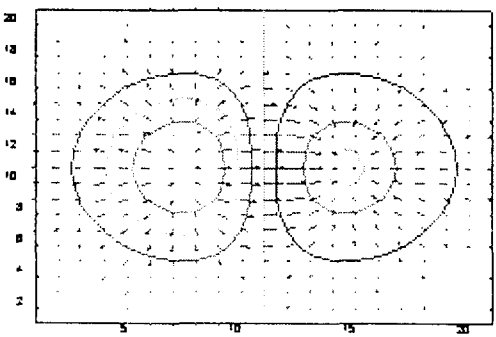


Рис. 8.1

Сопутствующие функции: DIFF, DEL2, QUIVER, CONTOUR.

DEL2 **Пятиточечная аппроксимация лапласиана**

Синтаксис:

L = del2(U)
L = del2(U, h)
L = del2(U, hx, hy)
L = del2(U, hx, hy, hz, ...)

Определения:

Если массив U рассматривать как функцию U(x, y), вычисленную в точках квадратной сетки, то 4del2(U) является конечно-разностной аппроксимацией дифференциального оператора Лапласа, примененного к функции U, то есть

$$4L = \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}.$$

Конечно-разностная аппроксимация оператора L имеет вид

$$L_{ij} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - u_{i,j}$$

внутри области, а на границе та же формула используется для кубической экстраполяции.

Для функции N переменных U(x, y, z, ...) оператор L определяется следующим образом:

$$L = \frac{\nabla^2 u}{2N} = \frac{1}{2N}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} + \ldots\right).$$

Описание:

Функция L = del2(U), где U - прямоугольный массив, возвращает дискретную аппроксимацию лапласиана в виде массива L того же размера, каждый элемент которого равен разности среднего значения четырех соседних элементов и элемента рассматриваемого узла. Узлы сетки во внутренней области имеют четырех соседей, а на границе и в углах - только трех или двух соседей. В случае многомерного массива возвращается дискретная аппроксимация оператора $\frac{\nabla^2 u}{2N}$, где N = ndims(u).

Функция L = del2(U, h) вычисляет аппроксимацию лапласиана, используя постоянный шаг h (по умолчанию h равно 1).

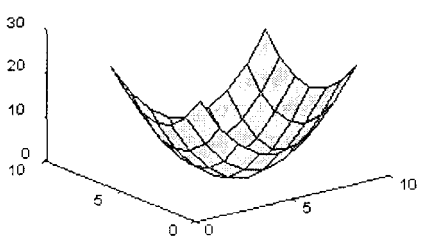
Функция L = del2(U, hx, hy), где U - прямоугольный массив, использует сетку, заданную переменными hx и hy. Если hx скаляр, то он определяет расстояние между узлами вдоль оси x; если hx вектор, то он задает x-координаты узлов и должен иметь длину size(u, 2). Аналогично если hy скаляр, то он определяет расстояние между узлами вдоль оси y; если hy вектор, то он задает y-координаты узлов и должен иметь длину size(u, 1).

Функция L = del2(U, hx, hy, hz, ...), где U - многомерный массив, использует разный шаг разбиения сетки по переменным.

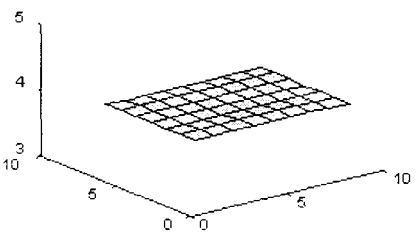
Пример:

Функция u(x, y) = $x^2 + y^2$ имеет лапласиан, равный $\Delta^2 u = 4$, в чем можно убедиться, взглянув на графики этих функций.

```
[x, y] = meshgrid(-4:4, -3:3);
U = x.* x + y.* y,
U =
     25    18    13    10     9    10    13    18    25
     20    13     8     5     4     5     8    13    20
     17    10     5     2     1     2     5    10    17
     16     9     4     1     0     1     4     9    16
     17    10     5     2     1     2     5    10    17
     20    13     8     5     4     5     8    13    20
     25    18    13    10     9    10    13    18    25
V = 4 * del2(U),
V =
     4     4     4     4     4     4     4     4     4
     4     4     4     4     4     4     4     4     4
     4     4     4     4     4     4     4     4     4
     4     4     4     4     4     4     4     4     4
     4     4     4     4     4     4     4     4     4
     4     4     4     4     4     4     4     4     4
     4     4     4     4     4     4     4     4     4
surfl(U), surfl(V)
```



а



б

Рис. 8.2

Сопутствующие функции: GRADIENT, DIFF.

Аппроксимация и интерполяция данных

POLYFIT — **Аппроксимация данных полиномом**

Синтаксис:

```
p = polyfit(x, y, n)
[p, S] = polyfit(x, y, n)
```

Описание:

Функция p = polyfit(x, y, n) находит коэффициенты полинома p(x) степени n, который аппроксимирует функцию y(x) в смысле метода наименьших квадратов. Выходом является строка p длины n+1, содержащая коэффициенты аппроксимирующего полинома p(x) в порядке убывания степеней

$$p(x) = p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1}.$$

Функция [p, S] = polyfit(x, y, n) возвращает коэффициенты полинома p и массив записей S, который можно использовать совместно с функцией polyval для оценки погрешностей или предсказания. Если ошибки задания исходной функции y(x) независимы и распределены по нормальному закону с постоянной дисперсией, то функция polyval обеспечивает 50-процентный доверительный интервал.

Пример:

Рассмотрим аппроксимацию функции ошибки erf(x), которая является ограниченной сверху функцией, в то время как аппроксимирующие полиномы неограниченны, что приводит к ошибкам аппроксимации.

```
x = (0:0.1:2.5)';
y = erf(x);
```

вычислим коэффициенты аппроксимирующего полинома степени 6:

```
p = polyfit(x, y, 6)
p = 0.0084  -0.0983   0.4217  -0.7435   0.1471   1.1064   0.0004
```

вычислим значения полинома в точках сетки:

```
f = polyval(p, x);
```

сформируем следующую таблицу данных:

```
table = [x y f y-f]
table =
```

0	0	0.0004	-0.0004	2.1000	0.9970	0.9969	0.0001
0.1000	0.1125	0.1119	0.0006	2.2000	0.9981	0.9982	-0.0001
0.2000	0.2227	0.2223	0.0004	2.3000	0.9989	0.9991	-0.0003
0.3000	0.3286	0.3287	-0.0001	2.4000	0.9993	0.9995	-0.0002
0.4000	0.4284	0.4288	-0.0004	2.5000	0.9996	0.9994	0.0002
.....	.....	.....	.....				

Из таблицы видно, что на отрезке [0 2.5] точность аппроксимации находится в пределах 3-4 знаков; построим графики функции и аппроксимирующего полинома на отрезке [0 5].

```
x = (0:0.1:5)';
y = erf(x);
f = polyval(p, x);
plot(x, y, 'ob', x, f, '-g'),  axis([0 5 0 2]),  grid
```

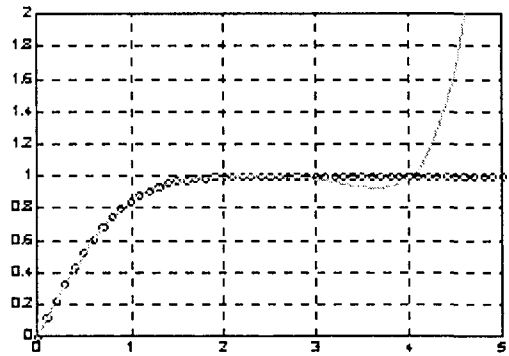


Рис. 8.3

Как следует из анализа графика, аппроксимация вне отрезка [0 2.5] расходится.

Алгоритм:

Аппроксимация полиномом связана с вычислением матрицы Вандермонда V, элементами которой являются базисные функции

$$v_{ij} = x_i^{n-j},$$

и последующим решением переопределенной системы линейных уравнений

Vp = y.

Пользователь может самостоятельно внести изменения в M-файл polyfit, чтобы применить другие базисные функции.

Сопутствующие функции: POLY, POLYVAL, ROOTS.

INTERPFT

Аппроксимация периодической функции на основе быстрого преобразования Фурье

Синтаксис:

```
yp = interpft(y, n)
Yp= interpft(Y, n, dim)
```

Описание:

Функция yp = interpft(y, n) возвращает одномерный массив чисел, который является периодической функцией, определенной в n точках и аппроксимирующей одномерный массив y. Если length(y) = m, а интервал дискретности dy,

то интервал дискретности для yp определяется по формуле dyp = dy * m/n, причем n всегда превышает m.

Если входной аргумент Y двумерный массив, то обрабатываются столбцы и возвращается массив Yp с тем же количеством столбцов, а количество строк равно n.

Функция Yp= interpft(Y, n, dim) выполняет аппроксимацию данных, записанных вдоль указанной размерности многомерного массива.

Пример:

Рассмотрим аппроксимацию функции y = sin(x), которая задана 11 точками на интервале [0 10].

```
x = 0:10; y = sin(x);
xp = 0:0.25:10;
yp = interpft(y, 41);
xt = 0:0.01:10;  yt = sin(xt);
plot(xt, yt, 'r'),  hold on,  plot(x, y, 'ob', xp, yp),  grid
```

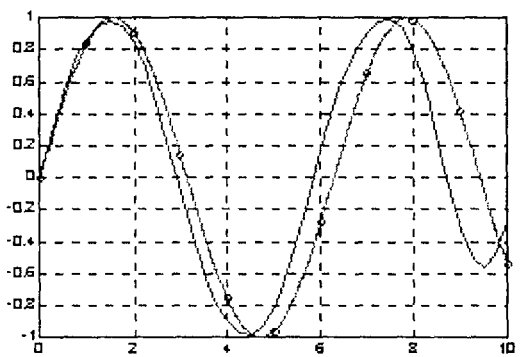


Рис. 8.4

На графике построена точная функция y = sin(x) с указанием точек съема данных и ее аппроксимация в 41 точке. Как видно из графика, аппроксимация вне интервала [0 1.5] имеет нарастающую погрешность.

Сопутствующие функции: INTERP1.

SPLINE, PPVAL, MKPP, UNMKPP

Интерполяция функции одной переменной кубическим сплайном

Синтаксис:

```
yi = spline(x, y, xi)
pp = spline(x, y)
```

```
v = ppval(pp, xx)
[breaks, coefs, l, k] = unmkpp(pp)
 pp = mkpp(breaks, coefs)
```

Описание:

Функция yi = spline(x, y, xi) интерполирует значения функции y в точках xi внутри области определения функции, используя кубические сплайны [1].

Функция pp = spline(x, y) возвращает pp-форму сплайна, используемую в M-файлах ppval, mkpp, unmkpp.

Функция v = ppval(pp, xx) вычисляет значение кусочно-гладкого полинома pp для значений аргумента xx.

Функция [breaks, coefs, l, k] = unmkpp(pp) возвращает характеристики кусочно-гладкого полинома pp:

breaks - вектор разбиения аргумента;

coefs - коэффициенты кубических сплайнов;

l = length(breaks) - 1;

k = length(coefs)/l.

Функция pp = mkpp(breaks, coefs) формирует кусочно-гладкий полином pp по его характеристикам.

Пример:

Зададим синусоиду всего 10 точками и проведем интерполяцию кубическими сплайнами, используя мелкую сетку.

```
x = 0:10; y = sin(x);
xi = 0:.25:10;
yi = spline(x, y, xi);
plot(x, y, 'o', xi, yi, 'g'), grid
```

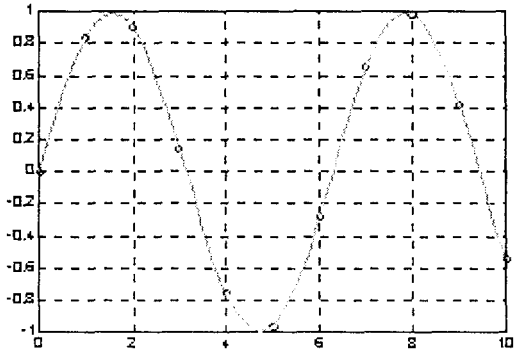


Рис. 8.5

Определим pp-форму сплайна.

```
pp = spline(x, y);
[breaks, coeffs, l, k] = unmkpp(pp)
breaks =    0    1    2    3    4    5    6    7    8    9   10
```

```
coeffs =
  -0.0419  -0.2612   1.1446   0
  -0.0419  -0.3868   0.4965   0.8415
   0.1469  -0.5124  -0.4027   0.9093
   0.1603  -0.0716  -0.9867   0.1411
   0.0372   0.4095  -0.6488  -0.7568
  -0.1234   0.5211   0.2818  -0.9589
  -0.1684   0.1509   0.9538  -0.2794
  -0.0640  -0.3542   0.7506   0.6570
   0.1190  -0.5463  -0.1499   0.9894
   0.1190  -0.1894  -0.8856   0.4121
l =   10
k =    4
```

Вычислим pp-форму в узловых точках сетки.

```
v = ppval(pp,x)
v = 0 0.8415 0.9093 0.1411 -0.7568 -0.9589 -0.2794 0.6570 0.9894 0.4121 -0.5440
```

Алгоритм:

Интерполяция сплайнами использует вспомогательные функции ppval, mkpp, unmkpp, которые образуют небольшой пакет для работы с кусочно-гладкими полиномами.

Существенно большие возможности для работы со сплайнами предоставляет пользователю специализированный пакет Spline Toolbox [2].

Сопутствующие функции: INTERP1, INTERP2, INTERP3, INTERPN, PPVAL, Spline Toolbox.

Ссылки:

1. Carl de Boor. A Practical Guide to Splines. Berlin, 1978.
2. *Spline Toolbox.* User's Guide. Natick, MA: The MathWorks, Inc., 1992.

INTERP1, INTERP1Q — Одномерная табличная интерполяция

Синтаксис:

```
yi = interp1(x, y, xi)
yi = interp1(x, y, xi, '<метод>')
yi = interp1q(x, y, xi)
```

Описание:

Функция yi = interp1(x, y, xi) строит кусочно-линейную интерполирующую кривую для одномерного массива y, заданного на сетке x; выходной массив yi может быть определен на более мелкой сетке xi. Если Y - двумерный массив, то интерполирующая кривая строится для каждого столбца и массив Yi будет иметь размер length(xi)×size(Y, 2). Для значений xi вне области задания x присваивается значение NaN.

Функция yi = interp1(x, y, xi, '<*метод*>') позволяет задать метод интерполяции:

'nearest' - ступенчатая интерполяция;
'linear' - линейная интерполяция;
'cubic' - кубическая интерполяция;
'spline' - кубические сплайны.

Для всех интерполяционных методов предполагается, что аргумент x изменяется монотонно. Если сетка является равномерной, то можно применить ускоренную интерполяцию, используя методы '*linear', '*cubic', '*nearest', '*spline'.

Специальная функция yi = interp1q(x, y, xi) реализует очень быструю линейную интерполяцию одномерной функции. Если Y - двумерный массив, то интерполирующая кривая строится для каждого столбца и массив Yi будет иметь размер length(xi)×size(Y, 2). Для значений xi вне области задания x присваивается значение NaN.

Функция interp1q работает на неравномерной сетке быстрее, чем функция interp1. Однако функция interp1(..., '*linear') работает на неравномерной сетке еще быстрее.

Пример:

Зададим синусоиду всего 10 точками и проведем интерполяцию, используя мелкую сетку.

```
x = 0:10; y = sin(x);
xi = 0:.25:10;
yi = interp1(x, y, xi);
plot(x, y, 'o', xi, yi, 'g'), hold on
yi = interp1(x, y, xi, 'nearest' );
plot(x, y, 'ob', xi, yi, 'm')
yi = interp1(x, y, xi, 'spline' );
plot(x, y, 'ob', xi, yi, 'b'), grid, hold off
```

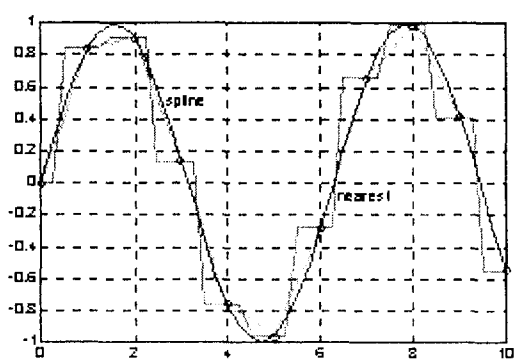


Рис. 8.6

Алгоритм:

Методы линейной и кубической интерполяции реализуются довольно просто; что же касается интерполяции сплайнами, то в этом случае используются вспомогательные функции ppval, mkpp, unmkpp, которые образуют небольшой пакет для работы с кусочно-гладкими полиномами.

Существенно большие возможности пользователям для решения проблем интерполяции и аппроксимации предоставляет специализированный пакет Spline Toolbox [1].

Сопутствующие функции: INTERPFT, INTERP2, INTERP3, INTERPN, SPLINE.

Ссылки:

1. *Spline Toolbox.* User's Guide. Natick, MA: The MathWorks, Inc., 1992.

INTERP2 **Двумерная табличная интерполяция**

Синтаксис:

ZI = interp2(X, Y, Z, XI, YI)
ZI = interp2(Z, XI, YI)
ZI = interp2(Z, ntimes)
ZI = interp2(X, Y, Z, XI, YI, '<*метод*>')

Описание:

Функция ZI = interp2(X, Y, Z, XI, YI) возвращает массив ZI, соответствующий массивам XI и YI и полученный интерполяцией данных, заданных в массивах X, Y и Z. Массивы X и Y должны быть монотонными и иметь формат, соответствующий функции meshgrid. Выходной массив ZI может быть определен на более мелкой сетке {XI, YI}, но если точки новой сетки выходят за пределы сетки {X, Y}, то им присваивается значение NaN.

Аргументы XI и YI могут быть массивами, и в этом случае значения ZI определяются в точках (XI(i, j), YI(i, j)). Наконец, можно задать xi как вектор-строку, а yi как вектор-столбец; тогда эти векторы определят сетку в формате функции meshgrid(xi, yi).

Функция ZI = interp2(Z, XI, YI) предполагает, что X = 1:n и Y = 1:m, где [m, n] = size(Z).

Функция ZI = interp2(Z, ntimes) выполняет рекурсивную интерполяцию, повторяя ее ntimes раз. Команда interp2(Z, 1) равносильна команде interp2(Z).

Функция ZI = interp2(X, Y, Z, XI, YI, '<*метод*>') позволяет задать метод интерполяции:

'nearest' - ступенчатая интерполяция;
'linear' - линейная интерполяция (по умолчанию);
'cubic' - кубическая интерполяция;
'spline' - кубические сплайны.

Для всех интерполяционных методов предполагается, что аргументы X и Y изменяются монотонно и заданы в формате функции meshgrid. Если сетка является равномерной, то можно применить ускоренную интерполяцию, используя методы '*linear', '*cubic', '*nearest', '*spline'.

Пример:

Проведем интерполяцию функции peaks, используя мелкую сетку.

```
[X, Y] = meshgrid(-3:0.25:3);
Z = peaks(X, Y);
[XI, YI] = meshgrid(-3:0.125:3);
ZI = interp2(X, Y, Z, XI, YI);
mesh(X, Y, Z), hold on, mesh(XI, YI, ZI+15), hold off
```

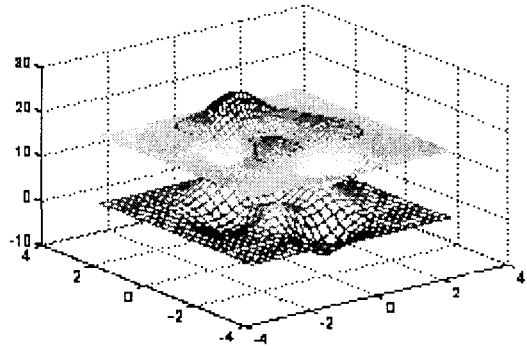


Рис. 8.7

Сопутствующие функции: INTERP1, INTERP3, INTERPN, MESHGRID, GRIDDATA.

INTERP3 **Трехмерная табличная интерполяция**

Синтаксис:

```
VI = interp3(X, Y, Z, V, XI, YI, ZI)
VI = interp3(V, XI, YI, ZI)
VI = interp3(V, ntimes)
VI = interp3(..., '<метод>')
```

Описание:

Функция VI = interp3(X, Y, Z, V, XI, YI, ZI) возвращает массив VI, соответствующий массивам XI, YI и ZI и полученный интерполяцией данных, заданных в массивах X, Y, Z и V. Выходной массив VI может быть определен на более мелкой сетке {XI, YI, ZI}, но если точки новой сетки выходят за пределы сетки {X, Y, Z}, то им присваивается значение NaN.

Аргументы XI, YI и ZI могут быть массивами, и в этом случае значения VI определяются в точках (XI(i, j), YI(i, j), ZI(i, j)). Наконец, можно задать xi, yi и zi в виде векторов, определяющих сетку, соответствующую функции meshgrid(xi, yi, zi).

Функция VI = interp3(V, XI, YI, ZI) предполагает, что X = 1:n, Y = 1:m и Z = 1:p, где [m, n, p] = size(V).

Функция VI = interp3(V, ntimes) выполняет рекурсивную интерполяцию, повторяя ее ntimes раз. Команда interp3(V, 1) равносильна команде interp3(V).

Функция VI = interp3(..., '*<метод>*') позволяет задать метод интерполяции:

'nearest' - ступенчатая интерполяция;
'linear' - линейная интерполяция (по умолчанию);
'cubic' - кубическая интерполяция;
'spline' - кубические сплайны.

Для всех интерполяционных методов предполагается, что аргументы X, Y и Z изменяются монотонно и заданы в формате функции meshgrid. Если сетка является равномерной, то можно применить ускоренную интерполяцию, используя методы '*linear', '*cubic', '*nearest', '*spline'.

Пример:

Проведем интерполяцию функции flow, используя мелкую сетку.

```
[X, Y, Z, V] = flow(10);
[XI, YI, ZI] = meshgrid(.1:.25:10, -3:.25:3, -3:.25:3);
VI = interp3(X, Y, Z, V, XI, YI, ZI); % Размер V - 31×41×27
slice(XI, YI, ZI, VI, [6 9.5], 2, [-2 .2]) shading flat
```

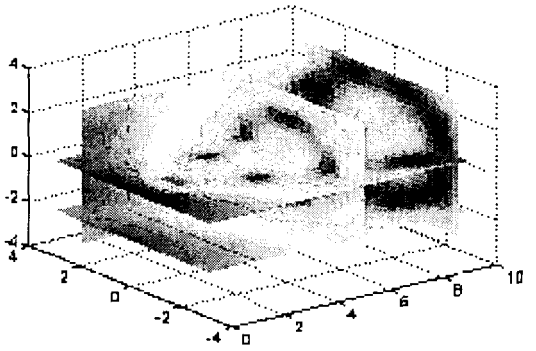


Рис. 8.8

Сопутствующие функции: INTERP1, INTERP2, INTERPN, MESHGRID.

INTERPN **Многомерная табличная интерполяция**

Синтаксис:

```
VI = interpn(X1, X2, X3, ..., V, Y1, Y2, Y3, ...)
VI = interpn(V, Y1, Y2, Y3, ...)
VI = interpn(V, ntimes)
VI = interpn(..., '<метод>')
```

Описание:

Функция VI = interpn(X1, X2, X3, ..., V, Y1, Y2, Y3, ...) возвращает массив VI, соответствующий массивам Y1, Y2, Y3, ... и полученный интерполяцией данных, заданных в массивах X1, X2, X3, ..., V. Выходной массив VI может быть определен на более мелкой сетке {Y1, Y2, Y3, ... }, но если точки новой сетки выходят за пределы сетки {X1, X2, X3, ..., }, то им присваивается значение NaN.

Аргументы Y1, Y2, Y3, ... могут быть массивами, и в этом случае значения VI определяются в точках (Y1(i, j), Y2(i, j), Y3(i, j), ...). Наконец, можно задать y1, y2, y3, ... в виде векторов, определяющих сетку, соответствующую функции ndgrid(y1, y2, y3, ...).

Функция VI = interpn(V, Y1, Y2, Y3, ...) предполагает, что X1 = 1:size(V,1), X2 = 1:size(V,2), X3 = 1:size(V,3) и т. д.

Функция VI = interpn(V, ntimes) выполняет рекурсивную интерполяцию, повторяя ее ntimes раз. Команда interpn(V, 1) равносильна команде interpn(V).

Функция VI = interpn(..., '*<метод>*') позволяет задать метод интерполяции:

'nearest' - ступенчатая интерполяция;
'linear' - линейная интерполяция (по умолчанию);
'cubic' - кубическая интерполяция;
'spline' - кубические сплайны.

Для всех интерполяционных методов предполагается, что аргументы X1, X2, X3, ..., изменяются монотонно и заданы в формате функции ndgrid. Если сетка является равномерной, то можно применить ускоренную интерполяцию, используя методы '*linear', '*cubic', '*nearest', '*spline'.

Сопутствующие функции: INTERP1, INTERP2, INTERP3, MESHGRID.

GRIDDATA — Двумерная интерполяция на неравномерной сетке

Синтаксис:

ZI = griddata(x, y, z, XI, YI)
[XI, YI, ZI] = griddata(x, y, z, XI, YI)
[...] = griddata(..., '*<метод>*')

Описание:

Функция ZI = griddata(x, y, z, XI, YI) возвращает массив ZI, который определен на новой сетке {XI, YI} в результате интерполяции исходной функции z, заданной на неравномерной сетке {x, y}. Аргументы XI и YI могут быть массивами, и в этом случае значения ZI определяются в точках (XI(i, j), YI(i, j)). Наконец, можно задать xi как вектор-строку, а yi - как вектор-столбец; тогда эти векторы определят сетку в формате функции meshgrid(xi, yi).

Функция [XI, YI, ZI] = griddata(x, y, z, XI, YI) кроме массива ZI возвращает массивы XI, YI, упакованные в формате функции meshgrid.

Функция [...] = griddata(..., '<*метод*>') позволяет задать метод интерполяции:

'linear' - линейная на основе триангуляции (по умолчанию);
'cubic' - кубическая интерполяция на основе триангуляции;
'nearest' - ступенчатая интерполяция;
'v4' - интерполяция, реализованная в версиях MATLAB 4.x.

Методы 'cubic' и 'v4' реализуют гладкую интерполяцию, в то время как методы 'linear' и 'nearest' дают разрывы соответственно по первой и второй производным. Все методы, за исключением 'v4' основаны на триангуляции Делоне.

Алгоритм:

Функция griddata(..., 'v4') использует метод, изложенный в работе [1]; остальные функции основаны на триангуляции Делоне [2].

Пример:

Определим функцию на сетке, заданной 100 точками, выбранными случайно на отрезке [-2 2].

```
rand('seed', 0)
x = rand(100, 1) * 4 - 2;
y = rand(100, 1) * 4 - 2;
z = x.*exp(-x.^2 - y.^2);
```

Векторы x, y, z определяют 100 случайных точек на поверхности функции ZI, которую зададим на следующей равномерной сетке:

```
ti = -2:0.25:2;
[XI, YI] = meshgrid(ti, ti);
ZI = griddata(x, y, z, XI, YI);
```

Построим поверхность функции, полученной в результате интерполяции на неравномерной случайной сетке.

```
mesh(XI, YI, ZI), hold on, plot3(x, y, z, 'or'), hold off
```

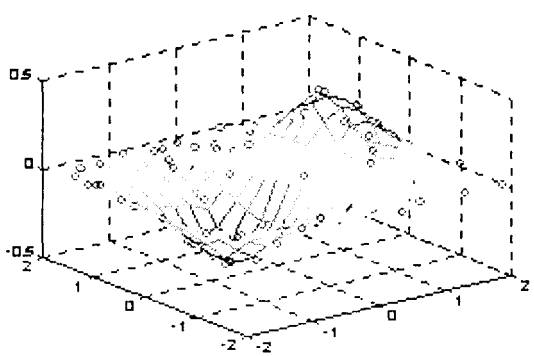


Рис. 8.9

Сопутствующие функции: DELAUNAY, INTERP2, MESHGRID.

Ссылки:
1. Sandwell D. T. Biharmonic Spline Interpolation of GEOS-3 and SEASAT Altimeter Data// *Geophysical Research Letters.* 1987. Vol. 2. P.139-142.
2. Watson D. E. *Contouring: A Guide to the Analysis and Display of Spatial Data,* Tarrytown; New York: Pergamon (Elsevier Science, Inc.), 1992.

Геометрический анализ данных

DELAUNAY — **Построение триангуляционной сетки**

Синтаксис:

```
TRI = delaunay(x, y)
TRI = delaunay(x, y, 'sorted')
k = dsearch(x, y, TRI, xi, yi)
k = dsearch(x, y, TRI, xi, yi, S)
T = tsearch(x, y, TRI, xi, yi)
```

Определение:

Пусть задано некоторое множество точек; тогда *триангуляцией Делоне* называется множество линий, соединяющих каждую точку с ее непосредственными соседями. Триангуляция Делоне связана с диаграммой Вороного, которая представляет собой многоугольник, вершинами которого являются центры кругов, описанных вокруг треугольников Делоне.

Описание:

Функция TRI = delaunay(x, y) возвращает набор треугольников, такой, что ни одна точка из числа заданных не принадлежит кругам, описанным относительно этих треугольников. Каждая строка матрицы TRI размера m×3 определяет один такой треугольник; вершины треугольника задаются как индексы соответствующих элементов векторов x и y, так что x(TRI) и y(TRI) - это координаты вершин.

Функция TRI = delaunay(x, y, 'sorted') кроме того выполняет сортировку точек, сначала по переменной y, а затем по переменной x, исключая повторяющиеся вершины.

Триангуляция Делоне создает триангуляционную сетку для случайно распределенных точек и используется как утилита функциями griddata, convhull, voronoi.

Функция k = dsearch(x, y, TRI, xi, yi) возвращает индекс точки (x, y) в массиве [x y], которая принадлежит триангуляции Делоне и является ближайшей к исследуемой точке (xi, yi).

Функция k = dsearch(x, y, TRI, xi, yi, S) использует специальную разреженную матрицу S= sparse(TRI(:,[1 1 2 2 3 3]), TRI(:, [2 3 1 3 1 2]), 1, nxy, nxy), где nxy = prod(size(x)), что позволяет избежать ее формирования в процессе вычислений.

Функция T = tsearch(x, y, TRI, xi, yi) возвращает номер строки массива TRI (иными словами, треугольник триангуляционной сетки), в котором находится точка (xi, yi). Если точка находится вне пределов триангуляционной сетки, то результатом будет NaN. Массив [x y] принадлежит триангуляции Делоне.

Пример:

Построить триангуляцию Делоне для 10 случайных точек.

```
rand('state',0);
x = rand(1, 10);
y = rand(1, 10);
TRI = delaunay(x, y);
subplot(1, 2, 1),
trimesh(TRI, x, y, zeros(size(x))); view(2), axis([0 1 0 1]); hold on;
plot(x, y, 'o');
set(gca, 'box', 'on');
```

Сравните с диаграммой Вороного для тех же точек:

```
[vx, vy] = voronoi(x, y, TRI);
subplot(1, 2, 2), plot(x, y, 'r+', vx, vy, 'b-'), axis([0 1 0 1]), grid, hold off
```

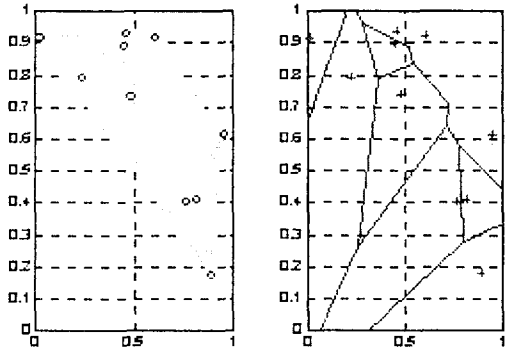


Рис. 8.10

```
k = dsearch(x, y, TRI, 1/2, 1/2)
k =    4
```

Ближайшим к точке (0.5, 0.5) является узел с координатами

```
[x(4)  y(4)]
ans =    0.4860    0.7382

T = tsearch(x, y, TRI, 1/2, 1/2)
T =    NaN
```

Точка (0.5, 0.5) расположена вне пределов триангуляционной сетки

```
T = tsearch(x, y, TRI, 0.6, 0.6)
T =    4
```

Точка (0.6, 0.6) принадлежит треугольнику с вершинами
[x(TRI(4, :))' y(TRI(4, :))']
ans =
0.2311 0.7919
0.4860 0.7382
0.7621 0.4057

Сопутствующие функции: CONVHULL, GRIDDATA, TRIMESH, TRISURF, VORONOI.

CONVHULL **Построение выпуклой оболочки**

Синтаксис:

K = convhull(x, y)
K = convhull(x, y,TRI)

Описание:

Функция K = convhull(x, y) возвращает индексы тех точек, описываемых векторами x и y, которые принадлежат выпуклой оболочке.

Функция K = convhull(x, y, TRI) использует триангуляционную сетку, полученную с помощью функции delaunay, что позволяет избежать ее формирования в процессе вычислений.

Пример:

Построить выпуклую оболочку для функции $|\sqrt{x}|$.

```
xx = -1:.05:1; yy = abs(sqrt(xx));
[x, y] = pol2cart(xx, yy);
k = convhull(x, y);
plot(x(k), y(k), 'r-', x, y, 'b+')
```

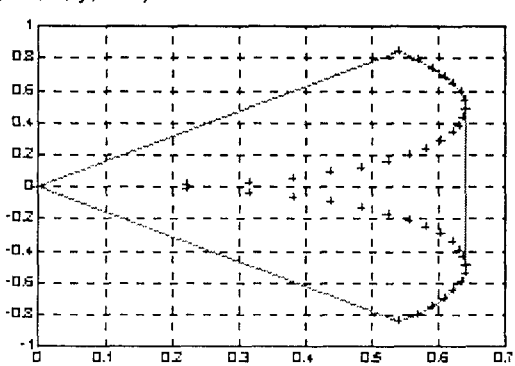


Рис. 8.11

Сопутствующие функции: DELAUNAY, POLYAREA, VORONOI.

VORONOI **Построение диаграммы Вороного**

Синтаксис:

```
voronoi(x, y)
voronoi(x, y, TRI)
h = voronoi(..., 'LineSpec')
[vx, vy] = voronoi(...)
```

Определение:

Пусть задано некоторое множество точек на плоскости; тогда для каждой такой точки можно провести линии, которые отделяют эту точку от ближайших. Эти линии, локализуя данную точку, образуют многоугольник Вороного. Множество таких многоугольников определяет *диаграмму Вороного.*

Описание:

Команда voronoi(x, y) строит диаграмму Вороного для точек, координаты которых заданы векторами x и y.

Команда voronoi(x, y, TRI) использует триангуляционную сетку, полученную с помощью функции delaunay, что позволяет избежать ее формирования в процессе вычислений.

Функция h = voronoi(..., 'LineSpec') строит диаграмму Вороного, используя цвет и тип линий, определяемый параметром 'LineSpec', а также возвращает дескриптор h.

Функция [vx, vy] = voronoi(...) возвращает координаты вершин многоугольников Вороного, так что команда plot(vx, vy, '-', x, y, '.') позволяет построить диаграмму Вороного.

Пример:

Построить диаграмму Вороного для 10 случайных точек.

```
rand('state', 0);
x = rand(1, 10); y = rand(1, 10);
[vx, vy] = voronoi(x, y);
plot(x, y, 'r+', vx, vy, 'b-'); axis equal, grid
```

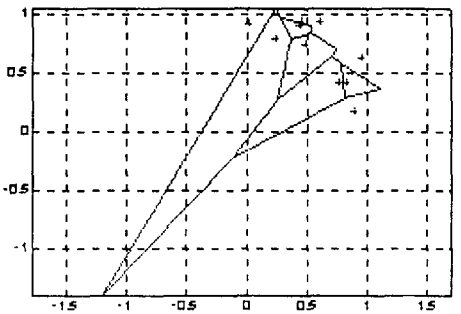


Рис. 8.12

Сопутствующие функции: CONVHULL, DELAUNAY, DSEARCH.

INPOLYGON **Принадлежность точки области, ограниченной многоугольником**

Синтаксис:

IN = inpolygon(X, Y, xv, yv)

Описание:

Функция IN = inpolygon(X, Y, xv, yv) возвращает матрицу IN, размеры которой совпадают с размерами матриц X и Y. Каждый элемент матрицы IN принимает значение 1, 0.5 или 0, в зависимости от того, расположена ли точка внутри, на границе или вне многоугольника, координаты вершин которого заданы векторами xv или yv. Более точно:

IN(p, q) = 1 если точка (X(p, q),Y(p, q)) расположена внутри многоугольника;

IN(p, q) = 0.5 если точка (X(p, q),Y(p, q)) расположена на грани многоугольника;

IN(p,q) = 0 если точка (X(p, q),Y(p, q)) расположена за пределами многоугольника.

Пример:

```
L = linspace(0, 2.*pi, 6); xv = cos(L)'; yv = sin(L)';
xv = [xv; xv(1)]; yv = [yv; yv(1)];
x = randn(250,1); y = randn(250,1);
in = inpolygon(x,y,xv,yv);
plot(xv, yv, x(in), y(in), 'r+', x(~in), y(~in), 'bo')
```

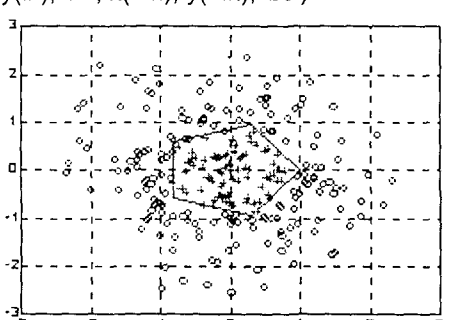


Рис. 8.13

Сопутствующие функции: CONVHULL, DELAUNAY, DSEARCH.

RECTINT **Площади областей пересечения двух семейств прямоугольников**

Синтаксис:

out = rectint(A, B)

Описание:

Функция out = rectint(A, B) возвращает значения площадей для областей пересечения двух семейств прямоугольников. Семейства прямоугольников определены следующим образом. Каждый прямоугольник задается вектором-

строкой из четырех элементов: первые два элемента определяют координаты x и y левого нижнего угла прямоугольника, следующие два - длины сторон вдоль осей x и y. Если семейство A содержит один прямоугольник, то его размер 1×4, а семейство B содержит m прямоугольников, то его размер m×4. В этом случае выходом является вектор размером 1×m. Если массив A имеет размер n×4, а массив B - m×4, то выходом является массив размером n×m.

Пример:

Построим прямоугольники для двух семейств A и B и закрасим области их пересечения.

```
Ax = [ 1   1   7   7   1];
Bx = [ 3   3   4   4   3
       6   6   8   8   6]
plot(Ax, Ay, 'r'), grid, hold on
plot(Bx', By', 'b'), axis([0, 9, 0, 8]), grid
xi = [ 3   3   4   4   3
       6   6   7   7   6]
yi = [ 2   5   5   2   2
       4   5   5   4   4]
fill(xi', yi', 'm')
```

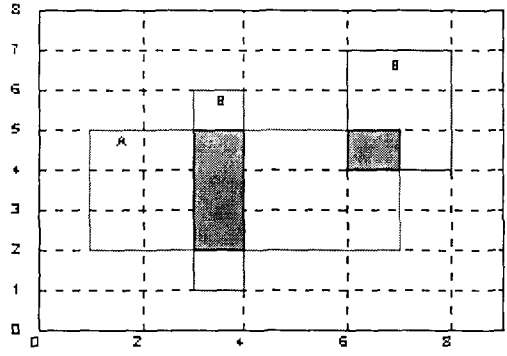


Рис. 8.14

Вычислим площади областей пересечения.

```
A = [1   2   6   3];
B = [3   1   1   4
     6   4   2   3];
rectint(A, B)
ans =  3   1
```

Сопутствующие функции: POLYAREA.

POLYAREA **Площадь многоугольника**

Синтаксис:

S = polyarea(X, Y)
S = polyarea(X, Y, dim)

Описание:

Функция S = polyarea(X, Y) возвращает площадь многоугольника, координаты вершин которого заданы векторами x и y. Если X и Y - массивы одинакового размера, то возвращаются площади многоугольников, заданных соответствующими столбцами этих массивов. Если X и Y - многомерные массивы, то возвращаются площади многоугольников, заданных столбцами первой неединичной размерности этих массивов.

Функция S = polyarea(X, Y, dim) возвращает площади многоугольников, заданных столбцами размерности dim.

Пример:

Построим прямоугольники для двух семейств A и B и закрасим области их пересечения:

```
L = linspace(0, 2*pi, 6); xv = cos(L)'; yv = sin(L)';
xv = [xv ; xv(1)]; yv = [yv ; yv(1)];
A = polyarea(xv, yv);
plot(xv, yv); fill(xv, yv, 'm'); title(['Площадь = ' num2str(A)]); axis image
grid, axes([-1 1 -1 1])
```

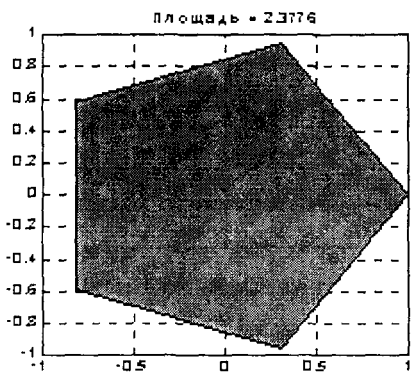


Рис. 8.15

Сопутствующие функции: CONVHULL, INPOLYGON.

Численное интегрирование

TRAPZ, CUMTRAPZ — **Интегрирование методом трапеций**

Синтаксис:

I = trapz(x, Y)	I = cumtrapz(x, Y)
I = trapz(Y)	I = cumtrapz(Y)
I = trapz(..., dim)	I = cumtrapz(..., dim)

Описание:

Функция I = trapz(x, Y) вычисляет определенный интеграл от функции y по переменной x в заданных пределах, используя метод трапеций. Аргументы x и y могут быть одномерными массивами одинакового размера, либо массив Y может быть двумерным, но тогда должно выполняться условие size(Y, 1) = length(x). В последнем случае вычисляется интеграл для каждого столбца. Если массив Y многомерный и его первая неединичная размерность имеет размер length(x), то интеграл вычисляется вдоль этой размерности.

Функция I = trapz(Y) вычисляет интеграл, предполагая, что шаг интегрирования постоянен и равен единице; в случае, когда шаг h отличен от единицы, но постоянен, достаточно вычисленный интеграл умножить на h.

Функции I = trapz(..., dim) вычисляют интеграл вдоль размерности, указанной аргументом dim.

Функции I = cumtrapz(x, Y), I = cumtrapz(Y), I = cumtrapz(..., dim) вычисляют, кроме того, все промежуточные результаты интегрирования.

Примеры:

Вычислим интеграл

$$I = \int_0^{\pi} \sin(x)\,dx$$

Его точное значение равно двум.

Выберем равномерную сетку

x = 0:pi/100:pi; y = sin(x);

тогда оба интеграла

I = trapz(x, y) и I = pi/100*trapz(y)

дают одинаковый результат:

I = 1.9998.

Образуем неравномерную сетку, используя генератор случайных чисел.

x = sort(rand(1,101)*pi); y = sin(x);

I = trapz(x, y)

I = 1.9987.

Результат еще менее точен, поскольку максимальный из шагов max(diff(x)) равен 0.1810.

Сопутствующие функции: SUM, CUMSUM.

QUAD, QUAD8 **Вычисление интегралов методом квадратур**

Синтаксис:

I = quad('<*имя функции*>', a, b)
I = quad('<*имя функции*>', a, b, tol)
I = quad('<*имя функции*>', a, b, tol, trace)
I = quad('fun', a, b, tol, trace, P1, P2, ...)
I = quad8(...)

Описание:

Квадратура - это численный метод вычисления площади под графиком функции, то есть вычисление определенного интеграла вида

$$I = \int_a^b f(x)\,dx$$

Функции quad и quad8 используют разные квадратурные формулы.

Функции I = quad('<*имя функции*>', a, b) и I = quad8('<*имя функции*>', a, b) вычисляют интеграл от заданной функции; последняя может быть как встроенной функцией, так и М-файлом.

Функции I = quad('<*имя функции*>', a, b, tol) и I = quad8('<*имя функции*>', a, b, tol) вычисляют интеграл с заданной относительной погрешностью tol. По умолчанию tol = 1e - 3.

Функции I = quad('<*имя функции*>', a, b, tol, trace) и I = quad8('<*имя функции*>', a, b, tol, trace), когда аргумент trace не равен нулю, строят точечный график подынтегральной функции.

Функции I = quad('<*имя функции*>', a, b, tol, trace, P1, P2, ...) и I = quad8('<*имя функции*>', a, b, tol, trace, P1, P2 ,...) позволяют передать значения параметров P1, P2, ... подынтегральной функции. Для того чтобы воспользоваться значениями аргументов tol и trace по умолчанию, достаточно при обращении указать на их месте пустые массивы, например: quad('fun', a, b, [ ], [ ], P1).

Примеры:

Вычислим интеграл

$$I = \int_0^{\pi} \sin(x)\,dx$$

Его точное значение 2.

```
I = quad('sin', 0, pi, 1e-4, 1)
I =    2.0000
```

Точечный график подынтегральной функции показан на рис. 8.16.

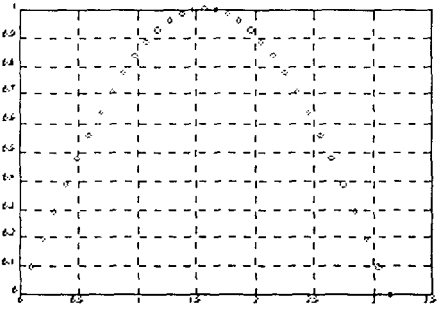


Рис. 8.16

Алгоритм:

Функция quad использует квадратурные формулы Ньютона - Котеса 2-го порядка (правило Симпсона), а функция quad8 - формулы 8-го порядка [1,2]. При наличии в подынтегральной функции особенностей типа

$$I = \int_0^1 \sqrt{x}\, dx$$

предпочтительнее использовать процедуру quad8.

Диагностические сообщения:

Функции quad и quad8 используют рекурсивный вызов. Для того чтобы предотвратить бесконечную рекурсию при вычислении сингулярных интегралов, глубина рекурсии ограничена уровнем 10. При достижении этого ограничения выдается сообщение

Recursion level limit reached in quad. Singularity likely.

В процедуре quad *достигнута предельная глубина рекурсии. Функция, возможно, сингулярна.*

Ограничения:

Функции quad и quad8 не позволяют интегрировать функции с особенностями типа

$$I = \int_0^1 \frac{1}{\sqrt{x}}\, dx$$

В этом случае рекомендуется выделить такие члены и проинтегрировать их аналитически, а к остатку применить процедуры quad и quad8.

Сопутствующие функции: SUM, CUMSUM, TRAPZ, QUADDEMO.

Ссылки:

1. Forsythe G. E., Malcolm M. A., Moler C. B. *Computer Methods for Mathematical Computations.* Prentice-Hall, 1977.

2. *Справочник по специальным функциям:* Пер. с англ. Под ред. М. Абрамовица и И. Стиган. М.: Наука, 1979. С. 683-684.

DBLQUAD **Вычисление двойных интегралов методом квадратур**

Синтаксис:

I = dblquad('<*имя функции*>', inmin, inmax, outmin, outmax)

I = dblquad('<*имя функции*>', inmin, inmax, outmin, outmax, tol)

I = dblquad('<*имя функции*>', inmin, inmax, outmin, outmax, tol, <*метод*>)

Описание:

Функция I = dblquad('<*имя функции*>', inmin, inmax, outmin, outmax) вычисляет двойной интеграл от функции <*имя функции*>(inner, outer), используя квадратурные формулы. Переменные inner и outer - это переменные внутреннего и внешнего интегралов с пределами интегрирования соответственно inmin, inmax и outmin, outmax. Сама подынтегральная функция fout = <*имя функции*>(inner, outer) является функцией двух переменных и должна иметь векторный аргумент inner и скалярный аргумент outer; выход fout является вектором.

Функция I = dblquad('<*имя функции*>', inmin, inmax, outmin, outmax, tol) позволяет передать параметр tol функции quad или quad8.

Функция I = dblquad('<*имя функции*>', inmin, inmax, outmin, outmax, tol, <*метод*>) позволяет задать метод интегрирования; допустимые имена для параметра <*метод*> - это quad, quad8 или имя функции, написанной пользователем, при условии, что эта функция имеет такой же вызов и формирует такой же выход, как функции quad и quad8.

Примеры:

Вычислим интеграл

$$I = \int_0^{\pi} dy \int_{\pi}^{2\pi} (y\sin(x) + x\cos(y))\,dx .$$

```
I= dblquad('integrnd', pi, 2*pi, 0, pi )
I =  -9.8698
I= dblquad('integrnd', pi, 2*pi, 0, pi, 1e-4 )
I =  -9.8696
```

Сопутствующие функции: QUAD, QUAD8, INLINE.

Интегрирование обыкновенных дифференциальных уравнений

Решение систем обыкновенных дифференциальных уравнений (ОДУ) с заданными начальными условиями (задача Коши) играет важную роль в практике инженерных вычислений. В систему MATLAB, начиная с версии 5.0, включен специальный решатель ОДУ, который упрощает пользователю выбор метода решения, задание начальных условий и установление специальных опций для повышения эффективности используемых методов.

В таблице приведены основные операции, применяемые при решении систем ОДУ, функции, их поддерживающие, а также описано назначение этих функций.

Операция	*Функция*	*Назначение*
Решение ОДУ	ode113	Гладкие системы, порядок метода - переменный
	ode15s	Жесткие системы, порядок метода - переменный
	ode23	Гладкие системы, порядок метода - низкий
	ode23s	Жесткие системы, порядок метода - низкий
	ode23t	Умеренно жесткие системы
	ode23tb	Жесткие системы, порядок метода - низкий
	ode45	Гладкие системы, порядок метода - средний
Формирование задачи Коши	odefile	Задать правые части системы ОДУ
	odeget	Получить опции решателя
	odeset	Задать/изменить опции решателя
Формирование выхода решателя	odeplot	Переходные процессы (функции времени)
	odephas2	Двумерная фазовая плоскость
	odephas3	Трехмерная фазовая плоскость
	odeprint	Графическое окно вывода
	odezero	Фиксация нулевого значения

SOLVER **Решатель ОДУ**

Синтаксис:

```
[t, X] = solver('F', tspan, x0)
[t, X] = solver('F', tspan, x0, <опции>)
[t, X] = solver('F', tspan, x0, <опции>, p1, p2, ...)
[t, X, te, ye, ie] = solver('F', tspan, x0, <опции>)
[t, X, S] = solver( ...)
```

Описание:

Решатель ОДУ использует различные численные методы, которые должны быть указаны в качестве имени, то есть заменять имя solver. Все функции позволяют решать уравнения, заданные в явной форме Коши x' = F(t, x); методы ode15s, ode23s, ode23t, ode23tb решают также и уравнения в неявной форме вида Mx' = F(t, x); кроме того, эти методы, за исключением ode23s могут быть использованы и для решений ОДУ вида M(t)x' = F(t, x).

Функция [t, X] = solver('F', tspan, x0), где tspan = [t0 tf], интегрирует систему ОДУ на интервале времени [t0 tf] с начальными условиями x0. Функция вычисления правых частей определяется именем F. Каждая строка выходного массива X соответствует решению в моменты времени, определяемые

вектором t. Для того чтобы получить решения в фиксированные моменты времени t0, t1, ..., tfinal, которые должны быть указаны в строго возрастающем или строго убывающем порядке, необходимо задать соответствующий вектор tspan = [t0 t1 ... tfinal].

Функция [t, X] = solver('F', tspan, x0, <*опции*>) позволяет указать дополнительные опции, формируемые с помощью функции odeset. В числе часто используемых опций следует отметить относительную погрешность RelTol (по умолчанию 1e-3) и вектор абсолютных погрешностей AbsTol (по умолчанию все компоненты равны 1e-6).

Функция [t, X] = solver('F', tspan, x0, <*опции*>, p1, p2, ...) позволяет передать значения параметров p1, p2, ... при вычислении функции правых частей; если при этом опции используются по умолчанию, то на их месте следует ввести пустой массив [ ].

Функция [t, X, te, xe, ie] = solver('F', tspan, x0, <*опции*>) при условии, что опция Events установлена в состояние 'on', позволяет зафиксировать момент, когда какая-либо из указанных переменных принимает значение 0. При этом функция вычисления правых частей должна иметь вид F(t, x, 'events'). Более подробно это обсуждается при описании функции odefile. Выход te - это вектор-столбец моментов времени, когда фиксировалось событие, вектор xe - перечень соответствующих переменных, а вектор ie - индексы событий, указывающие, в каком направлении изменялось решение в момент пересечения нуля: (-1) - убывало, (+1) - возрастало, (0) - только фиксировалось.

Функции вида [t, X, s] = solver(...) с тремя выходными аргументами позволяют получить статистические характеристики решателя; при этом вектор-столбец s имеет 6 компонентов следующего назначения:

- количество успешных шагов;
- количество неудачных попыток;
- количество вызовов функции $\mathbf{F}(t, \mathbf{x})$;
- количество вызовов функции Якоби $d\mathbf{F}/d\mathbf{x}$;
- количество LU-разложений;
- количество решений вспомогательных систем линейных уравнений.

Последние 3 параметра этого списка применяются только к решателям жестких систем. Решатель также выводит эти параметры, если в аргументе options установлено свойство Stats.

При обращении к решателю ОДУ без указания выходных параметров solver(...) он вызывает функцию вывода odeplot для построения графиков решения. Другой способ организации вывода графиков - установить для опции OutputFcn значение odeplot. Значения этой опции odephas2 и odephas3 позволяют вывести двумерные и трехмерные фазовые портреты.

При интегрировании жестких систем с помощью функций ode15s и ode23s необходимо надежно и эффективно вычислять якобиан системы уравнений, для чего предназначены специальные опции. В частности, если якобиан d**F**/d**x** постоянен, опция JConstant должна быть установлена в состояние 'on'; если функция вычисления правых частей F(t, [x1 x2 ...]) сформирована так, что ее выходом являются функции [F(t, x1) F(t, x2) ...], то опция Vectorized должна быть установлена в состояние 'on'; если матрица системы d**F**/d**x** является разреженной и требуется ее маска, состоящая из 0 и 1, фиксирующих положение ненулевых элементов, то опция JPattern должна быть установлена в состояние 'on', а функция вычисления правых частей должна вызываться в форме F([ ], [ ], 'jpattern'); если необходимо вычисление якобиана, то должна быть установлена опция Jacobian.

Функции ode15s, ode23s, ode23t, ode23tb применимы для интегрирования ОДУ вида Mx' = F(t, x) с постоянной матрицей M, которая обычно представлена в разреженной форме и является невырожденной. Для вывода на экран матрицы M необходимо установить для опции MassConstant значение 'on' и задать вызов функции вычисления правых частей в форме F([ ], [ ], 'mass'). Из перечисленных функций все, кроме функции ode23s, могут применяться для решения ОДУ вида M(t)x' = F(t, x), где матрица M(t) является нестационарной, невырожденной для любого t и обычно разреженной. Для вывода значений матрицы M(t) необходимо установить для опции Mass значение 'on' и задать вызов функции вычисления правых частей в форме F([ ], [ ], 'mass').

Следующая таблица может помочь в выборе функции, используемой для интегрирования.

Функция	*Тип системы*	*Порядок точности*	*Назначение*
ode45	Гладкая	Средний	Наиболее употребимый
ode23	"	Низкий	При низких требованиях к точности или для решения умеренно жестких задач
ode113	"	Переменный: от низкого до высокого	При высоких требованиях к точности и для решения задач со сложными правыми частями
ode15s	Жесткая	Переменный: от низкого до среднего	Если алгоритм ode45 сходится медленно и для решения систем ОДУ в неявной форме Коши
ode23s	"	Низкий	При низких требованиях к точности и для решения систем ОДУ в неявной форме Коши с постоянной матрицей при производных
ode23t	"	Умеренный	Для решения умеренно жестких задач и если необходимо решение для осциллятора без демпфирования
ode23tb	"	Низкий	При низких требованиях к точности и для решения систем ОДУ в неявной форме Коши с нестационарной матрицей при производных

Алгоритмы, применяемые в решателе ОДУ, существенно зависят от требований к точности решений и типа интегрируемой системы (гладкая, жесткая) и достаточно полно описаны в работе [5].

Пользователь может указать интервал интегрирования, начальные условия и опции решателя, не прибегая к заданию входных аргументов, а используя функцию описания правых частей odefile. Если аргументы - пустые матрицы, то решатель вызывает функцию odefile в форме [tspan, y0, options] = F([ ], [ ], 'init') с целью вычислить значения аргументов, отсутствующих во входном списке. Пустые аргументы в конце списка ввода могут быть опущены. Это означает, что допустимы следующие формы вызова решателя:

```
[t, X] = solver('F')
[t, X] = solver('F', [ ], x0)
[t, X] = solver('F', tspan, [ ], options)
[t, X] = solver('F', [ ], [ ], options)
```

Опции интегрирования могут быть заданы как с помощью функции, так и из командной строки. Если использовались оба способа задания опций, то приоритет имеют опции, заданные из командной строки.

Алгоритм:

Функция ode45 реализует явные формулы Рунге - Кутты 4-го и 5-го порядков; это одношаговый решатель, который для вычисления вектора состояния $\mathbf{x}(t_n)$ в момент времени t_n использует только значение состояния $\mathbf{x}(t_{n-1})$ в предыдущий момент времени t_{n-1}. Функцию ode45 рекомендуется использовать всегда в качестве пробной для интегрирования новой системы ОДУ [1].

Функция ode23 реализует явные формулы Рунге - Кутты 2-го и 3-го порядков; этот одношаговый решатель может быть полезен при низких требованиях к точности решения и при умеренной жесткости системы [2].

Функция ode113 - это решатель переменного порядка, основанный на формуле Адамса - Башворта - Мултона; он может оказаться эффективнее функции ode45 при высоких требованиях к точности и в тех случаях, когда высока трудоемкость вычисления правых частей; это многошаговый решатель, который требует знания решений в нескольких предшествующих точках для вычисления решения в текущей точке [3].

Вышеперечисленные алгоритмы предназначены для интегрирования гладких систем. Если оказывается, что они работают очень медленно, воспользуйтесь одним из указанных ниже решателей для жестких систем.

Функция ode15s - это решатель переменного порядка, основанный на формулах численного дифференцирования. По умолчанию используются формулы прямого дифференцирования, но в качестве опции могут быть использованы формулы обратного дифференцирования (метод Гира), которые, как правило, снижают скорость интегрирования. Формулы прямого и обратного дифференцирования порядка 1 и 2 А-устойчивы, то есть устойчивы

во всей левой полуплоскости; формулы более высокого порядка имеют меньшие области устойчивости. В функции ode15s реализованы формулы от 1-го до 5-го порядка (по умолчанию принят порядок 5). При интегрировании осцилляторов рекомендуется понижать порядок используемых формул численного дифференцирования до 2, чтобы расширить область устойчивости. Подобно функции ode113 функция ode15s использует многошаговый метод. Если предполагается, что система ОДУ жесткая или применение функции ode45 оказалось неэффективным, рекомендуется обращаться к функции ode15s [7].

Функция ode23s использует модифицированную формулу Розенброка 2-го порядка; поскольку это одношаговый решатель, то он может оказаться эффективнее функции ode15s при низких требованиях к точности [7].

Функция ode23t использует метод трапеций с интерполяцией. Эта функция рекомендуется в тех случаях, если задача умеренно жесткая и численное интегрирование не должно вносить искусственного демпфирования.

Функция ode23tb реализует неявный метод Рунге - Кутты, в первой фазе которого используется метод трапеций, а во второй - формулы обратного дифференцирования 2-го порядка. Подобно функции ode23s эта функция эффективнее, чем ode15s, при низких требованиях к точности [8, 9].

Для эффективного моделирования сложных динамических систем с непрерывным и дискретным временем рекомендуется использовать специализированную систему SIMULINK [10], входящую в состав программных продуктов, выпускаемых фирмой The MathWorks, Inc.

Сопутствующие функции: ODEDEMO, Simulink.

Ссылки:

1. Dormand J. R., Prince P. J. A family of embedded Runge-Kutta formulae// *J. Comp. Appl. Math.* 1980. Vol. 6. P. 19–26.
2. Bogacki P., Shampine L. F. A 3(2) pair of Runge-Kutta formulas// *Appl. Math. Letters.* 1989. Vol. 2. P. 1–9.
3. Shampine L. F., Gordon M. K. *Computer Solution of Ordinary Differential Equations: the Initial Value Problem,* San Francisco: W. H. Freeman, 1975.
4. Forsythe G. E., Malcolm M. A., Moler C. B. *Computer Methods for Mathematical Computations.* Prentice-Hall, 1977.
5. Shampine L. F. Numerical Solution of Ordinary Differential Equations. New York: Chapman & Hall, 1994.
6. Kahaner D., Moler C., Nash S. Numerical Methods and Software. New Jersey: Prentice-Hall, 1989.
7. Shampine L. F., Reichelt M. W. The MATLAB ODE Suite// SIAM Journal on Scientific Computing. 1997. Vol. 18-1.
8. Shampine L. F., Hosea M. E. Analysis and Implementation of TR-BDF2// Applied Numerical Mathematics. 1996. Vol. 20.
9. Bank R. E., Coughran W. C., Fichtner W., Grosse E., Rose D., Smith R. Transient Simulation of Silicon Devices and Circuits// IEEE Trans. CAD. Vol. 4. 1985. P. 436-451.
10. SIMULINK. *User's Guide.* Natick: The MathWorks, Inc., 1998.

odefile — **Задание правых частей системы ОДУ**

Синтаксис:

varargout = odefile(t, x, '<*флаг*>', p1, p2, ...)

Описание:

M-файл odefile - это шаблон для написания M-файлов, описывающих системы дифференциальных уравнений. Он содержит необходимые заготовки для описания систем ОДУ, заданных как в явной, так и неявной форме Коши:

x' = F(t, x) - явная форма Коши;
Mx' = F(t, x) - неявная форма Коши;
M(t)x' = F(t, x) - неявная форма Коши,

где t - независимая переменная, обычно время;
x - вектор состояния;
F- вектор-функция правых частей системы ОДУ;
M, M(t) - невырожденные матрицы.

Для решения уравнений вида M(t)x' = F(t, x) предназначены только функции ode15s, ode23t, ode23tb, ориентированные на жесткие системы ОДУ; для решения уравнений вида Mx' = F(t, x) могут быть использованы функции ode15s, ode23s, ode23t, ode23tb.

Кроме задания правых частей в файле odefile можно задать начальные условия и интервал интегрирования.

Для того чтобы использовать шаблон, необходимо выполнить следующие операции:

- ввести команду help odefile;
- скопировать текст файла odefile в отдельный файл;
- отредактировать скопированный файл, удалив из него ненужные фрагменты;
- вставить необходимую информацию, где это указано в файле.

По умолчанию в решателях предполагается, что системы ОДУ заданы в явной форме Коши dx/dt = F(t, x) и они на каждом шаге выполняют вызов функции F = odefile(t, X). Следует помнить, что аргументы t и X обязательно должны быть переданы функции odefile, даже если они в ней не используются. Таким образом, простейший шаблон функции odefile выглядит следующим образом:

```
function F = odefile(t, x)
  F = < Ввести функцию от t и/или x. >;
```

В более общем случае odefile может использовать большее количество входных аргументов, а именно odefile(t, x, '<*флаг*>', p1, p2, ...), где t и x - переменные интегрирования, <*флаг*> - строковая переменная, которая вид возвращаемой информации, а p1, p2, ... - дополнительные параметры.

Допустимые значения флагов и соответствующая им информация представлены в следующей таблице.

Флаг	*Возвращаемая информация*
'' (пустой)	F(t, x)
'init'	Интервал интегрирования tspan, начальные условия x0, опции
'jacobian'	Матрица Якоби J(t, x) = dF(t, x)/dx
'jpattern'	Маска разреженной матрицы Якоби, состоящая из 0 и 1, фиксирующих положение ненулевых элементов
'mass'	Матрица M(t) при производных
'events'	Фиксация нулевого значения

Ниже приведен шаблон, который иллюстрирует, как можно записать функцию odefile с двумя дополнительными параметрами p1 и p2. Этот файл включает также подфункции и все возможные варианты значения флага.

```
function varargout = odefile(t, x, flag, p1, p2)
switch flag
case ''                              % Правые части  dx/dt = f(t, x).
  varargout{1} = f(t, x, p1, p2);
case 'init'                          % Выход по умолчанию  [tspan, x0, options].
  [varargout{1:3}] = init(p1, p2);
case 'jacobian'                       % Матрица Якоби df/dx.
  varargout{1} = jacobian(t, x, p1, p2);
case 'jpattern'                       % Маска разреженной матрицы S.
  varargout{1} = jpattern(t, x, p1, p2);
case 'mass'                           % Матрицы при производных M(t) или M.
  varargout{1} = mass(t, x, p1, p2);
case 'events'                         % Выход  [value, isterminal, direction].
  [varargout{1:3}] = events(t, x, p1, p2);
otherwise
  error(['Unknown flag ''' flag '''.']);
end

------------------------------------------------------------------------
function dxdt = f(t, x, p1, p2)
   dxdt = <Введите функцию  переменных t и/или x, а также параметры p1, p2. >
------------------------------------------------------------------------
function [tspan, x0, options] = init(p1, p2)
   tspan = < Введите интервал интегрирования. >;
   x0 = < Введите начальное условие x0. >;
   options = < Введите options = odeset(...) или [ ]. >;
------------------------------------------------------------------------
function dfdx = jacobian(t, x, p1, p2)
   dfdx = < Введите матрицу Якоби. >;
```

```
---------------------------------------------------------------------
function S = jpattern(t, x, p1, p2)
  S = < Введите  маску матрицы Якоби. >;
---------------------------------------------------------------------
function M = mass(t, x, p1, p2)
  M = < Введите матрицу при производных. >;
---------------------------------------------------------------------
function [value, isterminal, direction] = events(t, x, p1, p2)
 value = < Введите вектор переменных, для которых фиксируется пересечение нуля >
 isterminal = <Укажите одну или несколько переменных, для которых дос-
   тижение нуля завершает интегрирование.>;
 direction = < Укажите направление изменения функции в момент фиксации
   события: -1 - в сторону уменьшения, 1 - в сторону возрастания, 0 - в лю-
   бом направлении.>;
```

Сопутствующие функции: B5ODE, BALLODE, BRUSSODE, FEM1ODE, ORBITODE, RIGIDODE, VDPODE.

odeset — Задание опций для решателя ОДУ

Синтаксис:

```
options = odeset('имя_1', значение_1, 'имя_2', значение_2, ...)
options = odeset(oldopts, 'имя_1', значение_1, ...)
options = odeset(oldopts, newopts)
odeset
```

Описание:

Функция options = odeset('*имя_1*', *значение_1*, '*имя_2*', *значение_2*, ...) формирует массив записей options, в котором каждому свойству приписывается некоторое значение; неприсвоенному значению приписывается пустой массив [ ]. Достаточно указать только первый символ, используя любой регистр клавиатуры.

Функция options = odeset(oldopts, 'имя_1', значение_1, ...) заменяет значения отдельных опций.

Функция options = odeset(oldopts, newopts) заменяет набор старых опций набором новых опций, но если при этом для какой-либо из новых опций указано [ ], то сохраняется прежнее значение. Пусть, например,

oldopts

F	1	[]	4	's'	's'	[]	[]	[]	...

newopts

T	3	F	[]	''	[]	[]	[]	[]	...

тогда

odeset(oldopts,newopts)

T	3	F	4	''	's'	[]	[]	[]	...

Функция odeset сама по себе выводит список всех опций с указанием их возможных значений и значений по умолчанию в фигурных скобках:

```
odeset
AbsTol: [ positive scalar or vector {1e-6}]
BDF: [ on | {off} ]
Events: [ on | {off} ]
InitialStep: [ positive scalar ]
Jacobian: [ on | {off} ]
JConstant: [ on | {off} ]
JPattern: [ on | {off} ]
Mass: [ on | {off} ]
MassConstant: [ on | {off}]
MaxOrder: [ 1 | 2 | 3 | 4 | {5} ]
MaxStep: [ positive scalar ]
OutputFcn: [ string ]
OutputSel: [ vector of integers ]
Refine: [ positive integer ]
RelTol: [ positive scalar {1e-3} ]
Stats: [ on | {off} ]
Vectorized: [ on | {off} ]
```

Те или иные опции могут оказаться допустимыми только для отдельных методов и функций, поэтому они в первую очередь разбиваются на следующие категории:

- Границы погрешностей интегрирования.
- Выходные параметры решателя.
- Характеристики якобиана.
- Фиксация события.
- Характеристики матрицы при производных.
- Шаг интегрирования.
- Параметры функции ode15s.

Опишем каждую из категорий в виде отдельной таблицы.

Границы погрешностей интегрирования

Свойство	*Значение*	*Описание*
RelTol	Скаляр - {1e-3}	Допустимая относительная погрешность для всех компонентов вектора решения
AbsTol	Скаляр или вектор {1e-6}	Абсолютная погрешность; если скаляр, то применяется для всех компонентов вектора решения; если вектор, то применяется к соответствующим компонентам

Выходные параметры решателя

Свойство	*Значение*	*Описание*
OutputFcn	Строка	Имя выбираемой функции формирования выхода: odeplot, odephas2, odephas3, odeprint
OutputSel	Вектор индексов	Указывает, какие компоненты вектора состояния формируют выход функции
Refine	Целое	Производит более гладкий вывод за счет увеличения количества выводимых точек, определяемого параметром Refine. Для большинства функций по умолчанию этот параметр равен 1. Однако для метода ode45 Refine равен 4. Для того чтобы выполнить вывод только в моменты отсчета, соответствующие шагу интегрирования, следует задать параметр Refine равным 1
Stats	on | {off}	Определяет, выводятся или нет статистические характеристики решателя

Характеристики якобиана (для функций ode15s и ode23)

Свойство	*Значение*	*Описание*
JConstant	on | {off}	Указывает, что якобиан dF/dx постоянен
Jacobian	on | {off}	Сообщает решателю, что odefile запрашивает аргументы (t, x, 'jacobian') и возвращает dF/dx
JPattern	on | {off}	Сообщает решателю, что odefile запрашивает аргументы ([ ], [ ], 'jpattern') и возвращает маску якобиана dF/dx (см. функцию brussode)
Vectorized	on | {off}	Сообщает решателю, что odefile будет формировать выход F(t, x) в векторизованной форме [F(t, x1) F(t, x2) ...]. Использование такой формы для решателей жестких систем возможно только в том случае, когда матрица Якоби формируется численно (режим по умолчанию) и если с помощью функции odeset свойство Vectorized было установлено в состояние 'on'

Фиксация события

Свойство	*Значение*	*Описание*
Events	on | {off}	Сообщает решателю, что odefile запрашивает аргументы (t, x, 'events'), и возвращает параметры, связанные с событием "решение пересекло нулевое значение" (см. функцию odefile)

Характеристики матрицы при производных (для функций ode15s и ode23)

Свойство	*Значение*	*Описание*
Mass	on | {off}	Сообщает решателю, что odefile сформирован так, что функция F(t, [ ], 'mass') возвращает матрицу M или M(t)
MassConstant	on | {off}	Сообщает решателю, что odefile сформирован так, что функция F(t, [ ], 'mass') возвращает постоянную матрицу M

Шаг интегрирования

Свойство	*Значение*	*Описание*
MaxStep	Положительное число	Максимально допустимое значение шага интегрирования
InitialStep	" "	Начальное значение шага; решатель использует этот шаг и, если ошибка значительна, уменьшает его

Параметры функции ode15s

Свойство	*Значение*	*Описание*
MaxOrder	1 | 2 | 3 | 4 | {5}	Максимальный порядок формулы интегрирования
BDF	on | {off}	Определяет, должны ли использоваться формулы обратного дифференцирования вместо используемых по умолчанию формул прямого дифференцирования

Сопутствующие функции: ODEGET.

odeget **Извлечь значение опции**

Синтаксис:

```
o = odeget(options, 'имя')
o = odeget(options, 'имя', default)
```

Описание:

Функция o = odeget(options, *'имя'*) извлекает значение опции *'имя'* из массива записей options; если значение не задано, возвращается пустой массив. Достаточно указать только первый символ имени опции, используя любой регистр клавиатуры.

Функция o = odeget(options, *'имя'*, default) возвращает значение опции по умолчанию, если текущее значение не специфицировано.

Сопутствующие функции: ODESET.

odeplot **Построение графиков решений для системы ОДУ**

Синтаксис:

```
odeplot(tspan, x0, 'init'),
status = odeplot(t, X)
odeplot([ ], [ ], 'done')
```

Описание:

Функция status = odeplot(t, X) вызывается решателем на каждом шаге интегрирования и строит графики по всем переменным состояния. Для того чтобы построить графики по отдельным переменным, необходимо указать их индексы в опции OutputSel. Применение функции odeplot должно быть указано в опции OutputFcn. Функция odeplot используется по умолчанию в том случае, когда решатель вызывается без указания выходных аргументов.

В начале интегрирования решатель вызывает функцию odeplot(tspan, x0, 'init'), а затем на каждом шаге функцию status = odeplot(t, x), где t - текущее время, а x - текущий вектор состояния.

Если опция решателя Refine больше 1, то t - вектор-столбец используемых моментов времени, а X - соответствующий массив векторов состояния. Выходной параметр status принимает значение 1, если нажата кнопка STOP прекращения процесса интегрирования, и 0 - в остальных случаях. При завершении процедуры интегрирования вызывается команда odeplot([ ], [ ], 'done').

Сопутствующие функции: ODEPHAS2, ODEPHAS3, ODEPRINT.

odephas2

Построение двумерного фазового портрета

Синтаксис:

```
odephas2(tspan, x0, 'init'),
status = odephas2(t, X)
odephas2([ ], [ ], 'done')
```

Описание:

Функция status = odephas2(t, X) вызывается решателем на каждом шаге интегрирования и строит фазовый портрет, используя две первые переменные состояния. Для того чтобы построить фазовый портрет для двух произвольно выбранных переменных, необходимо указать их индексы в опции OutputSel. Применение функции odephas2 должно быть указано в опции OutputFcn.

В начале интегрирования решатель вызывает функцию odephas2(tspan, x0, 'init'), а затем на каждом шаге функцию status = odephas2(t, x), где t - текущее время, а x - текущий вектор состояния.

Если опция решателя Refine больше 1, то t - вектор-столбец используемых моментов времени, а X - соответствующий массив векторов состояния. Выходной параметр status принимает значение 1, если нажата кнопка STOP прекращения процесса интегрирования, и 0 - в остальных случаях. При завершении процедуры интегрирования вызывается команда odephas2([ ], [ ], 'done').

Сопутствующие функции: ODEPHAS3, ODEPLOT, ODEPRINT.

odephas3

Построение трехмерного фазового портрета

Синтаксис:

```
odephas3(tspan, x0, 'init'),
status = odephas3(t, X)
odephas3([ ], [ ], 'done')
```

Описание:

Функция status = odephas3(t, X) вызывается решателем на каждом шаге интегрирования и строит фазовый портрет, используя 3 первые переменные состояния. Для того чтобы построить фазовый портрет для трех произвольно

выбранных переменных, необходимо указать их индексы в опции OutputSel. Применение функции odephas3 должно быть указано в опции OutputFcn.

В начале интегрирования решатель вызывает функцию odephas3(tspan, x0, 'init'), а затем на каждом шаге функцию status = odephas3(t, x), где t - текущее время, а x - текущий вектор состояния.

Если опция решателя Refine больше 1, то t - вектор-столбец используемых моментов времени, а X - соответствующий массив векторов состояния. Выходной параметр status принимает значение 1, если нажата кнопка STOP прекращения процесса интегрирования, и 0 - в остальных случаях. При завершении процедуры интегрирования вызывается команда odephas3([ ], [ ], 'done').

Сопутствующие функции: ODEPHAS2, ODEPLOT, ODEPRINT.

odeprint **Печать решений для системы ОДУ**

Синтаксис:

```
odeprint(tspan, x0, 'init'),
status = odeprint(t, X)
odeplrint([ ], [ ], 'done')
```

Описание:

Функция status = odeprint(t, X) вызывается решателем на каждом шаге интегрирования и выводит на печать решения по всем переменным состояния. Для того чтобы вывести отдельные переменные, необходимо указать их индексы в опции OutputSel. Применение функции odeprint должно быть указано в опции OutputFcn.

В начале интегрирования решатель вызывает функцию odeprint(tspan, x0, 'init'), а затем на каждом шаге функцию status = odeprint(t, x), где t - текущее время, а x - текущий вектор состояния.

Если опция решателя Refine больше 1, то t - вектор-столбец используемых моментов времени, а X - соответствующий массив векторов состояния. Выходной параметр status принимает значение 1, если нажата кнопка STOP прекращения процесса интегрирования, и 0 - в остальных случаях. При завершении процедуры интегрирования вызывается команда odeprint([ ], [ ], 'done').

Сопутствующие функции: ODEPHAS2, ODEPHAS3, ODEPLOT.

Примеры применения решателя

Пример 1. Уравнения Эйлера движения твердого тела (гладкая система):

Рассмотрим систему ОДУ в форме уравнений Эйлера, описывающих свободное движение твердого тела:

$$\dot{x}_1 = x_2 x_3, \qquad x_1(0) = 0;$$
$$\dot{x}_2 = -x_1 x_3, \qquad x_2(0) = 0;$$
$$\dot{x}_3 = -0.51 x_1 x_2, \quad x_3(0) = 0.$$

Аналитические решения этих уравнений известны и описываются эллиптическими функциями Якоби [1].

Программа формирования правых частей оформлена в виде М-файла rigidode. Выполним интегрирование этой системы методом Рунге - Кутты на интервале времени [0 12] сек с начальными условиями [0 1 1] и построим трехмерный фазовый портрет решений этой системы. Для этого следует задать опцию вывода фазового портрета

options = odeset('OutputFcn', 'odephas3').

Точностные характеристики и значения шага интегрирования используются по умолчанию.

[t, X, S] = ode45('rigidode', [ ], [ ], options);

S =	
19	Количество успешных шагов
2	Количество неудачных попыток выбора шага
127	Количество вызовов функции rigidode
0	Количество вызовов функции dF/dx
0	Количество LU-разложений
0	Количество решений вспомогательных систем линейных уравнений

Трехмерный фазовый портрет для уравнений Эйлера, описывающих свободное движение твердого тела, показан на рис. 8.17.

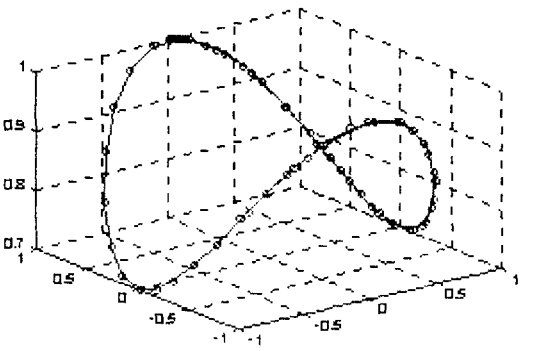


Рис. 8.17

Пример 2. Уравнение Ван дер Поля (жесткая система):

Рассмотрим дифференциальное уравнение 2-го порядка, известное как уравнение Ван дер Поля [2]:

$$\ddot{x} + \mu(x^2 - 1)\dot{x} + x = 0 .$$

Это уравнение может быть представлено в виде системы ОДУ в явной форме Коши:

$$\dot{x}_1 = x_2;$$

$$\dot{x}_2 = \mu(1 - x_1^2)x_2 - x_1.$$

Программа формирования правых частей оформлена в виде М-файла vdpode. Этот файл позволяет использовать как аналитическую (опция Jacobian), так и численную (опция Vectorized) форму представления матрицы Якоби. По умолчанию значение μ равно 1, и в этом случае решения и сама система являются гладкими. При значении μ, равном 1000, возникают релаксационные колебания, когда имеются участки медленного и очень быстрого развития процессов; система в этом случае характеризуется как жесткая. Выполним интегрирование этой системы, используя функцию ode15s методом Рунге - Кутты на интервале времени [0 3000] сек с начальными условиями [2 0].

Построим переходные процессы (рис. 8.18, *а*), используя опцию Jacobian и значение параметра μ, равное 1000:

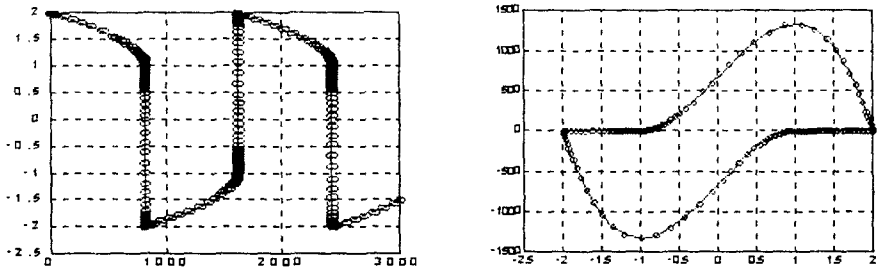
```
options = odeset('Jacobian', 'on');
[t, X, S] = ode15s('vdpode', [ ], [ ], options, 1000);
```

S =	
591	Количество успешных шагов
225	Количество неудачных попыток выбора шага
1749	Количество вызовов функции vdpode
45	Количество вызовов функции dF/dx
289	Количество LU-разложений
1747	Количество решений вспомогательных систем линейных уравнений

Построим двумерный фазовый портрет решений этой системы (рис. 8.18, *б*). Зададим следующие опции и параметры интегрирования:

```
options = odeset('OutputFcn', 'odephas2', 'Vectorized', 'on');
[t, X, S] = ode15s('vdpode', [ ], [ ], options, 1000);
```

S =	
455	Количество успешных шагов
201	Количество неудачных попыток выбора шага
1569	Количество вызовов функции vdpode
43	Количество вызовов функции dF/dx
249	Количество LU-разложений
1439	Количество решений вспомогательных систем линейных уравнений



а б

Рис. 8.18. Уравнение Ван-дер-Поля: а - переходный процесс; б - фазовый портрет

Пример 3. Брюсселятор (жесткая система):

Рассмотрим дифференциальные уравнения, описывающие процессы диффузии в химической реакции [4],

$$u_i' = 1 + u_i^2 v_i - 4u_i + \alpha(N+1)^2(u_{i-1} - 2u_i + u_{i+1});$$

$$v_i' = 3u_i - u_i^2 v_i + \alpha(N+1)^2(v_{i-1} - 2v_i + v_{i+1})$$

на сетке из n узлов и на интервале времени [0 10] с параметром α, равным 0.02, и начальными условиями

$$u_i(0) = 1 + \sin(2\pi x_i);$$

$$v_i(0) = 3,$$

где $x_i = i/(n+1)$, $i = 1, \ldots, n$.

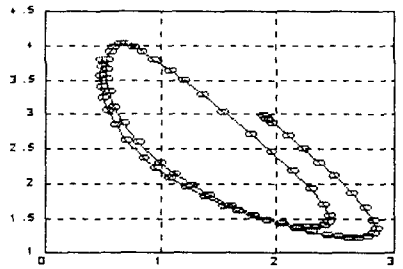
Полное количество уравнений этой системы 2n, но матрица Якоби имеет максимальную ширину, равную 5, если переменные упорядочены как u_1, v_1, u_2, v_2, ... Количество точек сетки n по умолчанию равно 2, но может быть увеличено, если передать этот параметр решателю в виде

```
[t, X] = ode15s('brussode', [ ], [ ], [ ], n);
```

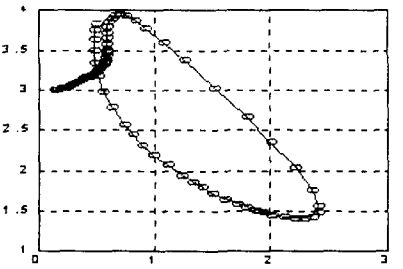
Чем больше значение n, тем более жесткой будет система.

Построим фазовые портреты для двух ячеек, полагая n = 2.

```
options = odeset('OutputFcn', 'odephas2',  'OutputSel', [ 1 2]);
[t, X] = ode15s('brussode', [ ], [ ], options);
options = odeset('OutputFcn', 'odephas2',  'OutputSel', [ 3 4]);
[t, X] = ode15s('brussode', [ ], [ ], options);
```



а



б

Рис. 8.19

Пример 4. Метод конечных элементов (неявная форма Коши):

Рассмотрим одномерное нестационарное уравнение в частных производных вида

$$e^{-t}\frac{\partial u}{\partial t}=\frac{\partial^2 u}{\partial x^2}$$

с начальными условиями

u(0, x) = sin(x)

и граничными условиями

u(t, 0) = u(t, π) = 0.

Интервал интегрирования по пространственной переменной x разбивается на n участков с шагом h, равным $1/(n+1)$. Тогда решение уравнения может быть аппроксимировано на основе метода конечных элементов следующим образом:

$$u(t,\ x_k)\approx\sum_{k=1}^{n}c_k(t)\phi_k(x)\ ,$$

где $x_k = k\pi h$, ϕ_k - кусочно-линейная функция, равная 1 в точке x_k и 0 в остальных точках x_j.

Такая дискретизация, известная под названием метода Галеркина, приводит к следующей системе ОДУ:

M(t)dc/dt = Rc,

где c(t)=[c_1(t) ... c_n(t)], а трехдиагональные матрицы M(t) и R определяются соотношениями

$$M_{ij}=\begin{cases}\exp(-t)2h/3, & i=j\\ \exp(-t)h/6, & i=j\pm1\\ 0, & i\neq j-1,j,j+1\end{cases}$$

$$R_{ij}=\begin{cases}-2/h, & i=j\\ 1/h, & i=j\pm1\\ 0, & i\neq j-1,j,j+1\end{cases}$$

Найдем решение этой системы для n = 20 в моменты времени 0:0.5:1.5

```
[t, X] = ode15s('fem1ode', [0:0.5:1.5], [ ], odeset('Mass', 'on'), 20);
```

Построим пространственное распределение u(x) в указанные моменты времени:

```
plot(1:20, X(1:4, :)), grid, hold on
gtext('t=0'), gtext('0.5'), gtext('1.0'), gtext('1.5')
```

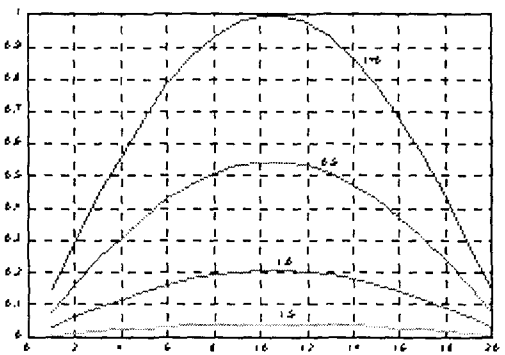


Рис. 8.20

Пример 5. Ограниченная проблема трех тел (фиксация событий):

Рассмотрим движение точки в поле тяготения двух тел, которое описывается следующей системой ОДУ:

$$x_1' = x_3;$$

$$x_2' = x_4;$$

$$x_3' = 2x_4 + x_1 - \frac{\mu^*(x_1+\mu)}{r_1^3} - \frac{\mu(x_1-\mu^*)}{r_2^3};$$

$$x_4' = -2x_3 + x_2 - \frac{\mu^* x_2}{r_1^3} - \frac{\mu x_2}{r_2^3}$$

где $\mu = 1.82.45$, $\mu^* = 1 - \mu$;

$$r_1 = \sqrt{(x_1+\mu)^2 + x_2^2}; \quad r_2 = \sqrt{(x_1-\mu^*)^2 + x_2^2} \quad .$$

Первые два компонента вектора состояния являются координатами положения точки в пространстве, что позволяет построить орбиту движения, как это показано на рис. 8.21. Начальные условия выбраны так, чтобы орбита была периодической. Это позволяет интерпретировать эту орбиту как траекторию облета Луны и возвращения на Землю. Требования к расчету траектории достаточно высоки: относительная ошибка RelTol не должна превышать 1e - 5, а абсолютная AbsTol - 1e - 4.

События, которые фиксируются в данной задаче, - это точка старта и точка максимального удаления от Земли.

```
options = odeset('Events', 'on');
[t, X, te, xe, ie] = ode45('orbitode', [ ], [ ], options);
```

te =	xe =				ie =
0.0000	1.2000	-0.0000	-0.0000	-1.0494	1
3.0953	-1.2616	-0.0012	-0.0005	1.0485	2

```
plot(X(:, 2), X(:, 1)),  hold on,  plot(xe(:, 2), xe(:, 1), 'or')
```

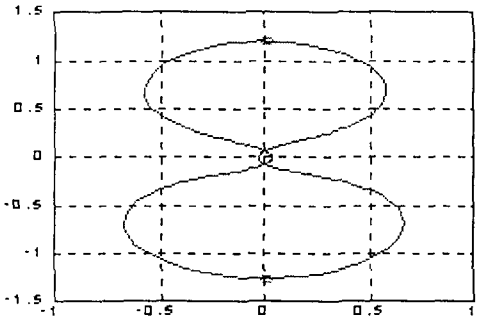


Рис. 8.21

С другими примерами интегрирования систем ОДУ можно ознакомиться, обратившись к демонстрационному файлу odedemo.

Ссылки:

1. Shampine L. F., Gordon M. K. Computer Solution of Ordinary Differential Equations. Freeman & Co. San Francisco: 1975. P. 318.
2. Сю Д., Мейер А. Современная теория автоматического управления и ее применение: Пер. с англ. М.: Машиностроение, 1972.
3. Shampine L. F. Evaluation of a test set for stiff ODE solvers// ACM Trans. Math. Soft. 1981. Vol. 7. P. 409-420.
4. Hairer E., Wanner G. Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems. Berlin: Springer:Verlag, 1991. P. 5-8.

Вычисление минимумов и нулей функции

FMIN, FOPTIONS — **Минимизация функции одной переменной**

Синтаксис:

```
xmin = fmin('<имя функции>', x1, x2)
xmin = fmin('<имя функции>', x1, x2, options)
xmin = fmin('<имя функции>', x1, x2, options, p1,..., p10)
[xmin, options] = fmin( ... )
options = foptions
```

Описание:

Функция xmin = fmin('<*имя функции*>', x1, x2) возвращает значение локального минимума функции в интервале x1 ≤ x ≤ x2.

Функция xmin = fmin('<*имя функции*>', x1, x2, options) использует вектор управляющих параметров options, который включает 18 компонентов, описанных ниже. Он предназначен для настройки алгоритмов оптимизации, применяемых как в системе MATLAB, так и в пакете программ Optimization Toolbox [1]. Функция fmin использует только 3 из этих параметров: options(1), options(2), options(14).

Функция xmin = fmin('<*имя функции*>', x1, x2, options, p1, ..., p10) позволяет передать до 10 дополнительных параметров; если опции используются по умолчанию, то вместо аргумента options следует указать пустой массив.

Функция [xmin, options] = fmin(...) возвращает вектор управляющих параметров options, которые использовались алгоритмом, и в частности параметр options(10), фиксирующий количество выполненных итераций, и параметр options(8), содержащий минимальное значение функции.

Функция options = foptions возвращает вектор-строку исходных значений параметров, используемых функциями fmin и fmins системы MATLAB и функциями fminu, constr, attgoal, minimax, leastsq, fsolve пакета Optimization Toolbox. Значения по умолчанию присваиваются внутри функций оптимизации и могут отличаться от исходных значений.

Опция	*Назначение опции*	*Исходное значение*	*Значение по умолчанию*
options(1)	Вывод промежуточных результатов: 0 - *не выводятся*; 1 - *выводятся*	0	0
options(2)	Итерационная погрешность для аргумента	1e-4	1e-4
options(3)	Итерационная погрешность для функции	1e-4	1e-4
options(4)	Терминальный критерий соблюдения ограничений	1e-6	1e-6
options(5)	Стратегия	0	0
options(6)	Оптимизация	0	0
options(7)	Алгоритм линейного поиска	0	0
options(8)	Значение целевой функции	0	0
options(9)	Для контроля градиента установить в 1	0	0
options(10)	Количество выполненных итераций	0	0
options(11)	Количество вычисленных градиентов	0	0
options(12)	Количество вычисленных ограничений	0	0
options(13)	Количество ограничений в виде равенств	0	0
options(14)	Максимальное количество итераций, n - количество переменных	0	fmin: 500 fmins: 200×n

options(15)	Параметр, используемый при наличии целевой функции	0	0
options(16)	Минимальное приращение переменных при вычислении градиента	1e-8	1e-8
options(17)	Максимальное приращение переменных при вычислении градиента	0.1	0.1
options(18)	Размер шага минимизации h	0	h ≤ 1

Пример:

Вычислим приближенное значение π путем минимизации функции y = cos(x) на отрезке [3 4] с итерационной погрешностью по x - 1e-12.

```
[xmin, opt] = fmin('cos', 3, 4, [1, 1e-12]);
 Func evals    x        f(x)       Procedure
    1       3.38197   -0.971249     initial
    2       3.61803   -0.888633     golden
    3       3.23607   -0.995541     golden
    4       3.13571   -0.999983     parabolic
    5       3.1413        -1        parabolic
    6       3.14159       -1        parabolic
    7       3.14159       -1        parabolic
    8       3.14159       -1        parabolic
    9       3.14159       -1        parabolic
xmin =   3.14159265402771
   pi =   3.14159265358979
norm(xmin - pi) = 4.3792e-010
x = 3 : 0.01 : 3.8;
y = cos(x);
plot(x, y, 'r'), grid, hold on
xt = [3.38 3.62 3.24 3.13];  yt=cos(xt);
h = plot(xt, yt, '-og');  set(h, 'LineWidth', 2)
title('Функция y = cos(x)')
```

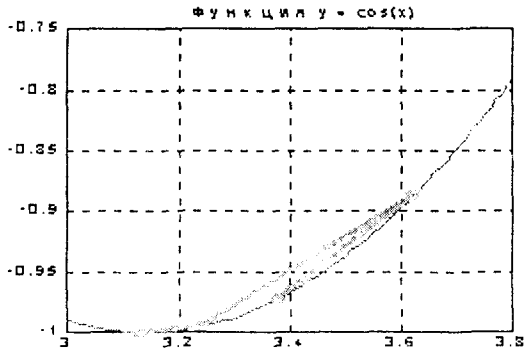


Рис. 8.22

Обратите внимание, что *итерационная погрешность* (разница между двумя соседними итерациями) отличается от *погрешности вычисления* (разница между вычисленным и машинным значением π, что показывает следующая таблица:

Номер опции opt	*Исходное значение*	*Значение при выходе из функции* fmin
1	0	0
2	1.0000e-004	1.0000e-012
3	1.0000e-004	1.0000e-004
4	1.0000e-006	1.0000e-006
5	0	0
6	0	0
7	0	0
8	0	-1.0000e+000
9	0	0
10	0	9.0000e+000
11	0	0
12	0	0
13	0	0
14	0	5.0000e+002
15	0	0
16	1.0000e-008	1.0000e-008
17	1.0000e-001	1.0000e-001
18	0	0

Анализ таблицы подтверждает, что заданная итерационная погрешность opt(2) равна 1.0000e-012; минимальное значение функции opt(8) равно -1.0000e+000; число выполненных итераций opt(10) равно 9.0000e+000 при максимально допустимом числе итераций opt(14) = 5.0000e+002.

Алгоритм:

Функция fmin использует методы "золотого сечения" и параболической интерполяции [2].

Сопутствующие функции: FMINS, FZERO, Optimization Toolbox.

Ссылки:

1. Optimization Toolbox. User's Guide. Natick, MA: The MathWorks, Inc., 1997.
2. Forsythe G. E., Malcolm M. A., Moler C. B. Computer Methods for Mathematical Computations. N.Y.: Prentice-Hall, 1976.

FMINS — Минимизация функции нескольких переменных

Синтаксис:

```
xmin = fmins('<имя функции>', x0)
xmin = fmins('<имя функции>', x0, options)
xmin = fmins('<имя функции>', x0, options, [ ], arg1,..., arg10)
[xmin, options] = fmins( ... )
```

Описание:

Функция xmin = fmins('*<имя функции>*', x0) возвращает вектор xmin, соответствующий значению локального минимума функции в окрестности точки x0.

Функция xmin = fmins('*<имя функции>*', x0, options) использует вектор управляющих параметров options, который включает 18 компонентов, соответствующих функции foptions. Функция fmins использует только 4 из этих параметров: options(1), options(2), options(3), options(14).

Функция xmin = fmins('*<имя функции>*', x1, x2, options, p1, ..., p10) позволяет передать до 10 дополнительных параметров; если опции используются по умолчанию, то вместо аргумента options следует указать пустой массив.

Функция [xmin, options] = fmins(...) возвращает вектор управляющих параметров options, которые использовались алгоритмом, и в частности параметр options(10), фиксирующий количество выполненных итераций, и параметр options(8), содержащий минимальное значение функции.

Пример:

Рассмотрим классический пример минимизации функции Розенброка от двух переменных.

$$f(x) = 100(x_2 - x_1^2)^2 + (a - x_1)^2.$$

Минимум этой функции достигается в точке (a, a^2) и равен нулю. В качестве начальной точки обычно выбирается точка (-1.2 1).

Определим М-файл rosenbr(x, a), обеспечивающий вычисление этой функции для заданного параметра a.

```
function y = rosenbr(x, a)
  if nargin < 2, a = 1; end
  y = 100 * (x(2) - x(1)^2)^2 + (a - x(1))^2;
```

Рассмотрим случай a = 1.

```
[xmin, opt] = fmins('rosenbr', [-1.2 1], [0, 1e-6]);
xmin = 9.999997969915502e-001   9.999995750683192e-001
```

Номер опции opt	*Исходное значение*	*Значение при выходе из функции* fmin
1	0	0
2	1.0000e-004	1.0000e-006
3	1.0000e-004	1.0000e-004
4	1.0000e-006	1.0000e-006
5	0	0
6	0	0
7	0	0
8	0	7.6989e-014
9	0	0
10	0	1.9900e+002
11	0	0
12	0	0
13	0	0
14	0	4.0000e+002
15	0	0
16	1.0000e-008	1.0000e-008
17	1.0000e-001	1.0000e-001
18	0	0

Анализ таблицы подтверждает, что заданная итерационная погрешность opt(2) равна 1.0000e-06; минимальное значение функции opt(8) равно 7.6989e-014; число выполненных итераций opt(10) равно 1.9900e+002 при максимально допустимом числе итераций opt(14) = 4.0000e+002.

Алгоритм:

Функция fmins реализует метод Нелдера - Мида [2, 3], который является прямым методом, не требующим вычисления градиента или иной информации о производной, и связан с построением симплекса в n-мерном пространстве, который задается n+1-й вершиной. В двумерном пространстве симплекс - это треугольник, а в трехмерном - пирамида.

На каждом шагу алгоритма новая точка выбирается внутри или вблизи симплекса. Значение функции в этой точке сравнивается со значениями в вершинах симплекса, и, как правило, одна из вершин заменяется этой точкой, задавая тем самым новый симплекс. Эти итерации повторяются до тех пор, пока диаметр симплекса не станет меньше заданной величины итерационной погрешности по переменным x.

Сопутствующие функции: FMIN, FZERO, Optimization Toolbox.

Ссылки:

1. Optimization Toolbox: User's Guide. Natick: The MathWorks, Inc., 1997.
2. Nelder J. A., Mead R. A Simplex Method for Function Minimization//Computer Journal. Vol. 7. P. 308-313.
3. Dennis J. E., Woods D. J. New Computing Environments: Microcomputers in Large-Scale Computing/ Ed. by A. Wouk// *SIAM*. 1987. P. 116-122.

FZERO — Нахождение нулей функции одной переменной

Синтаксис:

```
z = fzero('<имя функции>', x)
z = fzero('<имя функции>', x, tol)
z = fzero('<имя функции>', x, tol, trace)
z = fzero('<имя функции>', x, tol, trace, P1, P2, ...)
```

Описание:

Функция z = fzero('*<имя функции>*', x) находит нуль одномерной действительной функции от действительной переменной. Когда x скалярная величина, то решение ищется в окрестности заданной этой точки путем отыскания интервала, где функция меняет знак. Если такой интервал не находится, то возвращается значение NaN. Когда x вектор длины 2, то он интерпретируется как интервал поиска, в котором функция меняет знак. В том случае, если это не так, формируется сообщение об ошибке.

Функция z = fzero('*<имя функции>*', x, tol) возвращает результат с относительной погрешностью tol, задаваемой пользователем. По умолчанию tol = eps.

Функция z = fzero('*<имя функции>*', x, tol, trace) при значении trace, отличном от нуля, выводит на экран терминала промежуточные результаты.

Функция z = fzero('*<имя функции>*', x, tol, trace, P1, P2, ...) позволяет передать дополнительные параметры для определения функции; если значения аргументов tol и trace используются по умолчанию, то на их месте следует указать пустые массивы.

Пример:

Вычислим действительные нули полинома $f(x) = x^4 - 4 * x^3 + 12$.

Сначала сформируем M-файл polynom4 для вычисления этой функции:

```
function y = polynom4(x)
y = x.^4 - 4*x.^3 + 12;
```

затем найдем корень полинома, стартуя из точки x0 = -0.5:

```
z = fzero(' polynom4', -0.5, eps)
z = 1.746170944975038e+000;
```

теперь найдем корень полинома, стартуя из точки x0 = 3.0:

```
z = fzero(' polynom4', 3, eps)
z = 3.777351952771215e+000;
```

точные значения корней полинома могут быть вычислены с помощью функции

```
roots([1 -4 0 0 12])
ans =
   3.777351952771212e+000
   1.746170944975038e+000
  -7.617614488731261e-001 +1.113117638472703e+000i
  -7.617614488731261e-001 - 1.113117638472703e+000i
```

Нетрудно видеть, что оба подхода дают согласованные результаты.

Алгоритм:

Функция fzero использует методы деления отрезка пополам, секущей и обратной квадратической интерполяции [1, 2].

Ограничения:

Функция fzero определяет нуль как точку, где функция пересекает ось x; точки, в которых функция только касается оси x, не считаются нулями. Функция выполняется до тех пор, пока не будет найдено комплексное решение или сгенерированы решения Inf или NaN.

Сопутствующие функции: FMIN, ROOTS.

Ссылки:

1. Brent R. *Algorithms for Minimization Without Derivatives.* Prentice-Hall, N. Y. 1973.
2. Forsythe G. E., Malcolm M. A., Moler C. B. *Computer Methods for Mathematical Computations.* Prentice-Hall, N. Y. 1976.

Преобразования Фурье

FFT, IFFT **Одномерные дискретные прямое и обратное преобразования Фурье**

Синтаксис:

Y = fft(X)	X = ifft(Y)
Y = fft(X, n)	X = ifft(Y, n)
Y = fft(X, [ ], dim)	Y = ifft(X, [ ], dim)
Y = fft(X, n, dim)	Y = ifft(X, n, dim)

Определения:

Дискретные прямое и обратное преобразования Фурье для одномерного массива x длины N определяются следующим образом:

$$X(k) = \sum_{j=1}^{N} x(j) e^{2\pi/N(j-1)(k-1)};$$

$$x(k) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{-2\pi/N(j-1)(k-1)}.$$

Описание:

Функция Y = fft(X) вычисляет для массива данных X дискретное преобразование Фурье, используя FFT-алгоритм быстрого Фурье-преобразования. Если массив X двумерный, вычисляется дискретное преобразование каждого столбца. Если X многомерный массив, то вычисляется преобразование Фурье по первой неединичной размерности.

Функция Y = fft(X, n) вычисляет n-точечное дискретное преобразование Фурье. Если length(X) < n, то недостающие строки массива X заполняются нулями; если length(X) > n, то лишние строки удаляются.

Функции Y = fft(X, [ ], dim) и Y = fft(X, n, dim) вычисляют дискретные преобразования Фурье по размерности dim.

Функция X = ifft(Y) вычисляет обратное преобразование Фурье для массива Y. Если Y многомерный массив, то вычисляется обратное преобразование Фурье по первой неединичной размерности.

Функция X = ifft(Y, n) вычисляет n-точечное обратное преобразование Фурье для массива Y.

Функции X = ifft(Y, [ ], dim) и X = fft(Y, n, dim) вычисляют обратные преобразования Фурье по размерности dim.

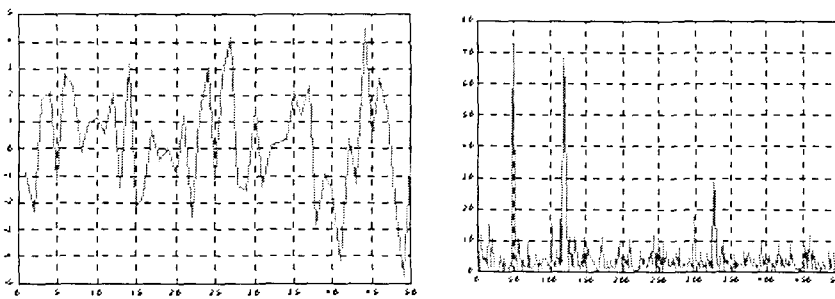
Примеры:

Основное назначение преобразования Фурье - выделить частоты регулярных составляющих сигнала, зашумленного помехами. Рассмотрим данные, поступающие с частотой 1000 Гц. Сформируем сигнал, содержащий регулярные составляющие с частотами 50 и 120 Гц и случайную аддитивную компоненту с нулевым средним.

```
t = 0:0.001:0.6;
x = sin(2 * pi * 50 * t) + sin(2 * pi * 120 * t);
y = x + 2 * randn(size(t));
plot(y(1:50)), grid
```

На рис. 8.23, *а* показан этот сигнал. Глядя на него, трудно сказать, каковы частоты его регулярных составляющих. Реализуя одномерное преобразование Фурье этого сигнала на основе 512 точек и построив график спектральной плотности (рис. 8.23, *б*), можно выделить две частоты, на которых амплитуда спектра максимальна. Это частоты 120 и 50 Гц.

```
Y = fft(y, 512);
Pyy = Y.*conj(Y)/512;
f = 1000 * (0:255)/512;
plot(f, Pyy(1:256)), grid
```



а б

Рис. 8.23

Алгоритм:

Если длина последовательности входных данных является степенью числа 2, то применяется алгоритм быстрого преобразования Фурье с основанием 2, имеющий максимальную производительность. Этот алгоритм оптимизирован для работы с действительными данными; если данные комплексные, то реализуется комплексное преобразование Фурье. Эффективность первого на 40 % выше второго.

Если длина входной последовательности не является степенью числа 2, то применяется преобразование со смешанными основаниями, которые определяются как простые множители длины входной последовательности, которая при необходимости подвергается усечению.

Время расчета существенно зависит от значения длины последовательности. Если значение длины точно разлагается на простые множители, то вычисления для такой последовательности выполняются достаточно быстро; если же не все множители оказываются простыми, и даже их будет меньше, то время вычисления существенно возрастает.

Если сравнивать эффективность вычислений, то преобразование Фурье с основанием 2 действительной последовательности из 4096 точек занимает 2.1 с, а комплексной - 3.7 с. Обычное преобразование Фурье для последовательности из 4096 точек занимает 7 с, а для последовательности из 4098 точек - 58 с.

Сопутствующие функции: FFT2, IFFT2, FFTSHIFT, Signal Processing Toolbox [1].

Ссылки:

1. Signal Processing Toolbox. *User's Guide.* Natick, MA: The MathWorks, Inc., 1993.

FFT2, IFFT2 **Двумерные дискретные прямое и обратное преобразования Фурье**

Синтаксис:

```
Y = fft2(X)          X = ifft2(Y)
Y = fft2(X, m, n)    X = ifft2(Y, m, n)
```

Описание:

Функция Y = fft2(X) вычисляет для массива данных X двумерное дискретное преобразование Фурье. Если массив X двумерный, вычисляется дискретное преобразование для каждого столбца.

Функция Y = fft2(X, m, n) вычисляет двумерное дискретное преобразование Фурье в виде массива размера m×n; при этом недостающие строки и столбцы массива X заполняются нулями, а лишние строки и столбцы удаляются.

Функция X = ifft2(Y) вычисляет обратное преобразование Фурье для массива Y.

Функция X = ifft2(Y, m, n) вычисляет обратное преобразование Фурье для массива Y в виде массива размера m×n.

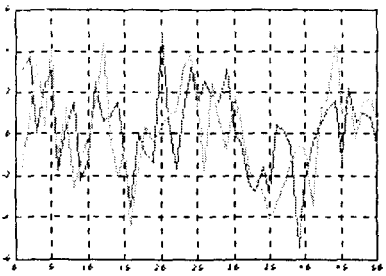
Примеры:

Рассмотрим тот же пример, что и для функции fft, но сформируем 2 входных последовательности (рис. 8.24, *а*):

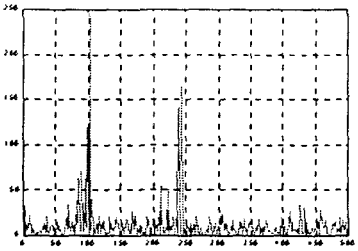
```
t = 0:0.001:0.6;
x = sin(2*pi*50*t) + sin(2*pi*120*t);
y1 = x + 2*randn(size(t));
y2 = x + 2*randn(size(t));
y = [y1; y2];
plot(y(1, 1:50)), hold on, plot(y(2, 1:50)),  grid, hold off
```

Применим двумерное преобразование Фурье для сигнала y на основе 512 точек и построим график спектральной плотности. Теперь можно выделить 2 частоты, на которых амплитуда спектра максимальна. Это частоты 100/2 и 240/2 Гц (рис. 8.24, б).

```
Y = fft2(y, 2, 512);
Pyy = Y.*conj(Y)/512;
f = 1000*(0:255)/512;
plot(f, Pyy(1:256)), grid
```



а



б

Рис. 8.24

Алгоритм:

Двумерное дискретное преобразование связано с одномерным дискретным преобразованием Фурье следующим образом:

fft2(X) = fft(fft(X).').'

Сопутствующие функции: FFT, IFFT, FFTSHIFT, Signal Processing Toolbox.

Ссылки:

1. Signal Processing Toolbox. *User's Guide*. Natick, MA: The MathWorks, Inc., 1993.

FFTN, IFFTN

Многомерные дискретные прямое и обратное преобразования Фурье

Синтаксис:

Y = fftn(X)	X = ifftn(Y)
Y = fftn(X, siz)	X = ifftn(Y, siz)

Описание:

Функция Y = fftn(X) вычисляет для многомерного массива данных X n-мерное дискретное преобразование Фурье. Результирующий массив Y имеет те же размеры, что и входной массив X.

Функция Y = fftn(X, siz) вычисляет n-мерное дискретное преобразование Фурье в виде многомерного массива размера siz; при этом недостающие размерности входного массива X заполняются нулями, а лишние размерности удаляются.

Функция X = ifftn(Y) вычисляет n-мерное обратное преобразование Фурье. Результирующий массив X имеет те же размеры, что и входной массив Y.

Функция X = ifftn(Y, siz) вычисляет n-мерное дискретное преобразование Фурье в виде многомерного массива размера siz; при этом недостающие размерности входного массива Y заполняются нулями, а лишние размерности удаляются.

Описание:

Вычисление n-мерных прямого и обратного преобразований Фурье равносильно следующим последовательностям одномерных преобразований:

```
Y = X;
for p = 1:length(size(X))
  Y = fft(Y, [ ], p);
end
```

```
X = Y;
for p = 1:length(size(Y))
  X = ifft(X, [ ], p);
end
```

Время, требуемое для вычислений, зависит от вида разложения на простые множители значений размерностей входного массива; в том случае, когда значения размерностей являются степенями по основанию 2, скорость вычислений максимальна.

Замечание:

С учетом ошибок округления возможно, что последовательность преобразований ifftn(fftn(X)) не будет восстанавливать X, а может включать малые значения мнимых частей.

Сопутствующие функции: FFT, FFT2, Signal Processing Toolbox [1].

Ссылки:

1. Signal Processing Toolbox. *User's Guide.* Natick, MA: The MathWorks, Inc., 1998.

FFTSHIFT, IFFTSHIFT — Перегруппировка выходных массивов преобразований Фурье

Синтаксис:

```
Y = fftshift(X)
X = ifftshift(Y)
```

Описание:

Функция Y = fftshift(X) перегруппировывает выходные массивы функций fft, fft2 и fftn, размещая нулевую частоту в центре спектра.

Если X - одномерный массив, то выполняется циклическая перестановка правой и левой его половины:

X	fftshift(X)
1 2 3 4 5	4 5 1 2 3

Если X - двумерный массив, то меняются местами квадранты: I ↔ IV и II ↔ III:

```
X = [' I '   ' II ';...
     ' III '   ' IV ';]
fftshift(X)
```

X	fftshift(X)
I II III IV	IV III II I

Если X - многомерный массив, то выполняется перегруппировка элементов по каждой размерности. Для четырехмерного массива X размера 1×2×2×2 это выглядит следующим образом:

```
X(:,:,2,1) =
     3     4
```

```
X(:,:,2,2) =
     7     8
```

```
X(:,:,1,1) =
     1     2
```

```
X(:,:,1,2) =
     5     6
```

fftshift(X)

```
X(:,:,2,1) =
     6     5
```

```
X(:,:,2,2) =
     2     1
```

```
X(:,:,1,1) =
     8     7
```

```
X(:,:,1,2) =
     4     3
```

Функция X = ifftshift(Y) выполняет обратную операцию.

Пример:

Рассмотрим тот же пример, который рассматривался и для функции fft, но введем постоянную составляющую с уровнем 0.3:

```
t = 0:0.001:0.6;
x = sin(2 * pi * 50 * t) + sin(2 * pi * 120 * t);
y = x + 2 * randn(size(t)) + 0.3;
plot(y(1:50)), grid
```

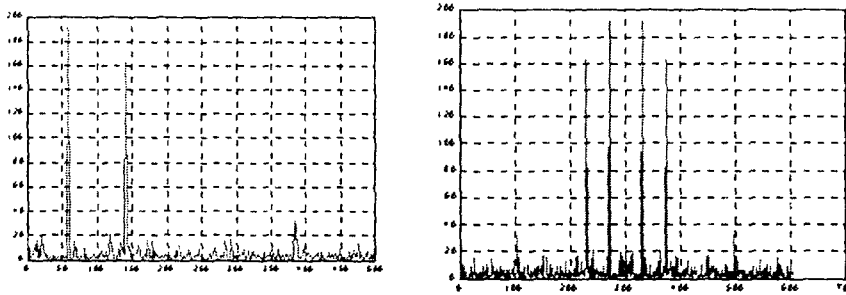
Применим одномерное преобразование Фурье для сигнала y и построим график спектральной плотности. Здесь можно выделить 3 частоты, на которых амплитуда спектра максимальна. Это частоты 0, 58.4 и 140.1 Гц (рис. 8.25, *а*).

```
Y = fft(y);
Pyy = Y.*conj(Y)/512;
f = 1000 * (0:255)/512;
plot(f, Pyy(1:256)), grid
ginput
```

58.4677 191.0979 **140.1210** 162.6113

Выполним операции

```
Y = fftshift(Y);
Pyy = Y.*conj(Y)/512;
plot(Pyy), grid
```



а б

Рис. 8.25

Из анализа рис. 8.25, *б* следует, что нулевая частота сместилась в середину спектра.

Сопутствующие функции: FFT, FFT2, Signal Processing Toolbox [1].

Ссылки:

1. Signal Processing Toolbox. *User's Guide.* Natick, MA: The MathWorks, Inc., 1998.

Свертка и фильтрация

CONV, DECONV **Свертка одномерных массивов**

Синтаксис:

z = conv(x, y) [q, r] = deconv(z, x)

Описание:

Если заданы одномерные массивы x и y длины соответственно m = length(x) и n = length(y), то свертка z - это одномерный массив длины m + n -1, k-й элемент которого определяется по формуле

$$z(k) = \sum_{j=\max(1,k+1-n)}^{\min(k,m)} x(j)\,y(k+1-j).$$

Функция z = conv(x, y) вычисляет свертку z двух одномерных массивов x и y.

Рассматривая эти массивы как выборки из двух сигналов, можно сформулировать теорему свертки в следующей форме:

Если X = fft([x zeros(1, length(y)-1]) *и* Y = fft([y zeros(1, length(x) - 1]) - *согласованные по размерам преобразования Фурье сигналов* x *и* y, *то справедливо соотношение* conv(x, y) = ifft(X.*Y).

Иначе говоря, свертка двух сигналов эквивалентна умножению преобразований Фурье этих сигналов.

Функция [q, r] = deconv(z, x) выполняет операцию, обратную операции свертки. Эта операция равносильна определению импульсной характеристики фильтра. Если справедливо соотношение z = conv(x, y), то q = y, r = 0.

Сопутствующие функции: Signal Processing Toolbox [1].

Ссылки:

1. Signal Processing Toolbox User's. *Guide*. Natick, MA: The MathWorks, Inc., 1998.

CONV2 — Свертка двумерных массивов

Синтаксис:

Z = conv2(X, Y)
Z = conv2(hcol, hrow, A)
Z = conv2(X, Y, '<*опция*>')

Описание:

Функция Z = conv2(X, Y) вычисляет свертку Z двумерных массивов X и Y. Если массивы имеют размеры соответственно mx×nx и my×ny, то размер массива Z равен (mx + my - 1) × (nx + ny - 1).

Функция Z = conv2(hcol, hrow, A) вычисляет свертку по столбцам, используя вектор hcol, и по строкам, используя вектор hrow.

Функция Z = conv2(X, Y, '<*опция*>') имеет опцию для управления размером массива Z, которая может принимать следующие значения:

'full' - полноразмерная свертка (по умолчанию);
'same' - центральная часть размера mx×nx;
'valid' - центральная часть размера (mx-my+1)×(nx-ny+1) при условии, что (mx×nx) > (my×ny).

Процедура conv2 выполняется наиболее эффективно, если выполнено условие (mx×nx) > (my×ny), то есть количество элементов массива X превосходит количество элементов массива Y.

Пример:

Рассмотрим два массива X и Y.

X

8	1	6
3	5	7
4	9	2

Y

1	1
1	1

И вычислим их свертку для различных значений опции.

Z = conv2(X, Y)				Zs = conv2(X, Y, 'same')			Zv = conv2(X, Y, 'valid')	
8	9	7	6	17	19	13	17	19
11	17	19	13	21	23	9	21	23
7	21	23	9	13	11	2		
4	13	11	2					

Сравнивая столбцы последней таблицы, можно прояснить назначение опций 'full', 'same', 'valid'.

Сопутствующие функции: CONV, DECONV, FILTER2, Signal Processing Toolbox [1].

Ссылки:

1. Signal Processing Toolbox. *User's Guide.* Natick, MA: The MathWorks, Inc., 1998.

CONVN — Свертка многомерных массивов

Синтаксис:

```
Z = convn(X, Y)
Z = convn(X, Y, '<опция>')
```

Описание:

Функция Z = conv2(X, Y) вычисляет свертку Z многомерных массивов X и Y. Если массивы имеют размеры соответственно size(X) и size(Y), то размер массива Z равен size(X) + size(Y) - 1.

Функция Z = conv2(X, Y, '*<опция>*') имеет опцию для управления размером массива Z, которая может принимать следующие значения:

'full' - полноразмерная свертка (по умолчанию);
'same' - центральная часть размера size(X);
'valid' - центральная часть размера max(size(X)–size(Y) + 1, 0) в предположении, что массив X не дополняется нулями.

Сопутствующие функции: CONV, CONV2, Signal Processing Toolbox [1].

Ссылки:

1. *Signal Processing Toolbox User's Guide.* Natick: The MathWorks, Inc., 1998.

FILTER — Дискретная одномерная фильтрация

Синтаксис:

```
y = filter(b, a, X)
[y, zf] = filter(b, a, X)
[y, zf] = filter(b, a, X, zi)
y = filter(b, a, X, zi, dim)
[...] = filter(b, a, X, [ ], dim)
```

Описание:

Функция y = filter(b, a, X) фильтрует сигнал, заданный в виде одномерного массива X, используя дискретный фильтр, описываемый конечно-разностными уравнениями вида

$$a(1)*y(n) = b(1) * x(n) + b(2) * x(n - 1) + ... + b(nb + 1) * x(n - nb) - a(2) * y(n - 1) - ... - a(na + 1) * y(n - na),$$

при этом входной параметр b = [b(1) b(2) ... b(nb + 1)], а параметр a = [a(1) a(2) ... ... a(na+1)]. Если коэффициент a(1) не равен 1, то выполняется его приведение к 1; если коэффициент a(1) равен 0, то выдастся сообщение об ошибке.

Если X двумерный массив, то обработка выполняется по столбцам.

Если X многомерный массив, то обработка выполняется начиная с первой неединичной размерности.

Функция [y, zf] = filter(b, a, X) позволяет учесть запаздывание в выходном сигнале с помощью вектора zf размера max(size(a), size(b)) или массива таких векторов для каждого столбца массива X.

Функция [y, zf] = filter(b, a, X, zi) учитывает запаздывания во входном и выходном сигналах соответственно с помощью векторов zi и zf. Входной вектор zi (или массив векторов) имеет размер max(length(a), length(b))-1.

Функции y = filter(b, a, X, zi, dim) и [...] = filter(b, a, X, [ ], dim) выполняют фильтрацию вдоль размерности dim.

Сопутствующие функции: FILTER2, Signal Processing Toolbox [1].

Ссылки:

1. Signal Processing Toolbox. *User's Guide.* Natick, MA: The MathWorks, Inc., 1998.

FILTER2 **Дискретная двумерная фильтрация**

Синтаксис:

Y = filter2(B, X)

Y = filter2(B, X, '<*опция*>')

Описание:

Функция Y = filter2(B, X) фильтрует сигнал, заданный в виде двумерного массива X, используя дискретный фильтр, описываемый матрицей B. Результат имеет те же размеры, которые имеет и массив X, и вычисляется с использованием двумерной свертки.

Функция Y = filter2(B, X, '<*опция*>') имеет опцию для управления размером массива Y, которая может принимать следующие значения:

'same'	- size(Y) = size(X) (по умолчанию);
'valid'	- size(Y) < size(X);
'full'	- size(Y) > size(X).

Пример:

Рассмотрим фильтр B и массив X.

B		X		
1	1	8	1	6
1	1	3	5	7
		4	9	2

И вычислим результаты для различных значений опции.

Y = filter2(B, X)	Y = filter2(B, X, 'valid')	Zv = conv2(X, Y, 'full')
17 19 13	17 19	8 9 7 6
21 23 9	21 23	11 17 19 13
13 11 2		7 21 23 9
		4 13 11 2

Сопутствующие функции: CONV2, FILTER, Signal Processing Toolbox [1].

Ссылки:

1. Signal Processing Toolbox. *User's Guide.* Natick, MA: The MathWorks, Inc., 1998.

UNWRAP — Корректировка фазовых углов

Синтаксис:

```
Q = unwrap(P)
Q = unwrap(P, cutoff)
Q = unwrap(P, [ ], dim)
Q = unwrap(P, cutoff, dim)
```

Описание:

Функция Q = unwrap(P) корректирует фазовые углы элементов одномерного массива P при переходе через значение π, дополняя их значениями $\pm 2\pi$ для того, чтобы убрать разрывы функции; если P двумерный массив, то соответствующая функция применяется к столбцам; если P многомерный массив, то функция применяется к первой неединичной размерности.

Функция Q = unwrap(P, cutoff) позволяет пользователю изменить значение cutoff критического угла; по умолчанию cutoff = π.

Функции Q = unwrap(P, [ ], dim) и Q = unwrap(P, cutoff, dim) корректируют фазовые углы вдоль размерности dim многомерного массива P.

Пример:

Рассмотрим непрерывный неминимально-фазовый фильтр, описываемый передаточной функцией

$$\frac{num}{den} = \frac{[1 \quad -1]}{[1 \quad 2 \quad 2 \quad 1 \quad 1]}.$$

Вычислим частотную характеристику этого фильтра в диапазоне w = 0.1:0.01:10, используя функцию freqresp пакета Control System Toolbox [1]:

```
w = 0.1:0.01:10; num = [1 -1]; den = [1 2 2 1 1];
g = freqresp(num, den, sqrt(-1) * w);
```

Вычислим фазовый угол частотной характеристики без использования и с использованием функции unwrap:

```
ph1 = (180./pi) * (atan2(imag(g), real(g)));
ph2 = (180./pi) * unwrap(atan2(imag(g), real(g)));
```

w	ph1	ph2
0.6000	126.7350	126.7350
0.7000	141.9270	141.9270
0.8000	***-158.7123***	201.2877
0.9000	***-135.6888***	224.3112
1.0000	***135.0000***	-225.0000
1.1000	***-139.3435***	220.6565
1.2000	***-145.0993***	214.9007
1.3000	***-151.1794***	208.8206
1.4000	***-157.1667***	202.8333
1.5000	***-162.8839***	197.1161
1.6000	***-168.2575***	191.7425
1.7000	***-173.2642***	186.7358
1.8000	***-177.9068***	182.0932
1.9000	177.7986	177.7986
2.0000	173.8298	173.8298

Из таблицы следует, что при входе и выходе из диапазона частот 0.8-2.0 Гц фазовые углы ph1 при достижении критического угла π терпят разрывы, которые устраняются функцией unwrap.

Построим фазовые частотные характеристики ph1(w) и ph2(w)-360, которые подтверждают сделанные выводы.

```
semilogx(w, ph1), hold on, grid
semilogx(w, ph2-360, 'b')
```

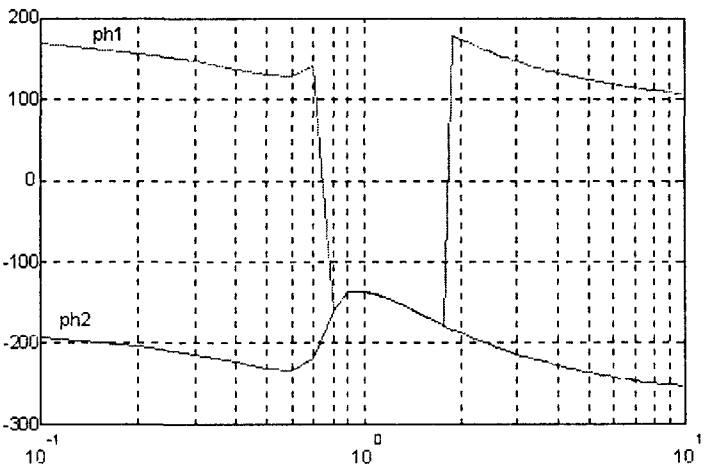


Рис. 8.26

Сопутствующие функции: ANGLE, ABS, Control System Toolbox [1].

Ссылки:

1. Control System Toolbox. *User's Guide*. Natick, MA: The MathWorks, Inc., 1997.

9. РАБОТА С РАЗРЕЖЕННЫМИ МАТРИЦАМИ

При решении многих прикладных задач приходится иметь дело с *разреженными матрицами*, то есть с матрицами, имеющими много нулевых элементов. К числу таких задач в первую очередь следует отнести граничные задачи для систем дифференциальных уравнений в частных производных. Возникающие при этом модели - это, как правило, квадратные матрицы высокого порядка с малым количеством ненулевых диагоналей.

Известно, что матрица порядка n требует для своего хранения n^2 байт оперативной памяти, а время выполнения матричных операций пропорционально n^3. Поэтому, когда количество неизвестных переменных достигает нескольких сотен, вычисления с полными матрицами становятся неэффективными и необходимо принимать во внимание *степень разреженности* матрицы, то есть отношение количества ненулевых элементов к общему количеству элементов матрицы.

Для разреженной матрицы S порядка m×n с количеством ненулевых элементов nnz(S) требования к объему оперативной памяти пропорциональны nnz. Вычислительная сложность операций с элементами разреженной матрицы также пропорциональна nnz, линейно зависит от m и n и не зависит от произведения m×n. Сложность такой операции, как решение системы линейных уравнений с разреженной матрицей, зависит от упорядочения элементов и заполненности матрицы, что обсуждается позднее в этой главе.

При работе с разреженными матрицами соблюдается фундаментальный принцип вычислительной сложности: время, необходимое для выполнения матричных операций с разреженной матрицей, пропорционально количеству арифметических операций с ненулевыми элементами. Это подтверждает широко распространенное правило: время вычислений пропорционально количеству операций с плавающей точкой.

В этой главе описан новый класс операций, реализованных в системе MATLAB, для работы с *разреженными матрицами*. Это операции хранения, преобразования, упорядочения, графического представления и решения задач линейной алгебры.

Элементарные разреженные матрицы

SPARSE — **Формирование разреженной матрицы**

Синтаксис:

```
S = sparse(A)
S = sparse(i, j, s, m, n, nzmax)
S = sparse(i, j, s, m, n)
S = sparse(i, j, s)
S = sparse(m, n)
```

Описание:

Функция sparse - это встроенная функция, которая формирует матрицы в соответствии с правилами записи разреженных матриц, принятыми в системе MATLAB; количество входных аргументов этой функции - от 1 до 6.

Функция S = sparse(A) преобразовывает полную матрицу в разреженную, удаляя из описания нулевые элементы; если исходная матрица разреженная, то sparse(S) возвращает S.

Общая форма функции S = sparse(i, j, s, m, n, nzmax) использует строки массива [i j s] для формирования разреженной матрицы размера m×n с ненулевыми элементами, количество которых не превышает nzmax. Векторы i и j задают позиции элементов и являются целочисленными, а вектор s определяет числовые значения элементов, которые могут быть действительными или комплексными. При формировании разреженной матрицы все строки вида [i j 0] из описания удаляются. Длина результирующего вектора s точно совпадает с количеством ненулевых элементов разреженной матрицы.

Обращение вида S = sparse(i, j, s, m, n) предполагает по умолчанию, что nzmax = lenght(s).

Обращение вида S = sparse(i, j, s) предполагает, что m = max(i) и n = max(j); эти функции вычисляются раньше, чем будут удалены строки с нулевыми значениями s.

Обращение вида S = sparse(m, n) резервирует пространство для разреженной матрицы и равносильно обращению sparse([ ], [ ], [ ], m, n, 0), где все m×n элементов являются нулевыми.

Над разреженными матрицами могут совершаться любые арифметические, логические и индексные операции, то же справедливо и для смешанных операндов - полных и разреженных массивов. Операции с однородными операндами возвращают тот же тип операнда; в случае смешанных операндов, как правило, возвращается полный тип, за исключением случаев, когда в явном виде сохраняется разреженный тип. Например, при поэлементном умножении массивов A .* S, где S - разреженный массив.

Некоторые операции, например S >= 0, приводят к генерации так называемых BS(Big Space)-матриц, которые имеют разреженную структуру, но очень большое количество нулевых элементов.

Примеры:

Функция S = sparse(1:n, 1:n, 1) формирует единичную матрицу n×n с разреженной структурой и коэффициентом заполнения 1/n. Тот же результат может быть получен с помощью оператора S = sparse(eye(n, n)), но при этом промежуточная матрица будет иметь полную структуру.

Массив вида B = sparse(10000, 10000, pi), состоящий из одного элемента, вероятно, не очень полезен, но такое обращение допустимо. Не пытайтесь применить функцию full(B); вам потребуется 800 Мбайт памяти.

Приводимая ниже последовательность операторов сначала определяет структуру, а затем формирует разреженную матрицу.

```
[i, j, s] = find(S);
[m, n] = size(S);
S = sparse(i, j, s, m, n);
```

Сопутствующие функции: DIAG, FIND, FULL, NNZ, NONZEROS, NZMAX, SPONES, SPRAND, SPRANDSYM, SPY, каталог SPARFUN.

SPDIAGS **Формирование диагоналей разреженной матрицы**

Синтаксис:

```
[B, d] = spdiags(A)
B = spdiags(A, d)
A = spdiags(B, d, A)
A = spdiags(B, d, m, n)
```

Описание:

Функция spdiags расширяет возможности встроенной функции diag и позволяет работать с различными комбинациями следующих трех матриц, которые могут быть как входами, так и выходами функции spdiags:

A - матрица размера m×n, как правило (но не обязательно), разреженная с ненулевыми элементами на p диагоналях;

B - матрица размера min(m, n)×p, как правило (но не обязательно), полная, столбцы которой являются диагоналями A;

d - вектор длины p, целочисленные элементы которого определяют номера ненулевых диагоналей A (верхние диагонали нумеруются положительными числами, нижние - отрицательными).

Грубо говоря, матрицы A, B и вектор d связаны друг с другом следующим образом:

```
for k = 1:p
   B(:, k) = diag(A, d(k))
end
```

Функция spdiags определяет следующие 4 операции в зависимости от количества входных аргументов:

- выделить все ненулевые диагонали:
  `[B, d] = spdiags(A);`
- выделить указанные диагонали:
  `B = spdiags(A, d);`
- заменить указанные диагонали матрицы A:
  `A = spdiags(B, d, A);`
- сформировать разреженную матрицу размера m×n по известным диагоналям:
  `A = spdiags(B, d, m, n);`

Пример:

Сформировать трехдиагональную разреженную матрицу для разностного оператора 2-го порядка, заданного на сетке из n узлов.

```
e = ones(n, 1);
A = spdiags([e -2*e e], -1:1, n, n)
```

Теперь из нее можно сформировать тестовую матрицу Уилкинсона wilkinson(n), изменяя элементы главной диагонали:

```
A = spdiags(abs(-(n-1)/2:(n-1)/2)', 0, A)
```

В заключение выделим ненулевые диагонали:

```
B = spdiags(A)
```

Сопутствующие функции: DIAG.

SPEYE — **Единичная разреженная матрица**

Синтаксис:

```
S = speye(m, n)
S = speye(n)
```

Описание:

Функция speye(m, n) формирует разреженную матрицу размера m×n с единицами на главной диагонали и нулевыми внедиагональными элементами.

Функция speye(n) равносильна функции speye(n, n).

Пример:

Оператор A = speye(1000) формирует разреженный массив, соответствующий единичной матрице размера 1000 × 1000 и требует для этого всего 16 Кбайт памяти. Аналогичный конечный результат может быть получен с помощью оператора A = sparse(eye(1000,1000)), но в этом случае промежуточная матрица будет полной.

Сопутствующие функции: SPALLOC, SPONES, SPRAND, SPRANDN, SPRANDSYM.

SPRAND, SPRANDN — **Случайные разреженные матрицы**

Синтаксис:

```
R = sprand(S)                    R = sprandn(S)
R = sprand(m, n, alpha)          R = sprandn(m, n, alpha)
R = sprand(m, n, alpha, rcond)   R = sprandn(m, n, alpha, rcond)
```

Описание:

Матрица вида R = sprand(S) имеет ту же структуру, которую имеет и разреженная матрица S, но при этом значения ее ненулевых элементов распределены по равномерному закону в интервале (0, 1).

Матрица вида R = sprand(m, n, alpha) - это случайная разреженная матрица, которая имеет приблизительно alpha * m * n ненулевых элементов, распределенных по равномерному закону, где alpha - коэффициент заполнения со значением в пределах 0 ≤ alpha ≤ 1.

Матрица вида R = sprand(m, n, alpha, rcond) в дополнение к вышеперечисленным условиям имеет также число обусловленности по отношению к операции обращения, близкое к значению rcond. Матрица R формируется в виде суммы матриц ранга 1.

Если rcond - вектор длины lr ≤ min(m, n), то матрица R имеет в качестве первых lr сингулярных чисел значения вектора rcond, а остальные сингулярные числа равны нулю. В этом случае матрица R генерируется с помощью матриц случайных плоских вращений, которые применяются к диагональной матрице с заданным спектром сингулярных чисел.

Матрица вида R = sprandn(S) имеет ту же структуру, которую имеет и разреженная матрица S, но при этом значения ее ненулевых элементов распределены по нормальному закону со средним, равным нулю, и дисперсией 1.

Матрица вида R = sprandn(m, n, alpha) - это случайная разреженная матрица, которая имеет приблизительно alpha * m * n ненулевых элементов, распределенных по нормальному закону, где alpha - коэффициент заполнения со значением в пределах 0 ≤ alpha ≤ 1.

Матрица вида R = sprandn(m, n, alpha, rcond) в дополнение к вышеперечисленным условиям имеет также число обусловленности по отношению к операции обращения, близкое к значению rcond. Если rcond - вектор длины lr ≤ min(m, n), то матрица R имеет в качестве первых lr сингулярных чисел значения вектора rcond, а остальные сингулярные числа равны нулю. В этом случае матрица R генерируется с помощью матриц случайных плоских вращений, которые применяются к диагональной матрице с заданным спектром сингулярных чисел. Такие матрицы играют важную роль при анализе алгебраических и топологических структур.

Сопутствующие функции: SPRANDSYM.

SPRANDSYM **Случайная разреженная симметрическая матрица**

Синтаксис:

```
R = sprandsym(S)
R = sprandsym(n, alpha)
R = sprandsym(n, alpha, rcond)
R = sprandsym(n, alpha, rcond, kind)
```

Описание:

Матрица R = sprandsym(S) - случайная симметрическая матрица, главная диагональ и нижние поддиагонали которой имеют ту же структуру, которую имеет и матрица S; значения ее ненулевых элементов распределены по нормальному закону со средним, равным нулю, и дисперсией 1.

Матрица вида R = sprandsym(n, alpha) - это случайная симметрическая разреженная матрица, которая имеет приблизительно alpha×m× n ненулевых элементов, распределенных по нормальному закону, где alpha - коэффициент заполнения со значением в пределах $0 \le$ alpha ≤ 1 и каждый элемент сформирован в виде суммы нескольких нормально распределенных чисел.

Матрица R = sprandsym(n, alpha, rcond) имеет число обусловленности по отношению к операции обращения, равное rcond. Значения случайных элементов находятся в пределах [-1 1] и симметричны относительно нуля, однако закон распределения не является равномерным. Если rcond - вектор длины n, то матрица R имеет собственные значения, равные элементам вектора rcond; таким образом, если вектор rcond имеет положительные элементы, то матрица R является положительно определенной. В любом случае матрица R генерируется с помощью матриц случайных плоских вращений (матриц Якоби), которые применяются к диагональной матрице с заданным спектром собственных значений или числом обусловленности. Такие матрицы играют важную роль при анализе алгебраических и топологических структур.

Матрица R = sprandsym(n, alpha, rcond, kind) всегда является положительно определенной:

- если kind = 1, то матрица R формируется из положительно определенной диагональной матрицы с помощью матриц случайных плоских вращений с точно заданным числом обусловленности;
- если kind = 2, то матрица R формируется как смещенная сумма матриц внешних произведений; число обусловленности не соответствует заданному, но структура по сравнению с предыдущим случаем более компактна (участвует меньшее число поддиагоналей);
- если kind = 3, то предполагается форма R = sprandsym(S, alpha, rcond, 3) и матрица R имеет ту же структуру, которую имеет и матрица S, и число обусловленности приближенно равно 1/rcond, значение коэффициента заполнения alpha игнорируется.

Сопутствующие функции: SPRAND, SPRANDN.

Преобразование разреженных матриц

FIND — **Определение индексов ненулевых элементов**

Синтаксис:

k = find(x)	k = find(<*условие*>)
[i, j] = find(X)	[i, j] = find(<*условие* >)
[i, j, s] = find(X)	[i, j, s] = find(<*условие* >)

Описание:

Функция k = find(x) определяет индексы ненулевых элементов вектора x; если таких элементов нет, то результатом является пустой вектор. Если входом является матрица X, то при данном способе вызова функции find она рассматривается как вектор-столбец x(i), образованный объединением столбцов исходной матрицы.

Функция [i, j] = find(X) возвращает индексы строк и столбцов ненулевых элементов матрицы X; часто используется при работе с разреженными матрицами.

Функция [i, j, s] = find(X) возвращает индексы, а также вектор-столбец s ненулевых элементов матрицы X.

Если в качестве аргумента функции find используется <*условие*>, то первые две функции обладают теми же свойствами, а функция [i, j, s] = = find(<*условие*>) будет формировать в качестве вектора s вектор единиц вместо значений ненулевых элементов.

Пример:

Пусть

```
M = magic(3)
M =
     8     1     9
     3     5     7
     4     9     2
```

Тогда применение функции find в форме

```
[i, j, m] = find(M);  [i j m]
```

дает результат

```
ans =
     1     1     8
     2     1     3
     3     1     4
     1     2     1
     2     2     5
     3     2     9
     1     3     6
     2     3     7
     3     3     2
```

а в случае

```
[i, j, m] = find(M > 6);  [i j m]
```

получим

```
ans =
     1     1     1
     3     2     1
     2     3     1.
```

Сопутствующие функции: NONZEROS, SPARSE, RELOP (операции отношения).

FULL **Преобразование разреженной матрицы в полную**

Синтаксис:

A = full(S)

Описание:

Функция A = full(S) изменяет способ хранения матрицы в памяти компьютера; если исходная матрица A была полной, то функция full(A) возвращает A.

Пусть X - матрица размера m×n с nz = nnz(X) ненулевыми элементами. Тогда для размещения выхода функции full(X) требуется пространство для m * n действительных чисел, в то время как входная разреженная матрица sparse(X) требует пространства для размещения только nz действительных и nz + n целых чисел. Большинству компьютеров для хранения действительного числа нужно вдвое больше памяти, чем для целого. Для таких компьютеров хранение в форме sparse(X) будет экономичнее, чем в форме full(X), если коэффициент заполнения nnz/(m×n) < 2/3. Однако обработка разреженных матриц требует существенно большего числа операций, чем полных, так что коэффициент заполнения должен быть существенно меньше, чем 2/3, чтобы работа с разреженной матрицей оказалась более эффективной.

Пример:

Рассмотрим разреженную матрицу с коэффициентом заполнения около 2/3.

```
S = sparse(rand(200,200) < 2/3);
A = full(S); whos
```

Name (переменная)	*Size (размер массива)*	*Bytes (размер памяти)*	*Class (тип массива)*
A	200 by 200	320000	double array (logical)
S	200 by 200	318432	sparse array (logical)

Grand total is 66469 elements using 638432 bytes.

Общее количество элементов - 66469, использовано 638432 байт.

В этом случае функции sparse(S) и full(S) требуют приблизительно одинакового объема памяти для хранения элементов.

Сопутствующие функции: SPARSE.

SPCONVERT **Преобразование данных в ASCII-формате в массив разреженной структуры**

Синтаксис:

S = spconvert(D)

Описание:

Обычные функции load и save поддерживают работу с массивами разреженной структуры, поэтому нет необходимости вводить специальные

команды для загрузки и выгрузки разреженных массивов. Однако если внешний файл содержит данные о массиве разреженной структуры в ASCII-формате, то требуется преобразование этих данных во внутреннюю форму хранения. Предполагается, что внешний файл может быть организован в виде массива со структурой [i j s] или [i j r s], а число строк должно быть равно nnz или nnz + 1. Массив с тремя столбцами соответствует действительным элементам, а с четырьмя столбцами - комплексным. Последняя строка массива типа [m n 0] или [m n 0 0] может служить для задания размеров разреженной матрицы.

Функция spconvert применяется только для .mat- и ASCII-файлов. Если матрица D имеет разреженную структуру, то никаких преобразований не требуется.

Пример:

Допустим, что ASCII-файл uphill.dat содержит следующий массив данных:

```
1    1        1.000000000000000
1    2        0.500000000000000
2    2        0.333333333333333
1    3        0.333333333333333
2    3        0.250000000000000
3    3        0.200000000000000
1    4        0.250000000000000
2    4        0.200000000000000
3    4        0.166666666666667
4    4        0.142857142857143
4    4        0.000000000000000
```

Массив состоит из 11 строк.

Последовательность операторов

```
load uphill.dat
H = spconvert(uphill)
```

загружает данные и восстанавливает разреженную матрицу sparse(triu(hilb(4))) с учетом ошибок округления.

В данном случае последняя строка не является необходимой, поскольку размер матрицы был определен указанием ненулевого элемента (4, 4).

Сопутствующие функции: HTML-справка.

Работа с ненулевыми элементами

ISSPARSE **Проверка принадлежности к классу разреженных матриц**

Синтаксис:

```
k = issparse(X)
```

Описание:

Функция issparse(X) принимает значение 1, если матрица X является разреженной, и 0, если полной.

Поскольку большинство функций системы MATLAB работает корректно с обоими типами матриц, то на практике применение функции issparse для распознавания типа матрицы достаточно редко.

Пример:

```
if issparse(X)
  nrm = normest(X)
else
  nrm = norm(X)
end
```

Сопутствующие функции: FULL, SPARSE.

NNZ **Количество ненулевых элементов**

Синтаксис:

```
nz = nnz(S)
```

Описание:

Функция nnz(S) определяет количество ненулевых элементов в разреженной матрице S.

Коэффициент заполнения разреженной матрицы, который выводится на экран по команде whos, определяется по формуле nnz(S)/prod(size(S)).

Пример:

Матрица Уилкинсона - это трехдиагональная матрица порядка 21 с 20 ненулевыми элементами на каждой диагонали.

```
w = sparse(wilkinson(21));
nnz(w)
 ans =  60
```

Сопутствующие функции: FIND, ISA, NNZ, NZMAX, SIZE, WHOS.

NONZEROS **Формирование вектора ненулевых элементов**

Синтаксис:

```
v = nonzeros(S)
```

Описание:

Функция nonzeros(S) формирует вектор ненулевых элементов матрицы, выбирая их по столбцам. Эта функция аналогична функции [i, j, v] = find(X), но в отличие от последней формирует только третий выход v. При этом, как правило, выполняется условие

length(v) = nnz(S) ≤ nzmax(S) ≤ prod(size(S)).

Сопутствующие функции: NNZ, NZMAX, FIND, SIZE, WHOS, ISSPARSE.

NZMAX — **Количество ячеек памяти для размещения ненулевых элементов**

Синтаксис:

```
n = nzmax(S)
```

Описание:

Для полной матрицы F - это общее количество элементов, определяемое соотношением nzmax(F) = prod(size(F)); для разреженной - это количество ячеек для размещения ненулевых элементов. Как правило, функции nnz(S) и nzmax(S) дают одинаковые значения. Однако в случае операций над разреженными матрицами, в результате которых возникают новые ненулевые элементы, например операций умножения и LU-разложения, может быть выделено больше элементов памяти, чем это требуется при конкретном вычислении, то есть выполняется условие nnz(S) ≤ nzmax(S).

Сопутствующие функции: FIND, ISA, NNZ, NONZEROS, SIZE, WHOS.

SPALLOC — **Выделить пространство памяти для разреженной матрицы**

Синтаксис:

```
S = spalloc(m, n, nzmax)
```

Описание:

Функция S= spalloc(m, n, nzmax) создает массив для разреженной матрицы размера m×n с учетом того, что количество ненулевых элементов не превышает nzmax. Затем матрица может быть заполнена по столбцам. Функция spalloc(m, n, nzmax) равносильна форме функции sparse([ ], [ ], [ ], m, n, nzmax).

Пример:

Если известно, что матрица имеет максимум 3 ненулевых элемента на столбец, то наиболее эффективный способ ее формирования следующий:

```
S = spalloc(n,n,3*n);
for j = 1:n
  S(:, j) = [zeros(n-3, 1)' round(rand(3, 1))']';
end
full(S)
ans =
   0   0   0   0   0
   0   0   0   0   0
   1   1   0   1   0
   1   0   1   0   1
   1   0   1   1   1
```

Сопутствующие функции: SPARSE.

SPFUN **Вычисление функции только для ненулевых элементов**

Синтаксис:

```
f = spfun('fun', S)
```

Описание:

Функция spfun применяет заданную функцию только к ненулевым элементам разреженной матрицы. Имя fun - это имя М-файла, в котором задана вычисляемая функция и который в качестве входного аргумента использует матрицу S.

Пример:

Функция вида f = spfun('exp', S) вычисляет разреженную матрицу с тем же коэффициентом заполнения, который имеет и матрица S, если не учитывать появления элементов exp(-inf) → 0. В то же время функция exp(S) применяет функцию exp к каждому элементу матрицы S и нулевые элементы становятся единичными.

Пусть задана диагональная разреженная матрица размера 4×4:

```
S =
(1, 1) 1
(2, 2) 2
(3, 3) 3
(4, 4) 4
```

Функция f = spfun('exp', S) имеет тот же коэффициент заполнения, что и матрица S:

```
f =
(1, 1) 2.7183
(2, 2) 7.3891
(3, 3) 20.0855
(4, 4) 54.5982
```

В то же время exp(S) содержит единицы на месте нулей.

```
full(exp(S))
ans =
2.7183  1.0000  1.0000   1.0000
1.0000  7.3891  1.0000   1.0000
1.0000  1.0000 20.0855   1.0000
1.0000  1.0000  1.0000  54.5982
```

Сопутствующие функции: FEVAL.

SPONES **Формирование матрицы связности**

Синтаксис:

```
R = spones(S)
```

Описание:

Функция spones генерирует для заданной разреженной матрицы матрицу связности, которая имеет структуру матрицы S, в которой ненулевые элементы заменены на 1.

Примеры:

Функция c = sum(spones(S)) вычисляет количество ненулевых элементов каждого столбца.

Функция r = sum(spones(S'))' вычисляет количество ненулевых элементов каждой строки. При этом соблюдается условие sum(c) = sum(r) = nnz(S).

Сопутствующие функции: NNZ, SPALLOC, SPFUN.

Характеристики разреженной матрицы

NORMEST — **Оценка 2-нормы разреженной матрицы**

Синтаксис:

```
nrm = normest(S)
nrm = normest(S, tol)
[nrm, cnt] = normest(S)
```

Описание:

Эта функция изначально предназначалась для работы с разреженными матрицами, хотя она работает корректно и может быть полезна для оценки нормы и больших полных матриц.

Функция nrm = normest(S) вычисляет оценку 2-нормы матрицы S с относительной погрешностью 1e - 6 (по умолчанию).

Функция nrm = normest(S, tol) вычисляет оценку 2-нормы матрицы S с заданной относительной погрешностью tol.

Функция [nrm, cnt] = normest(S) позволяет определить количество использованных операций.

Алгоритм:

Каждая итерация включает операцию умножения матрицы S на ее транспонированную S'. Такие операции выполняются до тех пор, пока две последовательные оценки нормы не будут различаться в пределах заданной погрешности tol.

Пример:

Матрица W = wilkinson(101) является трехдиагональной матрицей размера 101×101; ее порядок вполне достаточен, чтобы процедура вычисления нормы norm(full(W)), которая включает операцию svd(full(W)), проявила некоторые негативные свойства. Время вычисления на компьютере SPARC 1 составляет 4.13 с и определяет точное значение нормы 50,7462. В то же время процедура normest(sparse(W)) требует только 1.56 с и дает оценку нормы 50,7458.

Сопутствующие функции: COND, CONDEST, NORM, SVD.

CONDEST **Оценка числа обусловленности матрицы по 1-норме**

Синтаксис:

```
c = condest(A)
[c, v] = condest(A)
```

Описание:

Функция c = condest(A) вычисляет оценку числа обусловленности матрицы A по отношению к операции обращения. Для этого используется метод Хейджера (Hager) в модификации Хаема (Higham) [1]. Вычисленное значение *c* является нижней оценкой числа обусловленности матрицы A по 1-норме.

Функция [c, v] = condest(A), кроме того, вычисляет такой вектор v, который удовлетворяет соотношению $\|\mathbf{Av}\| = \|\mathbf{A}\|\|\mathbf{v}\|/c$. Таким образом, для больших c вектор v является аппроксимацией нуль-вектора матрицы A.

Эта функция применима как к действительным, так и к комплексным полным и разреженным матрицам.

Пример:

Сравнительная оценка нижней границы числа обусловленности матрицы W=wilkinson(101) при применении функций cond, rcond, condest:

```
      cond(W)              1/rcond(W)          condest(sparse(W))
ans =  199.9410      ans = 154.5382          ans = 276.2827
```

Наилучшая оценка нижней границы вычисляется функцией condest.

Сопутствующие функции: COND, RCOND, NORMEST.

Ссылки:

1. Higham N. J. Fortran codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation //*ACM Trans. Math. Soft.* 1988. Vol.14. P. 381-396.

SPRANK **Структурный ранг разреженной матрицы**

Синтаксис:

```
r = sprank(S)
```

Описание:

Функция r = sprank(S) вычисляет структурный ранг разреженной матрицы S. Он известен также под названиями *максимальное сечение* (maximum transversal), *максимальное соответствие* (maximum assignment) и *максимальное совпадение* (maximum matching), в терминах теории графов.

Для величины структурного ранга всегда выполняется условие

```
sprank(A) ≥ rank(A);
```

более того, в точной арифметике с вероятностью 1 выполняется условие

```
sprank(A) == rank(sprandn(A)).
```

Пример:

Матрица размера 3 × 3 следующего вида:

$$A = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 0 & 0 \\ 3 & 0 & x \end{bmatrix}$$

имеет структурный ранг sprank(A) = 2 при любом значении x; что касается ранга этой матрицы, то rank(A) = 2 всюду, кроме точки x = 3/2, где он равен единице.

Сопутствующие функции: DMPERM, RANK.

Визуализация разреженных матриц

GPLOT — **Построение графа**

Синтаксис:

```
gplot(A, xy)
gplot(A, xy, 'LineSpec')
[X, Y] = gplot(A, xy)
```

Описание:

Функция gplot(A, xy) осуществляет построение графа, заданного массивами A и xy. *Граф* G - это множество узлов, пронумерованных от 1 до n, и множество соединений (ребер) между ними. Для построения графа требуется две матрицы: *матрица взаимосвязей* A и *матрица координат* размещения узлов xy. *Матрица взаимосвязей* A такова, что ее элемент A(i, j) = 1, если узел i связан с узлом j; *матрица координат* узлов размера n×2 определяет координаты узлов xy(i, :) = [x(i) y(i)].

Обращение в форме gplot(A, xy, 'LineSpec') позволяет изменить цвет и тип линий, соединяющих узлы; по умолчанию принято LineSpec = g-.

Функция вида [X, Y] = gplot(A, xy) формирует массив координат вершин графа, причем каждая пара вершин выделена значениями типа NaN. Эти векторы могут быть использованы в дальнейшем для построения графа.

Пример:

Рассмотрим матрицу bucky размера 60×60, которую связывают с описанием формы купола (Buckminster Fuller dome, отсюда название bucky). Ее граф соединений, описываемый разреженной матрицей связности размера 60×60, напоминает футбольный мяч или структуру молекулы углерода C_{60} (рис. 9.1).

```
[B, xyz] = bucky;
gplot(B, xyz(:, [1 2]), 'r-'),  axis square
```

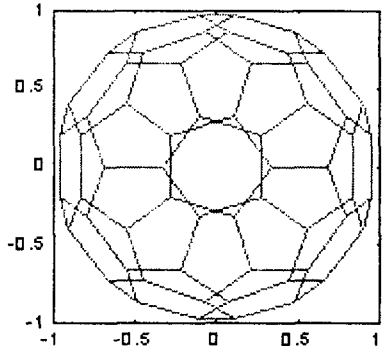


Рис. 9.1

Сопутствующие функции: PLOT, SPY, TREELAYOUT, TREEPLOT, UNMESH.

SPY **Визуализировать структуру разреженной матрицы**

Синтаксис:

```
spy(S)
spy(S, 'MarkSize')
spy(S, 'LineSpec')
spy(S, 'MarkSize', 'LineSpec')
spy(S, 'LineSpec', 'MarkSize')
```

Описание:

Функция spy(S) отображает на графике структуру произвольной матрицы S. Эта функция заменяет функцию format+, которая требует для вывода той же информации слишком много места на экране терминала.

Функция вида spy(S, 'MarkSize') позволяет управлять размером маркера, используемого для вывода точек; по умолчанию MarkSize равен 6 пикселям.

Функция вида spy(S, 'LineSpec') позволяет управлять типом и цветом выводимых точек по правилам указания свойств при использовании функции plot.

Функции spy(S, 'MarkSize', 'LineSpec') и spy(S, 'LineSpec', 'MarkSize') объединяют возможности двух вышеописанных функций.

Пример:

Отобразим в виде графика структуру матрицы bucky размера 60×60, используя размер маркера в 16 пикселей и красный цвет маркера в виде точки (рис. 9.2).

```
B = bucky;
spy(B, 16, '.r')
```

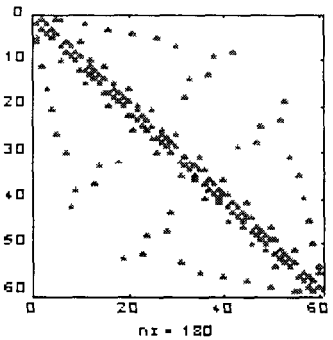


Рис. 9.2

Сопутствующие функции: FIND, SYMMMD, SYMRCM.

Алгоритмы упорядочения

RANDPERM **Формирование случайных перестановок**

Синтаксис:

```
p = randperm(n)
```

Описание:

Функция p = randperm(n) выполняет случайную перестановку целых чисел от 1 до n.

Сопутствующие функции: PERMUTE.

COLPERM **Упорядочение столбцов с учетом их разреженности**

Синтаксис:

```
j = colperm(S)
```

Описание:

Функция j = colperm(S) возвращает такой вектор перестановок j, что столбцы матрицы S(:, j) будут упорядочены по возрастанию числа ненулевых элементов. Эту процедуру целесообразно применять перед тем, как выполнить LU-разложение.

Если S - симметрическая матрица, то оказываются упорядоченными и строки и столбцы. Если к тому же матрица S является еще и положительно определенной, то такую процедуру перестановок целесообразно применять перед тем, как выполнить LL^T -разложение.

Алгоритм:

Алгоритм очень прост, он реализует сортировку столбцов по числу ненулевых элементов и выглядит следующим образом:

```
[I, j] = find(S);
[ignore, p] = sort(diff(find(diff([0 j' inf]))));
```

Примеры:

Рассмотрим матрицу вида

```
A = [ones(1, n); ones(n-1, 1) speye(n-1, n-1)]
```

при n = 4:

```
n=4;
A = [ones(1, n); ones(n-1, 1) speye(n-1, n-1)];
A = full(A)
A =
   1   1   1   1
   1   1   0   0
   1   0   1   0
   1   0   0   1
```

Ее LU-разложение представляет почти полную матрицу

```
lu(A)
ans =
   1   1   1   1
  -1  -1   0  -1
  -1   0  -1  -1
  -1  -1  -1   2
```

Функция упорядочения столбцов j = colperm(A) возвращает вектор перестановок j = [2:n 1], так что матрица A(j, j) имеет следующий вид:

```
j = colperm(A);
A(j, j)
ans =
   1   0   0   1
   0   1   0   1
   0   0   1   1
   1   1   1   1
```

а ее LU-разложение имеет такую же структуру ненулевых элементов:

```
lu(A(j,j))
ans =
   1   0   0   1
   0   1   0   1
   0   0   1   1
  -1  -1  -1  -2
```

Сопутствующие функции: CHOL, COLMMD, LU, SYMRCM.

DMPERM **DM-декомпозиция разреженной матрицы**

Синтаксис:

```
p = dmperm(S)
[p, q, r] = dmperm(S)
[p, q, r, s] = dmperm(S)
```

Описание:

Функция p = dmperm(S) возвращает вектор максимального соответствия; если исходная матрица S имеет полный столбцовый ранг, то матрица S(p, :) является квадратной с ненулевой диагональю. Матрица S(p, :) называется декомпозицией Далмейджа - Мендельсона (Dulmage - Mendelsohn) или DM-декомпозицией.

Если S - приводимая матрица, то есть линейная система Sx = b может быть решена приведением S к верхней блочной треугольной форме, то такое решение может быть найдено методом обратной подстановки.

Функция [p, q, r] = dmperm(S) определяет такую перестановку строк p и такую перестановку столбцов q для квадратной матрицы S, что S(p, q) - матрица в верхней треугольной форме. Третий выходной аргумент r - это целочисленный вектор, который описывает границы блоков, так что блок k матрицы S(p, q) имеет следующие границы: r(k) : r(k + 1) - 1.

Функция [p, q, r, s] = dmperm(S) определяет такую перестановку строк p и такую перестановку столбцов q для прямоугольной матрицы S, что S(p, q) есть матрица в верхней треугольной форме. Границы k-го блока определяются параметрами r и s согласно соотношению (r(k) : r(k + 1) -1, s(k) : s(k + 1) -1).

В терминах теории графов диагональные блоки соответствуют компонентам Холла смежного графа матрицы A.

Сопутствующие функции: SPRANK.

SYMRCM **RCM-упорядоченность**

Синтаксис:

r = symrcm(S)

Описание:

Функция r = symrcm(S) возвращает такой вектор упорядоченности для симметрической матрицы S, что S(r, r) будет концентрировать ненулевые элементы вблизи диагонали. Это хорошее упорядочение как для LU-, так и для LL^T-разложения матрицы, что позволяет отказаться от работы с элементами, удаленными от диагонали. Такое упорядочение называется упорядочением Катхилла - Макки (Cuthill - McKee).

Для действительных симметрических разреженных матриц собственные значения S(r, r) совпадают с собственными значениями S, но времени на вычисление eig(S(r, r)) будет затрачиваться существенно меньше, чем для eig(S).

Пример:

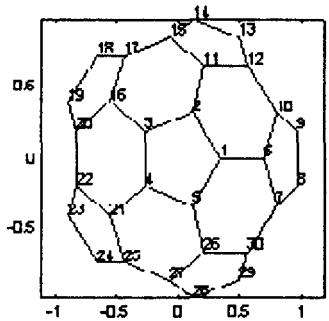
Интересный пример для анализа графов представляет модель, носящая название Bucky ball. Она представляет собой 60 точек, равномерно распределенных на сфере, так что расстояние до соседних точек одинаково; каждая точка связана только с тремя соседними. Модель Bucky ball ассоциируется с четырьмя физическими объектами:

геодезическим куполом (Buckminster Fuller dome);
моделью атома углерода C_{60};
усеченным 20-гранником (икосаэдром);
моделью футбольного мяча.

Матрица связности для модели Bucky ball - это симметрическая матрица размера 60×60 с тремя ненулевыми элементами в каждой строке и каждом столбце, так что общее количество ненулевых элементов 180.

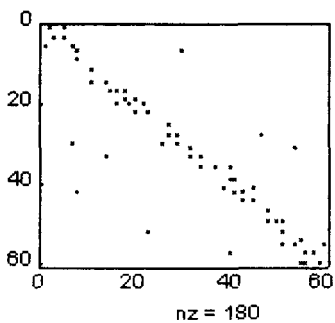
Ее граф имеет 60 вершин, которые пронумерованы так, что половина из них находится в одной, а половина в другой полусфере (рис. 9.3, *а*), и эти половины соединены вместе. Такая нумерация приводит к структуре матрицы, изображенной на рис. 9.3, *б*.

```
[B, V] = bucky;
k = 1:30;
gplot(B(k, k),V(k, [1 2])), axis equal
for j=1:30
  text(V(j, 1), V(j, 2)+0.04, int2str(j));
end
```

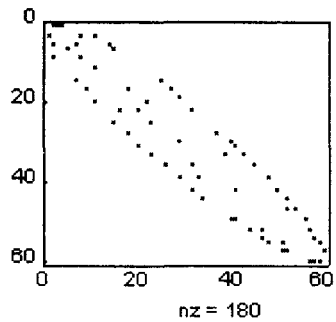


а

```
figure(2), spy(B)
```



б



в

Рис. 9.3

Применяя RCM-упорядочение

```
p = symrcm(B);
R = B(p, p);
spy(R)
```

получим матрицу с узкой полосой вблизи диагонали (рис. 9.3, *в*), ширина которой может быть вычислена следующим образом:

```
[i, j] = find(B);
bw = max(i - j) + 1
```

Ширина полосы матрицы A равна 35, а матрицы R - 12.

Сопутствующие функции: COLMMD, COLPERM, SYMMD.

Ссылки:

1. George A., Liu J. *Computer Solution of Large Sparse Positive Definite Systems*. N. Y.: Prentice-Hall, 1981.
2. Gilbert J. R., Moler C., Schreiber R. Sparse Matrices in MATLAB: Design and Implementation//*SIAM Journal on Matrix Analysis and Applications*. 1992. Vol. 13. P. 333-356.

COLMMD — Упорядочение по разреженности столбцов

Синтаксис:

```
p = colmmd(S)
```

Описание:

Функция p = colmmd(S) возвращает такой вектор упорядоченности столбцов для несимметрической матрицы S, что S(:, p) будет иметь более разреженное LU-разложение, чем S.

Такое упорядочение автоматически применяется системой MATLAB при выполнении операций обращения \ и / при решении систем линейных уравнений с разреженными матрицами.

Алгоритм:

Алгоритм упорядочения по разреженности столбцов для симметрических матриц описан в обзоре Джорджа (George) и Лиу (Liu) [1]. Для несимметрических матриц аналогичный алгоритм разработан заново и описан в работе Гилберта (Gilbert), Моулера (Moler) и Шрайбера (Schriber) [2]. Он напоминает прежний алгоритм для матрицы A' * A, но в действительности такая матрица не формируется.

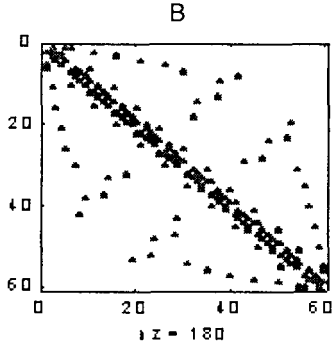
На каждом шагу алгоритма из оставшихся выбирается вершина низшего порядка, то есть столбец, имеющий минимальное количество ненулевых элементов, удаляет эту вершину и строит новый граф, объединяя строки. После n шагов все столбцы оказываются удаленными и перестановка завершается. Для ускорения этого процесса используются специальные приемы для одновременного выполнения нескольких шагов.

Пример:

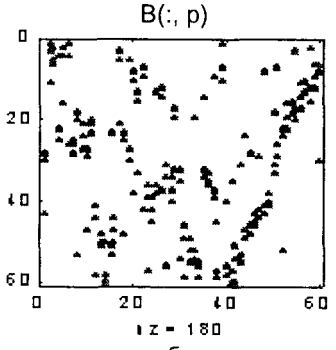
Вновь обратимся к матрице bucky.

```
[B, V]= bucky;
p = colmmd(B);
spy(B, 16, 'r')
spy(B(:, p), 16, 'r')
```

Образ матрицы B показан на рис. 9.4, *а*, из которого видно, что это разреженная матрица, элементы которой сосредоточены на диагонали и четырех полосах. Упорядочение по разреженности столбцов с использованием функции colmmd разрушает эту структуру (рис. 9.4, *б*)



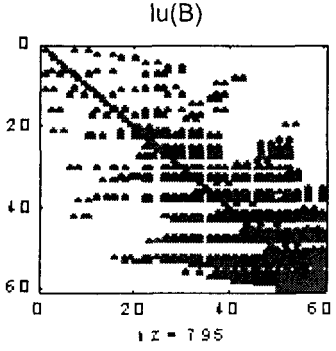
а



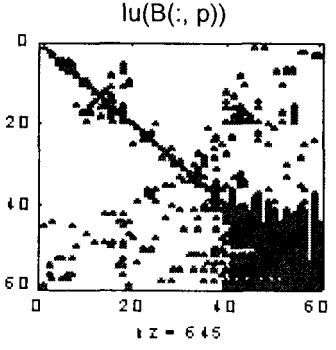
б

Сравнивая LU-разложения матрицы B (рис. 9.4, *в*) и переупорядоченной матрицы B(:, p) (рис. 9.4, *г*), можно увидеть, что количество ненулевых элементов в них равно соответственно 795 и 645.

```
spy(lu(B), 16, 'r')
spy(lu(B(:, p)), 16, 'r')
```



в



г

Рис. 9.4

Сопутствующие функции: COLPERM, LU, SPPARMS, SYMMMD, SYMRCM.

Ссылки:

1. George A., Liu J. The evolution of the minimum degree ordering algorithm. *//SIAM Review.* 1989. Vol. 31. P. 1-19.
2. Gilbert J. R., Moler C., Schreiber R. Sparse Matrices in MATLAB: Design and Implementation. *//SIAM Journal on Matrix Analysis and Applications.* 1992. Vol. 13. P. 333-356.

SYMMMD **Симметрическая упорядоченность**

Синтаксис:

```
p = symmmd(S)
```

Описание:

Функция p = symmmd(S) возвращает такой вектор упорядоченности столбцов для симметрической положительно определенной матрицы S, что S(:, p) будет иметь более разреженное LL^T-разложение, чем S.

Такое упорядочение автоматически применяется системой MATLAB при выполнении операций обращения \ и / при решении линейных систем с разреженными матрицами.

Алгоритм:

Алгоритм упорядочения для симметрических матриц основан на алгоритме упорядочения по разреженности столбцов. Фактически функция symmmd формирует матрицу K с такой структурой ненулевых элементов, что симметрическая матрица K' * K имеет такую же структуру ненулевых элементов, как и исходная матрица S, а затем вызывается функция colmmd для K.

Пример:

Сравним два алгоритма упорядочения, реализованные в виде функций symrcm и symmmd, которые предшествуют LL^T-разложению (разложение Холецкого) матрицы bucky размера 60×60, которая описывает граф связности bucky (рис. 9.5, *а)* и имеет структуру (рис. 9.5, *б*).

```
[B, V] = bucky; B = B + 4*speye(60);
gplot(B, V(:, [1 2])), axis equal
spy(B, 6, 'or')
```

Хотя эта задача и не является очень сложной, тем не менее поведение алгоритмов упорядочения является типичным. Применение функции symrcm порождает ленточную матрицу (рис. 9.5, *в*), которая после LL^T-разложения оказывается целиком заполненной внутри ленты (рис. 9.5, *д*).

```
p = symrcm(B);
spy(B(p, p), 16, 'r')
spy(chol(B(p, p)), 16, 'r')
```

Применение функции symmmd порождает ленточную матрицу с крупными блоками нулевых элементов (рис. 9.5, г), которые не заполняются и в процессе LL^T-разложения (рис. 9.5, *е*). Следовательно, алгоритм симметрической упорядоченности требует меньшего времени и объема памяти в процессе LL^T-разложения.

```
p = symmmd(B);
spy(B(p, p), 16, 'r')
spy(chol(B(p, p)), 16, 'r')
```

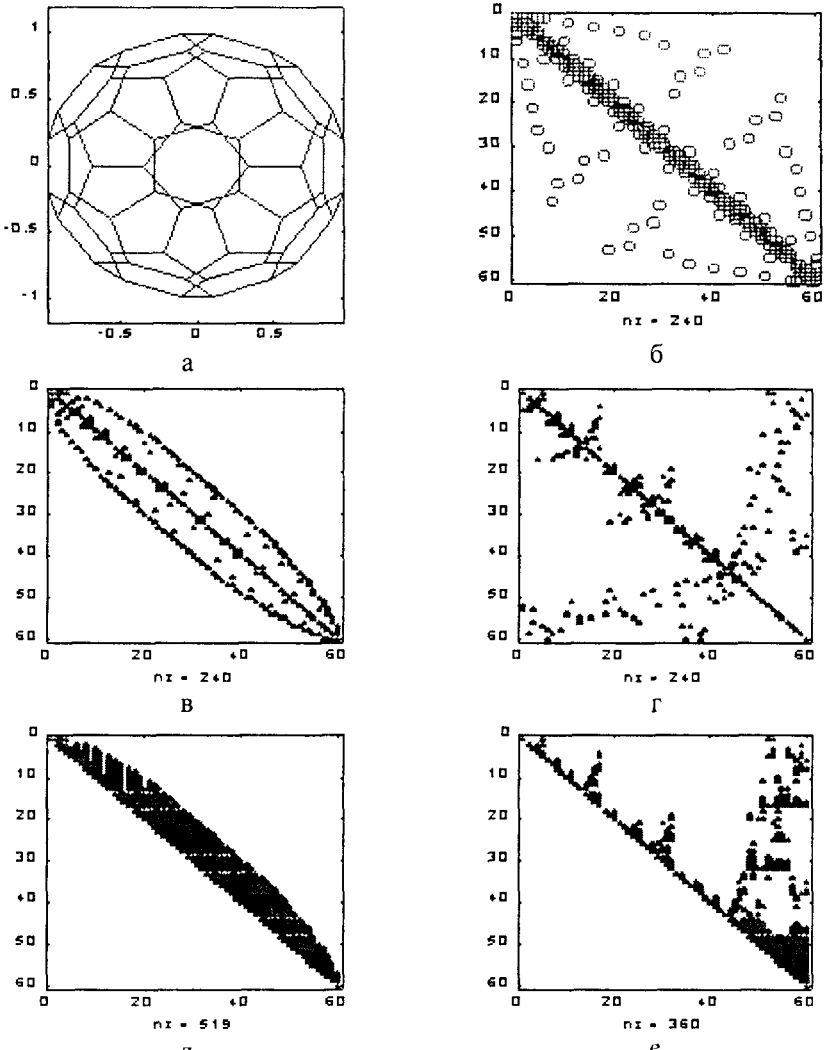


Рис. 9.5

Сопутствующие функции: COLMMD, COLPERM, SYMRCM.

Ссылки:

1. Gilbert J. R., Moler C., Schreiber R. Sparse Matrices in MATLAB: Design and Implementation//*SIAM Journal on Matrix Analysis and Applications*. 1992. Vol. 13. P. 333-356.

Алгоритмы линейной алгебры

Факторизация разреженных матриц

CHOLINC — **Неполное разложение Холецкого**

Синтаксис:

```
R = cholinc(S, droptol)      R = cholinc(S, '0')      R = cholinc(S, 'inf')
R = cholinc(S, options)    [R, p] = cholinc(S, '0')  [R, p] = cholinc(S, 'inf')
```

Описание:

Оператор cholinc реализует два варианта неполного разложения Холецкого: с заданным порогом разреженности и с нулевым уровнем расширения, которые могут быть использованы для предварительной обработки при решении итеративными методами больших систем линейных уравнений с симметрическими положительно определенными матрицами.

Функция R = cholinc(S, droptol) выполняет неполное разложение Холецкого для разреженной матрицы S с заданным порогом разреженности droptol. *Порог разреженности* - это минимальное значение элемента матрицы, при котором он приравнивается нулю.

Функция R = cholinc(S, options) позволяет указать дополнительные опции в массиве записей с тремя полями:

Поле	*Назначение*
options.droptol	Порог разреженности
options.michol	Модифицированное разложение Холецкого
options.rdiag	Заменить нули на диагонали матрицы R

При установке следует указывать только те поля, которые необходимы.

Эти поля имеют следующие спецификации:

Параметр droptol - это неотрицательное число, которое используется в качестве порога разреженности при реализации неполного разложения Холецкого. Это разложение основано на неполном LU-разложении без выбора ведущего элемента и последующем масштабировании строк верхней треугольной матрицы U, используя в качестве масштабирующего множителя квадратный корень диагонального элемента. Поскольку ненулевые значения U(i, j) ограничены снизу величиной droptol*norm(S(:, j)), то ненулевые значения R(i, j) ограничены снизу величиной droptol*norm(S(:, j))/R(i, i), определямой как *локальный порог разреженности* (localdroptol) для разложения Холецкого. При значении droptol, равном 0, и по умолчанию выполняется полное разложение Холецкого.

Параметр michol позволяет выбрать либо модифицированное (michol = 1), либо немодифицированное (michol = 0 и по умолчанию) разложение Холецкого.

Параметр rdiag принимает значение 0 или 1 (по умолчанию 0). Если rdiag равно 1, то все нулевые диагональные элементы верхней треугольной матрицы R заменяются значением sqrt(localdroptol), чтобы избежать сингулярности.

Функция R = cholinc(S, '0') выполняет неполное разложение Холецкого для разреженной симметрической положительно определенной матрицы S с нулевым уровнем расширения, то есть без увеличения количества ненулевых элементов. Верхняя треугольная матрица R имеет в основном ту же структуру, что и triu(S), хотя некоторые элементы S, не равные нулю с точностью до ошибок округления, могут иметь нулевые значения в матрице R. Следует заметить, что положительная определенность исходной матрицы не гарантирует существования разложения с требуемыми свойствами. Когда факторизация невозможна, формируется сообщение об ошибке. Если факторизация успешна, матрицы R'*R и S имеют одинаковую структуру разреженности.

Функция [R, p] = cholinc(S, '0') с двумя выходными аргументами не приводит к формированию сообщения об ошибке: если R существует, то p равно 0; но если факторизация невозможна, то p - положительное целое число, а R - верхняя треугольная матрица размера q×n, где q = p-1, так что структура R и верхней треугольной подматрицы размера q×n одинаковы. Матрицы R'*R и подматрица S размера q×q имеют одинаковую структуру разреженности.

Функция R = cholinc(S, 'inf') выполняет разложение Холецкого для положительно полуопределенных матриц, когда в процессе факторизации появляются нулевые ведущие элементы. В этом случае соответствующему диагональному элементу верхней треугольной матрицы R присваивается значение inf, а остальным элементам строки - нулевые значения. Таким образом, соответствующий элемент в векторе решения принудительно приравнивается нулю. В случае отрицательного значения ведущего элемента формируется сообщение об ошибке.

Функция [R, p] = cholinc(S, 'inf') с двумя выходными аргументами не приводит к формированию сообщения об ошибке: если S положительно полуопределенная матрица, то p равно 0; в иных случаях p - положительное целое число, а R - верхняя треугольная матрица размера (p-1)×n.

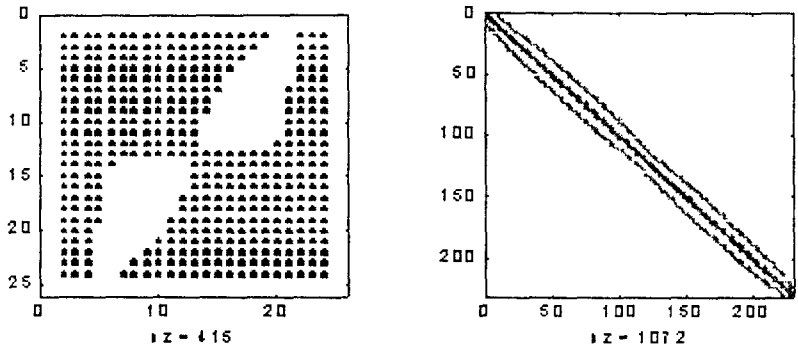
Ограничения:

Функция применима только к квадратным разреженным матрицам; при обращении к функциям cholinc(S, '0') и cholinc(S, 'inf') матрица S должна быть действительной.

Примеры:

Рассмотрим симметрическую положительно определенную матрицу S, связанную с пятиточечной аппроксимацией лапласиана на сетке размера n×n, задаваемой функцией numgrid.

```
n = 25; S = delsq(numgrid('B', n));
spy(numgrid('B', n));
spy(S, 'r')
```



а б

Рис. 9.6

```
nnz(S)
ans = 1072
nnz(chol(S))
ans =  2655
nnz(cholinc(S, 1e-3))
ans =       2109
```

Таким образом, исходная матрица S имеет 1072 ненулевых элемента, ее полное разложение Холецкого - 2655, а неполное разложение с порогом разреженности 1e-3 - 2109 элементов.

Рассмотрим функцию R = cholinc(S, '0'), которая выполняет неполное разложение Холецкого для разреженной матрицы S с нулевым уровнем расширения, то есть без увеличения количества ненулевых элементов.

```
S = delsq(numgrid('N', 10));
R = cholinc(S, '0');
isequal(spones(R), spones(triu(S)))
ans =  1
```

Матрица

```
D = (R'*R) .* spones(S) - S;
```

имеет элементы порядка eps.

Определим один из диагональных элементов равным 0:

```
S(8, 8) = 0;
[R2, p] = cholinc(S, '0');
```

В этом случае p не равно 0 и существует такая верхняя треугольная матрица R размера q×n, где q = p-1, что структура R и верхней треугольной подматрицы размера q×n матрицы S одинаковы.

```
p
p =  8
size(R2)
ans =  7   64
isequal(spones(R2), spones(triu(S(1:p-1, :))))
ans =  1
```

Пусть задана положительно полуопределенная разреженная матрица

```
S = sparse([ 1   0   3   0;
             0  25   0  30;
             3   0   9   0;
             0  30   0 661 ]);
```

Ее разложение Холецкого имеет нулевой ведущий элемент в строке 3:

```
[R, p] = chol(S);
p
p =  3
```

Тогда неполное разложение Холецкого формирует верхнюю треугольную матрицу Rinf, третий диагональный элемент которой равен inf:

```
Rinf = cholinc(S, 'inf');
full(Rinf)
ans =
   1   0   3   0
   0   5   0   6
   0   0 Inf   0
   0   0   0  25
```

Сопутствующие функции: CHOL, LUINC, PCG.

Ссылки:

1. Saad Y. *Iterative Methods for Sparse Linear Systems.* Boston, MA: PWS Publishing Company, 1996.
2. Zhang Y. *Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment:* Technical Report TR96-01. Baltimore: University of Maryland Baltimore County, 1996.

LUINC — Неполное LU-разложение

Синтаксис:

```
LU = luinc(S, '0')          LU =  luinc(S, droptol      LU = luinc(S, options)
[L, U] = luinc(S, '0')      [L, U] = luinc(S, droptol)  [L, U] = luinc(S, options)
[L, U, P] = luinc(S, '0')   [L, U, P] = luinc(S, droptol) [L, U, P] = luinc(S, droptol)
```

Описание:

Оператор luinc реализует два варианта неполного LU-разложения: с заданным порогом разреженности и с нулевым уровнем расширения, которые могут быть использованы для предварительной обработки при решении итеративными методами больших систем линейных уравнений с произвольными матрицами. Оператор luinc реализует разложение исходной матрицы S на нижнюю треугольную L, верхнюю треугольную U и матрицу перестановок P.

Функция LU = luinc(S, '0') выполняет неполное LU-разложение для разреженной квадратной матрицы S с нулевым уровнем расширения. Треугольные множители имеют ту же структуру разреженности, как и исходная матрица S, а их произведение согласуется с полной матрицей с учетом перестановок строк. Оператор LU = luinc(S, '0') возвращает нижний и верхний

треугольные множители в матрице LU либо в исходной матрице S, если выходной массив не указан. Матрица перестановок потеряна. Выполняется условие nnz(luinc(S, '0') = nnz(S) с точностью до тех элементов, которые равны 0 в пределах ошибок округления.

Функция [L, U] = luinc(S, '0') возвращает верхнюю треугольную матрицу U и произведение матрицы перестановок и нижнего треугольного множителя в матрице L. Структуры разреженности матриц L, U и S несопоставимы, но количество ненулевых элементов с точностью до ошибок округления удовлетворяет условию nnz(L) + nnz(U) = nnz(S) + n, где n - порядок матрицы S. Произведение L*U согласуется со структурой разреженности матрицы S, так что матрица (L*U).*spones(S) - S имеет элементы порядка eps.

Функция [L, U, P] = luinc(S, '0') возвращает нижнюю треугольную матрицу L, верхнюю треугольную матрицу U и матрицу перестановок P. Матрица L имеет такую же структуру разреженности, как и матрица S с учетом перестановок строк, так что выполняется условие spones(L) = spones(tril(P*S)) с точностью до возможного исключения единиц на диагонали L, где P*S может быть равно 0, и нулей в L, где P*S может быть не равно 0 с точностью до ошибок округления. Матрица U согласуется со структурой разреженности верхней треугольной матрицы P*S, так что spones(U) = spones(triu(P*S)) с точностью до возможного исключения нулей в U, где P*S может быть не равно 0 с точностью до ошибок округления. Произведение L*U согласуется со структурой разреженности матрицы P*S, так что матрица (L*U).*spones(P*S) - P*S имеет элементы порядка eps.

Функция LU = luinc(S, droptol) выполняет неполное LU-разложение для произвольной разреженной матрицы S с заданным порогом разреженности droptol. *Порог разреженности* - это минимальное значение элемента матрицы, при котором он приравнивается нулю. Неполное LU-разложение с заданным порогом разреженности является непрерывной аппроксимацией полного LU-разложения; по мере уменьшения порога разреженности аппроксимация становится все точнее и при пороге, равном 0, переходит в точное LU-разложение. Для каждого столбца треугольного множителя элементы, не превышающие локального порога разреженности, исключаются. *Локальный порог разреженности* - это произведение порога разреженности и нормы соответствующего столбца исходной матрицы localdroptol = droptol*norm(S(:, j). Исключение из этого правила составляют только диагональные элементы, которые не должны обнуляться, чтобы избежать сингулярности.

Функция [L, U] = luinc(S,droptol) возвращает верхнюю треугольную матрицу U и произведение матрицы перестановок и нижнего треугольного множителя в матрице L.

Функция [L, U, P] = luinc(S, droptol) возвращает нижнюю треугольную матрицу L, верхнюю треугольную матрицу U и матрицу перестановок P. Произведение L*U является аппроксимацией матрицы P*S.

Функция LU = luinc(S, options) позволяет указать дополнительные опции в массиве записей с четырьмя полями:

Поле	*Назначение*
options.droptol	Порог разреженности
options.milu	Модифицированное LU-разложение
options.udiag	Заменить нули на диагонали матрицы U
options.thresh	Порог для выбора ведущего элемента

При установке следует указывать только те поля, которые необходимы. Эти поля имеют следующие спецификации:

Параметр droptol - это неотрицательное число, которое используется в качестве порога разреженности при реализации неполного LU-разложения. Это разложение основано на LU-разложении с использованием локального порога разреженности. При значении droptol, равном 0, выполняется полное LU-разложение.

Параметр milu позволяет выбрать либо модифицированное (milu = 1), либо немодифицированное (milu = 0 и по умолчанию) LU-разложение. Модификация заключается в том, что малые элементы вновь формируемого столбца не отбрасываются, а вычитаются из диагонали верхнего треугольного множителя U.

Параметр udiag принимает значение 0 или 1 (по умолчанию 0). Если udiag равно 1, то все нулевые диагональные элементы верхней треугольной матрицы R заменяются значением локального порога разреженности localdroptol, чтобы избежать сингулярности.

Параметр thresh позволяет задать порог при выборе ведущего элемента; он принимает значение в диапазоне от 0 до 1 (по умолчанию 1). Если диагональный элемент по модулю меньше, чем произведение внедиагонального элемента на множитель thresh, то роль ведущего переходит к внедиагональному элементу. При значении thresh, равном 0, в качестве ведущего всегда выбирается диагональный элемент.

Функция [L, U] = luinc(S, options) возвращает верхнюю треугольную матрицу U и произведение матрицы перестановок и нижнего треугольного множителя в матрице L.

Функция [L, U, P] = luinc(S, options) возвращает нижнюю треугольную матрицу L, верхнюю треугольную матрицу U и матрицу перестановок P. Произведение L*U является аппроксимацией матрицы P*S.

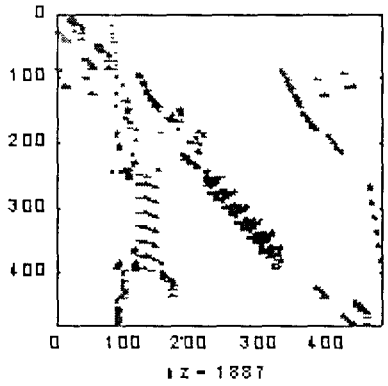
Ограничения:

Функция применима только к квадратным разреженным матрицам.

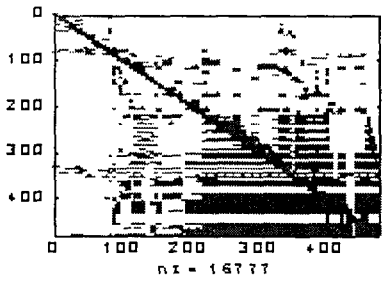
Примеры:

Рассмотрим тестовую матрицу west0479 размера 479×479.

```
load west0479
S = west0479;
spy(S, 10, 'r'), spy(lu(S), 10, 'r')
```



а



б

Рис. 9.7

```
nnz(S)
ans =  1887
nnz(lu(S))
ans =  16777
nnz(luinc(S, 1e-6))
ans =  10311
```

Таким образом, исходная матрица S имеет 1887 ненулевых элементов, ее полное LU-разложение - 16777, а неполное разложение с порогом разреженности 1e-6 - 10311 элементов.

Рассмотрим функцию luinc(S, '0'), которая выполняет неполное разложение Холецкого для разреженной матрицы S с нулевым уровнем расширения, то есть без увеличения количества ненулевых элементов.

```
[L, U, P] = luinc(S, '0');
isequal(spones(U), spones(triu(P*S)))
ans = 1
nnz(spones(L) ~= spones(tril(P*S)))
ans = 74
D = (L*U) .* spones(P*S) - P*S; normest(D)
ans =  7.10889595793317e-015
```

Структуры разреженности матриц L и P*S различаются в 74 элементах: 73 элемента на диагонали, где в матрице L стоят единицы, а в матрице P*S - нули, а также в позиции (206, 131), где в матрице L стоит 0, а в матрице P*S - (-1). Матрица имеет элементы порядка eps.

Сопутствующие функции: LU, CHOLINC, BICG.

Ссылки:

1. Saad Y. *Iterative Methods for Sparse Linear Systems.* Boston, MA: PWS Publishing Company, 1996.

Решение систем линейных уравнений

Решение систем линейных уравнений реализуется двумя классами методов - *прямыми* и *итерационными*. *Прямые* методы находят точные решения системы за конечное число шагов; они обычно используют различные варианты метода исключения Гаусса и выполняют факторизацию матрицы в виде LU-разложения или разложения Холецкого. *Итерационные* методы находят за конечное число шагов только приближенное решение и работают с матрицей в целом, а не с отдельными ее элементами.

Прямые методы работают достаточно быстро и широко применяются на практике, если достаточны объемы памяти для их реализации. Прямые методы реализованы в системе MATLAB как встроенные функции и обладают наивысшей эффективностью для матриц общего вида. Итерационные методы обычно применяются к специальным классам уравнений, как правило больших порядков. Они реализуются в системе MATLAB в виде M-файлов и могут включать в качестве вспомогательных операций алгоритмы прямых методов.

PCG **Предобусловленный метод сопряженных градиентов**

Синтаксис:

```
x = pcg(A, b)
x = pcg(A, b, tol)
x = pcg(A, b, tol, maxit)
x = pcg(A, b, tol, maxit, M)
x = pcg(A, b, tol, maxit, M1, M2)
x = pcg(A, b, tol, maxit, M1, M2, x0)
[x, flag] = pcg(...)
[x, flag, relres] = pcg(...)
[x, flag, relres, iter] = pcg(...)
[x, flag, relres, iter, resvec] = pcg(...)
```

Описание:

Функция x = pcg(A, b) реализует попытку решения системы линейных уравнений A*x = b относительно неизвестного вектора x. Матрица A порядка n предполагается симметрической и положительно определенной. Решение стартует с начального приближения, по умолчанию соответствующего нулевому вектору длины n, и итерации продолжаются до тех пор, пока метод сойдется, либо произойдет останов вычислений, либо будет превышено максимальное число итераций. Сходимость достигнута, когда относительная погрешность norm(b - A*x)/norm(b) становится меньше или равной заданной погрешности вычислений (по умолчанию 1e-6). Максимальное число итераций принимается по умолчанию равным min(n, 20). Никакой предварительной обработки матрицы не выполняется.

Функция x = pcg(A, b, tol) позволяет задать погрешность вычисления tol.

Функция x = pcg(A, b, tol, maxit) позволяет изменить значение максимального числа итераций.

Функции x = pcg(A, b, tol, maxit, M) и x = pcg(A, b, tol, maxit, M1, M2) используют вспомогательные матрицы M либо M=M1*M2, для того чтобы преобразовать исходную систему в вид M*y = b, где y = inv(M)*A*x, и достичь более высокой эффективности решения. Поскольку решения системы вида M*y = b ищутся с помощью встроенного решателя, разумно выполнить факторизацию матрицы M, применяя разложение Холецкого. Тогда решение будет использовать два оператора:

```
R = chol(M);
x = pcg(A, b, tol, maxit, R', R).
```

В этом случае матрица M должна быть симметрической и положительно определенной.

Если в качестве матриц M, M1, M2 будет указана пустая матрица, то в этом случае она интерпретируется как единичная, то есть никаких вспомогательных операций не выполняется.

Функция x = pcg(A, b, tol, maxit, M1, M2, x0) позволяет изменить вектор начальной оценки решения x0. Если в качестве вектора x0 будет указан пустой массив, то в этом случае он интерпретируется как нулевой, то есть используемый по умолчанию.

Для всех вышеописанных форм вызова возвращается решение x. Если решение получено, то оно все равно сопровождается сообщением о величине относительной погрешности решения и количестве использованных итераций. Если превышено максимальное число итераций или процедура прекратила счет, выводится соответствующее предупреждающее сообщение.

Функция [x, flag] = pcg(...) возвращает решение и флаг, указывающий на характер сходимости решения:

Значение флага	*Информация о сходимости*
0	Процедура pcg сходится к решению при заданной точности tol в пределах числа итераций maxit
1	Процедура pcg превысила максимальное число итераций maxit, не достигнув сходимости
2	Одна из вспомогательных систем вида M*y = r плохо обусловлена и не возвращает приемлемого решения из встроенного решателя
3	Процедура остановлена, поскольку две последовательные оценки решения оказались одинаковыми
4	Одна из величин в процессе выполнения процедуры вышла за пределы допустимых чисел в компьютере

Если значение флага не равно 0, то возвращается то решение, которое соответствует минимальной относительной погрешности для всех выполненных шагов. При указании в качестве выходного параметра флага никаких сообщений не выводится.

Функция [x, flag, relres] = pcg(...) возвращает также относительную погрешность решения norm(b - A*x)/norm(b). Если флаг равен 0, то relres ≤ tol.

Функция [x, flag, relres, iter] = pcg(...) возвращает номер итерации, на которой было получено решение; этот номер удовлетворяет условию 0 ≤ iter ≤ maxit.

Функция [x, flag, relres, iter, resvec] = pcg(...) возвращает нормы невязок на каждой итерации в виде вектора resvec начиная с невязки resvec(1) = norm(b - A*x0). Если флаг равен 0, то вектор resvec имеет длину iter+1 и значение последнего компонента должно удовлетворять условию resvec(end) ≤ tol*norm(b).

Примеры:

Рассмотрим систему линейных уравнений A*x = b, связанную с пятиточечной аппроксимацией лапласиана на сетке размера n×n.

```
n = 25; A = delsq(numgrid('C', n));
b = ones(length(A), 1);
[x, flag] = pcg(A, b);  flag
flag =    1
```

Значение флага равно 1, что означает отсутствие сходимости при точности 1e - 6 (по умолчанию) в пределах 20 итераций (по умолчанию).

Выполним факторизацию матрицы A с помощью неполного разложения Холецкого, установим относительную погрешность решения 1e - 8 и максимальное число итераций - 10:

```
R = cholinc(A, 1e-3);
[x2, flag2, relres2, iter2, resvec2] = pcg(A , b, 1e-8, 10, R', R);
flag2 =   0
relres2
relres2 =  1.2059e-009
iter2
iter2 =   6
resvec2
resvec2 =
      20.761
      1.5687
    0.060282
   0.0014501
  6.2128e-005
  1.4513e-006
  2.5036e-008
```

Значение флага flag2 равно 0, что означает сходимость с точностью relres2, равной 1.2e - 009, на 6 итерации (параметр iter2). Это достигнуто за счет предварительной факторизации матрицы с порогом разреженности 1e - 3. Значение resvec2(1) равно norm(b), а значение resvec2(7) равно norm(b - A*x2). Теперь можно построить относительную погрешность решения на каждой итерации (рис. 9.8), используя функцию:

```
semilogy(0:iter2, resvec2/norm(b), '-o'), grid
xlabel('Номер итерации', 'FontName', 'Arial Cyr', 'FontSize', 12)
ylabel('||b - Ax||/||b||', 'FontSize', 12)
```

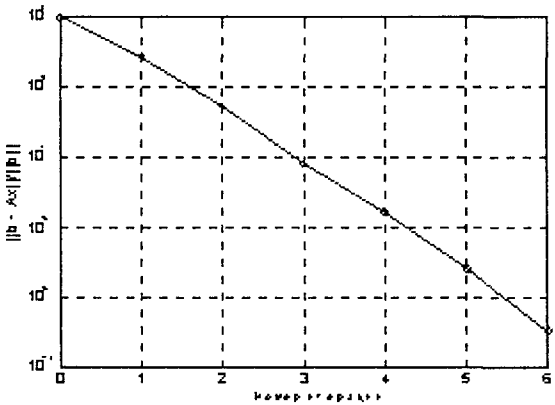


Рис. 9.8

Сопутствующие функции: BICG, BICGSTAB, CGS, CHOLINC, GMRES, QMR, \ .

Ссылки:

1. Barrett R., et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1994.

BICG **Метод бисопряженных градиентов**

Синтаксис:

```
x = bicg(A, b)
x = bicg(A, b, tol)
x = bicg(A, b, tol, maxit)
x = bicg(A, b, tol, maxit, M)
x = bicg(A, b, tol, maxit, M1, M2)
x = bicg(A, b, tol, maxit, M1, M2, x0)
[x, flag] = bicg(...)
[x, flag, relres] = bicg(...)
[x, flag, relres, iter] = bicg(...)
[x, flag, relres, iter, resvec] = bicg(...)
```

Описание:

Функция x = bicg(A, b) реализует попытку решения системы линейных уравнений A*x = b относительно неизвестного вектора x. Матрица A порядка n предполагается произвольной квадратной. Решение стартует с начального приближения, по умолчанию соответствующего нулевому вектору длины n, и итерации продолжаются до тех пор, пока метод сойдется, либо произойдет

останов вычислений, либо будет превышено максимальное число итераций. Сходимость достигнута, когда относительная погрешность norm(b - A*x)/norm(b) становится меньше или равной заданной погрешности вычислений (по умолчанию 1e-6). Максимальное число итераций принимается по умолчанию равным min(n, 20). Никакой предварительной обработки матрицы не выполняется.

Функция x = bicg(A, b, tol) позволяет задать погрешность вычисления tol.

Функция x = bicg(A, b, tol, maxit) позволяет изменить значение максимального числа итераций.

Функции x = bicg(A, b, tol, maxit, M) и x = bicg(A, b, tol, maxit, M1, M2) используют вспомогательные матрицы M либо M=M1*M2, для того чтобы преобразовать исходную систему в вид M*y = b, где y = inv(M)*A*x, и достичь более высокой эффективности решения. Поскольку решения системы вида M*y = b ищутся с помощью встроенного решателя, разумно выполнить факторизацию матрицы M, применяя LU-разложение. Тогда решение будет использовать два оператора:

R = lu(M);

x = bicg(A, b, tol, maxit, R', R).

Если в качестве матриц M, M1, M2 будет указана пустая матрица, то в этом случае она интерпретируется как единичная, то есть никаких вспомогательных операций не выполняется.

Функция x = bicg(A, b, tol, maxit, M1, M2, x0) позволяет изменить вектор начальной оценки решения x0. Если в качестве вектора x0 будет указан пустой массив, то в этом случае он интерпретируется как нулевой, то есть используемый по умолчанию.

Для всех вышеописанных форм вызова возвращается решение x. Если решение получено, то оно все равно сопровождается сообщением о величине относительной погрешности решения и количестве использованных итераций. Если превышено максимальное число итераций или процедура прекратила счет, выводится соответствующее предупреждающее сообщение.

Функция [x, flag] = bicg(...) возвращает решение и флаг, указывающий на характер сходимости решения:

Значение флага	*Информация о сходимости*
0	Процедура bicg сходится к решению при заданной точности tol в пределах числа итераций maxit
1	Процедура bicg превысила максимальное число итераций maxit, не достигнув сходимости
2	Одна из вспомогательных систем вида M*y = r плохо обусловлена и не возвращает приемлемого решения из встроенного решателя
3	Процедура остановлена, поскольку две последовательные оценки решения оказались одинаковыми
4	Одна из величин в процессе выполнения процедуры вышла за пределы допустимых чисел в компьютере

Если значение флага не равно 0, то возвращается то решение, которое соответствует минимальной относительной погрешности для всех выполненных шагов. При указании в качестве выходного параметра флага никаких сообщений не выводится.

Функция [x, flag, relres] = bicg(...) возвращает также относительную погрешность решения norm(b-A*x)/norm(b). Если флаг равен 0, то relres ≤ tol.

Функция [x, flag, relres, iter] = bicg(...) возвращает номер итерации, на которой было получено решение; этот номер удовлетворяет условию 0 ≤ iter ≤ maxit.

Функция [x, flag, relres, iter, resvec] = bicg(...) возвращает нормы невязок на каждой итерации в виде вектора resvec начиная с невязки resvec(1) = norm(b - A*x0). Если флаг равен 0, то вектор resvec имеет длину iter + 1 и значение последнего компонента должно удовлетворять условию resvec(end) ≤ tol*norm(b).

Примеры:

Рассмотрим систему линейных уравнений A*x = b, связанную с тестовой матрицей west0479 размера 479×479.

```
load west0479
A = west0479; b = sum(A, 2);
```

Поскольку вектор равен сумме столбцов матрицы системы, то решением такой системы является единичный вектор размера 479×1; порядок матрицы не слишком велик, поэтому найдем точное решение этой системы с помощью встроенного решателя

```
x = A \ b;
norm(b - A*x) / norm(b)
ans = 6.8476e-018
```

Теперь найдем решение, используя функцию bicg

```
[x, flag, relres, iter, resvec] = bicg(A, b);
flag
flag =    1
relres
relres = 1
iter
iter =  0
```

Значение флага равно 1, что означает отсутствие сходимости при точности 1e-6 (по умолчанию) в пределах 20 итераций (по умолчанию). Значение параметра iter, равное 0, свидетельствует о том, что процедура неустойчива, поскольку нулевое приближение имеет наименьшую относительную погрешность по сравнению с последующими шагами; минимальная погрешность relres равна 1.

Построим график относительной погрешности на каждой итерации (рис. 9.9), используя функцию

```
semilogy(0:20, resvec/norm(b), '-o'), grid
xlabel('Номер итерации', 'FontName', 'Arial Cyr', 'FontSize', 12)
ylabel('||b - Ax||/||b||', 'FontSize', 12)
```

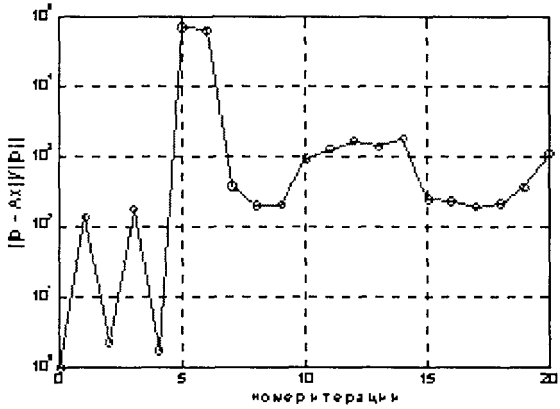


Рис. 9.9

Этот график свидетельствует о неустойчивости вычислительной процедуры.

Выполним факторизацию матрицы A с помощью неполного LU-разложения с порогом разреженности 1e-5:

```
[L1, U1] = luinc(A, 1e-5);
Warning: Incomplete upper triangular factor has 1 zero diagonal.
        It cannot be used as a preconditioner for an iterative method.
```

Предупреждение: Верхний треугольный множитель имеет 1 нулевой элемент на диагонали. Такое разложение неприменимо для итерационного метода.

Предупреждение указывает на сингулярность LU-разложения при заданном пороге разреженности.

Уменьшим порог разреженности до 1e-6, то есть будем использовать менее разреженную матрицу:

```
[L2, U2] = luinc(A, 1e-6);
nnz(A)
ans =   1887
nnz(L2)
ans =   6231
nnz(U2)
ans =   4559
```

В этом случае предупреждающих сообщений нет и все элементы матрицы U2 ненулевые. Теперь можно воспользоваться процедурой bicg:

```
[x, flag, relres, iter, resvec] = bicg(A, b,1e-15, 10, L2, U2);
flag
flag =    0
relres
relres =  2.8664e-016
iter
iter =    8
```

Требуемое решение получено на 8-й итерации.

Теперь можно построить относительную погрешность решения на каждой итерации (рис. 9.10), используя функцию

```
semilogy(0:iter, resvec/norm(b), '-o'), grid
xlabel('Номер итерации', 'FontName', 'Arial Cyr', 'FontSize', 12)
ylabel('||b - Ax||/||b||', 'FontSize', 12)
```

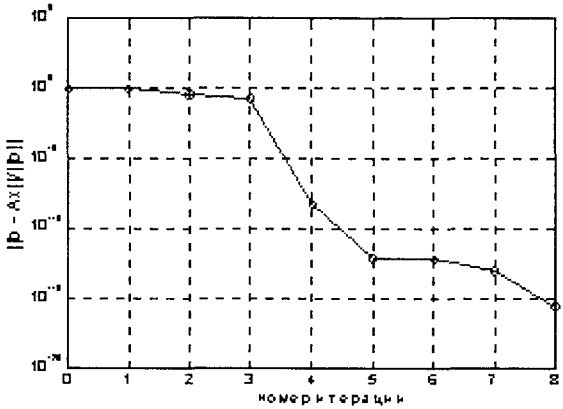


Рис. 9.10

Сопутствующие функции: BICGSTAB, CGS, GMRES, LUINC, PCG, QMR, \.

Ссылки:

1. Barrett R., et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1994.

BICGSTAB — Устойчивый метод бисопряженных градиентов

Синтаксис:

```
x = bicgstab(A, b)
x = bicgstab(A, b, tol)
x = bicgstab(A, b, tol, maxit)
x = bicgstab(A, b, tol, maxit, M)
x = bicgstab(A, b, tol, maxit, M1, M2)
x = bicgstab(A, b, tol, maxit, M1, M2, x0)
[x, flag] = bicgstab(...)
[x, flag, relres] = bicgstab(...)
[x, flag, relres, iter] = bicgstab(...)
[x, flag, relres, iter, resvec] = bicgstab(...)
```

Описание:

Функция x = bicgstab(A, b) реализует попытку решения системы линейных уравнений A*x = b относительно неизвестного вектора x. Матрица A порядка n предполагается произвольной квадратной. Решение стартует с начального

приближения, по умолчанию соответствующего нулевому вектору длины n, и итерации продолжаются до тех пор, пока метод сойдется, либо произойдет останов вычислений, либо будет превышено максимальное число итераций. Сходимость достигнута, когда относительная погрешность norm(b - A*x)/norm(b) становится меньше или равной заданной погрешности вычислений (по умолчанию 1e-6). Максимальное число итераций принимается по умолчанию равным min(n, 20). Никакой предварительной обработки матрицы не выполняется.

Функция x = bicgstab(A, b, tol) позволяет задать погрешность вычисления tol.

Функция x = bicgstab(A, b, tol, maxit) позволяет изменить значение максимального числа итераций.

Функции x = bicgstab(A, b, tol, maxit, M) и x = bicgstab(A, b, tol, maxit, M1, M2) используют вспомогательные матрицы M либо M=M1*M2, для того чтобы преобразовать исходную систему в вид M*y = b, где y = inv(M)*A*x, и достичь более высокой эффективности решения. Поскольку решения системы вида M*y = b ищутся с помощью встроенного решателя, разумно выполнить факторизацию матрицы M, используя LU-разложение. Тогда решение будет использовать два оператора:

R = lu(M);

x = bicgstab(A, b, tol, maxit, R', R).

Если в качестве матриц M, M1, M2 будет указана пустая матрица, то в этом случае она интерпретируется как единичная, то есть никаких вспомогательных операций не выполняется.

Функция x = bicgstab(A, b, tol, maxit, M1, M2, x0) позволяет изменить вектор начальной оценки решения x0. Если в качестве вектора x0 будет указан пустой массив, то в этом случае он интерпретируется как нулевой, то есть используемый по умолчанию.

Для всех вышеописанных форм вызова возвращается решение x. Если решение получено, то оно все равно сопровождается сообщением о величине относительной погрешности решения и количестве использованных итераций. Если превышено максимальное число итераций или процедура прекратила счет, выводится соответствующее предупреждающее сообщение.

Функция [x, flag] = bicgstab(...) возвращает решение и флаг, указывающий на характер сходимости решения:

Значение флага	*Информация о сходимости*
0	Процедура bicgstab сходится к решению при заданной точности tol в пределах числа итераций maxi
1	Процедура bicgstab превысила максимальное число итераций maxit, не достигнув сходимости
2	Одна из вспомогательных систем вида M*y = r плохо обусловлена и не возвращает приемлемого решения из встроенного решателя

3	Процедура остановлена, поскольку две последовательные оценки решения оказались одинаковыми
4	Одна из величин в процессе выполнения процедуры вышла за пределы допустимых чисел в компьютере

Если значение флага не равно 0, то возвращается то решение, которое соответствует минимальной относительной погрешности для всех выполненных шагов. При указании в качестве выходного параметра флага никаких сообщений не выводится.

Функция [x, flag, relres] = bicgstab(...) возвращает также относительную погрешность решения norm(b - A*x)/norm(b). Если флаг равен 0, то relres ≤ tol.

Функция [x, flag, relres, iter] = bicgstab(...) возвращает номер итерации, на которой было получено решение; этот номер удовлетворяет условию 0 ≤ iter ≤ maxit; параметр может принимать либо целочисленное значение, либо целочисленное значение плюс 0.5, поскольку для устойчивости метода используется половинный шаг.

Функция [x, flag, relres, iter, resvec] = bicgstab(...) возвращает нормы невязок на каждой итерации в виде вектора resvec начиная с невязки resvec(1) = norm(b - A*x0). Если флаг равен 0, то вектор resvec имеет длину 2*iter + 1; значение последнего компонента должно удовлетворять условию resvec(end) ≤ tol*norm(b).

Примеры:

Рассмотрим систему линейных уравнений A*x = b, связанную с тестовой матрицей west0479 размера 479×479.

```
load west0479
A = west0479; b = sum(A, 2);
```

Найдем решение, используя функцию bicgstab

```
[x, flag] = bicgstab(A, b);
flag
flag =    1
```

Значение флага равно 1, что означает отсутствие сходимости при точности 1e-6 (по умолчанию) в пределах 20 итераций (по умолчанию).

Выполним факторизацию матрицы A с помощью неполного LU-разложения с порогом разреженности 1e-5:

```
[L1, U1] = luinc(A, 1e-5);
Warning: Incomplete upper triangular factor has 1 zero diagonal.
        It cannot be used as a preconditioner for an iterative method.
```

Предупреждение: Верхний треугольный множитель имеет 1 нулевой элемент на диагонали. Такое разложение неприменимо для итерационного метода.

```
[x1, flag1, relres1, iter1] = bicgstab(A, b, 1e-6, 20, L1, U1);
Warning: Matrix is close to singular or badly scaled.
        Results may be inaccurate. RCOND = 0.000000e+000.
```

Предупреждение: Матрица близка к вырожденной или плохо масштабирована.

```
Результат может быть неточным.  RCOND = 0.000000e+000.
flag1
flag1 =    2
relres1
relres1 =   1
iter1
iter1 =    0
```

Предупреждение указывает на вырожденность системы U1*y = r на первой итерации.

Уменьшим порог разреженности до 1e-6:

```
[L2, U2] = luinc(A, 1e-6);
```

В этом случае предупреждающих сообщений нет и все элементы матрицы U2 ненулевые. Теперь можно воспользоваться процедурой bicgstab:

```
[x2, flag2, relres2, iter2, resvec2] = bicgstab(A, b,1e-15, 10, L2, U2);
flag2
flag2 =    0
relres2
relres2 =  2.8534e-016
iter2
iter2 =    6
```

Требуемое решение получено на 6-й итерации.

Теперь можно построить относительную погрешность решения на каждой итерации (рис. 9.11), используя функцию

```
semilogy(0:0.5:iter2, resvec2/norm(b), '-o'), grid
xlabel('Номер итерации', 'FontName', 'Arial Cyr', 'FontSize', 12)
ylabel('||b - Ax||/||b||', 'FontSize', 12)
```

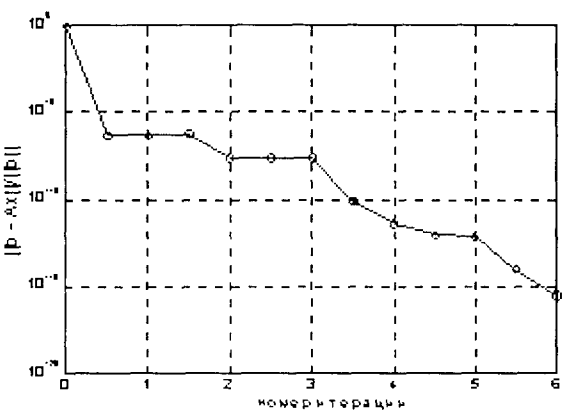


Рис. 9.11

Сопутствующие функции: BICG, CGS, GMRES, LUINC, PCG, QMR, \ .

Ссылки:

1. Van der Vorst H. A. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems// *SIAM J. Sci. Stat.Comput.* 1992. Vol. 13. No. 2. P. 631-644.
2. Barrett R., et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1994.

CGS — Квадратичный метод сопряженных градиентов

Синтаксис:

```
x = cgs(A, b)
x = cgs(A, b, tol)
x = cgs(A, b, tol, maxit)
x = cgs(A, b, tol, maxit, M)
x = cgs(A, b, tol, maxit, M1, M2)
x = cgs(A, b, tol, maxit, M1, M2, x0)
[x, flag] = cgs(...)
[x, flag, relres] = cgs(...)
[x, flag, relres, iter] = cgs(...)
[x, flag, relres, iter, resvec] = cgs(...)
```

Описание:

Функция x = cgs(A, b) реализует попытку решения системы линейных уравнений A*x = b относительно неизвестного вектора x. Матрица A порядка n предполагается произвольной квадратной. Решение стартует с начального приближения, по умолчанию соответствующего нулевому вектору длины n, и итерации продолжаются до тех пор, пока метод сойдется, либо произойдет останов вычислений, либо будет превышено максимальное число итераций. Сходимость достигнута, когда относительная погрешность norm(b - A*x)/norm(b) становится меньше или равной заданной погрешности вычислений (по умолчанию 1e-6). Максимальное число итераций принимается по умолчанию равным min(n, 20). Никакой предварительной обработки матрицы не выполняется.

Функция x = cgs(A, b, tol) позволяет задать погрешность вычисления tol.

Функция x = cgs(A, b, tol, maxit) позволяет изменить значение максимального числа итераций.

Функции x = cgs(A, b, tol, maxit, M) и x = cgs(A, b, tol, maxit, M1, M2) используют вспомогательные матрицы M либо M=M1*M2, для того чтобы преобразовать исходную систему в вид M*y = b, где y = inv(M)*A*x, и достичь более высокой эффективности решения. Поскольку решения системы вида M*y = b ищутся с помощью встроенного решателя, разумно выполнить факторизацию матрицы M, используя LU-разложение. Тогда решение будет использовать два оператора:

```
R = lu(M);
x = cgs(A, b, tol, maxit, R', R).
```

Если в качестве матриц M, M1, M2 будет указана пустая матрица, то в этом случае она интерпретируется как единичная, то есть никаких вспомогательных операций не выполняется.

Функция x = cgs(A, b, tol, maxit, M1, M2, x0) позволяет изменить вектор начальной оценки решения x0. Если в качестве вектора x0 будет указан пустой массив, то в этом случае он интерпретируется как нулевой, то есть используемый по умолчанию.

Для всех вышеописанных форм вызова возвращается решение x. Если решение получено, то оно все равно сопровождается сообщением о величине относительной погрешности решения и количестве использованных итераций. Если превышено максимальное число итераций или процедура прекратила счет, выводится соответствующее предупреждающее сообщение.

Функция [x, flag] = cgs(...) возвращает решение и флаг, указывающий на характер сходимости решения:

Значение флага	*Информация о сходимости*
0	Процедура cgs сходится к решению при заданной точности tol в пределах числа итераций maxit
1	Процедура cgs превысила максимальное число итераций maxit, не достигнув сходимости
2	Одна из вспомогательных систем вида M*y = r плохо обусловлена и не возвращает приемлемого решения из встроенного решателя
3	Процедура остановлена, поскольку две последовательные оценки решения оказались одинаковыми
4	Одна из величин в процессе выполнения процедуры вышла за пределы допустимых чисел в компьютере

Если значение флага не равно 0, то возвращается то решение, которое соответствует минимальной относительной погрешности для всех выполненных шагов. При указании в качестве выходного параметра флага никаких сообщений не выводится.

Функция [x, flag, relres] = cgs(...) возвращает также относительную погрешность решения norm(b - A*x)/norm(b). Если флаг равен 0, то relres ≤ tol.

Функция [x, flag, relres, iter] = cgs(...) возвращает номер итерации, на которой было получено решение; этот номер удовлетворяет условию 0 ≤ iter ≤ maxit.

Функция [x, flag, relres, iter, resvec] = cgs(...) возвращает нормы невязок на каждой итерации в виде вектора resvec начиная с невязки resvec(1) = norm(b - A*x0). Если флаг равен 0, то вектор resvec имеет длину iter + 1; значение последнего компонента должно удовлетворять условию resvec(end) ≤ tol*norm(b).

Примеры:

Рассмотрим систему линейных уравнений A*x = b, связанную с тестовой матрицей west0479 размера 479×479.

```
load west0479
A = west0479; b = sum(A, 2);
```

Найдем решение, используя функцию cgs

```
[x, flag] = cgs(A, b);
flag
flag =    1
```

Значение флага равно 1, что означает отсутствие сходимости при точности 1e-6 (по умолчанию) в пределах 20 итераций (по умолчанию).

Выполним факторизацию матрицы A с помощью неполного LU-разложения с порогом разреженности 1e-6:

```
[L2, U2] = luinc(A, 1e-6);
```

В этом случае предупреждающих сообщений нет и все элементы матрицы U2 ненулевые. Теперь можно воспользоваться процедурой cgs:

```
[x2, flag2, relres2, iter2, resvec2] = cgs(A, b,1e-15, 10, L2, U2);
flag2
flag2 =    0
relres2
relres2 =  7.8833e-016
iter2
iter2 =    5
```

Требуемое решение получено на 5-й итерации.

Теперь можно построить относительную погрешность решения на каждой итерации (рис. 9.12), используя функцию

```
semilogy(0:iter2, resvec2/norm(b), '-o'), grid
xlabel('Номер итерации', 'FontName', 'Arial Cyr', 'FontSize', 12)
ylabel('||b - Ax||/||b||', 'FontSize', 12)
```

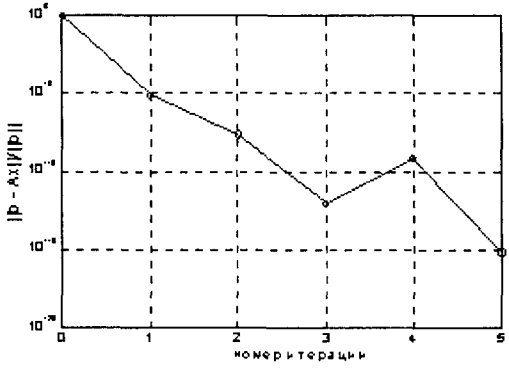


Рис. 9.12

Сопутствующие функции: BICG, BICGSTAB, GMRES, LUINC, PCG, QMR, \ .

Ссылки:

1. Sonneveld P. CGS: A fast Lanczos-type solver for nonsymmetric linear systems// *SIAM J. Sci. Stat. Comput.* 1989. Vol. 10, No. 1. P. 36-52.
2. Barrett R., et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1994.

GMRES **Обобщенный метод минимальной невязки**

Синтаксис:

```
x = gmres(A, b, restart)
x = gmres(A, b, restart, tol)
x = gmres(A, b, restart, tol, maxit)
x = gmres(A, b, restart, tol, maxit, M)
x = gmres(A, b, restart, tol, maxit, M1, M2)
x = gmres(A, b, restart, tol, maxit, M1, M2, x0)
[x, flag] = gmres(...)
[x, flag, relres] = gmres(...)
[x, flag, relres, iter] = gmres(...)
[x, flag, relres, iter, resvec] = gmres(...)
```

Описание:

Функция x = gmres(A, b, restart) реализует попытку решения системы линейных уравнений A*x = b относительно неизвестного вектора x. Матрица A порядка n предполагается произвольной квадратной. Решение стартует с начального приближения, по умолчанию соответствующего нулевому вектору длины n; процедура gmres выполняет рестарт по прошествии restart итераций, используя решение на последней итерации в качестве начального для следующего цикла; итерации продолжаются до тех пор, пока метод сойдется, либо произойдет останов вычислений, либо будет превышено максимальное число итераций. Сходимость достигнута, когда относительная погрешность norm(b - A*x)/norm(b) становится меньше или равной заданной погрешности вычислений (по умолчанию 1e-6). Максимальное число итераций принимается по умолчанию равным min(n, 20). Никакой предварительной обработки матрицы не выполняется.

Функция x = gmres(A, b, restart, tol) позволяет задать погрешность вычисления tol.

Функция x = gmres(A, b, restart, tol, maxit) позволяет изменить значение максимального числа итераций.

Функции x = gmres(A, b, restart, tol, maxit, M) и x = gmres(A, b, restart, tol, maxit, M1, M2) используют вспомогательные матрицы M либо M=M1*M2, для того чтобы преобразовать исходную систему в вид M*y = b, где y = inv(M)*A*x, и достичь более высокой эффективности решения. Поскольку решения системы вида M*y = b ищутся с помощью встроенного решателя, разумно выполнить факторизацию матрицы M, используя LU-разложение. Тогда решение будет использовать два оператора

```
R = lu(M);
x = gmres(A, b, restart, tol, maxit, R', R).
```

Если в качестве матриц M, M1, M2 будет указана пустая матрица, то в этом случае она интерпретируется как единичная, то есть никаких вспомогательных операций не выполняется.

Функция x = gmres(A, b, restart, tol, maxit, M1, M2, x0) позволяет изменить вектор начальной оценки решения x0. Если в качестве вектора x0 будет указан пустой массив, то в этом случае он интерпретируется как нулевой, то есть используемый по умолчанию.

Для всех вышеописанных форм вызова возвращается решение x. Если решение получено, то оно все равно сопровождается сообщением о величине относительной погрешности решения и количестве использованных итераций. Если превышено максимальное число итераций или процедура прекратила счет, выводится соответствующее предупреждающее сообщение.

Функция [x, flag] = gmres(...) возвращает решение и флаг, указывающий на характер сходимости решения:

Значение флага	*Информация о сходимости*
0	Процедура gmres сходится к решению при заданной точности tol в пределах числа итераций maxit
1	Процедура gmres превысила максимальное число итераций maxit, не достигнув сходимости
2	Одна из вспомогательных систем вида M*y = r плохо обусловлена и не возвращает приемлемого решения из встроенного решателя
3	Процедура остановлена, поскольку две последовательные оценки решения оказались одинаковыми

Если значение флага не равно 0, то возвращается то решение, которое соответствует минимальной относительной погрешности для всех выполненных шагов. При указании в качестве выходного параметра флага никаких сообщений не выводится.

Функция [x, flag, relres] = gmres(...) возвращает также относительную погрешность решения norm(b - A*x)/norm(b). Если флаг равен 0, то relres ≤ tol.

Функция [x, flag, relres, iter] = gmres(...) возвращает номера итераций внешнего и внутреннего циклов, на которых было достигнуто решение; номер итерации внешнего цикла iter(1) удовлетворяет условию 0 ≤ iter(1) ≤ maxit; номер итерации внутреннего цикла iter(2) удовлетворяет условию 0 ≤ iter(2) ≤ restart.

Функция [x, flag, relres, iter, resvec] = gmres(...) возвращает нормы невязок на каждой итерации в виде вектора resvec начиная с невязки resvec(1) = = norm(b-A*x0). Если флаг равен 0 и iter = [i j], то вектор resvec имеет длину (i - 1)*restart + j + 1; значение последнего компонента должно удовлетворять условию resvec(end) ≤ tol*norm(b).

Примеры:

Рассмотрим систему линейных уравнений A*x = b, связанную с тестовой матрицей west0479 размера 479×479.

```
load west0479
A = west0479; b = sum(A, 2);
```

Найдем решение, используя функцию gmres

```
[x, flag] = gmres(A, b, 5);
flag
flag =    1
```

Значение флага равно 1, что означает отсутствие сходимости при точности 1e-6 (по умолчанию) в пределах 20 итераций (по умолчанию).

Выполним факторизацию матрицы A с помощью неполного LU-разложения с порогом разреженности 1e-6:

```
[L2, U2] = luinc(A, 1e-6);
```

В этом случае предупреждающих сообщений нет и все элементы матрицы U2 ненулевые. Теперь можно воспользоваться процедурой gmres:

```
[x2, flag2, relres2, iter2, resvec2] = gmres(A, b, 4, 1e-15, 10, L2, U2);
flag2
flag2 =    0
relres2
relres2 =   1.4316e-016
 iter2(1)
ans =    4
iter2(2)
ans =    4
```

Требуемое решение получено на 4-й внешней после 16 внутренних итераций.

Теперь можно построить относительную погрешность решения на каждой внешней и внутренней итерации (рис. 9.13), используя такую последовательность операторов:

```
semilogy(0:length(resvec2)-1, resvec2/norm(b), '-o'), grid
xlabel('Номер внешней итерации', 'FontName', 'Arial Cyr', 'FontSize', 12)
ylabel('||b - Ax||/||b||', 'FontSize', 12)
```

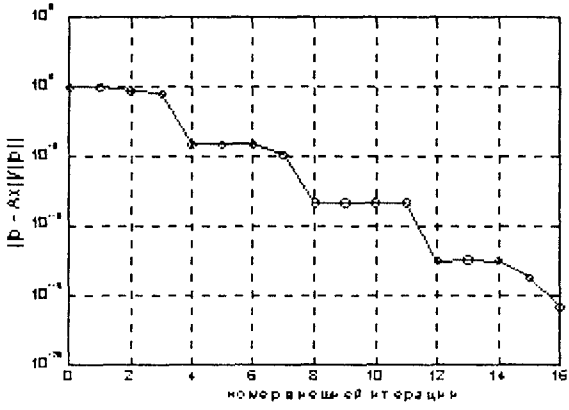


Рис. 9.13

Сопутствующие функции: BICG, BICGSTAB, CGS, LUINC, PCG, QMR, \ .

Ссылки:

1. Saad Y., Schultz M.H. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems// *SIAM J. Sci. Stat. Comput.* 1986. Vol. 7. No. 3. P. 856-869
2. Barrett R., et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1994.

QMR — Метод квазиминимальной невязки

Синтаксис:

```
x = qmr(A, b)
x = qmr(A, b, tol)
x = qmr(A, b, tol, maxit)
x = qmr(A, b, tol, maxit, M)
x = qmr(A, b, tol, maxit, M1, M2)
x = qmr(A, b, tol, maxit, M1, M2, x0)
[x, flag] = qmr(...)
[x, flag, relres] = qmr(...)
[x, flag, relres, iter] = qmr(...)
[x, flag, relres, iter, resvec] = qmr(...)
```

Описание:

Функция x = qmr(A, b) реализует попытку решения системы линейных уравнений A*x = b относительно неизвестного вектора x. Матрица A порядка n предполагается произвольной квадратной. Решение стартует с начального приближения, по умолчанию соответствующего нулевому вектору длины n, и итерации продолжаются до тех пор, пока метод сойдется, либо произойдет останов вычислений, либо будет превышено максимальное число итераций. Сходимость достигнута, когда относительная погрешность norm(b - A*x)/norm(b) становится меньше или равной заданной погрешности вычислений (по умолчанию 1e-6). Максимальное число итераций принимается по умолчанию равным min(n, 20). Никакой предварительной обработки матрицы не выполняется.

Функция x = qmr(A, b, tol) позволяет задать погрешность вычисления tol.

Функция x = qmr(A, b, tol, maxit) позволяет изменить значение максимального числа итераций.

Функции x = qmr(A, b, tol, maxit, M) и x = qmr(A, b, tol, maxit, M1, M2) используют вспомогательные матрицы M либо M=M1*M2, для того чтобы преобразовать исходную систему в вид M*y = b, где y = inv(M)*A*x, и достичь более высокой эффективности решения. Поскольку решения системы вида M*y = b ищутся с помощью встроенного решателя, разумно выполнить факторизацию матрицы M, применяя LU-разложение. Тогда решение будет использовать два оператора:

```
R = lu(M);
x = qmr(A, b, tol, maxit, R', R).
```

Если в качестве матриц M, M1, M2 будет указана пустая матрица, то в этом случае она интерпретируется как единичная, то есть никаких вспомогательных операций не выполняется.

Функция x = qmr(A, b, tol, maxit, M1, M2, x0) позволяет изменить вектор начальной оценки решения x0. Если в качестве вектора x0 будет указан пустой массив, то в этом случае он интерпретируется как нулевой, то есть используемый по умолчанию.

Для всех вышеописанных форм вызова возвращается решение x. Если решение получено, то оно все равно сопровождается сообщением о величине относительной погрешности решения и количестве использованных итераций. Если превышено максимальное число итераций или процедура прекратила счет, выводится соответствующее предупреждающее сообщение.

Функция [x, flag] = qmr(...) возвращает решение и флаг, указывающий на характер сходимости решения:

Значение флага	*Информация о сходимости*
0	Процедура qmr сходится к решению при заданной точности tol в пределах числа итераций maxit
1	Процедура qmr превысила максимальное число итераций maxit, не достигнув сходимости
2	Одна из вспомогательных систем вида M*y = r плохо обусловлена и не возвращает приемлемого решения из встроенного решателя
3	Процедура остановлена, поскольку две последовательные оценки решения оказались одинаковыми
4	Одна из величин в процессе выполнения процедуры вышла за пределы допустимых чисел в компьютере

Если значение флага не равно 0, то возвращается то решение, которое соответствует минимальной относительной погрешности для всех выполненных шагов. При указании в качестве выходного параметра флага никаких сообщений не выводится.

Функция [x, flag, relres] = qmr(...) возвращает также относительную погрешность решения norm(b - A*x)/norm(b). Если флаг равен 0, то relres ≤ tol.

Функция [x, flag, relres, iter] = qmr(...) возвращает номер итерации, на которой было получено решение; этот номер удовлетворяет условию 0 ≤ iter ≤ maxit.

Функция [x, flag, relres, iter, resvec] = qmr(...) возвращает нормы невязок на каждой итерации в виде вектора resvec начиная с невязки resvec(1) = norm(b - A*x0). Если флаг равен 0, то вектор resvec имеет длину iter + 1; значение последнего компонента должно удовлетворять условию resvec(end) ≤ tol*norm(b).

Примеры:

Рассмотрим систему линейных уравнений A*x = b, связанную с тестовой матрицей west0479 размера 479×479.

```
load west0479
A = west0479; b = sum(A, 2);
```

Найдем решение, используя функцию qmr

```
[x, flag] = qmr(A, b);
flag
flag =    1
```

Значение флага равно 1, что означает отсутствие сходимости при точности 1e-6 (по умолчанию) в пределах 20 итераций (по умолчанию).

Выполним факторизацию матрицы A с помощью неполного LU-разложения с порогом разреженности 1e-6:

```
[L2, U2] = luinc(A, 1e-6);
```

В этом случае предупреждающих сообщений нет и все элементы матрицы U2 ненулевые. Теперь можно воспользоваться процедурой cgs:

```
[x2, flag2, relres2, iter2, resvec2] = qmr(A, b,1e-15, 10, L2, U2);
flag2
flag2 =    0
relres2
relres2 =   2.337e-016
iter2
iter2 =    8
```

Требуемое решение получено на 8-й итерации.

Теперь можно построить относительную погрешность решения на каждой итерации (рис. 9.14), используя функцию

```
semilogy(0:iter2, resvec2/norm(b), '-o'), grid
xlabel('Номер итерации', 'FontName', 'Arial Cyr', 'FontSize', 12)
ylabel('||b - Ax||/||b||', 'FontSize', 12)
```

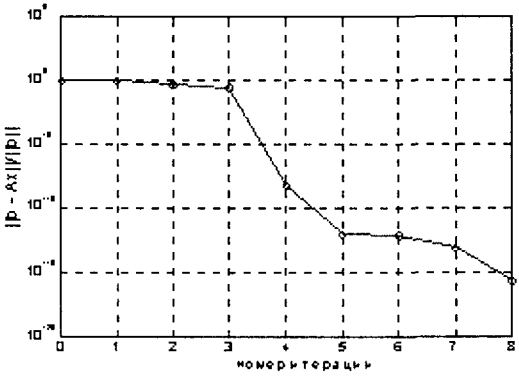


Рис. 9.14

Сопутствующие функции: BICG, BICGSTAB, CGS, GMRES, LUINC, PCG, \ .

Ссылки:

1. Freund R.W., Nachtigal N.M. QMR: A quasi-minimal residualmethod for non-Hermitian linear systems// *J. Numer. Math.* Vol. 60. 1991. P. 315-339.
2. Barrett R., et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1994.

Вычисление собственных значений и сингулярных чисел

EIGS **Вычисление отдельных собственных значений и векторов**

Синтаксис:

```
d = eigs(A)                          d = eigs('Afun', n)
d = eigs(A, k)                       d = eigs('Afun', n, k)
d = eigs(A, k, sigma)                d = eigs('Afun', n, k, sigma)
d = eigs(A, k, sigma, options)       d = eigs('Afun', n, k, sigma, options)
d = eigs(A, B, ...)                  d = eigs('Afun', n, B, ...)
[V, D] = eigs(A,...)                 [V, D] = eigs('Afun', n, ...)
[V, D, flag] = eigs(A, ...)          [V, D, f lag] = eigs('Afun', n, ...)
```

Описание:

Процедура eigs решает полную проблему A*v = lambda*v и обобщенную проблему A*v = lambda*B*v собственных значений для произвольной квадратной матрицы A. При этом вычисляется заданное количество собственных значений и векторов в противовес процедуре eig, которая вычисляет все собственные значения и векторы.

Функции d = eigs(A) и d = eigs('Afun', n) решают задачу на собственные значения, когда в качестве первого входного аргумента используется либо квадратная матрица (полная или разреженная, симметрическая или несимметрическая, действительная или комплексная), либо строка с именем М-файла, который применяет линейный оператор к столбцам матрицы. В последнем случае должен быть указан второй аргумент n, задающий порядок матрицы. Например, оператор eigs('fft', ...) eigs(F, ...) выполняется значительно быстрее, чем оператор eigs(F, ...), где F - явная форма матрицы после вычисления быстрого преобразования Фурье.

Функции d = eigs(A, k) и d = eigs('Afun', n, k) позволяют задать требуемое количество k максимальных по модулю собственных значений (по умолчанию k = min(n, 6)).

Функции d = eigs(A, k, sigma) и d = eigs('Afun', n, k, sigma) позволяют специфицировать вычисляемые собственные значения. Параметр sigma может быть либо числом, либо строкой из двух символов.

Если параметр sigma равен 0, то вычисляется k минимальных по модулю собственных значений; если sigma действительное или комплексное число, то оно интерпретируется как сдвиг, указывающий, что должно быть вычислено k ближайших к sigma собственных значений. Если значение sigma точно совпало с собственным значением, функция окажется несостоятельной из-за деления на нуль в выражении 1/(lambda - sigma). В этом случае следует начать вычисления с новым значением sigma2, близким, но не равным sigma.

Если параметр sigma - строка из двух символов, то определены следующие спецификации для собственных значений:

sigma	*Назначение*
'lm'	Вычислить максимальные по модулю значения (по умолчанию)
'sm'	Вычислить минимальные по модулю значения (эквивалентно sigma = 0)
'lr'	Вычислить значения с максимальными действительными частями
'sr'	Вычислить значения с минимальными действительными частями
'be'	Вычислить floor(k/2) минимальных и ceil(k/2) максимальных собственных значений

Функции d = eigs(A, k, sigma, options) и d = eigs('Afun', n, k, sigma, options) позволяют специфицировать следующие опции в массиве записей options:

Опция	*Описание*	*Значение по умолчанию*
options.tol	Условие сходимости: norm(A*V-V*D) ≤ tol*norm(A)	1e-10 (для симметрических матриц) 1e-6 (для несимметрических матриц)
options.p	Размерность базиса Арнольда	2*k
options.maxit	Максимальное количество итераций	300
options.disp	Количество собственных значений, выводимых на каждой итерации. Установить в 0, если не требуется промежуточного вывода	20
options.issym	Положительное число, если матрица A симметрическая	0
options.cheb	Положительное число, если Afun - строка, sigma - число или 'lr', 'sr' и если должно быть применено полиномиальное ускорение	0
options.v0	Начальный вектор для факторизации Арнольда	rand(n,1) - .5

Функции d = eigs(A, B, ...) и d = eigs('Afun', n, B, ...) решают обобщенную задачу на собственные значения. Матрица B имеет такие же размеры, как и матрица A; если матрица B не указана, то используется единичная матрица B = eye(size(A)).

Выход d - это вектор, содержащий k собственных значений.

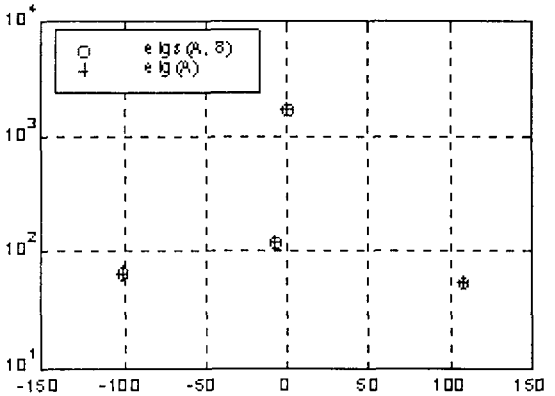
В случае двух выходных аргументов функции [V, D] = eigs(A,...) и [V, D] = = eigs('Afun', n, ...) формируются матрица V размера k×n и диагональная матрица D размера k×k, такие, что A*V = V*D или A*V = B*V*D.

В случае трех выходных аргументов функции [V, D, flag] = eigs(A,...) и [V, D, flag] = eigs('Afun', n, ...) выходной аргумент flag указывает, была ли решена проблема собственных значений с заданной точностью; значение flag = 0 указывает на сходимость, а значение flag = 1 на расходимость процедуры.

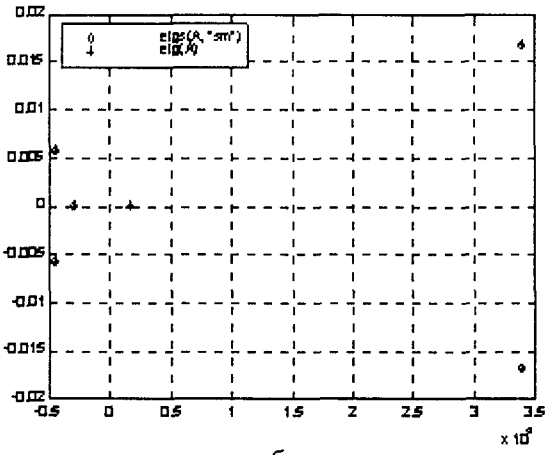
Примеры:

Рассмотрим разреженную матрицу west0479 размера 479×479, которая имеет действительные и комплексно-сопряженные собственные значения. Функция eig вычисляет все 479 собственных значений; с помощью функции eigs легко вычислить заданное количество минимальных и максимальных по модулю собственных значений

```
load west0479
d = eig(full(west0479));
dlm = eigs(west0479, 8);
dsm = eigs(west0479, 'sm');
% Рис. 9.15, а
semilogy(real(d(1:8)), abs(imag(d(1:8))), 'o'),  hold on
semilogy(real(dlm), abs(imag(dlm)), '+'),  grid
legend('  eigs(A, 8)',        '  eig(A)', 2)
% Рис. 9.15, б
plot(real(dsm), abs(imag(dsm)), 'o'),  hold on
plot(real(d(end-5:end)), abs(imag(d(end-5:end))), '+')
legend('  eigs(A, "sm")',        '  eig(A)', 2)
plot(real(dsm), -abs(imag(dsm)), 'o')
plot(real(d(end-5:end)),  -abs(imag(d(end-5:end))), '+')
grid
```



а



б

Рис. 9.15. Собственные значения по модулю: а - максимальные; б - минимальные

Сопутствующие функции: EIG, SVDS.

Ссылки:

1. Radke R. A MATLAB Implementation of the Implicitly Restarted Arnoldi Method for Solving Large-Scale Eigenvalue Problems. Houston: Dept. of Computational and Applied Math, Rice University, 1994.
2. Sorensen D. C. Implicit Application of Polynomial Filters in a k-step Arnoldi Method// *SIAM Journal on Matrix Analysis and Applications.* 1992. Vol. 13. P. 357-385.
3. Lehoucq B., Sorensen D.C. Deflation Techniques within an Implicitly Restarted Iteration// *SIAM Journal on Matrix Analysis and Applications.* 1996. Vol. 17. P. 789-821.

SVDS — Вычисление отдельных сингулярных чисел

Синтаксис:

```
s = svds(A)
s = svds(A, k)
s = svds(A, k, 0)
[U, S, V] = svds(A, ...)
```

Описание:

Функция s = svds(A) вычисляет 5 максимальных сингулярных чисел и связанных с ними сингулярных векторов произвольной прямоугольной матрицы A.

Функция s = svds(A, k) вычисляет k максимальных сингулярных чисел и связанных с ними сингулярных векторов.

Функция s = svds(A, k, 0) вычисляет k минимальных сингулярных чисел и связанных с ними сингулярных векторов.

Один выходной аргумент выводит вектор сингулярных чисел; в случае трех аргументов выводятся:

- матрица U размера m×k с ортонормированными столбцами;
- диагональная матрица сингулярных чисел S размера k×k;
- матрица V размера n×k с ортонормированными столбцами.

Произведение U*S*V' является наилучшей аппроксимацией ранга k для исходной прямоугольной матрицы A.

Алгоритм:

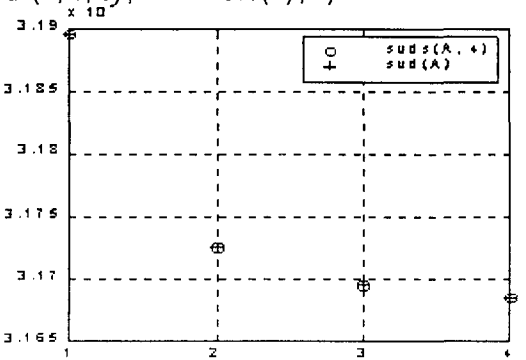
Функция svds(A, k) использует функцию eigs для вычисления k максимальных собственных значений и соответствующих собственных векторов для расширенной матрицы B = [0 A; A' 0] размера (m+n)×(m+n).

Функция svds(A, k, 0) использует функцию eigs для вычисления 2k минимальных собственных значений и соответствующих собственных векторов для матрицы B = [0 A; A' 0], а затем из них выбирается k положительных.

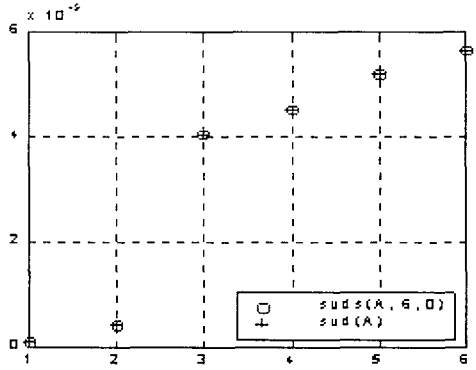
Примеры:

Рассмотрим разреженную матрицу west0479 размера 479×479. Функция svd вычисляет все 479 сингулярных чисел; с помощью функции svds легко вычислить заданное количество минимальных и максимальных сингулярных чисел.

```
load west0479
s = svd(full(west0479));
sl = svds(west0479, 4);
ss = svds(west0479, 6, 0);
% Рис. 9.16, а
plot(s(1:4), 'o'), hold on
plot(sl, '+'), grid
legend('  svds(A, 4)',        '  svd(A)', 1)
% Рис. 9.16, б
plot(sort(ss), 'o'), hold on
plot(sort(s(end-5:end)), '+'), grid
legend('  svds(A, 6, 0)',        '  svd(A)', 4)
```



а



б

Рис. 9.16. Сингулярные числа: а - максимальные; б - минимальные

Сопутствующие функции: SVD, EIGS.

Операции над деревьями

ETREE **Дерево матрицы**

Синтаксис:

p = etree(A)	[p, q] = etree(A)
p = etree(A, 'col')	[p, q] = etree(A, 'col')
p = etree(A, 'sym')	[p, q] = etree(A, 'sym')

Описание:

Функция p = etree(A) возвращает структуру дерева для квадратной симметрической матрицы A; значение p(j) - это родитель столбца с номером j в дереве; если p(j) = 0, то столбец j - корень дерева.

Функция p = etree(A, 'col') возвращает структуру дерева для матрицы A' * A.

Функция p = etree(A, 'sym') равносильна функции p = etree(A).

Функция [p, q] = etree(A, ...) в дополнение к структуре дерева возвращает также вектор перестановок столбцов q.

Пример:

Рассмотрим матрицу A следующего вида:

```
A = [1 0 0 1; 0 1 0 0; 0 0 1 0; 1 0 0 1]
A =
    1    0    0    1
    0    1    0    0
    0    0    1    0
    1    0    0    1
```

Эта матрица симметрическая, и применение функции etree дает следующий результат:

```
[p, q] = etree(A)
p =    4    0    0    0
q =    2    3    1    4
```

Сопутствующие функции: ETREEPLOT, TREELAYOUT, TREEPLOT.

ETREEPLOT **Построение дерева матрицы**

Синтаксис:

```
etreeplot(A)
etreeplot(A, c, d)
```

Описание:

Функция etreeplot(A) строит дерево для квадратной матрицы A; если матрица несимметрическая, то делает это для A + A'.

Функция etreeplot(A, c, d) в дополнение к построению дерева для квадратной матрицы A позволяет управлять путем задания параметров c и d соответственно цветом и символами обозначения узлов и ребер. Если один из параметров заменен на '', то на график не будут выводиться либо ребра, либо узлы. Если эти параметры не указаны, то принимаются их значения по умолчанию.

Пример:

Рассмотрим матрицу A следующего вида:

```
A = [1 0 0 1; 0 1 0 0; 0 0 1 0; 1 0 0 1]
A =
    1    0    0    1
    0    1    0    0
    0    0    1    0
    1    0    0    1
```

Эта матрица симметрическая, и применение функций etree и etreeplot дает следующие результаты:

```
[p, q] = etree(A)
p =    4    0    0    0
q =    2    3    1    4
etreeplot(A)
```

На рис. 9.17 показан график соответствующего дерева.

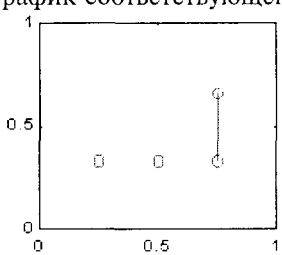


Рис. 9.17

Сопутствующие функции: ETREE, TREELAYOUT, TREEPLOT.

TREELAYOUT **Разметить дерево**

Синтаксис:

[x, y, h, s] = treelayout(p, q)

Описание:

Функция [x, y, h, s] = treelayout(p, q) использует в качестве входных параметров вектор p(j), отождествляемый либо с родителем столбца с номером j в дереве, либо с корнем дерева, если p(j) = 0, а также вектор q перестановки столбцов. Если вектор q не указан, то он вычисляется в процессе выполнения функции.

Выходные параметры x и y - это векторы координат узлов дерева, которое размещается в пределах единичного квадрата, чтобы сделать график удобным для восприятия.

Дополнительные выходные параметры h и s определяют соответственно высоту и количество узлов.

Эту функцию целесообразно использовать в сочетании с функцией построения графа gplot.

Пример:

Рассмотрим матрицу A следующего вида:

```
A = [1 0 0 1; 0 1 0 0; 0 0 1 0; 1 0 0 1]
A =
   1   0   0   1
   0   1   0   0
   0   0   1   0
   1   0   0   1
```

Эта матрица симметрическая, и применение функций etree, treelayout и gplot дает следующие результаты:

```
[p, q] = etree(A)
p =   4   0   0   0
q =   2   3   1   4
[x, y, h, s] = treelayout(p, q);
gplot(A, [x' y'], 'r-'),  hold on
gplot(A, [x' y'], 'bo')
```

На рис. 9.18 показан график соответствующего дерева.

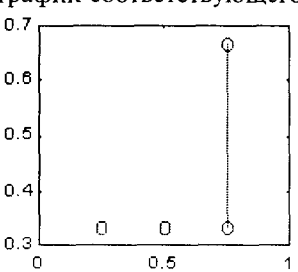


Рис. 9.18

Сопутствующие функции: ETREE, ETREEPLOT, TREEPLOT.

TREEPLOT **Построение дерева матрицы**

Синтаксис:

treeplot(p)
treeplot(p, c, d)

Описание:

Функция treeplot(p) строит дерево, если задан вектор p(j), отождествляемый либо с родителем столбца с номером j в дереве, либо с корнем дерева, если p(j) = 0.

Функция treeplot(p, c, d) в дополнение к построению дерева позволяет управлять цветом и символами обозначения узлов и ребер путем задания соответственно параметров c и d. Если один из параметров заменен на ", то на график не будут выводиться либо ребра, либо узлы. Если эти параметры не указаны, то принимаются их значения по умолчанию.

Пример:

Рассмотрим матрицу A следующего вида:

```
A = [1 0 0 1; 0 1 0 0; 0 0 1 0; 1 0 0 1]
A =
    1    0    0    1
    0    1    0    0
    0    0    1    0
    1    0    0    1
```

Эта матрица симметрическая, и применение функций etree и treeplot даст следующие результаты:

```
[p, q] = etree(A)
p =    4    0    0    0
q =    2    3    1    4
treeplot(p, 'bo', 'r')
```

На рис. 9.19 показан график соответствующего дерева.

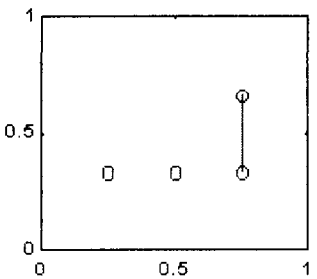


Рис. 9.19

Сопутствующие функции: ETREE, TREELAYOUT, TREEPLOT.

Вспомогательные операции

SPPARMS

Установка параметров для алгоритмов упорядочения и прямого решения линейных уравнений

Синтаксис:

spparms
values = spparms
[keys, values] = spparms
value = spparms('<*ключ*>')
spparms('<*ключ*>', <*значение*>)
spparms(<*значения*>)

Описание:

Работа с разреженными матрицами в системе MATLAB организована с помощью специального монитора SParse MONItor. Большинству пользователей знание служебных команд работы с монитором совершенно не обязательно. Эти команды предназначены только для тех пользователей, которые хотели бы увидеть и, возможно, изменить работу алгоритмов упорядочения и решения систем линейных уравнений.

Функция spparms сама по себе выводит на экран описания текущих установок.

Функция values = spparms выводит на экран значения этих установок.

Функция [keys, values] = spparms выводит вектор значений и символьную матрицу, строки которой являются ключами параметров монитора SParse MONItor.

Функция value = spparms('<*ключ*>') выводит значение параметра для данного ключа.

Функция spparms('<*ключ*>', <*значение*>) устанавливает один или более настраиваемых параметров, которые используются в функциях упорядочения colmmd и symmmd, а также в операциях решения систем линейных уравнений / и \.

Функция spparms(values) применяется без выходных параметров и устанавливает значения, указанные во входном векторе.

Функция spparms('default') устанавливает значения по умолчанию.

Функция spparms('tight') устанавливает параметры алгоритмов упорядочения по разреженности к наиболее "строгим" значениям, что приводит к процедурам упорядочения без удаления строк, которые требуют большего времени выполнения.

В нижеследующей таблице приведены расшифровки ключей для параметров и их значений по умолчанию и в "строгом смысле".

Параметр	*Ключ*	*Значения*	
		по умолчанию	*в "строгом смысле"*
values(1)	`spumoni`	0.0	
values(2)	`thr_rel`	1.1	1.0
values(3)	`thr_abs`	1.0	0.0
values(4)	`exact_d`	0.0	1.0
values(5)	`supernd`	3.0	1.0
values(6)	`rreduce`	3.0	1.0
values(7)	`wh_frac`	0.5	0.5
values(8)	`autommd`	1.0	
values(9)	`aug_rel`	0.001	
values(10)	`aug_abs`	0.0	

При этом ключи имеют следующие значения:

Ключи	*Значение*
`spumoni`	Флаг монитора SParse MONItor: 0 - не выводит диагностических сообщений, по умолчанию; 1 - выводит информацию о выбранном алгоритме; 2 - выводит детальную информацию о работе алгоритмов упорядочения
`thr_rel`, `thr_abs`	Порог минимальной разреженности thr_rel * mindegree + thr_abs
`exact_d`	0 - использование приближенных значений разреженности; ≠0 - использование точных значений разреженности
`supernd`	Целое ≥ 1 - объединение узлов каждые supernd шагов
`rreduce`	Целое ≥ 1 - удаление строк каждые rreduce шагов
`wh_frac`	В процедуре colmmd строки с коэффициентом заполнения, превышающим wh_frac, пропускаются
`autommd`	≠ 0 - в процедурах \ и / использовать упорядочение по разреженности
`aug_rel`, `aug_abs`	Параметр масштабирования невязки в проблеме наименьших квадратов, равный aug_rel * max(max(abs(A))) + aug_abs. Например, при значениях aug_rel = 0 и aug_abs = 1 единичная матрица главного верхнего блока будет немасштабированной

Примеры:

Ниже приведены примеры выполнения вспомогательных операций и указан русский перевод сообщений системы.

spparms (для значений по умолчанию)

No SParse MONItor output.	Выходных данных SParse MONItor не формирует
mmd: threshold = 1.1 * mindegree + 1, using approximate degrees in A'*A,	mmd: <порог> = 1.1 * mindegree + 1; приближенные значения разреженности при работе с A'*A;

supernode amalgamation every 3 stages,
row reduction every 3 stages,
withhold rows at least 50% dense in colmmd.
Minimum degree orderings used by \ and /.
Residual scale parameter = 0.001 * max(abs(A)).

объединение узлов - каждые 3 шага;
редукция строк - каждые 3 шага;
удерживать строки с разреженностью не менее 50 %.
Для операций \ и / применять упорядочение по разреженности.
Параметр невязки: 0.001 * max(abs(A)).

```
[keys, values] = spparms
keys =                    values =
        spumoni                   0
        thr_rel                1.1000
        thr_abs                1.0000
        exact_d                   0
        supernd                3.0000
        rreduce                3.0000
        wh_frac                0.5000
        autommd                1.0000
        aug_rel                0.0010
        aug_abs                   0
value = spparms('spumoni')
value =    0.
```

Сопутствующие функции: COLMMD, SYMMMD.

Ссылки:

1. Gilbert J. R., Moler C., Schreiber R. Sparse Matrices in MATLAB: Design and Implementation//*SIAM Journal on Matrix Analysis and Applications*. 1992. Vol. P. 333-356.

SYMBFACT — Характеристики разложения Холецкого

Синтаксис:

```
count = symbfact(A)
count = symbfact(A, f)
[count, h, parent, post, R] = symbfact(A, f)
```

Описание:

Функция count = symbfact(A) вычисляет количество элементов в строках верхнего треугольного множителя разложения Холецкого исходной симметрической матрицы A. Эта функция выполняется значительно быстрее, чем chol(A).

Функция count = symbfact(A, `col`) анализирует матрицу A'*A, не формируя ее в явном виде.

Функция count = symbfact(A, `sym`) аналогична процедуре count = symbfact(A).

Функция [count, h, p, q, R] = symbfact(A, f) возвращает несколько дополнительных выходных параметров:

- h - высоту графа;
- p - параметры графа;
- q - вектор перестановки столбцов;
- R - матрицу связности для процедуры chol(A).

Пример:

Рассмотрим матрицу A следующего вида:

```
A = [4 0 0 1; 0 1 0 0; 0 0 1 0; 1 0 0 1]
A =
   4   0   0   1
   0   1   0   0
   0   0   1   0
   1   0   0   1
```

Это симметрическая положительно определенная матрица, и применение функции chol(A) дает следующий результат:

```
L = chol(A)
L =
        2        0        0      0.5
        0        1        0        0
        0        0        1        0
        0        0        0  0.86603
```

Применение функции symbfact(A) вычисляет количество элементов в строках матрицы L:

```
c = symbfact(A)
c =    2    1    1    1
```

Если к матрице A применить алгоритм симметрической упорядоченности, то ненулевые элементы в разложении Холецкого будут концентрироваться вблизи диагонали. Рассмотрим последовательность процедур

```
r = symmmd(A);
L = chol(A(r, r))
L =
        1        0        0        0
        0        1        0        0
        0        0        2      0.5
        0        0        0  0.86603
c = symbfact(A(r, r))
c =  1    1    2    1
```

Обращение вида [count, h, p, q, R] = symbfact(A) вычисляет параметры для построения графа матрицы. Приводимая ниже последовательность операторов позволяет построить граф с пронумерованными вершинами.

```
[c, h, p, q] = symbfact(A)
c =    2    1    1    1
h =    2
p =    4    0    0    0
q =    2    3    1    4
[x, y] = treelayout(p, q);
gplot(A, [x' y'], 'bo'), hold on
gplot(A, [x' y'], 'r')
for j=1:size(A, 1)
   text(x(j) + 0.02, y(j) + 0.02, int2str(j))
end
```

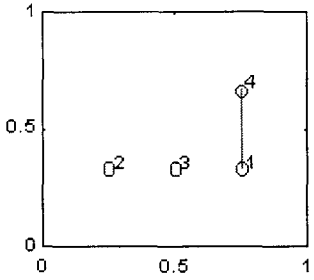


Рис. 9.20

Сопутствующие функции: CHOL, ETREE, TREELAYOUT.

SPAUGMENT

Формирование расширенной матрицы для метода наименьших квадратов

Синтаксис:

```
S = spaugment(A)
S = spaugment(A, c)
```

Описание:

Функции S = spaugment(A) и S = spaugment(A, c) формируют разреженную квадратную симметрическую матрицу следующей блочной структуры:

S = [c*I A; A' 0].

Такая расширенная матрица используется для минимизации квадратичной формы

$$\min_x \| b - Ax \| .$$

Если r = b - Ax выполняет роль невязки, а *c* - масштабирующий множитель, то решение методом наименьших квадратов сводится к решению следующей системы линейных уравнений:

$$\begin{bmatrix} cI & A \\ A' & 0 \end{bmatrix} \begin{bmatrix} r/c \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

Если исходная матрица A имеет размер m × n, то расширенная матрица S имеет размер (m + n) × (m + n). В обозначениях системы MATLAB эту систему можно описать так:

S * [r/c; x] = [b; z],

где z = zeros(n, 1).

Определение оптимального значения для масштабного множителя *c* основано на вычислении значений min(svd(A)) и norm(r), что требует больших объемов расчетов. Если значение *c* не указано явно в виде второго параметра, то по умолчанию применяется значение max(abs(A)))/1000. Можно

изменить это значение, если воспользоваться функцией spparms и задать другие значения для ключей aug_rel и aug_abs.

Операция решения систем линейных уравнений x = A\b автоматически формирует расширенную матрицу, определяя значение c по умолчанию.

Пример:

Рассмотрим простую систему из трех уравнений с двумя неизвестными.

$x_1 + 2x_2 = 5;$

$3x_1 = 6;$

$4x_2 = 7.$

Поскольку эта система небольших размеров, она может быть решена с использованием обычных конструкций системы MATLAB, в данном случае - полных матриц:

```
A = [ 1  2;  3  0;  0  4];
b = [5 6 7]';
x = A \ b
x =
    1.9592
    1.7041
```

Это решение по методу наименьших квадратов.

Подход на основе расширенной матрицы использует функцию spaugment:

```
S = spaugment(A, 1);,
```

которая формирует вспомогательную матрицу следующего вида:

```
full(S)
ans =
   1   0   0   1   2
   0   1   0   3   0
   0   0   1   0   4
   1   3   0   0   0
   2   0   4   0   0
```

В этом примере использовано значение масштабного множителя *c* = 1 и для удобства чтения распечатана полная матрица S. Даже в этом простом примере более половины элементов матрицы S равны нулю.

Вспомогательная система использует также расширенный вектор для правой части, который формируется в виде

```
y = [b; 0; 0];
```

в результате решение вычисляется с помощью решателя линейных уравнений

```
z = S \ y;,
```

а для решения применяется метод исключения Гаусса. В случае разреженных матриц при таком подходе используются меньшие объемы оперативной памяти.

Вычисленное решение имеет вид:

```
z =
   -0.36735
    0.12245
    0.18367
     1.9592
     1.7041
```

Здесь первые 3 компонента характеризуют вектор невязки r, а последние 2 компонента определяют решение x, которое совпадает с приведенным выше.

Сопутствующие функции: SPPARMS.

DELSQ — Формирование конечно-разностной аппроксимации лапласиана

Синтаксис:

```
L = delsq(G)
```

Описание:

Функция L = delsq(G) формирует разреженную матрицу, соответствующую пятиточечной конечно-разностной аппроксимации лапласиана на сетке G, которая может быть сформирована с помощью функции numgrid.

С применением функции delsq для решения уравнений в частных производных можно ознакомиться с помощью программы delsqdemo.

Сопутствующие функции: NUMGRID, DEL2, DELSQDEMO.

NUMGRID — Формирование двумерных сеток

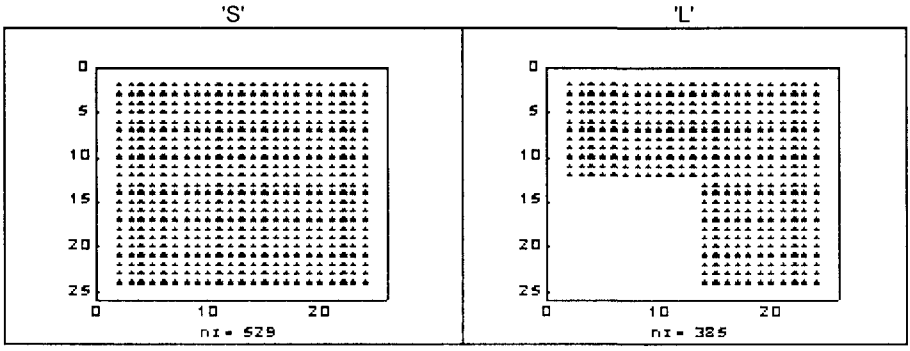
Синтаксис:

```
G = numgrid('R', n)
```

Описание:

Функция G = numgrid('R', n) задает точки двумерной сетки размера n×n, расположенной в области $\{-1 \le x \le 1,\ -1 \le y \le 1\}$, геометрия которой определяется символом 'R', который может принимать, значения, указанные на рис. 9.21.

Функция spy(numgrid('R', n) позволяет отобразить на графике вид этой области.



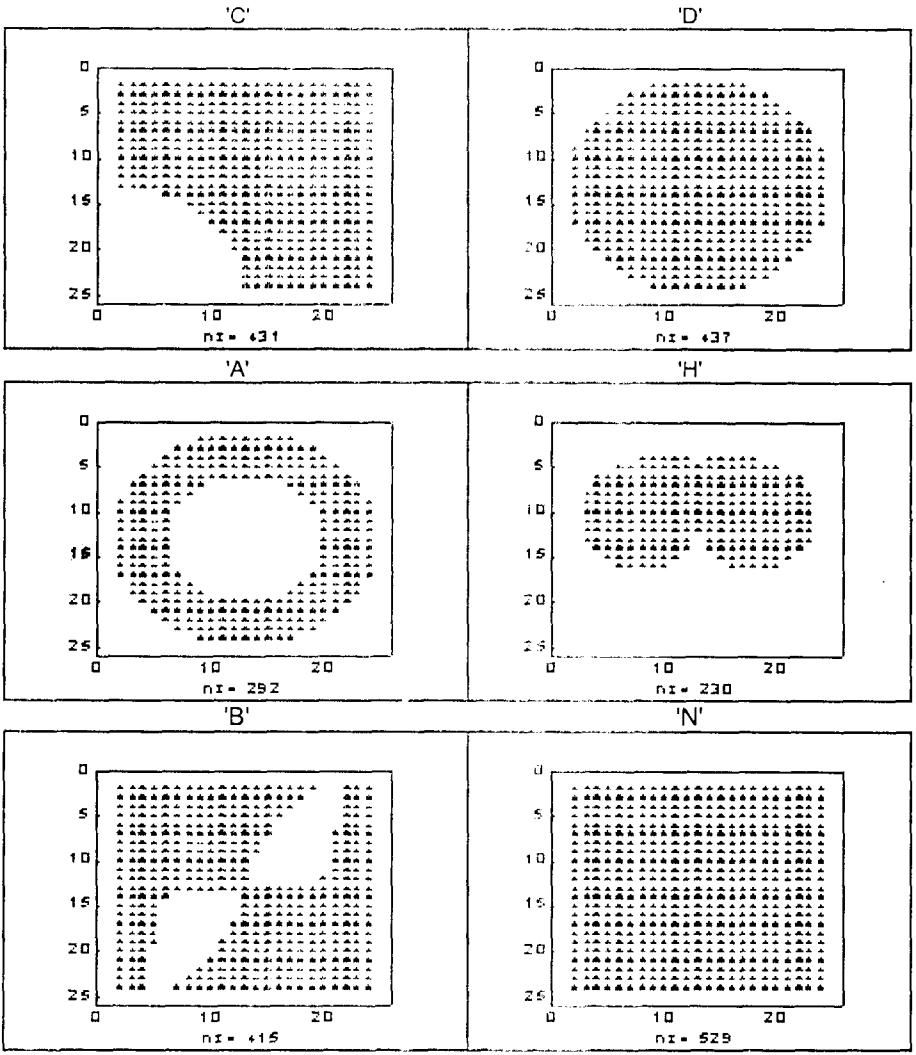


Рис. 9.21

Сетка 'N' отличается от сетки 'S' особой нумерацией точек.

Сопутствующие функции: NUMGRID, DEL2, DELSQDEMO.

10. ГРАФИЧЕСКИЕ КОМАНДЫ И ФУНКЦИИ

В состав системы MATLAB входит мощная графическая подсистема, которая поддерживает как средства визуализации двумерной и трехмерной графики, так и средства презентационной графики. Следует выделить несколько уровней работы с графическими объектами. В первую очередь это высокоуровневый интерфейс, включающий команды и функции, ориентированные на конечного пользователя и предназначенные для построения графиков в прямоугольных и полярных координатах, трехмерных поверхностей и линий уровня, гистограмм, столбцовых диаграмм и других специальных графиков. Графические команды высокого уровня автоматически контролируют масштаб, выбор цветов, не требуя манипуляций со свойствами графических объектов.

Соответствующий низкоуровневый интерфейс обеспечивается средствами дескрипторной графики, когда каждому графическому объекту в составе рисунка ставится в соответствие некоторый описатель (дескриптор), на который можно ссылаться при обращении к этому объекту. Используя дескрипторную графику, можно создавать графики со сносками, пояснениями, выделением отдельных элементов и т. п.

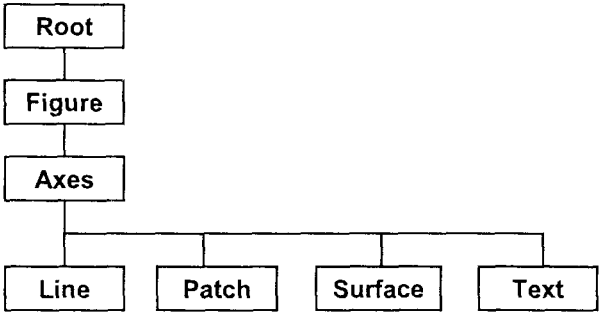
Из-за ограниченного объема справочного пособия в него не включено описание таких графических возможностей, как управление цветом, подсветкой, анимация, проектирование графического интерфейса. Заинтересованному читателю следует обратиться к документации по системе MATLAB, размещенной на компакт-диске и на WWW-сервере фирмы.

Элементы дескрипторной графики

Дескрипторная графика Handle Graphics - это объектно-ориентированная графическая система, которая поддерживает компоненты, необходимые для создания компьютерных графиков.

Объектами подсистемы дескрипторной графики являются базовые графические элементы, которые используются для визуализации данных. Каждому графическому объекту присваивается уникальный идентификатор, называемый *дескриптором*. Используя дескриптор, можно управлять *свойствами* графического объекта.

В системе MATLAB используется следующая иерархия графических объектов, связанных с визуализацией данных.



ROOT

Графический объект Root

Описание:

Графический объект Root находится в вершине иерархии и соответствует экрану компьютера. Существует единственный объект Root, который не имеет родителей, а все другие графические объекты являются его потомками. Прямым потомком этого объекта является графический объект Figure. Дескриптор объекта Root - 0. Этот объект создается в момент запуска системы MATLAB и не может быть удален. Тем не менее пользователь может управлять его свойствами.

В левом столбце приводимой ниже таблицы указаны свойства объекта Root с перечнем их допустимых значений и значений, принимаемых по умолчанию (в фигурных скобках), в правой части - текущие значения этих свойств. Читателю предоставляется возможность самому вникнуть в свойства графических объектов.

set(0)	get(0)
	CallbackObject = [ ]
Language	Language = russian
CurrentFigure	CurrentFigure = [1]
Diary: { on | off }	Diary = off
DiaryFile	DiaryFile = diary
Echo: { on | off }	Echo = off
ErrorMessage	ErrorMessage =
Format: { short | long | shortE | longE | shortG | longG | hex | bank | + | rational }	Format = shortG
FormatSpacing: { loose | compact }	FormatSpacing = loose
PointerLocation	PointerLocation = [65 51]
	PointerWindow = [0]
Profile: { on | off }	Profile = off
ProfileFile	ProfileFile =
ProfileCount	ProfileCount = []
ProfileInterval	ProfileInterval = [0.01]

RecursionLimit	RecursionLimit = [500]
ScreenDepth	ScreenDepth = [16]
	ScreenSize = [1 1 640 480]
ShowHiddenHandles: [on | {off}]	ShowHiddenHandles = off
Units: [inches | centimeters | normalized | points | pixels | characters]	Units = pixels
AutomaticFileUpdates: [on | off]	AutomaticFileUpdates = on
ButtonDownFcn	ButtonDownFcn =
Children	Children = [1]
Clipping: [{on} | off]	Clipping = on
CreateFcn	CreateFcn =
DeleteFcn	DeleteFcn =
BusyAction: [{queue} | cancel]	BusyAction = queue
HandleVisibility: [{on} | callback | off]	HandleVisibility = on
HitTest: [{on} | off]	HitTest = on
Interruptible: [{on} | off]	Interruptible = on
Parent	Parent = [ ]
Selected: [on | off]	Selected = off
SelectionHighlight: [{on} | off]	SelectionHighlight = on
Tag	Tag =
	Type = root
UIContextMenu	UIContextMenu = [ ]
UserData	UserData = [ ]
Visible: [{on} | off]	Visible = on

Сопутствующие команды: DIARY, ECHO, FIGURE, FORMAT, GCF, GET, SET.

FIGURE **Графический объект Figure**

Описание:

Графические объекты Figure соответствуют отдельным графическим окнам на экране терминала, где отображаются визуализируемые данные. В системе MATLAB нет ограничений на количество создаваемых окон, однако возможные ограничения могут быть обусловлены особенностями используемого компьютера. Графические объекты Figure являются прямыми потомками объекта Root.

Все высокоуровневые команды и функции, которые выводят графики, например plot, surf, contour, автоматически создают графический объект Figure, если он не существует. Если открыто много графических окон, то одно из них выделяется в качестве текущего графического объекта.

Дескриптором графического объекта Figure является номер графического окна.

В левом столбце приводимой ниже таблицы указаны свойства объекта Figure с перечнем их допустимых значений, в правой части - текущие значения этих свойств.

set(1)	get(1)
BackingStore: [{on} | off]	BackingStore = on
CloseRequestFcn	CloseRequestFcn = closereq
Color	Color = [1 1 1]
Colormap	Colormap = [(64 by 3) double array]
CurrentAxes	CurrentAxes = [1.00183]
	CurrentCharacter =
CurrentObject	CurrentObject = []
CurrentPoint	CurrentPoint = [0 0]
Dithermap	Dithermap = [(64 by 3) double array]
DithermapMode: [auto | {manual}]	DithermapMode = manual
	FixedColors = [(11 by 3) double array]
IntegerHandle: [{on} | off]	IntegerHandle = on
InvertHardcopy: [{on} | off]	InvertHardcopy = on
KeyPressFcn	KeyPressFcn =
MenuBar: [none | {figure}]	MenuBar = figure
MinColormap	MinColormap = [64]
Name	Name = 3-D Plots in Handle Graphics
NextPlot: [{add} | replace | replacechildren].	NextPlot = add
NumberTitle: [{on} | off]	NumberTitle = off
PaperUnits: [{inches} | centimeters | normalized | points]	PaperUnits = inches
PaperOrientation: [{portrait} | landscape]	PaperOrientation = portrait
PaperPosition	PaperPosition = [0.25 2.5 8 6]
PaperPositionMode: [auto | {manual}]	PaperPositionMode = manual
	PaperSize = [8.5 11]
PaperType: [{usletter} | uslegal | A0 | A1 | A2 | A3 | A4 | A5 | B0 | B1 | B2 | B3 | B4 | B5 | arch-A | arch-B | arch-C | arch-D | arch-E | A | B | C | D | E | tabloid]	PaperType = usletter
Pointer: [crosshair | fullcrosshair | {arrow} | ibeam | watch | topl | topr | botl | botr | left | top | right | bottom | circle | cross | fleur | custom]	Pointer = arrow
PointerShapeCData	PointerShapeCData = [(16 by 16) double array]
PointerShapeHotSpot	PointerShapeHotSpot = [1 1]
Position	Position = [5 156 436 282]
Renderer: [{painters} | zbuffer | OpenGL]	Renderer = zbuffer
RendererMode: [{auto} | manual]	RendererMode = auto
Resize: [{on} | off]	Resize = on
ResizeFcn	ResizeFcn =
	SelectionType = normal
ShareColors: [{on} | off]	ShareColors = on
Units: [inches | centimeters | normalized | points | {pixels} | characters]	Units = pixels

WindowButtonDownFcn	WindowButtonDownFcn =
WindowButtonMotionFcn	WindowButtonMotionFcn =
WindowButtonUpFcn	WindowButtonUpFcn =
WindowStyle: [{normal} | modal]	WindowStyle = normal
ButtonDownFcn	ButtonDownFcn =
Children	Children = [(15 by 1) double array]
Clipping: [{on} | off]	Clipping = on
CreateFcn	CreateFcn =
DeleteFcn	DeleteFcn =
BusyAction: [{queue} | cancel]	BusyAction = queue
HandleVisibility: [{on} | callback | off]	HandleVisibility = on
HitTest: [{on} | off]	HitTest = on
Interruptible: [{on} | off]	Interruptible = on
Parent	Parent = [0]
Selected: [on | off]	Selected = off
SelectionHighlight: [{on} | off]	SelectionHighlight = on
Tag	Tag =
	Type = figure
UIContextMenu	UIContextMenu = []
UserData	UserData = [4.00085 7.00049 9.00049 11.0005 13.0005]
Visible: [{on} | off]	Visible = on

Сопутствующие команды: AXES, CLOSE, CLF, GCF, ROOT-объект.

AXES **Графический объект Axes**

Описание:

Графический объект Axes задает область размещения графика в окне объекта Figure. Объект Axes является потомком для объекта Figure и в то же время родителем для графических объектов Line, Patch, Surface, Text, которые применяются для визуализации данных.

Все команды и функции, которые выводят графики, например plot, surf, contour, mesh, bar, автоматически создают графический объект Axes, если он не существует. Если в одном графическом окне создано несколько объектов Axes, то один из них выделяется в качестве текущего графического объекта.

Дескриптор графического объекта Axes может быть определен с помощью команды gca.

В левом столбце приводимой ниже таблицы указаны свойства объекта Axes с перечнем их допустимых значений, в правой части - текущие значения этих свойств.

```
set(gca)
  AmbientLightColor
  Box: [ on | {off} ]
  CameraPosition

  CameraPositionMode: [ {auto} | manual ]
  CameraTarget
  CameraTargetMode: [ {auto} | manual ]
  CameraUpVector
  CameraUpVectorMode: [ {auto} | manual ]
  CameraViewAngle
  CameraViewAngleMode: [ {auto} | manual ]
  CLim
  CLimMode: [ {auto} | manual ]
  Color

  ColorOrder
  DataAspectRatio
  DataAspectRatioMode: [ {auto} | manual ]
  DrawMode: [ {normal} | fast ]
  FontAngle: [ {normal} | italic | oblique ]
  FontName
  FontSize
  FontUnits: [ inches | centimeters | normalized |
                                {points} | pixels ]
  FontWeight: [ light | {normal} | demi | bold ]
  GridLineStyle: [ - | -- | {:} | -. | none ]
  Layer: [ top | {bottom} ]
  LineStyleOrder
  LineWidth
  NextPlot: [ add | {replace} | replacechildren ]
  PlotBoxAspectRatio
  PlotBoxAspectRatioMode: [ {auto} | manual ]
  Projection: [ {orthographic} | perspective ]
  Position
  TickLength
  TickDir: [ {in} | out ]
  TickDirMode: [ {auto} | manual ]
  Title
  Units: [ inches | centimeters | {normalized} | points |
                                pixels | characters ]
  View
  XColor
  XDir: [ {normal} | reverse ]
  XGrid: [ on | {off} ]
  XLabel
  XAxisLocation: [ top | {bottom} ]
```

```
get(gca)
  AmbientLightColor = [1 1 1]
  Box = off
  CameraPosition = [-121.971 -218.006
                                86.6025]
  CameraPositionMode = auto
  CameraTarget = [15 20 0]
  CameraTargetMode = auto
  CameraUpVector = [0 0 1]
  CameraUpVectorMode = auto
  CameraViewAngle = [10.3396]
  CameraViewAngleMode = auto
  CLim = [-6.32652 7.99662]
  CLimMode = auto
  Color = none
CurrentPoint = [ (2 by 3) double array]
ColorOrder = [ (6 by 3) double array]
  DataAspectRatio = [1.5 2 1]
  DataAspectRatioMode = auto
  DrawMode = normal
  FontAngle = normal
  FontName = Helvetica
  FontSize = [10]
  FontUnits = points

  FontWeight = normal
  GridLineStyle = :
  Layer = bottom
  LineStyleOrder = -
  LineWidth = [0.5]
  NextPlot = replace
  PlotBoxAspectRatio = [1 1 1]
  PlotBoxAspectRatioMode = auto
  Projection = orthographic
  Position = [0.07 0.45 0.6 0.5]
  TickLength = [0.01 0.025]
  TickDir = out
  TickDirMode = auto
  Title = [32.0001]
  Units = normalized

  View = [-37.5 30]
  XColor = [0 0 0]
  XDir = normal
  XGrid = on
  XLabel = [29.0001]
  XAxisLocation = bottom
```

XLim	XLim = [0 30]
XLimMode: [{auto} | manual]	XLimMode = auto
XScale: [{linear} | log]	XScale = linear
XTick	XTick = [0 10 20 30]
XTickLabel	XTickLabel = 0 10 20 30
XTickLabelMode: [{auto} | manual]	XTickLabelMode = auto
XTickMode: [{auto} | manual]	XTickMode = auto
YColor	YColor = [0 0 0]
YDir: [{normal} | reverse]	YDir = normal
YGrid: [on | {off}]	YGrid = on
YLabel	YLabel = [30.0001]
YAxisLocation: [{left} | right]	YAxisLocation = left
YLim	YLim = [0 40]
YLimMode: [{auto} | manual]	YLimMode = auto
YScale: [{linear} | log]	YScale = linear
YTick	YTick = [0 20 40]
YTickLabel	YTickLabel = 0 20 40
YTickLabelMode: [{auto} | manual]	YTickLabelMode = auto
YTickMode: [{auto} | manual]	YTickMode = auto
ZColor	ZColor = [0 0 0]
ZDir: [{normal} | reverse]	ZDir = normal
ZGrid: [on | {off}]	ZGrid = on
ZLabel	ZLabel = [31.0001]
ZLim	ZLim = [-10 10]
ZLimMode: [{auto} | manual]	ZLimMode = auto
ZScale: [{linear} | log]	ZScale = linear
ZTick	ZTick = [-10 0 10]
ZTickLabel	ZTickLabel = -10 0 10
ZTickLabelMode: [{auto} | manual]	ZTickLabelMode = auto
ZTickMode: [{auto} | manual]	ZTickMode = auto
ButtonDownFcn	ButtonDownFcn =
Children	Children = [16.0017]
Clipping: [{on} | off]	Clipping = on
CreateFcn	CreateFcn =
DeleteFcn	DeleteFcn =
BusyAction: [{queue} | cancel]	BusyAction = queue
HandleVisibility: [{on} | callback | off]	HandleVisibility = on
HitTest: [{on} | off]	HitTest = on

Interruptible: [{on} | off]	Interruptible = on
Parent	Parent = [1]
Selected: [on | off]	Selected = off
SelectionHighlight: [{on} | off]	SelectionHighlight = on
Tag	Tag =
	Type = axes
UIContextMenu	UIContextMenu = [ ]
UserData	UserData = [ ]
Visible: [{on} | off]	Visible = on

Для практического освоения свойств графического объекта Axes воспользуйтесь демонстрационным М-файлом hndlaxis. На рис. 10.1 показано графическое окно Handle Graphics and the Axis Object, которое позволяет управлять такими свойствами объекта Axes, как выбор линейной, логарифмической или полулогарифмической шкал по осям, использование сетки, направление изменения переменных по осям, цвет сетки и разметка осей.

Окно MiniCommand Window позволяет вводить собственные команды и управлять свойствами графика.

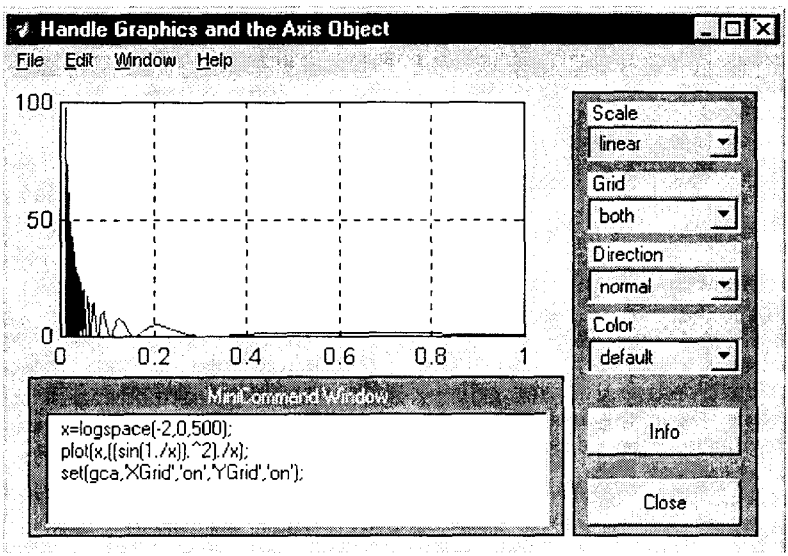


Рис. 10.1

Сопутствующие команды: AXIS, CLA, CLF, FIGURE, GCA, GRID, SUBPLOT, TITLE, XLABEL, YLABEL, ZLABEL, VIEW.

LINE **Графический объект Line**

Описание:

Графические объекты Line являются графическими примитивами, которые используются для создания двумерных и трехмерных графиков. Объект Line является потомком для объекта Axes.

Высокоуровневые команды и функции plot, plot3, contour создают графические объекты Line.

Дескриптор графического объекта Line может быть получен, например, с помощью функции h = plot(...).

В левом столбце приводимой ниже таблицы указаны свойства объекта Line с перечнем их допустимых значений, в правой части - текущие значения этих свойств для линии с дескриптором h(2).

set(h(2))	get(h(2))
Color	Color = [0 0 1]
EraseMode: [{normal} | background | xor | none]	EraseMode = normal
LineStyle: [{-} | -- | : | -. | none]	LineStyle = -
LineWidth	LineWidth = [0.5]
Marker: [+ | o | * | . | x | square | diamond | v | ^ | > | < | pentagram | hexagram | {none}]	Marker = none
MarkerSize	MarkerSize = [6]
MarkerEdgeColor: [none | {auto}] -or- a ColorSpec.	MarkerEdgeColor = auto
MarkerFaceColor: [{none} | auto] -or- a ColorSpec.	MarkerFaceColor = none
XData	XData = [(1 by 101) double array]
YData	YData = [(1 by 101) double array]
ZData	ZData = [ ]
ButtonDownFcn	ButtonDownFcn =
Children	Children = [ ]
Clipping: [{on} | off]	Clipping = on
CreateFcn	CreateFcn =
DeleteFcn	DeleteFcn =
BusyAction: [{queue} | cancel]	BusyAction = queue
HandleVisibility: [{on} | callback | off]	HandleVisibility = on
HitTest: [{on} | off]	HitTest = on
Interruptible: [{on} | off]	Interruptible = on
Parent	Parent = [2.00024]
Selected: [on | off]	Selected = off
SelectionHighlight: [{on} | off]	SelectionHighlight = on
Tag	Tag =
	Type = line
UIContextMenu	UIContextMenu = [ ]
UserData	UserData = [ ]
Visible: [{on} | off]	Visible = on

Для практического освоения свойств графического объекта Line воспользуйтесь демонстрационным М-файлом hndlgraf. На рис. 10.2 показано графическое окно Handle Graphics and Line Objects, которое позволяет управлять такими свойствами объекта Line, как тип, ширина линии, размер маркера, цвет линии.

Окно MiniCommand Window позволяет вводить собственные команды и управлять свойствами графика.

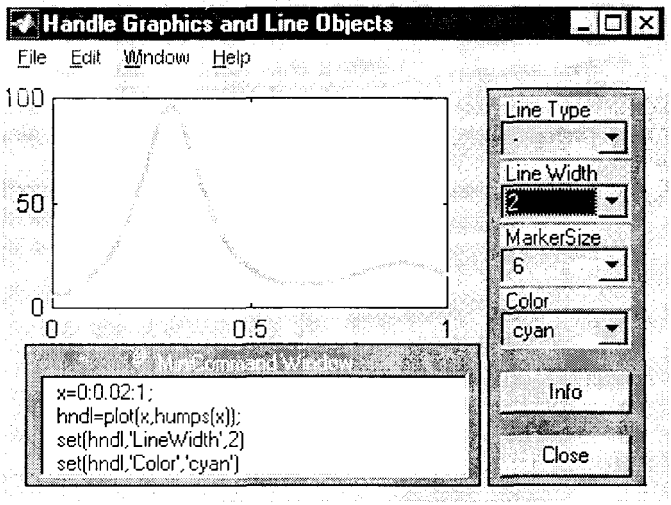


Рис. 10.2

Сопутствующие команды: PATCH, PLOT, PLOT3, TEXT.

PATCH **Графический объект Patch**

Описание:

Графический объект Patch - это один или несколько закрашенных многоугольников с выделенными границами. Объект Patch является потомком для объекта Axes.

Высокоуровневые команды и функции fill, fill3, contour, contour3 создают графические объекты Patch.

Дескриптор графического объекта Patch может быть получен, например, с помощью функции [c, h] = contour3(...).

В левом столбце приводимой ниже таблицы указаны свойства объекта Patch с перечнем их допустимых значений, в правой части - текущие значения этих свойств для объекта Patch с дескриптором h(1).

```
set(h(1))
  CData
  CDataMapping: [ direct | {scaled} ]
  FaceVertexCData

  EdgeColor: [ none | flat | interp ] -or- {a ColorSpec}.
  EraseMode: [ {normal} | background | xor | none ]
  FaceColor: [ none | flat | interp ] -or- {a ColorSpec}.
  Faces
  LineStyle: [ {-} | -- | : | -. | none ]
  LineWidth
  Marker: [ + | o | * | . | x | square | diamond | v | ^ | > | <
| pentagram | hexagram | {none} ]
  MarkerEdgeColor: [ none | {auto} | flat ] -or- a
ColorSpec.
  MarkerFaceColor: [ {none} | auto | flat ] -or- a
ColorSpec.
  MarkerSize
  Vertices
  XData
  YData
  ZData
 FaceLighting: [ none | {flat} | gouraud | phong ]
 EdgeLighting: [ {none} | flat | gouraud | phong ]
  BackFaceLighting: [ unlit | lit | {reverselit} ]
  AmbientStrength
  DiffuseStrength
  SpecularStrength
  SpecularExponent
  SpecularColorReflectance
  VertexNormals

  NormalMode: [ {auto} | manual ]

  ButtonDownFcn
  Children
  Clipping: [ {on} | off ]
  CreateFcn
  DeleteFcn
  BusyAction: [ {queue} | cancel ]
  HandleVisibility: [ {on} | callback | off ]
  HitTest: [ {on} | off ]
  Interruptible: [ {on} | off ]
  Parent
  Selected: [ on | off ]
  SelectionHighlight: [ {on} | off ]
  Tag
```

```
get(h(1))
 CData = [ (16 by 1) double array]
  CDataMapping = scaled
 FaceVertexCData = [ (16 by 1) double
array]
  EdgeColor = flat
  EraseMode = normal
  FaceColor = none
  Faces = [ (1 by 16) double array]
  LineStyle = -
  LineWidth = [0.5]
  Marker = none

  MarkerEdgeColor = auto

  MarkerFaceColor = none

  MarkerSize = [6]
 Vertices = [ (16 by 3) double array]
 XData = [ (16 by 1) double array]
 YData = [ (16 by 1) double array]
 ZData = [ (16 by 1) double array]
  FaceLighting = flat
  EdgeLighting = none
  BackFaceLighting = reverselit
  AmbientStrength = [0.3]
  DiffuseStrength = [0.6]
  SpecularStrength = [0.9]
  SpecularExponent = [10]
  SpecularColorReflectance = [1]
 VertexNormals = [ (16 by 3) double
array]
  NormalMode = auto

  ButtonDownFcn =
  Children = []
  Clipping = on
  CreateFcn =
  DeleteFcn =
  BusyAction = queue
  HandleVisibility = on
  HitTest = on
  Interruptible = on
  Parent = [1.0011]
  Selected = off
  SelectionHighlight = on
  Tag =
```

	Type = patch
UIContextMenu	UIContextMenu = []
UserData	UserData = [-6]
Visible: [{on} | off]	Visible = on

Сопутствующие команды: FILL, FILL3, LINE, SHADING, TEXT.

SURFACE **Графический объект Surface**

Описание:

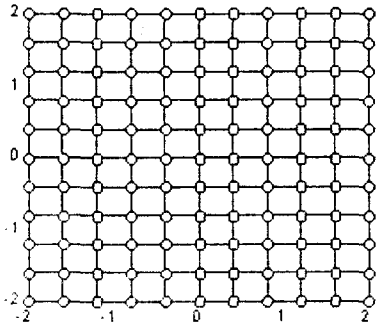
Графический объект Surface - это трехмерная визуализация массива данных, когда элемент массива определяет высоту точки над плоскостью x-y. Таким образом формируется трехмерная поверхность, состоящая из четырехугольников, вершины которых определяются массивом исходных данных. Поверхность может быть сплошной или интерполированной цветом либо представлять только сетку линий, соединяющих вершины.

Объект Surface является потомком объекта Axes.

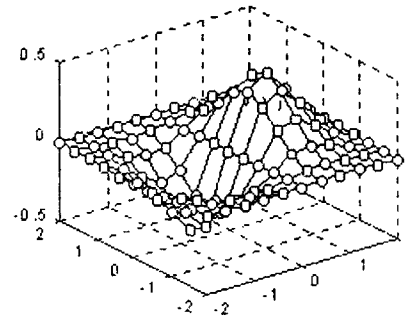
Высокоуровневые команды и функции pcolor, surf, mesh создают графический объект Surface.

Дескриптор графического объекта Line может быть получен, например, с помощью функции h = surf(...).

```
[X, Y] = meshgrid([-2:0.4:2]);
Z = X.*exp(-X.^2 - Y.^2);
% Рис. 10.3, а
fh = figure('Position', [350 275 400 300], 'Color', 'w');
ah = axes('Color', [.8 .8 .8], 'Xtick', [-2 -1 0 1 2], 'Ytick', [-2 -1 0 1 2]);
sh = surface('Xdata', X, 'Ydata', Y, 'Zdata', Z, ...
             'FaceColor', get(ah, 'Color') + 0.1, ...
             'EdgeColor', 'k', 'Marker', 'o', ...
             'MarkerFaceColor', [0.5 1 0.85]);
% Рис. 10.3, б
view(3), grid on
```



а



б

Рис. 10.3

В левом столбце приводимой ниже таблицы указаны свойства объекта Surface с перечнем их допустимых значений, в правой части - текущие значения этих свойств для поверхности с дескриптором sh.

set(sh)	get(sh)
CData	CData = [(3 by 3) double array]
CDataMapping: [direct | {scaled}]	CDataMapping = scaled
EdgeColor: [none | flat | interp] -or- {a ColorSpec}.	EdgeColor = [0 0 0]
EraseMode: [{normal} | background | xor | none]	EraseMode = normal
FaceColor: [none | {flat} | interp | texturemap] -or- a ColorSpec.	FaceColor = [0.9 0.9 0.9]
LineStyle: [{-} | -- | : | -. | none]	LineStyle = -
LineWidth	LineWidth = [0.5]
Marker: [+ | o | * | . | x | square | diamond | v | ^ | > | < | pentagram | hexagram | {none}]	Marker = o
MarkerEdgeColor: [none | {auto} | flat] -or- a ColorSpec.	MarkerEdgeColor = auto
MarkerFaceColor: [{none} | auto | flat] -or- a ColorSpec.	MarkerFaceColor = [0.5 1 0.85]
MarkerSize	MarkerSize = [6]
MeshStyle: [{both} | row | column]	MeshStyle = both
XData	XData = [(11 by 11) double array]
YData	YData = [(11 by 11) double array]
ZData	ZData = [(11 by 11) double array]
FaceLighting: [none | {flat} | gouraud | phong]	FaceLighting = flat
EdgeLighting: [{none} | flat | gouraud | phong]	EdgeLighting = none
BackFaceLighting: [unlit | lit | {reverselit}]	BackFaceLighting = reverselit
AmbientStrength	AmbientStrength = [0.3]
DiffuseStrength	DiffuseStrength = [0.6]
SpecularStrength	SpecularStrength = [0.9]
SpecularExponent	SpecularExponent = [10]
SpecularColorReflectance	SpecularColorReflectance = [1]
VertexNormals	VertexNormals = [(11 by 11 by 3) double array]
NormalMode: [{auto} | manual]	NormalMode = auto
ButtonDownFcn	ButtonDownFcn =
Children	Children = []
Clipping: [{on} | off]	Clipping = on
CreateFcn	CreateFcn =
DeleteFcn	DeleteFcn =
BusyAction: [{queue} | cancel]	BusyAction = queue
HandleVisibility: [{on} | callback | off]	HandleVisibility = on
HitTest: [{on} | off]	HitTest = on
Interruptible: [{on} | off]	Interruptible = on
Parent	Parent = [14.0002]

Selected: [on | off]	Selected = off
SelectionHighlight: [{on} | off]	SelectionHighlight = on
Tag	Tag =
	Type = surface
UIContextMenu	UIContextMenu = [ ]
UserData	UserData = [ ]
Visible: [{on} | off]	Visible = on

Для практического освоения свойств графического объекта Surface воспользуйтесь демонстрационным М-файлом graf3d. На рис. 10.4 показано графическое окно Handle Graphics and Line Objects, которое позволяет управлять такими свойствами объекта Line, как тип, ширина линии, размер маркера, цвет линии.

Окно MiniCommand Window позволяет вводить собственные команды и управлять свойствами графика.

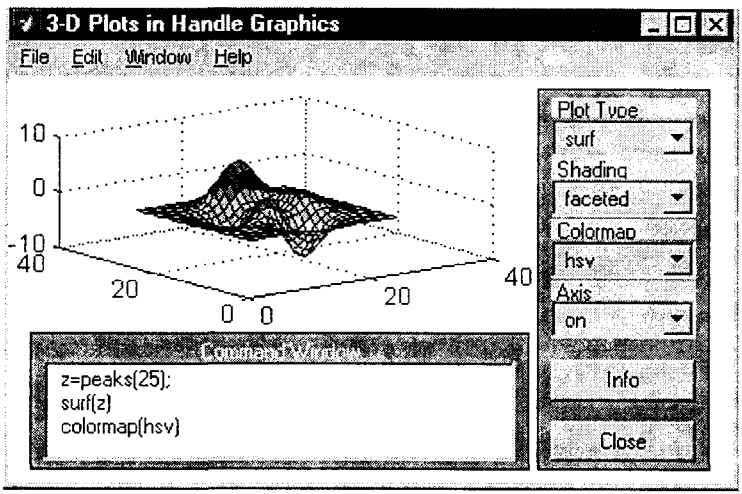


Рис. 10.4

Сопутствующие команды: LINE, PATCH, SHADING, SURF, TEXT.

TEXT **Графический объект Text**

Описание:

Графические объекты Text - это строки символов. Объект Text является потомком объекта Axes.

Высокоуровневые команды и функции title, xlabel, ylabel, zlabel, gtext создают графические объекты Text.

Дескриптор графического объекта Text может быть получен, например, с помощью функции h = title(...).

В левом столбце приводимой ниже таблицы указаны свойства объекта Text с перечнем их допустимых значений, в правой части - текущие значения этих свойств для объекта Text с дескриптором h.

set(h)	get(h)
Color	Color = [0 0 0]
EraseMode: [{normal} | background | xor | none]	EraseMode = normal
Editing: [on | off]	Editing = off
	Extent = [0.193548 1.01311 0.602151 0.0590164]
FontAngle: [{normal} | italic | oblique]	FontAngle = normal
FontName	FontName = Helvetica
FontSize	FontSize = [10]
FontUnits: [inches | centimeters | normalized | {points} | pixels]	FontUnits = points
FontWeight: [light | {normal} | demi | bold]	FontWeight = normal
HorizontalAlignment: [{left} | center | right]	HorizontalAlignment = center
Position	Position = [0.497297 1.02303 9.16025]
Rotation	Rotation = [0]
String	String = График функции
Units: [inches | centimeters | normalized | points | pixels | characters | {data}]	Units = data
Interpreter: [{tex} | none]	Interpreter = tex
VerticalAlignment: [top | cap | {middle} | baseline | bottom]	VerticalAlignment = bottom
ButtonDownFcn	ButtonDownFcn =
Children	Children = [ ]
Clipping: [{on} | off]	Clipping = off
CreateFcn	CreateFcn =
DeleteFcn	DeleteFcn =
BusyAction: [{queue} | cancel]	BusyAction = queue
HandleVisibility: [{on} | callback | off]	HandleVisibility = off
HitTest: [{on} | off]	HitTest = on
Interruptible: [{on} | off]	Interruptible = on
Parent	Parent = [1.0011]
Selected: [on | off]	Selected = off
SelectionHighlight: [{on} | off]	SelectionHighlight = on
Tag	Tag =
	Type = text
UIContextMenu	UIContextMenu = [ ]
UserData	UserData = [ ]
Visible: [{on} | off]	Visible = on

Сопутствующие команды: GTEXT, LINE, PATCH, TITLE, XLABEL, YLABEL, ZLABEL.

Элементарная графика

Элементарные графические функции системы MATLAB позволяют построить на экране и вывести на печатающее устройство следующие типы графиков: линейный, логарифмический, полулогарифмический, полярный.

Для каждого графика можно задать заголовок, нанести обозначение осей и масштабную сетку.

Двумерные графики

PLOT — **График в линейном масштабе**

Синтаксис:

```
plot(y)
plot(x, y)
plot(x, y, LineSpec)
plot(x1, y1, LineSpec1, x2, y2, LineSpec2, ...)
plot(..., 'PropertyName', PropertyValue, ...)
h =plot(...)
```

Описание:

Команда plot(y) строит график элементов одномерного массива y в зависимости от номера элемента; если элементы массива y комплексные, то строится график plot(real(y), imag(y)). Если Y - двумерный действительный массив, то строятся графики для столбцов; в случае комплексных элементов их мнимые части игнорируются.

Команда plot(x, y) соответствует построению графика функции, когда одномерный массив x соответствует значениям аргумента, а одномерный массив y - значениям функции. Когда один из массивов X или Y либо оба двумерные, реализуются следующие построения:

- если массив Y двумерный, а массив x одномерный, то строятся графики для столбцов массива Y в зависимости от элементов вектора x;
- если двумерным является массив X, а массив y одномерный, то строятся графики столбцов массива X в зависимости от элементов вектора y;
- если оба массива X и Y двумерные, то строятся зависимости столбцов массива Y от столбцов массива X.

Команда plot(x, y, LineSpec) позволяет выделить график функции, указав способ отображения линии, способ отображения маркера точек, цвет линий и маркера с помощью строковой переменной LineSpec, которая может включать до трех символов из следующей таблицы:

Тип линии		*Тип маркера*		*Цвет*	
Непрерывная	-	Точка	.	Желтый	y
Штриховая	--	Плюс	+	Фиолетовый	m
Двойной пунктир	:	Звездочка	*	Голубой	c
Штрихпунктирная	-.	Кружок	o	Красный	r
		Крест	x	Зеленый	g
		Квадрат	s	Синий	b
		Ромб	d	Белый	w
		Пятигранник	p	Черный	k
		Шестигранник	h		
		Стрелка вниз	v		
		Стрелка вверх	^		
		Стрелка влево	<		
		Стрелка вправо	>		

Если цвет линии не указан, он выбирается по умолчанию из шести первых цветов, с желтого до синего, повторяясь циклически.

Команда plot(x1, y1, LineSpec1, x2, y2, LineSpec2, ...) строит на одном графике несколько графических объектов Line для функций y1(x1), y2(x2), ..., определив для каждой из них свой способ отображения.

Команда plot(..., *'PropertyName'*, PropertyValue, ...) позволяет задать значения свойств графического объекта Line, соответствующего построенному графику.

Функция h = plot(...) возвращает вектор дескрипторов для всех графических объектов Line текущего объекта Axes.

Примеры:

Построим график функции y = exp(2*cos(x)) на отрезке [0 4π] с шагом π/15:

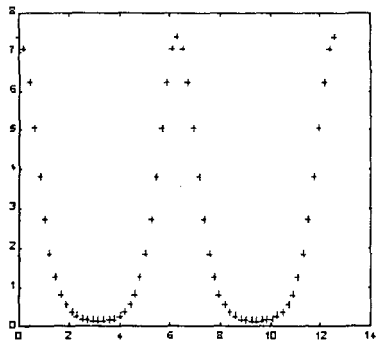
```
x = 0 : pi/15 : 4*pi;
y = exp(2*cos(x));
plot(x, y, 'b+')    % рис. а
```

График на рис. 10.5, *а* отображает значения одномерного массива y, состоящего из 61 элемента, как функцию элементов массива x, используя маркер "+".

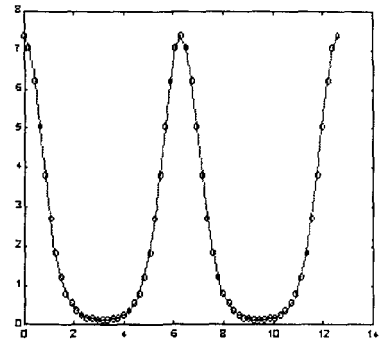
График на рис. 10.5, *б* соединяет точки сплошной линией синего цвета и использует маркер "o" черного цвета.

```
h = plot(x, y, 'b-', x, y, 'ko')    % рис. б
h =
    8.0001
    1.0013
```

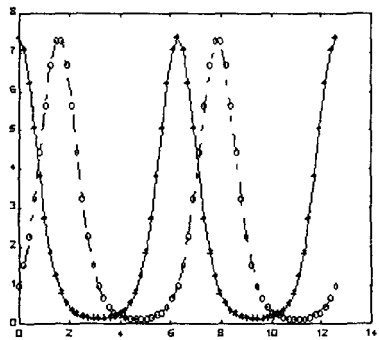
Этот график составлен из двух графических объектов Line.



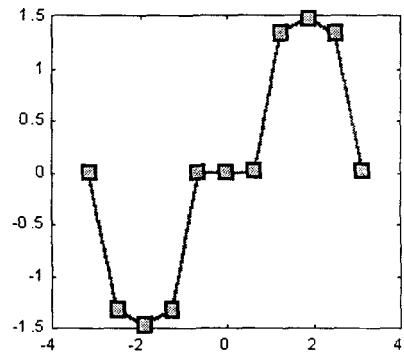
а



б



в



г

Рис. 10.5

На рис. 10.5, *в* построено два графика функций с различными спецификациями для линий и маркеров:

```
x = 0 : pi/15 : 4*pi;
y1 = exp(2*cos(x));
y2 = exp(2*sin(x));
h = plot(x, y1, '-k*', x, y2, '-.ko')     % рис. в
h =
    3.0004
    1.0015
```

На рис. 10.5, *г* построен график функции, для которого переопределен ряд свойств соответствующего объекта Line, а именно ширина линии - 2; цвет границы маркера - черный; цвет маркера - зеленый; размер маркера - 10:

```
x = -pi : pi/5 : pi;
y = tan(sin(x)) - sin(tan(x));
plot(x, y, '-rs', 'LineWidth', 2, ...
```

```
'MarkerEdgeColor', 'k', ...
'MarkerFaceColor', 'g', ...
'MarkerSize', 10)
```

Сопутствующие команды: AXES, AXIS, GRID, LOGLOG, PLOTEDIT, PLOTYY, SEMILOGX, SEMILOGY, XLABEL, XLIM, YLABEL,YLIM.

FPLOT, EZPLOT **Построение графиков функций**

Синтаксис:

```
fplot('<имя функции>', limits)
fplot('<имя функции>', limits, tol)
fplot('<имя функции>', limits, tol, n)
fplot('<имя функции>', limits, tol, n, LineSpec)
fplot('<имя функции>', limits, tol, n, LineSpec, p1, p2, ...)
[x, Y] = fplot(...)

ezplot('f(x)')
ezplot('f(x), [xmin xmax])
ezplot('f(x), [xmin xmax], fig)
```

Описание:

Команда fplot('*<имя функции>*', limits) строит график функции одной переменной в заданном интервале limits = [xmin xmax]. Если аргумент limits = = [xmin xmax ymin ymax], то принимается во внимание и заданный интервал по оси y. Аргумент '*<имя функции>*' - это либо имя M-файла, либо строка вида 'sin(x)', 'diric(x, 10)', '[sin(x) cos(x)]', которая может быть вычислена функцией eval. В том случае, когда вычисляемая функция многомерная, то формируется вектор-строка для каждого элемента вектора x. Например, если одновременно вычисляется 3 функции [f1(x), f2(x), f3(x)], то для входного вектора x = [x1; x2] будет сформирована следующая матрица Y:

```
f1(x1) f2(x1) f3(x1)
f1(x2) f2(x2) f3(x2)
```

Команда fplot('*<имя функции>*', limits, tol) строит график функции с относительной ошибкой tol (по умолчанию 2e-3), то есть с точностью 0.2 %. Таким образом, максимальное количество шагов по переменной x может составить (1/tol)+1.

Команда fplot('*<имя функции>*', limits, tol, n) строит график функции с минимальным количеством точек n+1 (по умолчанию n = 1). Таким образом, максимальный шаг по переменной x может составить (1/n)*(xmax-xmin).

Команда fplot('*<имя функции>*', limits, tol, n, LineSpec) строит график функции с заданной спецификацией линии LineSpec, которая задает тип линии, символ маркера и цвет по аналогии с функцией plot.

Аргументы tol, n, LineSpec могут включаться в список аргументов в любом порядке.

Команда fplot('<*имя функции*>', limits, tol, n, LineSpec, p1, p2, ...) позволяет передать параметры вычисляемой функции. Для использования по умолчанию аргументов tol, n, LineSpec достаточно указать пустые массивы [ ].

Функция [x, Y] = fplot(...) возвращает абсциссы и ординаты функции в виде одномерного массива x и, возможно, двумерного массива Y. График при этом не строится. Построение графика можно выполнить в дальнейшем с помощью функции plot(x, Y).

Замечание:

Новая функция реализует адаптивный механизм выбора шага, чтобы обеспечить презентационное качество графиков.

Функция fplot не использует средств дескрипторной графики.

Синтаксис новой команды и функции fplot отличается от используемого в ранних версиях.

Команда ezplot('f(x)') позволяет строить графики функций от одной переменной, записанных в виде символьного выражения по правилам языка MATLAB. Область определения такой функции - интервал [-2π 2π], принимаемый по умолчанию. В качестве заголовка графика используется имя или символьное выражение функции.

Команда ezplot('f(x)', [xmin xmax]) строит графики в указанном интервале.

Команда ezplot('f(x)', [xmin xmax], fig) строит графики в графическом окне с номером fig. В этом случае заголовок графика не выводится.

Пример:

Построим в четырех подокнах графики функций, используя различные способы обращения к функции fplot.

```
subplot(2, 2, 1), fplot('humps',  [0 1])
subplot(2, 2, 2), fplot('abs(exp(-j*x*(0:9))*ones(10, 1))',  [0 2*pi])
subplot(2, 2, 3), fplot('[tan(x), sin(x), cos(x)]', 2*pi*[-1 1 -1 1])
subplot(2, 2, 4), fplot('sin(1 ./ x)', [0.01 0.1], 1e-3)
```

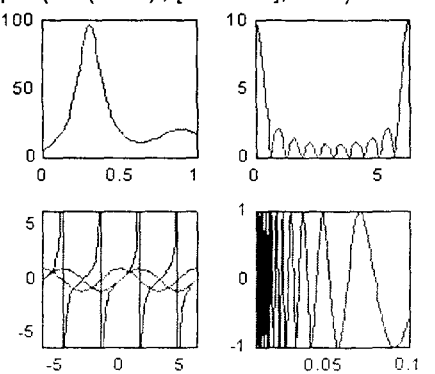


Рис. 10.6, а

Построим в четырех подокнах графики функций, используя различные способы обращения к функции fplot.

```
subplot(2, 2, 1), ezplot('sin(x)/ x')
subplot(2, 2, 2), ezplot('(x^2+x+1)/(2*x^2+5*x +11)')
subplot(2, 2, 3), ezplot('(real(exp(-j*t/2)))')
subplot(2, 2, 4), ezplot('humps(z)', [0 1])
```

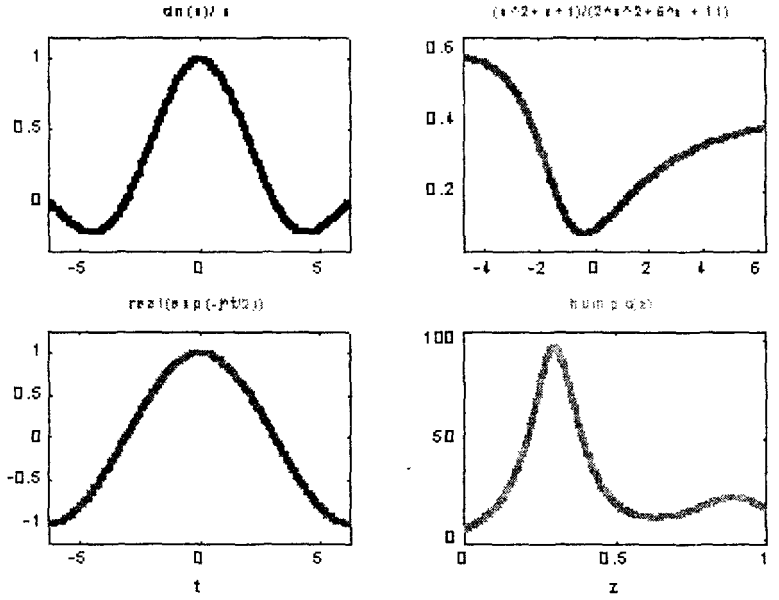


Рис. 10.6, б

Редактирование рис. 10.6, *б* выполнено в диалоговом редакторе plotedit системы MATLAB.

Сопутствующие команды: FEVAL, PLOT.

LOGLOG **График в логарифмическом масштабе**

Синтаксис:

```
loglog(y)
loglog(x, y)
loglog(x, y, LineSpec)
loglog(x1, y1, LineSpec1, x2, y2, LineSpec2, ...)
loglog(..., 'PropertyName', PropertyValue, ...)
h = loglog(...)
```

Описание:

Команды loglog(...) равносильны функциям plot, за исключением того, что они используют по обеим осям логарифмический масштаб вместо линейного.

Функция h = loglog(...) возвращает дескриптор графического объекта Line.

Замечание:

Если при построении более одного графика вы не указываете спецификацию цвета и типа линии, то эти характеристики выбираются последовательно из списков, определенных в свойствах текущего объекта Axes.

Примеры:

Построим график y = exp(x) в логарифмическом масштабе и установим толщину линии, равную 2:

```
x = logspace(-1, 2, 10);
hl = loglog(x, exp(x), 'LineWidth', 2),  hold on
hl =  1.0029
hm = loglog(x, exp(x),  'ks', 'MarkerFaceColor', 'y'), grid
hm = 3.0017
```

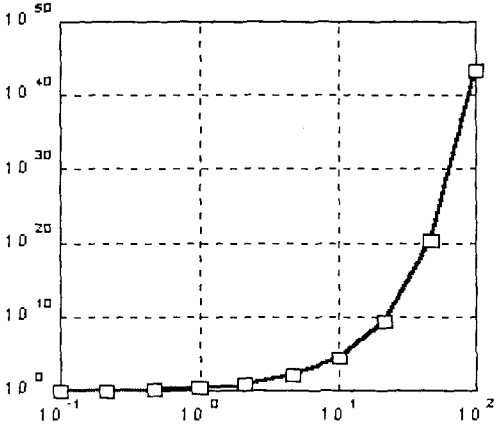


Рис. 10.7

Сопутствующие команды: LINE, PLOT, SEMILOGX, SEMILOGY.

SEMILOGX, SEMILOGY **График в полулогарифмическом масштабе**

Синтаксис:

```
semilogx(y)
semilogx(x, y)
semilogx(x, y, LineSpec)
semilogx(x1, y1, LineSpec1, x2, y2, LineSpec2, ...)
semilogx(..., 'PropertyName', PropertyValue, ...)
```

```
h = semilogx(...)
```

```
semilogy(...)
h = semilogy(...)
```

Описание:

Команды semilogx(...) используют логарифмический масштаб по оси x и линейный масштаб по оси y.

Команды semilogy(...) используют логарифмический масштаб по оси y и линейный масштаб по оси x.

Функции h = semilogx(...) и h = semilogy(...) возвращают дескрипторы графических объектов Line.

Пример:

Построим график y = exp(x) в полулогарифмическом масштабе по оси y и установим толщину линии, равную 2:

```
x = 0 : 10;
hl = semilogy(x, exp(x), 'LineWidth', 2),  hold on
hl =  1.0032
hm = semilogy(x, exp(x),  'ks', 'MarkerFaceColor', 'y'), grid
hm =  3.002
```

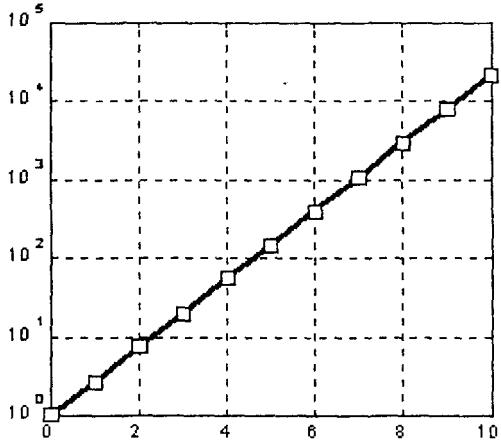


Рис. 10.8

Сопутствующие команды: PLOT, LOGLOG.

POLAR **График в полярных координатах**

Синтаксис:

```
polar(phi, rho)
polar(phi, rho, LineSpec)
h  = polar(...)
```

Описание:

Команды polar(...) реализуют построение графиков в полярных координатах, задаваемых углом phi и радиусом pho.

Пример:

Построим график функции rho = sin(2 * phi) * cos(2 * phi) в полярных координатах:

```
phi = 0 : 0.1 : 2*pi;  rho = sin(2 * phi).*cos(2 * phi);
hp = polar(phi, rho),  hold on
hp =    33
hm = polar(phi, rho, 'ks')
hm =    34
set(hp,  'LineWidth', 2)
set(hm, 'MarkerSize', 4,  'MarkerFaceColor', 'y')
```

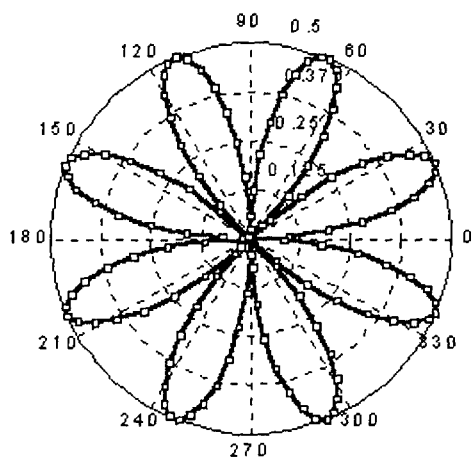


Рис. 10.9

Сопутствующие команды: PLOT, LOGLOG.

PIOTYY **График с двумя осями ординат**

Синтаксис:

```
plotyy(x1, y1, x2, y2)
plotyy(x1, y1, x2, y2, '<функция>')
plotyy(x1, y1, x2, y2, 'функция_1', 'функция_2)
[AX, h1, h2] = plotyy(...)
```

Описание:

Команда plotyy(x1, y1, x2, y2) строит графики функций, заданных массивами x1, y1 и x2, y2 с двумя осями ординат, левая из которых соответствует первому, а правая - второму графику, используя для построения функцию plot.

Команда plotyy(x1, y1, x2, y2, '*<функция>*') позволяет указать функцию, используемую для построения графиков, например plot, semilogx, semilogy, loglog, stem, или любую другую функцию, которая позволяет определить дескриптор объекта LINE или, иными словами, обращение к которой допускает форму h = function(x, y).

Команда plotyy(x1, y1, x2, y2, '*функция_1*', '*функция_2*) позволяет указать две разные функции для построения графиков соответственно для левой и правой осей ординат.

Функция [AX, h1, h2] = plotyy(...) возвращает вектор дескрипторов AX объектов Axes соответственно для левой AX(1) и правой AX(2) осей ординат, а также дескрипторы h1, h2 для соответствующих графиков.

Пример:

Построим графики функций y1 = sin(8*x)*exp(-x) и y2 = sin(2*x)*exp(-x)+1 со своими осями ординат для каждого графика.

```
x = 0 : 0.1 : 4;
[AX, h1, h2] = plotyy(x, sin(8*x).*exp(-x), x, sin(2*x).*exp(-x)+1), grid
set(h1, 'Marker', 's', 'MarkerSize', 5, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'y')
set(h2, 'Marker', 'o', 'MarkerSize', 5, 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'y')
```

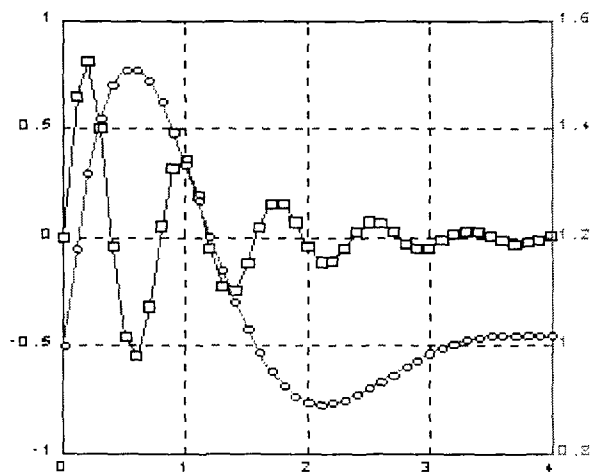


Рис. 10.10

Сопутствующие команды: PLOT, LOGLOG.

Трехмерные графики

В системе MATLAB предусмотрено несколько команд и функций для построения трехмерных графиков. Значения элементов числового массива рассматриваются как z-координаты точек над плоскостью, определяемой координатами x и y. Возможно несколько способов соединения этих точек. Первый из них - это построение одной линии в трехмерном пространстве или построение линий в сечениях (функция plot3), второй - построение поверхностей (функции mesh и surf). Поверхность, построенная с помощью функции mesh, - это сетчатая поверхность, ячейки которой имеют цвет фона, а их границы могут иметь цвет, который определяется свойством EdgeColor графического объекта Surface. Поверхность, построенная с помощью функции surf, - это сплошная поверхность, у которой может быть задан цвет не только границы, но и самой ячейки; последнее управляется свойством FaceColor графического объекта Surface.

Ниже приводится последовательность шагов, которая должна быть выполнена при построении трехмерных графиков.

Шаг	*Действие*	*Используемые функции*
1	Подготовить исходные данные	[X, Y] = meshgrid([-2 : 0.1 : 2]) Z = X .* exp(-X.^2 - Y.^2)
2	Выделить графическое окно и указать положение графика внутри окна	figure(1) subplot(2, 1, 2)
3	Вызвать функцию построения трехмерного графика	h = surf(Z)
4	Установить палитру и способ закраски поверхности	colormap autumn shading interp set(h, 'EdgeColor', 'k')
5	Установить точку просмотра	view([30, 25]) set(gca, 'CameraViewAngleMode', 'Manual')
6	Установить пределы и разметку осей	axis([5 15 5 15 -8 8]) set(gca, 'ZtickLabel', 'Negative | | Positive')
7	Установить масштабы по осям	set(gca, ' PlotBoxAspectRatio', [2 2 1])
8	Сделать надписи и пояснения к графику	xlabel(' x '), ylabel(' y '), zlabel(' z ') title('График функции') legend(...)
9	Вывести график на печать	set(gcf, ' PaperPositionMode', 'auto') print -dps2

PLOT3 **Построение линий и точек в трехмерном пространстве**

Синтаксис:

```
plot3(X, Y, Z)
plot3(X1, Y1, Z1, LineSpec1, X2, Y2, Z2, LineSpec2, ...)
plot3(..., 'PropertyName', PropertyValue, ...)
h = plot3(...)
```

Описание:

Команды plot3(...) являются трехмерными аналогами функции plot(...).

Команда plot3(X, Y, Z), где X, Y, Z - двумерные массивы одинакового размера, строит точки с координатами X(i, :), Y(i, :), Z(i, :) для каждого столбца и соединяет их прямыми линиями.

Команда plot3(X1, Y1, Z1, LineSpec1, X2, Y2, Z2, LineSpec2, ...) позволяет выделить график функции, указав способ отображения линии, способ отображения маркера точек, цвет линий и маркера с помощью строковой переменной LineSpec, которая может включать до трех символов из следующей таблицы:

Тип линии		*Тип маркера*		*Цвет*	
Непрерывная	-	Точка	.	Желтый	y
Штриховая	--	Плюс	+	Фиолетовый	m
Двойной пунктир	:	Звездочка	*	Голубой	c
Штрихпунктирная	-.	Кружок	o	Красный	r
		Крест	×	Зеленый	g
		Квадрат	s	Синий	b
		Ромб	d	Белый	w
		Пятигранник	p	Черный	k
		Шестигранник	h		
		Стрелка вниз	v		
		Стрелка вверх	^		
		Стрелка влево	<		
		Стрелка вправо	>		

Если цвет линии не указан, он выбирается по умолчанию из шести первых цветов, с желтого до синего, повторяясь циклически.

Команда plot3(..., *'PropertyName'*, PropertyValue, ...) позволяет задать значения свойств графического объекта Line, соответствующего построенному графику.

Функция h = plot(...) возвращает вектор дескрипторов для всех графических объектов Line текущего объекта Axes.

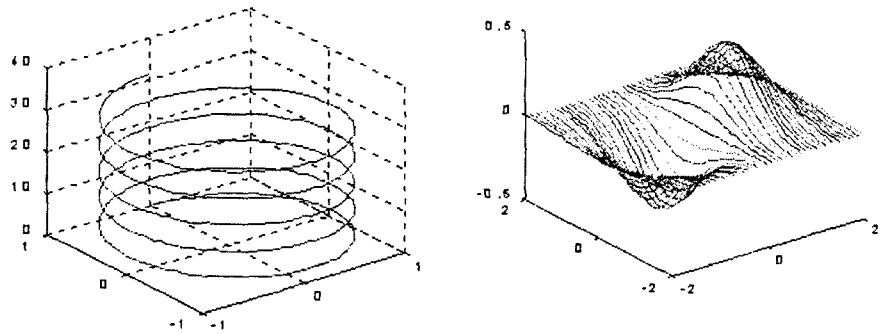
Примеры:

Построим график спирали в виде линии в трехмерном пространстве (рис. 10.11, *а*).

```
t = 0 : pi/50 : 10*pi;
plot3(sin(t), cos(t), t)
grid on
axis square
```

Построим график функции z = x * exp(-x2 - y2) в виде сечений в трехмерном пространстве (рис. 10.11, *б*).

```
[ X, Y ] = meshgrid([ -2 : 0.1 : 2 ]);
Z = X .* exp(- X .^ 2 - Y .^ 2);
plot3(X, Y, Z)
```



а *б*

Рис. 10.11

Сопутствующие команды: AXIS, LINE, MESH, PLOT, SURF, VIEW.

MESHGRID **Формирование прямоугольной сетки**

Синтаксис:

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

Описание:

Функция [X, Y] = meshgrid(x, y) задает сетку на плоскости x-y в виде двумерных массивов X, Y, которые определяются одномерными массивами x и y. Строки массива X являются копиями вектора x, а столбцы - копиями вектора y. Формирование таких массивов упрощает вычисление функций двух переменных, позволяя применять операции над массивами.

Функция [X, Y] = meshgrid(x) представляет собой упрощенную форму записи для функции [X, Y] = meshgrid(x, x).

Функция [X, Y, Z] = meshgrid(x, y, z) формирует трехмерную сетку для вычисления функций от трех переменных.

Функция meshgrid аналогична функции ndgrid, за исключением лишь того, что порядок следования первых двух аргументов у них различен, то есть функция [X, Y, Z] = meshgrid(x, y, z) - это то же самое, что [Y, X, Z] = = ndgrid(y, x, z). Из-за этого функция meshgrid лучше приспособлена для решения задач в декартовых координатах, а функция ndgrid - для многомерных задач. Применение функции meshgrid ограничено только двумерными и трехмерными сетками.

Пример:

Зададим двумерную сетку и построим функцию z = sin(r)/r , заданную на квадрате $-8 \le x \le 8$, $-8 \le y \le 8$.

```
[X, Y] = meshgrid(-8 : 0.5 : 8);
R = sqrt(X .^ 2  + Y .^ 2) + eps;
Z = sin (R) ./ R;
mesh(X, Y, Z)
```

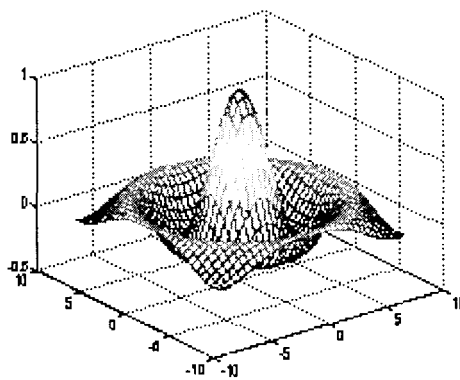


Рис. 10.12

Сопутствующие команды: NDGRID, SLICE, SURF.

MESH, MESHC, MESHZ **Трехмерная сетчатая поверхность**

Синтаксис:

mesh(X, Y, Z, C)	meshc(X, Y, Z, C)	meshz(X, Y, Z, C)
mesh(Z, C)	meshc(Z, C)	meshz(Z, C)
mesh(Z)	meshc(Z)	meshz(Z)
h = mesh(...)	h = meshc(...)	h = meshz(...)

Описание:

Команда mesh(X, Y, Z, C) выводит на экран сетчатую поверхность для значений массива Z, определенных на множестве значений массивов X и Y. Цвета узлов поверхности задаются массивом C. Цвета ребер определяются

свойством EdgeColor объекта Surface. Можно задать одинаковый цвет для всех ребер, определив его в виде вектора [r g b] интенсивности трех цветов - красного, зеленого, синего. Если определить спецификацию none, то ребра не будут прорисовываться. Если определить спецификацию flat, то цвет ребер ячейки определяется цветом того узла, который был первым при обходе этой ячейки. Поскольку одни и те же ребра обходятся несколько раз, то цвета будут замещаться. Если определить спецификацию interp, то будет реализована линейная интерполяция цвета между вершинами ребра.

Применение функции shading (затенение сетчатой поверхности) после обращения к функции mesh изменяет спецификации свойств EdgeColor и FaceColor согласно следующей таблице.

Свойство	*Применяемая функция*		
	mesh	shading flat	shading interp
EdgeColor	flat	flat	interp
FaceColor	Цвет фона	Цвет фона	Цвет фона

Команда mesh(Z, C) выполняет ту же операцию, но при этом по осям x, y используются отсчеты X = 1:n, Y = 1:m, где [m, n] = size(Z).

Команда mesh(Z) выводит поверхность, как в предшествующем случае, но в качестве массива цвета используется массив C = Z, то есть цвет в этом случае пропорционален высоте поверхности.

Функция h = mesh(...) возвращает дескриптор графического объекта Surface.

Группа команд meshc(...) в дополнение к трехмерным поверхностям строит проекцию линий постоянного уровня.

Группа команд meshz(...) в дополнение к трехмерным поверхностям строит плоскость отсчета на нулевом уровне, закрывая поверхность, лежащую ниже этого уровня.

Функции h = meshc(...), h = meshz(...) возвращают дескриптор h для графического объекта Surface.

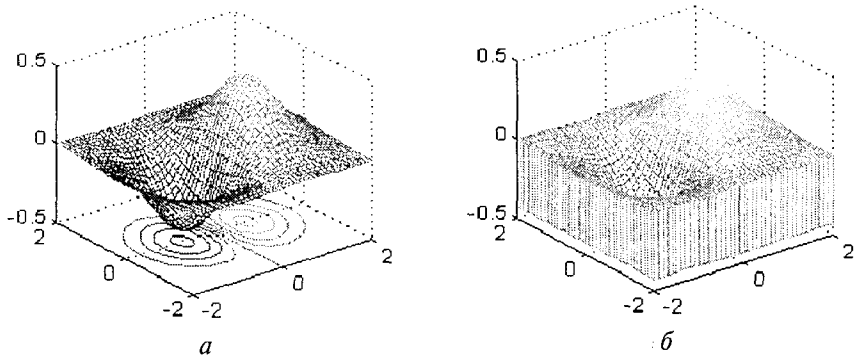
Примеры:

Построим трехмерную поверхность функции $z = x * \exp(-x^2 - y^2)$ с проекциями линий постоянного уровня (рис. 10.13, *а*).

```
[ X, Y ] = meshgrid([ -2 : 0.1 : 2 ]);
Z = X .* exp(- X .^ 2 - Y .^ 2);
meshc(X, Y, Z)
```

На рис. 10.13, *б* построена та же функция с пьедесталом отсчета

```
meshz(X, Y, Z).
```



а *б*

Рис. 10.13

Сопутствующие команды: SURF, WATERFALL.

HIDDEN **Удаление невидимых линий**

Синтаксис:

hidden on
hidden off
hidden

Описание:

Команда hidden on включает режим удаления невидимых линий. Этот режим используется по умолчанию.

Команда hidden off выключает режим удаления невидимых линий.

Команда hidden реализует переключение с одного режима на другой.

Пояснение:

Команды hidden изменяют значение свойства FaceColor объекта Surface: hidden on присваивает свойству FaceColor цвет Color объекта Axes либо цвет Color объекта Figure, если свойство Color объекта Axes имеет значение none.

Сопутствующие команды: MESH, SHADING.

SURF, SURFC **Трехмерная сплошная поверхность**

Синтаксис:

surf(X, Y, Z, C)	surfc(X, Y, Z, C)
surf(Z, C)	surfc(Z, C)
surf(Z)	surfc(Z)
h = surf(...)	h = surfc(...)

Описание:

Команда surf(X, Y, Z, C) выводит на экран сплошную поверхность с ребрами для значений массива Z, определенного на множестве значений массивов X и Y. Цвет ячейки задается массивом C. Цвет ребер - черный,

определяется свойством EdgeColor, специфицированным как [0 0 0]. Можно задать одинаковый цвет для всех ребер, определив его в виде вектора [r g b] интенсивности трех цветов - красного, зеленого, синего. Если определить спецификацию none, то ребра не будут прорисовываться.

Применение функции shading (затенение сплошной поверхности) после обращения к функции surf изменяет спецификации свойств EdgeColor и FaceColor графического объекта Surface согласно следующей таблице.

Свойство	*Применяемая функция*		
	surf	shading flat	shading interp
EdgeColor	[0 0 0]	none	none
FaceColor	flat	flat	interp

Команда surf(Z, C) выполняет ту же операцию, но при этом по осям x, y используются отсчеты X = 1:n, Y = 1:m, где [m, n] = size(Z).

Команда surf(Z) выводит поверхность, используя в качестве цвета массив C = Z, что означает задание цвета пропорционально высоте поверхности.

Группа команд surfc(...) в дополнение к трехмерным поверхностям строит проекцию линий постоянного уровня.

Функции h = surf(...), h = surfc(...) возвращают дескриптор h для графического объекта Surface.

Примеры:

Построим сплошную поверхность с ребрами черного цвета для функции $z = x*\exp(-x^2 - y^2)$.

```
[ X, Y ] = meshgrid([ -2 : 0.1 : 2 ]);
Z = X .* exp(- X .^ 2 - Y .^ 2);
surf(X, Y, Z, gradient(Z))
colormap pink
brighten(0.5)
```

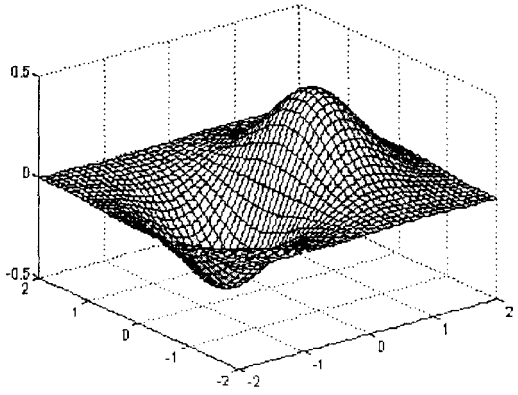


Рис. 10.14

Рассмотрим пример сферы, которая раскрашена в соответствии с матрицей Адамара (Hadamard), часто используемой в теории кодирования сигналов и составленной только из двух чисел 1 и -1.

```
k = 5; n = 2 ^ k - 1;
[X, Y, Z]=sphere(n);
C = hadamard(2 ^ k);
surf(X, Y, Z, C);
colormap([1 1 1; 2/3 2/3 2/3])
colorbar
```

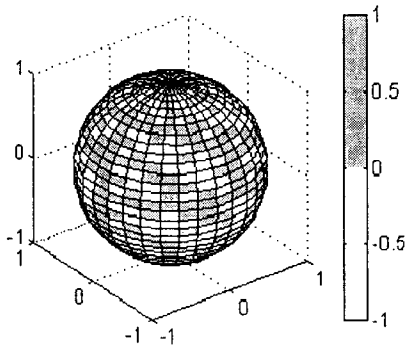


Рис. 10.15

Алгоритм:

В общем виде для задания поверхности можно использовать два независимых параметра i и j, которые изменяются непрерывно, например в прямоугольнике $1 \le i \le m$, $1 \le j \le n$; тогда поверхность будет определяться тремя функциями x(i, j), y(i, j), z(i, j). Когда i и j целые числа, они задают прямоугольную сетку с целочисленными значениями для узлов. Функции x(i, j), y(i, j), z(i, j) становятся двумерными массивами X, Y, Z размера m×n. Четвертая функция - цвет ячеек поверхности c(i, j) задает четвертую матрицу C.

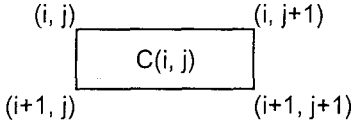
Каждая точка сетчатой поверхности имеет в общем случае четырех соседей, как показано на следующей схеме.

```
          i-1, j
            |
i, j-1 —  i, j  — i, j+1
            |
          i+1, j
```

Такая прямоугольная сетка приводит к разбиению поверхности на ячейки, ограниченные четырьмя ребрами. Каждый внутренний узел поверхности имеет четырех соседей, узел на границе - трех, узел в углу поверхности - двух.

Цвет такой сетчатой поверхности может быть задан двумя способами: либо цветом одной из вершин, либо цветом в центре ячейки.

Рассмотрим, как в этом случае действует функция закраски shading. Если задано значение shading interp, то цвет ячейки определяется как билинейная функция местных координат. Если задано значение shading faceted (принято по умолчанию) или shading flat, то цвет ячейки постоянен и определяется цветом верхней левой вершины, как показано на следующей схеме.



Сопутствующие команды: AXIS, CAXIS, COLORMAP, CONTOUR, MESH, PCOLOR, SHADING, VIEW.

SHADING **Затенение поверхностей**

Синтаксис:

shading faceted
shading flat
shading interp

Описание:

Команды группы shading устанавливают способ затенения графических объектов Surface и Patch, которые создаются при использовании функций mesh, surf, pcolor, fill и fill3.

Команда shading faceted устанавливает равномерную раскраску ячеек с нанесением черных граней. Такое затенение поверхности часто оказывается наиболее эффективным и принято по умолчанию.

Команда shading flat устанавливает раскраску каждой ячейки или грани определенными цветами, которые зависят от цвета узлов сетки.

Команда shading interp устанавливает раскраску каждой ячейки или грани цветами, которые определяются билинейной интерполяцией цветов в узлах сетки.

Алгоритм:

Команды группы shading устанавливают требуемые значения свойств EdgeColor и FaceColor графических объектов Surface и Patch. Устанавливаемые значения зависят от того, является ли трехмерный график сетчатой (функция mesh) или сплошной (функция surf) поверхностью.

```
[X, Y] = meshgrid(-3 : 1/8 : 3);
Z = peaks(X, Y);
h = surfl(X, Y, Z),
h =  2.001
shading interp, colormap(gray)
axis([-3 3 -3 3 -8 8])
```

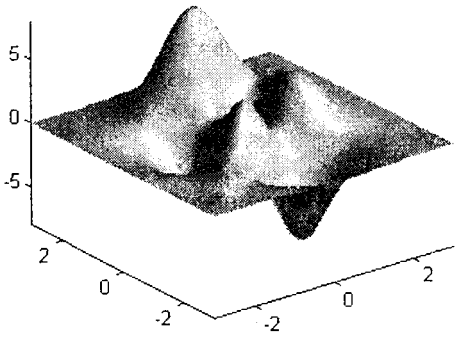


Рис. 10.16

Сопутствующие команды: FILL, FILL3, HIDDEN, MESH, PATCH, PCOLOR, SURF.

BRIGHTEN **Управление яркостью**

Синтаксис:

```
brighten(beta)
brighten(h, beta)
newmap = brighten(beta)
newmap = brighten(cmap, beta)
```

Описание:

Команды группы brighten позволяют управлять яркостью палитры.

Команда brighten(beta) заменяет текущую палитру более яркой (0 <beta <1) или более темной (-1 < beta < 0), сохраняя при этом цвета. Две последовательные команды brighten(beta) и brighten(-beta) восстанавливают исходную яркость палитры.

Команда brighten(h, beta) изменяет яркость палитры для всех потомков графического объекта Figure с дескриптором h.

Функция newmap = brighten(beta) возвращает матрицу новой палитры, не изменяя при этом текущую палитру.

Функция newmap = brighten(cmap, beta) возвращает матрицу новой палитры, полученную из палитры cmap и, не изменяя при этом текущую палитру.

Алгоритм:

Для изменения яркости палитры элементы ее матрицы возводятся в степень γ:

$$\gamma = \begin{cases} 1-\beta, & \beta > 0 \\ \dfrac{1}{1+\beta}, & \beta < 0 \end{cases}$$

Сопутствующие команды: COLORMAP, RGBPLOT.

SURFL **Сплошная поверхность с подсветкой**

Синтаксис:

```
surfl(Z)
surfl(X, Y, Z)
surfl(..., s)
surfl(..., s, k)
h = surfl(...)
surfl(..., 'light')
h = surfl(..., 'light')
```

Описание:

Графические команды surfl(...) позволяют вывести на экран поверхность с учетом моделей рассеяния, отражения и зеркального эффекта.

Команды surfl(Z), surfl(X, Y, Z) создают трехмерные поверхности, используя принятые по умолчанию коэффициенты модели подсветки и фиксированное положение источника света. Массивы X, Y, Z определяют координаты поверхности.

Команды surfl(..., s) позволяют указать направление на источник света с помощью вектора s = [Sx, Sy, Sz] в декартовых координатах или вектора s = [az, elev] в сферических координатах. По умолчанию направление на источник сдвинуто на 45° против часовой стрелки относительно текущего направления просмотра. По умолчанию текущее направление просмотра имеет азимут az = -37.5° и возвышение elev = 30°.

Команды surfl(..., s, k) позволяют управлять параметрами рассеяния, отражения и зеркального эффекта, используя вектор k = [ka, kd, ks, shine], который учитывает эффекты отраженного света ka, диффузного отражения kd, зеркального отражения ks и зеркального блеска shine. По умолчанию вектор k имеет значения [0.55 0.6 0.4 10].

Функции h = surfl(...) возвращают дескриптор h для графического объекта Surface.

Команды в форме surfl(..., 'light') создают трехмерные поверхности, используя свойства графического объекта Light. Это дает несколько иные результаты по сравнению с используемой по умолчанию функцией surfl(..., 'cdata'), которая изменяет цвет поверхности с учетом только коэффициентов отражения.

Функции surfl(..., 'light') возвращают векторный дескриптор h для графических объектов Surface и Light.

Команда surfl(X, Y, Z) использует значения параметров по умолчанию.

Команды surfl(Z, ...) строят графики, не учитывая численных значений массивов X и Y.

Замечание:

При визуализации поверхностей с подсветкой следует использовать палитры с линейным изменением интенсивности: gray, copper, bone, pink.

Порядок размещения точек в массивах позволяет отображать различные точки поверхности; если, например, требуется просмотреть поверхность с противоположной стороны, то следует использовать обращение в форме surfl(X', Y', Z').

Из-за того что алгоритм surfl вычисляет нормали к поверхности, необходимо, чтобы входные матрицы имели размер по крайней мере 3×3.

Примеры:

Построим изображение функции peaks, используя подсветку.

```
[X, Y] = meshgrid(-3 : 1/8 : 3);
Z = peaks(X, Y);
h = surfl(X, Y, Z)
h =  2.001
colormap copper,  shading interp
```

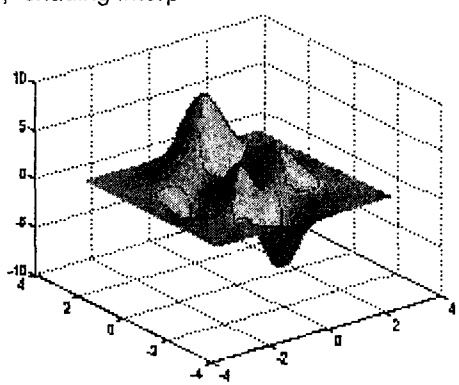


Рис. 10.17

Сопутствующие команды: COLORMAP, LIGHT, SHADING.

Задание осей координат

AXES **Создать графический объект Axes**

Синтаксис:

```
axes
axes('PropertyName', PropertyValue, ...)
axes(h)
h = axes(...)
```

Описание:

Команда axes предназначена для создания графических объектов Axes в текущем графическом окне Figure, используя значения свойств по умолчанию.

Команда axes('*PropertyName*', PropertyValue, ...) создает графический объект Axes с указанными значениями свойств; если какое-либо значение не специфицировано явно, то принимается его значение по умолчанию.

Команда axes(h) делает графический объект Axes с дескриптором h текущим. При этом в свойствах текущего графического объекта Figure дескриптор h указывается первым в списке свойства Children, а свойству CurrentAxes присваивается значение дескриптора h. Текущий графический объект Axes является родителем для графических объектов Image, Light, Line, Patch, Surface, Text.

Функция h = axes(...) возвращает дескриптор текущего объекта Axes.

Замечание:

При работе с системой MATLAB графический объект Axes создается автоматически при обращении к командам, которые порождают графические объекты: Image, Light, Line, Patch, Surface, Text. Команда допускает в качестве входных аргументов пары {'*PropertyName*', PropertyValue}, а также массивы записей и массивы ячеек.

Для получения текущих значений свойств объекта Axes следует использовать команду get, а для указания новых значений - команду set. Команда gca предназначена для получения значения дескриптора текущего объекта Axes.

Команда axis, но не axes, обеспечивает упрощенный способ управления некоторыми свойствами, связанными с масштабированием и изображением осей координат.

В то время как основное назначение объекта Axes создать систему координат для построения графиков, свойства этого объекта позволяют управлять способами вывода данных на график.

Свойство Stretch-to-fill

По умолчанию MATLAB растягивает оси координат так, чтобы заполнить весь объем параллелепипеда, определяемого значениями свойства Position. Это порождает графики, которые используют все доступное пространство параллелепипеда. Однако в этом случае некоторые трехмерные объекты (типа сферы) оказываются искаженными и для них лучше применять специальные трехмерные коэффициенты сжатия.

Свойство Stretch-to-fill (растянуть-заполнить) активизировано, если свойства DataAspectRatioMode, PlotBoxAspectRatioMode, CameraViewAngleMode имеют значение auto, используемое по умолчанию. Если значения одного или нескольких свойств устанавливались вручную, то свойство Stretch-to-fill оказывается неактивным.

Примеры:

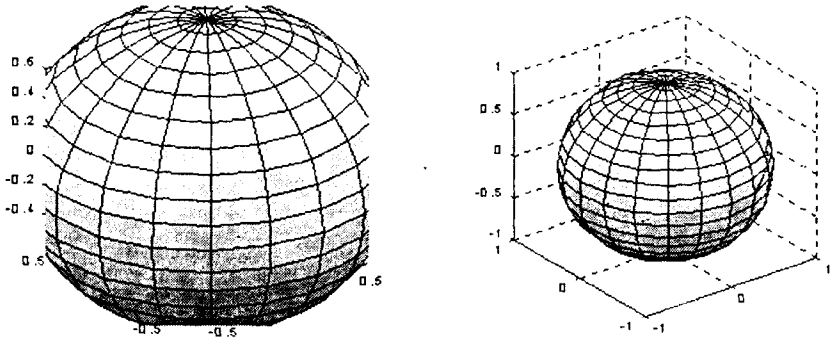
Проиллюстрируем свойство Stretch-to-fill на следующем примере.

Построим сферу, не затрагивая свойств DataAspectRatioMode, PlotBoxAspectRatioMode, CameraViewAngleMode, которые по умолчанию имеют значение auto, то есть сохраняя свойство Stretch-to-fill активным. Сфера на рис. 10.18, *а* растянута по всем осям и полностью заполняет выделенный параллелепипед.

```
sphere
set(gca, 'DataAspectRatio', [1 1 1], ...
'PlotBoxAspectRatio', [1 1 1], 'ZLim', [-0.6 0.6])
colormap pink
```

Изменим значение свойства CameraViewAngle, сделав тем самым свойство Stretch-to-fill неактивным (рис. 10.18, *б*):

```
sphere
set(gca, 'CameraViewAngle', get(gca, 'CameraViewAngle')+5)
colormap pink
```



а *б*

Рис. 10.18

Другой пример иллюстрирует возможность создания множественных объектов в рамках одного графического окна.

```
h(1) = axes('Position', [0 0 1 1]); sphere
h(2) = axes('Position', [0 0 0.4 0.6]); sphere
h(3) = axes('Position', [0 0.5 0.5 0.5]); sphere
h(4) = axes('Position', [0.5 0 0.4 0.4]); sphere
h(5) = axes('Position', [0.5 0.5 0.5 0.3]); sphere
set(h, 'Visible', 'off')
colormap pink
```

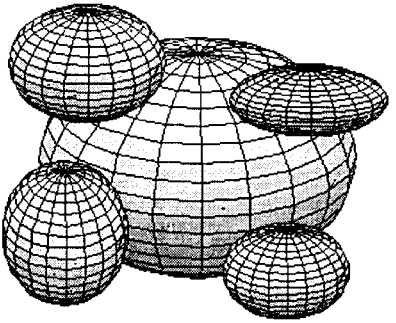


Рис. 10.19

Сопутствующие команды: AXIS, CLA, FIGURE, GCA, SUBPLOT.

AXIS **Масштабирование осей и вывод на экран**

Синтаксис:

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax])
v = axis

 axis auto          axis ij          axis square      axis off
 axis manual        axis xy          axis vis3d       axis on
 axis tight         axis equal       axis normal
 axis fill          axis image
[mode, visibility, direction] = axis('state')
```

Описание:

Команды axis позволяют манипулировать отдельными свойствами графического объекта Axes.

Команда axis([xmin xmax ymin ymax]) устанавливает масштаб по осям x, y для текущего графического объекта Axes.

Команда axis([xmin xmax ymin ymax zmin zmax]) устанавливает масштаб по осям x, y, z для активного графического окна.

Функция v = axis возвращает вектор-строку масштабов по осям для активного графика. Если график двумерный, то v имеет 4 компонента; если трехмерный - 6 компонентов. Возвращаемые значения соответствуют свойствам XLim, Ylim, ZLim объекта Axes.

Команда axis('auto') вызывает механизм автоматического установления предельных значений по осям; при этом такой механизм может быть применен к отдельным осям или сочетанию осей. Например, команда axis 'auto x' автоматически вычисляет пределы только по оси x, а команда axis 'auto yz' - по осям y и z.

Команды axis manual и axis(axis) фиксируют текущие значения масштабов для последующих графиков, как если бы был включен режим hold. В этом случае свойства XLimMode, YLimMode, ZLimMode устанавливаются вручную.

Команда axis tight устанавливает такие коэффициенты масштабирования по осям, что диапазон изменения переменных по всем осям становится одинаковым. Это отличается от действия команды axis equal, которая устанавливает одинаковые расстояния между метками.

Команда axis fill приводит пределы по осям к диапазону изменения данных.

Команда axis ij перемещает начало отсчета в левый верхний угол, при этом ось i расположена вертикально и значения возрастают в направлении сверху вниз, ось j расположена горизонтально и значения возрастают в направлении слева направо.

Команда axis xy возвращает декартову систему координат; начало отсчета находится в нижнем левом углу; ось x горизонтальна и размечается слева направо, ось y вертикальна и размечается снизу вверх.

Команда axis equal устанавливает масштаб, который обеспечивает одинаковые расстояния между метками по осям x, y, z.

Команда axis image равносильна команде axis equal, за исключением лишь того, что размер по осям строго привязан к данным.

Команда axis square устанавливает одинаковый диапазон и одинаковые расстояния между метками по осям.

Команда axis vis3d фиксирует свойство DataAspectRatio и делает неактивным свойство Stretch-to-fill, чтобы поддержать вращение трехмерного объекта.

Команда axis normal восстанавливает полноразмерный масштаб, отменяя масштабы, установленные командами axis square и axis equal.

Команда axis off снимает с осей их обозначения и маркеры.

Команда axis on восстанавливает на осях их обозначения и маркеры.

Функция [mode, visibility, direction] = axis('state') возвращает 3 строки, соответствующие некоторым свойствам объекта Axes:

mode = 'auto' | 'manual'.
visibility = 'on' | 'off'.
direction = 'xy' | 'ij'.

Например, значение параметра mode, равное 'auto', означает, что свойства XLimMode, YLimMode, ZLimMode принимают значение auto; параметр visibility соответствует свойству Visible, а параметр direction - выбору направления осей.

По умолчанию эти параметры принимают значения ['auto', 'on', 'xy'].

Алгоритм:

В командах, где указываются максимальные и минимальные значения переменных, свойства XLim, Ylim, Zlim устанавливаются в соответствии с этими значениями; при этом свойства XLimMode, YLimMode, ZLimMode принимают значение auto.

Следующая таблица иллюстрирует изменение значений свойств объекта Axes при выполнении различных команд группы axis.

Axes Property	axis equal	axis normal	axis square	axis tightequal
DataAspectRatio	[1 1 1]	not set	not set	[1 1 1]
DataAspectRatioMode	manual	auto	auto	manual
PlotBoxAspectRatio	[3 4 4]	not set	[1 1 1]	auto
PlotBoxAspectRatioMode	manual	auto	manual	auto
Stretch-to-fill	disabled	active	disabled	disabled

Сопутствующие команды: AXES, GET, GRID, SET, SUBPLOT.

SUBPLOT **Создание подокон**

Синтаксис:

subplot(m, n, p)
subplot(mnp), subplot mnp
subplot(h)
subplot('Position', [left bottom width height])
h = subplot(...)

Описание:

Команды subplot производят разбивку графического окна на несколько прямоугольных подокон; каждому подокну соответствует свой графический объект Axes. График выводится в то подокно, которое является в данный момент текущим.

Команды subplot(m, n, p), subplot(mnp) или subplot mnp, где m, n, p (mnp) - 3 цифры, производят разбивку графического окна на несколько подокон; значение m указывает количество подокон по горизонтали, n - по вертикали, а p - номер подокна, куда будет выводиться очередной график. Эти же команды могут использоваться для перехода от одного подокна к другому.

Команда subplot(h) делает оси с дескриптором h текущими.

Команда subplot('Position', [left bottom width height]) создает координатные оси в позиции, определяемой вектором [left bottom width height] в нормализованных координатах, изменяющихся в диапазоне от 0.0 до 1.0, где left = = xmin/xmax, bottom = ymin/ymax, width = 1- left, height = 1 - bdtom.

Функция h = subplot(...) возвращает дескриптор графического объекта Axes.

Замечание:

Команда subplot(111) не совпадает с командой subplot(1,1,1) и оставлена только для совместимости с предшествующими версиями. Ее действие заключается в том, что она присваивает свойству NextPlot объекта Figure значение replace, это приводит к тому, что следующая графическая команда выполняет команду clf reset, удаляя потомков объекта Figure и создавая новый объект Figure в позиции

по умолчанию. Эта команда, если к ней обращаются в форме функции, не возвращает дескриптор, а фиксирует ошибку выходного аргумента.

Пример:

Выделим в графическом окне 4 подокна и построим графики функций, заданных параметрически.

```
t = 0 : pi/20 : 2*pi;
[x, y] = meshgrid(t);

h221 = subplot(221)
plot(sin(t), cos(t), 'LineWidth', 2)
axis equal
h221 =  1.0011

h222 = subplot(222)
z = sin(x) + cos(y);
plot(t, z)
axis([0  2*pi  -2  2])
h222 =  3.0001

h223 = subplot(223)
z = sin(x) .* cos(y);
plot(t, z)
axis([0  2*pi  -1  1])
h223 =  45.0001

h224 = subplot(224)
z = sin(x) .^ 2  - cos(y) .^ 2;
plot(t, z)
axis([0  2*pi  -1  1])
h224 =  87.0001
```

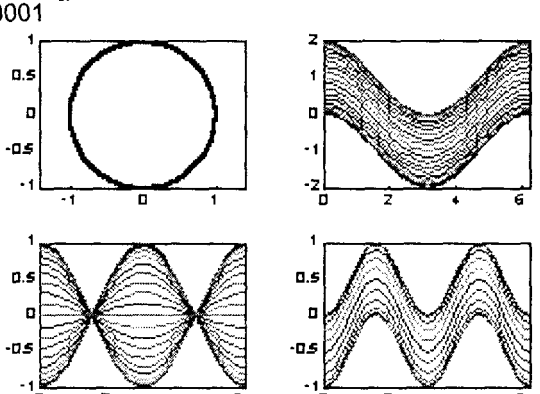


Рис. 10.20

Сопутствующие команды: AXES, CLA, CLF, FIGURE, GCA.

DASPECT, PBASPECT

Коэффициенты масштабирования данных и координатных осей

Синтаксис:

daspect
daspect([aspect_ratio])
daspect('mode')
daspect('auto')
daspect('manual')
daspect(h, ...)

pbaspect
pbaspect([aspect_ratio])
pbaspect('mode')
pbaspect('auto')
pbaspect('manual')
pbaspect(h, ...)

Определения:

Коэффициенты масштабирования данных по осям определяются свойством DataAspectRatio графического объекта Axes.

Коэффициенты масштабирования координатных осей определяются свойством PlotBoxAspectRatio графического объекта Axes.

Описание:

Команда daspect возвращает значения свойства DataAspectRatio для текущего графического объекта Axes.

Команда daspect([aspect_ratio]) устанавливает значения свойства DataAspectRatio для текущего графического объекта Axes в виде вектора из трех компонентов. Например, вектор вида [1 1 3] означает, что единица измерения по осям x, y соответствует трем единицам измерения по оси z.

Команда daspect('mode') возвращает текущее состояние свойства DataAspectRatioMode, которое может принимать значения auto или manual.

Команда daspect('auto') устанавливает свойство DataAspectRatioMode в состояние auto.

Команда daspect('manual') устанавливает свойство DataAspectRatioMode в состояние manual.

Команда daspect(h, ...) определяет или устанавливает значения свойств DataAspectRatio и DataAspectRatioMode для графического объекта Axes с дескриптором h.

Команда pbaspect возвращает значения свойства PlotBoxAspectRatio для текущего графического объекта Axes.

Команда pbaspect([aspect_ratio]) устанавливает значения свойства PlotBoxAspectRatio для текущего графического объекта Axes в виде вектора из трех компонентов. Например, вектор вида [1 1 1], принимаемый по умолчанию, означает, что область размещения графика - это куб, хотя если активизировано свойство Stretch-to-fill, то изображение может отличаться от куба.

Команда pbaspect('mode') возвращает текущее состояние свойства PlotBoxAspectRatioMode, которое может принимать значения auto или manual.

Команда pbaspect('auto') устанавливает свойство PlotBoxAspectRatioMode в состояние auto.

Команда pbaspect('manual') устанавливает свойство PlotBoxAspectRatioMode в состояние manual.

Команда pbaspect(h, ...) определяет или устанавливает значения свойств PlotBoxAspectRatio и PlotBoxAspectRatioMode для графического объекта Axes с дескриптором h.

Пояснения:

Если значение свойства DataAspectRatioMode соответствует auto, то система устанавливает коэффициенты масштабирования для свойства DataAspectRatio такими, чтобы изображение объекта заполняло графическое окно в максимальной степени. Для реалистичного изображения трехмерного объекта следует устанавливать значение этого свойства равным [1 1 1].

При ручном задании значений для свойств DataAspectRatio и DataAspectRatioMode свойство Stretch-to-fill становится неактивным. Это означает, что присваивание даже текущего значения в форме daspect(daspect) может вызвать искажение графика.

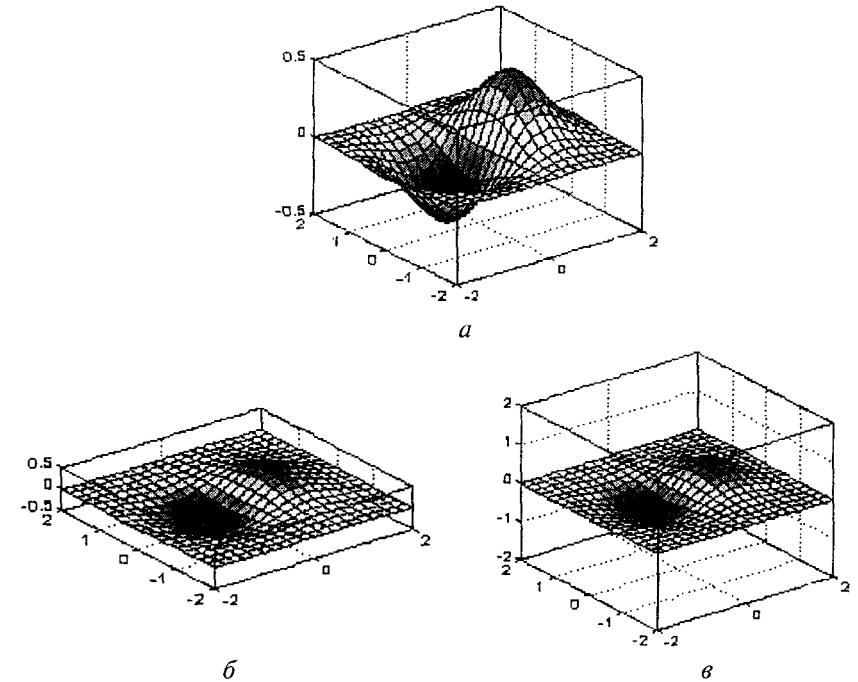
Если значение свойства PlotBoxAspectRatioMode соответствует auto, то система устанавливает коэффициенты масштабирования для свойства PlotBoxAspectRatio равными [1 1 1]. Однако это может измениться, если будет выполнена ручная установка свойств DataAspectRatio, CameraViewAngle, Xlim, Ylim, Zlim.

При ручном задании значений для свойств PlotBoxAspectRatio и PlotBoxAspectRatioMode свойство Stretch-to-fill становится неактивным. Это означает, что присваивание даже текущего значения в форме pbaspect(pbaspect) может вызвать искажение графика.

Примеры:

Рассмотрим трехмерную поверхность, чтобы проиллюстрировать влияние масштабирования данных и осей на изображаемую поверхность.

```
[x, y] = meshgrid([-2 : .2 : 2]);
z = x.*exp(-x.^2 - y.^2);
surf(x, y, z)
% Рис. 10.21 а
pbaspect
ans =    1    1    1
daspect
ans =    4    4    1
```



а

б *в*

Рис. 10.21

Для того чтобы проиллюстрировать связь между свойствами PlotBoxAspectRatio и DataAspectRatio, установим для коэффициента масштабирования данных DataAspectRatio значение [1 1 1] и проверим значение PlotBoxAspectRatio (рис. 10.21, *б*).

```
daspect([1 1 1]);
pbaspect
ans = 4 4 1
```

В этом случае вектор свойства изменился, чтобы приспособиться к новому масштабу данных.

Теперь установим для коэффициента масштабирования осей PlotBoxAspectRatio значение [1 1 1] и проверим значение DataAspectRatio (рис. 10.21, *в*).

```
pbaspect([1 1 1]);
daspect
ans =    1    1    1
```

Обратим внимание, что теперь изменились пределы изменения по осям, поскольку специфицированы оба свойства- DataAspectRatio и PlotBoxAspectRatio.

Команда pbaspect позволяет дезактивировать свойство Stretch-to-fill (рис. 10.22).

```
h211 = subplot(211);
surf(x, y, z), shading interp
h212 = subplot(212);
surf(x, y, z),
pbaspect(h211, 'manual')
shading interp
```

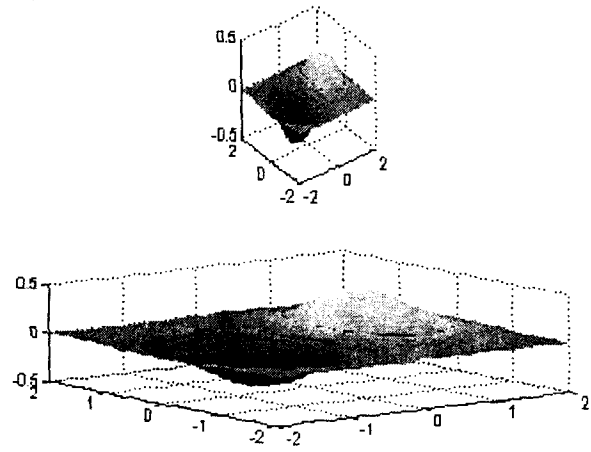


Рис. 10.22

Сопутствующие команды: AXIS, XLIM, YLIM, ZLIM.

XLIM, YLIM, , ZLIM **Допустимые диапазоны по осям координат**

Синтаксис:

xlim	ylim	zlim
xlim([xmin xmax])	ylim([ymin ymax])	zlim([zmin zmax])
xlim('mode')	ylim('mode')	zlim('mode')
xlim('auto')	ylim('auto')	zlim('auto')
xlim('manual')	ylim('manual')	zlim('manual')
xlim(h, ...)	ylim(h, ...)	zlim(h, ...)

Определения:

Допустимые диапазоны по осям координат соответствуют значениям свойств XLim, YLim, ZLim графического объекта Axes.

Описание:

Команда *lim возвращает значение свойства *Lim для текущего графического объекта Axes.

Команда *lim([*min *max]) устанавливает значения свойства *Lim для текущего графического объекта Axes в виде вектора из двух компонентов.

Команда *lim('mode') возвращает текущее состояние свойства *LimMode, которое может принимать значения auto или manual.

Команда *lim('auto') устанавливает свойство *LimMode в состояние auto.

Команда daspect('manual') устанавливает свойство *LimMode в состояние manual.

Команда *limt(h, ...) определяет или устанавливает значения свойств *Lim и *LimMode для графического объекта Axes с дескриптором h.

Пояснения:

Если значение свойства *LimMode соответствует auto, используемому по умолчанию, то система сама устанавливает допустимые диапазоны, которые приближенно соответствуют отображаемым данным. При ручном задании одного из этих диапазонов значение свойства *LimMode переключается на manual.

Следует отметить, что высокоуровневые графические команды типа plot, surf изменяют свойства *Lim, *LimMode. Если установлены предельные диапазоны и они должны быть сохранены при добавлении новых графиков, необходимо использовать команду hold.

Сопутствующие команды: AXIS, DASPECT, PBASPECT.

GRID — **Нанесение сетки**

Синтаксис:

grid on
grid off
grid

Описание:

Команда grid on наносит координатную сетку на текущие оси.

Команда grid off удаляет координатную сетку.

Команда grid выполняет роль переключателя с одной функции на другую.

Команды группы grid переключают состояние свойств 'XGrid', 'YGrid', 'ZGrid' объекта Axes со значения on на off и наоборот.

Сопутствующие команды: TITLE, XLABEL, YLABEL.

BOX — **Нанесение контура области**

Синтаксис:

box on
box off
box

Описание:

Команда box on рисует контур параллелепипеда, в котором размещается трехмерный объект.

Команда box off удаляет контур.

Команда box выполняет роль переключателя с одной функции на другую.

Команды группы box переключают состояние свойства 'Box' объекта Axes со значения on на off и наоборот.

Сопутствующие команды: AXES, GRID.

HOLD — **Управление режимом сохранения текущего графического окна**

Синтаксис:

```
hold on
hold off
hold
```

Описание:

Команда hold on включает режим сохранения текущего графика и свойств объекта Axes, так что последующие команды приведут к добавлению новых графиков в графическом окне.

Команда hold off выключает режим сохранения графика.

Команда hold реализует переключение с одного режима на другой.

Пояснение:

Команды hold воздействуют на значения свойства NextPlot объектов Figure и Axes:

- hold on присваивает свойству NextPlot для текущих объектов Figure и Axes значение add;
- hold off присваивает свойству NextPlot для текущих объектов Figure и Axes значение replace.

Сопутствующие команды: AXES, FIGURE, ISHOLD, NEWPLOT.

ZOOM — **Управление масштабом графика**

Синтаксис:

```
zoom on
zoom off
zoom
zoom out
zoom reset
zoom xon
zoom yon
zoom(factor)
zoom(fig, option)
```

Описание:

Команда zoom on включает режим интерактивного масштабирования текущего объекта Figure. Теперь при нажатии левой клавиши мыши вблизи интересующей вас точки графика масштаб окна увеличивается в 2 раза; при нажатии правой клавиши масштаб в 2 раза уменьшается. Удерживая левую клавишу, можно выделить нужную область графика, рисуя растягивающийся прямоугольник.

Для однокнопочной мыши увеличение масштаба реализуется простым нажатием кнопки, а уменьшение - одновременным нажатием кнопки и клавиши Shift.

Двойное нажатие на кнопку мыши в области объекта Axes возвращает этот объект в начальное состояние.

Команда zoom off выключает режим интерактивного масштабирования.

Команда zoom out возвращает график в исходное состояние.

Команда zoom reset запоминает текущий график как исходный. В этом случае команда zoom out или двойное нажатие на кнопку мыши будет возвращать текущий график в новое исходное состояние.

Команды zoom xon и zoom yon позволяют выполнять масштабирование только по выбранной оси.

Команда zoom(factor) позволяет устанавливать коэффициент масштабирования для графика не влияя на интерактивный режим. Значения параметра factor, большие 1, увеличивают масштаб, а значения, меньшие 1, должны быть равны 1/factor, и их можно использовать только после увеличения масштаба.

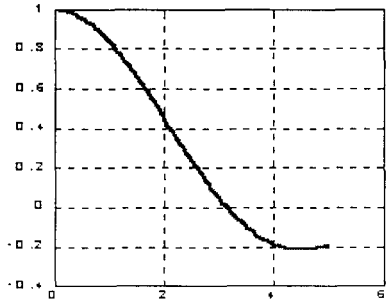
Команда zoom(fig, option) позволяет установить коэффициент масштабирования для графического окна с номером fig.

Пример:

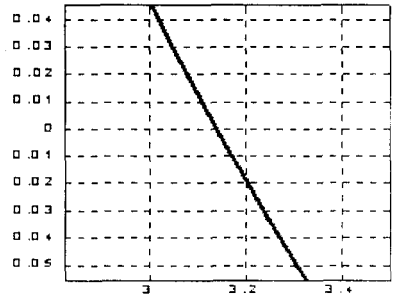
Рассмотрим, как можно найти решение уравнения sin(x)/x = 0 на отрезке $0 < x \le 5$ с помощью интерактивного масштабирования.

```
x = [0 : 0.05 : 5] + eps;
 plot(x, sin(x)./x)
 grid
 zoom
```

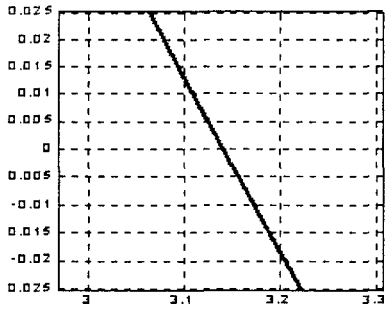
Для перехода от рис. 10.23, *а* к рис. 10.23, *б* была выделена область вблизи нулевого значения функции, что позволило локализовать решение в области $3 < x < 3.2$. При переходе от рис. 10.23, *б* к рис. 10.23, *в* масштаб был увеличен вдвое простым нажатием левой кнопки мыши; это позволило локализовать решение в области $3.1 < x < 3.2$. Дальнейшие манипуляции дали возможность локализовать решение в области $3.1415 < x < 3.1420$ (рис. 10.23, *г*).



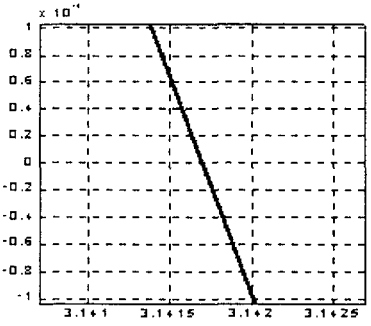
а



б



в



г

Рис. 10.23

Сопутствующие команды: HTML-справка.

Управление цветом

COLORMAP **Палитра цветов**

Синтаксис:

```
colormap(C)
colormap('default')
C = colormap
```

Описание:

Палитра цветов C - это матрица размера m × 3 действительных чисел из диапазона [0.0 1.0]. Строка k палитры сформирована из трех чисел, которые указывают интенсивность красного, зеленого и синего цветов, то есть C(k, :) = [r(k) g(k) b(k)].

Команда colormap(C) устанавливает палитру согласно матрице C. Если значение элемента матрицы выходит за пределы интервала [0 1], выдается сообщение об ошибке

Colormap must have values in [0,1].

Значения элементов палитры должны быть в интервале [0 1].

Команды colormap('default') или colormap(hsv) устанавливают штатную палитру, которая соответствует модели hue-saturation-value (*оттенок-насыщенность-значение*). Последовательность цветов этой палитры соответствует цветам радуги:

red	Красный
-	-
yellow	Желтый
green	Зеленый
cyan	Голубой
blue	Синий
magenta	Фиолетовый

Функция C = colormap присваивает матрице C значение текущей палитры.

В каталоге graph3d размещены M-файлы, которые генерируют большое количество палитр; каждый из M-файлов допускает в качестве входного аргумента длину палитры. Например, команда создает палитру из 128 цветов; если размер не указан, то создается палитра текущей длины.

В системе MATLAB реализованы следующие палитры:

autumn	Палитра от красного к желтому через оранжевый
bone	Серая палитра с высокой интенсивностью синего
colorcube	Палитра радуги с большим количеством серого, красного, зеленого и синего
cool	Палитра с оттенками голубого и фиолетового
copper	Линейная палитра от черного до яркой меди
flag	Палитра с чередованием красного, белого, синего и черного
gray	Линейная палитра в оттенках серого
hot	Палитра с чередованием черного, красного, желтого и белого
hsv	Палитра радуги
jet	Разновидность hsv-палитры от синего до красного через голубой, желтый и оранжевый
lines	Палитра с оттенками серого, определяемая свойством Axes ColorOrder
pink	Розовая палитра с оттенками пастели
prism	Палитра с чередованием красного, оранжевого, желтого, зеленого, синего и фиолетового
spring	Палитра с оттенками фиолетового и желтого
summer	Палитра с оттенками зеленого и желтого
white	Белая палитра
winter	Палитра с оттенками синего и зеленого

Сопутствующие команды: BRIGHTEN, CAXIS, FIGURE, HSV, RGBPLOT, SPINMAP.

CAXIS

Установление соответствия между палитрой цветов и масштабированием осей

Синтаксис:

```
caxis([cmin cmax])
caxis auto
caxis manual
caxis(caxis)
v = caxis
```

Описание:

Команды caxis изменяют значения свойств Clim, CLimMode графического объекта Axes. Эти команды управляют преобразованием данных в палитру. Они изменяют свойства графических объектов Surface, Patch, Image, если массив CData нецелочисленный, а свойство CDataMapping имеет значение

scaled. Они не влияют на свойства этих объектов, если массив CData целочисленный, а свойство CDataMapping имеет значение direct.

Команда caxis([cmin cmax]) устанавливает диапазон из палитры цветов для отображения данных. Значения данных, которые меньше cmin, отображаются в cmin, а большие cmax - в cmax. Значения в интервала [cmin cmax] отображаются линейно в цвета текущей палитры.

Команда caxis auto устанавливает штатное, то есть используемое по умолчанию отображение данных в палитру цветов; в этом случае отсекаются цвета, которые соответствуют значениям Inf и NaN.

Команды caxis manual и caxis(caxis) фиксируют текущую палитру для последующих графиков, если они будут выводиться в режиме hold.

Функция v = caxis возвращает вектор-строку [cmin cmax].

Пояснение:

При выводе на экран графических объектов Surface, Patch, Image, если массив CData нецелочисленный, а свойство CDataMapping имеет значение scaled, выполняется преобразование массива в цвета палитры. Это линейное преобразование устанавливает соответствие между значением C из массива CData и номером строки index в массиве палитры длины m согласно следующей формуле:

index = fix((C-cmin)/(cmax-cmin)*m)+1.

Пример:

```
[X, Y, Z] = sphere(32); % Сфера радиусом 1
C = Z;                  % Значения для C - в диапазоне [-1 1].
surf(X, Y, Z, C)        % Изображение сферы
colormap pink           % Палитра pink
colorbar                % Шкала палитры
```

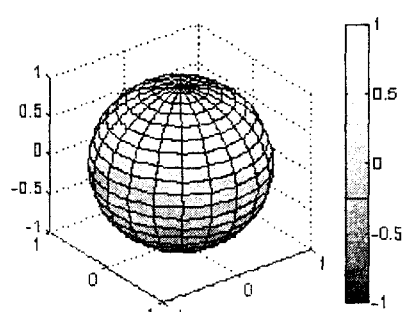


Рис. 10.24

Сопутствующие команды: AXES, AXIS, COLORMAP, GET, MESH, PCOLOR, SET, SURF.

Изображение линий уровня

CONTOUR **Изображение линий уровня для трехмерной поверхности**

Синтаксис:

```
contour(Z)                    contour(x, y, Z)
contour(Z, n)                 contour(x, y, Z, n)
contour(Z, v)                 contour(x, y, Z, v)
contour(..., LineSpec)
[C, h] = contour(...)
```

Описание:

Команда contour(Z) рисует двумерные линии уровня для массива данных Z, определяющих поверхность в трехмерном пространстве без учета диапазона изменения координат x и y.

Команда contour(x, y, Z), где x и y - векторы, рисует линии уровня для массива данных Z с учетом диапазона изменения координат x и y.

Команды contour(Z, n), contour(x, y, Z, n) рисуют n линий уровня для массива данных Z; по умолчанию n равно 10.

Команды contour(Z, v), contour(x, y, Z, v) рисуют линии уровня для заданных значений, которые указаны в векторе v.

Команда contour(..., LineSpec) рисует линии уровня, тип и цвет которых определяются параметром LineSpec команды plot.

Функция [C, h] = contour(...) возвращает массив C и вектор-столбец дескрипторов h графических объектов для каждой линии уровня. Если указан параметр LineSpec, то функция contour формирует графические объекты Line; если нет, то графические объекты Patch.

Пример:

Построить линии уровня функции $z = x e^{-x^2-y^2}$ в области $-2 \le x \le 2, -2 \le \le y \le 3$.

```
[X, Y] = meshgrid(-2 : .2 : 2, -2 : .2 : 3);
Z  = X.*exp(-X.^2 - Y.^2);
[C, h] = contour(X, Y, Z);
clabel(C, h)
```

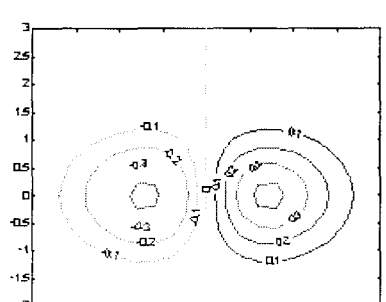
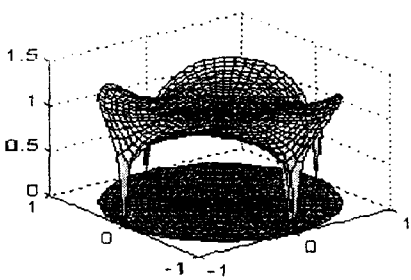


Рис. 10.25

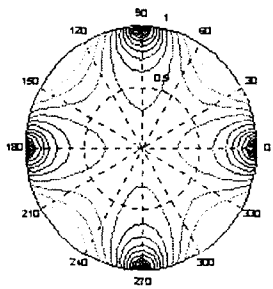
Построить изолинии комплексной функции $F(z) = \sqrt[4]{z^4 - 1}$ в полярных координатах.

```
% Рис. 10.26, а
[th, r] = meshgrid((0 : 5 : 360)*pi/180,  0 : 0.05 : 1);
[X, Y] = pol2cart(th, r);
Z = X + i*Y;
F = (Z .^ 4 - 1) .^ (1/4);
surf(X, Y, abs(F)), hold on
surf(X, Y, zeros(size(X)), hold off

% Рис. 10.26, б
 hp = polar([0 2*pi], [0 1]);
delete(hp)
hold on
contour(X, Y, abs(F),  30)
```



а



б

Рис. 10.26

Диагностические сообщения:

Если меньшая размерность массива Z оказывается меньше 2, то выдается сообщение

Matrix must be 2-by-2 or larger.

Матрица должна иметь размер 2×2 *или больше.*

Ограничения:

При работе с командой и функцией contour предполагается, что элементы массивов x и y монотонно возрастают.

Сопутствующие команды: CLABEL, CONTOUR3, CONTOURF, INTERP2, QUIVER.

CONTOURF — **Закрашенные графики линий уровня**

Синтаксис:

```
contourf(Z)
contourf(Z, n)
contourf(Z, v)
[C, h, CF] = contourf(...)
contourf(x, y, Z)
contourf(x, y, Z, n)
contourf(x, y, Z, v)
```

Описание:

Команды contourf(...) изображают линии уровня и закрашивают области между ними, используя некоторый постоянный цвет. Этот цвет определяется свойством Colormap графического объекта Figure.

Команда contourf(Z) рисует двумерные линии уровня для массива данных Z, определяющих поверхность в трехмерном пространстве, без учета диапазона изменения координат x и y.

Команда contourf(x, y, Z), где x и y - векторы, рисует линии уровня для массива данных Z с учетом диапазона изменения координат x и y.

Команды contourf(Z, n), contourf(x, y, Z, n) рисуют n линий уровня для массива данных Z; по умолчанию n равно 10.

Команды contourf(Z, v), contourf(x, y, Z, v) рисуют линии уровня для заданных значений, которые указаны в векторе v.

Функция [C, h, CF] = contourf(...) возвращает массив C, используемый функцией clabel, вектор-столбец дескрипторов h графических объектов Patch и матрицу CF, используемую для раскраски областей.

Пример:

Построить 10 линий уровня функции peaks(20).

```
contourf(peaks(20), 10);
colormap pink
brighten(0.7)
```

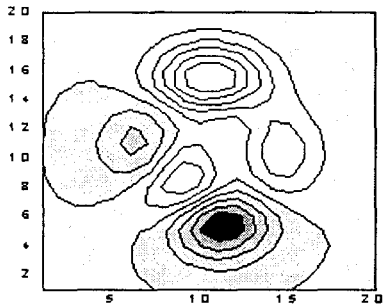


Рис. 10.27

Сопутствующие команды: CLABEL, CONTOUR, CONTOUR3, QUIVER.

CONTOUR3 **Изображение трехмерных линий уровня**

Синтаксис:

```
contour3(Z)                    contour3(X, Y, Z)
contour3(Z, n)                 contour3(X, Y, Z, n)
contour3(Z, v)                 contour3(x, y, Z, v)
contour3(..., LineSpec)
[C, h] = contour3(...)
```

Описание:

Команда contour3(Z) рисует трехмерные линии уровня для массива данных Z, определяющих поверхность в трехмерном пространстве, без учета диапазона изменения координат x и y.

Команда contour3(X, Y, Z), где X и Y - двумерные массивы, вычисленные с помощью функции meshgrid, рисует линии уровня для массива данных Z с учетом диапазона изменения координат x и y.

Команды contour3(Z, n), contour(X, Y, Z, n) рисуют n линий уровня для массива данных Z; по умолчанию n равно 10.

Функция [C, h] = contour3(...) возвращает массив C и вектор-столбец дескрипторов h графических объектов для каждой линии уровня. Если указан параметр LineSpec, то функция contour формирует графические объекты Line; если нет, то графические объекты Patch.

Пример:

Построить трехмерные линии уровня функции $z = xe^{-x^2-y^2}$ в области $-2 \le x \le 2, -2 \le \le y \le 2$.

```
[X, Y] = meshgrid([-2 : .25 : 2]);
Z = X.*exp(-X.^2 -  Y.^2);
contour3(X, Y, Z, 30)
surface(X, Y, Z, 'EdgeColor', [.8 .8 .8], 'FaceColor', 'none')
grid off
view(-15, 25)
colormap gray
brighten(-0.5)
```

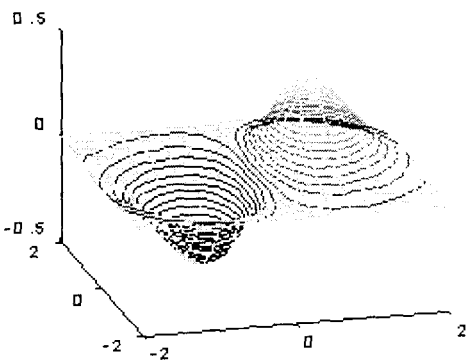


Рис. 10.28

Ограничения:

При работе с командой и функцией contour3 предполагается, что элементы массивов x и y монотонно возрастают.

Сопутствующие команды: CONTOUR, MESHC, MESHGRID, SURFC.

Надписи и пояснения к графикам

TITLE

Заголовки для двух- и трехмерных графиков

Синтаксис:

title('<*текст*>')
title(<*имя_функции*>)
title(..., '*PropertyName*', PropertyValue, ...)
h = title(...)

Описание:

Каждый графический объект Axes может сопровождаться своим заголовком, который размещается над графиком.

Команда title('<*текст*>') размещает заголовок в виде указанного текста над графиком.

Команда title(<*имя_функции*>) позволяет применить функцию, которая возвращает результат в виде строки, используемой в качестве заголовка.

Команда title(..., '*PropertyName*', PropertyValue, ...) позволяет указать пару свойство/значение для графического объекта Text, который создается командой title.

Функция h = title(...) возвращает дескриптор графического объекта Text.

Пример:

Построим разные графики в двух окнах и снабдим их разными заголовками. В первом окне построим зависимость температуры по шкале Цельсия от температуры по шкале Фаренгейта. Во втором окне построим график функции $y = Ae^{-\alpha t}\sin(\beta t)$ и снабдим его заголовком, связанным с моментом построения.

```
subplot(211)
f = -40 :100;
c = (f - 32)/1.8;
plot(f, c, 'LineWidth', 3), grid
title(' Температура C(F) ', 'FontSize', 14)

subplot(212)
t = 0 : 1000;
y = 0.25*exp(-5e-3*t) .* sin(0.2*t);
plot(t, y, 'LineWidth', 1.5), grid
title(datestr(now), 'FontSize', 9)
```

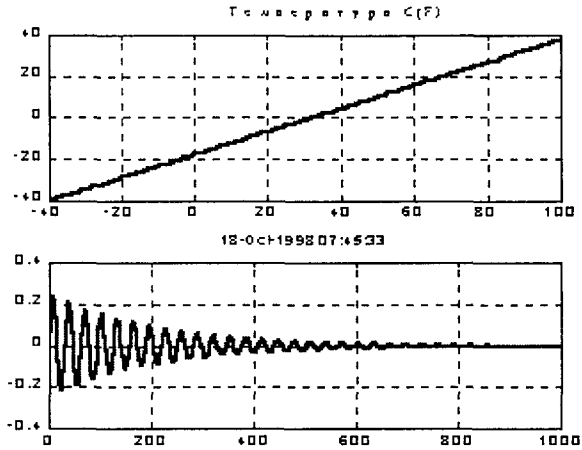


Рис. 10.29

Сопутствующие команды: GTEXT, INT2STR, NUM2STR, PLOT, TEXT, XLABEL, YLABEL, ZLABEL.

XLABEL YLABEL ZLABEL **Обозначение осей**

Синтаксис:

xlabel('<*текст*>')
xlabel(<*имя_функции*>)
xlabel(..., '*PropertyName*', PropertyValue, ...)
h = xlabel(...)
ylabel(...)
h = ylabel(...)
zlabel(...)
h = zlabel(...)

Описание:

Графический объект Axes может иметь только одну метку для каждой из координатных осей x, y, z.

Команда *label('<*текст*>') помещает метку вдоль соответствующей координатной оси.

Команда *label(<*имя_функции*>) позволяет применить функцию, которая возвращает результат в виде строки, и использовать ее в качестве метки оси.

Команда *label(..., '*PropertyName*', PropertyValue, ...) позволяет указать пару свойство/значение для графического объекта Text, который создается одной из команд *label.

Функции h = *label(...) возвращают дескриптор графического объекта Text.

Повторное использование команды приводит к замене старого текста новым.

Пример:

Построим зависимость температуры по шкале Цельсия от температуры по шкале Фаренгейта и снабдим график заголовком и метками координатных осей.

```
f = -40:100;
c = (f - 32)/1.8;
plot(f, c, 'LineWidth', 3), grid
title(' C(F) ', 'FontSize', 14)
xlabel(' Температура, F° ', 'FontAngle', 'italic', 'FontWeight', 'bold')
ylabel(' Температура, C° ', 'FontAngle', 'italic', 'FontWeight', 'bold')
```

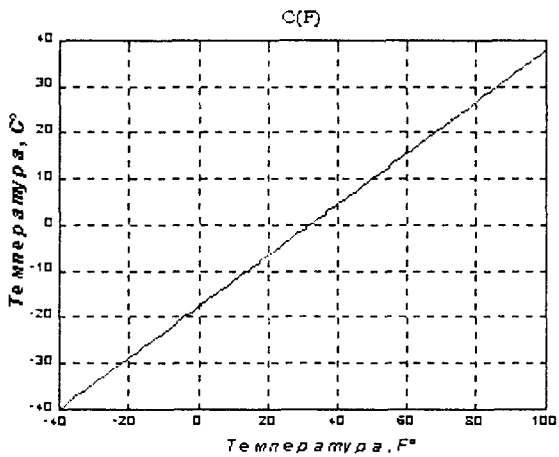


Рис. 10.30

Сопутствующие команды: TEXT, TITLE.

CLABEL **Маркировка линий уровня**

Синтаксис:

```
clabel(C, h)
clabel(C, h, v)
clabel(C, h, 'manual')
clabel(C)
clabel(C, v)
clabel(C, 'manual')
```

Описание:

Команды группы clabel маркируют линии уровня, сформированные функциями contour, contour3, contourf.

Команда clabel(C, h) маркирует линии уровня, размещая метки в разрывах линий уровня и поворачивая их для удобства чтения.

Команда clabel(C, h, v) маркирует только линии уровня, заданные вектором v.

Команда clabel(C, h, 'manual') маркирует линии уровней в позициях, указываемых перемещением курсора. Нажатие левой клавиши мыши или клавиши пробела маркирует указанную линию уровня; нажатие клавиши Return или правой кнопки мыши завершает процесс маркировки.

Команда clabel(C) добавляет метки к линиям уровня в случайно выбранных позициях.

Команда clabel(C, v) маркирует линии уровня контура, которые заданы в векторе v.

Команда clabel(C, 'manual') маркирует линии уровней в позициях, указываемых с помощью мыши.

Пояснение:

В тех случаях, когда команда использует дескриптор h, маркер линии уровня вставляется в разрыв линии и поворачивается нужным образом. Если команда не использует дескриптор, то в этом случае производится маркировка знаком + и числовым значением.

Пример:

Сгенерировать, нарисовать и маркировать линии уровня для функции $z = xe^{(-x^2-y^2)}$.

```
[X, Y] = meshgrid([-2 : .2 : 2]);
Z = X.* exp(- X.^2 - Y.^2);
[C, h] = contour(X, Y, Z);
clabel(C, h)
```

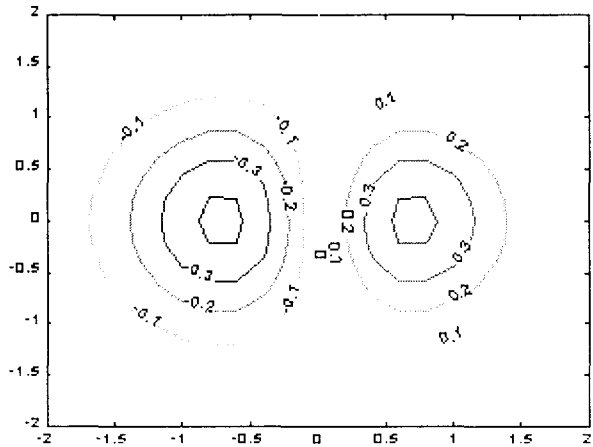


Рис. 10.31

Сопутствующие команды: CONTOUR, CONTOURF.

TEXT **Вывод текста на график**

Синтаксис:

text(x, y, '<*текст*>')
text(x, y, z, '<*текст*>')
text(...'*PropertyName*', PropertyValue...)
h = text(...)

Описание:

Команда text всегда создает графический объект Text.

Команда text(x, y, '<*текст*>') помещает в заданной точке (x, y) двумерного графика начало текста, указанного в качестве третьего аргумента. Если x и y одномерные массивы, заданный текст помещается во все позиции, определяемые координатами [x(i) y(i)].

Команда text(x, y, z, '<*текст*>') выводит текст на трехмерный график.

Команда text(...'*PropertyName*', PropertyValue ...) позволяет задать пару свойство/значение для графического объекта Text.

Функция h = text(..) возвращает вектор дескрипторов для графических объектов Text, являющихся потомками объекта Axes.

Пояснение:

Координаты размещения текста (аргументы x, y, z) определяются единицами измерения, заданными свойством Units объекта Text (по умолчанию это значение 'data'; за позиционирование символов текста отвечают свойства Extent, VerticalAlignment, HorizontalAlignment.

Следует обратить внимание на свойство Interpreter {tex} | none, которое указывает, что по умолчанию используется интерпретатор графических символов языка TeX. Это позволяет во всех командах (title, *label, legend, text), порождающих графический объект Text, использовать синтаксис языка TeX для написания текста с символами греческого алфавита и другими знаками.

Например, формула Эйлера $e^{j\varphi} = cos(\varphi) + jsin(\varphi)$ может быть представлена в виде текстовой строки на языке TeX: '\ite^{j\phi} = cos(\phi) + j sin(\phi)'). Для вывода эта строка либо непосредственно вводится в самой команде, либо передается через свойство String объекта Text.

Свойство String объекта Text

Значение этого свойства может быть задано либо в виде единственной строки, заключенной в кавычки, либо в виде нескольких строк, представленных массивами строк или ячеек. Сформированные таким образом строки выводятся на графике в указываемых позициях. Вертикальный слэш выводится как часть текстовой строки и не интерпретируется как символ конца строки. Когда свойство Interpreter имеет значение Tex (по умолчанию), можно использовать подмножество команд языка TeX, чтобы вывести на график греческие буквы или математические символы. Следующая таблица содержит команды языка для вывода соответствующих символов.

Команда	*Символ*	*Команда*	*Символ*	*Команда*	*Символ*
\alpha	α	\upsilon	υ	\sim	~
\beta	β	\phi	φ	\leq	≤
\gamma	γ	\chi	χ	\infty	∞
\delta	δ	\psi	ψ	\clubsuit	♣
\epsilon	ε	\omega	ω	\diamondsuit	♦
\zeta	ζ	\Gamma	Γ	\heartsuit	♥
\eta	η	\Delta	Δ	\spadesuit	♠
\theta	θ	\Theta	Θ	\leftrightarrow	↔
\vartheta	υ	\Lambda	Λ	\leftarrow	←
\iota	ι	\Xi	Ξ	\uparrow	↑
\kappa	κ	\Pi	Π	\rightarrow	→
\lambda	λ	\Sigma	Σ	\downarrow	↓
\mu	μ	\Upsilon	Υ	\circ	°
\nu	ν	\Phi	Φ	\pm	±
\xi	ξ	\Psi	Ψ	\geq	≥
\pi	π	\Omega	Ω	\propto	∝
\rho	ρ	\forall	∀	\partial	∂
\sigma	σ	\exists	∃	\bullet	•
\varsigma	ζ	\ni	∋	\div	÷
\tau	τ	\cong	≅	\neq	≠
\equiv	≡	\approx	≈	\aleph	ℵ
\Im	ℑ	\Re	ℜ	\wp	℘
\otimes	⊗	\oplus	⊕	\oslash	∅
\cap	∩	\cup	∪	\supseteq	⊇
\supset	⊃	\subseteq	⊆	\subset	⊂
\int	∫	\in	∈	\o	o
\rfloor	⌋	\lceil	⌈	\nabla	∇
\lfloor	⌊	\cdot	·	\ldots	...
\perp	⊥	\neg	¬	\prime	'
\wedge	∧	\times	×	\0	∅
\rceil	⌉	\surd	√	\mid	|
\vee	∨	\varpi	ϖ	\copyright	©
\langle	⟨	\rangle	⟩		

Можно задать модификаторы потока, которые управляют шрифтом:

\bf - жирный шрифт (bold font);
\it - наклонный шрифт (italics font);
\sl - косой шрифт (oblique font), применяется редко;
\rm - нормальный шрифт (normal font);
\fontname{fontname} - определить новое семейство шрифтов;
\fontsize{fontsize} - определить новый размер шрифта.

Первые 4 модификатора являются взаимно исключающими. Однако с помощью модификатора \fontname можно создавать комбинации модификаторов.

Область действия модификатора - вся строка, за исключением фрагментов, заключенных в фигурные скобки { }.

Символы нижнего _ и верхнего ^ индексов воздействуют на символ или подстроку, определенную в фигурных скобках, непосредственно следующие за символом.

Для вывода на печать специальных символов интерпретатора TeX необходимо, чтобы им предшествовал обратный слэш \, например: \\, \{, \}, _, \^.

Если свойство Interpreter установлено в состояние 'none', то символы строки не интерпретируются, а непосредственно выводятся на экран.

Замечания:

Символы кириллицы интерпретатором TeX не воспринимаются.

По умолчанию свойство FontName имеет значение Helvetica, то есть при написании текста используется шрифт Helvetica, который поддерживает символы кириллицы. При задании другого имени кириллического шрифта возникают проблемы при копировании рисунка в формате Windows Metafile в буфер обмена.

Примеры:

Записать в заголовке графика выражение для функции $Ae^{-\alpha t}\sin(\beta t)$, используя синтаксис языка TeX.

```
t = 0 : 1000;
y = 0.25*exp(-5e-3*t) .* sin(0.2*t);
plot(t, y, 'LineWidth', 1.5), grid
set(get(ht, 'Title'), 'String', ' \itAe^{\rm\alpha\itt} \rmsin(\rm\beta\itt\rm) ', 'FontSize', 12)
set(get(hl, 'Xlabel'), 'String', ' \itt, \rm\mu\rmsec ', 'FontSize', 12)
```

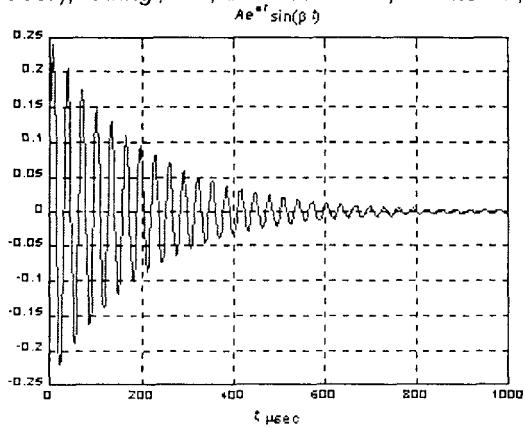


Рис. 10.32

Сопутствующие команды: GTEXT, INT2STR, NUM2STR, TITLE, XLABEL, YLABEL, ZLABEL .

GTEXT

Разместить заданный текст на графике с использованием мыши

Синтаксис:

gtext('<*текст*>')
h = gtext('<*текст*>')

Описание:

Команда gtext('<*текст*>') высвечивает в активном графическом окне перекрестие, перемещение которого позволяет указать место ввода заданного текста; по завершении позиционирования нажатие кнопки мыши или любой клавиши вводит заданный текст.

Функция h = gtext('<*текст*>') возвращает дескриптор графического объекта Text.

Примеры:

Разместить на графике рис. 10.32 текст *"Затухающая синусоида"*.

```
t = 0 : 1000;
y = 0.25*exp(-5e-3*t) .* sin(0.2*t);
plot(t, y, 'LineWidth', 1.5), grid
set(get(ht, 'Title'), 'String', ' \itAe^{\rm\alpha\itt} \rmsin(\rm\beta\itt\rm) ', 'FontSize', 12)
set(get(hl, 'Xlabel'), 'String', ' \itt, \rm\mu\rmsec ',  'FontSize', 12)
hgt = gtext('Затухающая синусоида');
hg =gtext(' ')
set(hg, 'String', {' A = 0.25 '; ' \alpha = 0.005 '; ' \beta = 0.2 '})
```

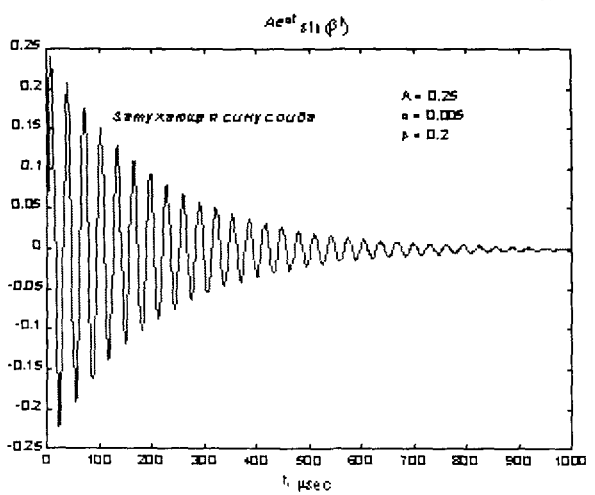


Рис. 10.33

Сопутствующие команды: TEXT, GINPUT.

LEGEND

Пояснение к графику

Синтаксис:

legend('<*текст*1>', '<*текст*2>', '<*текст*3>', ...)
legend(M)
legend(h, M)
legend('off'), legend off
legend(h,...)
legend(..., pos)
h = legend(...)

Описание:

Команды группы legend позволяют сопроводить график пояснением, или легендой, в которой отображается тип каждой линии или маркера с пояснительным текстом.

Команда legend('<*текст*1>', '<*текст*2>', '<*текст*3>', ...) добавляет к текущему графику пояснение в виде указанных текстовых строк.

Команда legend(M) формирует пояснения из строк массива M. Следует помнить, что строки массива M должны иметь одинаковую длину.

Команда legend(h, M) ассоциирует каждую строку массива M с соответствующим графическим объектом, определяемым вектором дескрипторов h.

Команды legend('off') и legend off удаляют пояснение с текущего графика.

Команда legend(h, ...) выводит пояснение к графическому объекту Axes с дескриптором h.

Команда legend(..., pos) использует следующие значения параметра pos, чтобы задать местоположение легенды:

Значение pos	*Местоположение легенды*
-1	Вне графика, справа
0	В одном из четырех углов области графика, так, чтобы закрывать наименьшее количество точек графика
1	В правом верхнем углу графика (используется по умолчанию)
2	В левом верхнем углу графика
3	В левом нижнем углу графика
4	В правом нижнем углу графика
[x y]	Координаты левого нижнего угла области легенды

Функция h = legend(...) возвращает дескриптор h графического объекта Axes.

Для перемещения области легенды следует нажать левую кнопку мыши, находясь в этой области, а затем переместить пояснение в нужную позицию.

Примеры:

Построить на одном графике функции a = sin(t), b = cos(t), а также их сумму в виде дискретного графика в 60 точках и сопроводить его пояснением.

```
t = linspace(0, 2*pi, 60);
a = sin(t); b = cos(t);
x = 1 : 60;
stem(x, a + b), hold on
hp  = plot(x, a', 'g', x, b', 'r');  set(hp, 'LineWidth', 2)
legend( ' a + b ', ' a = sin(t) ', ' b = cos(t) ' , 3), hold off
hx = xlabel(' t, мс ',  'FontSize', 12, 'FontAngle', 'italic', 'FontWeight', 'bold' );
title(' y = sin(t) + cos(t) ',  'FontSize', 12, 'FontWeight', 'bold' )
```

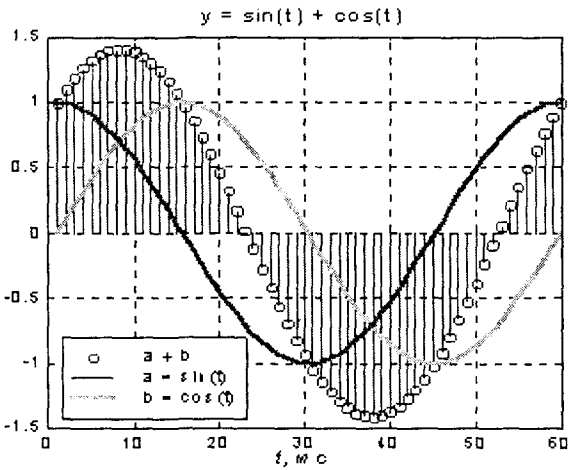


Рис. 10.34

Сопутствующие команды: PLOT.

COLORBAR **Шкала палитры**

Синтаксис:

```
colorbar('vert')
colorbar('horiz')
colorbar(h)
colorbar
h = colorbar(...)
```

Описание:

Команда colorbar('vert') добавляет к текущему графику вертикальную шкалу палитры.

Команда colorbar('horiz') добавляет к текущему графику горизонтальную шкалу палитры.

Команда colorbar(h) добавляет к графику с дескриптором h шкалу палитры. Размещение шкалы реализуется автоматически в зависимости от соотношения ширины и высоты графика.

Команда colorbar без аргументов либо использует уже существующую вертикальную шкалу палитры, либо размещает на текущем графике новую вертикальную шкалу палитры.

Функция h = colorbar(...) возвращает дескриптор h графического объекта Axes.

Сопутствующие команды: COLORMAP.

PLOTEDIT **Диалоговый редактор графики**

Синтаксис:

```
plotedit on
plotedit off
plotedit
plotedit(fig)
plotedit(fig, on | off)
```

Описание:

Команда plotedit on добавляет меню Tools и инструментальную панель к текущему графическому объекту Figure (рис. 10.35).

Команда plotedit off удаляет инструментальную панель.

Команда plotedit переключает состояние текущего окна.

Команда plotedit(fig) переключает состояние графического окна с дескриптором fig.

Команда plotedit(fig, on | off) выполняет команды plotedit on и plotedit off для графического окна с дескриптором fig.

Когда окно диалогового редактора активно, с помощью кнопок диалоговой панели можно дополнить рисунок текстом, стрелками или пояснительными линиями. Такие аннотации в дальнейшем можно перемещать с помощью мыши.

Для редактирования графических объектов, из которых составлен рисунок, необходимо использовать правую кнопку мыши и руководствоваться возможностями ниспадающего меню.

Пример:

```
t = linspace(0, 2*pi, 60);
a = sin(t); b = cos(t);
x = 1 : 60;
stem(x, a + b), hold on
hp  = plot(x, a', 'g', x, b', 'r');  set(hp, 'LineWidth', 2)
legend( ' a + b ', ' a = sin(t) ', ' b = cos(t) ' , 3), hold off
hx = xlabel(' t, мс ',  'FontSize', 12, 'FontAngle', 'italic', 'FontWeight', 'bold' );
title(' y = sin(t) + cos(t) ',  'FontSize', 12, 'FontWeight', 'bold' )
plotedit  % Рис. 10.35, а
```

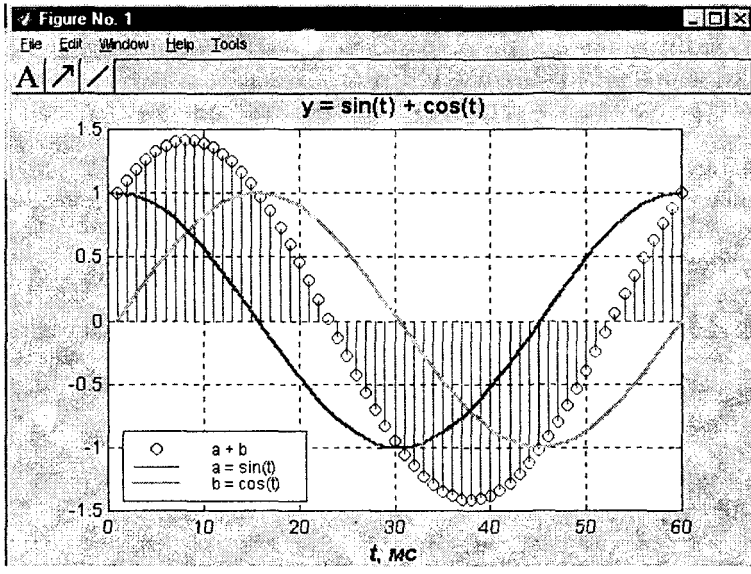


Figure No. 1
File Edit Window Help Tools
y = sin(t) + cos(t)
1.5
1
0.5
0
-0.5
-1
-1.5
0 10 20 30 40 50 60
t, мс
a + b
a = sin(t)
b = cos(t)

а

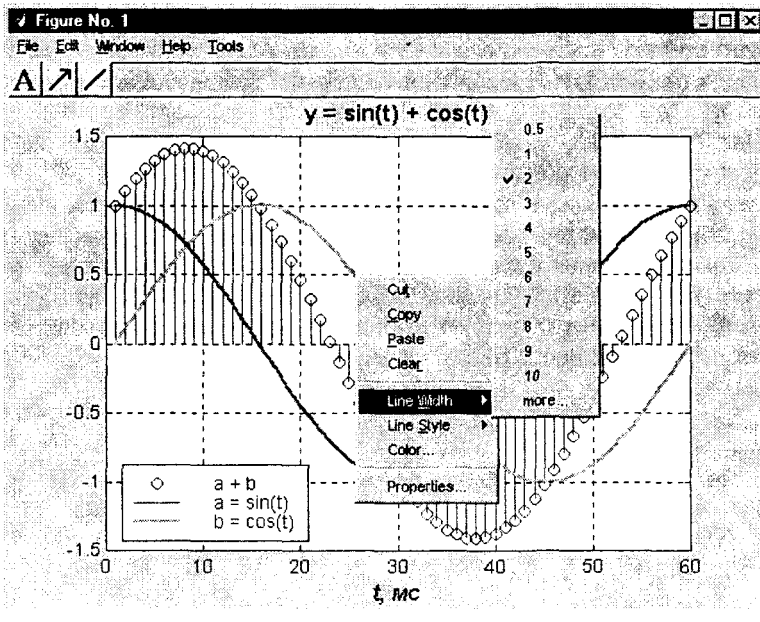
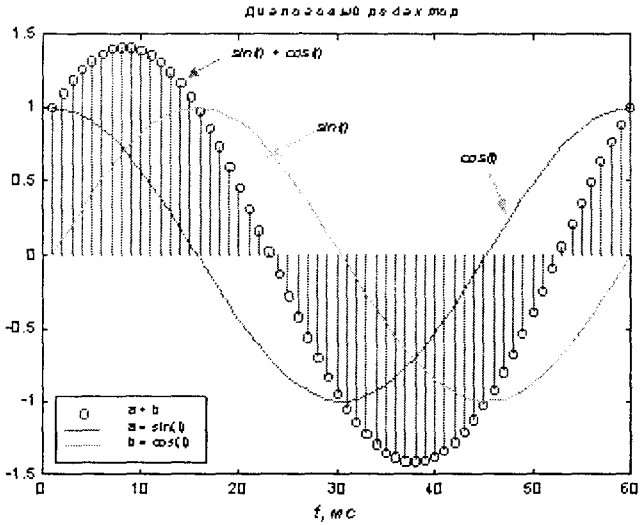


Figure No. 1
File Edit Window Help Tools
y = sin(t) + cos(t)
1.5
1
0.5
0
-0.5
-1
-1.5
0 10 20 30 40 50 60
t, мс
a + b
a = sin(t)
b = cos(t)
Cut
Copy
Paste
Clear
Line Width
Line Style
Color...
Properties...
0.5
1
2
3
4
5
6
7
8
9
10
more...

б



в

Рис. 10.35

Рис. 10.35, *б* показывает ниспадающее меню, которое ассоциировано с графическим объектом Line и позволяет корректировать некоторые свойства этого объекта, в данном случае толщину линии (LineWidth).

На рис. 10.35, *в* выведен результат диалогового редактирования эскиза с рис. 10.35, *а*. Читатель сам может найти различия в оформлении графиков.

Предложенный редактор является экспериментальным. Его действительные возможности значительно шире, чем продемонстрированные выше. В частности, он включает инструментальное средство Graphics Property Editor, которое позволяет определить и откорректировать любое из свойств графических объектов, составляющих редактируемый рисунок.

Предлагаем читателям самим включиться в освоение этого графического редактора.

Сопутствующие команды: HTML-справка.

Специальная графика

Раздел специальной графики включает графические команды и функции для построения столбцовых диаграмм, гистограмм, средств отображения векторов и комплексных элементов, вывода дискретных последовательностей данных, а также движущихся траекторий как для двумерной, так и для трехмерной графики.

Двумерная графика

BAR, BARH — **Столбцовые диаграммы**

Синтаксис:

```
bar(Y)                          barh(...)
bar(x, Y)                       [xb, yb] = barh(...)
bar(..., <ширина>)              h = barh(...)
bar(..., '<стиль>')
bar(..., LineSpec)
[xb, yb] = bar(...)
h = bar(...)
```

Описание:

Команды bar(...), barh(...) отображают значения вектора или двумерного массива в виде вертикальных или горизонтальных столбцовых диаграмм.

Команда bar(Y) в случае одномерного массива выводит каждый элемент вектора y в виде столбца диаграммы; если Y - двумерный массив, то формируются кластеры (группы) столбцов, соответствующие значениям элементов строки. В этом случае ось x размечается метками по количеству строк массива от 1 до size(Y, 1).

Команда bar(x, Y) в случае одномерного массива выводит элементы вектора y в виде столбцов диаграммы в позициях, определяемых вектором x, элементы которого должны быть расположены в порядке возрастания; если Y - двумерный массив, то в этом случае в позициях, определяемых вектором x, размещаются кластеры столбцов.

Команда bar(..., <*ширина*>) устанавливает относительную ширину столбца диаграммы. По умолчанию это значение равно 0.8 в тех случаях, когда отсутствует вектор x; это означает, что между столбцами есть небольшой просвет; если параметр <*ширина*> равен 1, то столбцы, образующие группу, касаются, а при больших значениях перекрывают друг друга.

Команда bar(..., '<*стиль*>') позволяет задать способ группирования столбцов диаграммы в кластер; параметр <*стиль*> может принимать одно из следующих значений:

<*стиль*>	*Способ группирования*
'group'	Формируется n кластеров по m вертикальных столбцов в каждом, где n - число строк, а m - число столбцов массива Y
'stack'	Формируется n кластеров по одному столбцу, именуемых стеками, при этом высота стека соответствует сумме элементов строки, а вклад каждого элемента определяется частями этого стека, окрашенными в разные цвета

Команда bar(..., LineSpec) позволяет с помощью параметра LineSpec задать общий цвет для всех столбцов диаграммы.

Функция [xb, yb] = bar(...) не выводит графика, а формирует такие массивы xb и yb, которые позволяют построить столбцовую диаграмму с помощью команд plot(xb, yb) и patch(xb, yb, C). Это открывает большие возможности для формирования сложных графиков, позволяя, например, включать в них столбцовые диаграммы.

Функция h = bar(...) возвращает вектор h дескрипторов графического объекта Patch. При этом каждому столбцу массива Y соответствует один графический объект Patch.

Команды и функции barh(...), [xb, yb] = barh(...), h = barh(...) формируют столбцовые диаграммы, которые располагаются горизонтально.

Примеры:

Сформируем случайный массив размера 5×3 и отобразим его в виде четырех разновидностей столбцовых диаграмм.

```
Y = round(rand(5, 3)*10);
subplot(2, 2, 1), bar(Y, 'group'),  title 'Группа'
subplot(2, 2, 2), bar(Y, 'stack'),   title 'Стек'
subplot(2, 2, 3), barh(Y, 'stack'), title 'Стек'
subplot(2,2,4), bar(Y,1.5), title 'Ширина: 1.5'
```

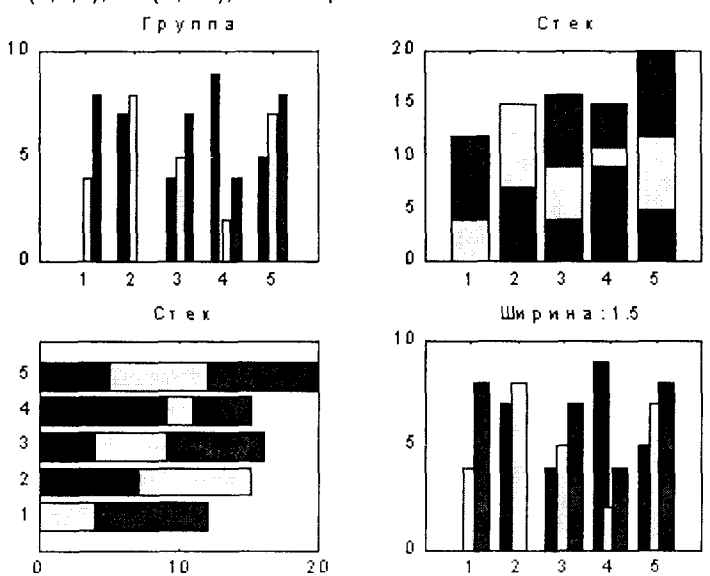


Рис. 10.36

Сопутствующие функции и команды: BAR3, HIST, PATCH, STAIRS.

PIE **Секторная диаграмма**

Синтаксис:

```
pie(x)
pie(x, v)
h = pie(...)
```

Описание:

Команда pie(x) отображает каждый элемент вектора x в виде сектора диаграммы.

Команда pie(x, v), где v - логический вектор, состоящий из 0 и 1, отделяет от диаграммы те секторы, которым соответствуют ненулевые элементы вектора v.

Функция h = pie(...) возвращает вектор h дескрипторов графических объектов Patch, связанных с секторами, и графических объектов Text, связанных с аннотацией секторов.

Замечание:

При построении секторной диаграммы используется нормированный вектор x = x/sum(x), который определяет размер каждого сектора. Если sum(x) ≤ 1, то непосредственные значения элементов вектора определяют размеры секторов; когда sum(x) < 1, секторная диаграмма оказывается незамкнутой.

Пример:

Построить секторную диаграмму и отделить от нее сектор 2.

```
x = [1 3 0.5 2.5 2];
v = [0 1 0 0 0];
pie(x, v)
colormap pink
```

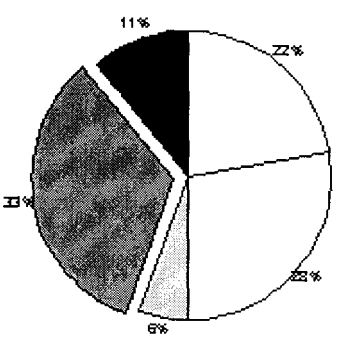


Рис. 10.37

Сопутствующие функции: PIE3.

ERRORBAR **График с указанием интервала погрешности**

Синтаксис:

```
errorbar(Y, E)
errorbar(X, Y, E)
errorbar(X, Y, L, U)
errorbar(..., LineSpec)
h = errorbar(...)
```

Описание:

Команды errorbar(y, e) и errobar(x, y, e) строят графики функций с погрешностью ±e относительно каждой точки графика.

Команда errorbar(X, Y, L, U) строит график функции y в зависимости от x с указанием интервала погрешности, который определяется массивами l и u. Массивы x, y, l, u должны быть одного размера. Погрешности в каждой точке {x(i), y(i)} определяются отклонениями вниз l(i) и вверх u(i) относительно точки графика, так что суммарная погрешность равна l(i) + u(i). Если X, Y, L и U - двумерные массивы, то в этом случае каждому столбцу соответствует свой график.

Команда errorbar(..., LineSpec) позволяет управлять типами линии и маркера, а также цветом по аналогии с тем, как это делается в команде plot.

Функция h = errorbar(...) возвращает вектор h дескрипторов графических объектов Line.

Пример:

Построить график функции y = sin(x) с интервалом погрешности ±e, который определяется как стандартное отклонение.

```
x = 1 : 10;
y = sin(x);
e = std(y)*ones(size(x))/5;
errorbar(x, y, e)
```

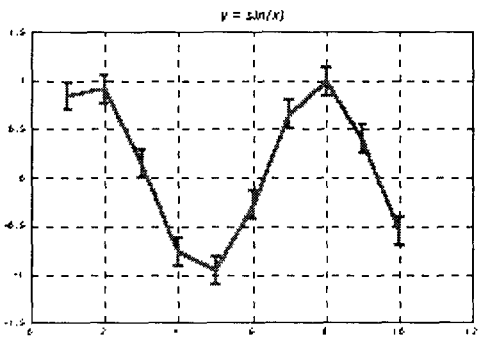


Рис. 10.38

Сопутствующие функции: PLOT, STD.

HIST **Построение гистограммы**

Синтаксис:

```
hist(y)
hist(y, x)
hist(y, n)
[p, x] = hist(y, ...)
```

Описание:

Команды hist(...) подсчитывают количество элементов массива y, значения которых попадают в заданные интервалы и отображают их на графике; для этого весь диапазон значений y делится на n интервалов (по умолчанию 10) и подсчитывается количество элементов в каждом из них.

Команда hist(y) выводит гистограмму для 10 интервалов.

Команда hist(y, n) выводит гистограмму для n интервалов.

Команда hist(y, x) выводит гистограмму с учетом диапазона изменения переменной x.

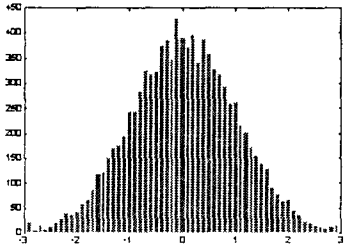
Функции [p, x] = hist(y, ...) формируют такие массивы p и x, что bar(x, p) является гистограммой.

Если массив Y двумерный, то гистограмма выводится как столбцовая диаграмма с группами столбцов.

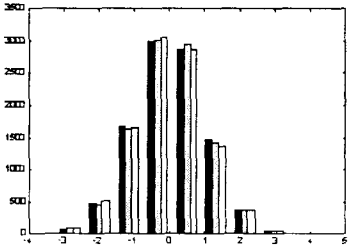
Примеры:

Построить гистограмму для 10 000 случайных чисел, распределенных по нормальному закону (рис. 10.39, *а*), и групповые гистограммы для массива Y размера 10000×3 (рис. 10.39, *б*).

```
x = -2.9 : 0.1 : 2.9;
% Рис. 10.39а
y = randn(10000, 1);
hist(y, x)
h = get(gca, 'Children');
set(h, 'FaceColor', 'r', 'EdgeColor', 'w')
% Рис. 10.39б
Y = randn(10000, 3);
hist(Y, x)
```



а



б

Рис. 10.39

Сопутствующие функции: BAR, PATCH, STAIRS.

STEM **Дискретные графики**

Синтаксис:

```
stem(Y)
stem(X, Y)
stem(..., 'fill')
stem(..., LineSpec)
h = stem(...)
```

Описание:

Команда stem(y) выводит график элементов одномерного массива y в виде вертикальных линий, которые заканчиваются маркером (по умолчанию - круг).

Команда stem(x, y) выводит график элементов массива y в виде вертикальных линий в позициях, определяемых массивом x, элементы которого должны быть расположены в порядке возрастания. Если массив Y двумерный, то дискретный график выводится в виде вертикальных линий и маркеров разного цвета.

Команды stem(..., 'fill') производят закраску маркера.

Команды stem(..., LineSpec) позволяют управлять типами линий и маркеров, используемых для построения дискретного графика, по аналогии с командой plot.

Функция h = stem(...) возвращает вектор дескрипторов h графических объектов Line для вертикальных линий и маркеров.

Примеры:

Построить дискретный график для выборки из 10 чисел, распределенных по нормальному закону.

```
y = randn(10, 1);
h = stem(y)
set(h(1), 'MarkerSize', 8, ' MarkerEdgeColor ', ' r ', ' LineWidth ', 2)
title(' randn(10, 1) ')
```

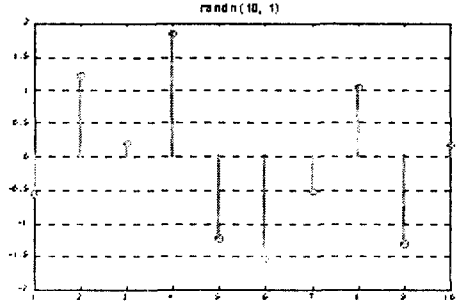


Рис. 10.40

Сопутствующие функции: PLOT, BAR, STAIRS, STEM3.

STAIRS **Ступенчатый график**

Синтаксис:

```
stairs(Y)
stairs(X, Y)
stairs(..., LineSpec)
[xb, yb] = stairs(...)
```

Описание:

Команда stairs(y) выводит график элементов одномерного массива y в виде ступенчатой функции.

Команда stairs(x, y) выводит график элементов массива y в виде ступенчатой функции в позициях, определяемых массивом x, элементы которого должны быть расположены в порядке возрастания. Если массив Y двумерный, то число выводимых ступенчатых функций равно числу строк этого массива.

Команды stairs(..., LineSpec) позволяют управлять типами линий, используемых для построения ступенчатого графика, по аналогии с командой plot.

Функция [xb, yb] = stairs(...) не выводит графика, а формирует такие массивы xb и yb, которые позволяют построить ступенчатую функцию с помощью команды plot(xb, yb).

Примеры:

Построить ступенчатый график функции x = sin(t).

```
kt= 0 : 0.25 : 10;
stairs(kt, sin(kt))
title(' sin(kT) '), xlabel(' kT ')
```

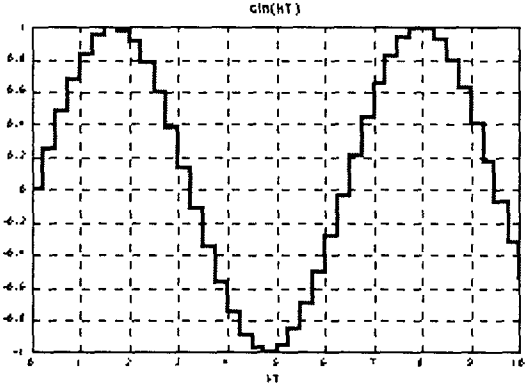


Рис. 10.41

Сопутствующие функции: BAR, HIST.

SCATTER **Поле точек**

Синтаксис:

```
scatter(x, y, s, c)
scatter(x, y)
scatter(x, y, s)
scatter(..., <тип_маркера>)
scatter(..., 'filled')
h = scatter(...)
```

Описание:

Команда scatter(x, y, s, C) строит поле точек путем вывода цветных маркеров в виде кругов в точках плоскости, определяемых векторами x и y, которые должны иметь одинаковую длину.

Вектор s той же длины позволяет задать размер каждого маркера как значение points^2, где points - это значение свойства Units (единица измерения) графического объекта Axes. Параметр может быть и скаляром, и тогда размеры всех маркеров одинаковы.

Массив C позволяет задать цвет каждого маркера. Если это вектор той же длины, как x и y, то его значения берутся в виде линейной функции из текущей палитры; если массив C имеет размер length(X)×3, то цвета маркера определяются rgb-значениями.

Команда scatter(x, y) выводит цветные маркеры с параметрами Marker = o (*круг*), EdgeColor = [0 0 1] (*синего цвета*) и MarkerSize = 6 (*размер*), принятыми по умолчанию.

Команда scatter(x, y, s) выводит маркеры в виде кругов синего цвета с размерами MarkerSize = sqrt(s).

Команда scatter(..., <*тип_маркера*>) позволяет изменить тип используемого маркера.

Команды scatter(..., 'filled) закрашивают маркеры синим цветом (по умолчанию). Цвет маркера управляется параметром MarkerFaceColor.

Функция h = scatter(...) возвращает вектор h дескрипторов графических объектов Line; длина этого вектора совпадает с длиной вектора x.

Примеры:

Построим поле случайных точек, полученных на основе выборки 100 нормально распределенных случайных величин по каждой координатной оси.

```
x = randn(100, 1);  y = randn(100, 1);
scatter(x, y, 36, 'r', 'filled')
```

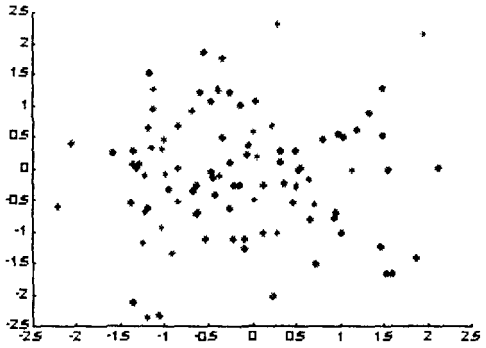


Рис. 10.42

Сопутствующие функции: PLOT, PLOTMATRIX, SCATTER3.

PLOTMATRIX **Матрица полей точек**

Синтаксис:

```
plotmatrix(X, Y)
plotmatrix(..., 'LineSpec')
[H, AX, GlobAx, P] = plotmatrix(...)
```

Описание:

Команда plotmatrix(X, Y) строит матрицу полей точек, где каждое поле точек отображает зависимость элементов столбцов массива X от элементов столбцов массива Y. Если X - массив размера p×m, а Y - массив размера p×n, то формируется матрица графических объектов Axes (координатных осей) размера n×m. Команда plotmatrix(Y) равносильна команде plotmatrix(Y, Y), но в этом случае диагональные поля точек заменяются гистограммами, которые строятся командами hist(Y(:,i)).

Команды plotmatrix(..., LineSpec) позволяют управлять типом и цветом маркера, используемого для построения полей точек.

Функция [H, AX, GlobAx, P] = plotmatrix(...) возвращает матрицу H дескрипторов графических объектов Line, которые описывают поля точек; матрицу дескрипторов AX локальных координатных осей; дескриптор GlobAx невидимых глобальных координатных осей, в которых размещаются локальные координатные оси; матрицу P дескрипторов объектов Patch, связанных с гистограммами. Глобальные координатные оси не являются текущими осями, поэтому команды title, xlabel, ylabel будут выполняться в локальных координатных осях, описываемых матрицей AX.

Примеры:

Сформируем массив нормально распределенных случайных чисел X размера 50×3, подвергнем его линейному преобразованию и построим матрицу полей точек для нового массива Y.

```
X = randn(50, 3); Y = X*[-1 2 1;2 0 1;1 -2 3;]';
plotmatrix(Y, ' *r ')
```

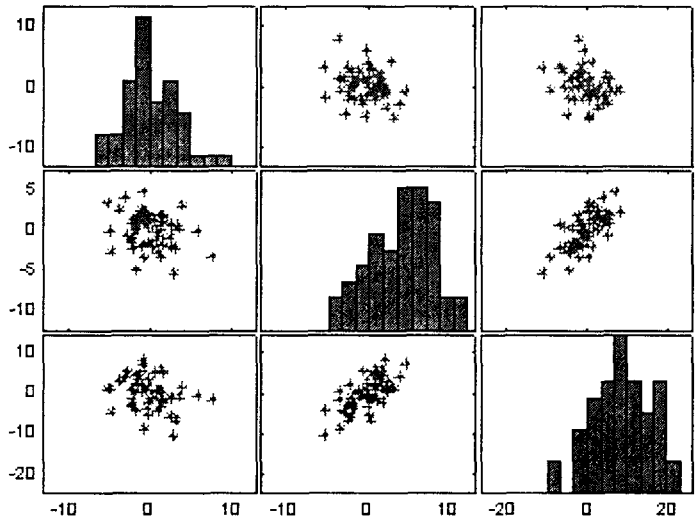


Рис. 10.43

Сопутствующие функции: SCATTER, SCATTER3.

ROSE **Гистограмма в полярных координатах**

Синтаксис:

```
rose(phi)
rose(phi, n)
rose(phi, x)
[phi, r] = rose(phi, ...)
```

Описание:

Команды rose(...) подсчитывают и отображают на графике количество угловых элементов в массиве phi, значения которых попадают в заданный интервал; для этого весь диапазон значений phi делится на n интервалов (по умолчанию 20) и подсчитывается количество угловых элементов в каждом интервале. Такая гистограмма известна под названием розы ветров.

Команда rose(phi) строит розу ветров для 20 интервалов.

Команда rose(phi, n) строит розу ветров для n интервалов.

Команда rose(phi, x) использует вектор x, чтобы задать количество и размещение секторов; длина вектора определяет количество секторов, а значения элементов соответствуют центральному углу каждого сектора.

Функции [phi, r] = rose(...) формируют такие массивы phi и r, что polar(phi, r) является гистограммой в полярных координатах.

Пример:

Построим розу ветров для 50 равномерно распределенных случйных чисел.

```
phi = 2*pi*rand(1,50);
rose(phi)
```

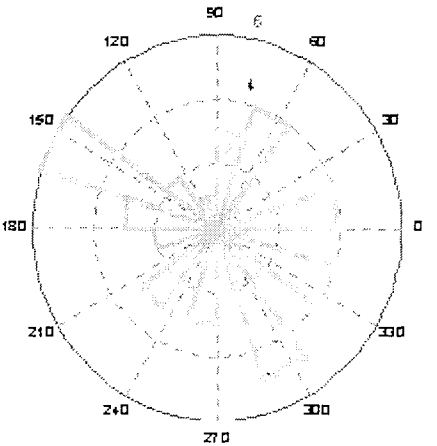


Рис. 10.44

Сопутствующие функции: COMPASS, FEATHER, HIST, POLAR.

COMPASS, FEATHER **Векторные графики**

Синтаксис:

```
compass(z)                    feather(z)
compass(x, y)                 feather(x, y)
compass(..., LineSpec)        feather(..., LineSpec)
h = compass(...)
```

Описание:

Команда compass(z) выводит график комплексных элементов одномерного массива z в виде векторов-стрелок, исходящих из одной точки (начала координат).

Команда compass(x, y) равносильна команде compass(x + i*y).

Команды compass(..., LineSpec) позволяют задать тип линии, символ маркера и цвет при построении векторов-стрелок.

Функция h = compass(...) возвращает вектор дескрипторов графических объектов Line.

Команда feather(z) выводит график комплексных элементов одномерного массива z в виде векторов-стрелок, исходящих из равноотстоящих точек горизонтальной оси.

Команда feather(x, y) равносильна команде feather(x + i * y).

Команды feather(..., LineSpec) позволяют задать тип линии, символ маркера и цвет при построении векторов-стрелок.

Функция h = feather(...) возвращает вектор дескрипторов графических объектов Line.

Примеры:

Построить векторный график собственных значений для случайной матрицы размера 20×20.

```
Z = eig(randn(20, 20));
compass(Z)
```

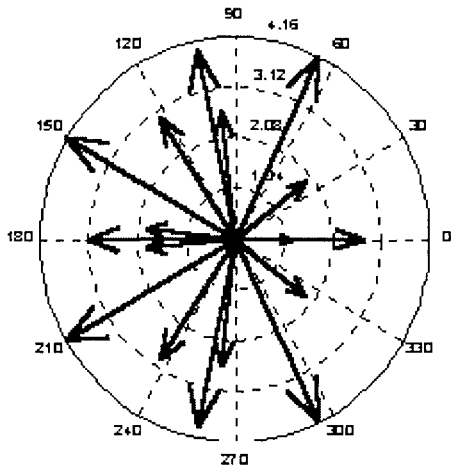


Рис. 10.45

Изобразить в виде векторов-стрелок график изменения углов от -90° до 90° с интервалом 10°.

```
phi = (-90 : 10 : 90)*pi/180;
r = 2*ones(size(theta));
[x, y] = pol2cart(phi, r);
feather(x, y);
```

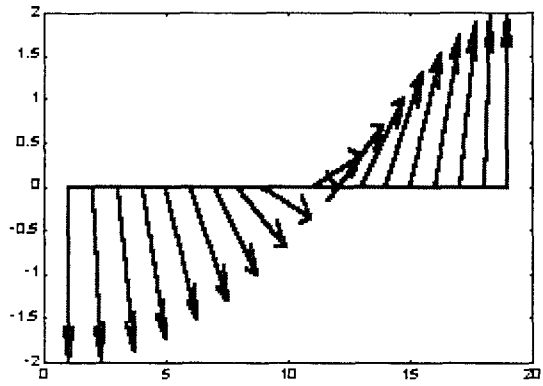


Рис. 10.46

Сопутствующие функции: ROSE, QUIVER.

QUIVER **Поле градиентов функции**

Синтаксис:

```
quiver(X, Y, DX, DY)
quiver(DX, DY)
quiver(..., s)
quiver(..., LineSpec)
quiver(..., LineSpec, ' filled ')
h = quiver(...)
```

Описание:

Команда quiver(X, Y, DX, DY) формирует и выводит на экран поле градиентов функции в виде стрелок для каждой пары элементов массивов X и Y, а пары элементов DX и DY используются для указания направления и размера стрелки. Если x и y - одномерные массивы размеров length(x) = n и length(y) = m, где [m, n] = = size(DX) = size(DY), то формируется и выводится на экран поле градиентов для каждой точки; стрелки задаются четверками {x(j), y(i), DX(i, j), DY(i, j)}. Обратите внимание, что x соответствует столбцам DX и DY, а y - строкам.

Команда quiver(DX, DY) использует массивы x = 1 : n и y = 1 : m.

Команды quiver(..., s) используют скаляр s как коэффициент масштаба стрелки, например s = 2 вдвое увеличивает, а s = 0.5 вдвое уменьшает размер стрелки.

Команды в форме quiver(..., LineSpec) позволяют задать тип линии, символ маркера и цвет при построении векторов-стрелок.

Функция h = quiver(...) возвращает вектор дескрипторов графических объектов Line.

Пример:

Построить поле направлений для функции $z = e^{-(x^2+y^2)}$ в области $-2 \le x \le 2$, $-2 \le y \le 2$.

```
[x, y] = meshgrid(-2 : .2 : 2);
z = x.*exp(-x.^2 - y.^2);
[dx, dy] = gradient(z, .2, .2);
contour(x, y, z), hold on
quiver(x, y, dx, dy)
```

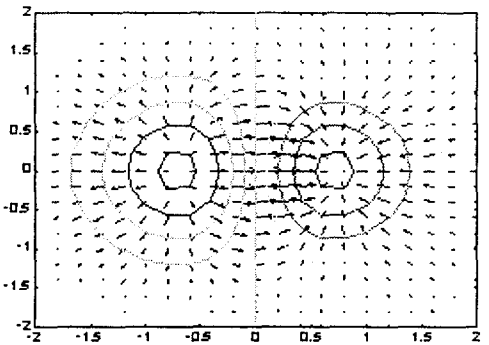


Рис. 10.47

Сопутствующие функции: CONTOUR, PLOT, QUIVER3.

COMET **Движение точки по траектории**

Синтаксис:

```
comet(y)
comet(x, y)
comet(x, y, p)
```

Описание:

Команда comet(y) рисует движение точки по траектории, заданной одномерным массивом y, в виде головы и хвоста кометы.

Команда comet(x, y) рисует движение точки по траектории, заданной массивами x и y.

Команда comet(x, y, p) управляет длиной хвоста кометы p*length(y) с помощью параметра p; по умолчанию p = 0.10.

Замечание:

При формировании траектории движения кометы свойство EraseMode для графического объекта Line имеет значение none. Это означает, что такой график нельзя распечатать; он также исчезнет, если изменить размеры графического окна.

Пример:

Следующая последовательность операторов позволяет наблюдать траекторию движения точки в виде кометы для разности функций tg(sin(t)) - sin(tg(t)).

```
t = -pi : pi/200 : pi;
comet(t, tan(sin(t)) - sin(tan(t)))
```

Сопутствующие функции: COMET3.

AREA — Кусочно-линейные графики с закраской

Синтаксис:

```
area(Y)
area(X, Y)
area(..., ymin)
area(...,'PropertyName', PropertyValue,...)
h = area(...)
```

Описание:

Команды area(...) выводят элементы массива Y в виде одной или нескольких линий и закрашивают пространство между ними. Когда Y - двумерный массив и линии располагаются одна над другой, закрашенные области между ними характеризуют вклад каждого элемента строки массива Y в высоту линии.

Команда area(Y) выводит либо элементы вектора, когда массив Y одномерный, либо сумму элементов каждого столбца, когда массив Y двумерный; при этом ось x масштабируется автоматически либо по длине вектора y, либо по числу строк массива Y.

Команда area(X,Y) в случае одномерных массивов строит зависимость y от x, причем длины векторов должны быть одинаковыми, а элементы вектора x монотонно возрастать. Если массив X двумерный, то должно выполняться условие size(X)=size(Y), а элементы столбцов массива X должны монотонно возрастать. Для расположения элементов вектора в порядке возрастания следует использовать функцию sort.

Команда area(..., ymin) позволяет указать нижний уровень закраски по оси y; по умолчанию ymin = 0.

Команда area(..., 'PropertyName', PropertyValue, ...) позволяет задать пару свойство/значение для графического объекта Patch, порождаемого функцией area.

Функция h = area(...) возвращает вектор дескрипторов для графических объектов Patch; по одному объекту на каждый столбец массива Y.

Замечание:

Команда создает одну линию для всех элементов вектора или для одного столбца массива. Цвета линий выбираются как линейная зависимость цветов выбранной палитры.

Пример:

Вывести значения элементов массива Y в виде кусочно-линейных графиков с закраской областей между ними.

```
Y = [ 1  5  3; 3  2  7; 1  5  3; 2  6  1];
area(Y)
grid on
colormap summer
set(gca, 'Layer', 'top')
```

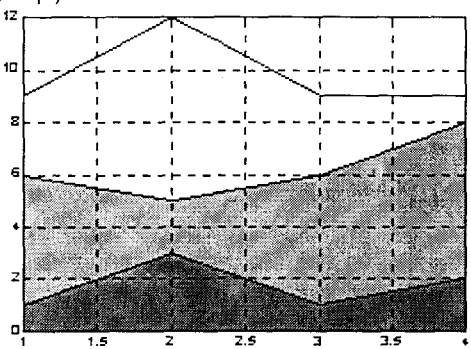


Рис. 10.48

Сопутствующие функции: PLOT.

FILL **Закраска многоугольника**

Синтаксис:

```
fill(X, Y, C)
fill(x, y, '<цвет>')
fill(X1, Y1, C1, X2, Y2, C2, ...)
fill(..., 'PropertyName', PropertyValue)
h = fill(...)
```

Описание:

Команда fill(X, Y, C), где X и Y - массивы одинаковых размеров, строит для каждого столбца свой многоугольник. Если массив C - вектор-строка, количество элементов которой равно числу столбцов массивов X и Y, или массив C - вектор-столбец, количество элементов которого равно числу строк массивов X и Y, то его элементы используются как индексы текущей палитры для задания цветов в вершинах многоугольника. Цвет внутри многоугольника определяется билинейной интерполяцией цветов в узлах. Если массив C двумерный тех же размеров, что и массивы X и Y, то закраска реализуется методом интерполяции, что равносильно применению команды shading interpolated.

Команда fill(X, Y, '<*цвет*>') закрашивает многоугольник, заданный массивами x, y, цветом, который может быть задан либо одним из символов r, g, b, c, m, y, w, k, либо вектором [r g b]. Вершины многоугольника задаются соответствующими парами элементов массивов x, y. Многоугольник должен быть замкнутым, поэтому его первая и последняя вершины, если это возможно, соединяются линией.

Команда fill(X1, Y1, C1, X2, Y2, C2, ...) позволяет выполнить закраску наборов многоугольников разными цветами.

Команда fill(..., 'PropertyName', PropertyValue, ...) позволяет задать пару свойство/значение для графического объекта Patch, порождаемого функцией fill.

Функция h = fill(...) возвращает вектор-столбец дескрипторов для графических объектов Patch, которые соответствуют закрашенным многоугольникам.

Примеры:

Построим многоугольник, соответствующий дорожному знаку STOP.

```
t = (1/16 : 1/8 : 1)'*2*pi;
x = sin(t);
y = cos(t);
fill(x, y, 'b')
axis square
```

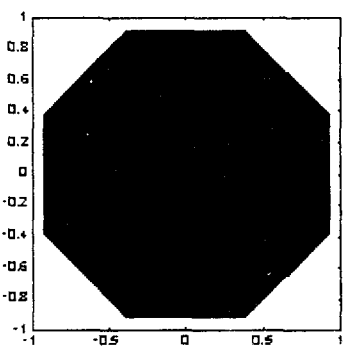


Рис. 10.49

Сопутствующие функции: AXIS, CAXIS, COLORMAP, FILL3, PATCH.

Специальная трехмерная графика

RIBBON — **Ленточная поверхность**

Синтаксис:

```
ribbon(Y)
ribbon(X, Y)
ribbon(X, Y, <ширина>)
h = ribbon(...)
```

Описание:

Команда ribbon(Y) выводит столбцы массива Y в виде ленточных сечений в трехмерном пространстве, при этом массив X определяется в виде вектора x = 1:size(Y,1).

Команда ribbon(X, Y) выводит столбцы массива Y в виде ленточных сечений в трехмерном пространстве, учитывая значения массива X; размеры массивов X и Y должны совпадать.

Команда ribbon(X, Y, *<ширина>*) позволяет задать ширину ленты; по умолчанию этот параметр равен 0.75.

Функция h = ribbon(...) возвращает вектор-столбец дескрипторов для графических объектов Surface, которые соответствуют отдельным лентам.

Примеры:

Построим ленточную поверхность функции peaks.

```
[x, y] = meshgrid(-3 : .5 :3, -3 : .1 : 3);
z = peaks(x, y);
ribbon(y, z)
colormap pink
```

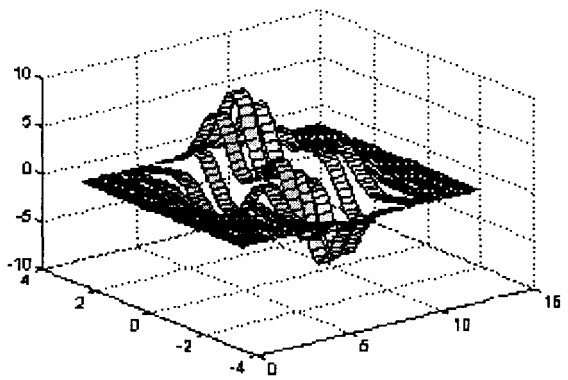


Рис. 10.50

Сопутствующие функции: PLOT, PLOT3, SURFACE.

TRIMESH, TRISURF **Трехмерные триангуляционные поверхности**

Синтаксис:

```
trimesh(Tri, X, Y, Z)
trimesh(Tri, X, Y, Z, C)
trimesh(...'PropertyName', PropertyValue...)
  h = trimesh(...)
```

```
trisurf(Tri, X, Y, Z)
trisurf(Tri, X, Y, Z, C)
trisurf(...' PropertyName ', PropertyValue...)
    h = trisurf(...)
```

Описание:

Команды trimesh(Tri, X, Y, Z) и trisurf(Tri, X, Y, Z) отображают триангуляционную сетку, заданную матрицей Tri размера m×3, в виде сетчатой или сплошной поверхности.

Команды trimesh(Tri, X, Y, Z, C) и trisurf(Tri, X, Y, Z, C) позволяют задать палитру цветов.

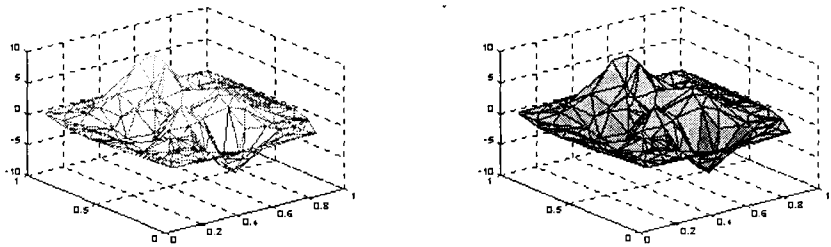
Команды trimesh(...' PropertyName', PropertyValue...) и trisurf(...'PropertyName', PropertyValue...) позволяют задать пару свойство/значение для графического объекта Patch.

Функции h = trimesh(...) и h = trisurf(...) возвращают дескрипторы для графических объектов Patch.

Примеры:

Построим триангуляционные поверхности для функции peaks, определенной на случайной сетке.

```
[x = rand(1, 50);
y = rand(1, 50);
z = peaks(6*x - 3, 6*x - 3);
tri = delaunay(x, y);
% Рис. 10.51,а
trimesh(tri, x, y, z)
% Рис. 10.51,б
trisurf(tri,x,y,z)
```



а *б*

Рис. 10.51

Сопутствующие функции: DELAUNAY, PATCH, SURF.

BAR3, BAR3H **Столбцовые диаграммы**

Синтаксис:

```
bar3(Y)
bar3(x, Y)
bar3(..., <ширина>)
bar3(..., '<стиль>')
bar3(..., LineSpec)
[xb, yb] = bar3(...)
    h = bar3(...)
bar3h(...)
[xb, yb] = bar3h(...)
h = bar3h(...)
```

Описание:

Команды bar3(...), bar3h(...) строят трехмерные вертикальные или горизонтальные столбцовые диаграммы.

Команда bar3(Y) в случае одномерного массива выводит каждый элемент вектора y в виде столбца диаграммы; если Y - двумерный массив, то формируются кластеры (группы) столбцов, соответствующие значениям элементов строки. В этом случае ось x размечается метками по количеству строк массива от 1 до size(Y, 1).

Команда bar3(x, Y) в случае одномерного массива выводит элементы вектора y в виде столбцов диаграммы в позициях, определяемых вектором x, элементы которого должны быть расположены в порядке возрастания; если Y - двумерный массив, то в этом случае в позициях, определяемых вектором x, размещаются кластеры столбцов.

Команда bar3(..., *<ширина>*) устанавливает относительную ширину столбца диаграммы. По умолчанию это значение равно 0.8 в тех случаях, когда отсутствует вектор x; это означает, что между столбцами есть небольшой просвет; если параметр *<ширина>* равен 1, то столбцы, образующие группу, касаются, а при больших значениях перекрывают друг друга.

Команда bar3(..., '*<стиль>*') позволяет задать способ группирования столбцов диаграммы в кластер; параметр *<стиль>* может принимать одно из следующих значений:

<стиль>	*Способ группирования*
'detached'	Для элементов строки формируются отдельные блоки, которые располагаются один за другим вдоль оси x
'grouped'	Формируется n кластеров по m вертикальных столбцов в каждом, где n - число строк, а m - число столбцов массива Y
'stacked'	Формируется n кластеров по одному столбцу, именуемых стеками, при этом высота стека соответствует сумме элементов строки, а вклад каждого элемента определяется частями этого стека, окрашенными в разные цвета

Команда bar3(..., LineSpec) позволяет с помощью параметра LineSpec задать общий цвет для всех столбцов диаграммы.

Функция [xb, yb] = bar3(...) не выводит графика, а формирует такие массивы xb и yb, которые позволяют построить столбцовую диаграмму с помощью команд plot(xb, yb) и patch(xb, yb, C). Это открывает возможности для формирования сложных графиков, позволяя, например, включать в них столбцовые диаграммы.

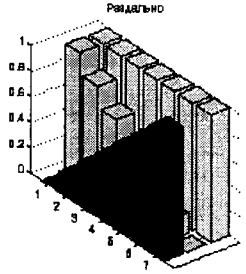
Функция h = bar3(...) возвращает вектор h дескрипторов графического объекта Patch. При этом каждому столбцу массива Y соответствует один графический объект Patch.

Команды и функции bar3h(...), [xb, yb] = bar3h(...), h = bar3h(...) формируют столбцовые диаграммы, которые располагаются горизонтально.

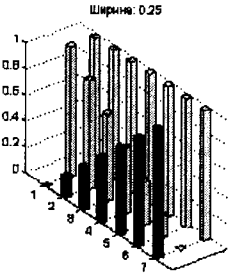
Примеры:

Сформируем случайный массив размера 5×3 и отобразим его в виде четырех разновидностей столбцовых диаграмм.

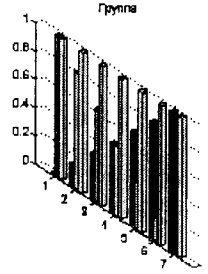
```
Y = round(rand(5, 3)*10);
subplot(2, 2, 1), bar(Y, 'group'),  title 'Группа'
subplot(2, 2, 2), bar(Y, 'stack'),   title 'Стек'
subplot(2, 2, 3), barh(Y, 'stack'), title 'Стек'
subplot(2,2,4), bar(Y,1.5), title 'Ширина: 1.5'
Y = cool(7);
bar3(Y, 'detached'),  title( 'Раздельно')          % Рис. а
bar3(Y, 0.25, 'detached '),  title 'Ширина: 0.25'  % Рис. б
bar3(Y, 'grouped '),  title 'Группа'               % Рис. в
bar3(Y, 0.5, 'grouped '), title 'Ширина: 0.5'      % Рис. г
bar3(Y, 'stacked'),  title 'Стек'                  % Рис. д
bar3(Y, 0.3,'stacked '), title 'Ширина: 0.3'       % Рис. е
colormap([1 0 0; 0 1 0; 0 0 1])
```



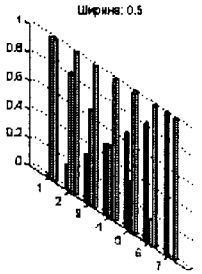
а



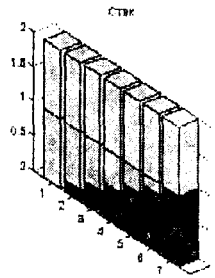
б



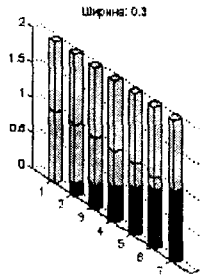
в



г



д



е

Рис. 10.52

Сопутствующие функции: BAR, PATCH.

PIE3 **Секторная диаграмма**

Синтаксис:

```
pie3(x)
pie3(x, v)
h = pie3(...)
```

Описание:

Команда pie3(x) отображает каждый элемент вектора x в виде сектора диаграммы.

Команда pie3(x, v), где v - логический вектор, состоящий из 0 и 1, смещает тот сектор от диаграммы, которому соответствует ненулевой элемент вектора v.

Функция h = pie3(...) возвращает вектор h дескрипторов графических объектов Patch, Surface и Text.

Замечание:

При построении секторной диаграммы используется нормированный вектор x = x/sum(x), который определяет размер каждого сектора. Если sum(x) ≤ 1, то непосредственные значения элементов вектора определяют размеры секторов; когда sum(x) < 1, секторная диаграмма оказывается незамкнутой.

Пример:

Построить секторную диаграмму и отделить от нее сектор 2.

```
x = [1 3 0.5 2.5 2];
v = [0 1 0 0 0];
pie3(x, v)
```

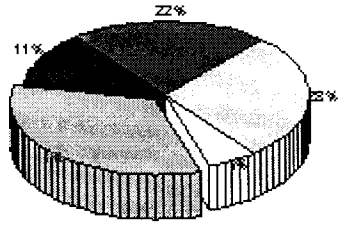


Рис. 10.53

Сопутствующие функции: PIE.

STEM3 **Дискретные графики**

Синтаксис:

```
stem3(z)
stem3(X, Y, Z)
stem3(..., 'fill')
stem3(..., LineSpec)
h = stem3(...)
```

Описание:

Команда stem3(z) выводит график элементов одномерного массива z в виде вертикальных линий, которые заканчиваются в точках графика маркером (по умолчанию - круг); при этом массивы x и y генерируются автоматически. Когда z - вектор-строка, дискретный график располагается вдоль оси x; когда z - вектор-столбец, дискретный график располагается вдоль оси y.

Команда stem3(X, Y, Z) выводит график элементов массива Z в виде вертикальных линий в позициях, определяемых массивами X и Y; все 3 массива должны быть векторами или двумерными массивами одинакового размера.

Команды stem3(..., ' fill ') производят закраску маркера.

Команды stem3(..., LineSpec) позволяют управлять типами линий и маркеров, используемых для построения дискретного графика, по аналогии с командой plot.

Функция h = stem3(...) возвращает вектор дескрипторов h графических объектов Line для вертикальных линий и маркеров.

Примеры:

Построить дискретный график для выборки из 10 чисел, распределенных по нормальному закону.

```
t = 0 : 0.1 : 10;
s = 0.1 + i;
y = exp(-s*t);
h = stem3(real(y), imag(y), t);  view(-37.5, 60)
```

```
set(h(1), 'LineWidth', 2)
set(h(2), 'MarkerSize', 8, 'MarkerEdgeColor', 'r', 'LineWidth', 1)
title('\ite^{st}')
```

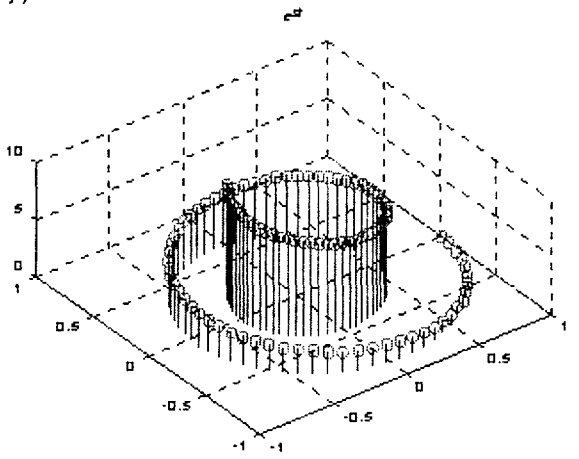


Рис. 10.54

Сопутствующие функции: BAR, PLOT, STAIRS, STEM3.

SCATTER3 — Поле точек

Синтаксис:

```
scatter3(x, y, z, s, c)
scatter3(x, y, z)
scatter3(x, y, z, s)
scatter3(..., <тип_маркера>)
scatter3(..., 'filled')
h = scatter3(...)
```

Описание:

Команда scatter3(x, y, z, s, C) строит поле точек путем вывода цветных маркеров в виде кругов в точках, определяемых векторами x, y, z, которые должны иметь одинаковую длину.

Вектор s той же длины позволяет задать размер каждого маркера как значение points^2, где points - это значение свойства Units (единица измерения) графического объекта Axes. Параметр может быть скаляром, и тогда размеры всех маркеров одинаковы.

Массив C позволяет задать цвет каждого маркера. Если это вектор той же длины, как x и y, то его значения берутся в виде линейной функции из текущей палитры; если массив C имеет размер length(X)×3, то цвета маркера определяются rgb-значениями.

Команда scatter3(x, y, z) выводит цветные маркеры с параметрами Marker = o (*круг*), EdgeColor = [0 0 1] (*синего цвета*) и MarkerSize = 6 (*размер*), принятыми по умолчанию.

Команда scatter3(x, y, z, s) выводит маркеры в виде кругов синего цвета с размерами MarkerSize = sqrt(s).

Команды scatter3(..., <*тип_маркера*>) позволяют изменить тип используемого маркера.

Команды scatter3(..., 'filled) закрашивают маркеры синим цветом (по умолчанию). Цвет маркера управляется параметром MarkerFaceColor.

Функция h = scatter3(...) возвращает вектор h дескрипторов графических объектов Line.

Примеры:

Построим поля точек разных размеров в трехмерном пространстве.

```
[x, y, z] = sphere(16);
X = [x(:)*.5  x(:)*.75  x(:)];
Y = [y(:)*.5  y(:)*.75  y(:)];
Z = [z(:)*.5  z(:)*.75  z(:)];
S = repmat([1  .75  .5]*10, prod(size(x)), 1);
C = repmat([1 2 3], prod(size(x)), 1);
scatter3(X(:), Y(:), Z(:), S(:), C(:), 'filled'),  view(-60,60)
```

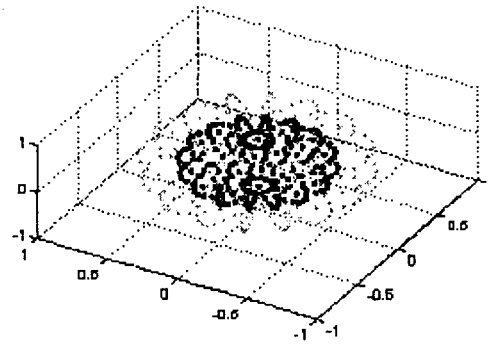


Рис. 10.55

Сопутствующие функции: PLOT3, SCATTER.

QUIVER3 **Поле градиентов функции**

Синтаксис:

```
quiver3(X, Y, Z, DX, DY, DZ)
quiver3(Z, DX, DY, DZ)
quiver3(..., s)
quiver3(..., LineSpec)
quiver3(..., LineSpec, ' filled ')
h = quiver3(...)
```

Описание:

Команда quiver3(X, Y, Z, DX, DY, DZ) формирует и выводит на экран поле градиентов функции в виде стрелок в точках трехмерного пространства, задаваемых элементами массивов X, Y, Z, а элементы DX, DY, DZ используются для указания направления и размера стрелки.

Команда quiver3(Z, DX, DY, DZ) использует целочисленную сетку.

Команды quiver3(..., s) используют скаляр s как коэффициент масштаба стрелки, например s = 2 вдвое увеличивает, а s = 0.5 вдвое уменьшает размер стрелки.

Команды в форме quiver3(..., LineSpec) позволяют задать тип линии, символ маркера и цвет при построении векторов-стрелок.

Функция h = quiver3(...) возвращает вектор дескрипторов графических объектов Line.

Пример:

Построить поле нормалей к поверхности функции $z = e^{-(x^2+y^2)}$ в области $-2 \le x \le 2$, $-1 \le y \le 1$.

```
[X, Y] = meshgrid(-2 : 0.25 : 2, -1 : 0.2 :1);
Z = X.* exp(-X.^2  - Y.^2);
[DX, DY, DZ] = surfnorm(X, Y, Z);
quiver3(X, Y, Z, DX, DY, DZ, 0.5);
hold on
surf(X, Y, Z),  colormap pink
view(-35, 45), axis ([-2 2 -1 1 -.6 .6])
plotedit
```

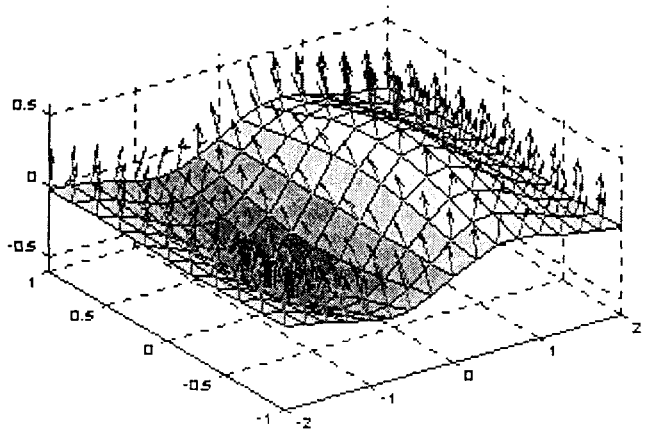


Рис. 10.56

Сопутствующие функции: AXIS, CONTOUR, PLOT, PLOT3, QUIVER, SURFNORM, VIEW.

COMET3 **Движение точки по пространственной траектории**

Синтаксис:

```
comet3(z)
comet3(x, y, z)
comet3(x, y, z, p)
```

Описание:

Команда comet3(z) рисует движение точки по траектории, заданной одномерным массивом z, в виде головы и хвоста кометы.

Команда comet3(x, y, z) рисует движение точки по траектории, заданной массивами x и y.

Команда comet3(x, y, z, p) управляет длиной хвоста кометы p*length(z) с помощью параметра p; по умолчанию p = 0.1.

Замечание:

При формировании траектории движения кометы свойство EraseMode для графического объекта Line имеет значение none. Это означает, что такой график нельзя распечатать; он также исчезнет, если изменить размеры графического окна.

Пример:

Приводимая ниже последовательность операторов позволяет наблюдать траекторию движения точки в виде кометы.

```
t = -10*pi : pi/250 : 10*pi;
comet3((cos(2*t).^2).*sin(t), (sin(2*t).^2).*cos(t), t);
```

Сопутствующие функции: COMET.

SLICE **Сечения функции от трех переменных**

Синтаксис:

```
slice(V, sx, sy, sz)
slice(X, Y, Z, V, sx, sy, sz)
slice(V, Xl, Yl, Zl)
slice(X, Y, Z, V, Xl, Yl, Zl)
slice(..., '<метод>')
h = slice(...)
```

Описание:

Команда slice(V, sx, sy, sz) строит плоские сечения функции от трех переменных V(x, y, z) вдоль осей x, y, z, которые имеют равномерное разбиение x = 1 : n, y = 1 : m, z = 1 : p; позиции сечений определяются векторами sx, sy, sz. Размер трехмерного массива V равен m×n×p, где m = length(y), n = length(x), p = length(z).

Команда slice(X, Y, Z, V, sx, sy, sz) строит плоские сечения функции от трех переменных V(X, Y, Z) вдоль осей x, y, z; позиции сечений определяются векторами sx, sy, sz. Трехмерные массивы X, Y, Z должны быть ортогональны друг к другу, и их элементы должны изменяться монотонно так, как это реализуется функцией meshgrid. Цвет каждой точки определяется 3-D интерполяцией в объеме V.

Команды slice(V, XI, YI, ZI) и slice(X, Y, Z, V, XI, YI, ZI) строят сечения, задаваемые двумерными массивами XI, YI, ZI одинакового размера. Эти массивы определяют поверхность, и функция V вычисляется в точках этой поверхности.

Команда slice(..., '*<метод>*') позволяет задать метод интерполяции, который может принимать одно из следующих значений:

Метод	*Описание*
linear	Интерполяция на основе линейной триангуляции
cubic	Интерполяция на основе кубической триангуляции
nearest	Интерполяция по ближайшим соседям

Функция h = slice(...) возвращает вектор-столбец дескрипторов для графических объектов Surface, которыми являются сечения трехмерной функции.

Пример:

Зафиксируем 3 положения перемещения сферы в некотором объеме.

```
[xsp, ysp, zsp] = sphere;
slice(x, y, z, v, [-2 2], 2, -2) % Создание объема с сечениями
for i = -3 : 3 : 3
  hsp = surface(xsp+i, ysp, zsp);
  rotate(hsp, [1 0 0], 90)
  xd = get(hsp, 'XData');   yd = get(hsp, 'YData');   zd = get(hsp, 'ZData');
  delete(hsp)
  hold on
  hslicer = slice(x, y, z, v, xd, yd, zd);
  axis tight
  xlim([-3 3])
  view(-10, 35)
  drawnow
%delete(hslicer)
  hold off
end
```

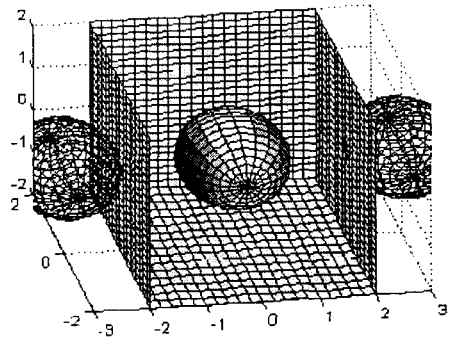


Рис. 10.57

Если при экспериментах с вышеописанным кодом читатель восстановит строку, помещенную в комментарий, и установит шаг изменения переменной в цикле порядка 0.5, то увидит процесс прохождения сферы через этот объем.

Сопутствующие функции: INTERP3, MESHGRID.

WATERFALL — Трехмерная поверхность

Синтаксис:

```
waterfall(Z)
waterfall(X, Y, Z)
waterfall(..., C)
h = waterfall(...)
```

Описание:

Команда waterfall(Z) строит поверхность для значений массива Z, определенных на множествах x = 1:size(Z, 1), y = 1:size(Z, 1). Она аналогична команде mesh, но не прорисовывает ребер сетки. Массив Z задает также палитру цветов, и, таким образом, цвет пропорционален высоте поверхности.

Команда waterfall(X, Y, Z) строит поверхность для значений массива Z, определенных на множествах значений двумерных массивов X, Y. Если массивы одномерные и length(x) = n, length(y) = m, то size(Z) = [m, n] .

Команды waterfall(..., C) используют массив C для задания палитры цветов, и его размер должен совпадать с размером массива Z.

Функция h = waterfall(...) возвращает вектор дескрипторов h для графических объектов Patch.

Замечание:

Поверхности, построенные с помощью функции waterfall, аналогичны поверхностям, построенным с помощью функции mesh, но используют графические объекты Patch. Для создания аналогичных графиков, но использующих графические объекты Surface, необходимо применить функцию meshz и присвоить свойству MeshStyle графического объекта Surface значение 'row'.

Примеры:

Построим трехмерную поверхность функции $z = e^{(-x^2-y^2)}$.

```
[X, Y] = meshgrid([ -2 : 0.1 : 2 ]);
Z = X .*exp(- X .^2 - Y .^2);
waterfall(X, Y, Z)
```

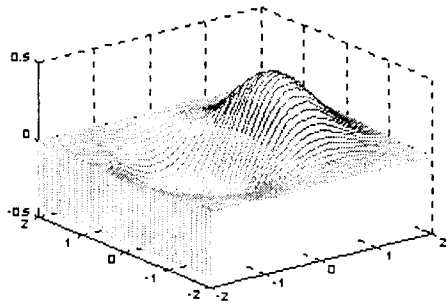


Рис. 10.58

Сопутствующие функции: AXES, AXIS, CAXIS, MESHZ, SURF.

FILL3 **Закраска многоугольников в трехмерном пространстве**

Синтаксис:

```
fill3(X, Y, Z, C)
fill3(X, Y, Z, '<цвет>')
fill3(X1, Y1, Z1, C1, X2, Y2, Z2, C2, ...)
fill3(...,'PropertyName', PropertyValue)
h = fill3(...)
```

Описание:

Команда fill3(X, Y, Z, C) выполняет построение и закраску многоугольников. Массивы X, Y, Z определяют вершины, и для каждого столбца строится свой многоугольник. Если массив C - вектор-строка, количество элементов которой равно числу столбцов массивов X и Y, или массив C - вектор-столбец, количество элементов которого равно числу строк массивов X и Y, то его элементы используются как индексы текущей палитры для задания цветов в вершинах многоугольника.

Команда fill3(X, Y, Z, '*<цвет>*') закрашивает многоугольники цветом, который может быть задан либо одним из символов r, g, b, c, m, y, w, k, либо вектором [r g b].

Команда fill3(X1, Y1, C1, X2, Y2, C2, ...) позволяет выполнить закраску наборов многоугольников разными цветами.

Команда fill3(..., 'PropertyName', PropertyValue, ...) позволяет задать пару свойство/значение для графического объекта Patch, порождаемого функцией fill3.

Функция h = fill3(...) возвращает вектор-столбец дескрипторов для графических объектов Patch, которыми являются закрашенные многоугольники. Команда fill3(...) присваивает свойству Facecolor объекта Patch одно из значений 'flat', 'interp' или [r g b].

Пример:

Построим 4 случайно заданных многоугольника в трехмерном пространстве.

fill3(rand(3, 4), rand(3, 4), rand(3, 4), rand(3, 4))

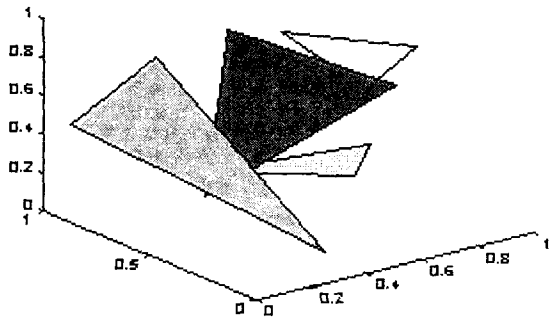


Рис. 10.59

Сопутствующие функции: AXIS, CAXIS, COLORMAP, FILL, PATCH.

VIEWMTX **Вычисление матрицы управления углом просмотра**

Синтаксис:

T = viewmtx(az, el)
T = viewmtx(az, el, phi)
T = viewmtx(az, el, phi, xc)

Описание:

Функция T = viewmtx(az, el) вычисляет матрицу управления углом просмотра, или *обобщенную матрицу преобразований* для *аффинного* изображения [1]. Переменные az и el определяют соответственно углы азимута и возвышения точки просмотра. Положительные значения угла азимута соответствуют вращению вокруг оси z против часовой стрелки. Положительные значения угла возвышения соответствуют точке просмотра, расположенной сверху над объектом, отрицательные - снизу под объектом.

Функция T = viewmtx(az, el, phi) вычисляет матрицу управления углом просмотра для *перспективного* изображения. Угол phi задает поворот системы координат относительно оси x и тем самым позволяет управлять степенью искажения перспективы в соответствии со следующей таблицей.

phi	*Описание*
0°	Аффинное изображение
10°	Телескопическое изображение
25°	Нормальное фотоизображение
60°	Широкоугольное изображение

Функция T = viewmtx(az, el, phi, vt) вычисляет матрицу управления углом просмотра для *перспективного* изображения, используя в качестве дополнительного параметра вектор координат vt наблюдаемой точки (точки наведения). Координаты рассматриваются как нормализованные в диапазоне [0 1]; по умолчанию вектор vt = [0 0 0].

Примеры:

Зададим векторы, которые определяют грани единичного куба:

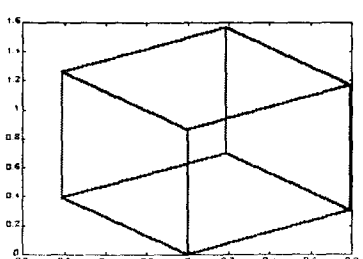
```
x = [0 1 1 0 0 0 1 1 0 0 1 1 1 1 0 0];
y = [0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1];
z = [0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0];
```

Построим аффинное изображение единичного куба.

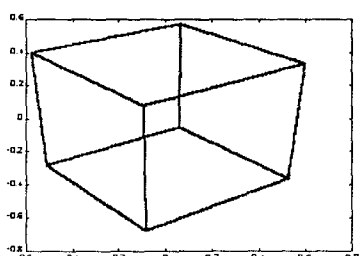
```
A = viewmtx(-37.5, 30);
[m, n] = size(x);
x4d = [x(:), y(:), z(:), ones(m*n,1)]';
x2d = A*x4d;
x2 = zeros(m, n); y2 = zeros(m, n);
x2(:) = x2d(1, :);
y2(:) = x2d(2, :);
plot(x2, y2)  % Рис. 10.60, а
```

Построим перспективное изображение единичного куба.

```
A = viewmtx(-37.5, 30, 25);
[m, n] = size(x);
x4d = [x(:), y(:), z(:), ones(m*n,1)]';
x2d = A*x4d;
x2 = zeros(m, n); y2 = zeros(m, n);
x2(:) = x2d(1, :)./x2d(4, :);
y2(:) = x2d(2, :)./x2d(4, :);
plot(x2, y2)  % Рис. 10.60, б
```



а



б

Рис. 10.60

Сопутствующие функции: VIEW.

Ссылки:

1. Роджерс Д., Адамс Дж. Математические основы машинной графики: Пер. с англ. М.: Машиностроение, 1980. 240 с.

VIEW — Управление положением точки просмотра

Синтаксис:

```
view(az, el)        view(2)
view([az  el])      view(3)
view([x y z])       view(T)
[az, el] = view
T = view
```

Описание:

Команды view(az, el) и view([az el]) задают положение точки просмотра, из которой наблюдается объект, используя углы азимута и возвышения.

Команда view([x, y, z]) задает положение точки просмотра в декартовой системе координат.

Команда view(2) устанавливает штатное положение точки просмотра для двумерной графики: az = 0°, el = 90°.

Команда view(3) устанавливает штатное положение точки просмотра для трехмерной графики: az = -37.5°, el = 30°.

Команда view(T) устанавливает положение точки просмотра в соответствии с обобщенной матрицей преобразований, вычисленной с помощью функции viewmtx.

Функция [az el] = view присваивает текущие значения углов азимута и возвышения соответственно переменным az и el.

Функция T = view присваивает текущее значение обобщенной матрицы преобразований переменной T.

Сопутствующие функции: AXES, VIEWMTX.

ROTATE — Поворот графического объекта

Синтаксис:

```
rotate(h, <ось>, <угол>)
rotate(..., <начало_оси>)
```

Описание:

Команда rotate(h, <*ось*>, <*угол*>) выполняет поворот графического объекта с дескриптором h вдоль оси, направление которой задается вектором <*ось*>, а угол поворота - аргументом <*угол*>. Поворот подчиняется правилу "правой руки". Точка закрепления оси - начало координат.

Команда rotate(..., <*начало_оси*>) позволяет указать новую точку закрепления оси с помощью вектора <*начало_оси*>. По умолчанию точкой закреп-

ления оси является центр трехмерного параллелепипеда, определяющего область размещения графика.

Замечание:

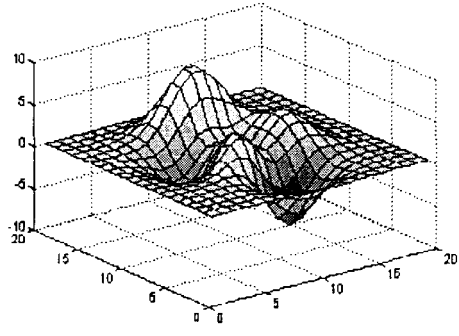
Графический объект, который подвергается вращению, должен быть потомком графического объекта Axes. В процессе поворота данные, которые описывают объект, модифицируются, что отличает команду rotate от команд view и rotate3d, которые изменяют только точку просмотра.

Ось вращения определяется точкой закрепления и еще одной точкой, которая может быть задана либо сферическими углами [az el], либо декартовыми координатами [x y z].

Пример:

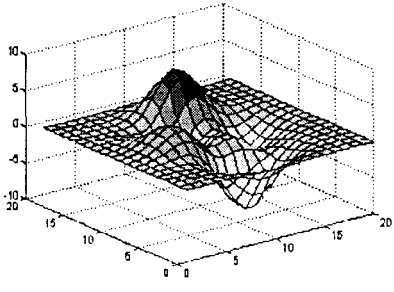
Повернуть графический объект Surface на 180° относительно оси x.

```
h = surf(peaks(20));
```



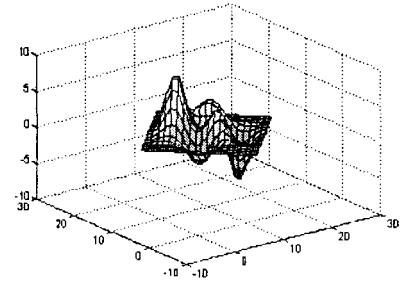
а

```
rotate(h, [1 0 0], 180)
```



б

```
zdir = [0 0 1];
center = [10 10 0];
rotate(h, zdir, 45, center)
```



в

Рис. 10.61

Сопутствующие функции: ROTATE3D, SPH2CART, VIEW.

ROTATE3D **Поворот графического объекта с помощью мыши**

Синтаксис:

rotate3d on | ON
rotate3d off
rotate3d

Описание:

Команды rotate3d on | ON позволяют выполнить поворот осей внутри графического объекта Figure, используя манипулятор мышь. При нажатии на клавишу мыши в области графика появляется анимационный параллелепипед, который изменяет свое положение при перемещении мыши. При этом в случае команды rotate3d on в левом нижнем углу появляется информация о текущих углах азимута и возвышения; в случае команды rotate3d ON эта информация подавляется.

Команда rotate3d off выключает интерактивный режим поворота.

Команда rotate3d выполняет роль переключателя из режимов on | ON в off и обратно.

Создание твердой копии и сохранение графика

PRINT **Вывод графика на печать или в файл**

Синтаксис:

print [-d<*тип_устройства*>] [-<*опция*>] [<*имя_файла*>]

Описание:

Все аргументы команды print являются необязательными, поэтому их можно применять в любой последовательности и в любых комбинациях.

Команда print посылает содержимое активного графического окна для печати на принтер, установленный по умолчанию.

Команда print <*имя_файла*> записывает содержимое активного графического окна в файл в формате устройства, установленного по умолчанию. Если файл с таким именем уже существует, то содержимое будет переписано, если только не использована опция -append. Если аргумент *имя_файла* не имеет расширения, то применяется расширение .ps или .eps, в зависимости от того, какое устройство определено параметром -d<*тип_устройства*>. Если этот параметр не определен, то используется расширение для устройства, принятого по умолчанию, за исключением опций -dmeta и -dbitmap, когда содержимое графического окна направляется в буфер обмена.

В среде MS Windows параметр -d<*тип_устройства*> может принимать следующие значения:

Параметр	*Назначение*
-dwin	Черно-белая печать на подключенном принтере
-dwinc	Цветная печать на подключенном принтере
-dmeta	Поместить в буфер обмена в формате Windows Metafile
-dbitmap	Поместить в буфер обмена в формате Windows Bitmap (BMP)
-dsetup	Активизировать диалоговое окно Print Setup
–v	Активизировать диалоговое окно Print (по умолчанию подавлено)

В состав системы MATLAB включены следующие драйверы для устройств печати в стандарте PostScript:

Параметр	*Назначение*
-dps	Черно-белая печать в стандарте PostScript уровня 1
-dpsc	Цветная печать в стандарте PostScript уровня 1
-dps2	Черно-белая печать в стандарте PostScript уровня 2
-dpsc2	Цветная печать в стандарте PostScript уровня 2
-deps	Черно-белая печать в стандарте Encapsulated PostScript (EPS) уровня 1
-depsc	Цветная печать в стандарте EPS уровня 1
-deps2	Черно-белая печать в стандарте EPS уровня 2
-depsc2	Цветная печать в стандарте EPS уровня 1
-dhpgl	Вывод на плоттер HP 7475A
-dill	Вывод в файл иллюстраций для Adobe Illustrator 88
-dmfile	Вывод в M- или MAT-файл
-dtiff	Вывод в формате RGB TIFF (24 бита) со сжатием (только для рисунков)
-dtiffn	Вывод в формате RGB TIFF (24 бита) без сжатия (только для рисунков)
-djpeg	Вывод графических образов в формате JPEG с фактором качества 75 (только для рисунков)
-djpegnn	Вывод графических образов в формате JPEG с фактором качества nn (00-99) (только для рисунков)

Как правило, файлы в формате PostScript уровня 2 меньше по размерам и выводятся на печать быстрее, чем файлы в формате PostScript уровня 1. Однако не все принтеры поддерживают формат уровня 2, поэтому перед началом печати следует убедиться в этом.

Формат TIFF, используемый для вывода графических образов, поддержан всеми текстовыми редакторами.

Формат JPEG - это формат с высокой степенью сжатия, который используется при обработке изображений и для включения изображений в HTML-документы.

Дополнительные устройства, поддержанные с помощью драйверов постпроцессора GhostScript:

Параметр	*Устройства печати*
-dlaserjet	HP LaserJet
-dljetplus	HP LaserJet+
-dljet2p	HP LaserJet IIP
-dljet3	HP LaserJet III
-ddeskjet	HP DeskJet and DeskJet Plus
-ddjet500	HP DeskJet 500
-dcdjmono	HP DeskJet 500C (черно-белая печать)
-dcdjcolor	HP DeskJet 500C (24 бита/пиксель)
-dcdj500	HP DeskJet 500C
-dcdj550	HP Deskjet 550C
-dpaintjet	HP PaintJet (цветная печать)
-dpjxl	HP PaintJet (цветная печать)
-dpjetxl	HP PaintJet XL (цветная печать)
-dpjxl300	HP PaintJet XL300 (цветная печать)
-ddnj650c	HP DesignJet 650C
-dbj10e	Canon BubbleJet BJ10e
-dbj200	Canon BubbleJet BJ200
-dbjc600	Canon Color BubbleJet BJC-600 и BJC-4000
-dln03	DEC LN03
-depson	Совместимые с Epson 9- и 24-игольчатые принтеры
-depsonc	Epson LQ-2550 и Fujitsu 3400/2400/1200 (цветная печать)
-deps9high	Совместимые с Epson 9-игольчатые принтеры с тройным разрешением
-dibmpro	9-игольчатые IBM-пропринтеры
-dbmp256	Файл в формате BMP (8 бит)
-dbmp16m	Файл в формате BMP (24 бита)
-dpcxmono	Монохромный формат PCX-файла
-dpcx16	Старый формат PCX-файла (EGA/VGA, 16 цветов)
-dpcx256	Новый формат PCX-файла (256 цветов)
-dpcx24b	Файл в формате PCX (24 бита)
-dpbm	Переносимый формат Bitmap (простой формат)
-dpbmraw	Переносимый формат Bitmap (необработанный формат)
-dpgm	Переносимый формат Graymap (простой формат)
-dpgmraw	Переносимый формат Graymap (необработанный формат)
-dppm	Переносимый формат Pixmap (простой формат)
-dppmraw	Переносимый формат Pixmap (необработанный формат)

Следующие опции поддерживаются на всех вычислительных платформах.

Параметр	*Назначение*
-tiff	Добавить к формату Encapsulated PostScript предварительный просмотр цветных образов в формате TIFF
-loose	Учитывать пробелы при использовании устройств EPS и PS
-cmyk	Использовать палитру CMYK вместо RGB в формате PostScript
-append	Добавить к существующему файлу PostScript
-rnumber	Определить разрешение в единицах точка/дюйм
-adobecset	Использовать по умолчанию набор символов PostScript
-fhandle	Вывести на печать графический объект с указанным дескриптором (по умолчанию текущий объект Figure)
-swindowtitle	Вывести на печать окно системы SIMULINK графический объект с указанным именем (по умолчанию текущая система)
-painters	Вывести, используя алгоритм рисования
-zbuffer	Вывести, используя Z-буфер
-noui	Подавить вывод на печать управляющих элементов графического интерфейса пользователя

Сопутствующие функции: FIGURE, ORIENT.

ORIENT **Размещение твердой копии на странице**

Синтаксис:

```
o = orient
orient landscape
orient portrait
orient tall
orient(fig)
orient(fig, <ориентация>)
```

Описание:

Команды группы orient устанавливают значения свойств PaperOrientation и PaperPosition графического объекта Figure.

Функция o = orient возвращает строку, которая указывает способ размещения твердой копии на странице. Это - portrait, landscape, tall. Ориентация portrait означает, что наибольший размер ориентирован сверху вниз; такая ориентация используется по умолчанию. Ориентация landscape означает, что наибольший размер ориентирован слева направо. Размещение tall означает, что при ориентации portrait график занимает всю страницу.

Команда orient landscape размещает твердую копию графического окна на полную страницу с ориентацией landscape.

Команда orient portrait размещает твердую копию графического окна в прямоугольнике с отношением сторон 4/3 в середине страницы.

Команда orient tall размещает твердую копию графического окна на полную страницу с ориентацией portrait, оставляя отступ от края 1/4 дюйма.

Команда orient(fig) возвращает ориентацию графического объекта Figure с дескриптором fig.

Команда orient(fig, <*ориентация*>) устанавливает указанную ориентацию для графического объекта Figure с дескриптором fig.

Сопутствующие функции: PRINT, SET.

11. ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ NOTEBOOK

В состав программного обеспечения системы MATLAB версий 5.x включено новое средство для создания "живых" книг, то есть книг, которые могут быть вычислены из среды редактора Microsoft Word. Таким средством является ППП Notebook.

Понятие М-книги. Документ, созданный в среде ППП Notebook, называется М-книгой. М-книга включает текст, команды системы MATLAB и результаты их выполнения. Ее можно представлять себе либо как запись результатов интерактивного сеанса работы, сопровождаемую текстом, либо как документ, в который помещены исполняемые команды системы MATLAB и их результаты.

Шаблон М-книги. При создании или редактировании М-книги редактор Word использует специальный шаблон M-book. Этот шаблон позволяет получить доступ к системе MATLAB из документа редактора Word и управлять его форматированием.

Когда создается или открывается М-книга, шаблон:

- запускает систему MATLAB, если она не была запущена, и поддерживает динамический обмен данными между Word и MATLAB на основе интерфейса DDE (Dynamic Data Exchange);
- содержит макрокоманды, которые позволяют системе MATLAB обрабатывать специальные типы ячеек, в которые записываются команды и операторы языка MATLAB и результаты их исполнения;
- поддерживает в редакторе Word меню Notebook;
- поддерживает стили для текста и ячеек.

Работа в среде ППП Notebook

Для того чтобы начать работу с ППП Notebook, необходимо прежде всего запустить редактор Word. Следующий шаг состоит в том, чтобы либо открыть новую М-книгу, либо продолжить редактирование существующей.

Создание новой М-книги. Для этого следует выбрать опцию New из меню File редактора Word; затем в этом диалоговом окне выбрать шаблон M-book.dot (рис. 11.1).

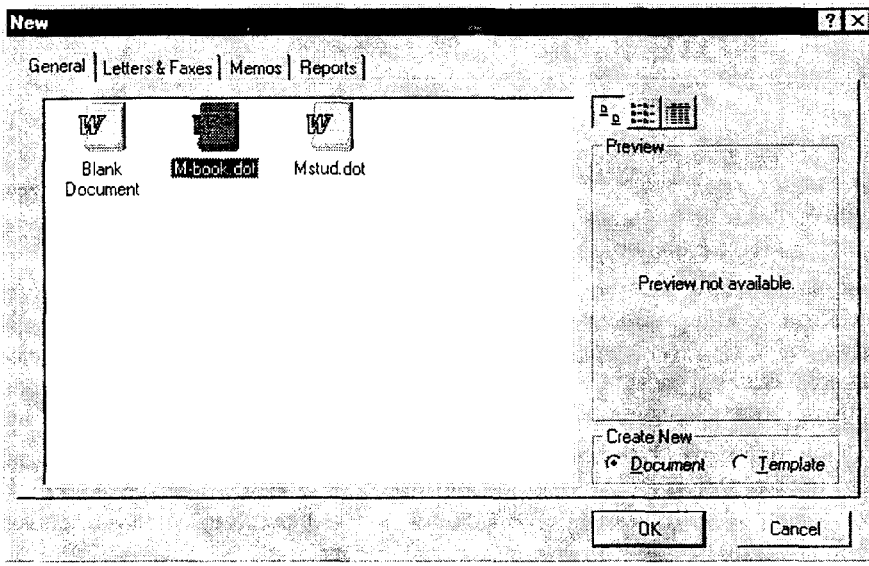


Рис. 11.1

Редактор Word создает новый документ, используя шаблон M-book, добавляет меню Notebook и запускает систему MATLAB (рис. 11.2).

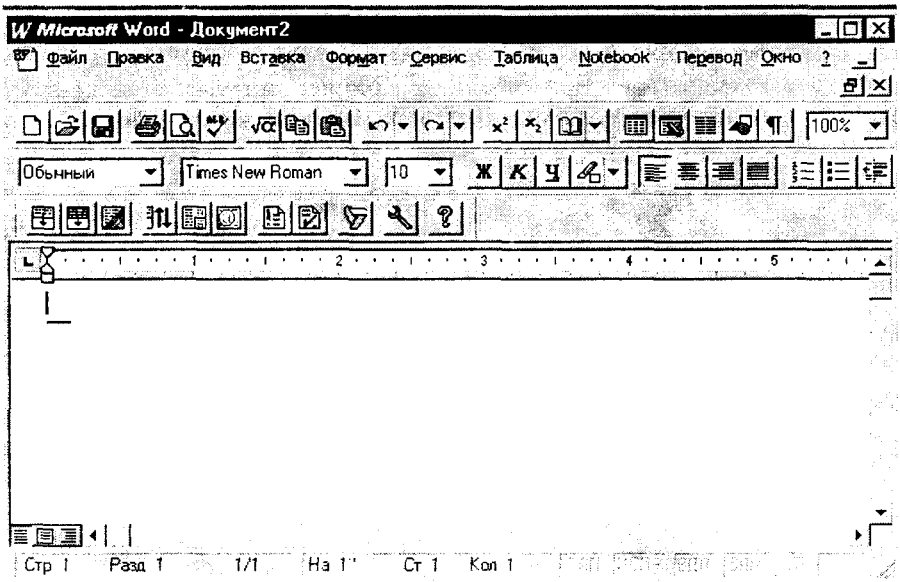


Рис. 11.2

Редактирование М-книги. Для того чтобы приступить к редактированию уже существующей М-книги, надо выполнить одно из следующих действий:

- запустить редактор Word и открыть существующую М-книгу, используя опцию Open из меню File;
- запустить редактор Word, выбрать и открыть М-книгу из списка последних редактированных документов из нижней части ниспадающего меню File;
- дважды щелкнуть на документе M-book.

Редактор Word открывает документ, используя шаблон M-book, стартует систему MATLAB, если она не была активной, и добавляет меню Notebook.

Преобразование документа Word в М-книгу. Для того чтобы конвертировать созданный ранее в редакторе Word документ в М-книгу, необходимо выполнить следующие шаги:

1. Открыть новую М-книгу.
2. Выбрать опцию File из меню Insert.
3. Выбрать файл, который должен быть преобразован.
4. Нажать клавишу Enter.

Написание М-книги

Написание М-книги связано с вводом текста, а также операторов и команд системы MATLAB.

Ввод текста. Эта операция аналогична вводу текста в произвольный документ, создаваемый в редакторе Word. Используя различные стили, можно управлять шрифтами и другими атрибутами представления текста; однако следует иметь в виду, что по умолчанию для текста принят стиль Normal.

Ввод операторов и команд системы MATLAB. Для записи команд и операторов системы MATLAB используются специальные *ячейки ввода*, которые либо включены в текст, либо состоят из одной или нескольких командных строк. Для создания входной ячейки следует:

1. Ввести команду в виде текста и, не нажимая клавиши Enter, оставить курсор в конце текста.
2. Выбрать команду Define Input Cell из меню Notebook либо использовать комбинацию клавиш Alt-D.

ППП Notebook определяет команду как *ячейку ввода*, помещая ее в специальные скобки и форматируя специальным стилем. Все входные ячейки отмечаются жирными скобками серого цвета, которые существенно отличаются от скобок, используемых для обозначения матриц размером и шириной; для изображения символов - жирный шрифт темно-зеленого цвета.

Исполнение команд. Для того чтобы выполнить команду системы MATLAB, ранее не определенную в качестве входной ячейки, необходимо:

Ввести команду как текст, оставить курсор на этой строке в конце текста и не нажимать клавишу Enter;

1. Ввести команду в виде текста и, не нажимая клавиши Enter, оставить курсор в конце текста.
2. Выбрать команду Evaluate Cell из меню Notebook либо использовать комбинацию клавиш Ctrl-Enter.

ППП Notebook кроме ячеек ввода использует также ячейки вывода, чтобы сохранить вычисленные результаты. Ячейки вывода следуют непосредственно за ячейками ввода и помечаются специальными скобками; для вывода чисел и текста используются символы синего цвета; сообщения об ошибках выводятся символами красного цвета.

Пример:

1. Напечатаем в строке команду системы MATLAB: a = magic(3)
2. Используем команду Evaluate Cell из меню Notebook или комбинацию клавиш Ctrl-Enter.

ППП Notebook отобразит команду как ячейку ввода и выведет результат в ячейку вывода:

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

Многострочные ячейки ввода. Если вводятся команды MATLAB, которые занимают несколько строк, то необходимо обязательно выделить все строки, чтобы либо определить их как ячейки ввода, либо вычислить их.

Автоматическая инициализация команд. Для автоматической инициализации команд при открытии М-книги необходимо определить команды как *ячейки автостарта* (autoinit cells). Это наиболее быстрый и простой способ формирования рабочей области. Ячейки автостарта - это те же ячейки ввода со следующими дополнительными свойствами:

- ППП Notebook вычисляет ячейки автостарта при открытии М-книги;
- команды в ячейках автостарта изображаются символами темно-синего цвета.

Создание ячеек автостарта. Создать ячейки автостарта можно двумя способами:

1. Ввести команды в виде текста, а затем определить их как ячейку автостарта, используя команду Define AutoInit Cell.
2. Если команды определены как ячейка ввода, то ее можно конвертировать в ячейку автовызова, используя команду Define AutoInit Cell.

Пример:

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

Принудительное вычисление ячейки автостарта. Для принудительного вычисления ячейки автостарта следует:

1. Позиционировать курсор в ячейке автостарта.
2. Использовать команду Evaluate Cell или сочетание клавиш Ctrl-Enter.

Объединение команд в группы

ППП Notebook позволяет вводить последовательность команд системы MATLAB и работать с ней как с группой ячеек ввода. *Группа ячеек* - это многострочная ячейка ввода или автостарта, которая включает более одной команды.

В группу ячеек нельзя включать текст или ячейки вывода. Результаты вычисления группы ячеек хранятся в одной ячейке вывода, которая располагается непосредственно за группой ячеек ввода.

При создании группы ячеек ППП Notebook определяет ее как ячейку ввода, если только первая строка не является ячейкой автостарта; в последнем случае вся группа объявляется ячейкой автостарта.

Если группа ячеек включает команды, которые производят текстовый или числовой выход, а также команды графического вывода, то числовые данные и текст всегда выводятся первыми, независимо от действительной последовательности команд в группе.

Группы ячеек чрезвычайно полезны, когда несколько команд полностью задают графический образ. Если это сделано именно так, то формируется единственный график, который отражает все свойства, заданные в командах. Если же команда построения графика оформлена в виде отдельной ячейки, то вычисляемые ячейки генерируют множество графиков.

Создание группы ячеек. Для создания группы ячеек необходимо:

1. Выделить ячейки ввода, которые предполагается объединить в группу: если в составе выделенных оказались ячейки вывода, то они удаляются; если выделенный фрагмент включает текст, то он размещается после группы, за исключением того случая, когда текст предшествует первой ячейке ввода; если оказалась выделенной часть или вся ячейка вывода и не затронута ячейка ввода, то в группу включается соответствующая входная ячейка.
2. Применить команду Group Cells или комбинацию клавиш Alt-G.

ППП Notebook преобразовывает выделенные ячейки в общую группу и заменяет маркеры ячеек единственной парой скобок.

Вычисление группы ячеек. Для вычисления группы ячеек необходимо выполнить те же операции, что и для выполнения ячейки ввода, а именно:

1. Позиционировать курсор в любом месте группы ячеек или в ячейке вывода.
2. Использовать команду Evaluate Cell или комбинацию клавиш Ctrl-Enter.

При вычислении группы ячеек результат размещается в единственной ячейке вывода. По умолчанию ячейка вывода располагается сразу за группой ячеек, как только начинаются вычисления. Если вычисления выполняются при наличии ячейки вывода, то результат помещается в нее.

Пример:

Этот пример показывает шаги, связанные с преобразованием ячеек ввода в группу ячеек и последующим ее вычислением.

```
Записать ячейки ввода:
   t = 0:pi/10:2*pi;
   [X, Y, Z] = cylinder(4*cos(t)+1);
   mesh(X, Y, Z)
Выделить ячейки:
    t = 0:pi/10:2*pi;
    [X, Y, Z] = cylinder(4*cos(t)+1);
    mesh(X, Y, Z)
Применить команду Group Cells, чтобы создать группу ячеек
   t = 0:pi/5:2*pi;
   [X, Y, Z] = cylinder(4*cos(t)+1);
   mesh(X, Y, Z)
Чтобы вычислить группу ячеек, воспользуйтесь командой
Evaluate Cell или комбинацией клавиш Ctrl-Enter.
```

На экран будет выведено то же, что изображено на рис. 11.3

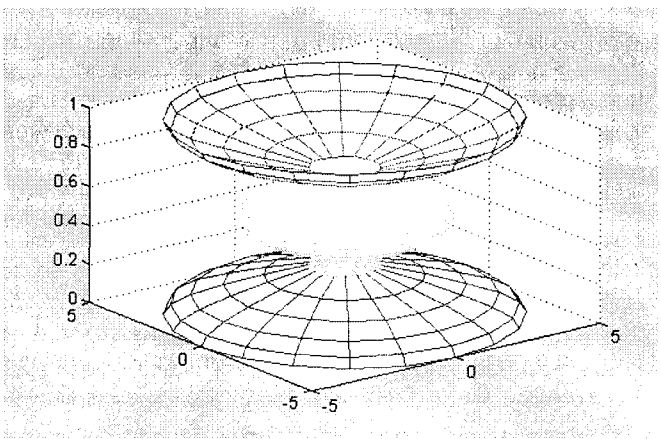


Рис. 11.3

Использование команд и операторов MATLAB внутри текста

ППП Notebook позволяет помещать команды и операторы MATLAB непосредственно в текст строки или параграфа. Для этого надо выполнить такую последовательность операций:

1. Напечатать текст вместе с оператором или командой.
2. Выделить в тексте команду или оператор.
3. Применить команду Evaluate Cell или комбинацию клавиш Ctrl-Enter.

Рассмотрим эту возможность на следующем примере.

Пример:

Команда `z = magic(3)` генерирует магический квадрат размера 3x3

```
z =
     8     1     6
     3     5     7
     4     9     2
```

Как показывает этот пример, ППП Notebook преобразовывает команду в ячейку ввода, посылает на исполнение в систему MATLAB и выводит результаты в ячейке вывода.

Зоны вычислений

Существует возможность разбивать М-книгу на автономные секции, называемые *зонами вычислений.* Зона вычислений - это непрерывный блок, который включает текст, ячейки ввода и вывода, связанные с описанием некоторой операции или проблемы. ППП Notebook определяет зону вычислений как секцию документа и помечает ее начало и конец, за исключением начала и конца документа. М-книга может содержать любое количество таких зон.

Преподаватели могут использовать эту возможность, чтобы подготовить, например, множество контрольных вопросов или задач. Определяя для каждой задачи зону вычислений, можно создавать и отлаживать их автономно. Стоит отметить, что переменные отдельной зоны не являются локальными переменными только этой зоны.

Задание зон вычислений. Как только написан текст и созданы ячейки ввода, можно определить зону вычислений, используя следующие шаги:

1. Выделить ячейки и текст, включаемые в зону вычислений.
2. Применить команду Define Calc Zone.

Если в документе уже существуют как ячейки ввода, так и ячейки вывода, то при определении зоны вычислений должны быть выделены и те и другие.

Вычисление зон. Для того чтобы исполнить команды зоны вычислений, необходимо:

1. Позиционировать курсор в зоне вычислений.
2. Применить команду Evaluate Calc Zone или комбинацию клавиш Alt-Enter.

ППП Notebook посылает каждую ячейку ввода из зоны вычислений в систему MATLAB для исполнения. По умолчанию Notebook размещает ячейку вывода сразу после зоны вычислений. Если зона вычислений включает ячейку вывода, то результат помещается в эту ячейку, где бы ни было ее расположение в создаваемой М-книге.

Преобразование ячейки в текст

Для того чтобы преобразовать ячейку ввода (ячейку автовызова или группу ячеек) в текст, необходимо:

1. Позиционировать курсор в любом месте ячейки.
2. Применить команду Undefine Cells или комбинацию клавиш Alt-U.

Notebook преобразует ячейку в текст, применяя стиль Normal, при этом ячейка вывода во внимание не принимается.

Вынесение окна MATLAB на передний план. Для размещения окна MATLAB на переднем плане следует использовать команду Bring MATLAB to Front.

Поддержание целостности рабочей области. Когда в одном сеансе работы с редактором Word обрабатывается более одной М-книги, выполняются следующие условия:

- все М-книги используют одну и ту же копию системы MATLAB (один процесс);
- все М-книги используют одну и ту же рабочую область.

Если несколько одинаковых имен для переменных используются в нескольких М-книгах, то возможно их взаимное влияние. Чтобы обеспечить целостность рабочей области для каждой М-книги, надо в первой ячейке автовызова для каждой М-книги определить команду clear.

Вычисление ячеек

М-книгу можно рассматривать как дневник сеанса работы с системой MATLAB, при этом в М-книге аккуратно отслеживаются все связи между использованными операторами. Однако, если приходится изменять или удалять ячейку ввода в процессе написания М-книги, надо помнить, что Notebook не выполняет автоматического перевычисления ячеек, которые могут оказаться зависимыми от внесенных изменений. В результате может оказаться нарушенной непротиворечивость данных.

При работе над книгой целесообразно периодически применять команду Evaluate M-book, чтобы гарантировать непротиворечивость используемых

данных. Можно применять механизм зон вычислений, чтобы объединить связанные команды в отдельную секцию М-книги, а затем применить команду Evaluate Calc Zone, чтобы исполнить их.

В этом разделе приведены особенности вычисления отдельных ячеек, последовательности ячеек, вычислений в цикле и вычисления М-книг.

Вычисление ячеек ввода, ячеек автовызова и групп ячеек. Для того чтобы вычислить такие ячейки, необходимо:

1. Позиционировать курсор в любом месте ячейки ввода или в соответствующей ячейке вывода.
2. Применить команду Evaluate Cell или комбинацию клавиш Ctrl-Enter.

Если ячейки вывода отсутствуют, то Notebook размещает их сразу после ячейки ввода; если ячейки вывода уже созданы, то новые результаты размещаются в них, где бы в книге они ни находились.

Вычисление последовательности ячеек ввода. Для вычисления более чем одной команды или оператора системы MATLAB, размещенных в разных, но непрерывно следующих одна за другой ячейках ввода, необходимо:

1. Выделить последовательность ячеек и текст, который включает ячейки ввода.
2. Применить команду Evaluate Cell или комбинацию клавиш Ctrl-Enter.

Notebook вычисляет каждую ячейку ввода в выделенном фрагменте, создавая, если необходимо, ячейки вывода или размещая результат в существующих ячейках.

Вычисление М-книги в целом. Для этого следует использовать команду Evaluate M-book или комбинацию клавиш Alt-R. ППП Notebook начинает вычисление М-книги с самого начала, независимо от места расположения курсора, и вычисляет каждую ячейку. По мере вычисления Notebook включает новые ячейки вывода или размещает результаты в существующих ячейках.

Контроль вычисления ячеек. Для контроля результатов исполнения ячеек при вычислении М-книги рекомендуется использовать опцию Stop evaluating on error. Если опция включена, то при возникновении ошибки дальнейшее вычисление прекращается; если нет, то вычисления выполняются полностью, независимо от имеющихся ошибок.

Вычисление зон. Для вычисления зон необходимо:

1. Позиционировать курсор в любом месте зоны.
2. Применить команду Evaluate Calc Zone или комбинацию клавиш Alt-Enter.

Notebook вычисляет зону независимо от места расположения курсора и создает, если необходимо, ячейки вывода или размещает результат в существующих ячейках.

Вычисление команд в цикле. Для того чтобы вычислить последовательность команд повторно, необходимо:

1. Выделить ячейки ввода, включая текст и ячейки вывода, размещенные между ними.
2. Применить команду Evaluate Loop или комбинацию клавиш Alt-L. ППП Notebook выведет на экран следующую диалоговую панель (рис. 11.4).

```
[a =
     8     1     6
     3     5     7
     4     9     2]

[a = a + a ]

[a =
   256    32   192
    96   160   224
   128   288    64]
```

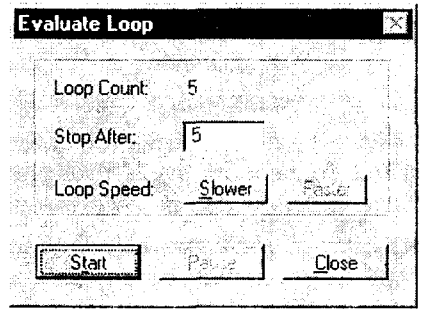


Рис. 11.4

С помощью этой диалоговой панели можно реализовать следующие операции:

- задать в поле Stop After количество циклов вычисления команд или операторов;
- запустить вычисления, используя кнопку Start, которая при этом инвертируется в кнопку Stop;
- приостановить выполнение команд с помощью кнопки Pause, которая при этом преобразовывается в кнопку Continue;
- изменить скорость вычислений с помощью кнопок Faster и Slower;
- прервать выполнение цикла, используя кнопку Stop.

Пример:

Для матрицы

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

выполнить следующий оператор 3 раза:

```
a = a+a
a =
    64     8    48
    24    40    56
    32    72    16
```

Операции с результатами вычислений

В этом разделе обсуждаются возможные операции с результатами вычислений, выполняемых в М-книге, в том числе операции над ячейками вывода, вывод числовой и графической информации, а также печать результатов.

Операции с ячейками вывода. Если вычисляется ячейка ввода, для которой не создано ячейки вывода, ППП Notebook размещает ячейку вывода сразу за ячейкой ввода; если ячейка вывода уже существует, то результат помещается в нее, где бы она ни располагалась в М-книге.

Если вычисляется группа ячеек, то результат следует сразу за этой группой и занимает одну ячейку вывода.

Вывод результатов вычислений на терминал. При выводе результатов вычислений на терминал с помощью команды Notebook Options можно управлять форматом вывода числовых данных и размерами графического окна.

Преобразование ячеек вывода в текст. Для преобразования ячеек вывода в текст применяется команда Undefine Cells. Если выходом являются числовые данные или текст, то Notebook удаляет маркеры ячейки и преобразовывает содержимое в текстовый формат, используя стиль Normal. Если выходом является графика, то Notebook удаляет маркеры ячейки вывода, сохраняя ее содержимое. При этом команда никак не действует на соответствующую ячейку ввода.

Для того чтобы преобразовать ячейку вывода в формат текста книги, необходимо выполнить следующие действия:

1. Выделить преобразовываемую ячейку.
2. Применить команду Undefine Cells или комбинацию клавиш Alt-U.

Для того чтобы удалить ячейку вывода, используйте команду Purge Output Cells или комбинацию клавиш Alt-P.

Вывод на печать М-книги. Для этого используется команда Print из меню File. Редактор Word соблюдает следующие правила при выводе на печать ячеек М-книги:

- маркеры ячеек не выводятся на печать;
- ячейки ввода, автовызова и вывода (включая сообщения об ошибках) печатаются в соответствии с выбранными для них стилями. Если требуется черно-белая печать вместо цветной и оттенков серого, то следует внести соответствующие изменения в стили.

Изменение стилей в шаблоне М-книги. Шаблон М-книги, в котором предопределены стили для текста и ячеек, использует стиль Normal редактора Word для печати текста и стили, задаваемые ППП Notebook, для распечатки ячеек.

Так, если М-книга распечатывается на цветном принтере, то ячейки ввода печатаются темно-зеленым цветом, ячейки вывода - синим, а ячейки автовызова - темно-синим цветом; сообщения об ошибках - красным цветом.

Если М-книга распечатывается на черно-белом принтере, то эти цвета преобразовываются в оттенки серого. Если требуется только черный цвет, то необходимо изменить цвета в стилях Input, Output, AutoInit и Error.

Следующая таблица описывает стили, принятые по умолчанию в ППП Notebook. Если требуется изменить стиль, то эта таблица поможет в дальнейшем вернуться к первоначальным стилям.

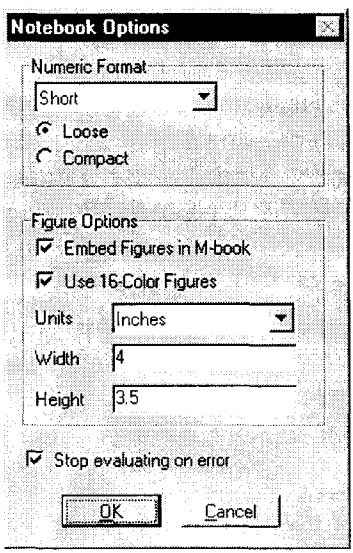
Стиль	*Шрифт*	*Размер, пункт*	*Выделение*	*Цвет*
Normal	Times New Roman	10		Черный
AutoInit	Courier New	10	Bold	Темно-синий
Error	" "	10	"	Красный
Input	" "	10	"	Темно-зеленый
Output	" "	10		Синий

Предупреждение:

Если вы изменяете стиль, то редактор Word применяет его ко всем символам М-книги, которые используют его, и предлагает опцию, чтобы изменить шаблон. Будьте внимательны, внося изменения в шаблон, поскольку это означает, что эти изменения отразятся на всех вновь создаваемых М-книгах.

Управление форматом вывода чисел

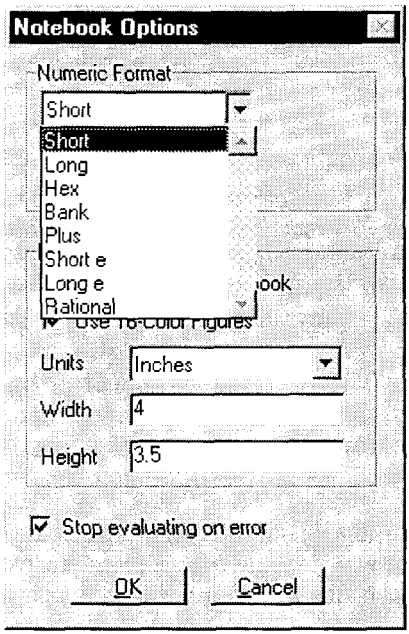
Управление форматом вывода чисел реализуется с помощью опции Numeric Format диалогового окна Notebook Options (рис. 11.5). Для доступа к диалоговому окну необходимо применить команду Notebook Options.



←——Установка формата чисел

Рис. 11.5

Опция Numeric Format реализована в виде ниспадающего списка, который соответствует всем возможностям команды format системы MATLAB. На рис. 11.6 показан список доступных форматов.



← Список числовых форматов

Рис. 11.6

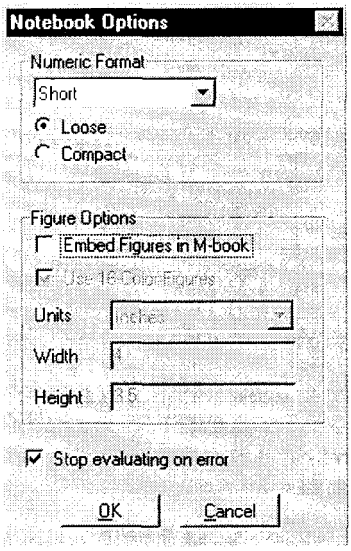
Кроме того, можно управлять использованием пробела между ячейками ввода и ячейками вывода: формат Loose добавляет пробел между ячейками, а формат Compact - нет.

После выбора всех опций формата надо подтвердить установки нажатием клавиши Enter. Эти установки не будут задействованы до тех пор, пока не будет перевычислена соответствующая ячейка ввода.

Управление графическим выводом

Среди опций команды Notebook Options присутствуют опции Figure Options, которые позволяют управлять включением рисунка в М-книгу, его размером, а также способом вывода на печать (палитра из 16 или 256 цветов).

Включение рисунка в текст М-книги. По умолчанию создаваемые рисунки помещаются в М-книгу, однако можно отменить опцию Embed Figures in M-book, и в этом случае рисунок будет виден только в графическом окне системы MATLAB (рис. 11.7).



Отменить этот флажок для отображения графика в отдельном окне

Рис. 11.7

Рисунки, размещаемые в тексте, не включают объектов графического интерфейса пользователя GUI, генерируемых функциями uicontrol и uimenu.

ППП Notebook проверяет, включать рисунок в М-книгу или нет, проверяя свойство фигуры Visible; если его состояние off, то рисунок включается в книгу, если on, то рисунок отображается только в графическом окне.

Управление выводом графика. Команда Toggle Graph Output for Cell разрешает или подавляет вывод рисунка для данной ячейки ввода. Для ее применения надо разместить курсор в поле ячейки ввода и выбрать эту команду из меню Notebook Options. В этом случае в строке команды графического вывода появится уведомление (no graph), а вслед за ячейкой ввода появится пустая строка, чтобы обозначить, что графический вывод подавлен. Для того чтобы разрешить графический вывод, надо разместить курсор в поле ячейки ввода и вновь воспользоваться командой Toggle Graph Output for Cell. При этом уведомление (no graph) исчезнет. Эта команда подавляет опцию Embed Figures in M-book, если она была установлена.

Управление размером рисунка. Размер рисунка устанавливается с помощью команды Notebook Options; указывая в полях Width и Height ширину и высоту рисунка и нажимая клавишу Enter, можно зафиксировать необходимый размер поля. Эти изменения вступают в действие только после пересчета соответствующей ячейки ввода.

Можно также изменять размеры фигуры, манипулируя ее графическими дескрипторами. Выделите фигуру, нажимая на левую кнопку мыши в ее произвольном месте; Word выделит контур этого рисунка, который выполняет роль графического дескриптора. Перемещение углового дескриптора корректирует положение смежных сторон, а перемещение срединного дескриптора корректирует расположение самой стороны. Если после таких изменений рисунок перевычисляется заново, то он восстанавливает свои первоначальные размеры.

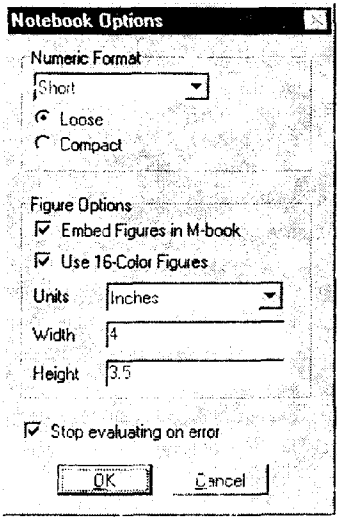
Удаление пустого пространства вокруг графика. Существует возможность вырезать области пустого пространства вокруг рисунка. Для этого, удерживая клавишу Shift, следует перемещать дескриптор установки размеров так, чтобы удалить ненужные области.

Добавление пространства вокруг графика. Для этого следует, удерживая клавишу Shift, перемещать дескриптор установки размеров от центра рисунка, регулируя размеры контура.

Вывод графика на печать. Если на печать выводится рисунок, который включает поверхности или многоугольники, то по умолчанию используется палитра из 16 цветов. Если требуется палитра из 256 цветов, то надо убедиться, что опция Use 16-Color Figures выключена (рис. 11.8).

Замечание:

Эта опция будет применяться только к вновь генерируемому графику. Таким образом, чтобы использовать нужную палитру, надо ее сначала установить, а уже затем сгенерировать график.



Флажок выбора 16- или 256-цветной палитры при печати рисунка

Рис. 11.8

Команды ППП Notebook

Ниже в порядке их следования описаны команды, представленные в меню Notebook редактора Microsoft Word и используемые ППП Notebook при работе с М-книгой (рис. 11.9).

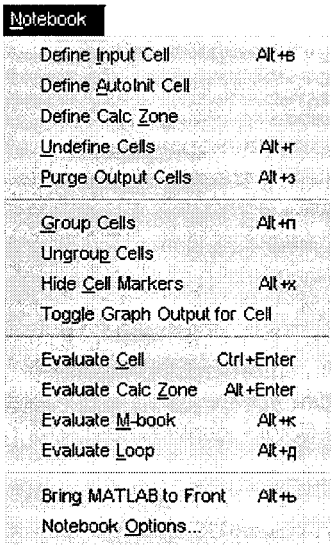


Рис. 11.9

Define Input Cell — **Создать ячейку ввода**

Синтаксис:

Define Input Cell

Описание:

Команда Define Input Cell создает ячейку ввода, включая в нее текущий параграф, выделенный текст или ячейку автовызова.

Если вы воспользуетесь этой командой, когда курсор находится в тексте параграфа, то весь параграф будет преобразован в ячейку ввода. Если выделен некоторый фрагмент текста, то именно он и будет преобразован в ячейку ввода. Если курсор находится в поле ячейки автовызова, то по этой команде она будет преобразована в ячейку ввода.

Формат ячейки ввода использует стиль Input, который включает жирный шрифт Courier New кегля 10 темно-зеленого цвета.

Define AutoInit Cell — **Создать ячейку автовызова**

Синтаксис:

Define AutoInit Cell

Описание:

Команда Define AutoInit Cell создает ячейку автовызова, включая в нее текущий параграф, выделенный текст или ячейку ввода. В дальнейшем ячейка автовызова будет вычисляться автоматически при открытии М-книги.

Если вы воспользуетесь этой командой, когда курсор находится в тексте параграфа, то весь параграф будет преобразован в ячейку автовызова. Если выделен некоторый фрагмент текста, то именно он и будет преобразован в ячейку автовызова. Если курсор находится в поле ячейки ввода, то по этой команде она будет преобразована в ячейку автовызова.

Формат ячейки автовызова использует стиль AutoInit, который включает жирный шрифт Courier New кегля 10 темно-синего цвета.

Define Calc Zone — **Создать зону вычисления**

Синтаксис:

Define Calc Zone

Описание:

Команда Define Calc Zone определяет выделенный текст, ячейки ввода и вывода в качестве зоны вычисления. Эта зона объединяет текст, ячейки ввода и вывода, которые вместе описывают некоторую операцию или задачу.

ППП Notebook определяет зону вычисления как секцию документа, выделяя ее маркерами, однако маркеры не выводятся, секция располагается в начале и в конце документа.

Undefine Cells — **Преобразовать ячейки в текст**

Синтаксис:

Undefine Cells

Описание:

Команда Undefine Cells преобразовывает выделенные ячейки в текст. Если никакие ячейки не выделены, но курсор находится в некоторой ячейке, то Notebook преобразовывает только эту ячейку, при этом удаляются маркеры ячейки, а ее формат соответствует стилю Normal.

При преобразовании ячейки ввода автоматически преобразовывается и ячейка вывода, но не наоборот. Если преобразовывается ячейка вывода, содержащая графику, то графический вывод в М-книге сохраняется, но он более не будет ассоциирован с ячейкой ввода.

Purge Output Cells — **Удалить ячейки вывода**

Синтаксис:

Purge Output Cells

Описание:

Команда Purge Output Cells удаляет все выделенные ячейки вывода.

Group Cells — **Создать многострочную ячейку ввода**

Синтаксис:

Group Cells

Описание:

Команда Group Cells преобразовывает выделенные ячейки ввода в единственную многострочную ячейку ввода, которая называется *группой ячеек*.

Если выделенный фрагмент включает текст, то Notebook помещает его после группы ячеек. Однако если текст предшествует первой ячейке ввода в группе, то он остается на месте.

Если выделенный фрагмент включает ячейки вывода, то Notebook удаляет их. Если перед использованием команды выделены все или часть ячеек вывода, то Notebook включает соответствующие ячейки ввода в группу ячеек.

Если первая строка в группе ячейки является ячейкой автовызова, то вся группа ячеек воспринимается как последовательность ячеек автовызова; в противном случае группа равносильна последовательности ячеек ввода. Группу ячеек можно преобразовать в ячейку автовызова командой Define AutoInit Cell.

Структура группы наиболее полезна, когда, например, последовательность команд или операторов формирует один график.

Вычислить группу ячеек можно с помощью команды Evaluate Cell, при этом все результаты вычислений будут размещаться в единственной ячейке вывода.

Ungroup Cells — **Преобразовать группу ячеек в ячейки ввода**

Синтаксис:

Ungroup Cells

Описание:

Команда Ungroup Cells преобразовывает текущую группу ячеек в последовательность ячеек ввода или ячеек автовызова. Если группа ячеек является ячейкой ввода, то она преобразовывается в ячейку ввода; если группа ячеек - ячейка автовызова, то она преобразовывается в ячейку автовызова. При этом ячейка вывода, ассоциированная с группой ячеек, удаляется.

Группа ячеек является текущей, если выполняются следующие условия:

- курсор находится в поле этой группы;
- курсор находится в конце строки, в которой расположен закрывающий маркер для группы ячеек;
- курсор находится в ячейке вывода, связанной с данной группой ячеек, если группа ячеек выделена.

Hide/Show Cell Markers — **Скрыть/показать маркеры ячейки**

Синтаксис:

Hide (Show) Cell Markers

Описание:

Команда Hide Cell Markers позволяет сделать невидимыми маркеры ячеек по всей М-книге. Когда команда выбрана, она изменяется на Show Cell Markers.

Эти команды никак не влияют на распечатку ячеек, поскольку Notebook вообще не печатает маркеры, независимо от того, видимы они на экране или нет.

Toggle Graph Output for Cell — **Запретить/разрешить вывод графика**

Синтаксис:

Toggle Graph Output for Cell

Описание:

Команда Toggle Graph Output for Cell позволяет запретить или разрешить выод графика для данной ячейки ввода.

Если ячейка ввода или автовызова создает рисунок, а вы желаете подавить его вывод, то следует поместить курсор в поле ячейки ввода и выбрать эту команду. Сообщение (no graph) будет помещено после маркера ячейки ввода, чтобы указать, что вывод графика для той ячейки подавлен. Чтобы разрешить вывод графика для этой ячейки, следует поместить курсор в поле ячейки ввода и выбрать эту же команду еще раз. Помета (no graph) будет удалена. Команда Toggle Graph Output for Cell отменяет опцию Embed Figures in M-book, если эта опция была установлена в диалоговой панели Notebook Options.

Evaluate Cell — **Вычислить ячейку**

Синтаксис:

Evaluate Cell

Описание:

Команда Evaluate Cell посылает текущую ячейку ввода или группу ячеек в среду системы MATLAB для обработки. Результаты вычисления или сообщения об ошибках размещаются в ячейках вывода. Ячейка ввода содержит

команду или оператор системы MATLAB. Группа ячеек - это одна многострочная ячейка ввода, которая содержит более одной команды или оператора системы MATLAB.

Когда вычисляется ячейка ввода, для которой не существует ячейки вывода, то ППП Notebook размещает такую ячейку сразу за ячейкой ввода; если ячейка вывода уже существует, то результаты размещаются в этой ячейке, где бы внутри М-книги она ни находилась. Если вычисляется группа ячеек, то все результаты размещаются в единственной ячейке вывода.

Ячейка ввода или группа ячеек являются текущими, если выполняются следующие условия:

- курсор находится в поле этих ячеек;
- курсор находится в конце строки, в которой расположен закрывающий маркер ячейки ввода или группы ячеек;
- курсор находится в ячейке вывода, связанной с данной ячейкой ввода или группой ячеек;
- если ячейка ввода или группа ячеек выделены.

Вычисление, которое включает длинную операцию, может вызвать ситуацию, называемую блокировкой таймера. Если это случается, Word выводит сообщение о блокировке таймера и спрашивает, будете ли вы продолжать вычисления или завершите их в данный момент. Если вы продолжаете вычисления, то Word сбрасывает блокировку и начинает отсчет допустимого интервала с начала. Допустимый интервал блокировки является внутренним параметром редактора Word и не может быть изменен.

Evaluate Calc Zone — **Вычислить зону**

Синтаксис:

Evaluate Calc Zone

Описание:

Команда Evaluate Calc Zone посылает ячейки ввода из текущей зоны вычисления в среду системы MATLAB для обработки. Текущая зона вычисления - это секция документа редактора Word, в которой находится курсор.

Для каждой ячейки ввода ППП Notebook генерирует ячейку вывода. Если ячейка вывода не существует, то она создается и размещается сразу за ячейкой ввода; если существует, то результаты размещаются в этой ячейке, где бы внутри М-книги она не находилась.

Evaluate M-book — **Вычислить М-книгу**

Синтаксис:

Evaluate M-book

Описание:

Команда Evaluate M-book вычисляет М-книгу целиком, посылая все ячейки ввода в среду системы MATLAB для обработки. Вычисление наяинается с начала книги, независимо от текущей позиции курсора.

Для каждой ячейки ввода ППП Notebook генерирует ячейку вывода. Если ячейка вывода не существует, то она создается и размещается сразу за ячейкой ввода; если существует, то результаты размещаются в этой ячейке, где бы внутри М-книги она не находилась.

Evaluate Loop **Вычислить ячейки ввода в цикле**

Синтаксис:

Evaluate Loop

Описание:

Команда Evaluate Loop позволяет организовать вычисление выделенных ячеек в цикле.

Для этого необходимо выполнить следующие шаги:

1. Выделить ячейки ввода, которые должны быть вычислены несколько раз.
2. Выбрать команду из меню Notebook, после чего на экране появится следующая диалоговая панель (рис. 11.10).

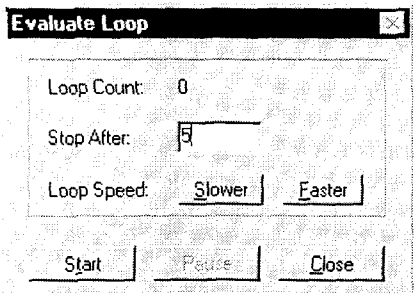


Рис. 11.10

Указать количество циклов, которое необходимо выполнить над выделенными ячейками в поле Stop After, затем нажать на клавишу Start. При этом метка на кнопке изменяется на Stop. ППП Notebook начинает исполнение команд и указывает количество завершенных итераций в поле Loop Count.

Для увеличения задержки в конце каждой итерации надо нажать кнопку Slower; для уменьшения - Faster.

Чтобы приостановить выполнение команд, следует нажать кнопку Pause. При этом метка на кнопке изменяется на Resume. Для продолжения вычислений - нажать кнопку Resume. Для прекращения обработки предназначена кнопка Stop.

Чтобы закрыть диалоговое окно Evaluate Loop, нажмите кнопку Close.

Bring MATLAB to Front

Вынести командное окно MATLAB на передний план

Синтаксис:

Bring MATLAB to Front

Описание:

Команда Bring MATLAB to Front предназначена для того, чтобы на экране терминала вынести командное окно MATLAB на передний план.

Notebook Options

Управление выводом на терминал результатов вычислений в М-книге

Синтаксис:

Notebook Options

Описание:

Команда Notebook Options позволяет проверить и изменить опции, связанные с выводом на экран числовых и графических результатов вычислений в М-книге. При выборе этой команды Notebook высвечивает на экране следующую диалоговую панель (рис. 11.11).

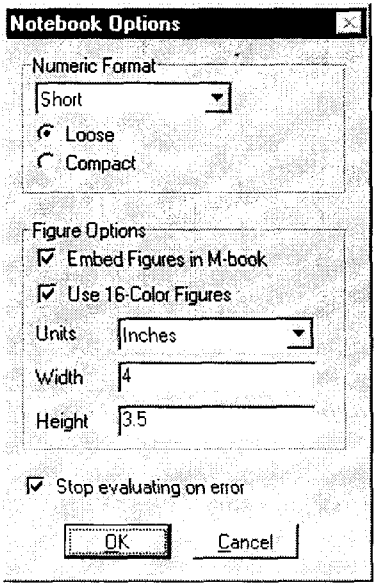


Рис. 11.11

Управление форматом вывода чисел. Установки Numeric Format управляют форматом числового вывода. Можно изменять способ представления числовых данных при их выводе на терминал, выбирая опцию формата из списка. Эти установки соответствуют опциям команды format системы MATLAB. Кроме того, можно управлять пробелами между ячейками ввода и ячейками вывода, выбирая форматы Loose или Compact.

Для подтверждения выбранных установок следует нажать кнопку OK. Эти установки не влияют на выведенные результаты, они действительны только тогда, когда вы вычисляете новые или перевычисляете прежние ячейки. Кнопка Cancel позволяет отказаться от только что выбранных установок.

Управление выводом графики. Следующие установки Figure Options позволяют управлять выводом графики:

- флажок Embed Figures in M-book управляет помещением рисунков в М-книгу. По умолчанию флажок вывода графики в М-книгу включен. Если флажок не отмечен, то Notebook выводит графику в отдельном графическом окне. Команда Toggle Graph Output for Cell игнорирует опцию Embed Figures in M-book, если она была установлена;
- флажок Use 16-Color Figures управляет выбором 16-цветной или 256-цветной палитры. Если на печать выводятся цветные многоугольники или поверхности, необходимо использовать 16-цветную палитру вместо 256-цветной, применяемой по умолчанию; 16-цветная палитра обеспечивает правильную печать на цветном или черно-белом с полутонами принтере; 256-цветная палитра выводит на печать в черном цвете независимо от устройства вывода;
- список Units и поля Width, Height управляют размером рисунка. Единица измерения устанавливается из списка Units, а точные размеры по ширине и высоте определяются полями Width и Height.

Контроль при вычислении множественных ячеек. Флажок Stop evaluating on error позволяет контролировать вычисление последовательно исполняемых ячеек при вычислении полной М-книги. Если эта опция отмечена, то при возникновении ошибки ни одна из последующих ячеек не вычисляется; если нет, то вычисления продолжаются независимо от того, возникают ошибки или нет.

Кнопки OK и Cancel. Для подтверждения выбранных установок используется кнопка OK. Эти установки действуют только на результаты, которые будут получены после нажатия кнопки. Они не влияют на ранее вычисленные результаты, если только соответствующие ячейки ввода не будут перевычислены. Кнопка Cancel служит для отказа от выбранных установок.

ИНДЕКСНЫЕ УКАЗАТЕЛИ

Команды, функции и операторы системы MATLAB 5.x

Команды, функции, операторы	*Назначение*
Команды общего назначения	
Справочные команды	
help	Текущая справка в командной строке
helpwin	Текущая справка в отдельном окне
helpdesk	Документация и диагностика в гипертексте
demo	Демонстрационные примеры
ver	Справка о текущей версии MATLAB
whatsnew	Вывод на экран файлов readme
readme	Новости о текущей версии MATLAB
Управление рабочей областью	
who	Список текущих переменных
whos	Список текущих переменных с подробностями
clear	Удаление переменных и функций из памяти
pack	Дефрагментация рабочей области памяти
load	Считывание переменных из MAT-файла
save	Запись переменных в MAT-файл
quit	Завершение работы в системе MATLAB
Управление командами и функциями	
what	Список файлов в текущем каталоге
type	Просмотр текста M-файла
edit	Редактирование текста M-файла
lookfor	Поиск M-файлов по ключу
which	Месторасположение функций и файлов
pcode	Создание P-файла псевдокода
inmem	Просмотр функций в рабочей памяти
mex	Компиляция MEX-функции
Управление путями доступа	
path	Определить/установить путь доступа
addpath	Добавить каталог к пути доступа
rmpath	Удалить каталог из пути доступа
editpath	Отредактировать путь доступа

Управление командным окном	
echo	Эхо-команда
more	Управление выводом страниц
diary	Ведение протокола сеанса работы
format	Управление форматом вывода
clc	Очистить командное окно
home	Поместить курсор в начальную позицию
pause	Прерывание выполнения
Команды операционной системы	
cd	Перейти в другой каталог
pwd	Путь доступа к текущему каталогу
dir	Содержимое текущего каталога
delete	Удалить файл
getenv	Получить значение переменной среды окружения
!	Выполнить команду операционной системы
dos	Выполнить команду DOS и возвратить результат
unix	Выполнить команду ОС UNIX и возвратить результат
vms	Выполнить команду DCL ОС VMS и возвратить результат
web	Подключиться к Web-серверу
computer	Определить тип используемого компьютера
Отладка М-файлов	
debug	Просмотреть список команд отладки
dbstop	Задать контрольную точку
dbclear	Удалить контрольную точку
dbcont	Продолжить выполнение
dbdown	Перейти в стеке вызываемых М-функций сверху вниз
dbstack	Вывести стек вызываемых М-функций
dbstatus	Просмотреть список контрольных точек
dbstep	Выполнить одну или несколько команд отладки
dbtype	Распечатать М-файл с пронумерованными строками
dbup	Перейти в стеке вызываемых М-функций снизу вверх
dbquit	Завершить отладку
Профилирование М-файлов	
profile	Профиль времени исполнения М-файла
pareto	Отчет о профиле
promsum	Диаграмма Парето

Операторы, специальные символы, переменные и константы

Арифметические операторы	
+ plus	Сложение
+ uplus	Унарное сложение
- minus	Вычитание

- uminus	Унарное вычитание
* mtimes	Умножение матриц
.* times	Поэлементное умножение для массивов
^ mpower	Возведение матрицы в степень
.^ power	Возведение в степень для массивов
\ mldivide	Левое деление матриц
/ mrdivide	Правое деление матриц
.\ ldivide	Левое деление для массивов
./ rdivide	Правое деление для массивов
kron	Тензорное произведение векторов

Операторы отношения

==	eq	Тождественно
~=	ne	Не тождественно
<	lt	Меньше
>	gt	Больше
<=	le	Меньше или равно
>=	ge	Больше или равно

Логические операторы

&	and	Логическое И
|	or	Логическое ИЛИ
~	not	Логическое НЕТ
xor		Логическое ИСКЛЮЧИТЕЛЬНОЕ ИЛИ
any		Истинно, если хотя бы 1 элемент вектора не равен нулю
all		Истинно, если все элементы вектора не равны нулю

Специальные символы

:	Сечение массива
( )	Указание последовательности выполнения операций
[ ]	Формирование массива
{ }	Многомерные массивы
.	Десятичная точка (разделитель)
.	Выделение поля структуры
..	Указатель на каталог-родитель
...	Продолжение строки
,	Разделитель
;	Подавление вывода эхо-результата
%	Комментарий
!	Вызов команды операционной системы
=	Присваивание
'	Кавычка
.' transpose	Транспонирование элементов массива
' ctranspose	Транспонирование элементов матрицы

[,] horzcat	Объединение элементов в строку
[;] vertcat	Объединение элементов в столбец
( ), { },. subsasgn	Присваивание подмассива
( ), { },. subsref	Ссылка на подмассив
subsindex	Индекс подмассива

Операторы поразрядной обработки

bitand	Поразрядное И
bitcmp	Биты дополнения
bitor	Поразрядное ИЛИ
bitmax	Максимальное число разрядов
bitxor	Поразрядное ИСКЛЮЧИТЕЛЬНОЕ ИЛИ
bitset	Задать бит
bitget	Узнать бит
bitshift	Поразрядный сдвиг

Операторы обработки множеств

union	Объединение множеств
unique	Выделение множества
intersect	Пересечение множеств
setdiff	Разность множеств
setxor	ИСКЛЮЧИТЕЛЬНОЕ ИЛИ для множеств
ismember	Истинно, если это элемент множества

Специальные переменные и константы

ans	Результат выполнения последней операции
eps	Машинная точность
realmax	Наибольшее число с плавающей точкой
realmin	Наименьшее число с плавающей точкой
pi	π = 3.141592653589793e+000
i, j	Мнимая единица, $\sqrt{-1}$
inf	Бесконечное значение, ∞
NaN	Нечисловое значение
isnan	Истинно, если нечисловое значение
isinf	Истинно, если бесконечное значение
isfinite	Истинно, если конечное значение
flops	Количество операций с плавающей точкой

Функции вычисления времени и дат

Текущее время и дата

clock	Текущее время и дата в форме вектора
date	Текущая дата в форме строки
now	Текущее время и дата в форме числа

Основные функции	
datenum	Последовательный номер даты с 01-Jan-0000
datestr	Строковое представление даты
datevec	Векторное представление даты
calendar	Календарь текущего месяца
weekday	День недели
eomday	Последний день месяца
datetick	Форматирование меток осей датой
cputime	Время работы центрального процессора в секундах
tic	Начало отсчета
toc	Конец отсчета
etime	Интервал использованного времени

Типы данных

Многомерные массивы	
cat	Объединить массивы
ndims	Размерность массива
ndgrid	Сгенерировать сетку для многомерной функции
permute	Перестановка размерностей массива
ipermute	Обратная перестановка размерностей массива
shiftdim	Изменить размерность массива
squeeze	Удалить одну из размерностей
Массивы ячеек	
cell	Создать массив ячеек
celldisp	Показать содержимое массива ячеек
cellplot	Показать графическую структуру массива ячеек
deal	Установить соответствие входов с выходами
iscell	Истинно, если это массив ячеек
cell2struct	Преобразовать массив ячеек в массив структур
num2cell	Преобразовать числовой массив в массив ячеек
struct2cell	Преобразовать массив структур в массив ячеек
Массивы записей	
struct	Создать массив записей
fieldnames	Получить имена полей
getfield	Получить содержимое полей
setfield	Установить содержимое полей
rmfield	Удалить поле
isfield	Истинно, если это поле массива записей
isstruct	Истинно, если это массив записей

Язык программирования

Массивы, матрицы и операции над ними

Массивы и матрицы специального вида

zeros	Формирование массива нулей
ones	Формирование массива единиц
eye	Формирование единичной матрицы
repmat	Формирование многомерного массива на основе данного
rand	Формирование массива элементов, распределенных по равномерному закону
randn	Формирование массива элементов, распределенных по нормальному закону
linspace	Формирование линейного массива равноотстоящих узлов
logspace	Формирование узлов логарифмической сетки
meshgrid	Формирование узлов двумерной и трехмерной сеток
:	Формирование векторов и подматриц

Характеристики массивов

size	Размер массива
length	Длина вектора
ndims	Количество размерностей массива
isempty	Истинно, если массив пустой
isequal	Истинно, если два массива идентичны
isnumeric	Истинно, если массив числовой
islogical	Истинно, если массив логический
logical	Преобразовать числовые элементы в логические

Операции над массивами и матрицами

reshape	Преобразование размеров матрицы
diag	Формирование или извлечение диагоналей матрицы
tril	Формирование нижней треугольной матрицы
triu	Формирование верхней треугольной матрицы
fliplr	Отражение матрицы относительно вертикальной оси
flipud	Отражение матрицы относительно горизонтальной оси
flipdim	Отражение многомерного массива относительно указанной размерности
rot90	Поворот матрицы на 90°
find	Определить индексы ненулевых элементов
end	Последний индекс многомерной матрицы
sub2ind	Преобразование многомерной нумерации в последовательную
ind2sub	Преобразование последовательной нумерации в многомерную

Математические функции

Тригонометрические функции	
sin	Синус
sinh	Гиперболический синус
asin	Арксинус
asinh	Гиперболический арксинус
cos	Косинус
cosh	Гиперболический косинус
acos	Арккосинус
acosh	Гиперболический арккосинус
tan	Тангенс
tanh	Гиперболический тангенс
atan	Арктангенс
atan2	Арктангенс от двух аргументов
atanh	Гиперболический арктангенс
sec	Секанс
sech	Гиперболический секанс
asec	Арксеканс
asech	Гиперболический арксеканс
csc	Косеканс
csch	Гиперболический косеканс
acsc	Арккосеканс
acsch	Гиперболический арккосеканс
cot	Котангенс
coth	Гиперболический котангенс
acot	Арккотангенс
acoth	Гиперболический арккотангенс

Трансцендентные функции	
exp	Экспоненциальная функция
log	Функция натурального логарифма
log10	Логарифм по основанию 10
log2	Логарифм по основанию 2
pow2	Экспонента по основанию 2
sqrt	Функция квадратного корня
nextpow2	Ближайшая степень экспоненты по основанию 2

Функции обработки комплексных чисел	
abs	Абсолютное значение комплексного числа
angle	Аргумент комплексного числа
conj	Комплексно-сопряженное число
imag	Мнимая часть комплексного числа
real	Действительная часть комплексного числа
unwrap	Непрерывная функция фазового угла
isreal	Истинно, если это массив действительных чисел
cplxpair	Сортировка комплексно-сопряженных пар

Округление и модульная арифметика

fix	Усечение дробной части числа
floor	Округление до меньшего целого
ceil	Округление до большего целого
round	Округление до ближайшего целого
mod	Остаток в смысле модульной арифметики
rem	Остаток от деления с учетом знака
sign	Знак числа

Специальные математические функции

airy	Функция Эйри
besselj	Функция Бесселя первого рода
bessely	Функция Бесселя второго рода
besselh	Функция Бесселя третьего рода (функция Ганкеля)
besseli	Модифицированная функция Бесселя первого рода
besselk	Модифицированная функция Бесселя второго рода
beta	Полная бета-функция
betainc	Неполная бета-функция
betaln	Натуральный логарифм полной бета-функции
ellipj	Эллиптические функции Якоби
ellipke	Полные эллиптические интегралы
erf	Функция ошибки
erfc	Остаточная функция ошибки
erfcx	Масштабированная остаточная функция ошибки
erfinv	Обратная функция ошибки
expint	Интегральная показательная функция
gamma	Полная гамма-функция
gammainc	Неполная гамма-функция
gammaln	Натуральный логарифм полной гамма-функции
legendre	Функция Лежандра
cross	Векторное произведение векторов

Теорстико-числовые функции

factor	Разложение числа на простые множители
isprime	Истинно, если число простое
primes	Формирование списка простых чисел
gcd	Наибольший общий делитель
lcm	Наименьшее общее кратное
rat	Приближение числа в виде рациональной дроби
rats	Вычисления в поле рациональных чисел
perms	Формирование всех перестановок элементов вектора
nchoosek	Вычисление числа сочетаний, C_n^k

Функции преобразования систем координат	
cart2sph	Преобразование декартовой системы в сферическую
cart2pol	Преобразование декартовой системы в полярную
pol2cart	Преобразование полярной системы в декартову
sph2cart	Преобразование сферической системы в декартову
hsv2rgb	Преобразование hsv-палитры в rgb-палитру
rgb2hsv	Преобразование rgb-палитры в hsv-палитру

Линейная алгебра

Коллекция матриц	
compan	Сопровождающая матрица
gallery	Пакет тестовых матриц
hadamard	Матрица Адамара
hankel	Матрица Ганкеля
hilb	Матрица Гильберта
invhilb	Матрица, обратная матрице Гильберта
magic	Магический квадрат
pascal	Матрица Паскаля
rosser	Матрица Рессера
toeplitz	Матрица Теплица
vander	Матрица Вандермонда
wilkinson	Матрица Уилкинсона

Матричный анализ	
norm	Вычисление норм векторов и матриц
rank	Вычисление ранга матрицы
det	Вычисление определителя матрицы
trace	Вычисление следа матрицы
null	Вычисление нуль-пространства матрицы
orth	Вычисление ортонормального базиса матрицы
rref	Приведение к треугольной форме
subspace	Вычисление угла между подпространствами

Решение систем линейных уравнений	
\ и /	Решатели систем уравнений
inv	Вычисление обратной матрицы
cond	Вычисление числа обусловленности по отношению к задаче обращения матрицы
chol	Разложение Холецкого
cholinc	Неполное разложение Холецкого
lu	LU-разложение
luinc	Неполное LU-разложение
qr	QR-разложение
nnls	Метод наименьших квадратов с ограничениями
pinv	Псевдообращение по Муру - Пенроузу
lscov	Метод наименьших квадратов в присутствии шумов

Собственные значения и сингулярные числа	
eig	Полная проблема собственных значений
svd	Сингулярное разложение
eigs	Вычисление отдельных собственных значений
svds	Вычисление отдельных сингулярных чисел
poly	Вычисление характеристического полинома
polyeig	Решение полиномиальных матричных уравнений
condeig	Вычисление числа обусловленности по отношению к задаче на собственные значения
hess	Приведение матрицы к форме Хессенберга
qz	Обобщенная проблема собственных значений
schur	Приведение матрицы к форме Шура

Вычисление функций от матриц	
expm	Вычисление матричной экспоненты
logm	Вычисление логарифма матрицы
sqrtm	Вычисление функции $A^{1/2}$
funm	Вычисление произвольных функций от матриц

Утилиты	
qrdelete	Удалить столбец из QR-разложения
qrinsert	Добавить столбец в QR-разложение
rsf2csf	Преобразование действительной формы Шура в комплексную
cdf2rdf	Преобразование комплексной формы Шура в действительную
balance	Масштабирование матрицы
planerot	Формирование матрицы вращения Гивенса

Полиномы и операции над ними	
polyval	Вычисление полинома
polyvalm	Вычисление матричного полинома
poly	Вычисление характеристического полинома
residue	Разложение на простые дроби
roots	Вычисление корней полинома
polyfit	Аппроксимация данных полиномом
polyder	Вычисление производной полинома
conv	Умножение полиномов
deconv	Деление полиномов

Анализ данных и преобразование Фурье

Базовые операции	
max	Максимальный компонент массива
min	Минимальный компонент массива
mean	Компонент средних значений массива
median	Компонент срединных значений массива
std	Компонент стандартных отклонений массива

sort	Сортировка по возрастанию
sortrows	Сортировка строк по возрастанию
sum	Суммирование элементов массива
prod	Произведение элементов массива
cumsum	Суммирование с накоплением
cumprod	Произведение с накоплением

Численное интегрирование

cumtrapz	Численное интегрирование методом трапеций с накоплением
trapz	Численное интегрирование методом трапеций
quad	Численное интегрирование методом квадратур
quad8	Численное интегрирование методом Ньютона - Котеса
dblquad	Вычисление двойного интеграла

Вычисление минимумов и нулей функций

fmin	Минимизация функции одной переменной
fmins	Минимизация функции нескольких переменных
fzero	Нахождение нулей функции одной переменной

Аппроксимация и интерполяция данных

interp1	Одномерная табличная интерполяция
interp1q	Быстрая одномерная интерполяция
interp2	Двумерная табличная интерполяция
interp3	Трехмерная табличная интерполяция
interpn	N-мерная табличная интерполяция
interpft	Аппроксимация периодической функции
griddata	Интерполяция на неравномерной сетке
ppval	Аппроксимация кусочно-гладкими полиномами
spline	Интерполяция кубическим сплайном

Геометрический анализ данных

delaunay	Триангуляция Делоне
dsearch	Триангуляция Делоне для ближайшей точки
tsearch	Поиск наилучшей триангуляции
convhull	Вычисление выпуклой оболочки
voronoi	Вычисление диаграммы Вороного
inpolygon	Истинно, если точка внутри полигона
rectint	Область пересечения прямоугольника
polyarea	Область многоугольника

Вычисление конечных разностей

diff	Аппроксимация производных конечными разностями
gradient	Вычисление градиента функции
del2	Аппроксимация Лапласиана

Корреляционный анализ

corrcoef	Вычисление коэффициентов корреляции
cov	Вычисление матрицы ковариаций

Преобразования Фурье

fft	Одномерное дискретное преобразование Фурье
fft2	Двумерное дискретное преобразование Фурье
fftn	N-мерное дискретное преобразование Фурье
ifft	Обратное одномерное преобразование Фурье
ifft2	Обратное двумерное преобразование Фурье
ifftn	Обратное N-мерное преобразование Фурье
fftshift	Сдвиг постоянной составляющей в центр спектра

Свертка и фильтрация

filter	Дискретная одномерная фильтрация
filter2	Дискретная двумерная фильтрация
conv	Свертка одномерных массивов
conv2	Свертка двумерных массивов
convn	Свертка N-мерных массивов
deconv	Операция, обратная свертке (деление полиномов)

Звуковое воспроизведение

sound	Озвучить одномерный массив чисел
soundsc	Масштабировать и озвучить одномерный массив чисел
mu2lin	Преобразование μ-кодированного сигнала в линейный
lin2mu	Преобразование линейного сигнала в μ-кодированный

Решение обыкновенных дифференциальных уравнений (ОДУ)

Решатели ОДУ

ode113	Нежесткие ОДУ, метод переменных состояний
ode15s	Жесткие ОДУ, метод переменных состояний
ode23	Нежесткие ОДУ, метод низкого порядка
ode23s	Жесткие ОДУ, метод Рунге - Кутты 3-го порядка
ode45	Нежесткие ОДУ, метод Рунге - Кутты 4-го порядка
odefile	Описание системы ОДУ

Дескрипторная поддержка опций решателя

odeset	Создать/изменить опции решателя
odeget	Получить опции решателя

Формирование выходов решателя

odeplot	Формирование процессов как функций времени
odephas2	Двумерная фазовая плоскость
odephas3	Трехмерная фазовая плоскость
odeprint	Командное окно вывода на печать

Работа с разреженными матрицами

Элементарные разреженные матрицы

sparse	Формирование разреженной матрицы
speye	Единичная разреженная матрица

sprand	Случайная разреженная матрица с элементами, распределенными по равномерному закону
sprandn	Случайная разреженная матрица с элементами, распределенными по нормальному закону
sprandsym	Случайная разреженная симметрическая матрица
spdiags	Формирование диагоналей разреженной матрицы

Характеристики разреженных матриц

normest	Оценка 2-нормы разреженной матрицы
condest	Оценка числа обусловленности по 1-норме
sprank	Вычисление структурного ранга

Преобразование разреженных матриц

full	Преобразование разреженной матрицы в полную
find	Определение индексов ненулевых элементов
spconvert	Восстановление разреженной матрицы из внешнего ASCII-формата

Работа с ненулевыми элементами

nnz	Количество ненулевых элементов
nonzeros	Формирование вектора ненулевых элементов
nzmax	Количество ячеек памяти для размещения ненулевых элементов
spones	Формирование матрицы связности
spalloc	Выделить память для разреженной матрицы
issparse	Истинно, если матрица разреженная
spfun	Вычисление функции только для ненулевых элементов

Операции над графом разреженной матрицы

etree	Вычисление дерева структуры
etreeplot	Построение дерева структуры
treelayout	Разметка дерева структуры
treeplot	Построение дерева структуры

Алгоритмы упорядочения

colmmd	Упорядочение по разреженности столбцов
symmmd	Симметрическая упорядоченность
symrcm	RCM-упорядоченность
colperm	Упорядочение столбцов с учетом их разреженности
randperm	Формирование случайных перестановок
dmperm	DM-декомпозиция разреженной матрицы

Решение систем уравнений с разреженными матрицами

pcg	Метод сопряженных градиентов
bicg	Двунаправленный метод сопряженных градиентов
bicgstab	Устойчивый двунаправленный метод
cgs	Квадратичный метод сопряженных градиентов
gmres	Метод минимизации обобщенной невязки
qmr	Квазиминимизация невязки

Визуализация разреженных матриц

gplot	Построение графа структуры
spy	Визуализация структуры разреженной матрицы

Вспомогательные операции

spparms	Установка параметров для алгоритмов обработки
symbfact	Характеристики разложения Холецкого
spaugment	Формирование расширенной матрицы для метода наименьших квадратов

Элементарная графика

Двумерные графики

lot	График в линейном масштабе
loglog	График в логарифмическом масштабе
semilogx	График в полулогарифмическом масштабе по оси x
semilogy	График в полулогарифмическом масштабе по оси y
polar	График в полярных координатах
plotyy	График с двумя вертикальными осями

Трехмерные графики

plot3	Построение линий и точек в трехмерном пространстве
contour	Изображение линий уровня для трехмерной поверхности
contourc	Формирование массива описания линий уровней
contour3	Изображение трехмерных линий уровня
meshgrid	Формирование двумерных массивов X и Y
mesh	Трехмерная сетчатая поверхность
meshc	Трехмерная сетчатая поверхность с проекцией линий постоянного уровня
meshz	Трехмерная сетчатая поверхность с плоскостью отсчета на нулевом уровне
surf	Затененная сетчатая поверхность
surfc	Затененная сетчатая поверхность с проекцией линий постоянного уровня
surfl	Затененная сетчатая поверхность с подсветкой

Задание осей координат

axis	Масштабирование и вывод осей координат
grid	Управление выводом сетки
hold	Управление режимом сохранения графического окна
subplot	Разбиение графического окна
zoom	Изменение масштаба в графическом окне

Управление цветом

caxis	Установление соответствия между палитрой цветов и масштабированием осей
colormap	Палитра цветов

colstyle	Выделить цвет и стиль для графика из заданного массива
pcolor	Палитра псевдоцветов
rgbplot	Изображение палитры
spinmap	Вращение палитры
hsv2rgb	Преобразование hsv-палитры в rgb-палитру
rgb2hsv	Преобразование rgb-палитры в hsv-палитру
shading	Затенение поверхностей
brighten	Управление яркостью
contrast	Палитра серого с повышенной контрастностью
hidden	Управление удалением невидимых линий
whitebg	Управление цветом фона

Палитры цветов

hsv	Палитра радуги
hot	Палитра с чередованием черного, красного, желтого и белого
gray	Линейная палитра в оттенках серого
bone	Серая палитра с оттенком синего
copper	Линейная палитра в оттенках меди
pink	Розовая палитра с оттенками пастели
white	Палитра белого
flag	Палитра с чередованием красного, белого, синего и черного
lines	Палитра, определяемая свойством ColorOrder
colorcube	RGB-палитра с оттенками серого
jet	Разновидность hsv-палитры
prism	Палитра с чередованием красного, оранжевого, желтого, зеленого, синего и фиолетового
cool	Палитра с оттенками голубого и фиолетового
autumn	Палитра с оттенками красного и желтого
spring	Палитра с оттенками желтого и фиолетового
winter	Палитра с оттенками голубого и зеленого
summer	Палитра с оттенками желтого и зеленого

Управление подсветкой

diffuse	Эффект диффузного рассеяния
lighting	Управление подсветкой
material	Эффект рассеяния материала поверхности
specular	Эффект зеркального отражения
surfnorm	Построение нормалей к поверхности

Управление углом просмотра

view	Управление положением точки просмотра
viewmtx	Вычисление матрицы управления углом просмотра
rotate3d	Интерактивные повороты трехмерного объекта

Надписи и пояснения к графикам	
xlabel	Обозначение оси x
ylabel	Обозначение оси y
zlabel	Обозначение оси z
clabel	Маркировка линий уровня
colorbar	Шкала палитры
title	Заголовок графика
text	Добавление к текущему графику текста
gtext	Размещение текста на графике с помощью мыши
legend	Пояснение к графику
Создание твердой копии и сохранение графика	
orient	Размещение твердой копии на странице
print	Вывод графика на печать или в файл
printopt	Задание опций печати по умолчанию

Специальная графика

Двумерные графики	
area	Закраска областей графика
bar	Столбцовая диаграмма
barh	Столбцовая диаграмма с горизонтальным расположением
comet	Движение точки по траектории
compass	График векторов-стрелок, исходящих из начала координат
errorbar	График с указанием интервала погрешности
ezplot	Построение графиков с использованием диалогового окна
feather	График векторов-стрелок, исходящих из равноотстоящих точек горизонтальной оси
fill	Закраска многоугольников
hist	Построение гистограммы
pareto	График результатов профилирования программы
pie	Круговая диаграмма
plotmatrix	График матрицы
quiver	График поля направлений
ribbon	Изображение линий на трехмерном графике
stairs	Ступенчатый график
stem	График дискретных значений
Трехмерная графика	
bar3	Трехмерная столбцовая диаграмма
bar3h	Трехмерная столбцовая диаграмма с горизонтальным расположением
comet3	Движение точки по траектории в трехмерном пространстве
contourf	График линий уровня с раскрашенными областями
fill3	Раскраска многоугольников в трехмерном пространстве
pie3	Секторная диаграмма

quiver3	График поля направлений в трехмерном пространстве
slice	Сечения функции от трех переменных
stem3	График дискретных значений в трехмерном пространстве
trimesh	Трехмерная поверхность с треугольными ячейками
trisurf	Трехмерная сетчатая поверхность с треугольными ячейками
waterfall	Трехмерная поверхность без прорисовки ребер сетки

Работа с графическими образами

image	Вывод графического образа
imagesc	Масштабирование и вывод графического образа
imfinfo	Информация о структуре графического файла

Анимационные возможности

capture	Захват графической фигуры
getframe	Создать фрейм для анимации
moviein	Выделить память под фреймы анимации
movie	Выполнить анимацию
rotate	Вращение графического объекта
frame2im	Преобразование фрейма в графический образ
im2frame	Преобразование графического образа в фрейм

Объемные графические объекты

patch	Закрашенный многоугольник
cylinder	Выполнить расчет цилиндра
sphere	Выполнить расчет сферы

Дескрипторная графика

Создание и управление графическим окном

figure	Открыть графическое окно (команда)
gcf	Получить дескриптор графического объекта figure
clf	Очистить графическое окно
shg	Показать графическое окно (для совместимости с версией 3.5)
close	Закрыть графическое окно
refresh	Обновить графическое окно

Создание и управление осями координат

axes	Создать оси координат (команда)
box	Окружить оси прямоугольником или параллелепипедом
cla	Очистить оси координат
gca	Получить дескриптор графического объекта axes
hold	Сохранить оси координат
ishold	Истинно, если оси координат сохранены

Объекты дескрипторной графики

figure	Графический объект figure
axes	Графический объект axes
line	Графический объект line
text	Графический объект text

patch	Графический объект patch
surface	Графический объект surface
image	Графический объект image
light	Графический объект light
uicontrol	Графический объект uicontrol
uimenu	Графический объект uimenu

Операции над графическими объектами

set	Установить свойства графического объекта
get	Получить свойства графического объекта
reset	Восстановить штатные значения свойств
delete	Удалить графический объект
gco	Получить дескриптор текущего объекта
gcbo	Получить дескриптор повторно вызываемого объекта
gcbf	Получить дескриптор повторно вызываемого графического окна
drawnow	Выполнить очередь задержанных графических команд
findobj	Найти объекты с заданными свойствами
copyobj	Скопировать сам объект и порожденные им графические объекты

Утилиты

closereq	Запрос на закрытие графического окна
ishandle	Истинно, если это дескриптор
newplot	Восстановление штатных значений свойства NextPlot

Графический интерфейс пользователя (GUI)

Функции GUI

uicontrol	Создать управляющий элемент (команда)
uimenu	Создать меню (команда)
ginput	Съем координат с помощью мыши
dragrect	Переместить прямоугольник с помощью мыши
rbbox	Растянуть прямоугольник с помощью мыши
selectmoveresize	Выбор, перемещение, изменение размеров, копирование объектов с помощью мыши
waitforbuttonpress	Ожидание нажатия клавиши клавиатуры или мыши в поле графического окна
waitfor	Прекратить выполнение в ожидании события
uiwait	Прекратить выполнение в ожидании возобновления
uiresume	Возобновить выполнение после блокировки

Средства проектирования GUI

guide	Редактирование управляющих элементов в графическом окне
align	Выравнять положение объектов
cbedit	Изменить повторный вызов объекта
menuedit	Изменить меню графического окна
propedit	Изменить свойства объекта

Диалоговые панели	
dialog	Создание панели сообщений
dialog	Создание графического окна диалога
axlimdlg	Ограничение размеров окна диалога
errordlg	Диалоговая панель сообщений об ошибках
helpdlg	Диалоговая панель подсказки
inputdlg	Диалоговая панель ввода
listdlg	Диалоговая панель просмотра списка
menu	Меню диалогового ввода
msgbox	Создание панели сообщений
questdlg	Диалоговая панель с вопросом
warndlg	Диалоговая панель предупреждения
uigetfile	Стандартная диалоговая панель открытия файла
uiputfile	Стандартная диалоговая панель записи файла
uisetcolor	Стандартная диалоговая панель выбора цвета
uisetfont	Стандартная диалоговая панель выбора шрифта
pagedlg	Диалоговая панель расположения страницы
printdlg	Диалоговая панель печати
waitbar	Панель ожидания

Создание меню	
makemenu	Создать структуру меню
menubar	Установить штатные значения свойства MenuBar
umtoggle	Изменить статус checked объекта uimenu
winmenu	Создать подменю для пункта меню Window

Создание кнопок инструментальной панели	
btngroup	Создать кнопку на инструментальной панели
btnstate	Запросить состояние кнопки
btnpress	Управление кнопками инструментальной панели
btndown	Нажать кнопку
btnup	Отпустить кнопку

Утилиты задания свойств объектов figure *и* axes	
setuprop	Задать свойство
getuprop	Запросить значение свойства
cluprop	Удалить свойство

Вспомогательные утилиты	
allchild	Запросить все порожденные объекты
hidegui	Скрыть-раскрыть GUI
edtext	Интерактивное редактирование объекта text
getstatus	Запросить свойства строки объекта figure
setstatus	Установить свойства строки объекта figure
popupstr	Запросить свойства строки выпадающего меню

remapfig	Изменить расположение объекта figure
setptr	Установить указатель на объект figure
getptr	Запросить указатель на объект figure
overobj	Запросить дескриптор объекта по его указателю

Обработка строк

Основные функции

blanks	Сформировать строку пробелов
cellstr	Преобразовать массив символов в массив ячеек для строк
char	Сформировать массив символов
deblank	Удалить пробелы в конце строки
double	Преобразовать символы строки в числовые коды

Проверка строк

ischar	Истинно, если это массив символов (строка)
iscellstr	Истинно, если это массив ячеек для строк
isletter	Истинно, если это символ алфавита
isspace	Истинно, если это пробел

Операции над строками

strcat	Горизонтальное объединение строк
strvcat	Вертикальное объединение строк
strcmp	Сравнить строки
strncmp	Сравнить n символов строк
findstr	Найти заданную строку в составе другой строки
strjust	Выравнять массив символов
strmatch	Найти все совпадения
strrep	Заменить одну строку другой
strtok	Найти часть строки, ограниченную разделителями
upper	Перевести все символы строки в верхний регистр
lower	Перевести все символы строки в нижний регистр

Преобразования строк

num2str	Преобразование числа в строку
int2str	Преобразование целого в строку
mat2str	Преобразование матрицы в строку
str2mat	Объединение строк в матрицу
str2num	Преобразование строки в арифметическое выражение и его вычисление
sprintf	Записать форматированные данные в виде строки
sscanf	Прочитать строку с учетом формата

Преобразования систем счисления

hex2num	Преобразовать шестнадцатеричное число в число удвоенной точности
hex2dec	Преобразовать шестнадцатеричное число в десятичное число
dec2hex	Преобразовать десятичное число в шестнадцатеричную число

bin2dec	Преобразовать двоичную строку в десятичное число
dec2bin	Преобразовать десятичное число в двоичную строку
base2dec	Преобразовать B-строку в десятичное число
dec2base	Преобразовать десятичное число в B-строку

Операции ввода-вывода файлов

Открытие и закрытие файлов	
fopen	Открыть файл
fclose	Закрыть файл
Двоичные файлы	
fread	Прочитать двоичные данные из файла
fwrite	Записать двоичные данные в файл
Форматированные файлы	
fscanf	Прочитать форматированные данные из файла
fprintf	Записать форматированные данные в файл
fgetl	Прочитать строку файла, удалив символ конца строки
fgets	Прочитать строку файла, сохранив символ конца строки
input	Интерактивный ввод
Позиционирование файла	
ferror	Запросить информацию об ошибке ввода-вывода
feof	Проверить признак конца файла
fseek	Установить указатель в заданную позицию
ftell	Запросить позицию указателя в файле
frewind	Установить указатель в начало файла
Работа с каталогами	
matlabroot	Имя каталога, где размещена система MATLAB
matlabrc	Список путей доступа
path	Управление списком путей доступа
addpath	Добавить путь доступа к списку
editpath	Отредактировать список путей доступа
filesep	Разделитель каталогов для данной платформы
pathsep	Разделитель путей доступа для данной платформы
mexext	Расширение MEX-файлов для данной платформы
fullfile	Построить полное имя файла из частей
partialpath	Разбить путь доступа на части
tempdir	Запросить имя временного каталога
tempname	Запросить имя временного файла
Импорт-экспорт файлов	
load	Прочитать переменные из MAT-файла
save	Записать переменные в MAT-файл
csvread	Преобразовать файл, элементы которого разделены запятыми, в массив

csvwrite	Преобразовать массив в файл, элементы которого разделены запятыми
dlmwrite	Преобразовать массив в файл с ASCII-разделителем
dlmread	Преобразовать файл с ASCII-разделителем в массив
wk1read	Прочитать файл электронной таблицы Lotus123
wk1write	Записать файл в электронную таблицу Lotus123

Импорт-экспорт графических образов

imread	Считать графический образ из файла
imwrite	Записать графический образ в файл

Импорт-экспорт звуковых файлов

wavwrite	Записать звуковой файл .wav
wavread	Считать звуковой файл .wav

Классы объектов и программирование

Классы объектов

cell	Создать массив ячеек
char	Создать массив символов
double	Преобразовать в массив чисел удвоенной точности
sparse	Создать разреженную матрицу
struct	Создать массив записей
inline	Создать объект inline
uint8	Преобразовать в 8-битовое целое без знака

Объектно-ориентированное программирование

class	Создать объект или возвратить класс объекта
methods	Показать методы данного класса
isa	Истинно, если объект принадлежит данному классу
isobject	Истинно, если это объект
inferiorto	Отношение низшего класса
superiorto	Отношение высшего класса

Переопределение методов

minus	Переопределить метод для a - b
plus	Переопределить метод для a + b
times	Переопределить метод для a. * b
mtimes	Переопределить метод для a * b
mldivide	Переопределить метод для a\b
mrdivide	Переопределить метод для a/b
rdivide	Переопределить метод для a./b
ldivide	Переопределить метод для a.\b
power	Переопределить метод для a.^b
mpower	Переопределить метод для a^b
uminus	Переопределить метод для -a
uplus	Переопределить метод для +a
horzcat	Переопределить метод для [a b]

vertcat	Переопределить метод для [a; b]
le	Переопределить метод для a <= b
lt	Переопределить метод для a < b
gt	Переопределить метод для a > b
ge	Переопределить метод для a >= b
eq	Переопределить метод для a==b
ne	Переопределить метод для a~=b
not	Переопределить метод для ~a
and	Переопределить метод для a&b
or	Переопределить метод для a|b
subsasgn	Переопределить метод для a(i)=b, a{i}=b, a.field=b
subsref	Переопределить метод для a(i), a{i}, a.field
colon	Переопределить метод для a:b
transpose	Переопределить метод для a.'
ctranspose	Переопределить метод для a'
subsindex	Переопределить метод для x(a)

Интерфейсы DDE и ActiveX

Интерфейс DDE

ddeadv	Установить консультативную связь
ddeexec	Послать строку на выполнение
ddeinit	Инициировать DDE-диалог
ddepoke	Послать данные в приложение
ddereq	Запросить данные от приложения
ddeterm	Завершить DDE-диалог
ddeunadv	Завершить консультативную связь

Интерфейс ActiveX

actxcontrol	Создать элемент управления
actxserver	Создать локальный или удаленный сервер
activex	Конструктор интерфейса ActiveX
get	Получить свойства интерфейса
set	Установить свойства интерфейса
invoke	Вызвать метод для объекта ActiveX
release	Выполнить интерфейс
delete	Удалить интерфейс

Команды ППП Notebook

Команды	*Назначение*
Define Input Cell	Создать ячейку ввода
Define AutoInit Cell	Создать ячейку автостарта
Define Calc Zone	Создать зону вычисления
Undefine Cells	Преобразовать ячейки в текст
Purge Output Cells	Удалить ячейки вывода
Group Cells	Создать многострочную ячейку ввода
Ungroup Cells	Преобразовать группу ячеек в ячейки ввода
Hide/Show Cell Markers	Скрыть/показать маркеры ячейки
Toggle Graph Output for Cell	Запретить/разрешить вывод графики
Evaluate Cell	Вычислить ячейку
Evaluate Calc Zone	Вычислить зону
Evaluate M-book	Вычислить М-книгу
Evaluate Loop	Вычислить ячейки ввода в цикле
Bring MATLAB to Front	Вынести командное окно MATLAB на передний план
Notebook Options	Управление выводом на терминал результатов вычислений в М-книге
Numeric Format:	Управление форматом вывода чисел:
Short	короткое число
Long	длинное число
Hex	шестнадцатеричное число
Bank	коммерческий формат
Plus	символьный формат
Short e	короткое число (экспоненциальная форма)
Long e	длинное число (экспоненциальная форма)
Rational	формат рационального числа
Compact	подавление пробелов между строками
Loose	восстановление пробелов между строками
Figure Options:	Управление выводом графики:
Embed Figures in M-book	помещать рисунок в М-книгу
Use 16-Color Figures	выбор 16- или 256-цветной палитры
Units:	Единица измерения:
Inches	дюйм
Centimeters	сантиметр
Points	пункт
Stop evaluating on error	Контроль при вычислении множественных ячеек

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ