

3D-ГРАФИКИ В MATLAB

Функция meshgrid.....	1
Syntax	1
Описание	2
Примечания	2
Функции mesh, meshc, meshz.....	3
Syntax:	3
Описание:	3
Примеры	4
% Пример mesh	4
% Пример meshc	4
% Пример meshz	5
Функция colormap.....	5
Syntax	5
Палитры (map) для формата colormap(map):	6
Описание:	6
Примеры для формата colormap(map):	7
Функция shading	7
Syntax	7
Описание	7
Примеры	7
Функция view	7
Syntax	7
Описание	8
Примеры	8
Функция surf	8
Syntax	8
Описание	8
Примеры:	9
% Пример 1	9
% Пример 2	9
% Пример 3	10
% Пример 4	10
% Пример 5	11
% Пример 6	12
Функции delaunay trimesh trisurf	13
% Пример 1	13
% Пример 2	13

ФУНКЦИЯ MESHGRID

Генерирует матрицы X и Y для трехмерных графиков

Syntax

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

Описание

$[X, Y] = \text{meshgrid}(x, y)$ преобразует область, указанную векторами x и y , в массивы X и Y , которые могут использоваться для вычисления функций двух переменных, отображаемых в виде трехмерных сетчатых / поверхностных графиков. Строки выходного массива X являются копиями вектора x ; столбцы выходного массива Y являются копиями вектора y .

$[X, Y] = \text{meshgrid}(x)$ совпадает с $[X, Y] = \text{meshgrid}(x, x)$.

$[X, Y, Z] = \text{meshgrid}(x, y, z)$ создает трехмерные массивы, используемые для вычисления функций трех переменных и трехмерных объемных графиков.

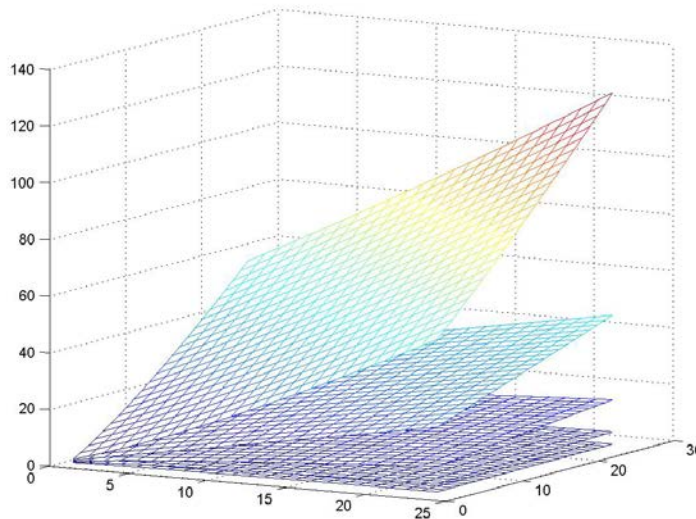
Примечания

Функция `meshgrid` похожа на `ndgrid`, за исключением того, что порядок первых двух входных и выходных аргументов переключается. То есть утверждение $[X, Y, Z] = \text{meshgrid}(x, y, z)$ дает тот же результат, что и $[Y, X, Z] = \text{ndgrid}(y, x, z)$.

Как следствие, `meshgrid` лучше подходит для задач в двумерном или трехмерном декартовом пространстве, а `ndgrid` лучше подходит для многомерных задач, которые не основаны на пространстве. `meshgrid`, которое ограничено двумерным или трехмерным декартовым пространством.

- $[X, Y] = \text{meshgrid}(x, y)$ преобразует область, указанную векторами x и y ,
- в массивы X и Y , которые могут использоваться для оценки функций двух
- переменных и трехмерных сетчатых / поверхностных графиков. Строки выходного
- массива X являются копиями вектора x ; столбцы выходного массива Y являются
- копиями вектора y .
- $[X, Y] = \text{meshgrid}(x)$ совпадает с $[X, Y] = \text{meshgrid}(x, x)$.
- $[X, Y, Z] = \text{meshgrid}(x, y, z)$ создает трехмерные массивы, используемые для вычисления функций трех переменных и трехмерных объемных графиков. Размеры всех векторов должны быть не менее чем 3. Например:

```
[x,y,z] = meshgrid(1:24,1:24,1:5);  
u=(x+y).^(z/4);  
[d1,d2,d3]=size(u);  
hold on  
for k=1:d3  
    mesh(x(:,:,k),y(:,:,k),u(:,:,k));  
end;  
grid on;  
hold off;
```



После прорисовки, график необходимо развернуть в удобное положение или выполнить поворот программно:

```
az = 30;
el = 30;
view(az, el);
```

ФУНКЦИИ MESH, MESHС, MESHZ

Прорисовка функций двух переменных X, Y в виде сетки Z

Syntax:

```
mesh(X, Y, Z)
mesh(Z)
mesh(..., C)
mesh(..., 'PropertyName', PropertyValue, ...)
meshc(...)
meshz(...)
h = mesh(...)
h = meshc(...)
h = meshz(...)
```

Описание:

Функции `mesh`, `meshc` и `meshz` создают параметрические поверхности в виде каркаса (сетки), заданные X , Y и Z , с цветом, заданным C .

`mesh(X, Y, Z)` рисует каркасную сетку с цветом, который определяется по значениям Z , поэтому цвет пропорционален высоте поверхности,

Если X и Y - векторы, $\text{length}(X) = n$ и $\text{length}(Y) = m$, то $[m, n] = \text{size}(Z)$. В этом случае пересечения линий сетки каркаса; X и Y соответствуют столбцам и строкам Z соответственно.

Если X и Y - матрицы, то пересечения линий сетки каркаса. `mesh(Z)` рисует каркасную сетку, используя $X = 1:n$ и $Y = 1:m$, где $[m, n] = \text{size}(Z)$. Высота, Z , является однозначной функцией, определенной над прямоугольной сеткой. Цвет пропорционален высоте поверхности.

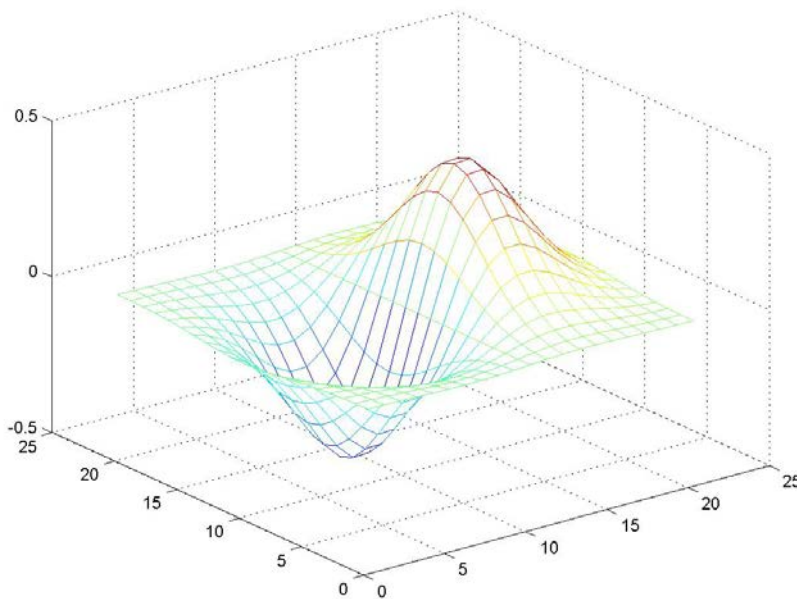
`mesh(..., C)` рисует каркасную сетку с цветом, определяемым матрицей `C`. MATLAB выполняет линейное преобразование данных в `C` для получения цветов из текущей цветовой карты. Если `X`, `Y` и `Z` являются матрицами, они должны быть того же размера, что и `C`.

- `mesh(..., 'PropertyName', PropertyValue, ...)` задает значение указанного свойства поверхности. Множественные значения свойств могут быть заданы с помощью одного оператора.
- `meshc(...)` рисует контурный график под сеткой.
- `meshz(...)` рисует график занавеса (то есть опорную плоскость) вокруг сетки.
- `h = mesh(...)`, `h = meshc(...)` и `h = meshz(...)` возвращает дескриптор объекта поверхности.

Примеры

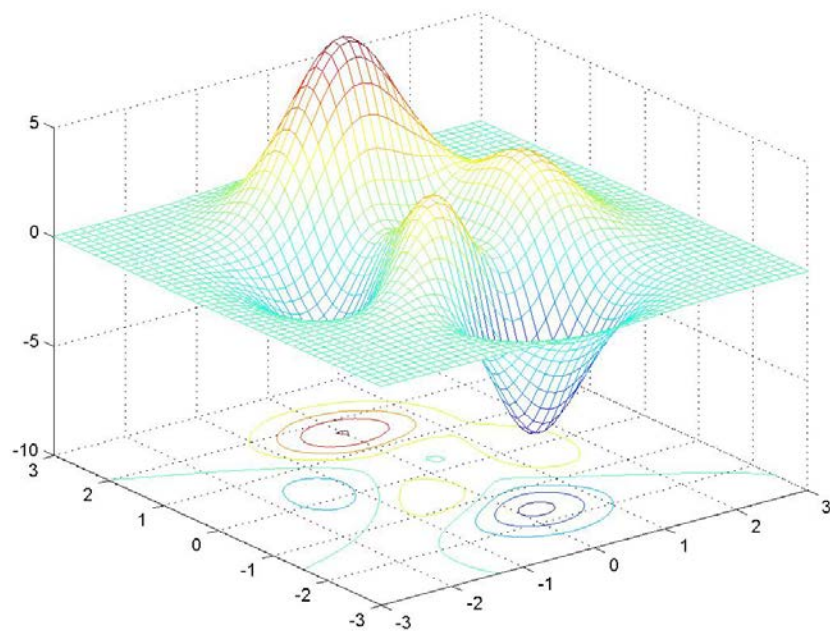
% Пример mesh

```
[X,Y] = meshgrid(-2:.2:2, -2:.2:2);  
Z = X .* exp(-X.^2 - Y.^2);  
mesh(Z)
```



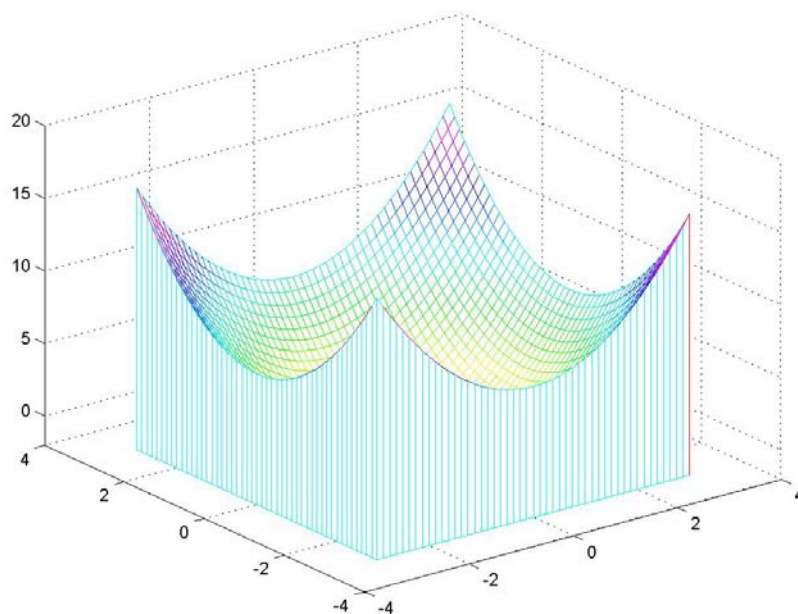
% Пример meshc

```
[X,Y] = meshgrid(-3:.125:3);  
Z = peaks(X,Y);  
meshc(X,Y,Z);  
axis([-3 3 -3 3 -10 5])
```



% Пример meshz

```
[X,Y] = meshgrid(-3:.125:3);
Z = -2+X.^2+Y.^2;
meshz(X,Y,Z);
colormap hsv
axis([-4 4 -4 4 -2 20])
```



ФУНКЦИЯ COLORMAP

MATLAB поддерживает несколько цветовых палитр.

Syntax

```
colormap(map)
```

```
colormap('default')
cm = colormap
```

Палитры (*map*) для формата *colormap(map)*:

- **autumn** - меняется плавно от красного, от оранжевого до желтого.
- **bone** - это цветовая палитра в оттенках серого с более высоким значением для синего компонента. Этот набор цветов полезен для добавления «электронного» вида к изображениям в оттенках серого.
- **colorcube** - содержит как можно больше цветных цветов в цветовом пространстве RGB, пытаясь обеспечить больше шагов серого, чистого красного, чистого зеленого и чистого синего.
- **cool** - состоит из цветов, которые являются оттенками голубого и пурпурного. Он изменяется плавно от голубого до пурпурного.
- **copper** - плавно меняется от черной до яркой меди.
- **flag** - состоит из красных, белых, синих и черных цветов. Эта цветовая палитра полностью изменяет цвет с каждым приращением индекса.
- **gray** - возвращает линейную цветовую шкалу серого. горячий плавно меняется от черного, через оттенки красного, оранжевого и желтого до белого.
- **hsv** - изменяет компонент оттенка цветовой модели оттенка насыщенности. Цвета начинаются с красного, проходят через желтый, зеленый, голубой, синий, пурпурный и возвращаются к красному. Цветовая палитра особенно подходит для отображения периодических функций. $\text{hsv}(m)$ совпадает с $\text{hsv2rgb}([h \text{ единиц}(m, 2)])$, где h - линейная рампа, $h = (0:m-1) / m$.
- **jet** - колеблются от синего до красного и проходят через цвета, голубые, желтые и оранжевые. Это вариация цветовой схемы hsv. Каркас струи связан с аэрофизическим моделированием струй жидкости из Национального центра суперкомпьютерных приложений. См. Раздел «Примеры».
- **lines** - генерируют цветовую палитру цветов, определяемую свойством ColorOrder, а также оттенком серого.
- **pink** - содержит пастельные оттенки розового. Розовая цветовая палитра обеспечивает сепию цветовую окраску оттенков серого.
- **prism** - повторяет шесть цветов: красный, оранжевый, желтый, зеленый, синий и фиолетовый.
- **spring** - состоит из цветов, которые являются оттенками пурпурного и желтого.
- **summer** - состоит из цветов, оттенков зеленого и желтого.
- **white** - это белая монохромная цветовая палитра.
- **winter** - состоит из цветов, которые являются оттенками синего и зеленого.

Описание:

Цветовая палитра представляет собой матрицу действительных чисел ($m \times 3$)- matrix, принимающую значения между 0.0 и 1.0. Каждая строка представляет собой вектор RGB, который определяет один цвет. k -я строка цветовой палитры определяет k -й цвет, где $\text{map}(k, :) = [r(k) \ g(k) \ b(k)]$ определяет интенсивность красного, зеленого и синего.

`colormap(map)` устанавливает цветовую карту в матричную карту. Если любые значения на карте находятся за пределами интервала $[0 \ 1]$, MATLAB возвращает ошибку: 'colormap must have values in $[0,1]$ '.

`colormap('default')` устанавливает текущую цветовую палитру в стандартную цветовую палитру.
`cmap = colormap`; извлекает текущую цветовую палитру в матрицу `cmap`. Возвращаемые значения находятся в интервале `[0 1]`.

Примеры для формата `colormap(map)`:

```
colormap(jet);  
colormap(gray);
```

ФУНКЦИЯ SHADING

Установка свойства затенения цвета

Syntax

```
shading flat  
shading faceted  
shading interp
```

Описание

Функция затенения управляет цветовой штриховкой поверхностей и патчей графических объектов.

shading flat -затенение каждого сегмента сетки и поверхности имеет постоянный цвет, определяемый значением цвета в конечной точке сегмента или углом поверхности, имеющим наименьший индекс или индексы.

shading faceted -затенение граненое плоское затенение с наложенными черными линиями сетки. Это режим затенения по умолчанию.

shading interp изменяет цвет в каждом сегменте линии и поверхности, путем интерполяции индекса `colormap` или истинного значения цвета по линии или поверхности.

Примеры

```
shading flat  
shading faceted  
shading interp
```

ФУНКЦИЯ VIEW

Установка точки просмотра

Syntax

```
view(az,el)  
view([az,el])  
view([x,y,z])  
view(2)  
view(3)  
view(T)
```

```
[az,el] = view  
T = view
```

Описание

Позиция зрителя (точка обзора) определяет ориентацию осей. Вы указываете точку зрения в терминах азимута и возвышения, или точкой в трехмерном пространстве.

`view (az, el)` и `view ([az, el])` задают угол обзора для трехмерного графика. Азимут, `az`, представляет собой горизонтальное вращение вокруг оси `z`, измеренное в градусах от отрицательной оси `y`. Положительные значения указывают вращение точки зрения против часовой стрелки. `el` - вертикальное возвышение точки зрения в градусах. Положительные значения высоты соответствуют движению над объектом; отрицательные значения соответствуют перемещению ниже объекта.

- `view ([x, y, z])` задает точку зрения на декартовы координаты `x`, `y` и `z`. Величина (`x`, `y`, `z`) игнорируется.
- `view (2)` устанавливает двумерный вид по умолчанию, `az = 0`, `el = 90`.
- `view (3)` задает трехмерный вид по умолчанию, `az = -37.5`, `el = 30`.
- `view (T)` задает вид согласно матрица преобразования `T`, которая представляет собой матрицу 4 на 4, такую как перспективное преобразование, созданное `viewmtx`. `[az, el] = view` возвращает текущий азимут и высоту.
- `T = view` возвращает текущую матрицу преобразования 4 на 4.

Примеры

```
[X,Y] = meshgrid(-2:.2:2, -2:.2:2);
Z = X .* exp(-X.^2 - Y.^2);
az = 0;
el = 30;
view(az,el);
mesh(Z);
% Для немедленного завершения цикла нажмите Ctrl+Break
for k=1:8
    pause(1);
    az = az+10;
    view(az,el);
end;
```

ФУНКЦИЯ SURFL

Прорисовка функций двух переменных `X`, `Y` в виде поверхности `Z`.
Размеры `X`, `Y` и `Z` должны быть одинаковыми и не менее чем 3 на 3.

Syntax

```
surfl (Z)
surfl (X, Y, Z)
surfl (..., 'цвет')
surfl (..., s)
surfl (X, Y, Z, S, k)
h = surfl (...)
```

Описание

Функция `surfl` отображает заштрихованную поверхность, основанную на комбинации окружающих, диффузных и зеркальных моделей освещения.

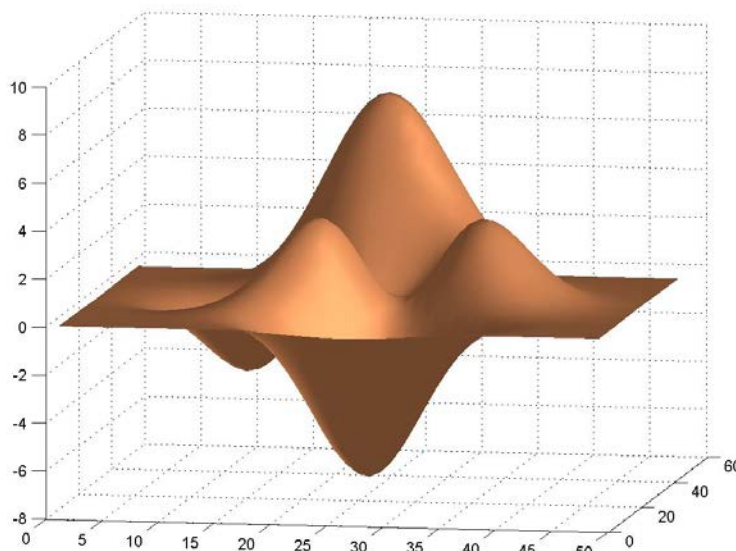
`surf` (*Z*) и `surf` (*X*, *Y*, *Z*) создают трехмерные затененные поверхности, используя направление по умолчанию для источника света и коэффициенты освещения по умолчанию для модели затенения. *X*, *Y* и *Z* - векторы или матрицы, которые определяют компоненты *x*, *y* и *z* поверхности.

- `surf` (... , 'light') создает цветную освещенную поверхность с использованием объектива MATLAB. Это дает результаты, отличные от метода освещения по умолчанию, `surf` (... , 'cdata'), который изменяет цветовые данные для поверхности как отражательную способность поверхности.
- `surf` (... , *s*) определяет направление источника света. *s* - двух- или трехэлементный вектор, определяющий направление от поверхности к источнику света. *s* = [*sx sy sz*] или *s* = [азимутальная высота]. Значение по умолчанию *s* составляет 45 ° против часовой стрелки от текущего направления просмотра.
- `surf` (*X*, *Y*, *Z*, *s*, *k*) определяет константу отражения *k*. *k* представляет собой четырехэлементный вектор, определяющий относительные вклады окружающего света, диффузного отражения, зеркального отражения и коэффициента зеркального блеска. *k* = [*ka kd ks shine*] и по умолчанию - [.55, .6, .4,10].
- *h* = `surf` (...) возвращает дескриптор объекта поверхности.

Примеры:

% Пример 1

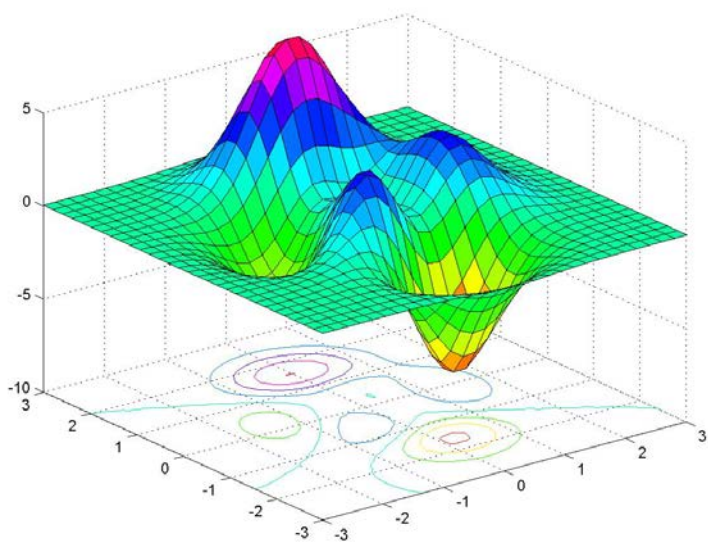
```
view([10 10])
grid on
hold on
surf(peaks)
shading interp
colormap copper
hold off
```



% Пример 2

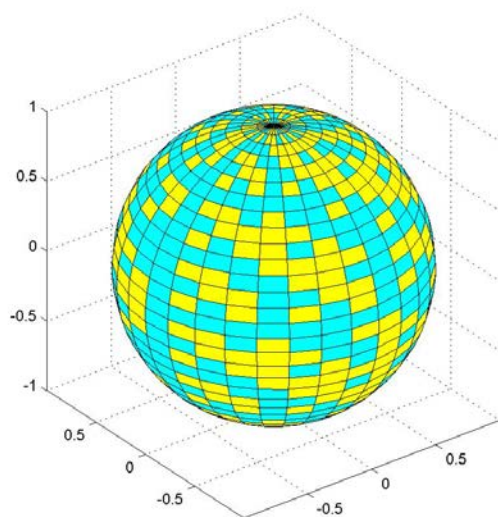
```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z)
```

```
colormap hsv
axis([-3 3 -3 3 -10 5])
```



% Пример 3

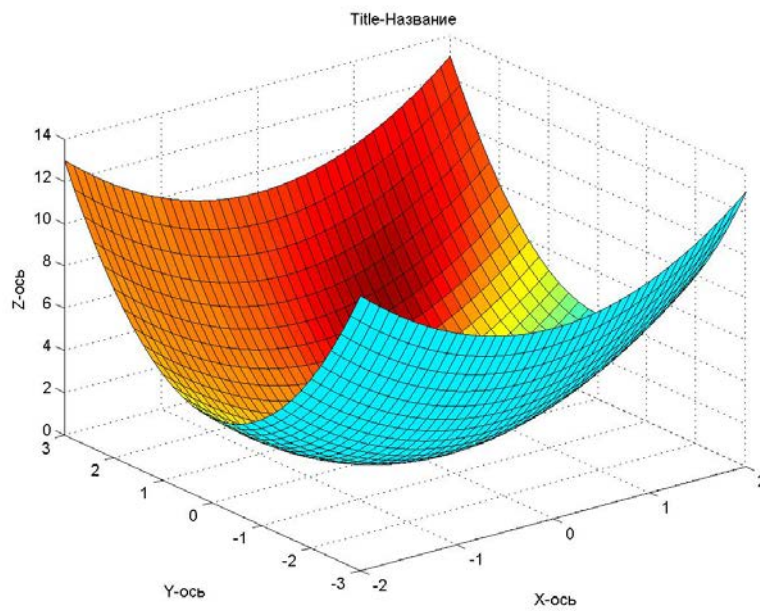
```
k = 5;
n = 2^k-1;
[x,y,z] = sphere(n);
c = hadamard(2^k);
surf(x,y,z,c);
colormap([1 1 0; 0 1 1])
axis equal
```



% Пример 4

```
[x,y] = meshgrid(-2:0.1:2, -3:0.2:3);
z = x.^2+y.^2;
surfl(x,y,z);
shading faceted
colormap(jet);
xlabel('X-ось');
ylabel('Y-ось');
```

```
Zlabel('Z-ось');
Title('Title-Название');
```



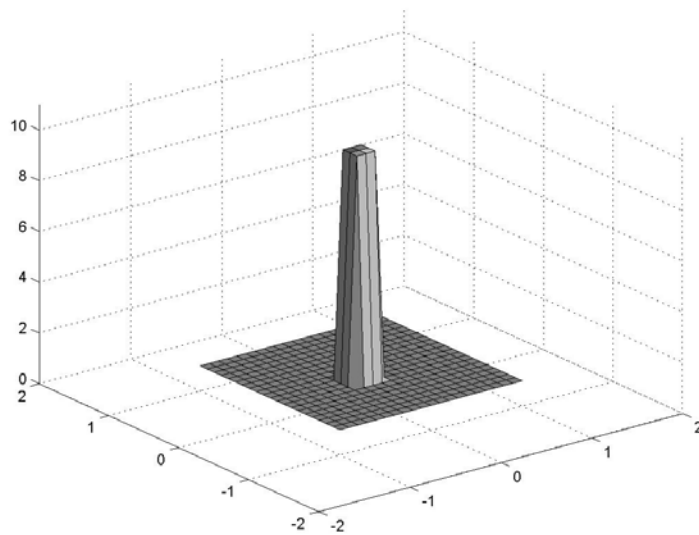
% Пример 5

```
[X,Y] = meshgrid(-1:0.1:1);
[d1,d2]=size(X);
Z=ones(d1,d2);
a=10;
Z(9,9)=a;
Z(9,10)=a;
Z(9,11)=a;

Z(10,9)=a;
Z(10,10)=a;
Z(10,11)=a;

Z(11,9)=a;
Z(11,10)=a;
Z(11,11)=a;

% mesh(X,Y,Z);
% colormap jet;
surfl(X,Y,Z);
colormap gray;
axis([-2 2 -2 2 0 11]);
```



% Пример 6

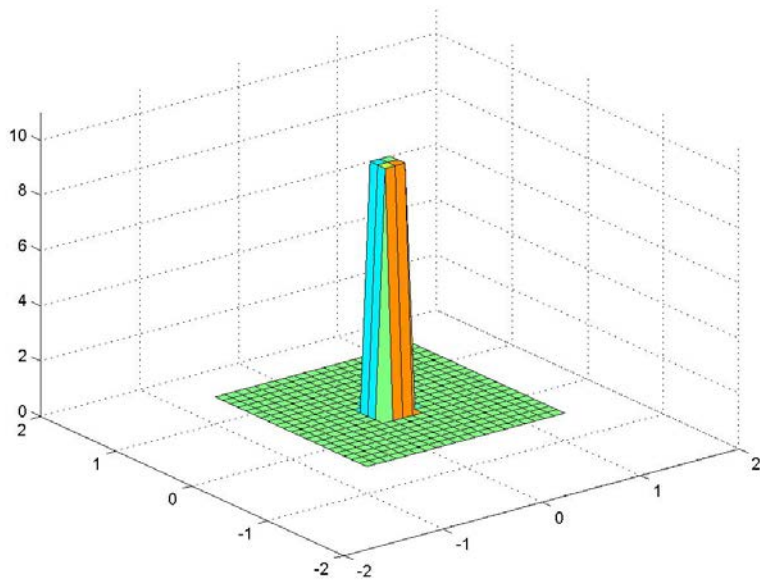
```
[X,Y] = meshgrid(-1:0.1:1);
[d1,d2]=size(X);
Z=ones(d1,d2);
a=10;
c1=fix(d1/2);
c2=fix(d2/2);

Z(c1-1,c2-1)=a;
Z(c1-1,c2)=a;
Z(c1-1,c2+1)=a;

Z(c1,c2-1)=a;
Z(c1,c2)=a;
Z(c1,c2+1)=a;

Z(c1+1,c2-1)=a;
Z(c1+1,c2)=a;
Z(c1+1,c2+1)=a;

% mesh(X,Y,Z);
% colormap jet;
surf1(X,Y,Z);
colormap jet;
axis([-2 2 -2 2 0 11]);
```

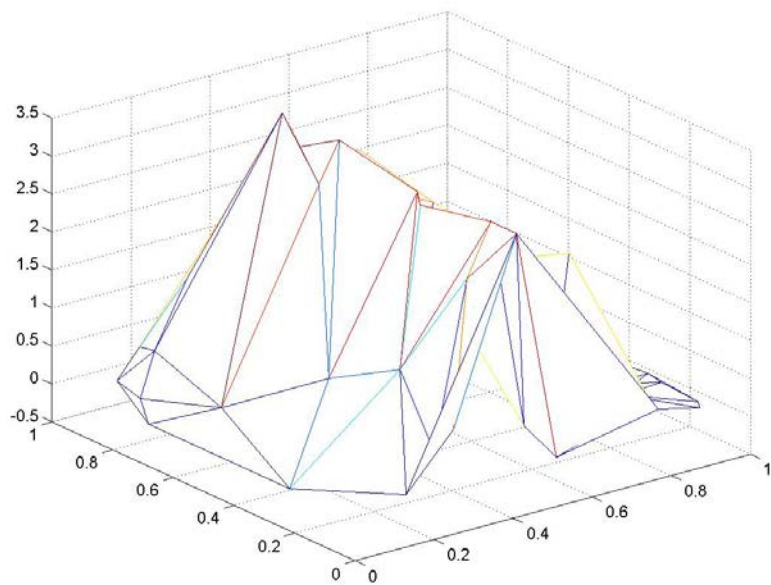


ФУНКЦИИ DELAUNAY TRIMESH TRISURF

Для самостоятельного изучения

% Пример 1

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
trimesh(tri,x,y,z)
```



% Пример 2

```
x = rand(1,50);
y = rand(1,50);
```

```
z = peaks(6*x-3,6*x-3);  
tri = delaunay(x,y);  
trisurf(tri,x,y,z)
```

