%% Лекция 7

```
%% inline функции не рекомендуется.
%% Используйте Анонимные функции вместо этого.
clc, clear;

% функциональная зависимость f(alpha, x)
f = inline('sin(alpha*x)');

% alpha=pi/2, x=1
x1=f(pi/2,1)

% alpha=pi/2, x=2
x2=f(pi/2,2)
```

%% Анонимные функции

```
%  Анонимные функции могут принять несколько входных параметров
%  и возвратить один выходной параметр.
%  Они могут содержать только один исполняемый оператор.
clc;

myf = @(alpha, x) sin(alpha.*x);

x_1=myf(pi/2,1)
x_2=myf(pi/2,2)

%% Проверка результатов на равенство
x1==x_1
x2==x_2
```

%% Интеграл от анонимной функции

```
clc
myf_x = @(x) x;
q1 = integral(myf_x,0,1)
```

%% Интеграл от анонимной функции

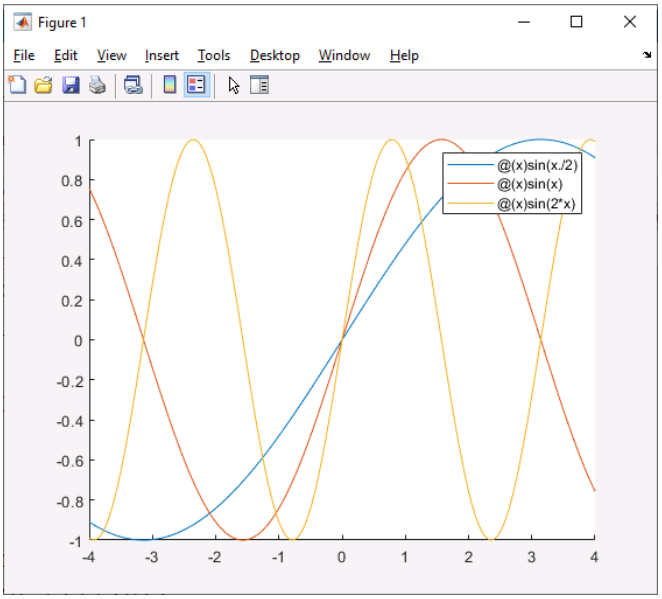
```
clc
q2 = integral(@(x) x,0,1)
%%
q1==q2
```

%% Анонимная функция "помнит" коэффициенты при их очиске-clear

```
a = 10;
b = 2.;
c = 3.5;

% обязательно определить a b c до функции
parabola = @(x) a*x.^2 + b*x + c;
clear a b c;

parabola(1)
```

```
%% новый указатель на функцию
% для пересчета с новыми парамерами,
% необходимо создать новый указатель на функцию:
a = 20;
b = 4.;
c = 1.5;
parabola = @(x) a*x.^2 + b*x + c;
parabola(1)

%% Несколько анонимных функций
% Выражение в анонимной функции может включать другую анонимную
функцию.

% подынтегральное выражение как анонимная функция
% 1.) @(x) (x.^2 + c*x + 1)
% передача указателя на функцию integral
% 2.) integral(@(x) (x.^2 + c*x + 1),0,1)
% анонимная функция от @(c)
% 3.)

g = @(c) (integral(@(x) (x.^2 + c*x + 1),0,1));
g(2)


%% Анонимная функция без параметров
t = @() datestr(now);
d = t()

% Результат
% d ='17-Mar-2022 12:38:19'


%% Анонимная функция с несколькими параметрами

myfunction = @(x,y,z) sqrt(x.^2 + y.^2 + z.^2);

x = 1;
y = 2;
z = 3;
r = myfunction(x,y,z)

%% Несколько выходных параметров у анонимной функции

% Однако анонимная функция возвращает только один выходной параметр.
% Если выражение в функции возвращает несколько выходных параметров,
% то можно запросить их, когда вы вызываете указатель на функцию.
c = 20;

mygrid = @(x,y) ndgrid((-x:x/c:x),(-y:y/c:y));

[x,y] = mygrid(pi,2*pi);
```

```
z = sin(x) + cos(y);
surf(x,y,z)


%% Массивы ячеек анонимных функций

f = {@(x)x.^2;
     @(y)y+10;
     @(x,y)x.^2+y+10};

x = 1;
y = 10;

% Вызов функций
f{1}(x)
f{2}(y)
f{3}(x,y)

%% feval - вычисляет функцию
fun = 'round';
x1 = pi;
y = feval(fun,x1)

%%
fun = 'ceil';
x1 = pi;
y = feval(fun,x1)

%%
fun = 'floor';
x1 = exp(1.);
y = feval(fun,x1)

%% Пример
clear, clc

f = {@(x) sin(x./2); @(x)sin(x); @(x)sin(2*x)};

x =-4:0.05:4
str={};

figure(1)
hold on

for i=1:numel(f)

    plot(x,f{i}(x))

    str{i}=func2str(f{i});

end

legend(str)
```



%% Примеры функций с переменным числом входных и выходных параметров

```
% Пример 1. Без входных параметров с одним выходным параметром
% Суммировать числа, вводимые с клавиатуры, пока пользователь не
введет ноль

clc, clear;

%% В файле-функции mySumm1_lecture_07

% function [s] = mySumm1_lecture_07()
% s=0;
% while true
%     x=input('Введите число ')
%     if (x~=0)
%         s=s+x;
%     else break;
%     end
% end
% end

%% Вызов функции
s=mySumm1_lecture_07()

%%
% Пример 2. nargin - количество входных параметров
% varargin - массив ячеек входных параметров
% cell2mat - ячейку преобразовать к матрице

% Если количество входных массивов - четное число, то найти сумму
% Если количество входных массивов - НЕчетное число,
% то найти произведение и сумму

%% В файле-функции mySumm2_lecture_07
% function [s,p] = mySumm2_lecture_07(varargin)
% s=0;
% p=1;

% if mod(nargin,2)==0

%     for i=1:nargin

%         s=s+sum(sum(cell2mat(varargin(i))));

%     end

% else
%     for i=1:nargin
%         s=s+sum(sum(cell2mat(varargin(i))));
%         p=p*prod(prod(cell2mat(varargin(i))));
%     end
```

```
% end
% end

%% Вызов функции
clc, clear

[s]=mySumm2_lecture_07([10, 20], [30, 40])

[s,p]=mySumm2_lecture_07([10, 20], [30, 40],[60])


%% Пример из help

%% В файле-функции mySumm3_lecture_07

% function [c] = mySumm3_lecture_07(a,b)

%     switch nargin
%         case 2
%             c = a + b;

%         case 1
%             c = a + a;

%         otherwise
%             c = 0;

%     end
% end


%% Вызов функции. 2 параметра
clc, clear
c=mySumm3_lecture_07([1, 2],[3, 4])


%% Вызов функции. 1 параметр
clc, clear
c=mySumm3_lecture_07([1,2,3])


%% Вызов функции. 0 параметров
clc, clear
c=mySumm3_lecture_07()
```

```
%% Пример произвольное количество входных параметров
%% varargin - массив ячеек входных параметров spy(matrix)

%% В файле-функции myMatrPlot_lecture_07

% cell2mat - преобразовать массив ячеек к матрице!!!

% function [] = myMatrPlot_lecture_07(varargin)

% if nargin~=0

% for i=1:nargin

%     figure(i)
%     spy(cell2mat(varargin(i)),16,'*')

% end

% else
%     disp(['No parametrs!']);
% end


%% Вызов функции. 1 параметр
clc, clear
myMatrPlot_lecture_07([1 2;0 4])

%% Вызов функции. 2 параметра
clc, clear

m1=tril(magic(5));

m2=eye(3)+fliplr(eye(3));

myMatrPlot_lecture_07(m1,m2)

%% Вызов функции. 0 параметров
clc, clear

myMatrPlot_lecture_07()


%% Пример
%%произвольное количество выходных и входных параметров

% Квадратные матрицы делаем нижними треугольными
% и выводим на экран структуру матрицы

%% varargin - массив ячеек входных параметров

%% varargout - массив ячеек выходных параметров

%% В файле-функции myMatr_lecture_07
```

```
% cell2mat - преобразовать массив ячеек к матрице!!!

% function [varargout] = myMatr_lecture_07(varargin)

% if nargin~=0

% for i=1:nargin

%     figure(i);

%     temp=cell2mat(varargin(i));

%     varargout(i)={tril(temp)};

%     spy(tril(temp),16,'*');

% end

% else
%     disp(['No parametrs!']);
% end


%% Вызов функции. 1 параметр
clc, clear

[r1]=myMatr_lecture_07([1 2;3 4])


%% Вызов функции. 2 параметра
clc, clear

m1=tril(magic(5));

m2=eye(3)+fliplr(eye(3));

[r1,r2]=myMatr_lecture_07(m1,m2)


%% Вызов функции. 3 параметра
clc, clear

m1=tril(magic(2));
m2=eye(3)+fliplr(eye(3));
m3=tril(magic(7));

[r1,r2,r3]=myMatr_lecture_07(m1,m2,m3)


%% Вызов функции. 0 параметров
clc, clear

myMatr_lecture_07()
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Сортировка пузырьком
% Сортировка простыми обменами, сортировка пузырьком
%(англ. bubble sort)
% простой алгоритм сортировки.
% Для понимания и реализации этот алгоритм простейший,
% но эффективен он лишь для небольших массивов.
% Сложность алгоритма: O(n^2).
%
% Алгоритм считается учебным и практически не применяется
% вне учебной литературы, вместо него на практике применяются
% более эффективные алгоритмы сортировки.
% В то же время метод сортировки обменами лежит в основе
% некоторых более совершенных алгоритмов, таких как
% шейкерная сортировка, пирамидальная сортировка и быстрая
% сортировка.

%% В файле-функции BubbleSort

% function [a] = BubbleSort(a)

% n=numel(a);

% for j=1:n

%     for i=1:n-j

%         if a(i)>a(i+1)

%             t=a(i);

%             a(i)=a(i+1);

%             a(i+1)=t;

%         end

%     end

% end

% end

%% Вызов функции
clc,clear
a=[6,5,3,1,8,7,2,4,9]
a=BubbleSort(a)
```

```
%% Установка точности вычислений
clc,clear

digits(10);

currentPrecision = digits

digitsNew = digits(25);

pi25 = vpa(pi)

currentPrecision = digits

%% Вывод результата на экран

% currentPrecision = 10

% pi25 = 3.1415926535897932384626433

% currentPrecision =25


%% Ввод и вывод
%% input


% Ввод числа
disp('Ввод числа');

x=input('Введите целое число ')

x=x+2


%% Ввод матрицы
disp('Ввод матрицы');

A=input('Введите матрицу ')

A=5*A


%% Ввод строки
clc, clear

disp('Ввод строки');

myName=input('Введите Ваше имя ','s')

reverse(myName)
```

```
%% Вывод на экран
%% display
clear, clc
A=[1 2; 3 4];

disp(A);

%%
disp(4 * 5 - 13)

%% fprintf - форматированный вывод
clear, clc

disp('fprintf - форматированный вывод');

a = [3.14; 2.71; 9.81]

fprintf('%d\n',floor(a));

%%
clc
a = [3.14; 2.71; 9.81]
fprintf('%5.2f\n',a);

%%
clc
a = [3.14; 2.71; 9.81]
fprintf('%4.1e\n',a);


%%
clc, clear
A1 = [9.9, 9900];
A2 = [8.8,  7.7 ; ...
      8800, 7700];

formatSpec = '%4.2f метров или %8.3f миллиметров\n';

fprintf(formatSpec,A1,A2)

%%
clc, clear
x=[10 10; 11 11; 12 12; 13 13; 14 14; 15 15; 16 16]'

formatSpec = '%4d (десятичный формат) или %4x (шестнадцатиричный
формат) \n';

fprintf(formatSpec,x)
```

```
%%
%d или %i - Основание 10
%o - Основание 8 (восьмеричное)
%u - Целое, беззнаковое
%x - Основание 16 (шестнадцатеричное), строчные буквы a-f
%X - Основание 16 (шестнадцатеричное), заглавные буквы A-F
%f - Представление с фиксированной запятой
%e - Экспоненциальное представление, например 3,141593e+00
%E - То же, что и %e, но в верхнем регистре, например 3.141593E+00
%g - Более компактное представление из %e или %f без нулей в конце
%G - Более компактное представление из %e или %f без нулей в конце
%c - Символы или строки
%s - Вектор символов или массив строк
```

%% Флаги
%% + показывать знак числа

```
clc,clear
fprintf('%+5.2f \n',-2.71);
fprintf('%+5.2f \n',2.71);
```

%% ' ' пробел перед значением % 5.2f

```
clc,clear
fprintf('% .2f \n',2.71);
```

%% 0 Дополнить ширину поля нулями перед значением.

```
clc,clear
fprintf('%06.2f \n',2.71);
```

%% #
%% Для %o, %x или %X выведите префикс 0, 0x или 0X.

```
clc
fprintf('%#o \n',8);
fprintf('%#x \n',15);
fprintf('%#X \n',15);
```

%% Для %f, %e или %E печатайте десятичную точку, даже если точность равна 0.

```
clc
fprintf('%#5.0f \n',2.71);
fprintf('%#5.0e \n',2.71);
fprintf('%#5.0E \n',2.71);
```

%% Для %g или %G не удаляйте конечные нули или десятичную точку.

```
clc
fprintf('%#.4g \n',3.);
fprintf('%#.4G \n',2.71);
```

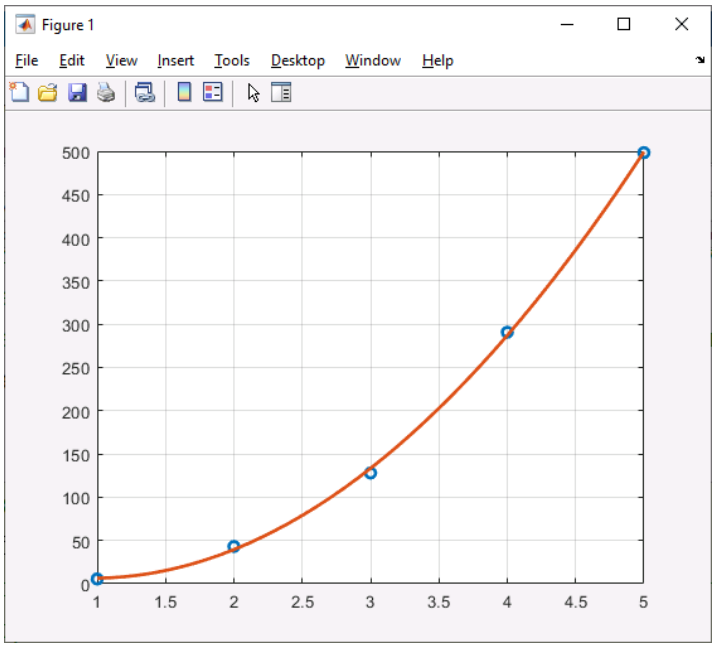
```
%% таблицу excel в matlab
clear
clc
values = {1, 2, 3 ; 4, 5, 'x' ; 7, 8, 9};
headers = {'First','Second','Third'};
xlswrite('D:\myExample.xlsx',[headers; values])

%% Таблицу в файл
clc, clear
x = 0:.1:1;
A = [x; exp(x)];
fileID = fopen('my_text001.txt','w');
fprintf(fileID,'%6s %12s\n','x','exp(x)');
fprintf(fileID,'%6.2f %12.8f\n',A);
fclose(fileID);

%%
%% Работа с полиномами

%% roots  - поиск корней полинома

% x^2+5*x+6=0
clc, clear
r=roots([1 5 6])
r(1)
r(2)
```



```
% poly - построить полином по заданным корням
% возвращает коэффициенты полинома
p=poly(r(1),r(2))

%% poly - построить полином по заданным корням
% возвращает коэффициенты полинома
p=poly([-2 -3])

%% polyval - вычисляет значения полиниома в заданных точках
x=-5:0.1:5;
p=[1 5 6];
y=polyval(p,x);
plot(x,y)
grid

%% conv  - умножение полиномов
clc
p=conv([1 2],[1 -2])
% a = s.^2 + 2*s + 3 и b = 4*s.^2 + 5*s + 6
a = [1 2 3];
b = [4 5 6];
c=conv(a,b)
```

```
%% Для получения из c полинома b воспользуемся функцией deconv
% deconv - деление полиномов. возвращает частное и
остаток
[q, r] = deconv(c, a)

%% Ответ
% q = 4     5     6
% r = 0     0     0     0     0

%%
[q, r] = deconv(c, b)

%% Ответ
% q = 1     2     3
% r = 0     0     0     0     0

%% Вычисление производных от полиномов
p = [1 0 -2 -5]
q = polyder(p)

%% производная произведения a*b полиномов
a = [1 3 5]; b = [2 4 6];
c = polyder(a, b)

%% производная от частного a/b

% 1. функция polyder с двумя выходными аргументами:
[q, d] = polyder(a, b)

% 2. отношение двух полиномов q/d является результатом операции
дифференцирования
deconv(q, d)

%% Аппроксимация кривых полиномами

% Функция polyfit находит коэффициенты полинома заданной
степени n ,
% который аппроксимирует данные (или функцию y(x)) в смысле
метода
% наименьших квадратов: p = polyfit(x, y, n)
% где x и y есть векторы, содержащие данные x и y,
% которые нужно аппроксимировать полиномом.
% Например, рассмотрим совокупность данных x-y, полученную
экспериментальным путем

clc, clear
x = [1 2 3 4 5];
y = [5.5 43.1 128 290.7 498.4];

%% Аппроксимация функциональной зависимости y(x) в виде полинома
третьего порядка
p = polyfit(x,y,3) % коэффициенты полинома
```

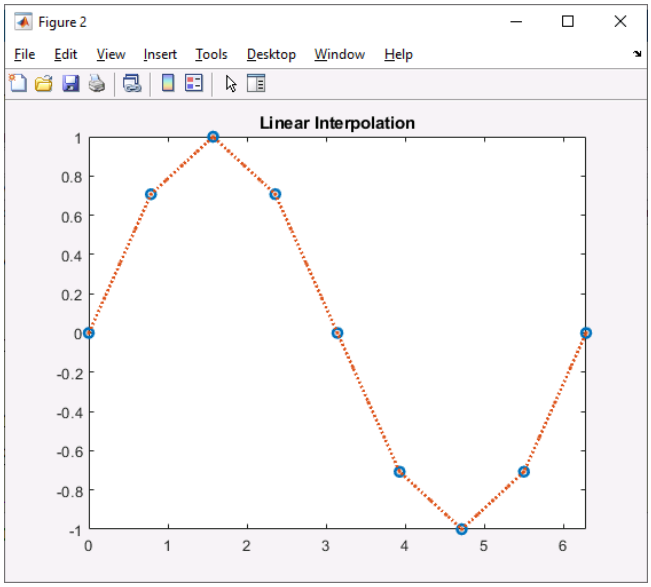
```
% Рассчитаем теперь значения полинома, полученного при помощи
функции
% polyfit, на более мелкой шкале (с шагом 0.1) и построим
% для сравнения графики (это делает функция plot) реальных
данных
% и аппроксимирующей кривой.
x2 = 1 : 0.1 : 5;
y2 = polyval(p, x2);

g=plot(x, y, 'o', x2, y2)
set(g,linewidth=2);
grid on;
```



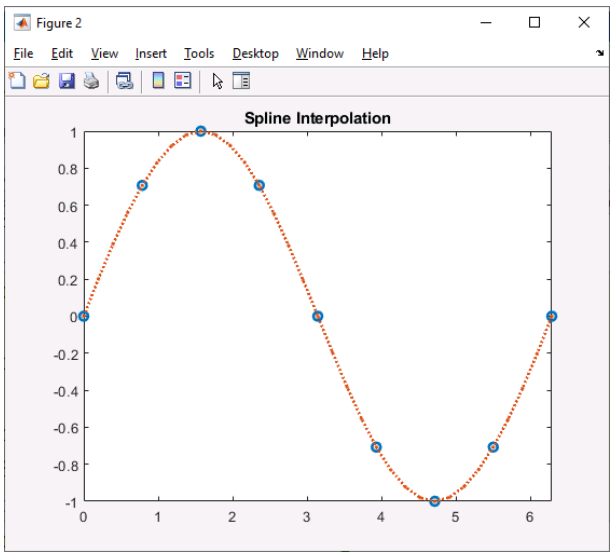
%% Интеполяция - способ нахождения промежуточных значений величины

```
%%  по имеющемуся дискретному набору известных значений
% Интерполяция является процессом вычисления (оценки)
% промежуточных значений функций, которые находятся
% между известными или заданными точками.
% Она имеет важное применение в таких областях как теория
сигналов,
% обработка изображений и других.
% MATLAB обеспечивает ряд интерполяционных методик,
% которые позволяют находить компромисс между точностью
представления
% интерполируемых данных и скоростью вычислений и используемой
памятью.
```

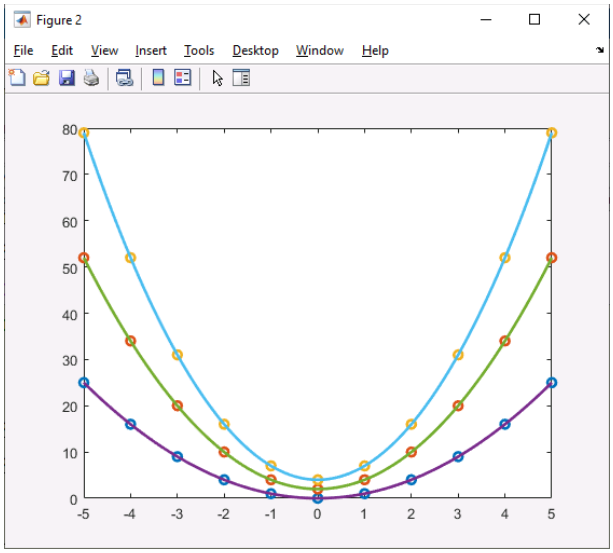
```
% Обзор функций интерполяции

% % Функции          % Описание
% % griddata         % Двумерная интерполяция на неравномерной сетке.
% % griddata3        % Трехмерная интерполяция на неравномерной сетке.
% % griddatan        % Многомерная интерполяция (n >= 3).
% % interp1          % Одномерная табличная интерполяция.
% % interp2          % Двухмерная табличная интерполяция.
% % interp3          % Трехмерная табличная интерполяция.
% % interpft         % Одномерная интерполяция с использованием быстрого
преобразования Фурье.
% % interpn          % Многомерная табличная интерполяция.
% % pchip            % Кубическая интерполяция при помощи полинома
Эрмита.
% % spline           % Интерполяция кубическим сплайном.


%% 1D Одномерная табличная интерполяция
%     vq = interp1(x,v,xq)
%     vq = interp1(x,v,xq,method)
%     vq = interp1(x,v,xq,method,extrapolation)
%     vq = interp1(v,xq)
%     vq = interp1(v,xq,method)
%     vq = interp1(v,xq,method,extrapolation)
%     pp = interp1(x,v,method,'pp')

x=0:pi/4:2*pi; % точки данных или базовые точки
v=sin(x)       % точки данных или базовые точки
xq= 0:pi/16:2*pi; % точки
figure
vq1=interp1(x,v,xq);
g=plot(x,v,'o',xq,vq1,':.')
set(g,linewidth=2);
xlim([0 2*pi]);
title('Linear Interpolation')
```



```
%% Интерполяция кубическим сплайном
figure
vq1=interp1(x,v,xq,'spline');
g=plot(x,v,'o',xq,vq1,':.')
set(g,linewidth=2);
xlim([0 2*pi]);
title('Spline Interpolation')
```




```
%% Интерполяция нескольких наборов данных
x=(-5:5)';
v1=x.^2;
v2=2*x.^2+2;
v3=3*x.^2+4;
v=[v1 v2 v3];
xq=-5:0.1:5;
vq=interp1(x,v,xq,'spline');
figure
g=plot(x,v,'o',xq,vq);
set(g,linewidth=2);
h=gca;
h.XTick=-5:5;
```



```
%% 2D Interpolation
% Vq = interp2(X,Y,V,Xq,Yq)example
% Vq = interp2(V,Xq,Yq)
% Vq = interp2(V)
% Vq = interp2(V,k)example
[X,Y]=meshgrid(-3:3);
V=peaks(X,Y);
figure
surf(X,Y,V)
title('Original Sampling');

%%
[Xq,Yq]=meshgrid(-3:0.25:3);
Vq=interp2(X,Y,V,Xq,Yq);
figure
surf(Xq,Yq,Vq)
title('Linear Interpolation');

%% cubic
[Xq,Yq]=meshgrid(-3:0.25:3);
Vq=interp2(X,Y,V,Xq,Yq,'cubic');
figure
surf(Xq,Yq,Vq)
title('Cubic Interpolation');
```