Evaluation of test data

Least square method

The method of least squares based on experimental data allows to select such an analytical function that passes as close to the experimental points as possible.

Suppose during the experiment were obtained, some data recorded in a table (see Table 6.1.). Required to construct an analytic dependence that best describes the experimental results.

Table 6.1

x_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	…	x_n
y_i	y_1	y_2	y_3	y_4	y_5	y_6	y_7	…	y_n

The goal is to find the parameter values for the model $Y=f(x, a_0, a_1, \ldots, a_k)$ which "best" fits the data. The least squares method finds its optimum when the sum, *S*, of squared residuals:

$$S = \sum_{i=1}^{n} (y_i - f(x_i, a_1, \ldots, a_k))^2 \rightarrow \min. \qquad (6.1)$$

The problem is reduced to the determination of the coefficients a_i in equation (6.1). For achieve this objective in Matlab we use command **datafit**:

x = **lsqcurvefit**(fun,x0,xdata,ydata)

where starts at **x0** and finds coefficients x to best fit the nonlinear function **fun**(**x**,**xdata**) to the data **ydata** (in the least-squares sense). **ydata** must be the same size as the vector (or matrix) F returned by **fun**.

[**x,resnorm**] = **lsqcurvefit**(___), for any input arguments, returns the value of the squared 2-norm of the residual at x: sum((fun(x,xdata)-ydata).^2).

Problem 6.1.

As a result of the experience of idling of the induction motor determined the dependence consumption from the network power *(P, W)* of the input voltage *(U, V)* for the induction motor (tab. 6.2).

Table 6.2. Dependence of power consumed from the network by the input voltage.

U, V	132	140	150	162	170	180	190	200	211	220	232	240	251
P, W	330	350	385	425	450	485	540	600	660	730	920	1020	1350

Choose the equation of the form $P = a_1 + a_2U + a_3U^2 + a_4U^3$using the method of least squares

The function calculates the difference between the experimental and theoretical values.

Editor - P:\DeryushevaVN\MatLabR2014b\MM_matlab\Pr6_1.m

Pr6_1.m | Pr6_2.m

```
%Input the observation times and responses.
xdata=[1.32 1.40 1.50 1.62 1.70 1.80 1.90...
2.00,2.11,2.20,2.32,2.40,2.51];

ydata=[3.30 3.50 3.85 4.25 4.50 4.85 5.40...
6.00 6.60 7.30 9.20 10.20 13.50];

%Create a model.
fun=@(a,xdata)a(1)-a(2)*xdata-a(3)*xdata.^2-a(4)*xdata.^3;

%the vector of initial approximations
a0=[0;0;0;0];
% Solution of the problem
[a,err]=lsqcurvefit(fun,a0,xdata,ydata);

%The geometric interpretation of the problem
t=1.32:0.01:2.51;
Ptc=a(1)-a(2)*t-a(3)*t.^2-a(4)*t.^3;
plot(xdata,ydata,'^',t,Ptc,'k');
grid on;
```

Command Window

```
a =

  -51.5767
  -95.5947
   55.6953
  -11.1115

>> err

err =

    0.5288
```

Thus, as a result of the function datafit it was selected analytical dependence of the form

$$P = -51.577 - 95.595U + 55.695U^2 - 11.111U^3$$

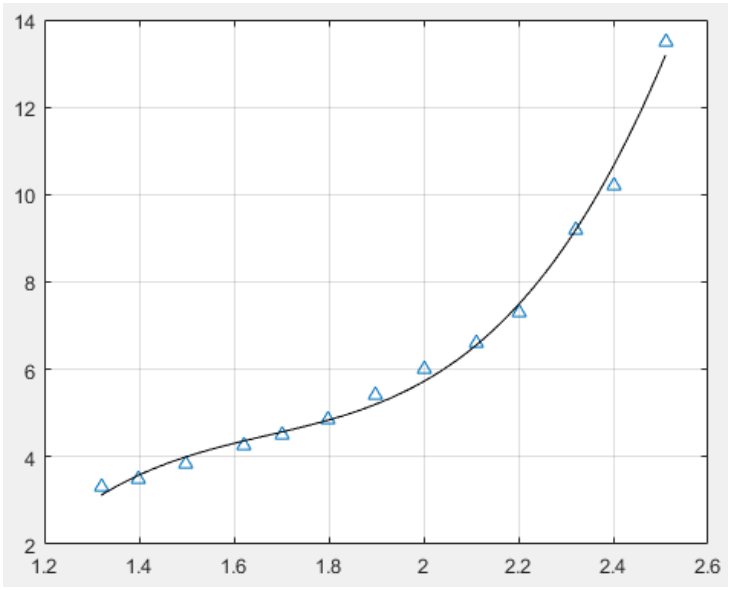
, and the sum of squared deviations of the measured values of the calculated amounted to 0,5288.

The geometric interpretation of the problem

Editor - P:\DeryushevaVN\MatLabR2014b\MM_matlab\Pr6_1.m

Pr6_1.m

```
%Input the observation times and responses.
xdata=[1.32 1.40 1.50 1.62 1.70 1.80 1.90...
2.00,2.11,2.20,2.32,2.40,2.51];

ydata=[3.30 3.50 3.85 4.25 4.50 4.85 5.40...
6.00 6.60 7.30 9.20 10.20 13.50];

%Create a model.
fun=@(a,xdata)a(1)-a(2)*xdata-a(3)*xdata.^2-a(4)*xdata.^3;

%the vector of initial approximations
a0=[0;0;0;0];
% Solution of the problem
[a,err]=lsqcurvefit(fun,a0,xdata,ydata);

%The geometric interpretation of the problem
t=1.32:0.01:2.51;
Ptc=a(1)-a(2)*t-a(3)*t.^2-a(4)*t.^3;
plot(xdata,ydata,'^',t,Ptc,'k');
grid on;
```



Problem 6.2.

Some experimental data choose the dependence of the form

$$Y = a_1 \cdot x^{a_2} + a_3$$

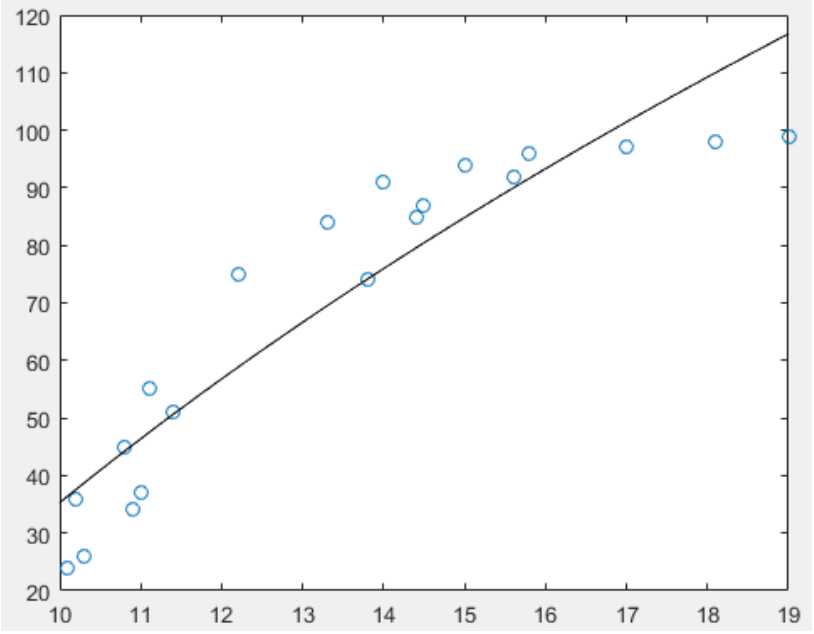
Table 6.3. The experimental data

10.1	10.2	10.3	10.8	10.9	11	11.1	11.4	12.2	13.3	13.8	14	14.4	14.5	15	15.6	15.8	17	18.1	19
24	36	26	45	34	37	55	51	75	84	74	91	85	87	94	92	96	97	98	99

Editor - P:\DeryushevaVN\MatLabR2014b\MM_matlab\Pr6_2.m

Pr6_2.m

```
%Input the observation times and responses.
xdata=[10.1,10.2,10.3,10.8,10.9,11,11.1,11.4,12.2,...
    13.3,13.8,14,14.4,14.5,15,15.6,15.8,17,18.1,19];
ydata=[24,36,26,45,34,37,55,51,75,84,74,91,85,87,...
    94,92,96,97,98,99];

%Create a model.
fun=@(a,xdata)a(1)*xdata.^a(2)+a(3);

%the vector of initial approximations
a0=[0;0;0;0];
% Solution of the problem
[a,err]=lsqcurvefit(fun,a0,xdata,ydata);

%The geometric interpretation of the problem
t=10:0.01:19;
Yt=a(1)*t.^a(2)+a(3);
plot(xdata,ydata,'o',t,Yt,'k');
```



One of the most frequently used method of least squares is a direct function described by the equation $y = a_1 + a_2x$, called the *regression line* of y on x. Parameters a_1 and a_2 are the *coefficients of the regression*. The indicator on the closeness of the linear relationship between x and y, called the *correlation coefficient* is calculated using the formula:

$$r = \frac{\sum_{i=1}^{n}(x_i - M_x)\cdot(y_i - M_y)}{\sqrt{\sum_{i=1}^{n}(x_i - M_x)^2 \sum_{i=1}^{n}(y_i - M_y)^2}}, \qquad M_x = \frac{\sum_{i=1}^{n} x_i}{n}, \qquad M_y = \frac{\sum_{i=1}^{n} y_i}{n}. \qquad (6.2)$$

The value of the correlation coefficient satisfies -1 ≤r ≤1. The smaller the absolute value of r is different from unity, the closer to the line regression of the experimental points are located. If the

coefficient of correlation close to zero, this means that between x and y is not linear link, but there may be other, non-linear dependence.

The analog of the correlation coefficient r for a nonlinear dependence are is the *index of the correlation* calculated by the formula:

$$R = \sqrt{1 - \frac{\sum_{i=1}^{n}(y_i - Y_i)^2}{\sum_{i=1}^{n}(y_i - M_y)^2}}. \qquad (6.3)$$

where y – the experimental values, Y – are the values that have been found by the method of least squares, M_y – average value of y. The index correlation absolute value ranges from 0 to 1. When the functional dependence of the correlation index is 1. If there is no connection then $R = 0$. If the correlation coefficient r is a measure of the closeness of the connection only for a linear form, the index of correlation R is for linear and curvilinear. With regard rectilinear correlation coefficient in absolute value equal to the index of correlation.

For the calculation of the regression coefficients in Matlab intended function

b = **regress**(y,X)

returns a p-by-1 vector b of coefficient estimates for a multilinear regression of the responses in y on the predictors in X. X is an n-by-p matrix of p predictors at each of n observations. y is an n-by-1 vector of observed responses.

regress treats NaNs in X or y as missing values, and ignores them.

If the columns of X are linearly dependent, regress obtains a basic solution by setting the maximum number of elements of b to zero

Problem 6.3.

In "Principles of Chemistry" by D.I. Mendeleev presents data on solubility of sodium nitrate NaNO3 depending on water temperature. The number of conventional parts of NaNO3, dissolved in 100 parts of water at appropriate temperatures, is shown in Table. 6.3. It is required to determine the solubility of sodium nitrate at a temperature of 32 degrees in the case of linear dependence and to find the correlation coefficient and the index.

Table 6.3. Solubility data $NaNO_3$ depending on water temperature.

0	4	10	15	21	29	36	51	68
66.7	71.0	76.3	80.6	85.7	92.9	99.4	113.6	125.1

Solution problem

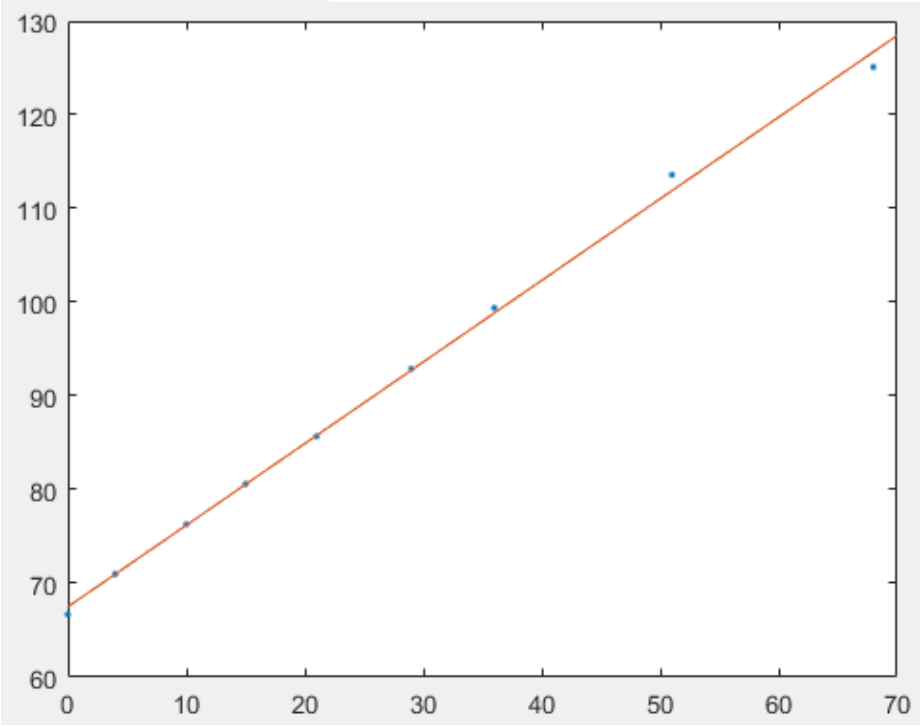
Editor - P:\DeryushevaVN\MatLabR2014b\MM_matlab\Pr6_3.m

Pr6_2.m | Pr6_3.m | exam6_1.m

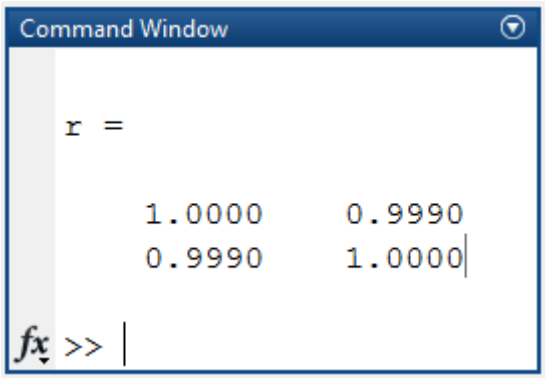
```
clear;
clc;
% data
x=[0 4 10 15 21 29 36 51 68];
y=[66.7 71 76.3 80.6 85.7 92.9 99.4 113.6 125.1];

%The calculation of the regression coefficients
%[b,bit]=regress(y,x,0.05);

% attach ones otherwise woud skip the first beta
X=[ones(size(x')) x'];
beta=regress(y',X)

% for temperature t=32
t=32; t32=beta(1)+beta(2)*t

% The correlation coefficient (6.2)
r=sum((x-mean(x)).*(y-mean(y)))/...
    sqrt(sum((x-mean(x)).^2)*sum((y-mean(y)).^2))

%The index of correlation (6.3)
R=sqrt(1-sum((y-(beta(1)+beta(2)*x)).^2)/...
    sum((y-mean(y)).^2))

%Plotting experimental data and the regression line
t=0:70; Yt=beta(1)+beta(2)*t;
plot(x,y,'.', t,Yt);
```

Command Window

```
beta =

   67.5078
    0.8706


t32 =

   95.3683


r =

    0.9990


R =

    0.9990
```



For the calculation of the correlation coefficient in Matlab is also built-in **rc**=**corrcoef**(x,y), where x and y - the experimental data.



Interpolation of functions

The simplest problem of interpolation is as follows. On the segment [a; b] given point x_0; x_1; x_2;…; x_n (total n + 1 point), called interpolation nodes and the values of a function *f (x)* at these points:

$$f(x_0) = y_0, \quad f(x_1) = y_1, \quad f(x_2) = y_2, \quad \dots, \quad f(x_n) = y_n. \qquad (6.4)$$

Required to construct interpolation function F (x), belongs to a known class and accepted in the interpolation nodes are the same values as *f (x):*

$$F(x_0) = y_0, \quad F(x_1) = y_1, \quad F(x_2) = y_2, \quad \dots, \quad F(x_n) = y_n. \qquad (6.5)$$

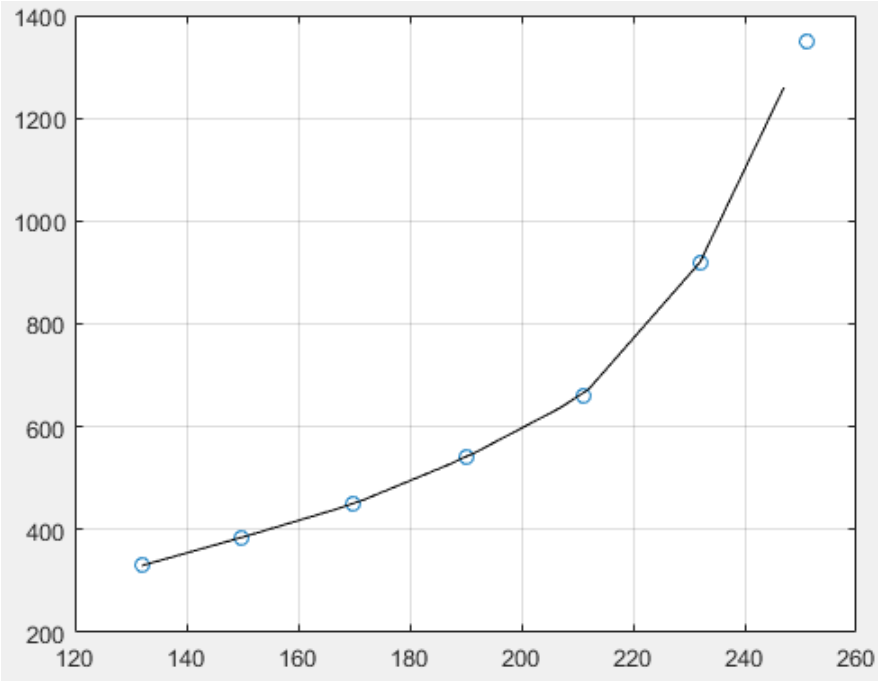
To solve this problem quite often use the spline interpolation. One of the most common variants of interpolation cubic spline interpolation. In addition, there are linear and quadratic spline.

In Matlab, to construct a linear interpolation function is

vq=**interp1**(x,v,xq)

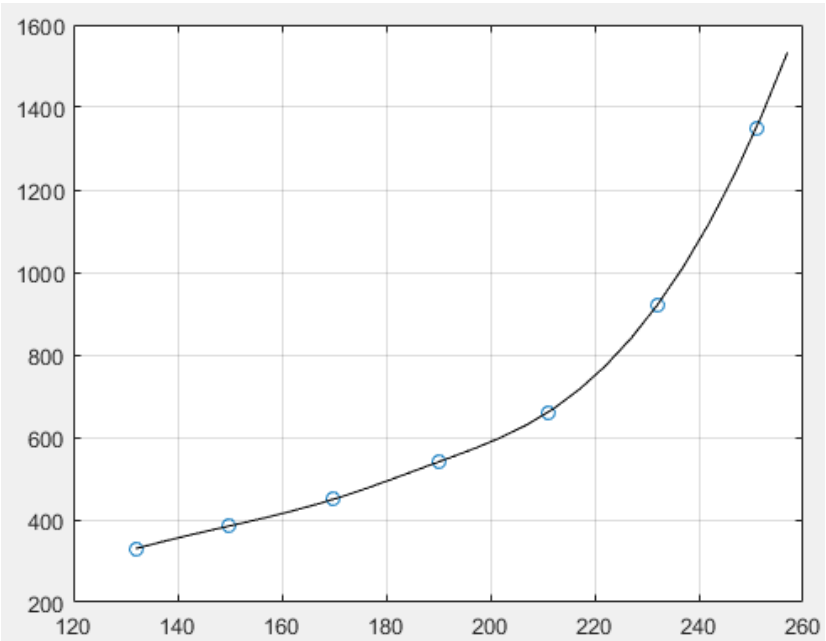
returns interpolated values of a 1-D function at specific query points using linear interpolation. Vector x contains the sample points, and v contains the corresponding values, v(x). Vector xq contains the coordinates of the query points.

If you have multiple sets of data that are sampled at the same point coordinates, then you can pass v as an array. Each column of array v contains a different set of 1-D sample values. We use command **interp1** for problem 6.1.

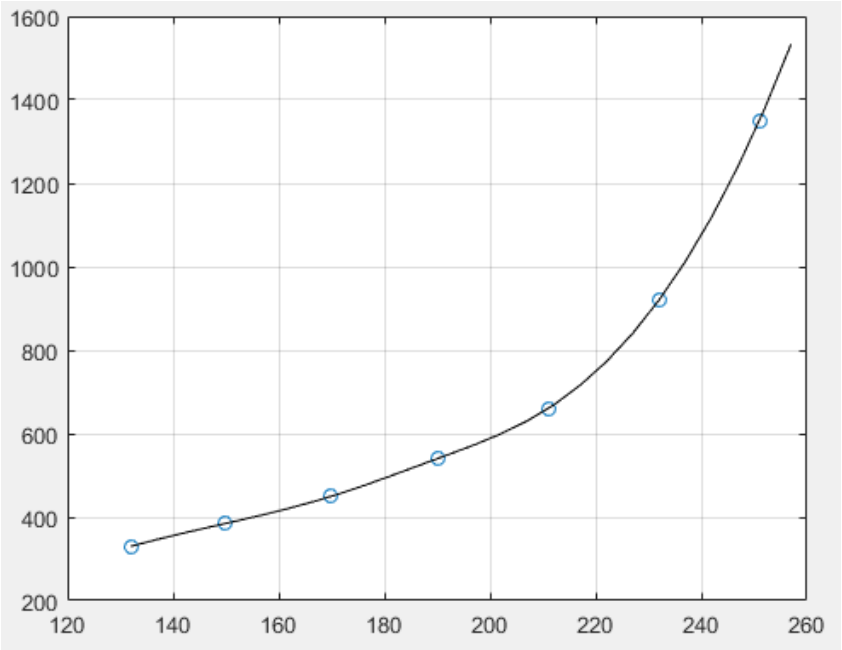


There are two ways for building a cubic spline in Matlab: first, using the **vq = interp1(x,v,xq,method)**, or using **yy = spline(x,Y,xx)**.

> **vq = interp1(x,v,xq,method)** specifies a string for choosing an alternative interpolation method: 'nearest', 'next', 'previous', 'linear','spline','pchip', or 'cubic'. The default method is 'linear'.
>
> **yy = spline(x,Y,xx)** uses a cubic spline interpolation to find yy, the values of the underlying function Y at the values of the interpolant xx. For the interpolation, the independent variable is assumed to be the final dimension of Y with the breakpoints defined by x. The values in x must be distinct.




```
ptd=interp1(x,y,t,'spline');
```




```
ptd=spline(x,y,t);
```

Problem 6.5.

Find the approximate value of the function at a given value of the argument using a cubic spline interpolation at x = 0; 702, x_2 = 0; 512, x_3 = 0; 608. The function given in tabular form (Table 6.5.).

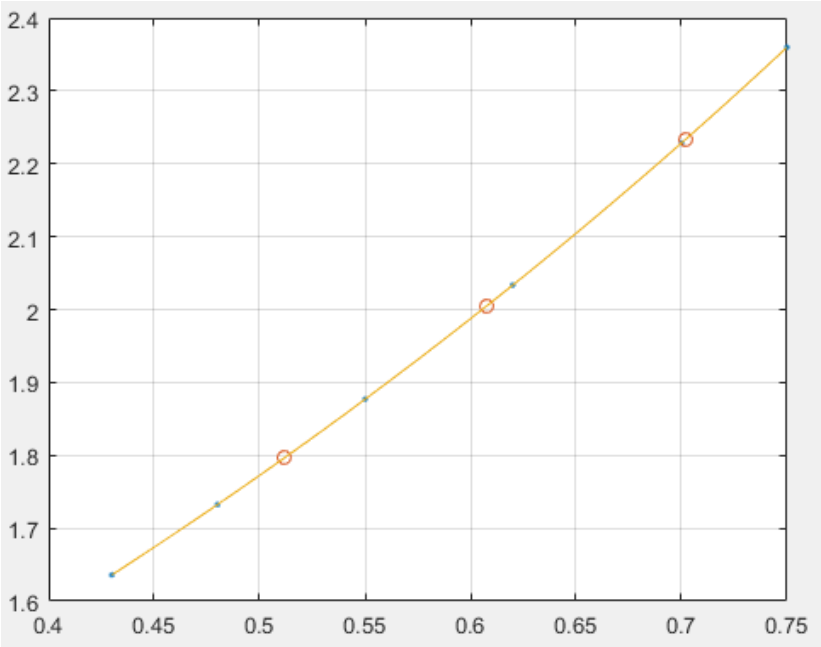
Table 6.5. Input data to the problem 6.5

0.43	0.48	0.55	0.62	0.7	0.75
1.63597	1.73234	1.87686	2.03345	2.22846	2.35973

Editor - P:\DeryushevaVN\MatLabR2014b\MM_matlab\Pr6_5.m

Pr6_2.m | Pr6_3.m | exam6_1.m | exam6_2.m | Pr6_4.m | Pr6_5.m

```
1 -    x=[0.43 0.48 0.55 0.62 0.7 0.75];
2 -    y=[1.63597 1.73234 1.87686 2.03345 2.22846 2.35973];
3
4 -    X=[0.702 0.512 0.608];
5      %The value of the function at the given points
6 -    Y=interp1(x,y,X,'splin');
7      %Building a cubic spline
8 -    t=0.43:0.01:0.75;
9 -    ptd=interp1(x,y,t, 'spline');
10
11 -   plot(x,y,'.',X,Y,'o',t,ptd);
12 -   grid on;
```


2.4
2.3
2.2
2.1
2
1.9
1.8
1.7
1.6
0.4
0.45
0.5
0.55
0.6
0.65
0.7
0.75