ПРИМЕРИ
Интерполационни полиноми и сплайн-интерполация – използване на функцията interp1

Задача 1: За функцията $y = f(x)$ са дадени следните възли и стойности в тях:

x_i	0	2	4	6	8	10
y_i	0	0.9	-0.76	-0.28	0.99	-0.54

При зададените данни да се построят интерполационните полиноми при линейна и кубична интерполация, както и кубичният сплайн. Да се намери приближената стойност на функцията за $x = 1.3$ при всяка от тези три интерполации. Коя измежду получените три стойности е с най-ниска точност?

Решение:
Първо да въведем данните:

```
>> x=0:2:10;
>> y=[0,0.9,-0.76,-0.28,0.99,-0.54];
```

Алтернативно, можехме да въведем x=[0,2,4,6,8,10] (т.е. посредством изброяване), но тук се възползваме от факта, че възлите са равноотдалечени.

Избираме достатъчно гъсто множество от точки върху зададения интервал [0,10], върху което ще изчертаваме получаваните интерполационни полиноми. В случая точките сме избрали със стъпка *0.25*.

```
>> xx=0:0.25:10;
```

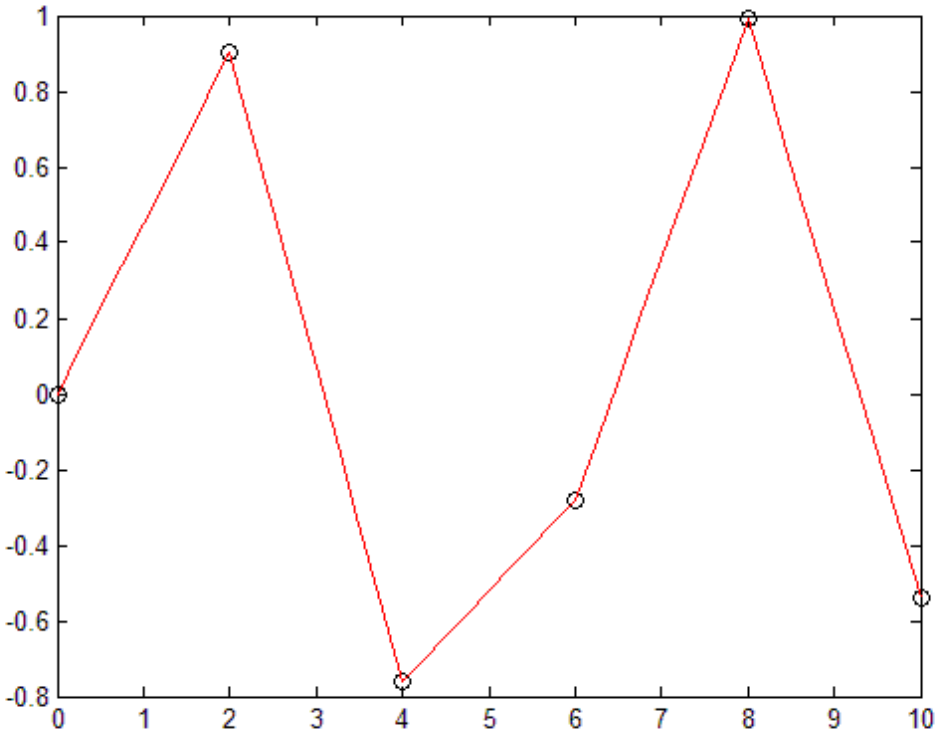
Линейна интерполация:

```
>> yy1=interp1(x,y,xx);
```

Алтернативно, по-горе можехме да зададем yy1=interp1(x,y,xx,'linear'). Т.е. последната опция на функцията interp1 за едномерна интерполация в Matlab е задаване на метод за интерполация, който по default е 'linear'.

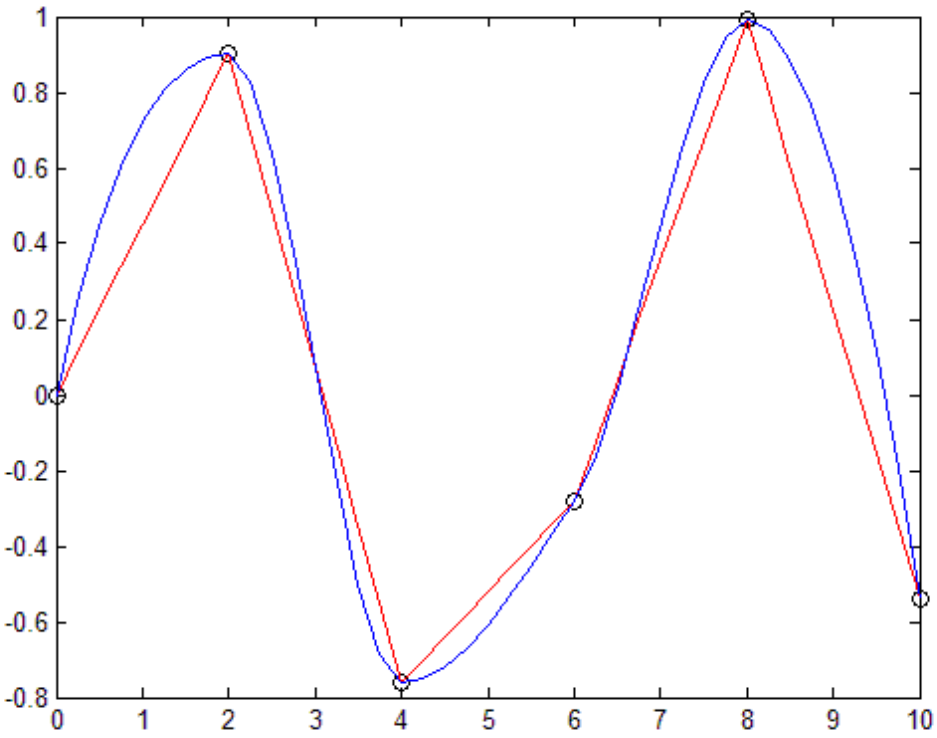
```
>> plot(x,y,'ok',xx,yy1,'r')
>> hold on
```

Получаваме следната графика на по части линейния интерполационен полином, като впоследствие върху същата координатна система ще изчертаем и другите два интерполационни полинома (така че hold on ☺):



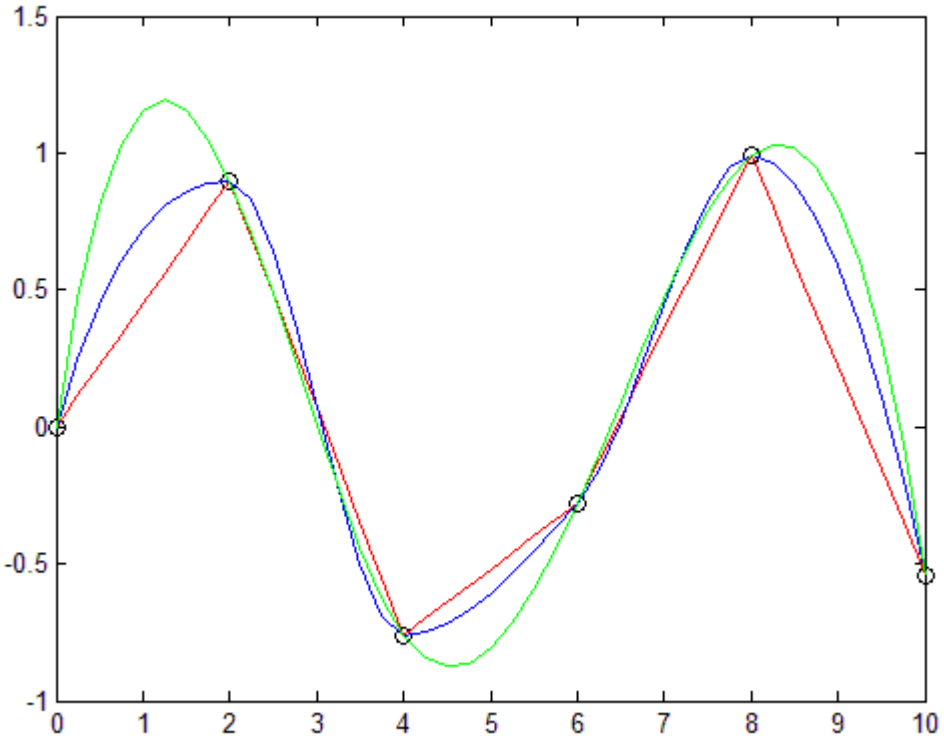
Да получим и изчертаем по части кубична интерполация, тук задаваме като метод 'cubic', а същия резултат се получава и ако зададем 'pchip' (Piecewise Cubic Hermite Interpolation Polynomial):

```
>> yy2 = interp1(x,y,xx,'cubic');
>> plot(xx,yy2,'b')
```



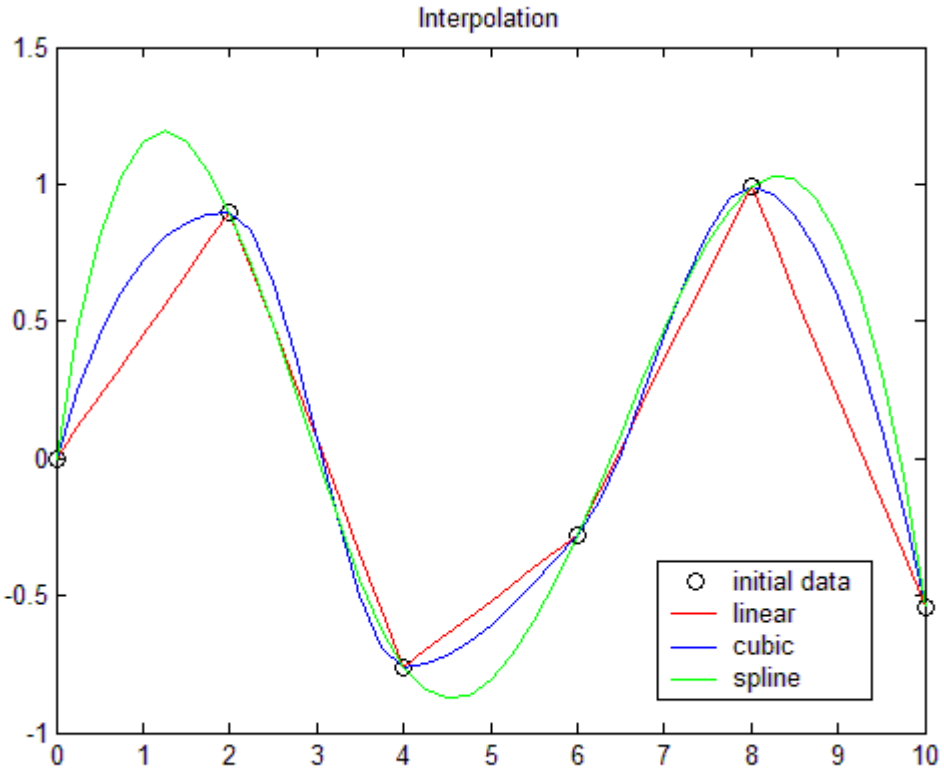
А най-накрая и кубичен сплайн – задаваме метод 'spline', който реализира кубична сплайн-интерполация:

```
>> yy3=interp1(x,y,xx,'spline');
>> plot(xx,yy3,'g')
```



Можем да поставим надпис на фигурата, както и легенда:

```
>> title('Interpolation')
>> legend('initial data','linear','cubic','spline')
```



Може да се постави и мрежа (grid on!) за по-лесно разчитане на графичните резултати.

Да намерим приближената стойност на апроксимираната функция при $x = 1.3$ по всеки от трите метода:

```
>> interp1(x,y,1.3)

ans =

   0.5850

>> interp1(x,y,1.3,'cubic')

ans =

   0.8200

>> interp1(x,y,1.3,'spline')

ans =

   1.1917
```

В заключение, най-ниска точност измежду използваните методи дава по части линейната интерполация. При другите два метода оценката на грешката е от един и същ, значително по-висок порядък, което ги прави предпочитани пред линейната интерполация.

Сплайн-интерполация и кубична интерполация – използване на функциите spline и pchip

Задача 2: За функцията $y = f(x)$ са дадени следните възли и стойности в тях:

x_i	-4	-3	-2	-1	0	1	2	3	4
y_i	0.1	0.25	1.2	2.41	2.39	1.5	0.45	0.3	0.15

При зададените данни да се построи кубична сплайн-интерполация. Да се намери приближената стойност на функцията за $x = -0.4$ и $x = 3.8$.

Решение:
(Първо clear all предвид това, че вече сме работили по Задача 1). Въвеждаме данните:

```
>> x=-4:4;
>> y=[0.1,0.25,1.2,2.41,2.39,1.5,0.45,0.3,0.15];
```

Когато стъпката при равноотдалечени възли е 1, то можем да не задаваме *x = -4:1:4*, а *x = -4:4*.

Кубичната сплайн-интерполация е реализирана в Matlab посредством функцията spline.

Разликата на spline с interp1, използвана при зададен метод 'spline' е тази, че докато interp1 пресмятаме при предварително въведени начални данни x и y върху отнапред фиксирано множество xx от стойности на променливата (xx е число или последователност от числа, т.е. вектор) – например interp1(x,y,1.3,'spline'), interp1(x,y,[0.2,1.3,3.6],'spline'), interp1(x,y,xx,'spline') и т.н., то при функцията spline въвеждаме само начални данни x и y, без предварително да определяме върху какво множество от стойности на променливата ще получаваме резултат:

```
>> cs=spline(x,y);
```

Можем евентуално да зададем наклона (slope) на сплайн-функцията в крайните точки да бъде 0 (допирателната в тях е хоризонтална!):

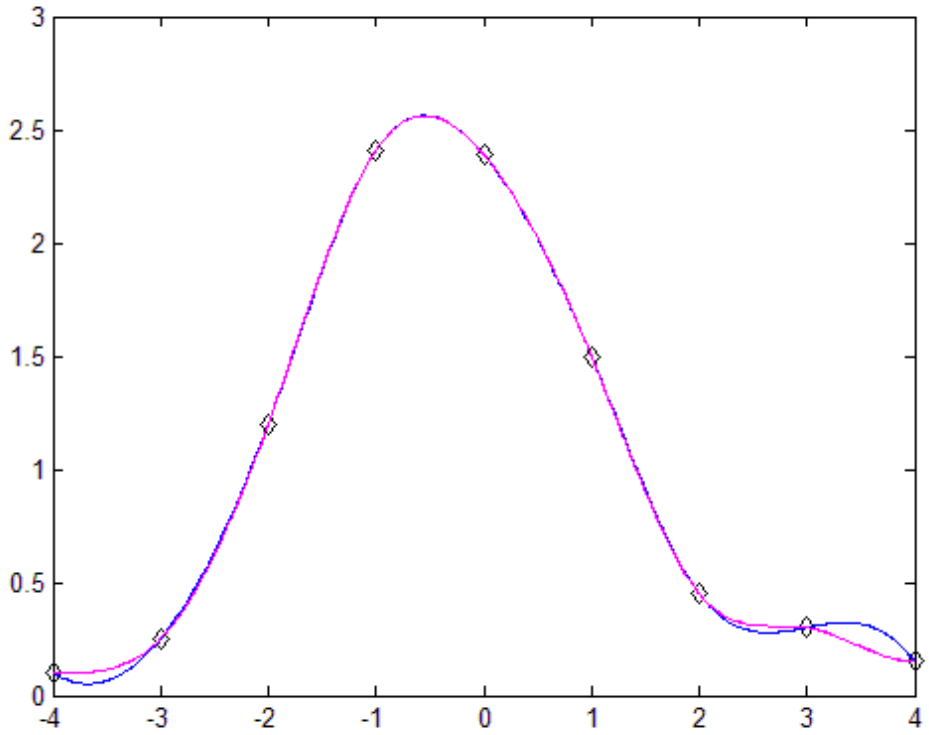
```
>> cs0=spline(x,[0,y,0]);
```

Двете разновидности на сплайн-функцията са получени за всевъзможни стойности на променливата вътре в интервала (интерполация), а и извън него (екстраполация). За конкретни стойности на променливата, а в случая за дадената задача се изисква за *-0.4* и *3.8*, стойностите на сплайн-функцията се извличат посредством функцията ppval:

```
>> ppval(cs,[-0.4,3.8])

ans =

   2.5476    0.2498

>> ppval(cs0,[-0.4,3.8])

ans =

   2.5463    0.1631
```

За да изчертаем двете получени разновидности на кубичен сплайн, дефинираме вектор от точки, достатъчно гъсто разположени. По-долният ред е еквивалентен на *xx=-4:0.04:4* – тук задаваме крайните точки на интервала *-4* и *4* и стъпката *0.04*, докато по-долу се задават крайните точки на интервала и броят на равноотдалечените точки (вкл. крайните):

```
>> xx=linspace(-4,4,201);
>> plot(x,y,'dk',xx,ppval(cs,xx),'b',xx,ppval(cs0,xx),'m')
```



Обърнете внимание на разликата в наклона на двете сплайн-графики в крайните точки! Графиката в цвят 'm' (т.е. magenta) е със зададения нулев наклон в краищата.

Задача 3: За функцията *y = f(x)* от предходната задача да се построи по части кубичен интерполационен полином, да се намери приближената стойност за *x = -0.4* и *x =3.8.* Резултатите да се сравнят аналитично и графично с кубичната сплайн-интерполация, получена в предходната задача

Решение:
Продължаваме работата от предходната задача в средата на Matlab.

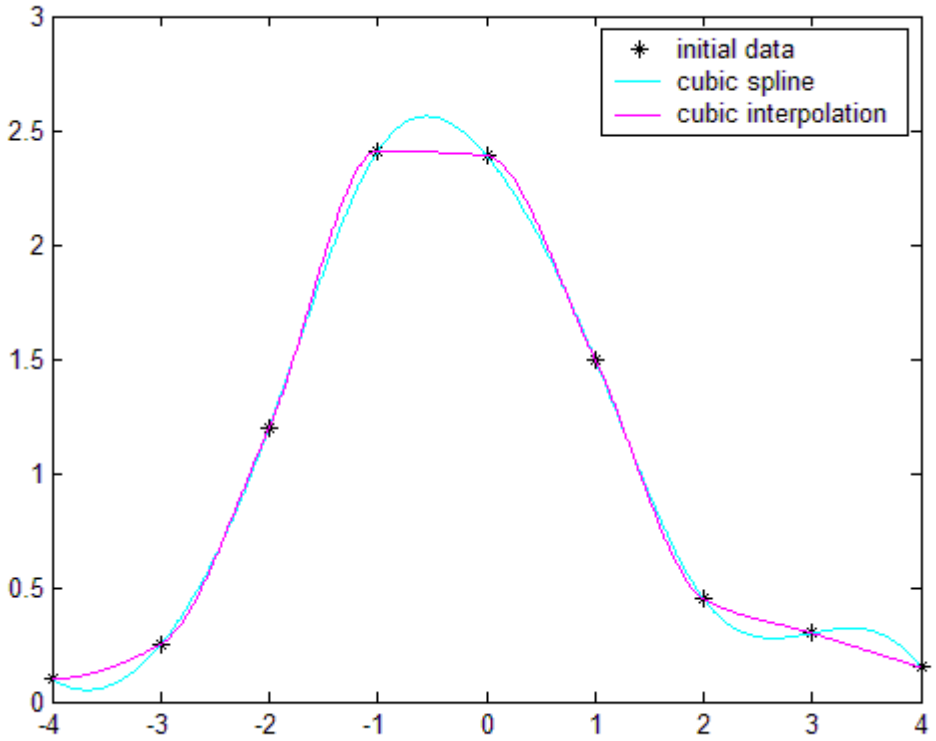
За построяване на кубичен интерполационен полином, наред с възможността да бъде получаван посредством функцията interp1 при задаване на метод 'cubic' или метод 'pchip', в Matlab съществува и отделна функция pchip.

Съответствието и разликите във възможностите и употребата между функцията pchip и функцията interp1 със зададен метод 'cubic' или 'pchip' е същото, както между функцията spline и функцията interp1 със зададен метод 'spline'.

```
>> ph=pchip(x,y);
>> ppval(ph,[-0.4,3.8])

ans =

    2.4027    0.1800
```

Да изчистим прозореца на фигурата, за да построим нова графика (сравнение между получения в предната задача сплайн и получения току-що по части кубичен интерполационен полином):

```
>> clf
>> plot(x,y,'*k',xx,ppval(cs,xx),'c',xx,ppval(ph,xx),'m')
>> legend('initial data','cubic spline','cubic interpolation')
```



Метод на най-малките квадрати

Задача 4:

Пример. В долната таблица са дадени данни от измерване на генерираната изходна мощност y_i при различни стойности на захранващата входна електрическа мощност x_i.

x_i, KW	2	2.3	2.5	3	3.3	3.5	4	4.5	5
y_i, W	40	42	60	68	100	90	100	106	120

а) Намерете полином от първа степен, който най-добре приближава тези експериментални данни. Единствен ли е полиномът?
б) Намерете полином от втора степен за същите данни.
в) А какво всъщност представлява полиномът от осма степен за тези данни?

г) Сравнете графично резултатите от а) и б).
д) Прогнозирайте каква мощност може да очакваме при x=3.8 и x=5.5?

Решение:
Въвеждаме данните (възлите не са равноотдалечени, въвеждаме ги чрез изброяване!):

```
>> clear all
>> x=[2,2.3,2.5,3,3.3,3.5,4,4.5,5];
>> y=[40,42,60,68,100,90,100,106,120];
```

Да получим полинома от I степен, приближаващ зададените експериментални данни по метод на най-малките квадрати – последната опция във функцията polyfit е естествено число, задаващо степента на търсения полином (в случая 1):

```
>> p1=polyfit(x,y,1)

p1 =

  27.1759  -10.2216
```

Изведените числа са коефициентите на търсения полином, т.е. той е

$$P_1(x)=27.1759x-10.2216.$$

Аналогично получаваме полинома от II степен, който приближава експерименталните данни по метод на най-малките квадрати:

```
>> p2=polyfit(x,y,2)

p2 =

  -6.3737   71.4699  -81.2179
```

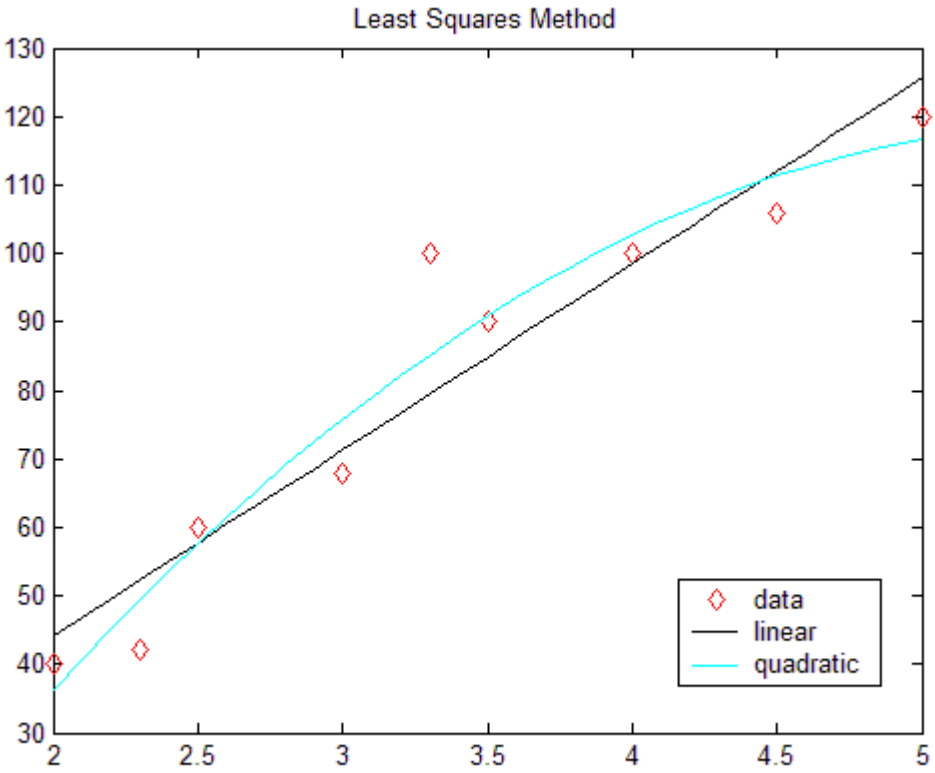
Съответно, полиномът от II степен е

$$P_2(x)=-6.3737x^2+71.4699x-81.2179.$$

Да сравним графично двата получени полинома. Избираме множество от точки в интервала *[2,5]* със стъпка *0.1*. Функцията polyval в Matlab служи за извличане на стойности на вече намерени апроксимиращи полиноми (в случая сме ги именували като *p1* и *p2* върху предварително зададено множество от стойности за променливата (в случая върху *xx*, защото ще изчертаваме!). По този начин *pp1* и *pp2* са вектори с размерност, съвпадаща с размерността на xx и съдържат стойностите на полиномите $P_1(x)$ и $P_2(x)$, когато x пробягва множеството от стойности на вектора *xx*.

```
>> xx=2:0.1:5;
>> pp1=polyval(p1,xx);
>> pp2=polyval(p2,xx);
```

```
>> plot(x,y,'dr',xx,pp1,'k',xx,pp2,'c')
>> legend('data','linear','quadratic')
>> title('Least Squares Method')
```



Остава да пресметнем приближените стойности при x=3.8 и x=5.5:

При приближаване с линейна функция:

```
>> polyval(p1,[3.8,5.5])

ans =

   93.0468  139.2458
```

При приближаване с квадратна функция:

```
>> polyval(p2,[3.8,5.5])

ans =

   98.3321  119.0634
```

Забележка: може и поотделно, т.е. например

```
>> polyval(p2,3.8)

ans =

   98.3321

>> polyval(p2,5.5)
```

```
ans =

  119.0634
```