```
In [1]: %plot inline -w 700 -h 700
function plot_points(x,y, x_legend, y_legend)
    plot(x,y,'ob','markersize', 5,'markerfacecolor','b')
    xlabel(x_legend);
    ylabel(y_legend);
    set(gca,'fontsize', 16);
    axis([min(x)-10,max(x)+10,min(y)-10 max(y)+10]);
    %set(gca, 'XTick',0:max(x)+2);
endfunction
```

```
In [2]: function p=lsquares(x,y,stepen)
    n=length(x);
    A=zeros(n,stepen+1);
    for i=1:n
        for j=1:stepen+1
            A(i,j)=x(i).^(j-1);
        end
    end
p=(A'*A)\(A'*y');
p=fliplr(p');
endfunction
```

Linearna regresija

Linearna regresija predstavlja alternativan način za pronalaženje trenda u podacima u odnosu na interpolaciju.

Videli smo da jedan interpolacioni polinom nije dobro rešenje kada imamo veliku količinu podataka. Na današnjem predavanju pokazaćemo da ni splajnovi nisu dobro rešenje u tom slučaju.

Uvodna napomena oko terminologije:

Termin "linearna" ne znači da je rezultat regresije prava odnosno linearna funkcija, već da je rezultat regresije linearna kombinacija koeficijenata i nekih funkcija. Funkcije se zadaju, a koeficijenti određuju.

Na primer, prava je linearna kombinacija koeficijenata k i n i fukcija $f(x)=x$ i $f(x)=1$, dok je polinom drugog stepena linearna kombinacija koeficijenata a, b i c i fukcija $f(x)=x^2, f(x)=x$ i $f(x)=1$.

Motivacioni primer

Kao jedan od primera upotrebe regresije uzećemo predikciju ishoda utakmica u Premier ligi.

Tim koji stoji iza sajta http://www.football-data.co.uk/ , želeo je da napravi model za "fer" kvote za utakmice. "Fer" znači da nemaju cilj da zarde od tuđeg klađenja (kao kladionice).

Krenuli su sa jednostavnim pristupom, da je dobar indikator kvaliteta tima razlika u golovima (RG)= broj datih golova - broj primljenih golova na prethodno odigranim utakmicama. Odabrali su da posmatraju 6 prethodno odigranih utakmica.

Na taj način kreirali su indikator koji su nazvali rejting utakmice (match rating, MR).

MR=RG_domaći_tim-RG_gostujući_tim

Npr. Ako igraju Manchester-Liverpool; Man. ima RG=+5, a Liv. RG=+2, MR=5-2=3.

Da bi kreirali svoj model posmatrali su mečeve odigrane u periodu 1993-2001. Za svaki meč zabeležili su MR i ishod meča.

Nakon toga su podatke organizovali po rejtingu i za svaki rejting izračunali su %pobeda_domaćina (%PD), %pobeda_gosta (%PG), %izjednačeno (%I).

U slećem redu učitavamo njihove podatke iz csv (comma separated values) fajla.

Kolone su redom: MR, Broj pobeda domaćeg, Broj pobeda gostujućeg, Broj izjednačenih, %pobeda_domaćina (%PD), %pobeda_gosta (%PG), %izjednačeno (%I)

```
In [3]: data = csvread('fudbal.csv')
% prvi red su sve 0 jer su to nazivi atributa u csv fajlu.
```

```
data =

 Columns 1 through 6:

     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000
   -26.00000     0.00000     1.00000     1.00000     0.00000    50.00000
   -23.00000     0.00000     0.00000     2.00000     0.00000     0.00000
   -22.00000     0.00000     0.00000     3.00000     0.00000     0.00000
   -21.00000     0.00000     2.00000     4.00000     0.00000    33.30000
   -20.00000     2.00000     2.00000     7.00000    18.20000    18.20000
   -19.00000     1.00000     1.00000     3.00000    20.00000    20.00000
   -18.00000     5.00000     7.00000     9.00000    23.80000    33.30000
   -17.00000     7.00000     9.00000    12.00000    25.00000    32.10000
   -16.00000     6.00000    14.00000    21.00000    14.60000    34.10000
   -15.00000    25.00000    12.00000    19.00000    44.60000    21.40000
   -14.00000    32.00000    30.00000    51.00000    28.30000    26.50000
   -13.00000    43.00000    38.00000    58.00000    30.90000    27.30000
   -12.00000    51.00000    50.00000    64.00000    30.90000    30.30000
   -11.00000    75.00000    54.00000    91.00000    34.10000    24.50000
   -10.00000    84.00000    94.00000    91.00000    31.20000    34.90000
    -9.00000   123.00000    91.00000   112.00000    37.70000    27.90000
    -8.00000   171.00000   113.00000   124.00000    41.90000    27.70000
    -7.00000   190.00000   121.00000   170.00000    39.50000    25.20000
    -6.00000   242.00000   202.00000   191.00000    38.10000    31.80000
    -5.00000   279.00000   212.00000   197.00000    40.60000    30.80000
    -4.00000   293.00000   219.00000   215.00000    40.30000    30.10000
    -3.00000   374.00000   246.00000   229.00000    44.10000    29.00000
    -2.00000   372.00000   233.00000   214.00000    45.40000    28.40000
    -1.00000   375.00000   251.00000   222.00000    44.20000    29.60000
     0.00000   414.00000   259.00000   235.00000    45.60000    28.50000
     1.00000   412.00000   243.00000   212.00000    47.50000    28.00000
     2.00000   401.00000   220.00000   189.00000    49.50000    27.20000
     3.00000   395.00000   224.00000   175.00000    49.70000    28.20000
     4.00000   391.00000   177.00000   137.00000    55.50000    25.10000
     5.00000   297.00000   180.00000   102.00000    51.30000    31.10000
     6.00000   260.00000   146.00000   131.00000    48.40000    27.20000
     7.00000   236.00000    98.00000    83.00000    56.60000    23.50000
     8.00000   197.00000    94.00000    56.00000    56.80000    27.10000
     9.00000   158.00000    86.00000    32.00000    57.20000    31.20000
    10.00000   125.00000    57.00000    42.00000    55.80000    25.40000
    11.00000   113.00000    34.00000    33.00000    62.80000    18.90000
    12.00000    90.00000    30.00000    22.00000    63.40000    21.10000
    13.00000    61.00000    23.00000    17.00000    60.40000    22.80000
    14.00000    48.00000    15.00000    11.00000    64.90000    20.30000
    15.00000    38.00000    21.00000     8.00000    56.70000    31.30000
    16.00000    30.00000     9.00000     2.00000    73.20000    22.00000
    17.00000    20.00000     8.00000     2.00000    66.70000    26.70000
    18.00000    15.00000     1.00000     1.00000    88.20000     5.90000
    19.00000     8.00000     4.00000     1.00000    61.50000    30.80000
    20.00000     5.00000     1.00000     0.00000    83.30000    16.70000
    21.00000     1.00000     0.00000     0.00000   100.00000     0.00000
    22.00000     1.00000     0.00000     1.00000    50.00000     0.00000
    23.00000     1.00000     0.00000     0.00000   100.00000     0.00000
    27.00000     1.00000     0.00000     0.00000   100.00000     0.00000

 Column 7:

     0.00000
    50.00000
   100.00000
   100.00000
    66.70000
    63.60000
    60.00000
    42.90000
    42.90000
    51.20000
    33.90000
    45.10000
    41.70000
    38.80000
    41.40000
    33.80000
    34.40000
    30.40000
    35.30000
    30.10000
    28.60000
    29.60000
    27.00000
    26.10000
    26.20000
    25.90000
    24.50000
    23.30000
    22.00000
    19.40000
    17.60000
    24.40000
    19.90000
    16.10000
    11.60000
    18.80000
    18.30000
    15.50000
    16.80000
    14.90000
    11.90000
     4.90000
     6.70000
     5.90000
     7.70000
     0.00000
     0.00000
    50.00000
     0.00000
     0.00000
```

Nakon toga kreirali su regresione funkcije (modele) za svaki od %, koristeći rejting meča kao x.

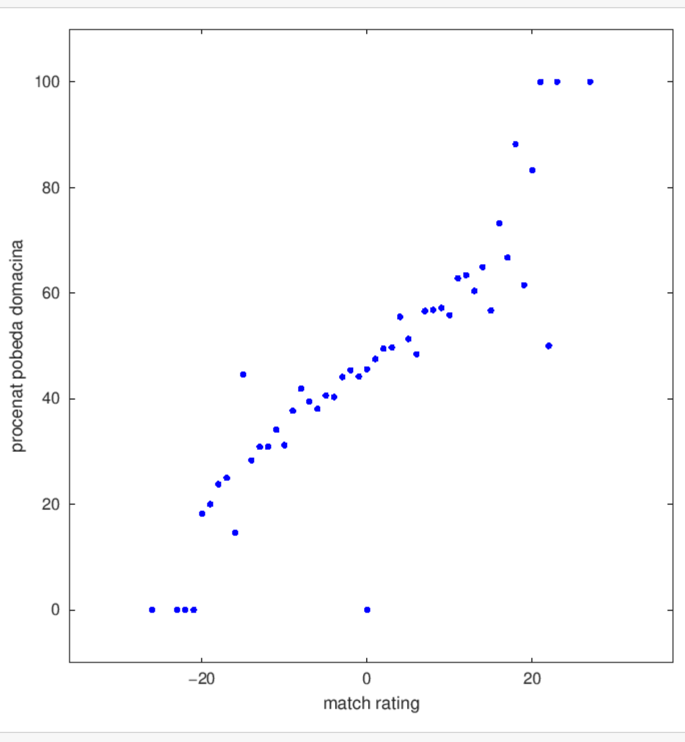
Sa ciljem da, kad je poznat MR imaju funkciju na osnovu koje će da predvide %PD, %PG i %I.

Kada su imali procente jednostavno su 100 podelili sa svakim od njih i tako dobili kvote za PD, PG i I.

Npr. ako bi procenat pobede domaćina bio 46.7%, kvota bi bila 100/46.7=2.15

U nastavku ponavljamo postupak kreiranja regresionih funkcija, a nakon toga objašnjavamo se potrebne teorijske koncepte.

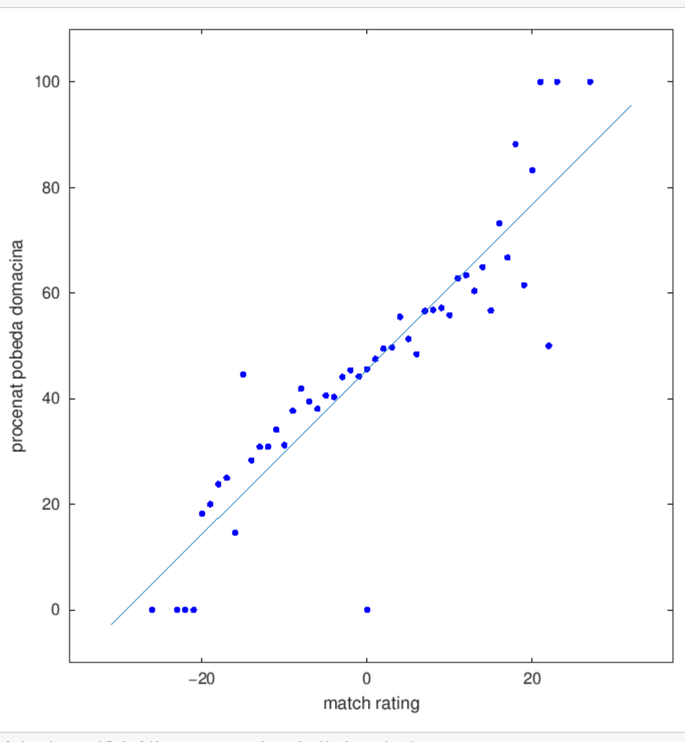
```
In [4]: match_rating = data(:,1)';
procenat_pobeda_domacina = data(:,5)';
plot_points(match_rating,procenat_pobeda_domacina,'match rating','procenat pobeda domacina')
```




```
In [5]: x=match_rating;
y=procenat_pobeda_domacina;
p=lsquares(x, y, 1)
polyout(p,'x')
```

```
p =

    1.5616   45.5368

1.5616*x^1 + 45.537
```

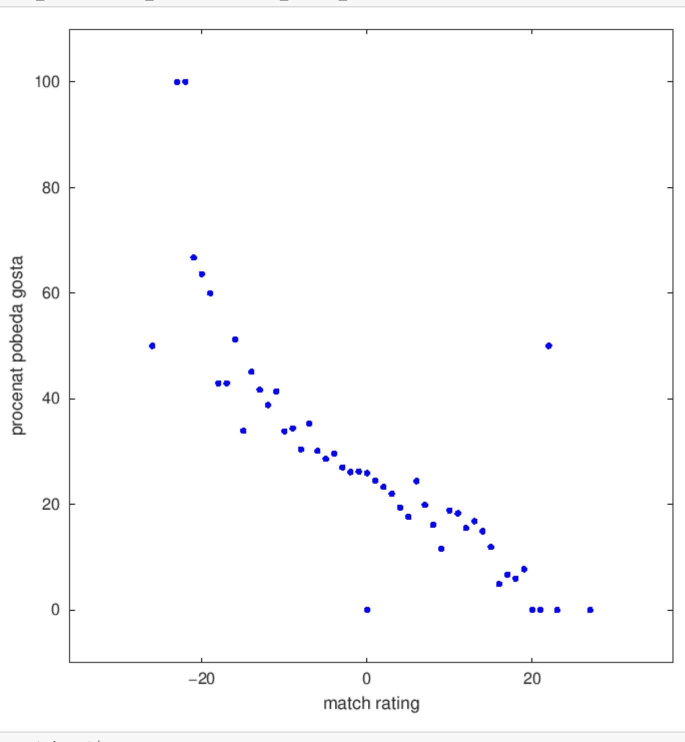
```
In [6]: plot_points(x,y,'match rating','procenat pobeda domacina');
hold on;
xp=linspace(min(x)-5,max(x)+5,100);
plot(xp,polyval(p,xp))
```




```
In [7]: %ako je mr 15 kolike su sanse da pobedi domaci tim.
polyval(p,15)
```

```
ans =  68.960
```

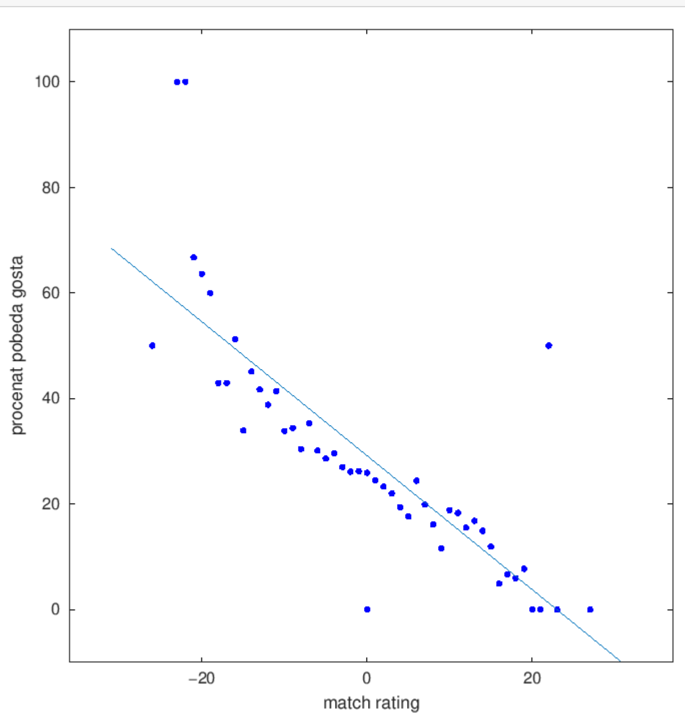
```
In [8]: match_rating = data(:,1)';
procenat_pobeda_gosta = data(:,7)';
plot_points(match_rating,procenat_pobeda_gosta,'match rating','procenat pobeda gosta')
```



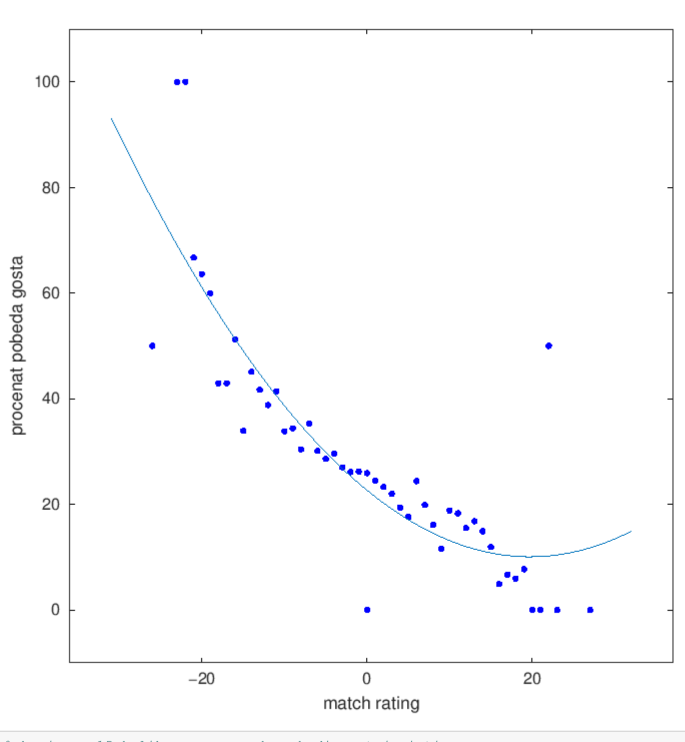

```
In [9]: x=match_rating;
y=procenat_pobeda_gosta;
p=lsquares(x, y, 1)
polyout(p,'x')
```

```
p =

   -1.2690   29.1414

-1.269*x^1 + 29.141
```

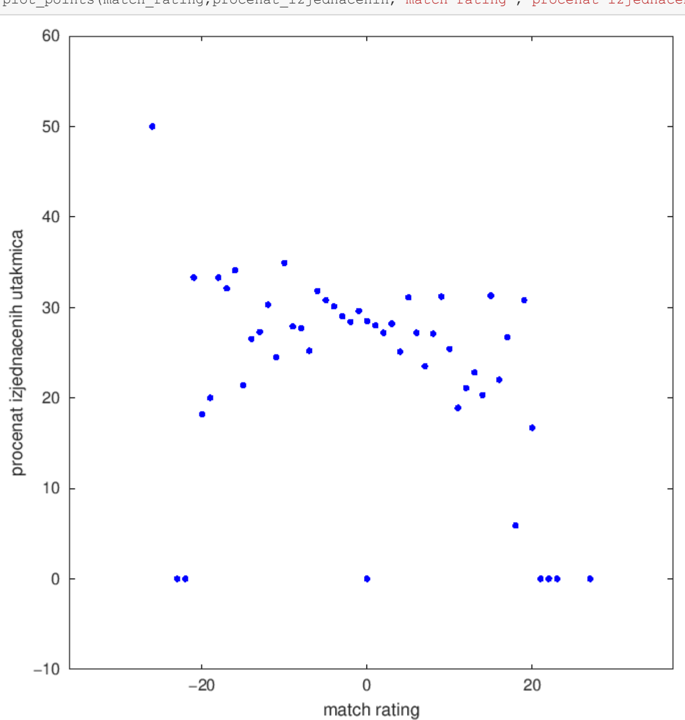
```
In [10]: plot_points(x,y,'match rating','procenat pobeda gosta');
hold on;
xp=linspace(min(x)-5,max(x)+5,100);
plot(xp,polyval(p,xp))
```



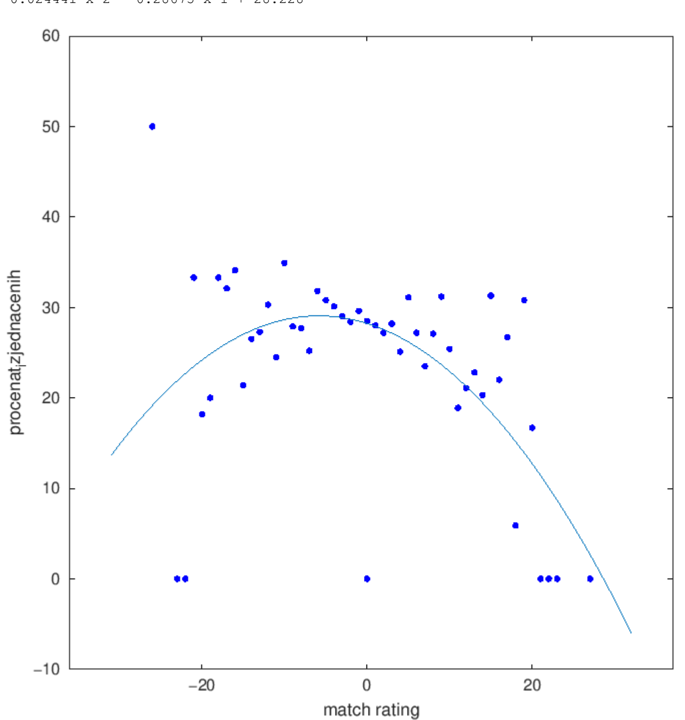
[illegible]



[illegible]

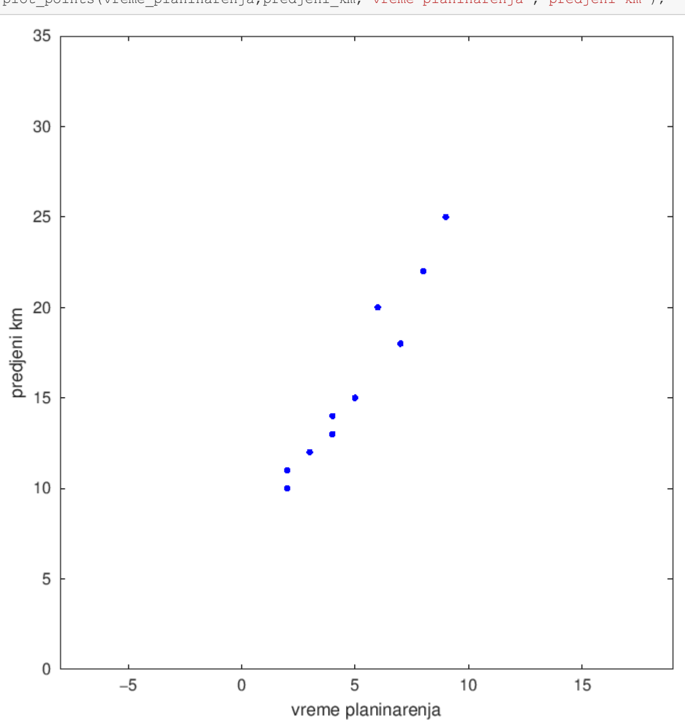


[illegible]



Algoritam linearne regresije

[illegible]



[illegible]

$$kx_1+n=y_1$$
$$kx_2+n=y_2$$
$$kx_3+n=y_3$$
$$kx_4+n=y_4$$
$$kx_5+n=y_5$$
$$kx_6+n=y_6$$
$$kx_7+n=y_7$$
$$kx_8+n=y_8$$
$$kx_9+n=y_9$$
$$kx_{10}+n=y_{10}$$

Dakle, ne možemo da odredimo pravu liniju koja će proći kroz svih 10 tačaka jer prava nije dovoljno kompleksan model da može da obuhavati sve tačke (osim ako baš tačke nisu generisane pomoću prave).

U ovom slučaju dovoljno kompleksan model koji bi prošao kroz sve tačke je polinom devetog stepena. Hajde da nacrtamo taj polinom.

```
In [20]: function p=linterp(x,y)
    n=length(x);
    p = 0;
    for i=1:n
        L=1;
        for j=1:n
            if i~=j
                L = conv(L,[1 -x(j)]/(x(i)-x(j)));
            end
        end
        p = p + y(i)*L;
    end
endfunction

x=vreme_planinarenja;
y=predjeni_km;
p=linterp(x, y, 9)
```

```
warning: division by zero
warning: called from
    linterp at line 8 column 19
warning: division by zero
warning: called from
    linterp at line 8 column 19
warning: division by zero
warning: called from
    linterp at line 8 column 19
warning: division by zero
warning: called from
    linterp at line 8 column 19
p =

   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
```

Zbog čega smo dobili grešku deljenja nulom u pozivu linterp?

Zato što x_1 i x_2 imaju istu vrednost (vrednost 2), odnosno za jedno x imamo dva različita y. Isti slučaj je i sa x_4 i x_5.

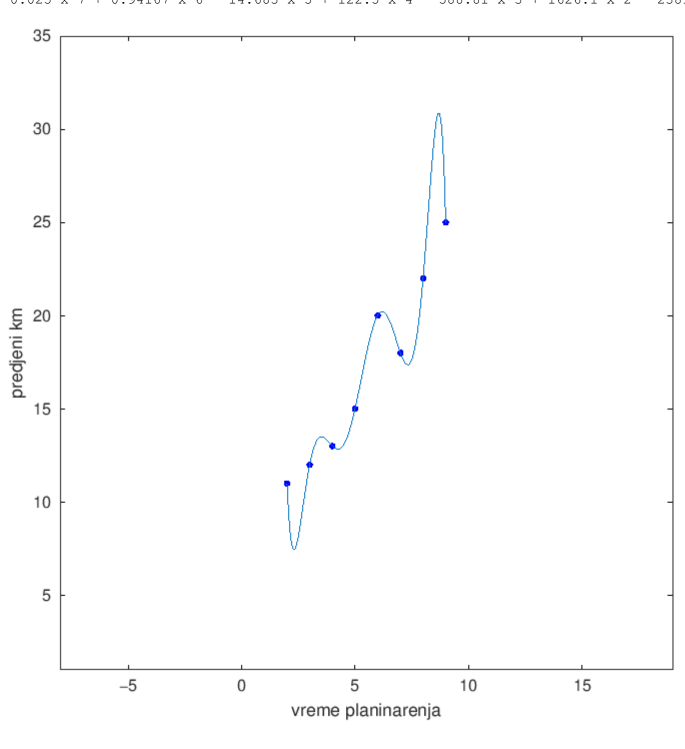
Već sada možemo videti zašto je interpolacija problematična ako imamo jako puno podataka. Vrlo retko ćete u velikoj količini podataka imati garanciju da ćete za jedno x imate samo jedno y.

U našem konkretnom primeru dva planinara su hodala isto vreme, a prešla zraličit put što je normalna pojava na koju intepolacija nije spremna.

Izbacićemo sada problematične tačke i pogledati kako izgleda interpolacioni polinom.

```
In [21]: x=vreme_planinarenja;
y=predjeni_km;
x(1)=[];
y(1)=[];
x(4)=[];
y(4)=[];
x
y
p=linterp(x, y, 7)

polyout(p,'x')
plot_points(x,y,'vreme planinarenja','predjeni km');
hold on;
xp=linspace(min(x),max(x),100);
plot(xp,polyval(p,xp))
```

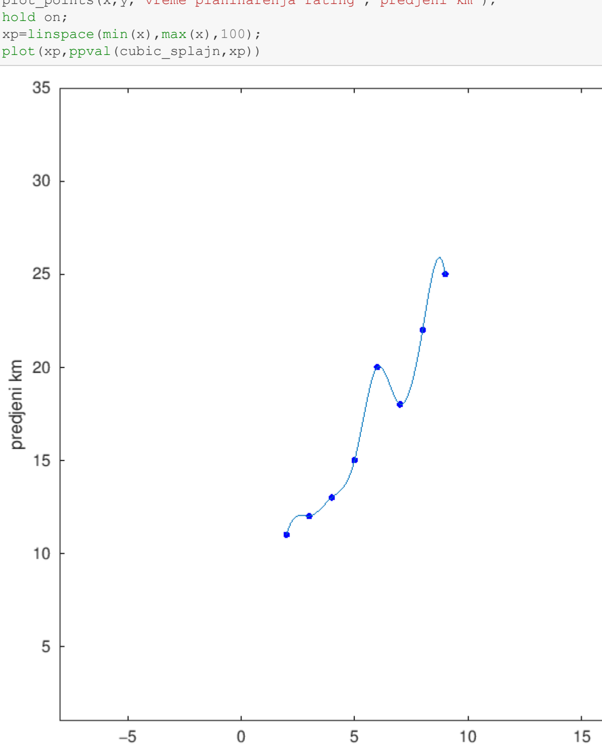
```
x =

   2   3   4   5   6   7   8   9

y =

   11   12   13   15   20   18   22   25

p =

 Columns 1 through 5:

    -0.025000      0.941667    -14.683333    122.500000   -588.808333

 Columns 6 through 8:

   1626.058333  -2380.983333   1432.000000

-0.025*x^7 + 0.94167*x^6 - 14.683*x^5 + 122.5*x^4 - 588.81*x^3 + 1626.1*x^2 - 2381*x^1 + 1432
```



Vidimo da interpolacioni polinom ima jako velike oscilacije i sigurno ne bi bio dobar predikcioni model za broj pređenih kilometara na osnovu vremena hodanja.

Može da pokušamo i splajnovima.

```
In [22]: cubic_splajn=splinefit(x, y, 7, "order", 3);

         plot_points(x,y,'vreme planinarenja rating','predjeni km');
         hold on;
         xp=linspace(min(x),max(x),100);
         plot(xp,ppval(cubic_splajn,xp))
```



Rezultat je malo bolji, ali još uvek imamo oscilacije. Male promene vremena hodanja daju velike promene pređenog puta, što nije realno. Cilj nam je da imamo prediktivni model koji što više moguće oslikava realnost, odnosno koji će davati predikcije koje imaju smisla u odnosu na domen u kome radimo.

Iz svih do sada navednih razloga odustajemo od upotrebe interpolacije već koristimo drugačiji pristup koji se naziva regresija.

Vraćmo se našem sistemu od dve nepoznate i deset jednačina. Probaćemo da ga rešimo tako da rezultujuća prava ne prolazi kroz svih 10 tačaka već da se u njih uklopi nabolje što može.

Najbolje moguće uklapanje definišemo tako što ćemo tražiti da prava koju određujemo za date tačke ima najmanji moguć zbir kvadrata grešaka. Prvo ćemo definisati šta je greška za tačku.

Greška e_i za svaku datu tačku x_i predstavlja razliku između vrednosti koju dobijemo kada x_i zamenimo u pravu (koju određujemo) i vrednosti y_i iz podataka (koja odgovara tački x_i).

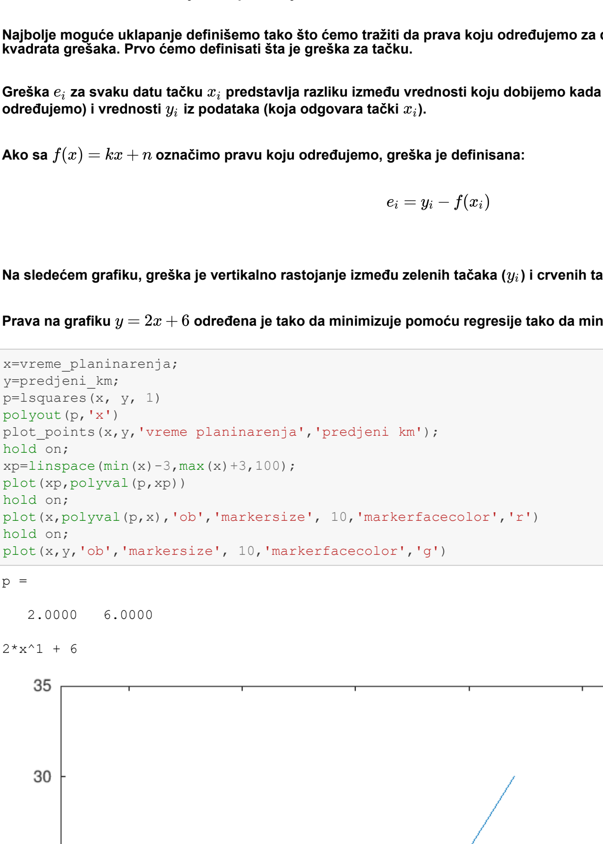
Ako sa $f(x) = kx + n$ označimo pravu koju određujemo, greška je definisana:

$$e_i = y_i - f(x_i)$$

Na sledećem grafiku, greška je vertikalno rastojanje između zelenih tačaka (y_i) i crvenih tačaka ($f(x_i)$).

Prava na grafiku $y = 2x + 6$ određena je tako da minimizuje pomoću regresije tako da minimizuje zbir kvadrata tih grešaka.

```
In [23]: x=vreme_planinarenja;
         y=predjeni_km;
         p=lsquares(x, y, 1)
         polyout(p,'x')
         plot_points(x,y,'vreme planinarenja','predjeni km');
         hold on;
         xp=linspace(min(x)-3,max(x)+3,100);
         plot(xp,polyval(p,xp))
         hold on;
         plot(x,polyval(p,x),'ob','markersize', 10,'markerfacecolor','r')
         hold on;
         plot(x,y,'ob','markersize', 10,'markerfacecolor','g')

p =

   2.0000   6.0000

2*x^1 + 6
```



Dali smo definiciju greške za tačku, sada ćemo definisati ukupnu grešku.

Ukupna greška definiše se kao zbir kvadrata grešaka za tačke:

$$SSE = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

gde je n ukupan broj datih tačaka, odnosno podataka (u našem slučaju 10).

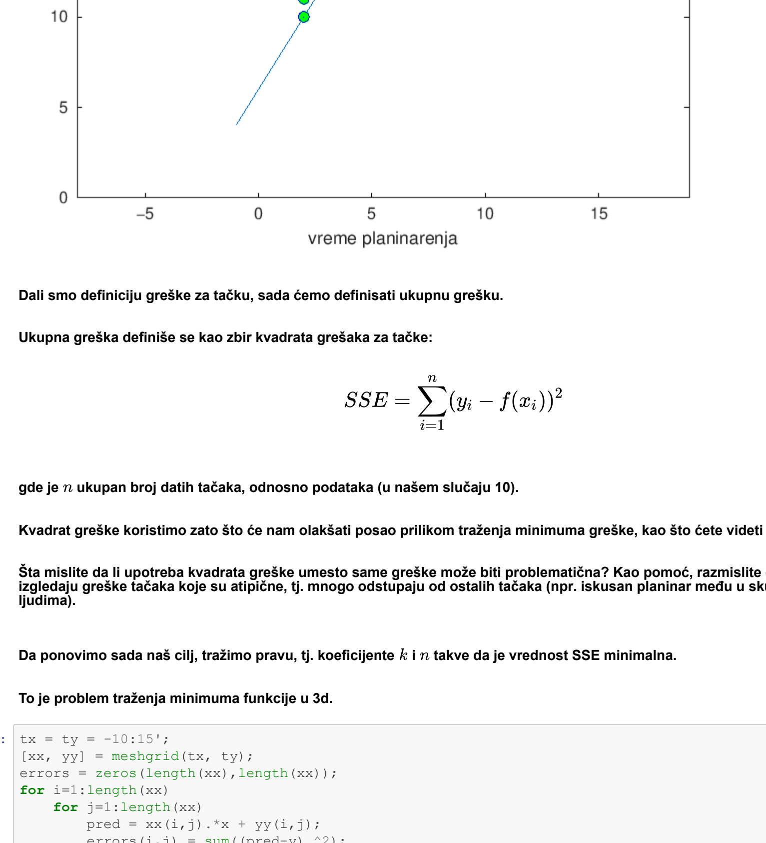
Kvadrat greške koristimo zato što će nam olakšati posao prilikom traženja minimuma greške, kao što ćete videti u nastavku.

Šta mislite da li upotreba kvadrata greške umesto same greške može biti problematična? Kao pomoć, razmislite o tome kako izgledaju greške tačaka koje su atipične, tj. mnogo odstupaju od ostalih tačaka (npr. iskusan planinar među u skupu sa "običnim" ljudima).

Da ponovimo sada naš cilj, tražimo pravu, tj. koeficijente k i n takve da je vrednost SSE minimalna.

To je problem traženja minimuma funkcije u 3d.

```
In [24]: tx = ty = -10:15';
         [xx, yy] = meshgrid(tx, ty);
         errors = zeros(length(xx),length(xx));
         for i=1:length(xx)
             for j=1:length(xx)
                 pred = xx(i,j).*x + yy(i,j);
                 errors(i,j) = sum((pred-y).^2);
             end
         end
         tz = errors;
         [minval, row] = min(min(errors,[],2));
         [minval, col] = min(min(errors,[],1));
         [minval, xx(row,col), yy(row,col)]
         mesh(tx, ty, tz);
         xlabel ("k");
         ylabel ("n");
         zlabel ("SSE");

ans =

   12    2    6
```

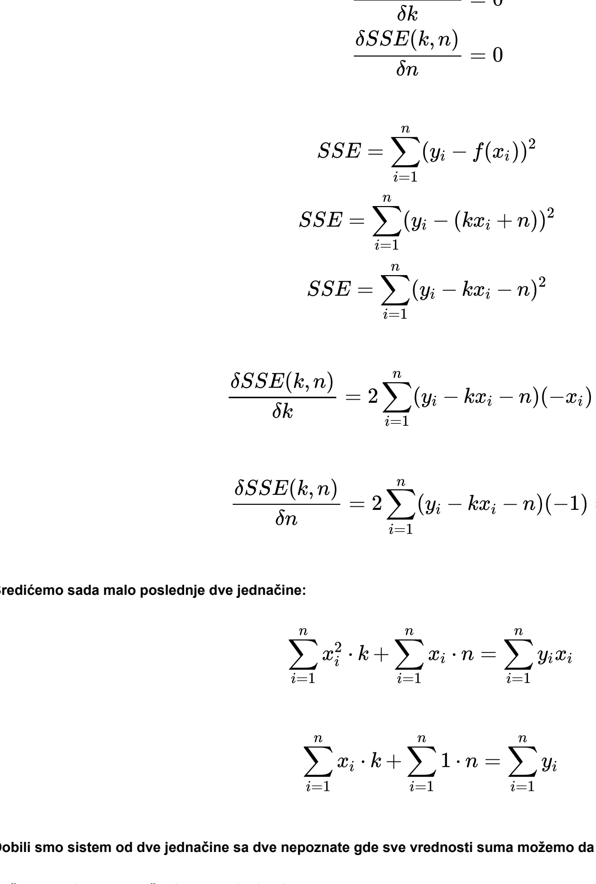


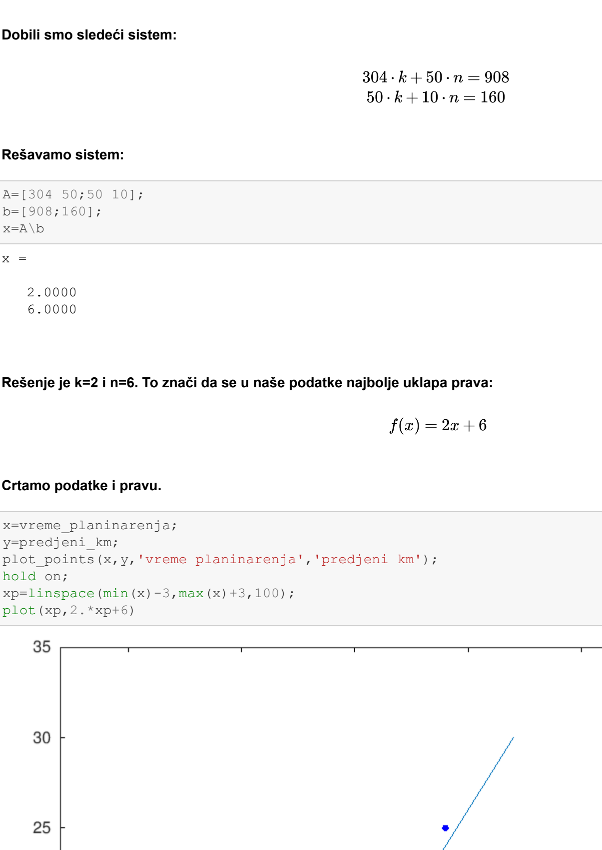
Na osnovu rezultata prethodnog koda vidimo da je minimum za $k = 2$ i $n = 6$, a minimalna vrednost SSE je 12.

Minimum funkcije $SSE(k, n)$ tražimo analitički pomoću parcijalnih izvoda.

Određujemo parcijalne izvode $SSE(k, n)$ po k i po n i izjednačavamo ih sa 0.

$$\frac{\delta SSE(k,n)}{\delta k} = 0$$
$$\frac{\delta SSE(k,n)}{\delta n} = 0$$





```
In [29]: SSE = sum((2.*x+6-y).^2)

SSE =  12
```

Vidimo da je ukupna kvadratna greška 12. Greška se meri u kvadratima jedinice podatka. Dakle, naša greška je $12km^2$.

Da li je greška koju smo dobili mala, odnosno da li naš model (prava) radi dobro?

Svaki odgovor na ovo pitanje bio bi subjektivan.

Da bi dobili objektivan odgovor moramo da uporedimo pravu sa nekim drugim modelom na istim podacima.

Takvi modeli se tipično nazivaju osnovni modeli (*baseline models*).

Osnovni model u našem slučaju odabraćemo tako što ćemo se pretvarati da nemamo x odnosno da nemamo informacije o vremenu planinarenja, već samo o pređenom putu.

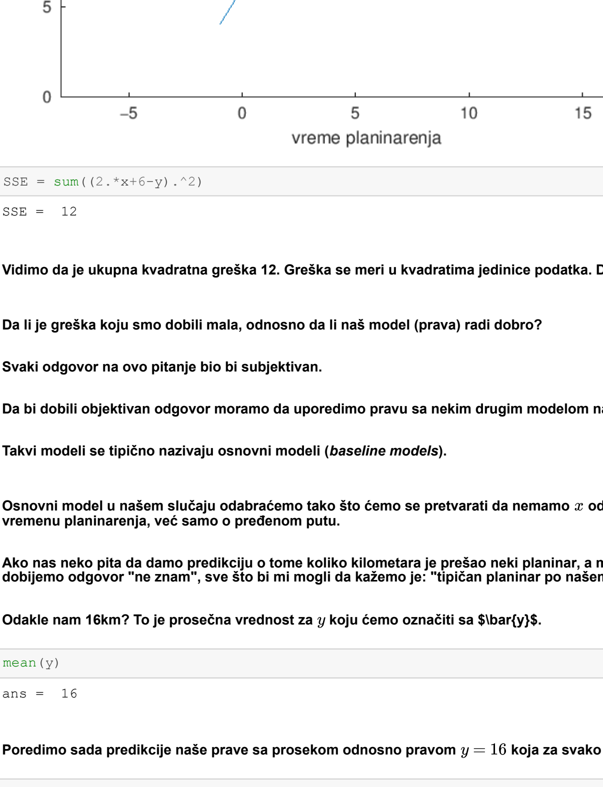
Ako nas neko pita da damo predikciju o tome koliko kilometara je prešao neki planinar, a mi pitamo koliko vremena je hodao, i dobijemo odgovor "ne znam", sve što bi mi mogli da kažemo je: "tipičan planinar po našem skupu podataka pređe 16km".

Odakle nam 16km? To je prosečna vrednost za y koju ćemo označiti sa $\bar{y}$.

```
In [30]: mean(y)

ans =  16
```

Poredimo sada predikcije naše prave sa prosekom odnosno pravom $y = 16$ koja za svako x predviđa uvek vrednost 16km.

```
In [29]: x=vreme_planinarenja;
         y=predjeni_km;
         plot_points(x,y,'vreme planinarenja','predjeni km');
         hold on;
         xp=linspace(min(x)-10,max(x)+10,100);
         plot(xp,2.*xp+6)
         hold on;
         plot(xp,ones(1,length(xp)).*16)
         hold on;
         plot(x,2.*x+6,'ob','markersize', 8,'markerfacecolor','r')
         hold on;
         plot(x,ones(1,length(x)).*16,'ob','markersize', 8,'markerfacecolor','g')
```



Na prvi pogled naša prava ima bolje predikcije, ali ćemo to ipak morati da formalizujemo.

Definišemo sada dve sume. Prva meri koliko vrednosti y odstupaju od svog proseka $\bar{y}$, odnosno koliko je proske dobar predikcioni model:

$$SST = \sum_{i=1}^{n}(y - \bar{y})^2$$

Oznaka ***SST*** je iz literature i označava tolanu sumu kvadrata (***sum of squares total***). U nastavku ćemo reći zašto se koristi termin "totalna".

```
In [31]: SST=sum((y-mean(y)).^2)

SST =  228
```

Vidimo da za naš primer SST ima vrednost $228km^2$. Na osnovu toga već možemo da tvrdimo da naš model (prava) radi jer je SSE mnogo manje od SST. U nastavku ćemo pokazati način na koji možemo da kvalitet modela izmerimo u procentima i time dobijemo meru kvaliteta modela koja ne zavisi od jedinice merenja y.

Računamo sada razliku između predikcija naše prave i $y = \bar{y}$:

$$SSR = \sum_{i=1}^{n}(f(x_i) - \bar{y})^2$$

Oznaka ***SSR*** je iz literature i označava sumu kvadrata koja je rezultat regresije (***sum of squares due to regression***), odnosno pokazuje koliko se regresija razlikuje od proseka. Na slici iznad ***SSR*** je vertikalno rastojanje između crevenih i zelenih tačaka.

```
In [32]: SSR=sum((2.*x+6-mean(y)).^2)

SSR =  216
```

Sada kada smo definisali SSR, objasnićemo zašto se SST zove totalna suma. Pod totalnom sumom misli se na ukupnu sumu rastojanja: od proseka do prave + od prave do y vrednosti iz podataka:

$$SST = SSR + SSE$$

Rastojanje od proseka do vrednosti y je zbir rastojanja od proseka do prave + od prave do y.

Definišemo sada još jednu sumu koja meri koliko vrednosti y odstupaju od svog proseka $\bar{y}$:

$$SST = \sum_{i=1}^{n}(y - \bar{y})^2$$

Oznaka ***SST*** je iz literature i označava tolanu sumu kvadrata (***sum of squares total***).

Pod totalnom sumom misli se na ukupnu sumu rastojanja: od proseka do prave + od prave do y vrednosti iz podataka:

$$SST = SSR + SSE$$

Rastojanje od proseka do vrednosti y je zbir rastojanja od proseka do prave + od prave do y.

Koeficijent determinacije

Kvalitet našeg modela izmerićemo pomoću koeficijenta determinacije (r^2) koji poredi koliko je naša prava (dobijena regresijom) doprinela tačnijoj predikciji y u odnosu na prosek:

$$r^2 = \frac{SSR}{SST}$$

Koefcijent determinacije meri koliki procenat varijabilnosti (rastojanja) y u odnosu na svoj prosek može da se objasni (predvidi) pomoću regresije za x.

Za naš konkretan primer to bio procenat varijabilnosti pređenog puta koji je objašnjen na osnovu regresione prave po vremenu hodanja.

Pokazaćemo sada koje su dve ekstremne vrednosti za r^2.

Ako je predikcija našeg modela uvek prosek, tj. rezultat regresije je prava koja takođe vraća uvek prosečnu vrednost, tada je $r^2 = 0$ jer:

$$f(x_i) = \bar{y}$$
$$SSR = \sum_{i=1}^{n}(f(x_i) - \bar{y})^2 = \sum_{i=1}^{n}(\bar{y} - \bar{y})^2 = 0$$
$$r^2 = \frac{SSR}{SST} = \frac{0}{SST} = 0$$

Ako je predikcija našeg modela uvek y, tj. rezultat regresije je prava koja prolazi kroz sve tačke y, tada je $r^2 = 1$ jer:

$$f(x_i) = y$$
$$SSR = \sum_{i=1}^{n}(f(x_i) - \bar{y})^2 = \sum_{i=1}^{n}(y - \bar{y})^2 = SST$$
$$r^2 = \frac{SSR}{SST} = \frac{SST}{SST} = 1$$

Izračunavamo r^2 za naš primer sa planinarima.

```
In [33]: SSE
         SSR
         SST

SSE =  12
SSR =  216
SST =  228
```

```
In [34]: r_2=SSR/SST

r_2 =  0.94737
```

Vidimo da je vrednost dosta blizu 1, odnosno da je ~95% varijabilnosti pređenog puta u našem skupu podatka objašneno pomoću regresije za vreme hodanja.

To znači da je naš model jako dobar.

Linearna regresija za polinom proizvoljnog stepena

Do sada smo pokazali na koji način se može pomoću linearne regresije u podatke može uklopiti prava. Sada ćemo pokzati opšti postupak pomoću koga se u podatke može uklopiti polinom proizvoljnog stepena m:

$$p(x) = \sum_{j=0}^{m} a_j x^j$$

Koristimo isti postupak kao i ranije. Tražimo minimum SSE, ali ovaj put ne za 2 parametra nego za $m + 1$.

Napomena: u nastavku su istaknuti najvažniji koraci postupka. Kompletan postupak dat je u udžbeniku i spada u informativni deo gradiva.

$$\frac{\delta SSE}{\delta a_0} = 0$$
$$\frac{\delta SSE}{\delta a_1} = 0$$
$$\dots$$
$$\frac{\delta SSE}{\delta a_m} = 0$$

$$\frac{\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} a_j x_i^j)^2}{\delta a_0} = 0$$
$$\frac{\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} a_j x_i^j)^2}{\delta a_1} = 0$$
$$\dots$$
$$\frac{\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} a_j x_i^j)^2}{\delta a_m} = 0$$

$$2\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} a_j x_i^j)(-x_i^0) = 0$$
$$2\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} a_j x_i^j)(-x_i^1) = 0$$
$$\dots$$
$$2\sum_{i=1}^{n}(y_i - \sum_{j=0}^{m} a_j x_i^j)(-x_i^m) = 0$$

$$\sum_{i=1}^{n}(\sum_{j=0}^{m} a_j x_i^j - y_i)x_i^0 = 0$$
$$\sum_{i=1}^{n}(\sum_{j=0}^{m} a_j x_i^j - y_i)x_i^1 = 0$$
$$\dots$$
$$\sum_{i=1}^{n}(\sum_{j=0}^{m} a_j x_i^j - y_i)x_i^m = 0$$

$$\sum_{i=1}^{n}(\sum_{j=0}^{m} a_j x_i^j)x_i^0 = \sum_{i=1}^{n} y_i x_i^0$$
$$\sum_{i=1}^{n}(\sum_{j=0}^{m} a_j x_i^j)x_i^1 = \sum_{i=1}^{n} y_i x_i^1$$
$$\dots$$
$$\sum_{i=1}^{n}(\sum_{j=0}^{m} a_j x_i^j)x_i^m = \sum_{i=1}^{n} y_i x^m$$

$$\sum_{j=o}^{m}\sum_{i=0}^{n} x_i^j x_i^0 a_j = \sum_{i=1}^{n} y_i x_i^0$$
$$\sum_{j=o}^{m}\sum_{i=0}^{n} x_i^j x_i^1 a_j = \sum_{i=1}^{n} y_i x_i^1$$
$$\dots$$
$$\sum_{j=o}^{m}\sum_{i=0}^{n} x_i^j x_i^m a_j = \sum_{i=1}^{n} y_i x_i^m$$

Kao malu pomoć u daljem izvođenju prikazaćemo sumu $\sum_{j=o}^{m}\sum_{i=0}^{n} x_i^j x_i^0 a_j$ po sabircima (suma ima m sabiraka):

$$\sum_{j=o}^{m}\sum_{i=0}^{n} x_i^j x_i^0 a_j = \sum_{i=0}^{n} x_i^0 x_i^0 a_0 + \sum_{i=0}^{n} x_i^1 x_i^0 a_1 + \dots + \sum_{i=0}^{n} x_i^m x_i^0 a_m$$

Ako posmatramo na primer x^0 i x^1 vidimo da je suma $\sum_{i=0}^{n} x^0 x^1$ ustvari skalarni proizvod ta dva vektora koji se može dobiti tako što se jedan od vektora uzme kao vrsta, a drugi kao kolona i onda se primeni množenje matrica:

$$\begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \end{bmatrix}_{1\times n} * \begin{bmatrix} x_1^0 \\ x_2^0 \\ \dots \\ x_n^0 \end{bmatrix}_{n\times 1} = \sum_{i=1}^{n} x^0 x^1$$

To znači da se sve sume $\sum_{i=0}^{n} x_i^j x_i^k$ gde je $k = 0, 1, 2, \dots, m$ iznad mogu dobiti kao proizvod matrica:

$$A^T * A$$

$$A^T = \begin{bmatrix} x_1^0 & x_2^0 & \dots & x_n^0 \\ x_1^1 & x_2^1 & \dots & x_n^1 \\ \dots \\ x_1^m & x_2^m & \dots & x_n^m \end{bmatrix}_{m+1\times n} \quad A = \begin{bmatrix} x_1^0 & x_1^1 & \dots & x_1^m \\ x_1^0 & x_2^1 & \dots & x_2^m \\ \dots \\ x_n^0 & x_n^1 & \dots & x_n^m \end{bmatrix}_{n\times m+1}$$

Ako sada u proizvod matrica $A^T A$ ubacimo i vektor nepoznatih koeficijenata polinoma $a = (a_0, a_1, \dots, a_m)$ sve sume oblika $\sum_{j=o}^{m}\sum_{i=0}^{n} x_i^j x_i^k a_j$ gde je $k = 0, 1, 2, \dots, m$ mogu se dobiti kao sledeći prizvod matrica:

$$A^T * A * a$$

Dodajemo dimenzionalnosti matrica i vektora u proizvod:

$$A^T_{m+1\times n} * A_{n\times m+1} * a_{m+1\times 1}$$

$$A^T = \begin{bmatrix} x_1^0 & x_2^0 & \dots & x_n^0 \\ x_1^1 & x_2^1 & \dots & x_n^1 \\ \dots \\ x_1^m & x_2^m & \dots & x_n^m \end{bmatrix}_{m+1\times n} \quad A = \begin{bmatrix} x_1^0 & x_1^1 & \dots & x_1^m \\ x_1^0 & x_2^1 & \dots & x_2^m \\ \dots \\ x_n^0 & x_n^1 & \dots & x_n^m \end{bmatrix}_{n\times m+1} * \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_m \end{bmatrix}_{m+1\times 1}$$

Na sličan način, sve sume oblika $\sum_{i=1}^{n} y_i x_i^k$ gde je $k = 0, 1, 2, \dots, m$ mogu se dobiti kao proizvod vektora $y = (y_1, y_2, \dots, y_n)$ i matrice A^T:

$$A^T = \begin{bmatrix} x_1^0 & x_2^0 & \dots & x_n^0 \\ x_1^1 & x_2^1 & \dots & x_n^1 \\ \dots \\ x_1^m & x_2^m & \dots & x_n^m \end{bmatrix}_{m+1\times n} * \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix}_{n\times 1}$$

Iz prethodnog dobijamo da se problem određivanja polinoma:

$$p(x) = \sum_{j=0}^{m} a_j x^j$$

koji se najbolje ukalpa u podatke $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$ svodi na rešavanje sledećeg sistema:

$$(A^T A)a = A^T y$$

Napisaćemo sada kod koji za date podatke i stepen polinoma m formira matricu A i rešava sistem prikazan u prethodom redu.

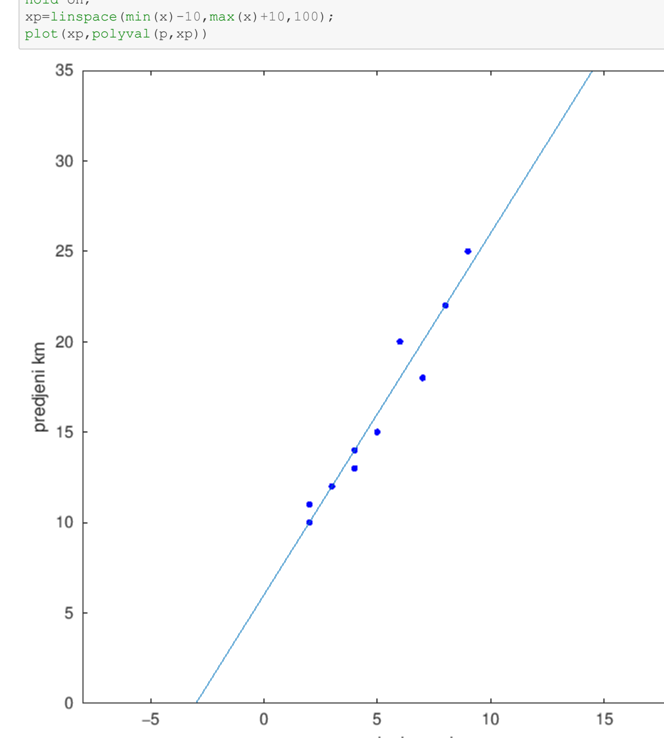
```
In [35]: function p=lsquares(x,y,stepen)
             n=length(x);
             A=zeros(n,stepen+1);
             for i=1:n
                 for j=1:stepen+1
                     A(i,j)=x(i).^(j-1);
                 end
             end
         p=(A'*A)\(A'*y');
         p=fliplr(p'); %ova funkcija je dodata jer je rezultat oblika a0,a1,...,am, a MATLAB radi sa polinomima
          u obrnutom redosledu.
         endfunction
```

```
In [36]: fliplr([0 1 2 3 4])

ans =

   4   3   2   1   0
```

In [37]:
```
x=vreme_planinarenja;
y=predjeni_km;
p=lsquares(x,y,1)
polyout(p,'x')
```

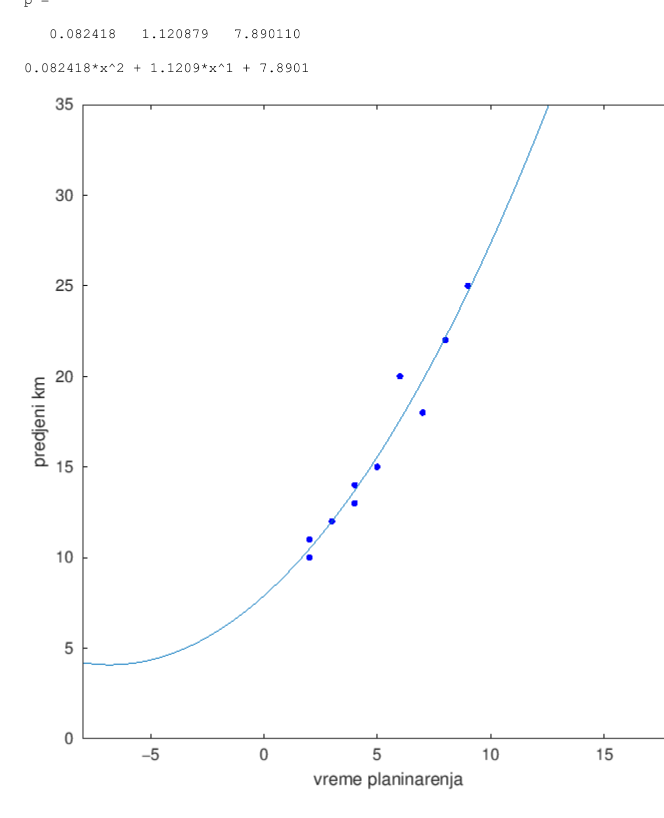
```
p =

   2.0000   6.0000

2*x^1 + 6
```

In [38]:
```
plot_points(x,y,'vreme planinarenja','predjeni km');
hold on;
xp=linspace(min(x)-10,max(x)+10,100);
plot(xp,polyval(p,xp))
```



In [39]:
```
p=lsquares(x,y,2)
polyout(p,'x')

plot_points(x,y,'vreme planinarenja','predjeni km');
hold on;
xp=linspace(min(x)-10,max(x)+10,100);
plot(xp,polyval(p,xp))
```

```
p =

   0.082418   1.120879   7.890110

0.082418*x^2 + 1.1209*x^1 + 7.8901
```



Linearna regresija pomoću polinoma je samo specijalan slučaj linearne regresije za funkcije oblika $y = x^i$. Zašto?

Rezultat linearne regresije je linearna kombinacija koeficjenata i funkcija. Funkcije zadajemo, a koeficijente određujemo rešavanjem sistema koji formiramo na način na koji smo pokazali iznad.

Linearna regresija može se raditi za bilo koje funkcije $f_1(x), f_2(x), \ldots, f_m(x)$, a tada matrica ima oblik:

$$A = \begin{bmatrix} f_0(x_1) & f_1(x_1) & \ldots & f_m(x_1) \\ f_0(x_2) & f_1(x_2) & \ldots & f_m(x_2) \\ \ldots & & & \\ f_0(x_n) & f_1(x_n) & \ldots & f_m(x_n) \end{bmatrix}$$

Funkcija koja je data u nastavku deo je informativnog gradiva, a pomoću nje je moguće uraditi linearnu regresiju za proizvoljan niz funkcija.

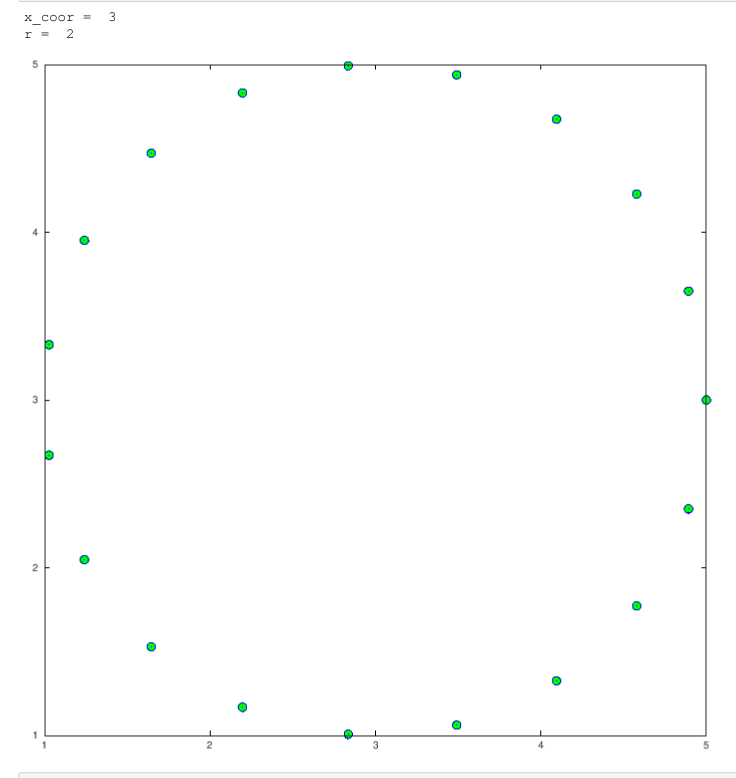
In [40]:
```
function p=lsquares_general(x,y,functions)
    n=length(x);
    m=length(functions);
    A=zeros(n,m);
    for i=1:n
        for j=1:m
            A(i,j)=feval(functions{j},x(i));
        end
    end
p=(A'*A)\(A'*y');
endfunction
```

In [41]:
```
x=vreme_planinarenja;
y=predjeni_km;
p=lsquares_general(x,y,{@(x)x,@(x)1})
```

```
p =

   2.0000
   6.0000
```

In [42]:
```
x_coor=y_coor=3 %centra kruga
r=2 %poluprečnik
t = linspace(0,2*pi,20)';
circsx = r.*cos(t) + x_coor;
circsy = r.*sin(t) + y_coor;
plot(circsx,circsy,'ob','markersize', 8,'markerfacecolor','g')
%za objašnjenje pogledati definiciju sinusa i konsinus na jediničnom krugu
%za tačku (x,y) na jediničnom krugu koja odgovara uglu t važi x=sin(t), a y=sin(t)
%pomeramo koordiante za zadati centar i množimo sa r ako ne želimo baš jedinični krug
```

```
x_coor =  3
r =  2
```

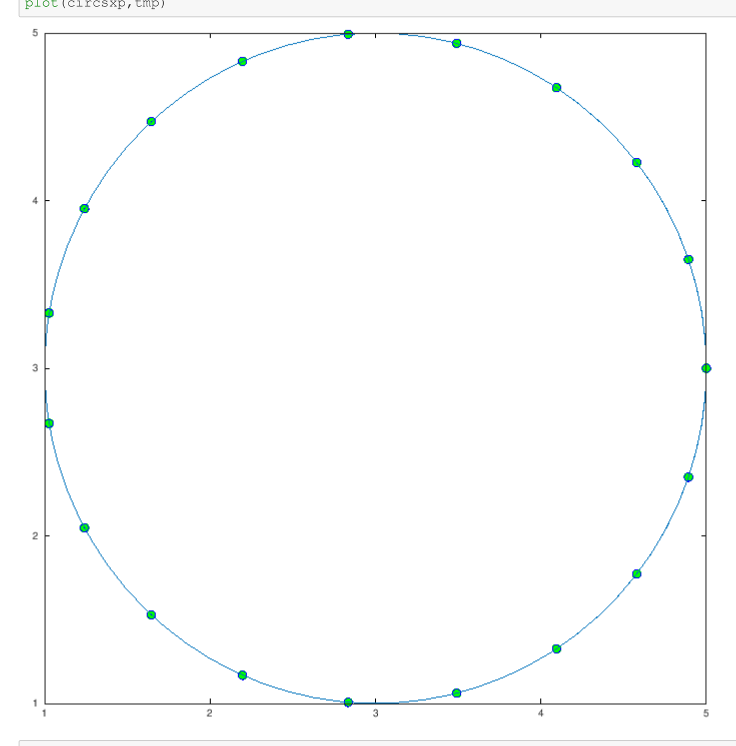


In [45]:
```
x=linspace(0,2*pi,20)';
y=circsy';
```

In [46]:
```
aproks=lsquares_general(x,y,{@(x)sin(x),@(x)1})
```

```
aproks =

   2.0000
   3.0000
```

In [48]:
```
plot(circsx,circsy,'ob','markersize', 8,'markerfacecolor','g')
hold on;
xp=linspace(min(x)-1,max(x)+1,100);
circsxp = r.*cos(xp) + x_coor;
tmp = aproks(1).*sin(xp)+aproks(2);
plot(circsxp,tmp)
```



In [ ]: