Reševanje nelinearnih enačb

Martin Vuk

29. april 2015

1 Newtonova metoda

Pri Newtonovi metodi iščemo ničlo dane funkcije

$$f(x) = 0,$$

za katero znamo izračunati odvod. Približek za ničlo izračunamo z rekurzivno danim zaporedjem približkov. Za funkcijo ene spremenljivke, je rekurzivna formula enaka

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

Formulo lahko posplošimo na vektorske funkcije več spremenljivk $F : \mathbb{R}^n \to \mathbb{R}^n$, tako da namesto odvoda funckije uporabimo Jacobijevo matriko odvodov

$$JF(x) = \begin{bmatrix} \frac{\partial F_1(x)}{\partial x_1} & \frac{\partial F_1(x)}{\partial x_2} & \cdots & \frac{\partial F_1(x)}{\partial x_n} \\ \frac{\partial F_2(x)}{\partial x_1} & \frac{\partial F_2(x)}{\partial x_2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n(x)}{\partial x_1} & \frac{\partial F_n(x)}{\partial x_2} & \cdots & \frac{\partial F_n(x)}{\partial x_n} \end{bmatrix}.$$

Pri matričnem množenju moramo paziti na vrstni red. Rekurzivna fomrula za Newtonovo iteracijo se glasi

$$\mathbf{x}_{n+1} = \mathbf{x}_n - JF(\mathbf{x}_n)^{-1} F(\mathbf{x}_n).$$

1.1 Program

In [8]:
```
function [x,it]= newton(F, JF, x0, tol, maxit)
% Funkcija [x,it]= newton(F, JF, x0, tol, maxit)
% poišče približek za rešitev enačbe F(x)=0 z Newtonovo iteracijo
% F ... ročica na funkcijo F (sprejme vektor in vrne vektor)
% JF ... ročica na odvof F (sprejme vektor in vrne matriko)
% x0 ... začetni približek
% tol ... razlika dveh zaporednih približkov, pri kateri se iteracija ustavi
% maxit ... maksimalno število korakov
x = x0;
for it=1:maxit
  z = F(x);
  x = x - JF(x)\z;
  if abs(x-x0) < tol
    break
  end
  x0=x;
endfor
endfunction
```

1.2 Naloga

Poišči rešitev sistema enačb

$$x_1^2 - x_2^2 = 1,$$
$$x_1 + x_2 - x_1 x_2 = 1.$$

z Newtonovo metodo.

Sistem enačb najprej preoblikujemo v vektrosko enačbo $F(x) = 0$, kjer je $F : \mathbb{R}^2 \to \mathbb{R}^2$ dana s predpisom

$$F(x) = \begin{bmatrix} F_1(x) \\ F_2(x) \end{bmatrix} = \begin{bmatrix} x_1^2 - x_2^2 - 1 \\ x_1 + x_2 - x_1 x_2 - 1 \end{bmatrix}$$

njen odvod pa je dan z matriko

$$JF(x) = \begin{bmatrix} \frac{\partial F_1(x)}{\partial x_1} & \frac{\partial F_1(x)}{\partial x_2} \\ \frac{\partial F_2(x)}{\partial x_1} & \frac{\partial F_2(x)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 & -2x_2 \\ 1 - x_2 & 1 - x_1 \end{bmatrix}.$$

Če želimo uporabiti funkcijo `newton`, moramo definirati funkcije za F in JF.

```
In [9]: % funkcija F
        F = @(x) [x(1)^2-x(2)^2-1; x(1) + x(2)-x(1)*x(2)-1]
```

```
Out[9]: F =

        @(x) [x(1) ^ 2 - x(2) ^ 2 - 1; x(1) + x(2) - x(1) * x(2) - 1]
```

```
In [10]: JF = @(x) [2*x(1) -2*x(2); 1-x(2) 1-x(1)]
```

```
Out[10]: JF =

         @(x) [2 * x(1), -2 * x(2); 1 - x(2), 1 - x(1)]
```

Najprej si moramo izbrati začetni približek

```
In [11]: x0=[1 2]';
         [x,it]=newton(F,JF,x0,1e-10,100)
```

```
Out[11]: warning: matrix singular to machine precision, rcond = 0
         x =

            1.4142
            1.0000

         it =  9
```

Najveja težava Newtonove metode je prav izbira začetnega približka. Konvergenca zaporedja približkov je namreč zelo odvisna od izbire začetnega približka. V našem primeru smo si izbrali neroden začetni približek, saj je Jacobijeva matrika v izbranem začetnem približku singularna.

```
In [12]: x=[1;2]; x=x-JF(x)\F(x); JF(x)
```

```
Out[12]: ans =

           2  -2
           0   0
```

Da je program kljub singularni Jacobijevi matriki poiskal ničlo, se moramo zahvaliti robustno sprogramiranemu operatorju \ v octave. Operator \ vrne rešitev za skoraj vse mogoče primere linearnih sistemov (predoločene, poddoločene, ...). Iteracija se tako kljub slabi izbiri začetnega približka ni končala.

Če izberemo različne začetne približke, lahko zaporedje približkov konvergira k različnim ničlam:

```
In [21]: x0 = [-1.5,-1.5]'; [x,it]=newton(F,JF,x0,1e-10,100)

Out[21]: x =

          -1.4142
           1.0000

         it =  8
```

1.3 Konvergenčna območja Newtonove metode

V prejšnjem primeru smo videli, da za različne približke, Newtonova metoda konvergira k različnim rešitvam sistema. Območja konvergence za posamezne rešitve so v resnici fraktalna. Napišimo program, ki bo za dano območje preveril, h kateri ničli konvergira Newtonova metoda in ustrezno pobarval točke.

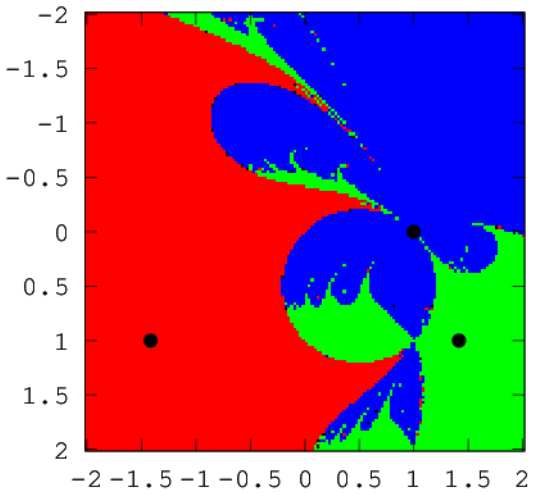
```
In [2]: warning ("off", "Octave:broadcast");

        function [Z, nicle] = fraktal(F,JF,n,meje,tol,maxit)
        # Funkcija fraktal(F,JF,n,meje,tol,maxit) generira fraktalno sliko,
        # ki obarva točke, glede na to h kateri ničli konvergira
        # Newtonova metoda
        Z = zeros(n);
        xx = linspace(meje(1),meje(2),n);
        yy = linspace(meje(3),meje(4),n);
        nicle = [];
        for i=1:n
          for j=1:n
            [x,it] = newton(F,JF,[xx(i);yy(j)],tol,maxit);
            if (it>=maxit-1) | (norm(x)>2*max(abs(meje)))
              continue
            endif
            if length(nicle<=0)
              idx = find(max(abs(x-nicle))<10*tol);
              if length(idx)<=0
                 nicle = [nicle x];
                 Z(j,i) = size(nicle,2)+1;
              else
                 Z(j,i)=idx;
              endif
            else
              nicle = [nicle x];
              Z(j,i) = 1;
```

```
      endif
    endfor
  endfor
endfunction
```

In [35]:
```
n = 150; tol = 1e-3; maxit=20; meje=[-2,2,-2,2];
Z = fraktal(F,JF,n,meje,tol,maxit);
```

Out[35]:
```
warning: matrix singular to machine precision, rcond = 1.6218e-17
warning: matrix singular to machine precision, rcond = 2.29754e-17
```

In [32]:
```
colormap([0 0 0;1 0 0; 0 1 0; 0 0 1])
image(xx,yy,Z+1)
hold on
plot(nicle(1,:),nicle(2,:),'.k','markersize',15)
print -dpng fraktal.png
```



Na sliki so različne točke v pravokotniku $[-2,2]\times[-2,2]$ obarvane z različnimi barvami, glede na to, kam konvergira zaporedje približkov Newtonove metode

- rdeča: $\mathbf{x}_n \to [-\sqrt{2},1]$
- zelena: $\mathbf{x}_n \to [\sqrt{2},1]$
- modra: $\mathbf{x}_n \to [1,0]$

2 Metoda najmanjših kvadratov

Poišči parametre a, b, pri katerih bo funkcija

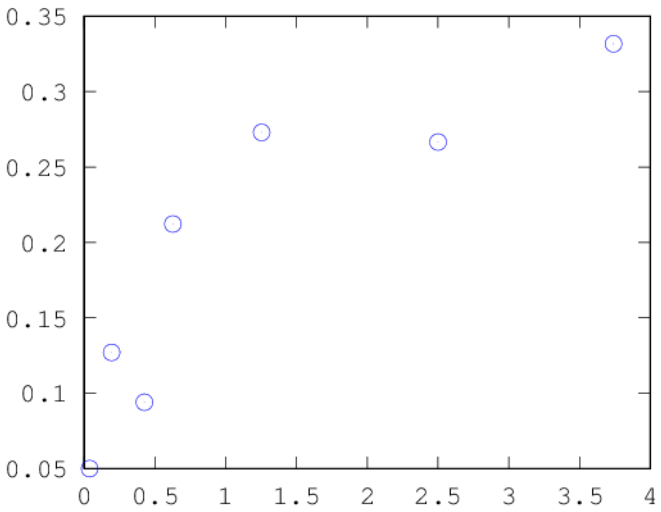
$$f(x) = \frac{ax}{b+x}$$

najbolje aproksimirala podatke. Iščemo torej vrednost parametrov a, b, pri katerih je vrednost napake

$$E(a,b) = \sum_i \left(y_i - \frac{ax_i}{b+x_i} \right)^2$$

najmanjša.

```
In [3]: x = [0.038  0.194  0.425  0.626  1.253  2.500  3.740];
        y = [0.050 0.127  0.094  0.2122  0.2729  0.2665  0.3317];
        plot(x,y,'o')
```



2.1 Newtonova metoda

Funkcija napake doseže minimum v stacionarni točki. Iščemo torej vrednosti parametrov a, b, pri katerih je gradient funkcije napake $E(a,b) enak 0:

$$\text{grad} \sum \left(y_i - \frac{ax_i}{b + x_1} \right)^2 = 2 \sum_i (y_i - \frac{ax_i}{b + x_i}) \begin{bmatrix} -\frac{x_i}{b+x_i} \\ \frac{ax_i}{(b+x_i)^2} \end{bmatrix} = \sum \begin{bmatrix} -\frac{x_i y_i}{b+x_i} + \frac{ax_i^2}{(b+x_i)^2} \\ \frac{ax_i y_i}{(b+x_i)^2} - \frac{a^2 x_i}{(b+x_i)^3} \end{bmatrix}$$

Odvod gradienta je Hessejeva matrika

$$H = \sum_i \begin{bmatrix} \frac{x_i^2}{(b+x_i)^2} & \frac{x_i y_i}{(b+x_i)^2} - 2\frac{ax_i}{(b+x_i)^3} \\ \frac{x_i y_i}{(b+x_i)^2} - 2\frac{ax_i}{(b+x_i)^3} & -2\frac{ax_i y_i}{(b+x_i)^3} - 3\frac{a^2 x_i}{(b+x_i)^4} \end{bmatrix}$$

```
In [3]: function y = F2a(x)
        %y = F2a(x) vrne vrednost funkcije za nalogo 2a.

        %podatki
        xi = [0.038; 0.194; 0.425; 0.626; 1.253; 2.5; 3.74];
        yi = [0.05; 0.127; 0.094; 0.2122; 0.2729; 0.2665; 0.3317];

        a = x(1);
        b = x(2);

        y1 = sum(2*(yi - a*xi./(b + xi)).*(-xi./(b + xi)));
        y2 = sum(2*(yi - a*xi./(b + xi)).*(a*xi./(b + xi).^2));

        y = [y1; y2];
        endfunction

        function J = JF2a(x)
```

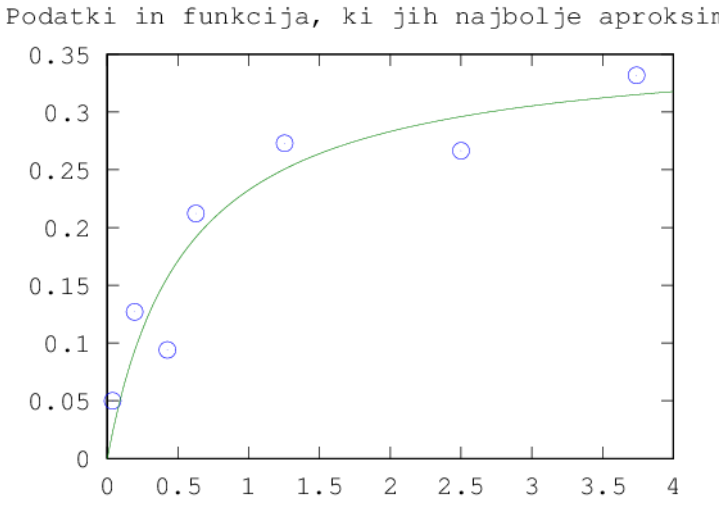
```
%J = JF2a(x) vrne Jacobijevo matriko funkcije za nalogo 2a.

%podatki
xi = [0.038; 0.194; 0.425; 0.626; 1.253; 2.5; 3.74];
yi = [0.05; 0.127; 0.094; 0.2122; 0.2729; 0.2665; 0.3317];

a = x(1);
b = x(2);

J11 = sum(2*(xi./(b + xi)).^2);
J12 = sum(2./(b + xi).^2 .* (xi.*yi - 2*a*xi.^2./(b + xi)));
J21 = J12;
J22 = sum(2*(3*a^2*xi.^2./(b + xi).^4 - 2*a*xi.*yi./(b + xi).^3));

J = [J11, J12; J21, J22];
endfunction
```

Če primerno izberemo začetni približek, dobimo optimalne vrednosti parametrov a in b.

```
In [6]: alfa0 = [0.5;0.5];
        [alfa,it] =  newton(@F2a,@JF2a,alfa0,1e-5,100)
```

```
Out[6]: alfa =

           0.36184
           0.55627

        it =  7
```

```
In [7]: t = linspace(0,4);
        plot(x,y,'o',t,alfa(1)*t./(alfa(2)+t))
        title("Podatki in funkcija, ki jih najbolje aproksimira")
```

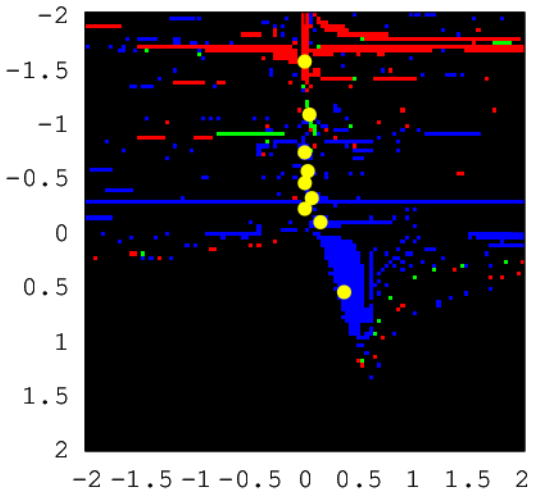


Problem je, da je Newtonova metoda precej občutljiva na izbiro začetnega približka. Množica začetnih približkov, v katerih metoda konvergira k optimalni rešitvi je zelo majhna. Na naslednji sliki je z modro

označeno območje za katerega Newtonova metoda konvergira k optimalni rešitvi, s črno pa območje, ko Newtonova metoda ne konvergira (po določenem številu korakov).

```
In [ ]: n = 100; tol = 1e-3; maxit=15; meje=[-2,2,-2,2];
        [Z,nicle] = fraktal(@F2a,@JF2a,n,meje,tol,maxit);
```

```
In [15]: xx = linspace(meje(1),meje(2),n);
         yy = linspace(meje(3),meje(4),n);
         colormap([0 0 0;1 0 0; 0 1 0; 0 0 1])
         image(xx,yy,Z+1)
         hold on
         plot(nicle(1,:),nicle(2,:),'.y','markersize',15)
         print -dpng fraktal2a.png
```



2.2 Funkcija napake

Razlog za slabo konvergenco je sama narava funkcije napake

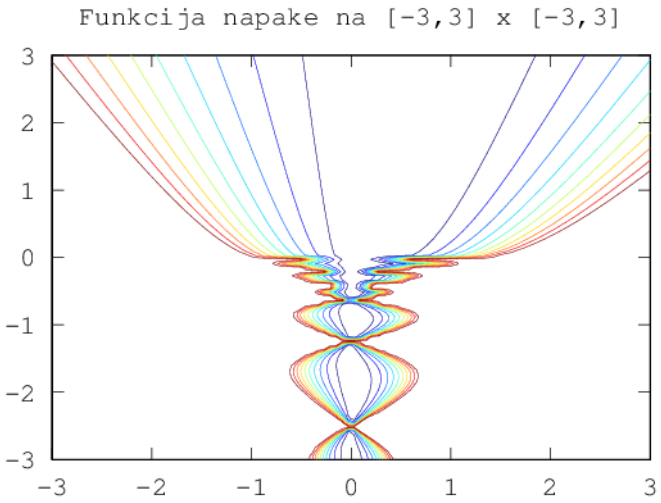
$$E(a,b) = \sum_i \left(\frac{ax_i}{b + x_i} - y_i \right)^2,$$

ki ima veliko število stacionarnih točk (lokalnih minimumov in sedel).

```
In [ ]: function e = error(a,b)
        %e = error(ab) vrne vrednost funkcije napake za nalogo 2a.

        %podatki
        xi = [0.038; 0.194; 0.425; 0.626; 1.253; 2.5; 3.74];
        yi = [0.05; 0.127; 0.094; 0.2122; 0.2729; 0.2665; 0.3317];


        %e = sum((yi - a*xi./(b + xi).^2);
        e = zeros(size(a));
        for i=1:length(xi)
          e = e + (yi(i) - (a*xi(i))./(b + xi(i))).^2;
```

```
        endfor

        endfunction
```

```
In [55]: x = linspace(-3,3); y = x;
         [X,Y] = meshgrid(x,y);
         e  = error(X,Y);
         e(find(abs(e)>10)) = 10;
         contour(x,y,e)
         title("Funkcija napake na [-3,3] x [-3,3]")
```



2.3 Gauss-Newtonova iteracija

Videli smo, da je Newtonova metoda precej obutljiva pri iskanju minima funkcije napake $E(a, b)$. Bolj robustna metoda za iskanje optimalnih vrednosti parametrov po metodi najmanjših kvadratov je Gauss-Newtonova iteracija.

Problem zastavimo nekoliko drugače. Iščemo rešitev predoločenega sistema z metodo najmanjših kvadratov. Namesto sistema, ki ga dobimo za minimum funkcije napake, za vsak podatek zapišemo eno nelinearno enačbo:

$$y_i = \frac{ax_i}{b + x_i}.$$

Sistem enačb, ki ga dobimo, je nelinearen in ima več enačb kot neznank. Iščemo vrednosti parametrov a, b, pri katerem bo kvadratna norma vektorja razlik med levimi in desnimi stranmi minimalna.

Pri Gauss Newtonovi metodi na vsakem koraku rešimo linearen predoločen sistem, ki je linearizacija originalnega nelinearnega sistema v trenutnem približku. Formula za naslednji približek je podobna kot pri Newtonovi metodi, le da namesto inverza JF^{-1}, ki ne obstaja, nastopa psevdoinverz JF^{+}

$$\mathbf{x}_{n+1} = \mathbf{x}_n - JF(\mathbf{x}_n)^{+} F(\mathbf{x}_n).$$

V praksi dejansko ne računamo pseudoinverza, saj obstajajo bolj učinkovite metode za reševanje predoločenih sistemov po metodi najmanjših kvadratov. Namesto formule, ki nas lahko zavede, da uporablimo neučinkovito metodo, raje zapišemo algoritem

```
na vsakem koraku:
```

1. reši sistem JF(x)y=F(x) po metodi najmanjših kvadratov
2. popravi približek x = x - y

V octaveu lahko preprosto uporabimo operator , ki za predoločeni sistem vrne natanko rešitev, pri kateri je vsota kvadratov razik minimalna. Tako lahko uporabimo kar isto funkcijo kot za Newtonovo metodo.

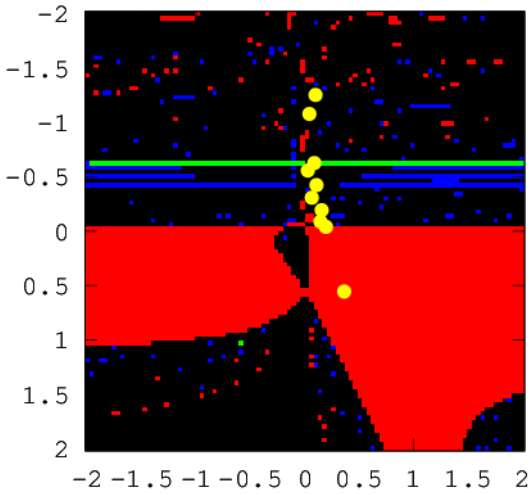
V prejšnjem primeru dobimo sistem s 7 enačbami in 2 neznankama. Funkcija F je sedaj $F : \mathbb{R}^2 \to \mathbb{R}^7$

$$F = \begin{bmatrix} y_1 - \frac{ax_1}{b+x_1} \\ y_1 - \frac{ax_2}{b+x_2} \\ \vdots \\ y_1 - \frac{ax_n}{b+x_n} \end{bmatrix}$$

matrika odvodov pa $n \times 2$

$$JF = \begin{bmatrix} -\frac{x_1}{b+x_1} & \frac{ax_1}{(b+x_1)^2} \\ -\frac{x_2}{b+x_2} & \frac{ax_2}{(b+x_2)^2} \\ \vdots & \vdots \\ -\frac{x_n}{b+x_n} & \frac{ax_n}{(b+x_n)^2} \end{bmatrix}$$

```
In [44]: x = [0.038  0.194  0.425  0.626  1.253  2.500  3.740]';
         y = [0.050 0.127  0.094  0.2122  0.2729  0.2665  0.3317]';
         F2b = @(alfa) y - alfa(1)*x./(alfa(2)+x)
         JF2b = @(alfa) [-x./(alfa(2)+x) alfa(1)*x./(alfa(2)+x).^2]

Out[44]: JF2b =

         @(alfa) [-x ./ (alfa(2) + x), alfa(1) * x ./ (alfa(2) + x) .^ 2]



In [45]: alfa0 = [1;2];
         [alfa,it] =  newton(F2b,JF2b,alfa0,1e-5,100)

Out[45]: alfa =

            0.36184
            0.55627

         it =  6


In [53]: n = 100; tol = 1e-3; maxit=20; meje=[-2,2,-2,2];
         [Z,nicle] = fraktal(F2b,JF2b,n,meje,tol,maxit);

In [54]: xx = linspace(meje(1),meje(2),n);
         yy = linspace(meje(3),meje(4),n);
         colormap([0 0 0;1 0 0; 0 1 0; 0 0 1])
         image(xx,yy,Z+1)
         hold on
         plot(nicle(1,:),nicle(2,:),'.y','markersize',15)
         print -dpng fraktal2b.png
```



Območje konvergence k optimalni rešitvi (rdeča barva) je za Gauss-Newtonovo metodo precej večje, kot pri Newtonovi metodi.

In [ ]: