Matematično modeliranje *II domača naloga*

Avtor: Melanija Kraljevska, 63170368

1. Če hočemo poiskati minimalno razdaljo med dve podani krivulji F in G, poiščemo minimum funkcije H, ki jo predstavimo kot: $H(t,s) = ||F(t) - G(s)||^2$, to lahko izračunamo s pomočjo **gradientne metode**. Izračunamo gradient funkcije: *H(t,s)*. Naj bosta funkciji F in G:

F(t) = [f, df, ddf] in G(s)=[g, dg, ddg]

kjer sta f in g vektorja s koordinatami na krivulji, vektorja df in dg prvi odvod ter ddf in ddg drugi odvod ustrezne funkcije po parametru. Odvod funkcije *H(t,s)* naredimo po obe spremenljivki in ga shranimo v matriko H:

*бH/ бt = (б(||F(t) - G(s)||2)) / бt = 2 (F(t) - G(s)) * dF/ dt*
*бH/ бs = (б(||F(t) - G(s)||2)) / бs = 2 (F(t) - G(s)) * dG/ dt*

H=[2*(f – g)*df ; -2*(f – g)*dg];

<u>*function [d, t, s] = gradientna(F, G, t0, s0)*</u>

Funkcija gradientna.m ima kot vhodne argumente dve funkciji F in G ki sta bodisi krivulji bodisi ploskvi. Tocka t0 (funkcija F) in tocka s0 (funkcija G) sta zacetni tocki na krivuljah ali pa sta dva dvodimenzionalna vektorja (z dimenzijo 2x1) na ploskvami. Funkcija vrne najmanjšo razdaljo d in parametra t in s.

V vsaki iteraciji (ki gre od 1 do največ *maxit*) izracunamo vrednost funkcije H v točko (t0,s0), kjer funkcija začne gradientni spust. Gradientni spust izracunamo rekurzivno po formuli: $a_{k+1} = a_k - h*gradf(a_k)$, kjer je a_{k+1} novi približek, a_k prethodni približek, *gradf* e odvod ustrezne funkcije, *h* pa korak (v tem primeru uporabimo korak 0,01). Spremenljivka *tol* pove koliko je lahko najmanjši spust (ker se vsakič približuje enki). Torej najprej postavimo *maxit* in tol na default vrednost, potem se začne zanka:

```
for k=1:maxit

    [f,df,ddf]=feval(F,t0);
    [g,dg,ddg]=feval(G,s0);

    h1=2*(f-g)'*df;
    h2=-2*(f-g)'*dg;
    H=[h1; h2];

    if(norm(H)<tol)
        break;
    endif

     t0=t0-0.01*h1;
    s0=s0-0.01*h2;
```

```
end
t=t0;
s=s0;
r=(feval(F,t)-feval(G,s));
d=sqrt(sum(r.^2));
endfunction
```

2. Minimalno razdaljo lahko izračunamo tudi s pomočjo **newtonove metode**. Pri minimalni razdalji zveznica med točkama na obeh krivuljah mora biti pravokotna na obe tangenti. Torej skalarni produkti med razdaljo funkcije F po parametru t in G po parametru s, oziroma f(t)-g(s), in tangenti df(t) in dg(s) morata biti enaki 0. Torej:

*(F(t) - G(s)) * dF/dt = 0* in
*(F(t) - G(s)) * dG/ds = 0*.

Obe enačbi vstavimo kot dva stolpca *h1* in *h2* matrike *H*, torej računamo rešitev sistema *H(t,s)=0*. Izračunamo Jakobievo matriko *JH* in naredimo newtonov korak za t in s po formuli: $a_{k+1} = a_k - JH(a_k)^{-1} * H(a_k)$.

function [d, t, s] = newtonova(F, G, t0, s0)
Funkcija gradientna.m ima kot vhodne argumente dve funkciji F in G ki sta bodisi krivulji bodisi ploskvi. Tocka t0 (funkcija F) in tocka s0 (funkcija G) sta zacetni tocki na krivuljah ali pa sta dva dvodimenzionalna vektorja (z dimenzijo 2x1) na ploskvami. Funkcija vrne najmanjšo razdaljo d in parametra t in s.

Podobno kot pri gradientni, najprej postavimo vrednosti spremenljivk *maxit* in *tol*. Potem v vsaki iteraciji izracunamo vrednosti funkcij F in G.

```
for k = 1:maxit

[f,df,ddf]=feval(F,t0);
[g,dg,ddg]=feval(G,s0);

h1 = (f-g)'*df;
h2 = (f-g)'*dg;
H = [h1 ; h2];
```

Jacobievo matriko *JH* izačunamo tako da odvajamo *h1* in *h2* po oba parametra, torej dobimo matriko dimenzije *2x2*.

```
h1f = df'*df+((f-g)'*ddf);
h2f = -df'*dg;
h1g = dg'*df;
h2g = -dg'*dg+((f-g)'*ddg);
JH = [h1f  h2f; h1g  h2g];
```

Izvedemo korak Newtonove iteracije:

```
tocki=tocki0-JH\H;

if(norm(tocki0-tocki) < tol)
      break;
endif

t0=tocki(1);
s0=tocki(2);

end
t=t0;
s=s0;
r=(feval(F,t)-feval(G,s));
d=sqrt(sum(r.^2));
endfunction
```

3. Izpeljava rešitve za **ploskve**

Če F in G predstavljata ploskvi, potem bosta začetni približki *t0* in *s0* podani kot dvodimenzionalni vektor.

<u>gradientna.m</u>

Matrika H je sedaj odvisna od dva vektorja $t = [t_1 ; t_2]$ in $s = [s_1 ; s_2]$, in jo posplošimo:
$H(t_1,t_2,s_1,s_2) = ||F(t_1,t_2) - G(s_1,s_2)||^2$. Gradient matrike H je matrika dimenzije *4x1,* ki vsebuje odvode funkcije H po 4 parametrov t_1,t_2,s_1,s_2. Torej funkcijo v octave-u nadopolnimo tako da najprej preverimo dimenzije *t0* in *s0* in potem transponiramo oba stolpca *h1* in *h2* da bi se potem ustrezno odšteli, ker tokrat dobimo dve vrednosti.

```
if(rows(t0)==2 && rows(s0)==2)
    h1=h1';
     h2=h2';
endif
```

<u>newtonova.m</u>
V tem primeru imamo 4 enacbe za vsak parametar, oziroma:

$(F(t_1,t_2) - G(s_1,s_2)) * dF/dt_1 = 0,$
$(F(t_1,t_2) - G(s_1,s_2)) * dF/dt_2 = 0,$
$(F(t_1,t_2) - G(s_1,s_2)) * dG/ds_1 = 0$ in
$(F(t_1,t_2) - G(s_1,s_2)) * dG/ds_2 = 0.$

Poiščemo rešitev sistema $H(t_1,t_2,s_1,s_2) = 0$. *U octave-u funkcijo nadopolnimo:*

```
elseif(rows(t0)==2 && rows(s0)==2)

   h1=[ ((f-g)'*df(:,1)) ; ((f-g)'*df(:,2)) ];
   h2=[ ((f-g)'*dg(:,1)) ; ((f-g)'*dg(:,2)) ];

   H=[ h1; h2 ];
```

Matrika *H* je *4x1* matrika ki vsebuje vse 4 enačbe ki so opisani zgoraj. Za izračun Jakobievo matriko *JH*, bomo dobili matriko dimenzije *4x4,* zarad tega ker vsak element matrike *H* odvajamo po svak parametar. Matriko *JH* izračunamo:

```
JH= [df(:,1)'*df(:,1)+(f-g)'*ddf(:,1,1), df(:,2)'*df(:,1)+(f-g)'*ddf(:,1,2), -dg(:,1)'*df(:,1),        -dg(:,2)'*df(:,1);

     df(:,1)'*df(:,2)+(f-g)'*ddf(:,2,1), df(:,2)'*df(:,2)+(f-g)'*ddf(:,2,2), -dg(:,1)'*df(:,2),        -dg(:,2)'*df(:,2);

     df(:,1)'*dg(:,1),        df(:,2)'*dg(:,1),        -dg(:,1)'*dg(:,1)+(f-g)'*ddg(:,1,1), -dg(:,2)'*dg(:,1)+(f-g)'*ddg(:,1,2);

     df(:,1)'*dg(:,2),        df(:,2)'*dg(:,2),        -dg(:,1)'*dg(:,2)+(f-g)'*ddg(:,2,1), -dg(:,2)'*dg(:,2)+(f-g)'*ddg(:,2,2)];
```

Če matematično zapišemo recimo prvega elementa, izgleda tako:
$(бF/ бt_1)(t_1,t_2) * (бF/ бt_1)(t_1,t_2) + (F(t_1,t_2) - G(s_1,s_2)) * (б^2F/ б^2t_1)(t_1,t_2)$

Primer

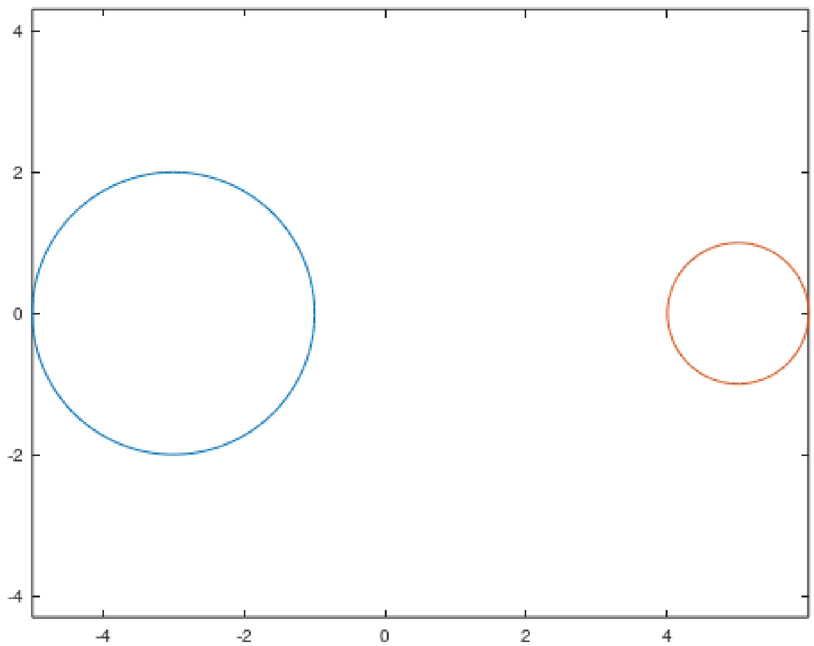
V octave-u definiramo dve funkciji, kroznica1.m in kroznica2.m.

```
function [ f, df, ddf ]=kroznica1(t)
  f=[2.*sin(t) - 3; 2.*cos(t); 0];
  df=[ 2.*cos(t); -2.*sin(t); 0];
  ddf= [ -2.*sin(t); -2.*cos(t); 0];
endfunction

function [ g, dg, ddg ]=kroznica2(s)
  g=[ sin(s)+ 5; cos(s); 0 ];
  dg=[ cos(s) ; -sin(s) ; 0 ];
  ddg= [ -sin(s) ; -cos(s); 0 ];
endfunction
```

Narišemo obe kroznici:

```
t = linspace(0,2*pi,100)';
cx1 = 2.*sin(t)-3;
cy1 = 2.*cos(t);
plot(cx1,cy1);
hold on
cx2 = sin(t)+5;
cy2 = cos(t);
plot(cx2,cy2);
```



Najmanjša razdalja med kroznici je 5, za t = pi/2 in s = -pi-2.
Kot začetni približek vzamemo $t_0 = 1$ in $s_0 = -1$.

```
[d ,t ,s] = gradientna(@kroznica1, @kroznica2, 1, -1)
d =  5
t =  1.5708
s = -1.5708

[d, t, s] = newtonova(@kroznica1, @kroznica2, 1, -1)
d =  5
t =  1.5708
s = -1.5708
```