Obligatorisk oppgave 1
MAT110 VÅREN 2009
Løsningsforslag

Øyvind Ryan (oyvindry@ifi.uio.no)

5. februar 2009

a)

Vi regner ut

$$
\begin{aligned}
\mathbf{F}'(x,y) &= \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix} \\
&= \begin{pmatrix} -ye^{-xy} & -xe^{-xy}-1 \\ 4x^3 & 4y^3 \end{pmatrix}.
\end{aligned}
$$

b)

Koden under tegner grafen til f og g hver for seg, og i samme koordinatsystem:

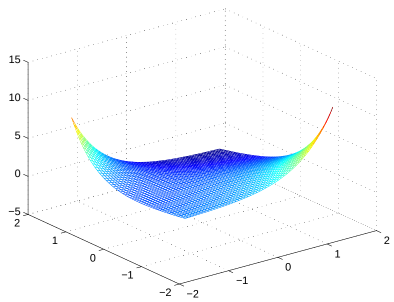
```
r=linspace(-1.5,1.5,50);
s=linspace(-1.5,1.5,50);
[x,y]=meshgrid(r,s);
f=exp(-x.*y)-y;
g=x.^4+y.^4-2;
mesh(x,y,f);
figure(2)
mesh(x,y,g);
figure(3)
mesh(x,y,f)
hold on
mesh(x,y,g)
```

De resulterende plottene se du i figur 1

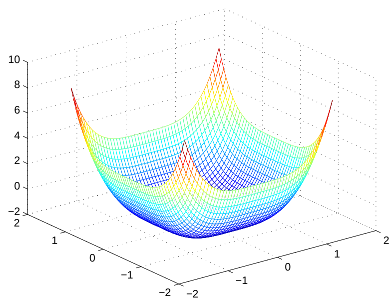
c)

La oss først se på nivåkurvene for $c = 0$. Disse kan plottes slik

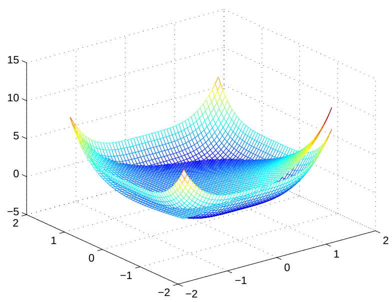
```
contour(x,y,f,[0 0]);
figure(2)
```



(a) Plott av $f(x, y)$

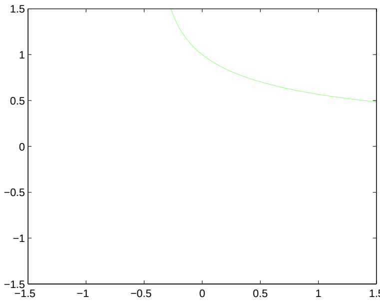


(b) Plott av $g(x, y)$

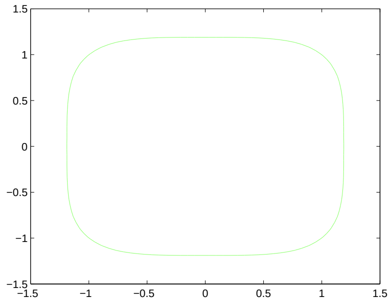


(c) Plott av f og g sammen

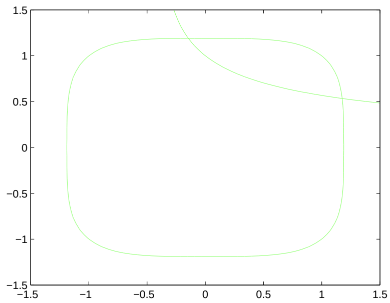
Figur 1: Plottene i b).



(a) Plott av nivåkurven $c = 0$ for f



(b) Plott av nivåkurven $c = 0$ for g



(c) Plott av nivåkurvene for f og g sammen

Figur 2: Plottene i c).

```
contour(x,y,g,[0 0]);
figure(3)
contour(x,y,f,[0 0]);
hold on
contour(x,y,g,[0 0]);
```

De resulterende plottene se du i figur 2. Vi ser her to skjæringspunkter, der både f og g er 0. Disse er (grovt) tilnærmet $(1.2, 0.6)$ og $(-0.2, 1.2)$. Plotter vi nivåkurvene for $c = 1, 2$ også kan vi bytte ut `[0 0]` med `[0 1 2]` i koden ovenfor.

d)

Anta $\mathbf{F}'(\mathbf{x})$ er inverterbar når $\mathbf{x}$ er punktet følgen konvergerer mot. Vi har at

$$\begin{aligned} \mathbf{x}_{n+1} &\to \mathbf{x} \\ \mathbf{x}_n - (\mathbf{F}'(\mathbf{x}_n))^{-1}\mathbf{F}(\mathbf{x}_n) &\to \mathbf{x} - (\mathbf{F}'(\mathbf{x}))^{-1}\mathbf{F}(\mathbf{x}). \end{aligned}$$

Siden de to størrelsene på venstre side er like har vi at $\mathbf{x} = \mathbf{x} - (\mathbf{F}'(\mathbf{x}))^{-1}\mathbf{F}(\mathbf{x})$, slik at $(\mathbf{F}'(\mathbf{x}))^{-1}\mathbf{F}(\mathbf{x}) = \mathbf{0}$. Siden $(\mathbf{F}'(\mathbf{x}))^{-1}$ er inverterbar har vi da at $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, slik at $\mathbf{x}$ er et nullpunkt for $\mathbf{F}$.

e)

Prorammet nedenfor beregner følgen $\mathbf{x}_n$:

```
function [x,y]=beregnxn(a,b,N)
  x(1)=a;
  y(1)=b;
  for n=1:(N-1)
    fxn = [exp(-x(n)*y(n))-y(n) ; x(n)^4+y(n)^4-2];
    dfxn = [ -y(n)*exp(-x(n)*y(n))    -x(n)*exp(-x(n)*y(n)) - 1 ; ...
             4*x(n)^3    4*y(n)^3];
    outval = [ x(n) ; y(n) ] - inv(dfxn)*fxn;
    x(n+1)=outval(1);
    y(n+1)=outval(2);
  end
```

f)

I Matlab skriver du her `beregnxn(a,b,N)` for de forskjellige verdiene som er oppgitt. Vi summerer opp det du skal få:

For $(-1, 1)$: Matlab stopper på $x = -0.1457, y = 1.1891$ hvis vi velger $N = 14$ eller mer. For $(-0.5, 0.5)$ ser det ut til at vi nærmer oss samme løsning, men nå trenger vi 10 iterasjoner.

For $(1, 1)$: Matlab stopper på $x = 1.1770, y = 0.5336$ hvis vi velger $N = 6$ eller mer. For $(0.5, 0.5)$ ser det ut til at vi nærmer oss samme løsning, men nå trenger vi 10 iterasjoner.

Koden for å plotte de forskjellige følgende blir slik:

```
[x1,y1]=beregnxn(-1,1,20);
[x2,y2]=beregnxn(1,1,20);
[x3,y3]=beregnxn(0.5,0.5,20);
[x4,y4]=beregnxn(-0.5,0.5,20);
plot(x1,y1,'r',x2,y2,'g',x3,y3,'b',x4,y4,'y');
legend('Start (-1,1)','Start (1,1)','Start (0.5,0.5)','Start (-0.5,0.5)');
```

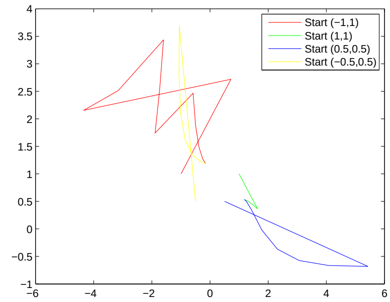
Det resulterende plottet ser i figur 3.

g)

Du kan kjøre programmet fem ganger med tilfeldig genererte startverdier slik:

```
for k=1:5
  [x,y]=beregnxn(rand,0.5+rand,20)
end
```

Legg spesielt merke til at `0.5+rand` gir et tilfeldig generert tall mellom 0.5 og 1.5, siden `rand` returnerer et tilfeldig generert tall mellom 0 og 1.



Figur 3: Plottet i f).

Hvis du har levert inn kode skrevet i Python

Funksjonen `beregnxn(a,b,N)` som du skrev i oppgave e) vil i Python bli seende slik ut:

```
from math import *
from numpy import *

def beregnxn(a,b,N):
  x = zeros(N)
  y = zeros(N)
  x[0]=a
  y[0]=b
  for n in xrange(N-1):
    ael = exp(-(x[n])*(y[n]))-(y[n])
    bel = x[n]**4+y[n]**4-2
    fxn = matrix([[ael],[bel]])
    dfxn = matrix([[-y[n]*exp(-x[n]*y[n]),-x[n]*exp(-x[n]*y[n]) - 1],\
                   [4*x[n]**3,4*y[n]**3]])
    outval = matrix([[x[n]],[y[n]]]) - linalg.inv(dfxn)*fxn
    x[n+1]=outval[0,0]
    y[n+1]=outval[1,0]
  return [x,y]
```

Resten av koden i fasiten kan bli seende slik ut i Python:

```
from math import *
from numpy import *
from beregnxn import *
from scitools.easyviz import *

# b):
r=arange(-1.5,1.5,0.05,float)
s=arange(-1.5,1.5,0.05,float)
x,y=meshgrid(r,s,sparse=False,indexing='ij')
f=exp(-x*y)-y
```

```
g=x**4+y**4-2
figure(1)
mesh(x,y,f)
figure(2)
mesh(x,y,g)
figure(3)
mesh(x,y,f)
hold('on')
mesh(x,y,g)
hold('off')

# c):
figure(4)
contour(x,y,f,[0,0])
figure(5)
contour(x,y,g,[0,0])
figure(6)
contour(x,y,f,[0,0])
hold('on')
contour(x,y,g,[0,0])

# f):
x1,y1=beregnxn(-1,1,20)
x2,y2=beregnxn(1,1,20)
x3,y3=beregnxn(0.5,0.5,20)
x4,y4=beregnxn(-0.5,0.5,20)
figure(7)
plot(x1,y1,'r',x2,y2,'g',x3,y3,'b',x4,y4,'y')
legend('Start (-1,1)','Start (1,1)','Start (0.5,0.5)','Start (-0.5,0.5)')

# g):
for k in range(5):
  x,y=beregnxn(random.rand(),0.5+random.rand(),20)
  print x,y
```