Noen MATLAB-koder

Fredrik Meyer

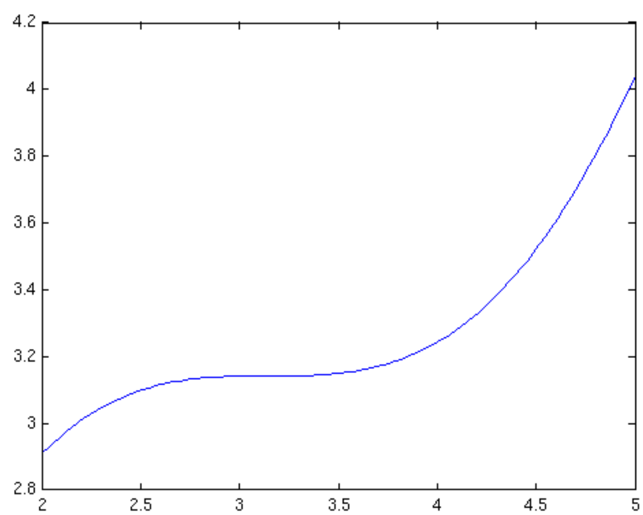
23. april 2013

1 Plotte en *vanlig* funksjon

Anta at $f : [a,b] \to \mathbb{R}$ er en "vanlig" funksjon. La for eksempel $f(x) = \sin x + x$ for x i intervallet $[2,5]$. Da kan vi bruke følgende tre linjer for å plotte f:

```
>> t = linspace(2,5, 100);
>> f = sin(t) + t;
>> plot(f)
```

Resultatet er under:



Figur 1: Funksjonen $f(x) = \sin t + t$.

2 Plotte en kurve i 2 dimensjoner

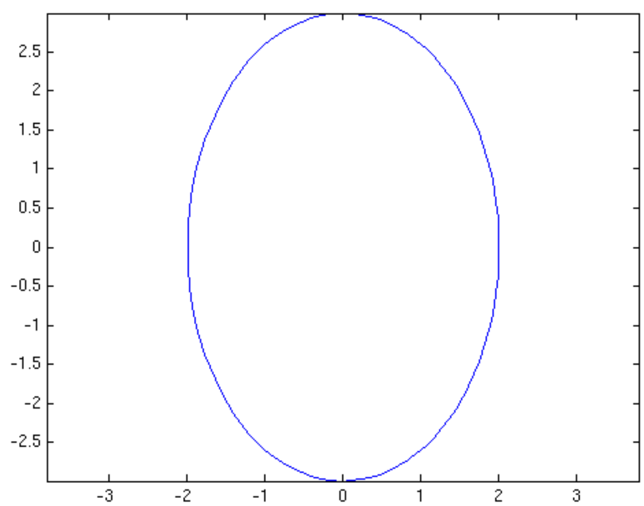
Anta at $\mathbf{r} : [a, b] \to \mathbb{R}^2$ er en kurve i to dimensjoner. La for eksempel $\mathbf{r}(t)$ være gitt ved

$$\mathbf{r}(t) = \big(2\cos t, 3\sin t\big),\ t \in [0, 2\pi]$$

Dette er en ellipse med sentrum i origo og halvakser 2 og 3. Vi kan plotte med følgende MATLAB-kode:

```
>> t = linspace(0,2*pi,100);
>> x = 2*cos(t);
>> y = 3*sin(t);
>> plot(x,y); axis equal
```

Resultatet:



Figur 2: Ellipse.

3 Plotte en kurve i 3 dimensjoner

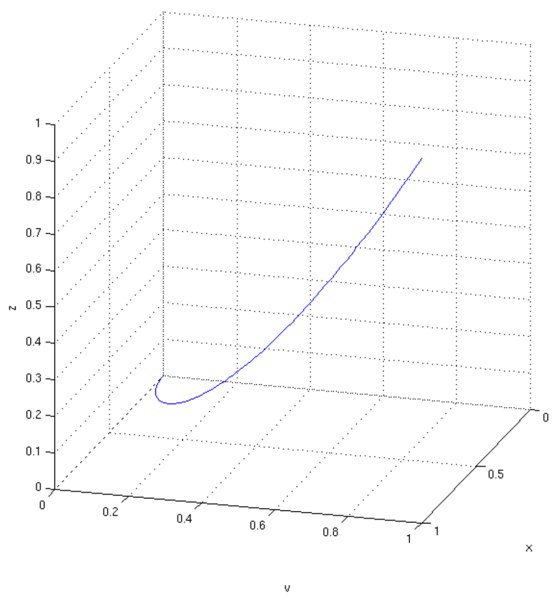
Anta at $\mathbf{r} : [a, b] \to \mathbb{R}^3$ er en parametrisert kruve i tre dimensjoner. La for eksempel $\mathbf{r}(t)$ være gitt ved

$$\mathbf{r}(t) = \big(t, t^2, t^3\big),\ t \in [0, 1]$$

Dette er en såkalt "vridd kubikk"og er et elsket objekt av mange geometre. Vi kan plotte med følgende kode:

```
>> t = linspace(0,1,100);
>> i = t;
>> j = t.^2;
>> k = t.^3;
>> plot3(i,j,k); axis equal
>> grid on
>> xlabel(’x’); ylabel(’y’); zlabel(’z’);
```

Resultatet:



Figur 3: Vridd kubikk.

4 Numerisk integrasjon, én variabel

Om $f : [a, b] \to \mathbb{R}$ er en funksjon, kan MATLAB integrere denne numerisk. Om for eksempel

$$f(x) = x\sqrt{1 + 5x^2 + 4x^3},$$

og vi har lyst å integrere over intervallet $[-1, 1]$, så kan vi bruke koden

```
>> sym x;
>> quad('x.*sqrt(1+5*x.^2+4*x.^3)', -1, 1)

ans =

    0.3995
```

5 Plotte en funksjon av 2 variable

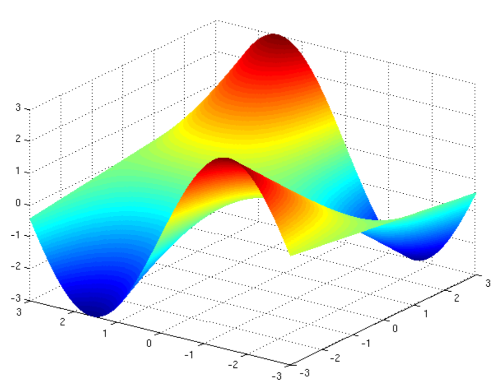
Anta at $f : \mathbb{R}^2 \to \mathbb{R}$ er en funksjon av to variable (i FVLA kalles dette for et *skalarfelt*). La for eksempel $f(x, y)$ være gitt ved

$$f(x, y) = x \sin(y), \quad \text{for } (x, y) \in [-3, 3] \times [-3, 3]$$

Vi kan bruke følgende MATLAB-kode for å plotte dette:

```
>> f = inline('x.*sin(y)', 'x','y');
>> [X,Y] = meshgrid(-3:.01:3, -3:.01:3);
>> Z  = f(X,Y);
>> mesh(X,Y,Z)
```

I første linje lager vi funksjonen. Husk punktum før gange-/deletegn for at operasjonene skal være komponentvise. I neste linje lager vi et `meshgrid`. Tenk på dette som en haug med (x, y)-punkter. Her går både x- og y-koordinatene fra -3 til 3 med et mellomrom på 0.01. Dette gjør at MATLAB blir litt treg, men bildet blir mye bedre. Se under:



Figur 4: Mesh.

6 Når du ikke vil dele på null

Dette kan ses på et delvis løsningsforslag til oppgave 3.7.6 i FVLA. La $f : \mathbb{R}^2 \to \mathbb{R}$ være funksjonen definert ved

$$f(x,y) = \begin{cases} \frac{x^2y}{x^4+y^2} & \text{for } (x,y) \neq (0,0) \\ 0 & \text{ellers} \end{cases}$$

Vi ønsker å plotte f i MATLAB. Vi kan ikke naivt skrive denne inn slik som i forrige seksjon fordi vi da får nulldivisjon. Én måte å løse dette på, er å definere en egen funksjon i en m-fil. Vi lager en fil som vi kaller `oppg.m`:

```
function tall = oppg(x,y)
if (x == 0) & (y == 0)
    tall = 0;
else
    tall = (x.^2.*y)./(x.^4+y.^2);
end
```

Er $(x,y) = (0,0)$, spytter funksjonen bare automatisk ut 0, så vi unngår nulldivisjon. Vi kan plotte med kommandoene:

```
>> [x,y] = meshgrid(-2:0.05:2, -2:0.05:2);
>> z = f(x,y);
>> z = oppg(x,y);
>> surfc(x,y,z)
```

Resultatet er figur 5.

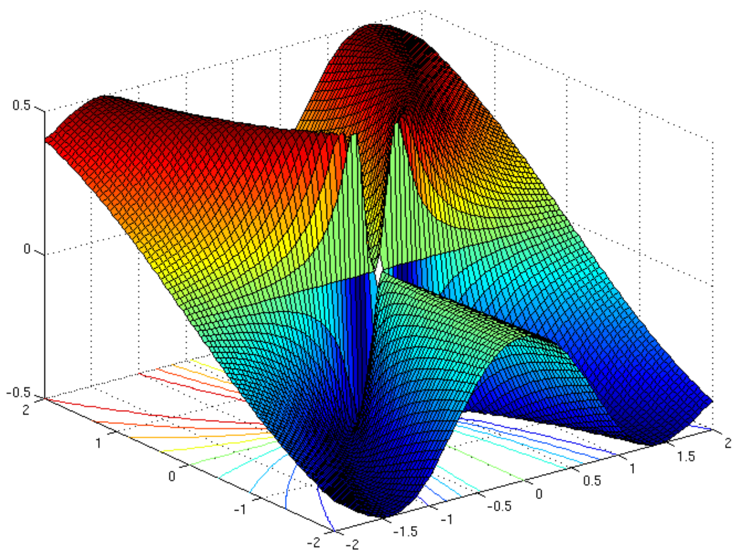
7 Regne Jacobi-matriser og gradienter

Om $f : \mathbb{R}^n \to \mathbb{R}^m$ er en deriverbar funksjon, så kan vi regne ut Jacobimatrisen i MATLAB. Anta for eksempel at $f : \mathbb{R}^2 \to \mathbb{R}^3$ er definert ved

$$f(x,y) = (x^2, xy, y^2).$$

(for de interesserte, så kalles dette for *den andre Veronese-embeddingen av* $\mathbb{P}^1$ *inn i* $\mathbb{P}^2$). I MATLAB skriver vi bare:

```
>> syms x y
>> f = [x^2, x*y, y^2];
>> jacobian(f)
```



Figur 5: Oppgave 3.7.6.

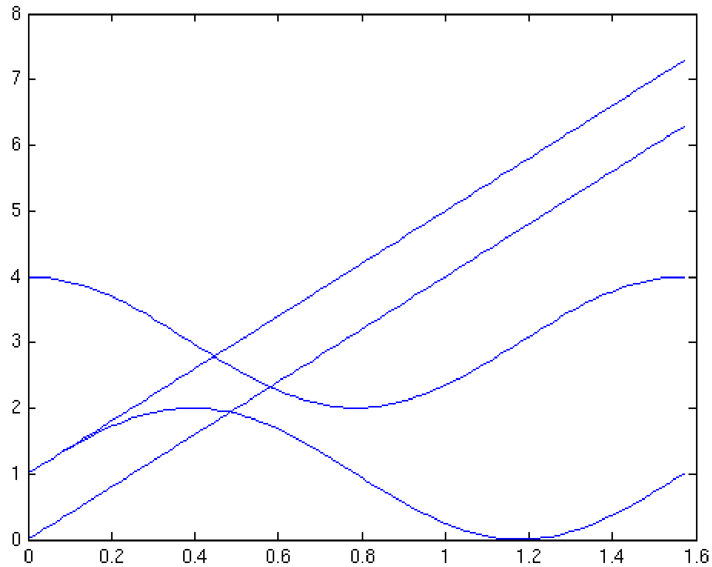
```
ans =

[ 2*x,   0]
[   y,   x]
[   0, 2*y]
```

Husk at en gradient er det samme som Jacobi-matrisen til en funksjon $f : \mathbb{R}^n \to \mathbb{R}$, så du kan bruke samme kode som over.

8 Dobbeltintegraler avgrenset av funksjoner

Anta at vi ønsker å integrere funksjonen $f(x) = xy\sin(x^2y)$ over området avgrenset av funksjonsgrafene til $y = x$, $y = x + 1$ og $y = \sin x + 1$ og $y = 3\cos x + 3$. Dette kan vi gjøre i MATLAB med funksjonen `dblquad`. Vi viser først hvordan dette området ser ut i Figur 6.

Dette kan vi regne ut i MATLAB med kommandoene (`f` må skrives på én linje!):



Figur 6: Området vi integrerer over.

```
>> f = @(x,y)(x.*y.*sin(x.^2.*y)).*(x <= y).*(y <= x+1).*
       ( y <= cos(x)+3).*(sin(x)+1 <= y)

>> dblquad(f,0,3,0,4)

ans =
   0.077338484886870
```

Det er viktig at grensene i `dblquad` er *større* enn området du skal integrere over.

9 Løse ligningssystemer

Anta at du skal løse følgende ligningssystem:

$$\begin{aligned} 2x + 3y &= 5 \\ x - 2y &= 2 \end{aligned}$$

Dette kan du gjøre i MATLAB ved å følgende kommandoer:

```
>> A = [2 3 5; 1 -2 2]
A =
     2     3     5
     1    -2     2
>> rref(A)

ans =

   1.000000000000000                   0   2.285714285714286
                   0   1.000000000000000   0.142857142857143
```

Dette betyr at $x = 2.2857\cdots$ og $y = 0.1428\cdots$. Vil du heller a svarene som brøk, kan du skrive `format rat` først. Vi får da:

```
>> format rat
>> rref(A)
ans =
       1              0              16/7
       0              1               1/7
```

... som er endel penere å se på.

10 Generelle ting å huske

Skal du plotte flere grafer i samme vindu, kan du skrive `hold on` etter første plott. Da havner de over hverandre.

Om du vil ha like proporsjoner på aksene, kan du skrive `axis equal`. Det er ikke alltid du vil gjøre dette, for eksempel på $f(x)$ gir veldig høye verdier for små x (hva skjer da??).

Når du skriver formler og funksjoner vil vi gange komponentvis - for å få MATLAB til å gjøre dette, må du huske å skrive `.*` når du ganger sammen, i stedet for bare `*`.