Matematikk 1000

Øvingsoppgaver i numerikk – leksjon 2

Funksjoner og plotting

I denne øvinga skal vi først og fremst lære oss å lage plott i MATLAB. I tillegg skal vi lære oss hvordan vi manøvrerer oss omkring i ulike mapper.

Ellers minner vi om at der er mange MATLAB-ressurser tilgjengelig. Vi har heftet "MATLAB Primer". Man kan søke etter dokumentasjon i MATLAB selv, eller man kan "google" etter hjelp. Sist, men ikke minst, man kan spørre lærerne om hjelp.

Det forutsettes at man har gjort det forrige oppgavesettet.

I en del av disse oppgavene er det mye tekst. Det betyr ikke nødvendigvis at det krever mye arbeid å gjøre dem. Om det oppleves som mye arbeid: Spør om hjelp!

Vi har sett at en hel del funksjoner allerede er lagt inn i MATLAB. Dette gjelder for eksempel de trigonometriske funksjonene – med inversfunksjoner. De heter `sin, cos, tan, asin, acos` og `atan`. Eksponentialfunksjonen, e^x, og den naturlige logaritme-funksjonen, $\ln x$, finnes også; de heter `exp` og `log`. I tillegg kan vi lage våre egne funksjoner, som vi skal se mot slutten i dette oppgavesettet.

Oppgave 1 – Å lage et plott

a) La vektoren `x` starte med 0 og så gå i steg på 0.1 opp til 7π. (Med fare for å være over-tydelig: Denne tilordninga skal utføres i MATLAB.)

b) La vektoren `y` bestå, elementvis, av sinusverdiene av vektoren `x`. Det gjør du slik i MATLAB: '`» y=sin(x);`'. Denne funksjonen kan ta både skalare (tall-) argumenter og vektorer, slik som her.

c) Plott sinusfunksjonen: '`» plot(x,y)`'.

d) Undersøk hva som skjer når du skriver '`plot(x,y,'r')`' og '`plot(x,y,'linewidth',3)`'. Gå gjerne inn på dokumentasjonen for `plot`-kommandoen for å se hvilke andre muligheter du har. Ser du hvordan man kan få grafen stipla i stedet for heltrukket?

e) Bruk MATLAB til å plotte følgende funksjoner:

i) $f(x) = \sqrt{x} + \ln x, \quad D_f = [1/4, 5]$.

ii) $g(x) = \arctan x, \quad D_g = [-10, 10]$.
("$\arctan x$" er det samme som "$\tan^{-1} x$".)

iii) $h(x) = \cos(2x) \cdot e^{-x^2/10}$. Velg selv et passende intervall for x her.

Oppgave 2 – Å lagre et plott

Når vi skal lagre noe, er det selvsagt viktig at vi har kontroll på hvor det blir lagra. Det er nok en god idé å opprette ei mappe for de MATLAB-relaterte filene som etterhvert blir oppretta i dette kurset. Det vil nok bli nødvendig å dele denne inn i flere under-mapper også. Det aller meste av det som har med lagring å gjøre, kan gjøres ved hjelp av klikkbare menyer. Men det kan også gjøres med kommandoer i kommandovinduet. Selv om de fleste antakeligvis vil foretrekke "klikking", vil vi kort forklare hvordan ting gjøres fra kommandovinduet også.

a) Når du starter MATLAB, vil du i utgangspunktet bli plassert et eller annet sted i fil-systemet ditt. Hvor dette er, ser du i mappevinduet. (Alternativt kan du i kommandovinduet finne ut hvilken mappe du er i ved å skrive `pwd` – *print working directory*.)

   Ta utgangspunkt i ett av plottene du lagde i oppgave 1). Dette kan du lagre ved å klikke `Save` i `File`-menyen oppe til venstre i viduet med plottet. Gjør dette og pass på at figuren blir lagra i ei passende mappe. Figuren bli da lagra som ei `fig`-fil – MATLABs eget figurformat. Prøv gjerne å lukke figuren og åpne den igjen.

b) Man kan fint skrive ut grafer direkte fra MATLAB. Men ofte kan det være en fordel å lagre figuren som ei bildefil av et eller annet slag først. Dette er nødvendig å gjøre om du ønsker å ta med plott i andre typer dokumenter – som for eksempel tekstbehandlingsfiler (OpenOffice, Word etc.). De vanliste tilgjengelige formatene er `jpg/jpeg`, `png` og `pdf`[1].

   Når man skal lagre et plot som ei bildefil, kan man også gjøre dette ved å bruke `Save` eller `Save As` i fil-menyen oppe til venstre i figur-vinduet. Du velger det formatet du vil ha i `Save as type`-menyen nederst i det vunduet som dukker opp – og passer på å plassere fila i den katalogen du vil ha den i før du trykker `Save`.

   Alternativt kan det hele gjøres fra kommandolinja:

```
>> print -f<figurnummer> -d<format> <Filnavn>.<format>}
```

[1] Forkortelsene står for henholdsvis *Joint Photographic Expert Group*, som kom opp med `jpeg`-formatet, *Portable Network Graphics* og *Portable Document Format*.

Om du for eksempel vil at figur 2 – man kan ha flere figurer oppe samtidig – skal eksporteres til `png`-formatet og hete `FigurenMin` (i tillegg til "etternavnet" `png`), skriver du

```
>> print -f2 -dpng FigurenMin.png
```

Når du gjør dette blir figuren lagra i den arbeidsmappa du er i; om du vil lagre den et annet sted, kan du manøvrere deg dit[2].

Velg en av metodene over og lagre plottet ditt som ei `png`-fil i ei passende mappe. Forsøk å åpne `png`-fila med en eller annen bilde-applikasjon.

Oppgave 3 – Flere grafer samtidig

a) Lag et plott av grafen til funksjonen $f(x) = x^2 + 1$. Velg selv hvilket intervall x skal tilhøre og hvor fin inndeling du vil ha på den tilsvarende vektoren. (Tips: Gjør som over. Lag først en vektor med de aktuelle x-verdiene ved å bruke skrivemåten `a:d:b` – du velger selv start og slutt, `a` og `b`, og *steglengda* `d`. Så bruker du denne x-vektoren til å lage en y-vektor som du kan bruke i `plot`-kommandoen. Pass på at `d` blir så liten at figuren ikke ser "hakkete" ut.)

b) Lag en vektor med funksjonsverdier for funksjonen $g(x) = e^x$ med den samme x-vektoren som i a). Lag et plott av $g(x)$ også. Lag en figur som inneholder både grafen til f og g samtidig. Dette kan gjøres på to måter. Enten slik:

```
>> plot(x,y,x,z)
```

eller slik:

```
>> plot(x,y)
>> hold on
>> plot(x,z)
>> hold off
```

Her har vi valgt å kalle vektorene med funksjonsverdier for f og g henholdsvis `y` og `z`.

Oppgave 4 – Funksjonsfiler

Vi skal nå se hvordan vi kan lage våre egne funksjoner i MATLAB. Disse kan brukes på samme måte som de funksjonene MATLAB har inne fra før. Vi skal

[2]Man kan som sagt manøvrere mellom ulike mapper i mappevinduet. Men også dette kan gjøres i kommandovinduet. Man bruker å så fall kommandoen `cd`, *change directory*, på samme måte som i `dos`-vinduet i Windows eller i terminalvinduet i Linux/Unix.

ta utgangspunkt i funksjonen $f(x) = \sin(2x) - x^2$. Først åpner vi en *teksteditor*. Det mest naturlige er nok å bruke MATLAB sin egen. Den kan åpnes ved å trykke på "New Script"-knappen oppe til venstre. Du får da opp et vindu hvor du kan skrive inn tekst.

a) Skriv av følgende i teksteditoren:

```
function F=FunksjonenMin(x)

% Funksjonen f(x)=sin(2x) - x^2.
% Funksjonen tar bare skalarer som input.

F=sin(2*x)-x^2;
```

Lagre fila i en passende mappe og kall den `FunksjonenMin.m`. Om du er i den samme mappa i MATLAB, kan du regne ut funksjoneverdier for denne funksjonen på akkurat samme måte som for funksjonene over; $f(5)$ kan for eksempel finnes ved å skrive '`» FunksjonenMin(5)`' i kommandovinduet.

Regn ut $f(x)$ for noen x-verdier du velger selv på denne måten.

Man kan lage funksjonsfiler på flere måter. Men to ting kommer man ikke utenom:
1) Den første linja *skal* ha denne strukturen:
`function <Navn på ut-variabel>=<Navn på funksjon>(<Liste med inn-variabler>)`
I eksempelet over, ser vi at ut-variabelen har fått navnet `F` og at funksjonen har fått navnet `FunksjonenMin`. Dette navnet bør være det samme som navnet på fila (utenom '`.m`'). Til sist, inni en parantes, lister man opp de variablene som skal inn. I dette tilfellet er det bare én, `x`, men det er ingenting i veien for at det kan være flere[3].
2) Til sist i funksjonsfila *skal* ut-variabelen, `F` i vårt tilfelle, ha blitt tilordna.

b) Hva skjer hvis du skriver '`» help FunksjonenMin`'?

c) Hvis vi skal plotte en funksjon gitt ved ei funksjonsfil, er det en stor fordel om funksjonsfila kan ta vektorer som inn-variabel på en slik måte at den gir som svar en vektor (ut-variabel) som består av funksjonsverdien av hvert element i vektoren vi gir inn. Vi har allerede sett at for eksempel `sin` – funksjonen har denne egenskapen; hvis `x=[1 2 3]`, vil '`» sin(x)`' gi vektoren $[\sin 1, \sin 2, \sin 3]$ som svar.

Med ei ørlita endring kan funksjonen over bli i stand til dette. Gjør denne endringa og plott funksjonen. Velg selv hvilket intervall argumentet skal gå over og hvilken steglengde du vil ha.

d) Lag ei funksjonsfil for en eller annen elementær funksjon du velger selv og bruk denne til å plotte grafen til funksjonen.

[3] Det kan gjerne være flere ut-variable også.

Oppgave 5 – Delt forskrift – `if`-satser

I matematikken har vi sett av funksjoner kan bli gitt med *delt forskrift* – altså at funksjonen er gitt ved ett uttrykk for visse x-verdier og et annet uttrykk for andre x-verdier. Det kan også være snakk om flere enn to ulike uttrykk. I denne oppgava skal vi se hvordan vi kan lage funksjonsfiler for, eller *implementere*, slike funksjoner. Men først skal vi se litt på logiske utsagn i MATLAB.

a) Som tidligere nevnt, betyr '`=`' *tilordning* – ikke likhet – i MATLAB. Likhet skrives slik: '`==`'. Større enn og mindre enn, derimot skrives som normalt. 'Er ulik', og ikke-strenge ulikheter kan skrives slik: '`~=`', '`<=`' og '`>=`'. Forsøk å skrive noen sanne og noen usanne logiske påstander i kommandovinduet, som for eksempel $3 = 2$, $-1 < 0$, og $-1 \neq 0$, og se hva du får til svar. Forsøk gjerne å kombinere med *eller* og *og* også. 'Eller' kan skrives som '`|`' og 'og' kan skrives som '&'. For eksempel kan påstanden $x \in [-2, 1]$ skrives slik: `x>=-2 & x<=1`.

   I en slik sammenheng, hva betyr "0" og "1"?

b) Skriv av denne funksjonsfila i editoren din, lagre den og kall den '`DeltForskrift.m`':

```
function F=DeltForskrift(x)

% Her bør det stå noe om hvilken funksjon det er snakk om

if x<2
  F=cos(pi*x)+2;
else
  F=x^2-2;
end
```

   Regn ut noen funksjonverdiene for noen x-verdier du velger selv. Hvilken funksjon er dette ei implementering av? Merk at denne fuksjonsfila bare tar skalarer (tall) som input (ikke vektorer)[4].

Litt mer om `if`-satser til slutt (ingen oppgaver):

De enkleste `if`-satsene har denne strukturen

```
if <logisk påstand>
  <Utfør kommandoer>
end
```

[4] Her kan det være fristende å forandre funksjonsuttrykkene slik at de tar vektor-argumenter. Om vi gjør denne justeringa, vil det *se ut til* at funksjonen tar vektor argument. Men dette er ei fallgruve; hvis `x` er en vektor, vil bare det siste elementet i vektoren bli brukt i `if`-satsen. Dette problemet er selvsagt løsbart...

Kommandoen eller kommandoene vil bare bli utført hvis den logiske påstanden er sann. Legg merke til at vi har gjort et lite "innhopp" i teksten i linja mellom `if` og `end`. Dette gjør funksjonsfila, eller hva det måtte være, mye mer oversiktlig. Så vi anbefaler at du også legger deg til denne vanen.

Denne strukturen er også mye brukt:

```
if <logisk påstand>
  <Utfør kommandoer>
else
  <Utfør andre kommandoer>
end
```

Vi kjenner denne strukturen igjen fra funksjonsfila over. Her blir altså "andre kommandoer" utført i stedet for "kommandoer" dersom den logiske påstanden er feil. I eksempelet over er "andre kommandoer" tilordninga '`F=x ^ 2-2;`', mens "kommandoer" er tilordninga '`F=cos(pi*x)+2;`'.

En tredje vanlig struktur er denne:

```
if <logisk påstand>
  <Utfør kommandoer>
elseif <annen logisk påstand>
  <Utfør andre kommandoer>
else
  <Utfør et tredje sett av kommandoer>
end
```

Her kan man bygge på med så mange `elseif`-satser man bar vil etter hverandre.

For spesielt interesserte kan det også være verd å nevne at MATLAB tillater en noe mer kompakt måte å definere funksjoner på. Man kan for eksempel bruke `inline`-funksjonen til å definere funksjoner direkte i kommandovinduet. Dette kan også gjøres bruk av @-symbolet. Spør gjerne om dette på rekneøving om du er interessert i vite mer.