Matematikk 1000

Øvingsoppgaver i numerikk – leksjon 4

Funksjonsfiler

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 i dette oppgavesettet.

Oppgave 1 – Funksjonsfiler

a) Vi skal her lage ei funksjonsfil for funksjonen $f(x) = \sin(2x) - x^2$. Det kan gjøres ved å åpne teksteditoren og skrive inn følgende:

```
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 samme mappe i MATLAB, kan du regne ut funksjoneverdier for denne funksjonen på akkurat samme måte som for funksjonene over; '`>FunksjonenMin(5)`' o.s.v. Regn ut $f(x)$ for noen x-verdier du velger selv på denne måten.

Man kan lage funksjonsfiler på flere måter. Men noen 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[1].
2) Til sist i funksjonsfila skal ut-variabelen, `F` i vårt tilfelle, tilegnes.

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

c) Hvis vi skal plotte en funksjon, 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 plott grafen til funksjonen.

Oppgave 2 – 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 en forklaring om hva funksjonen gjør

if x<2
```

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

```
  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)[2].

c) De enkleste `if`-satsene har denne strukturen

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

Kommandoen eller kommandoene vil bare bli utført hvis den logiske påstanden er sann. Legg merke til at vi, akkurat som med `for`-løkker, har gjort et lite "innhopp" i teksten i linja mellom `if` og `end`. Dette gjør funksjonsfila, eller hva det måtte være, mye lettere å holde oversikt over. 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
```

[2]Her kan det være fristende å forandre funksjonsuttrykkene slik at de tar vektorargumenter. 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...

Med utgangspunkt i dette, lag ei funksjonsfil som implementerer denne funksjonen:

$$a(t) = \begin{cases} 0, & t < 10 \\ 9.8e^{-0.2(t-10)}, & 10 \leq t < 30 \\ -50e^{-1.2(t-30)}, & t \geq 30 \end{cases} .$$

Funksjonen er en modell for akselerasjonen til en fallskjermhopper. (a er gitt i $\mathrm{m/s^2}$ og t er gitt i sekund.) Hvilke faser beskriver funksjonen; hva skjer når $t = 10$ og når $t = 30$?

Til sist kan det være verd å nevne at MATLAB tillater en noe mer kompakt måte å definere funksjoner på. Denne skrivemåten, som gjør bruk av @-symbolet, gjør at vi kan definere funksjoner inni skript uten å lage noen egen funksjonsfil for funksjonen. Spør gjerne om du ønsker å vite mer om dette.