Programmeringsteknik och Matlab
Övning 1

Övningsgrupp 2 (Sal Q22/E32)

Johannes Hjorth
hjorth@nada.kth.se
Rum 4538 på plan 5 i D-huset
08 - 790 69 02

Kurshemsida:
http://www.nada.kth.se/kurser/kth/2D1312

Övningsanteckningar:
http://www.nada.kth.se/∼hjorth/teaching/prgi05

Dagens program

Kort introduktion till matlab.

- Att använda variabler
- Matriser och vektorer
- Plotta grafer (`plot`, `title`, `hold`)
- Skriva funktioner
- Flödeskontroll (`if`, `for`, `while`)
- Lösa ekvationssystem

Tips: Matlab har bra hjälp. För att få reda på mer om till exempel `hold` skriver ni `help hold`.

Administrativt

Det är viktigt att ni har registrerat er på kursen för att vi ska kunna rapportera in er.

```
res checkin prgi05
```

Kontrollera att ni har fått `Lab1` inrapporterad i res

```
faun:~>res show prgi05
RAPPORTERADE RESULTAT FÖR: 750301-0931 Mosavat, Vahid
ANVNAMN: vahid STUDIESTATUS: D-97 GRUPP: 1

Moment   Nr    Datum      Resultat   Rapp av

lab      1     050911     G          hjorth
```

Viktiga datum

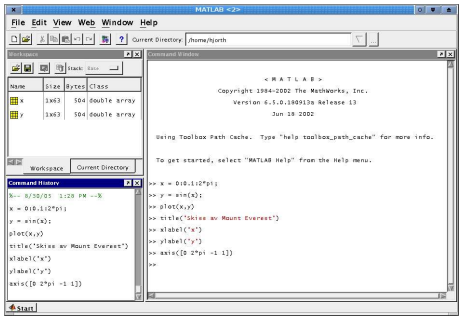
2005-09-16
Sista dag för redovisning av `Lab1` och `Lab2` med bonus

2005-09-20
Inlämning av hemtal 1 för bonus, handlar om matlab (obligatoriskt)

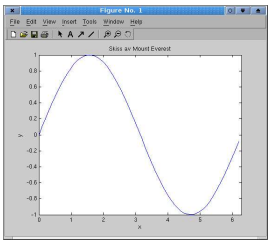
2005-09-23
Sista dag för redovisning av `Lab3` med bonus

Matlab

Ni kan ge matlab kommandon genom att skriva direkt i *command window*.



Koden ovan ritar upp grafen nedan



Skriv koden i m-filer

Det som skrivs i kommandofönstret sparas inte.

Om vi skriver kommandona in en textfil och sparar den med ändelsen `.m` så kan vi köra den i matlab.

Text i filen `minFil.m`:

```
%Kommentar börjar med procenttecken
disp('Hello world!')
```

Vi kör filen genom att skriva `minFil` i matlab

```
>> minFil
Hello world!
>>
```

Det är viktigt att vi är i rätt katalog, annars hittar matlab inte filen.

Variabler och tilldelning

Det är enkelt att skapa och tilldela värden till variabler i matlab

```
>> a = 1
a =
     1
>> b = 2
b =
     2
>> c = a + b
c =
     3
>>
```

Observera att en variabel måste ha fått ett värde innan vi kan använda den!

```
>> d = x + 1
??? Undefined function or variable 'x'.

>>
```

Vad är `ans` och `clear`

Om vi utför en beräkning utan att spara svaret

```
>> exp(log(1))
ans =
     1
>>
```

så ligger svaret tillfälligt sparat i variabeln `ans`.

```
>> ans
ans =
     1
>> x = ans
x =
     1
```

För att tömma variabeln `x` skriver vi

```
>> clear x
>> x
??? Undefined function or variable 'x'.

>>
```

För att rensa alla variabel skriver vi bara `clear`.

Skapa följder av tal

Antag att vi vill ha en vektor av alla tal mellan ett och sju, matlab låter oss enkelt skapa den:

```
>> x = 1:7
x =
     1     2     3     4     5     6     7
>>
```

Säg att vi vill ha de udda talen mellan ett och tio:

```
>> y = 1:2:10
y =
     1     3     5     7     9
>>
```

Det blir ganska mycket utskrifter. Om vi inte vill se dem skriver vi ett semikolon på slutet av raden

```
>> z = 10:-1:1;
>>
```

Hur skapar vi matriser?

Antag att vi vill skapa följande matris:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Det gör vi enkelt genom att skriva:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>>
```

Standardmatriser

Det finns inbyggda kommandon för standardmatriser

```
>> ones(2,4)
ans =
     1     1     1     1
     1     1     1     1
>> zeros(1,3)
ans =
     0     0     0
>> eye(2)
ans =
     1     0
     0     1
```

Den sistnämnda är enhetsmatrisen. Eftersom den alltid är kvadratisk tar den bara en parameter.

Det är enkelt att skapa en matris med tvåor:

```
>> T = 2*ones(1,3)
T =
     2     2     2
>>
```

Transponat

För att transponera en matris använder vi ’ (fnutt)

```
>> A = [1 2 3; 4 5 7]
A =
     1     2     3
     4     5     7
>> A'
ans =
     1     4
     2     5
     3     7
```

När man transponerar en matris (skrivs A^T) byter man radindex mot kolumnindex.

Elementet som stod på rad ett, kolumn två står efter transponering på rad två kolumn ett.

Kombinera matriser

Det går också att kombinera ihop flera matriser till en stor matris. Först skapar vi a, b och c.

```
>> a = 1:4
a =
     1     2     3     4
>> b = [0 7 3 9];
>> c = [1 2];
```

Med hjälp av matriserna ovan kan vi sedan bilda

```
>> B = [a; b; c -c]
B =
     1     2     3     4
     0     7     3     9
     1     2    -1    -2
>> D = [b-a; 2*a; 1 1 c]
D =
    -1     5     0     5
     2     4     6     8
     1     1     1     2
>> E = [c' zeros(2,1)]
E =
     1     0
     2     0
>>
```

Bilda delmatriser ur större matriser

```
>> A = [1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
```

Elementet på första raden tredje kolumnen:

```
>> x = A(1,3)
x =
     3
```

Hela första raden (alla kolumner):

```
>> y = A(1,:)
y =
     1     2     3
```

Rad ett och två ur tredje kolumnen:

```
>> A(1:2,3)
ans =
     3
     6
```

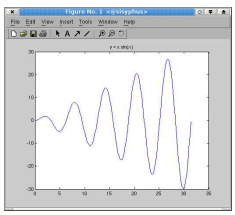
Använd vektorer i beräkningar

För att komponentvis multiplicera två vektorer använder vi `.*`-operatorn.

```
>> x = 1:5, y = 1:2:10
x =
     1     2     3     4     5
y =
     1     3     5     7     9
>> z = x .* y
z =
     1     6    15    28    45
```

Om vi vill plotta $y = x \sin x$ skapar vi först en x-vektor och beräknar sedan motsvarande y-värden för alla komponenter.

```
>> x = 0:0.1:pi;
>> y = x.*sin(x);
>> plot(x,y)
>> title('y = x sin(x)')
>>
```



Skapa funktioner

Skapa en fil med namn `minFunk.m`

```
% minFunk tar x och y som parametrar
function f = minFunk(x,y)

f = x.^2 + y;   % f innehåller svaret
```

Den kan sedan anropas från matlab med

```
>> minFunk(2,3)
ans =
     7
```

Den fungerar också för vektorer

```
>> x = 1:5, y = ones(1,5)
x =
     1     2     3     4     5
y =
     1     1     1     1     1
>> z = minFunk(x,y)
z =
     2     5    10    17    26
```

`If/else`-**satser**

En `if`-sats avslutas alltid med `end`. Vi kan hänga på en efterföljande `else` om vi behöver, se nedan.

I filen `ifelseexempel.m` står:

```
a = 1

if(a == 1)      % Två likamedtecken betyder jämförelse
  disp('a är ett')
else
  disp('a är inte ett')
end

if(a ~= 2)
  disp('a är inte två')
end
```

Vi kör den och får då följande utskrift (hur?):

```
>> ifelseexempel
a =
     1
a är ett
a är inte två
>>
```

`for`-**loopar**

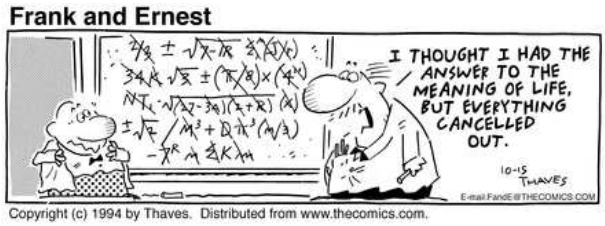
Vi använder `for`-loopar då vi vet hur många gånger en viss kod ska köras:

```
for i=1:3
  disp(['i är nu ' num2str(i)])
end
```

Här markerar `end` slutet på `for`-loopen.

Kör vi filen `forexempel.m` ovan får vi:

```
>> forexempel
i är nu 1
i är nu 2
i är nu 3
>>
```



`while`-**loopar**

Vi gör en `while`-loop om vi **inte** på förhand vet antalet gånger vi behöver göra något:

```
i = 0

while(i < 5)
  if(rand < 0.5) % slumptal mellan 0.0 och 1.0
    i = i + 1
  else
    i = i + 3    % Kom ihåg öka värdet på i inuti loopen,
  end            % annars fås oändlig loop... aj!
end
```

Vi kör `whileexempel.m` och ser att antalet varv i loopen varierar mellan körningarna:

```
>> whileexempel
i =
     0
i =
     1
i =
     4
i =
     5
```

```
>> whileexempel
i =
     0
i =
     3
i =
     6
```

`help`-**kommandot**

Tar för vana att använda matlabs inbyggda hjälp.

```
>> help break

 BREAK Terminate execution of WHILE or FOR loop.
    BREAK terminates the execution of FOR and WHILE loops.
    In nested loops, BREAK exits from the innermost loop only.

    BREAK is not defined outside of a FOR or WHILE loop.
    Use RETURN in this context instead.

    See also FOR, WHILE, RETURN, CONTINUE.
```

Längst ner står relaterade kommandon.

Lösning av ekvationssystem

Ett ekvationssystem

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= 10 \\ 4x_1 - 5x_2 + 6x_3 &= 11 \\ 7x_1 + 8x_2 - 9x_3 &= 12 \end{aligned}$$

skrivs med matrisnotation

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & -5 & 6 \\ 7 & 8 & -9 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 10 \\ 11 \\ 12 \end{pmatrix}$$

vilket är på formen $Ax = b$.

```
>> format compact
>> A = [1 2 3; 4 -5 6; 7 8 -9];
>> b = [10 11 12]';
>> x=A\b   %detta löser ekvationsystemet Ax=b
x =
    2.1356
    1.4746
    1.6384
>>
```

Det vill säga $x_1 = 2.14$, $x_2 = 1.47$ och $x_3 = 1.64$.

Kolla dig själv

Följande behöver du veta för att klara `Lab2` och första hemtalet:

- Skapa och tilldela variabler
- Skapa sekvenser av tal
- Skapa matriser direkt eller från delmatriser
- Ta ut delar av en matris
- Skriva och köra `m`-filer och funktioner
- Plotta grafer
- `if`/`else`-satser
- `for` och `while`-loopar
- Ekvationssytem