

---

# Matematikk 1000

## Øvingsoppgaver i numerikk – leksjon 2

### Løsningsforslag

---

#### Oppgave 1 – Vektorer

- a) Variablene i MATLAB kan være tall, vektorer eller matriser. Vi kan for eksempel gi vektoren  $\mathbf{x} = [1, 0, -3]$  på denne måten:

```
>> x=[1, 0, -3, 7]
```

```
x =
```

```
1    0   -3    7
```

```
>> x(2)
```

```
ans =
```

```
0
```

`x(2)` gir altså den andre komponenten i vektoren  $\mathbf{x}$ . Tilsvarende gir '`x(end)`' den siste komponenten i  $\mathbf{x}$ :

```
>> x(end)
```

```
ans =
```

```
7
```

Vi kan også referere til flere komponenter samtidig:

```
>> x(1:3)
```

```
ans =
```

```
1    0   -3
```

```
>> x(3:end)
```

```
ans =
```

```
-3    7
```

Om vi avslutter ei kommandolinje med semikolon, vil ikke noe bli skrevet til skjerm. Det betyr ikke at kommandoen ikke har blitt utført. Det har blitt gjort, vi bare slipper å se resultatet. Om vi for eksempel skriver

```
>> y=2+3;
```

så har faktisk variabelen  $y$  blitt 5 – selv om det ikke har blitt skrevet til skjerm.

b) Vi får:

```
>> x^2
```

```
Error using ^
```

```
Inputs must be a scalar and a square matrix.
```

```
To compute elementwise POWER, use POWER (.^) instead.
```

Vi ser at MATLAB “nekter” å regne ut  $x^2$ . Og det er ikke så rart; operasjonen er ikke veldefinert matematisk. Retnok har vi definert *skalarprodukter* for vektorer, men det blir noe annet. MATLAB tolker multiplikasjonstegnet ‘\*’ det som en matrise-multiplikasjon, og gitt at  $x$  er en vektor, er ikke multiplikasjonen vi ba MATLAB utføre, en meningsful operasjon. Om vi tar med punktum foran potens-sybolet, derimot, blir det tolket som at man skal kvadrere elementvis:

```
>> x.^2
```

```
ans =
```

```
1    0    9
```

Vi får altså vektoren der hvert element blir kvadrert,  $[x(1)^2 \ x(2)^2 \ x(3)^2]$ , til svar.

c)  $1:5$  gir som svar 1 2 3 4 5. Tilsvarende kan heltallene fra og med -3 til og med 3 skrives slik: -3:3. En vektor bestående av tallene 0, 0.25, 0.5, ..., 1.75, 2 kan uttrykkes slik: 0:.25:2 (‘.25’ betyr det samme som 0.25).

d) Vektoren  $y$  kan tilordnes slik:

```
>> y=2:2:10
```

```
y =
```

```
2    4    6    8   10
```

Kommandoen `length` brukes for å finne ut hvor “lang” en vektor er – altså hvor mange elementer den har:

```
>> length(y)
```

```
ans =
```

```
5
```

`y` har altså 5 elementer.

e) Tilordning:

```
>> Vektor=[9 pi -2 100 8]
```

```
Vektor =
```

```
9.0000    3.1416   -2.0000  100.0000    8.0000
```

Å sortere heter *sort* på engelsk. Om vi søker på dette, finner vi fort at der er en rutine som heter *sort*:

```
>> lookfor sort
```

```
issorted          - TRUE for sorted vector and matrices.  
sort              - Sort in ascending or descending order.  
sortrows          - Sort rows in ascending order.  
cplxpair          - Sort numbers into complex conjugate pairs.  
dee_find_system   - Get a sorted list of DEE related systems.  
...
```

Vi kan også lese om hvordan den fungerer:

```
>> help sort
```

```
sort - Sort array elements
```

```
This MATLAB function sorts the elements of A in ascending order  
along the first array dimension whose size does not equal 1.
```

```
B = sort(A)  
B = sort(A,dim)  
B = sort(___,mode)  
[B,I] = sort(___)
```

```
Reference page for sort
```

```
See also issorted, max, mean, median, min, sortrows, unique
```

Vi finner nok fort ut at den gjør det vi ønsker:

```
>> sort(Vektor)
```

```
ans =
```

```
-2.0000    3.1416    8.0000    9.0000   100.0000
```

Jo så menn, tallene kom i stigende rekkefølge.

Denne skrivemåten, navnet på funksjonen etterfulgt av det som vi skal utføre funksjonen på i parantes, er ganske gjennomgående i MATLAB.

Vi kan gå fram på same måte for å finne en summeringsfunksjon. Vi finner fort ut at funksjonen 'sum' gjør jobben:

```
>> sum(Vektor)
```

```
ans =
```

```
118.14
```

f) Vi ser på MATLABs dokumentasjon for `linspace`-funksjonen:

```
>> help linspace
```

```
linspace Linearly spaced vector.
```

```
linspace(X1, X2) generates a row vector of 100 linearly  
equally spaced points between X1 and X2.
```

```
linspace(X1, X2, N) generates N points between X1 and X2.  
For N = 1, linspace returns X2.
```

```
Class support for inputs X1,X2:  
float: double, single
```

```
See also logspace, colon.
```

```
Reference page in Help browser  
doc linspace
```

Vi kan altså bruke denne til å lage en vektor som deler intervallet fra en bestemt startverdi til en bestemt sluttverdi opp i like store deler. Om vi ikke spesifiserer hvor mange deler, blir dette satt til 100 (100 er såkalt *default* for funksjonen). Men vi kan også godt bestemme dette antallet. For eksempel gir dette en vektor med 5 elementer som går fra -1 til 1:

```
>> linspace(-1,1,5)
```

```
ans =
```

```
-1.0000   -0.5000         0    0.5000    1.0000
```

Det er ikke meningen at vi skal pugge navnet på en masse funksjoner i MATLAB. Men det er en klar fordel å lære seg å lete etter de funksjonene vi kunne ha bruk for eller nytte av. Men generelt er det ganske få funksjoner vi vil ha direkte bruk for i dette kurset.

## Oppgave 2 – Matriser

a) Det kan gjøres slik:

```
>> v=[2  
-4  
pi];
```

eller slik:

```
>> v=[2; -4; pi]
```

```
v =
```

```
2.0000  
-4.0000  
3.1416
```

b) Dette kan gjøres slik:

```
>> M=[2 0 7 4  
-2 2 9 8  
1.4 -7 14 0];
```

eller slik:

```
>> M=[2 0 7 4; -2 2 9 8; 1.4 -7 14 0];
```

c) Når vi skriver 'M(2,3)' i kommando-vinduet, får vi ut elementet i rekke 2, søyle 3:

```
>> M(2,3)
```

```
ans =
```

```
9
```

Når vi skriver 'M(3,:) ', viser dette til rekke nummer 3, mens kolon-symbolet for den andre indeksen tilsvarer at vi vil ha alle elementer. Vi får altså opp alle elementer i rekke 3 i  $M$ :

```
>> M(3,:)

```

```
ans =

```

```
    1.4000   -7.0000   14.0000         0

```

'M(3,2:4)' gir elementene fra og med nummer 2 til og med nummer 4 i rekke 3:

```
>> M(3,2:4)

```

```
ans =

```

```
    -7    14     0

```

Søyle 2 kan hentes ut slik:

```
>> M(:,2)

```

```
ans =

```

```
     0
     2
    -7

```

og den nevnte undermatrisa kan hentes ut slik:

```
>> M(2:3,2:4)

```

```
ans =

```

```
     2     9     8
    -7    14     0

```

## Oppgave 3 – Komplekse tall

Som vi har sett, spytter MATLAB ut mange mellomrom når den skal skrive svar til skjerm. Dette kan vi unngå om vi skriver '**» format compact**'.

a) i) I kommandovinduet:

```
>> format compact
>> exp(i*pi)
ans =
    -1.0000 + 0.0000i

```

Altså:  $e^{i\pi} = -1$ .

```

ii) >> sqrt(7)*exp(i*pi/4)
ans =
    1.8708 + 1.8708i
iii) >> 0.365*exp(i*1.21)
ans =
    0.1289 + 0.3415i

```

- b) Her kan vi ha nytte av å tilordne det komplekse tallet til en variabel: I MATLAB vil **abs**-funksjonen gi absoluttverdien til et komplekst tall, mens fasen kan finnes med en funksjon som heter **angle**. Bruk MATLAB til å finne polarforma til følgende tall:

```

i) >> z=1-i;
>> abs(z)
ans =
    1.4142
>> angle(z)
ans =
   -0.7854

```

I følge MATLAB har vi altså at  
 $1 - i = 1.4242 e^{-0.7854 i}$  (eksakt:  $\sqrt{2} e^{-i\pi/4}$ ).

- ii) Her er det nok ikke så vanskelig å se at absoluttverdien er 1 og fasen er  $\pi/2$  slik at  $i = e^{i\pi/2}$ . Men vi kan uansett gjøre det på samme måte som over:

```

>> z=i;
>> abs(z)
ans =
    1
>> angle(z)
ans =
    1.5708
>> pi/2
ans =
    1.5708

```

Vi ser at det vi skrev over stemmer (i alle fall med fire desimaler).

```

iii) >> z=7+2*i;
>> abs(z)
ans =
    7.2801
>> angle(z)
ans =
    0.2783
7 + 2i = 7.2801 e^{0.2783 i}.

```

- c) i) >> z=(1-i)\*(1+i)  
z =

2

Tallet viser seg å bli reelt;  $2 = 2 + 0 \cdot i = 2e^{i \cdot 0}$ .

ii) >> z=(1-i)\*(1+i)

z =

2

>> z=(3-7\*i)\*(2+i)

z =

13.0000 -11.0000i

>> abs(z)

ans =

17.0294

>> angle(z)

ans =

-0.7023

Altså:  $(3 - 7i)(2 + i) = 13 - 11i = 17.0294e^{-0.7023i}$ .

iii) >> z=(1-i)/(2-3\*i)

z =

0.3846 + 0.0769i

>> abs(z)

ans =

0.3922

>> angle(z)

ans =

0.1974

Vi får at  $\frac{1-i}{2-3i} = 0.3846 + 0.0769i = 0.3922e^{0.1974i}$ .

iv) >> z=sqrt(8)\*exp(i\*pi/3)/(sqrt(2)\*exp(-i\*pi/6))

z =

0.0000 + 2.0000i

>> abs(z)

ans =

2

>> angle(z)

ans =

1.5708

Svaret blir  $\frac{\sqrt{8}e^{i\pi/3}}{\sqrt{2}e^{-i\pi/6}} = 2i = 2e^{1.5708i}$ .

Til slutt sjekker vi at vi får det samme om vi regner ut svarene selv:

i)

$$(1-i)(1+i) = 1 \cdot 1 + 1 \cdot i - i \cdot 1 - i \cdot i = 1 - i^2 = 1 - (-1) = \underline{2}$$

ii)

$$(3-7i)(2+i) = 3 \cdot 2 + 3 \cdot i - 7i \cdot 2 - 7i \cdot i = 6 + (3-14)i - 7i^2 = 6 - 11i + 7 = \underline{13-11i}$$

iii)

$$\frac{1-i}{2-3i} = \frac{(1-i)(2+3i)}{(2-3i)(2+3i)} = \frac{2+3i-2i-3i^2}{2^2-3^2i^2} =$$
$$\frac{2+3+i}{4+9} = \frac{5+i}{13} = \frac{5}{13} + \frac{1}{13}i \approx \underline{0.3846 + 0.0769i}$$

iv)

$$\frac{\sqrt{8}e^{i\pi/3}}{\sqrt{2}e^{-i\pi/6}} = \frac{\sqrt{8}}{\sqrt{2}}e^{i\pi/3-(-i\pi/6)} = 2e^{i\pi/2} = 2(\cos \frac{\pi}{2} + i \sin \frac{\pi}{2}) =$$
$$2(0 + i \cdot 1) = \underline{2i}$$

Alle svarene stemmer med det som MATLAB gav. (Det hadde jo vært dumt hvis ikke...)

## Ekstraoppgave – Rekkeoperasjoner

Den første rekkeoperasjonen kan utføres slik:

```
>> A1=A;  
>> A1(3,:)=A(2,:);  
>> A1(2,:)=A(3,:);
```

Her blir A1 den matrisa vi får når vi bytter om rekke 2 og 3 i matrisa A. Vi illustrerer med et eksempel:

```
>> A=[1 2 3 4; -2 -1 0 1; 3 0 2 0]  
A =  
     1     2     3     4  
    -2    -1     0     1  
     3     0     2     0  
>> A1=A;  
>> A1(3,:)=A(2,:);  
>> A1(2,:)=A(3,:);  
>> A1  
A1 =  
     1     2     3     4  
     3     0     2     0  
    -2    -1     0     1
```

Vi viser et eksempel på den andre rekkeoperasjonen også:

```
>> A2=A;
```

```
>> A2(1,:)=2*A(1,:);
>> A2
A2 =
     2     4     6     8
    -2    -1     0     1
     3     0     2     0
```

Her er A2 den matrisa vi frå når vi ganger den første rekka i A med 2.

Den siste rekkeoperasjonen kan implementeres slik:

```
>> A3=A
A3 =
     1     2     3     4
    -2    -1     0     1
     3     0     2     0
>> A3(3,:)=A(3,:)+(-3)*A(1,:);
>> A3
A3 =
     1     2     3     4
    -2    -1     0     1
     0    -6    -7   -12
```

Her har vi lagt -3 ganger rekke 1 til rekke 3.