Matematikk 1000

Øvingsoppgaver i numerikk – leksjon 1
Løsningsforslag

Oppgave 2 – Litt aritmetikk

a) Her har vi skrevet ut det som kommer opp i kommandovinduet når vi utfører operasjonene.

```
>> 2+2

ans =

     4

>> 3-2

ans =

     1

>> 3*2

ans =

     6

>> 6^2

ans =

    36

>> 6/2

ans =

     3
```

b) Når vi utfører mer kompliserte utregninger, gjelder det å holde orden på parantesene:

```
>> 3^2.12*(2+4*(-3.1))/(2^2-1)

ans =

    -35.597
```

c) I kommandovinduet får vi:

```
>> sin(0.7)

ans =

    0.64422

>> log(5)

ans =

    1.6094

>> sqrt(9)

ans =

    3

>> exp(1)

ans =

    2.7183

>> atan(2)

ans =

    1.1071
```

'`sin`' er sinus-funksjonen, '`log`' er den naturlige logaritmen, $\ln x$, '`sqrt`' er kvadratrotfuksjonen, '`exp`' er eksponentialfunksjonen, e^x, og 'atan', er den inverse tangensfunksjonen, $\arctan x$. Legg merke til at argumentet skal stå i en parantes, og at desimaltall blir angitt med punktum – ikke komma.

d) Vi forsøker å regne ut disse uttrykkene i MATLAB:

```
>> 1/0

ans =

   Inf
```

Her blir vi advart mot å gjøre en ting vi vet vi ikke har lov til å gjøre i matematikk: Å dele på 0. Vi ser også at MATLAB velger å gi 'uendelig' som svar.

```
>> 0^0

ans =

     1
```

Her har også MATLAB gjort en 'tolkning' som kan diskuteres. I matematikk kan man både argumentere for at 0^0 er 1 siden x^0 er èn for alle andre verdier av x. Men man kan også argumentere for at det er 0, siden $0^x = 0$ for alle andre verdier av x enn 0. Vi ser at MATLAB har valgt den første tolkningen.

```
>> 5/Inf

ans =

     0
```

Dette virker vel rimelig, gjør det ikke? Det gjør i alle fall det om vi tolker `5/Inf` som grenseverdien $\lim_{x\to\infty} 5/x$.

```
>> 0*Inf

ans =

NaN

>> Inf/Inf

ans =

NaN
```

Det virker også rimelig at hverken "$0\cdot\infty$" eller "∞/∞" blir gitt noen verdi; disse uttrykkene gir ikke mening matematisk.

```
>> 10^999

ans =

Inf
```

Her blir en av de fundamentale forskjellene mellom numerisk og analytisk matematikk tydelig; selv om 10^{999} er et veldig stort tall, er det slett ikke uendelig, slik MATLAB påstår. Men der er grenser for hvor store tall MATLAB er i stand til å håndtere; dette tallet var tydeligvis for stort. På tilsvarende måte er MATLAB også ute av stand til å forholde seg til uendelig små tall.

```
>> 1.2e4

ans =

12000
```

Dette er måten vi skriver tall på normalform i MATLAB; '`1.2e4`' betyr altså $1.2 \cdot 10^4$.

```
>> NaN+1

ans =

NaN
```

At et udefinert tall pluss én forblir et udefinert tall er vel i grunn ganske opplagt.

e)
```
>> pi

ans =

3.1416
>> i

ans =

0 + 1i

>> eps

ans =

2.2204e-016
```

'`pi`' gir tallet π. Det bør her legges til at MATLAB opererer med en langt mer nøyaktige verdi av π enn det antall desimaler tilsier her. Bare et visst antall desimaler blir skrevet til skjermen. Vi kan velge å endre dette formatet. Hvis du vil ha flere desimaler, for eksempel, kan du skrive '`format long`':

```
>> format long
>> pi

ans =

   3.141592653589793
```

Man kan synes det er litt pussig at ikke Eulers tall, e, er tilordna til `e` fra før. Uansett er det ikke verre enn at man kan skrive '`e=exp(1)`' for å ordne dette.

Vi kan gå tilbake til det kortere formatet igjen ved å skrive '`» format short`'.

Oppgave 3 – Tilordning

a) Slik vil det se ut når vi sier at variabelen x skal ha verdien 7:

```
>> x=7

x =

     7
```

Merk at vi nå i tillegg til selve tilordninga også har sagt at *der skal være* ein variabel som heter `x`. Vi sier at vi har *initiert* variabelen.

Resultatene av reknestykkene blir:

```
>> x=7

x =

     7

>> x*2

ans =

    14
```

```
>> x^2

ans =

    49

>> y=2

y =

     2

>> x/y

ans =

    3.5000
```

Dersom det skulle bli litt mange variable å holde styr på, kan man fjærne variable med kommandoen `clear`; for eksempel kan man slette `y` ved å skrive '`clear y`'. Om man vil "renske tavla" helt, gjør man det ved å skrive bare '`clear`'.

b) Symbolet '`=`' betyr ikke helt det samme i MATLAB-sammenheng som det gjør i matematisk notasjon; her er likhetstegnet brukt som tilordning; altså for å gi en variabel en verdi. Derfor vil følgende gi mening i MATLAB:

```
>> x=2

x =

     2

>> x=x+1

x =

     3
```

Vi ser at den siste kommandoen la én til variabelen x og tilordna svaret tilbake til x; den *gamle* x var 2 og den *nye* x ble 3. Det er altså ikke samme `x` på hver side av likhetstegnet. Matematisk er den siste linja meningsløs siden den gir at

$$x = x + 1 \Leftrightarrow x - x = 1 \Leftrightarrow 0 = 1 \ .$$

Om vi ønsker å uttrykke likhet, ikke tilordning, i MATLAB, skriver vi '`==`' – altså et dobbelt likhetstegn. Dette skal vi komme tilbake til senere.

c) Vi tilordner i kommandovinduet:

```
>> pi

ans =

    3.1416

>> pi=10

pi =

    10
```

Vi ser at `pi`, som var π, nå har fått verdien 10. Tilsvarende skjer også med `i`. Vi ser at våre tilordninger overstyrer de som MATLAB måtte ha fra før. Dette gjelder også om vi gir variable navn på funksjoner som MATLAB har innebygd. For å unngår forvirring og feil, er det en klar fordel om vi kan unngå at dette skjer.