START MED MATLAB

Disse sidene er hovedsakelig ment for dem som ikke har brukt Matlab eller som trenger en oppfriskning. Start fra toppen og gå systematisk nedover. I tillegg brukes Matlablefsa. Noe av hensikten er også å bli vant til å bruke hjelpesystemet. Det finnes over tusen kommandoer i Matlab. Disse er beskrevet i detalj i manualene som du får tak i ved å gå til `help` og `helpdesk` i menylinja i Matlab. Legg spesielt merke til *Getting started* som er startmanual

Når du starter Matlab, kommer du inn i kommandovinduet. Dersom du har versjon 6.1, ser du dette :

```
  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, type tour or visit www.mathworks.com.

»
```

Tegnet » er en kommandoprompt og du kan begynne å gi kommandoer.
Med å utføre en kommando, menes å skrive noe etter » og trykke på enter-knappen.

Vi starter med å bruke Matlab som en kalkulator.

Utfør `1.2 + 2.3` . Skjermen viser da følgende :

```
» 1.2 + 2.3

ans =

    3.5000

»
```

Vi har fått svaret med 4 desimaler. Svaret er lagret i en variabel `ans` som er innebygget i Matlab. Når du bruker Matlab, blir det avsatt et område i RAM som kalles arbeidsområdet (workspace). For å finne ut hva som finnes i arbeidsområdet, kan du utføre kommandoen `whos` som gir dette resultatet :

```
  Name      Size            Bytes  Class

  ans       1x1                 8  double array

Grand total is 1 elements using 8 bytes
```

Vi ser at det er avsatt plass til en variabel med navn `ans` med plassbehov 8 bytes. Merk at variable i Matlab oppfattes som matriser slik at en skalar er en 1×1-matrise.

Innholdet i arbeidsområdet fjernes med kommandoen `clear`. (Ikke gjør dette nå). En enkelt variabel x fjernes ved `clear x`.

Eks : Utfør `a = 1`. Utfør deretter kommandoene `whos, clear a` og `whos` tilslutt for å kontrollere innholdet i arbeidsområdet. (Vi skal da ha bare `ans` tilbake). Se side 3 i Matlablefsa om tillatte navn på variable.

Anta at vi ønsker flere desimaler. Utfør `help format`. Du får da en liste med alternativer. Utfør `format long`. Ved å utføre `ans` på nytt, får vi 14 desimaler. Matlab regner med ca. 15 siffers nøyaktighet uavhengig av hvilket format du velger. Dersom du synes at utskriften tar mye plass på skjermen, kan du velge `format compact` (forsøk). (`format loose` gir tilbake de ekstra linjene.)

Vi går tilbake til 4 desimaler ved å uføre `format` som er en forkortelse for `format short`
Utfør 1/2 +1/3 +1/4 +1/5 som gir 1.2833. Velg `format rat` og skriv ut `ans` på nytt. Du skal da få 77/60. (Vi bruker 4 desimaler videre)

En hel del av kommandoene ovenfor finnes ved å utføre `help general`. Bruk av noen av disse kommandoene er behandlet side 4 i Matlablefsa.

Når du opererer i kommandovinduet, har du adgang til en primitiv editor. (`help cedit` gir en liste over alle mulighetene. Merk at tegnet ^ står for Ctrl-tasten). De fleste kommandoene er knyttet til taster på tastaturet og er derfor innlysende. Dersom du vil ha tilbake en kommando som du har brukt tidligere, kan du trykke på ↑- tasten. (Trykk flere ganger dersom du ikke finner den du ønsker med en gang). Dersom dette ikke virker, må du først utføre kommandoen `cedit on`. Dessuten kan du bruke de vanlige Windows-kommandoene Ctrl-C (kopier), Ctrl-V (lim) og Ctrl-X (slett). Du kan blanke skjermen ved å utføre kommandoen `clc`. En annen nyttig er `more` i forbindelse med lange utskrifter (`help more`).

Vi kan knytte navn til de enkelte uttrykkene.
Eks. :
```
a = sin(1)
b = cos(1)
c = a + b
```

Vi kan også skrive dette på en linje ved :
```
a = sin(1) ; b = cos(1); c = a + b;
```

Legg merke til at nå får vi ingen utskrift. Semikolon brukes til å hindre utskrift. Dette er viktig for å hindre utskrift av mellomregning vi ikke er interessert i. Dessuten brukes semikolon til å skille uttrykk på samme linje.

Strenger
Strenger er en samling med tegn mellom to apostrofer.
Eks. : Utfør `str = 'Dette er en streng'`
Srengvariable brukes helst ved programmering og ofte ved innlesning, utskrift og plotting. Du kan transformere strengen til et tall ved `x = double(str)`
Hvert tegn får nå den verdien den har i den såkalte ASCII-tabellen.
Går tilbake til en streng ved `str = char(x).` Vi kan f.eks sette sammen to strenger `s1` og `s2` ved kommandoen `strcat(str1,str2).`
Forsøk med `str1 = 'Dette er '` og `str2 = 'en streng'`
Utfør `a = '3.141592'.` Dette er en streng , ikke en tilnærmelse til `pi`.
`a` kan konverteres til et tall ved `b = eval(a)`. Kontroller dette ved å utføre `whos.`
Det finnes mange flere strengfunksjoner enn dem vi har brukt ovenfor.
Prøv `help strfun.`

Komplekse tall
`i` og `j` brukes som den komplekse enheten $\sqrt{-1}$.
Utfør `z1 = 2 + i*3` og `z2 = 3 - i*4.`
Utfør deretter `z1 + z2 , z1*z2, z1/z2.`
De komplekse funksjonene finnes ved `help elfun`

Bruk av innebygde matematiske funksjoner.

Du finner en liste over de vanlige matematiske funksjonene ved kommandoen `help elfun`. (Start med `help help` om nødvendig)
Prøv endel av funksjonene , også de mer ukjente. Bruk `help` for å finne den nøyaktige syntaksen. Forsøk eks. `fix(2.1), floor(-2.1), ceil(2.1) , round(-2.1)` og se om resultatet stemmer med beskrivelsen.

Utfør f.eks. `exp(1),sin(pi/2), tan(pi/2), atan(1), atan2(1,-1)`

Legg merke til at `pi` er innebygd i Matlab, se side 3 i Matlablefsa.

Vær klar over at når du f.eks. skriver `help sin`, vil du få referansen skrevet med store bokstaver; i dette tilfellether SIN(X). Dette er gjort for at svaret skal vise tydelig på skjermen, *ikke* at du skal bruke store bokstaver. Matlab skiller mellom store og små bokstaver. Vanligvis brukes små. (Utfør f.eks. kommandoen HELP)

I mange tilfeller har vi bruk for raskt å plotte funksjoner. Dette gjøres greit med kommandoen fplot. (Mer detaljer med `help fplot`).
Utfør f.eks. `fplot('sin(x)', [0 2*pi]).` Merk apostrofene rundt funksjonsnavnet. Det blir nå åpnet et eget figur-vindu. Null-punkter er lettere å plukke ut ved å legge på et nett med kommandoen `grid on`. (Eksempel på mer avansert plotting finnes side 32-34 i Matlablefsa)

La oss se på mer "eksotiske" funksjoner. Disse finnes med `help specfun`.
Vi ønsker å plotte funksjonen $J_0(x)$ for $0 \le x \le 10$ der $J_0(x)$ er en Besselfunksjon av 1. orden og nullte slag. Fra lista finner vi `besselj` som ifølge beskrivelsen er en Besselfunksjon av første slag. (Dersom du på forhånd visste at dette var en Besselfunksjon, kan du skrive `help bessel`). For å få flere detaljer, skriver vi `help besselj` Ifølge beskrivelsen får vi en Besselfunksjon av 1. slag og orden ν med argument x ved å bruke `besselj(nu,x).`
Vi utfører derfor følgende: `fplot('besselj(0,x)', [0 10]).`
Fra figuren finner vi nullpunkter rundt x = 2.4 , 5.5 og 8.6.

Vi vil nå bestemme disse nullpuntene nøyaktig ved å bruke Matlabfunksjonen **fzero**. (`help funfun`). Det første nullpunktet finner vi ved å utføre `fzero('besselj(0,x)',2.4)` og resultatet er 2.4048. (Dersom vi bruker `format long` får vi 2.40482555769577)
Finn også de to andre nullpunktene. (En mer detaljert beskrivelse av **fzero** er gitt i lefsa side 24).

Når vi først er i gang, kan vi også beregne noen bestemte integral.
La oss integrere sin(x) fra 0 til 90º der svaret (selvfølgelig) er 1
Utfører `help funfun` og finner Matlabfunksjonen **quad.**
Utfra beskrivelsen, utfører vi `quad('sin',0,pi/2)` som gir `1.0000`
Legg merke til at vi utførte `help funfun` for å finne et program som utførte integrasjonen. Hva skulle vi ha gjort dersom vi ikke visste at **quad** kunne finnes her? Det mest naturlige er å forsøke `help integral`. Dersom vi gjør det, får vi som svar `integral.m not found.` Vi har heldigvis kommandoen `lookfor`. (Utfør `help lookfor` for flere detaljer). Dersom vi utfører `lookfor integral`, får vi bl. annet :

```
QUAD   Numerically evaluate integral, low order method.
QUAD8  Numerically evaluate integral, higher order method.
```

Ved å bruke `help quad`, får vi flere opplysninger.

Matriser og vektorer

Gå gjennom side 5- 12 i Matlablefsa. Om du ikke går gjennom alt , så forsikre deg ihvertfall om at du kan skrive inn matriser og vektorer og kan kan utføre de vanligste matriseoperasjonene.

Bruk av editor ved programmering.

Velg *file* i menyen, deretter *new* og velg *M-file*. Du er nå klar til å skrive et program med den innebygde editoren. Skriv inn programmet som er gitt nedenfor :

```
%=== Program poly2 ===
% Finner de reelle røttene av ligningen Ax^2+Bx + C
a = input('Les koeffisient A: ');
b = input('Les koeffisient B: ');
c = input('Les koeffisient C: ');
d = b^2 - 4*a*c;
if d > 0  % To reelle røtter
   x1 = (-b + sqrt(d))/(2*a);
   x2 = (-b - sqrt(d))/(2*a);
   fprintf('x1 = %f\n',x1)
   fprintf('x2 = %f\n',x2)
elseif d == 0  % En reell rot
   x1 = -b/(2*a);
   fprintf('x1 = x2 = %f\n',x1)
else  % Imaginære røtter
   fprintf(' === Ingen reelle røtter ===\n')
end
```

Når programmet er skrevet, må det lagres. Velg *Save As* på fil-menyen og lagre det under navnet poly2. Matlab legger automatisk til endelsen m.
Work-mappa er en grei plass som temporær lagring av et program dersom du er på datasalen. Dersom du ønsker å ta vare på programmet, må det overføres til mappa di på filserver. Etter at programmet er lagret, kan det eksekveres i kommandovinduet ved å utføre `poly2.` Du kan teste det på følgende eksempler :

$$x^2 + 5x + 6 = 0$$

$$x^2 + 4x + 4 = 0$$

$$x^2 + 2x + 5 = 0$$

Du kan nå gå videre med programmeringsdelen i Matlablefsa.

Litt om feilfinning i program

Svært ofte vil det være feil i et program første gangen det brukes. Feilskrift er typisk. Dette fører til syntaksfeil som blir oppdaget av Matlab. Feil bruk av språket generelt fører også til syntaksfeil. I begynnelsen kan det ofte være vanskelig å finne ut hva Matlab mener når den påpeker feil, men det kommer med litt øvelse. Når syntaksen er i orden, kommer de logiske feilene : Programmet gjør ikke det du ønsker. I praksis betyr dette at du må be programmet skrive ut mellomregning som du normalt ikke er interessert i.
I programmet ovenfor kan du f.eks. skrive ut størrelsen `d` ved å fjerne semikolonet etter den linja der `d` er definert. Husk at du kan finne slutt-verdiene

av de størrelsene som er beregnet i programmet ved å utføre de aktuelle variable i kommandovinduet. I noen tilfeller ønsker du å teste ut en del av programmet fordi du vet at resten ikke funker. En vanlig teknikk er å sette kommentartegn (%) foran de linjene du ikke vil ha utført. Du kan stoppe eksekveringen av et program ved å legge inn kommandoen `break.` Men da `break` bare virker som stoppkommando utenfor løkker, er det ofte lurere å sette stopp-punkt (breakpoints). Dette gjøres i editoren ved å gå til den linja i programmet der du ønsker å stanse, klikke med høyre musetast og deretter velge *Set/Clear breakpoint*. Eventuelt gå til den aktuelle linja og trykke F12. Stopp-punktet markeres ved et rødt punkt helt til venstre på linja. Gå til kommandovinduet og eksekver programmet. Editorvinduet åpnes og en gul pil viser seg der stopp-punktet er satt. Ved å gå til *Debug* på menylinja, kan du velge enrekke alternativer. f.eks. eksekvere linje for linje ved å trykke F10.