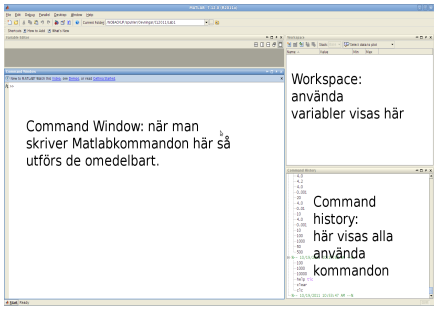
Snabb genomgång av viktiga Matlab kommandon[1]

I denna sammanställning behandlas några viktiga Matlab kommandon som vi kommer att använda för att implementera de numeriska problemen i våra uppgifter. Matlab lär man sig bäst genom att testa själv och därför uppmuntrar vi att lösa några MÖ-uppgifter som övning inför nästa labbtillfället.

- **Kommandofönster**
  Matlab-fönster består av olika delfönster bl a Command Window (kommandofönster), Workspace och Command History.



  Texten som följer >> är användarens inmatning och för att utföra ett kommando så trycker man på ENTER. Om raden avslutas med ; skrivs inte resultatet till terminalen. Om man inte tilldelar ett namn till resultatet så hamnar resultatet i variabeln ans.
  >>3+2-1
  ans = 4

- **Hjälp**
  För att få hjälp för ett specifikt kommando som t.ex. *plot* skriv
  >> help plot

- **Rensa**
  För att rensa alla variabler i minnet skriv
  >> clear
  För att rensa kommandofönstret skriv
  >> clc

- **Kommentar**
  För att kommentera bort ett kommando skriv % i början av raden.
  % y= sin(x);

- **Vektorer**
  I Matlab kan man skapa vektorer på olika sätt. Med följande kommandon skapas samma vektor.
  >> x = [4 6 8 10 12 14]; *% ange elementen inom hårdklamrar*
  >> x = 4:2:14; *% startvärde:steglängd:slutvärde*
  >> x = 4 + 2*[0:5]; *% det inbyggda värdet för steglängden är 1*
  >> x = linspace(4,14,6); *% linspace(start, slut, antal), elementen är ekvidistant fördelade mellan star och slutvärde*

  Vi kan konkatenera två vektorer (akta på hur man använder kommat):
  >> x = 4:2:8
  >> y = [x, x];
  >> y =
  4 6 8 4 6 8

1 *Exempel från http://www.eecs.umich.edu/~fessler/course/556/r/matlab.pdf%27 och Matlab 7 i korthet*

Man kan transponera en radvektor till en kolumnvektor och vice versa med operatorn ':

```
>> x = [4:2:8]'
>> y = [x; x];
>> y =
        4
        6
        8
        4
        6
        8
```

- **Matriser**

Här listas några exempel för att skapa en matris

```
>> A = [1 2 ; 3 4]          % elementen anges inom hårdklamrar
A =
        1    2
        3    4
```

```
>> b1 = 1:3;
>> b2 =  4:6;
>> B = [b1; b2]             % en matris kan byggas upp av flera vektorer
B =
        1    2    3
        4    5    6
```

Testa också följande kommandon:

```
>> B = eye(3)
>> C = ones(2,3)
>> D = zeros(3,2)
>> E = rand(1,5)
```

- **Punktnotation**

Med punktnotation menar man att sätta en punkt framför den aktuella operatorn som .*, ./ , .^ etc. När man använder punktnotationen innebär det att operationen utförs elementvis.

Exempel: (1 2 3) .* (4 5 6) => (1* 4 2*5 3*6) = > (4 10 18)

Testa:

```
>> x=1:5
>> y=x.*x;          %vad händer om man skriver y = x*x?
 >>y=x./(x+x);
```

- **Plotta en funktion**

För att rita en kurva måste man definiera x- och y-värdena av funktionen och använda kommandot plot.

```
>> x = 0:pi/100:pi;
>> y = sin(x);
>> plot(x,y);
```

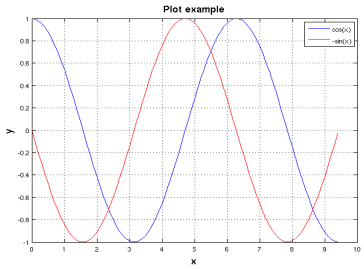
För att plotta flera grafer i samma fönster skriver man hold on så att nya plottar läggas till utan att de gamla tas bort.

```
>> x = 0:0.1:3*pi;
>> y = cos(x);
>> plot(x, y);
>> hold on;                 % Mellan hold on och hold off läggas till nya plottar till samma plott
>> y = -sin(x);
>> plot(x,y, 'r');          % kurvan för -sin(x) är rödmarkerad ('r')
>> grid on;                 % sätter ett rutnät i grafen
>> title('Plot example','fontsize',14,'fontweight','b');  %sätter titeln med skriftstorlek 14 och skriver den fet
>> legend('cos(x)', '-sin(x)'); % skriver beskrivningen av graferna
```

```
>> xlabel('x','fontsize',14,'fontweight','b');          % märka x-axeln
>> ylabel('y','fontsize',14,'fontweight','b');          % märka y-axeln
>> hold off;
```

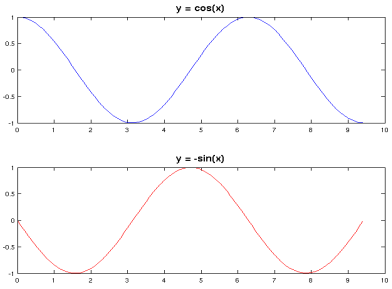
Alternativet som ger samma plot:

```
>> x = 0:0.1:3*pi;
>> y1 = cos(x);
>> y2 = - sin(x);
>> plot(x,y1, x, y2);
>> grid on; % sätter ett rutnät i grafen
>> title('Plot example','fontsize',14,'fontweight','b');  %sätter titeln med skriftstorlek 14 och skriver den fet
>> legend('cos(x)', '-sin(x)');                           %skriver beskrivningen av graferna
>> xlabel('x','fontsize',14,'fontweight','b');            %märka x-axeln
>> ylabel('y','fontsize',14,'fontweight','b');            %märka y-axeln
```



För att göra flera separata plottar kan man använda funktionen *subplot(rader, kolumner, aktuell). Subplot* delar upp ett grafikfönster i en matris och med *rader, kolumner* och *aktuell* anger man vilket grafikfönster som är aktuellt för att visa plotten.

```
>> x = 0:0.1:3*pi;
>> y = cos(x);
>> subplot(2, 1,1);
>> plot(x,y);
>> title('y = cos(x)','fontsize',14,'fontweight','b');
>> y = - sin(x);
>> subplot(2,1,2);
>> plot(x,y, 'r');
>> title('y = -sin(x)','fontsize',14,'fontweight','b');
```



- **Skapa egna funktioner**

För att underlätta arbetet i Matlab kan man skriva egna funktioner . Oftast skapar man funktionsfiler.

Funktionsfiler:

En funktionsfil är en Matlabfil (M-fil) som man kan skicka indata till och som kan returnerar utdata. Funktionsfilen sparas i *funktionsnamn.m.*

Exempel 1 :

Följande är sparat som *square.m*

```
function result = square(x)
result = x.^2;
```

x är indata, square är namnet av funktionen och måste ha samma namn som funktionsfilen. I result sparas utdata.
I kommandofönstret anropas funktionen square med

```
>> svar = square(2);
>> svar
 svar =
          4
```

Exempel 2 :
Man kan ha flera in- och ut data

```
function [square, sum]= square(x, y)
square = x.^2;
sum = x+y;
```

I kommandofönster skriver man
>> [svar1, svar2]= square(2,3);

Två andra varianter att definiera en funktionen
med @

```
>> square = @(x) x.^2      %@ markerar variabeln för indata
>> result = square(2)
>> result =
 4
```

Med inline:

```
>> square = inline(' x.^2', 'x')          % först funktionen sen en lista med indata
>> result = square(2)
>> result =
 4
```

- **Logiska operatorer**
  De logiska operatorerna är:
  och &
  eller |

- **Villkorlig sats**
  IF – sats

```
if villkor
    body
else
    body
end
```

```
if villkor1
    body
elseif villkor2
    body
elseif villkor3
    body
end
```

Exempel

```
if x<0
   disp(-x)
else
    disp(x)
end
```

FOR-sats

```
for variable = uttryck
    body
end
```

Exempel

```
>> v = 0;
>> for i = 1:4
          v= v+i;
   end
```

- **Repetition**
  WHILE-sats

```
while villkor
      body
end
```

Exempel

```
n = 0;
while n<5
    n = n+1;
end
```