Virtuel Instrumentering: Matlab Introduktion 1

Samuel Alberg Thrysøe, PhD, PostDoc,
Kontakt info: Email: sat@iha.dk, Tlf: +45 25533552

Nyttig Info

- God Matlab Tutorial
  - An Introduction to Matlab
    David F Griffiths
  - Uploadet til sttvi.samle.dk
- Cody: godt øvelses site
  - www.mathworks.com/matlabcentral/cody
  - Site med mange opgaver
  - Point ved korrekt løsning
  - Giver adgang til nye opgaver

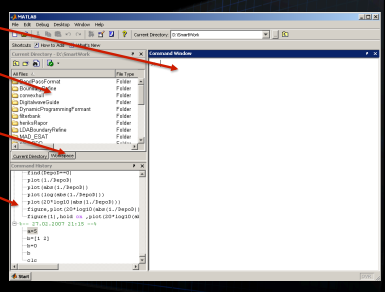
Matlab Opbygning + Variabler

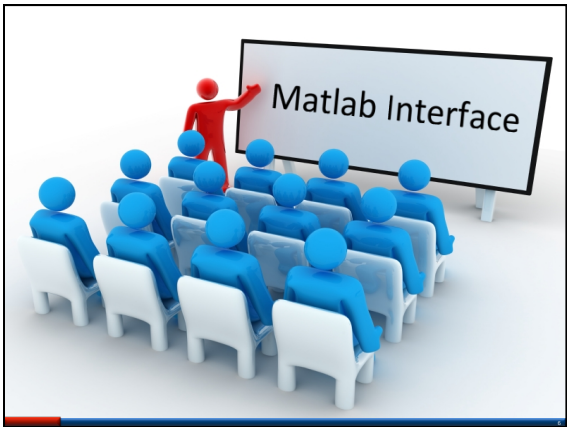
Hvad er Matlab?

- Høj-niveau programmerings sprog
- Let adgang til plots og data visualisering
- Fortolket sprog – ikke kompileret
  - For loops er ilde set
  - Kan dog benytte prækompilerede funktioner
    - MEX funktioner
- Brugerdefinerede funktioner & GUI design

Matlab Skærm

- Command Window
  - "Her og nu" kommandoer
- Current Directory
  - Filer/foldere i nuværende sti
- Workspace
  - Vis variable
  - Dobbeltklik variablen for at gennemse den
- Command History
  - Oversigt over tidligere kommandoer





Matlab hjælp

- Help *funktion*
  - Fx help mean
  - Giver inline funktionshjælp direkte i command window
- Doc *funktion*
  - Fx doc mean
  - Giver den fulde hjælpetekst i separat hjælpevindue

```
>> help mean
mean   Average or mean value.
    For vectors, mean(X) is the mean value of the elements in X. For
    matrices, mean(X) is a row vector containing the mean value of
    each column.  For N-D arrays, mean(X) is the mean value of the
    elements along the first non-singleton dimension of X.

    mean(X,DIM) takes the mean along the dimension DIM of X.

    Example: If X = [1 2 3; 3 3 6; 4 6 8; 4 7 7];

    then mean(X,1) is [3.0000 4.5000 6.0000] and
    mean(X,2) is [2.0000 4.0000 6.0000 6.0000].'
```

mean

Average or mean value of array

Syntax

```
M = mean(A)
M = mean(A,dim)
```

Description

`M = mean(A)` returns the mean values of the elements along different dimensions of an array.

If `A` is a vector, `mean(A)` returns the mean value of `A`.

If `A` is a matrix, `mean(A)` treats the columns of `A` as vectors, returning a row vector of mean values.

Variable

- Ikke nødvendigt at definere variable før brug

```
int a;
double b;
float c;
```

- Variable er som standard doubles

```
Eksempel:
>>x=5;
>>x1=2;
```

- Så x og x1 er 1x1 matricer med double præcision

Definitioner

- En vektor:

```
v = [1 2 3 4]
v=
    1 2 3 4
```

- En matrix:

```
m = [1 2 3;6 5 4;7 2 9]
m=
    1 2 3
    6 5 4
    7 2 9
```

Transponering

- En vektor:

```
v = [1 2 3 4]
v=
    1 2 3 4
```

- Transponering:

```
v'
v=
    1
    2
    3
    4
```

Genvej til lange matricer

- `1:10`

```
    1 2 3 4 5 6 7 8 9 10
```

- `5:-1:-5`

```
    5 4 3 2 1 0 -1 -2 -3 -4 -5
```

- `-1:0.25:1`

```
    -1 -0.75 -0.5 0.25 0 0.25 0.5 0.75 1
```

- `B=[1:3;7:9]`

```
B=
    1 2 3
    7 8 9
```

Vektorer fra funktioner

- `zeros(M,N)` MxN matrice af nuller

```
x = zeros(1,3)
x =
    0     0     0
```

- `ones(M,N)` MxN matrice af ettaller

```
x = ones(1,3)
x =
    1     1     1
```

- `rand(M,N)` MxN matrice af tilfældige tal mellem 0 og 1

```
x = rand(1,3)
x =
    0.9501   0.2311   0.6068
```

- `randi` Tilfældige heltal

Vektorer fra funktioner

- `linspace(x1,x2,n)`
  - Giver n værdier fra x1 til x2
  - `linspace(1,3,8)`
    - `1.0000  1.2857  1.5714  1.8571`
      `2.1429  2.4286  2.7143  3.0000`
- `logspace(x1,x2,n)`
  - Giver n værdier fra 10^x1 til 10^x2
  - `logspace(-4,1,6)`
    - `0.0001  0.0010  0.0100  0.1000`
      `1.0000  10.0000`

Matrix indeksering

- Matlab indexering begynder fra 1
  - Ikke 0 som i LabView!
- Indekseringer skal være positive heltal

```
A =
     3     5     3
     6     8     2
     2     7     3
```

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

- `A(-2) A(0)` ➔
  - Error: ??? Subscript indices must either be real positive integers or logicals.
- `A(4,2)` ➔
  - Error: ??? Index exceeds matrix dimensions.

Matricer

- Alt i Matlab gemmes som en matrix, som kan tilgås via rækker og kolonner.
  - HUSK: Først række, så kolonne
- `m(5,4)` ➔
  - 45
- `m(3:5,2:3)` ➔
  - `61  18`
    `19  17`
    `11  19`

	1	2	3	4	5
1	12	13	13	14	15
2	15	14	15	15	16
3	12	61	18	14	30
4	18	19	17	14	85
5	19	11	19	45	45
6	20	9	20	20	40
7	19	10	23	21	38
8	17	12	11	10	16
9	16	14	10	9	15
10	25	84	63	45	78

Tilføjelse af elementer

```
>> A=1:3
A=
   1   2   3
>> A(4:6)=5:2:9
A=
   1   2   3   5   7   9

>> B=1:2
B=
   1   2
>> B(5)=7;
B=
   1   2   0   0   7
>> B(end+1)=9;
B=
   1   2   0   0   7   9
```

```
>> C=[1 2; 3 4]
C=
   1   2
   3   4
>> C(3,:)=[5 6];
C=
   1   2
   3   4
   5   6
>> D=linspace(4,12,3);
D=
   4   8  12
>> E=[C D']
E=
   1   2   4
   3   4   8
   5   6  12
```

Matrice Sammenklistring Concatenation

- `x = [1 2], y = [4 5], z=[0 0]`
- `A = [x y]`
  `1  2  4  5`
- `B = [x ; y]`
  `1 2`
  `4 5`
- `C = [x y ;z]`

Error:
??? Error using ==> vertcat CAT arguments dimensions are not consistent.

Tekst Concatenation

- Fx dannelse af fødselsdato ud fra CPR nummer
  - `cpr = '260485-2142';`
- Dette skal på formen 'YYYY-MM-DD'
  - `foeddato = ['19' cpr(5:6) '-' cpr(3:4) '-' cpr(1:2)]`

```
foeddato =
    1985-04-26
```

Variabel typer

- Matrix
- String
  - Dette er også en matrix
- Cell
  - Kan indeholde forskellige typer/ længder af data
- Struct
  - Svarer til LabView cluster

- `A=[1 2 3;4 5 6;7 8 9]`
- `A='tekst'`
  - `A(3:end)` → `kst`
- `A={'Dette','er','en','tekst'};`
  - `A{1} = 'tekst';`
  - `A{2} = 3;`
- `A.tekst = 'Hello';`
- `A.aktiv = 1;`
- `A.matrix = [1 2 3];`

Operatorer (aritmetiske)

+ addition
- subtraktion
* multiplikation
/ division
^ power
‘ transponering

Matrix Operationer

Definer A and B:

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

```
>> B = [3 5 2; 5 2 8; 3 6 9]
B =
     3     5     2
     5     2     8
     3     6     9
```

Addition

```
>> X = A + B
X =
     4     7     5
     9     7    14
    10    14    18
```

Subtraction

```
>> Y = A - B
Y =
    -2    -3     1
    -1     3    -2
     4     2     0
```

Product

```
>> Z = A * B
Z =
    22    27    45
    55    66   102
    88   105   159
```

Transpose

```
>> T = A'
T =
     1     4     7
     2     5     8
     3     6     9
```

Array Operationer

- Evalueres element for element
  - `.'` : array transpose
  - `.^` : array power
  - `.*` : array multiplikation
  - `./` : array division
- Meget anderledes end Matrix operationer

```
>> A=[1 2;3 4];
>> B=[5 6;7 8];
>> A*B
    19    22
    43    50
```

Men:

```
>> A.*B
     5    12
    21    32
```

Brug af Element Operationer

```
A = [1 2 3; 5 1 4; 3 2 1]
A =
     1     2     3
     5     1     4
     3     2    -1
```

`x = A(1,:)` `y = A(3 ,:)`

```
x=
     1     2     3
```

```
y=
     3     4    -1
```

`b = x .* y` `c = x ./ y` `d = x .^2`

```
b=
     3     8    -3
```

```
c=
    0.3333    0.5000   -3.0000
```

```
d=
     1    16     9
```

Error:
??? Error using ==> mpower Matrix must be square.

Error:
??? Error using ==> mtimes Inner matrix dimensions must agree.

Nyttige teknikker/kommandoer

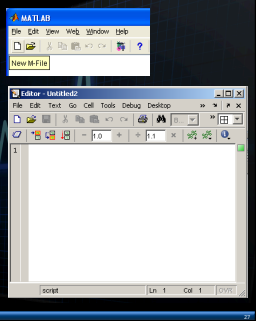
- Undertrykning af output:
  - Afslut kommandoer med ; (semikolon)
  - Fx: `a=[1 2 3];`
- Ryd command window:
  - `clc` (clear command window)
- Slet alle variable
  - `clear`
  - Eller et subset: `clear x a t`
- Slet figur
  - `clf` (clear figure)
- Luk alle figurer
  - `close all`
- Find funktion
  - `find(x>10)`
  - Giver index på alle x-værdier over 10

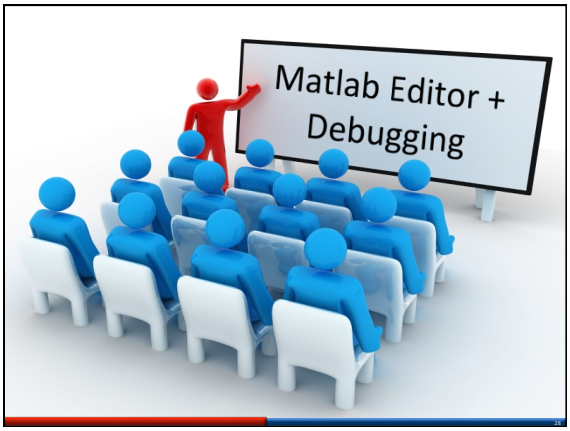
Nyttige kommandoer

- `mean(A)`: middelværdi af vektor
- `max(A)`, `min (A)`: maximum og minimum.
- `sum(A)`: summation.
- `sort(A)`: sorteret vektor
- `median(A)`: median værdi
- `std(A)`: standard deviation.
- `det(A)` : determinant af en kvadratisk matrix
- `dot(a,b)`: prik produkt af to vektorer
- `Cross(a,b)`: kryds produkt af to vektorer
- `size(A)`: Størrelsen på matrix A
- `length(A)`: Max. størrelse på matrix A

Matlab Editor

- Ctrl + N
- Tekst-editor
  - Keywords fremhæves
  - Code Analyzer
    - Viser fejl / forslag til programforbedring
  - Breakpoints
    - Klik tv for at sætte/fjerne breakpoint
    - Højreklik for at sætte betinget breakpoint





To typer af Editor Dokumenter

Scripts

- Samling af koder
- Fx plot af data
  - `x=linspace(-4,4,100);`
  - `y=x^2;`
  - `plot(x,y);`
  - `title('x^2');`
  - `xlabel('x');`
  - `ylabel('y');`

Funktioner

- Samling af kode, som modtager input og kun videregiver output
  - Interne variable kun tilgængelige for funktionen
  - Scope
- Eksempel:
  - `function res = kvadrat(tal)`
    - `res = tal ^ 2;`
  - `kvadrat(2)` → 4

Funktioner

- Definition af funktion:

```
function out1=functionname(in1)
function out1=functionname(in1,in2,in3)
function [out1,out2]=functionname(in1,in2)
```

- Dette skal stå øverst og funktionen skal gemmes med samme navn
- Mulighed for flere outputs og inputs
- Antallet af inputs kan være valgfrit
  - Det leverede antal inputs: varargin

Hjælp til Funktioner

- Under funktionsdefinitionen kan tilføjes hjælpetekst

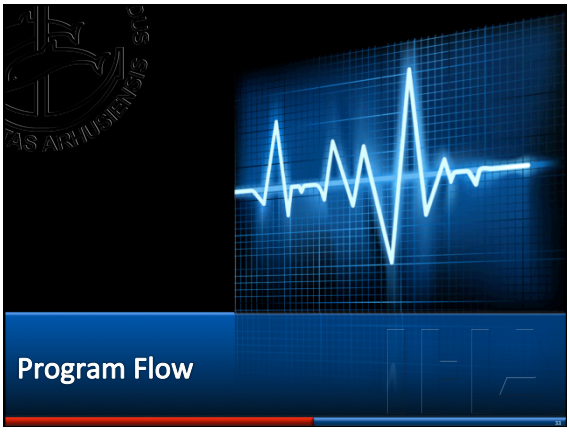
```
function res = kvadrat(tal)
% Denne funktion kan levere kvadrede tal
% Funktionskald:
% res = kvadrat(tal)
%   hvor:
%   res = Det kvadrerede tal, som returneres
%   tal = Input tallet, som skal kvadreres
res = tal^2;
```

- Skrives `help kvadrat` i command window'et bliver denne tekst vist

Cell Mode

- Inddeling af koder i mindre bidder, som kan køres uafhængigt
- Den aktuelle blok eller celle kan køres ved at trykke Ctrl+Enter
- Kun denne celle køres

```
%% Danne data
rnd = randi(100,20,5);
m = mean(rnd);
sd = std(rnd);

%% Plot af data
bar(m);

%% Visning af rådata
disp(sprintf('%2.2f (±%2.2f), ',m,sd));
```



Program Flow

Flow Control

- if-strukturer

```
if x>10
   status='high';
else
   status = 'low';
end
```

- for loops

```
for i=1:100
   b(i)=sin(i/10);
end
```

- Alternativ

```
x=0.1:0.1:10;
b=sin(x);
```

- Man kan ofte undgå for loops
- God ide i Matlab
  - Fortolket, ikke kompileret

Flow Control

- while loops

```
while(test<threshold)
   kommandoer
end
```

- switch strukturer

```
a=randi(4;
switch(a)
    case 1
        disp('low');
    case 2
        disp('medium');
    otherwise
        disp('high');
end
```

Logiske Operatorer

- == Lig med
- ~= Ikke lig med
- < Mindre end
- > Større end
- <= Mindre end eller lig med
- >= Større end eller lig med
- & Og operator
- | Eller operator

- `a == b`
- `a ~= b`
- `a < b`
- `a > b`
- `a <= b`
- `a >= b`
- `a>3 & b<c`
- `a>3 | b<c`

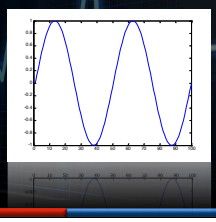
7.1 – Matlab Funktioner

- 7.1.1 – Kvadrater
- 7.1.2 – SumProd
- 7.1.3 – Temperatur Konvertering
- 7.1.4 – Fibonacci Sekvenser

Basale plots

Plotning af grafer

- Plot funktionen sin(x) mellem 0≤x≤4π
  - Beregn X-aksen, diskretiseret til 100 punkter:
    `x=linspace(0,4*pi,100)`
  - Beregn Y-aksen:
    `y=sin(x)`
  - Plot grafen:
    `plot(x,y)`



Styring af plot egenskaber

- `Plot(x,y,'ko:')`
  - Tekststrengen efter x,y angiver formatet
    - k = black
    - o = marker stil (runde cirkler)
    - : = linietypen; stiplet

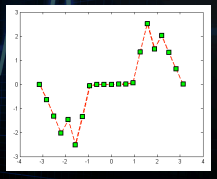
Plot muligheder

Kode	Liniefarve	Kode	Marker stil
b	blue	.	point
g	green	o	circle
r	red	x	x-mark
c	cyan	+	plus sign
m	magenta	*	star
y	yellow	s	square
k	black	d	diamond
		v	triangle down
Kode	**Liniestil**	^	triangle up
-	solid	<	triangle left
:	dotted	>	triangle right
-.	dashdot	p	pentagram
--	dashed	h	hexagram

Yderligere plot-egenskaber

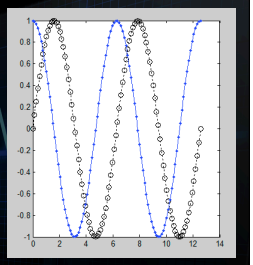
```
>> x = -pi:pi/10:pi;
>> y = tan(sin(x)) - sin(tan(x));
>> plot(x,y,'--rs','LineWidth',2,...
                  'MarkerEdgeColor','k',...
                  'MarkerFaceColor','g',...
                  'MarkerSize',10)
```

Mere info på
`doc plot`



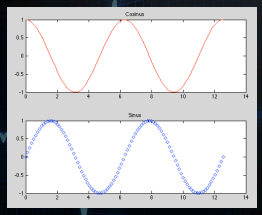
Flere plots i et

- På en gang
  - `x=linspace(0,4*pi,100);`
  - `y1=sin(x);y2=cos(x);`
  - `plot(x,y1,'ko:',x,y2,'b.-')`
- Brug af `hold` funktion
  - `hold on;`
  - `x=linspace(0,4*pi,100);`
  - `y=sin(x);`
  - `plot(x,y,'ko')`
  - `y=cos(x);`
  - `plot(x,y,'b.-')`
  - `hold off;`



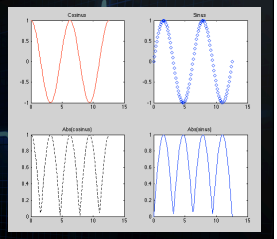
Subplots

```
>> x=linspace(0,4*pi,100);
>> f=figure;

>> f1=subplot(1,2,1);
>> plot(x,cos(x),'r');
>> title('Cosinus')

>> f2=subplot(1,2,2);
>> plot(x,sin(x),'d');
>> title('Sinus');
```



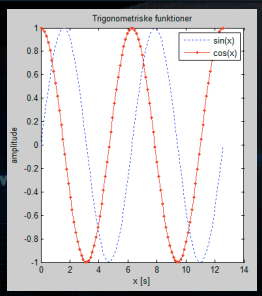
Subplots

```
>> x=linspace(0,4*pi,100);
>> f=figure;
>> f1=subplot(2,2,1);
>> plot(x,cos(x),'r');
>> title('Cosinus')
>> f2=subplot(2,2,2);
>> plot(x,sin(x),'d');
>> title('Sinus');
>> f3=subplot(2,2,3);
>> plot(x,abs(cos(x)),'k--');
>> title('Abs(cosinus)')
>> f4=subplot(2,2,4);
>> plot(x,abs(sin(x)),'b');
>> title('Abs(sinus)');
```



Plot annotering

- Data
  - `x=linspace(0,4*pi,100);`
  - `plot(x,sin(x),'b', ... x,cos(x),'r')`
- xlabel, ylabel
  - `xlabel('x [s]')`
  - `ylabel('amplitude')`
- title
  - `title('Trigonometriske funktioner')`
- legend
  - `legend({'sin(x)', ... 'cos(x)'};`



Plot annotering

- Mulighed for specialsymboler i titler og aksetitler
  - Græske bogstaver
  - Sub- og superscript
  - Matematiske symboler
- Indskrives med foranstående \ (backslash)
- Fx
  - `xlabel('0 \leq x \leq 4*\pi')`

$0 \leq x \leq 4\pi$

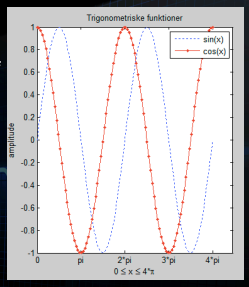
- Superscript
  - x^2 → x^2
- Subscript
  - x_1 → x_1

Get Egenskaber

- Get(handle)
  - Returnerer alle egenskaber til handlet
- Set(handle)
  - Returnerer alle de egenskaber, som kan ændres
- For at hente en specifik egenskab, fx Xdata for plot:
  - `x=linspace(0,4*pi,100);`
  - `y=sin(x);`
  - `h = plot(x,y);`
  - `get(h,'Xdata')`
- Nederste kommando giver x-værdierne

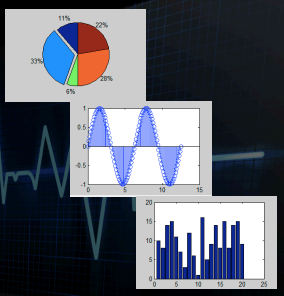
Set Egenskaber: Eksempler

- X-Tick labels
  - `set(gca,'XTick',0:pi:4*pi);`
  - `set(gca,'XTickLabel', …`
    `{'0','pi','2*pi', …`
    `'3*pi', '4*pi'});`
- Logaritmiske skalaer
  - `set(gca,'XScale','log');`
  - `set(gca,'YScale','log');`



Andre plot-typer

- pie charts
  - `x = [1 3 0.5 2.5 2];`
  - `explode = [0 1 0 0 0];`
  - `pie(x,explode)`
- Stem charts
  - `x = linspace(0,4*pi, 100);`
  - `y = sin(x);`
  - `stem(x,y)`
- Søjlediagram
  - `x=randi(20,[1,20]);`
  - `bar(x)`



At gemme plots

- Brug kommandoen:
  `saveas(h,'filename.ext')`
  for at gemme en figur til en fil.

```
>> f=figure;
>> x=-5:0.1:5;
>> h=plot(x,cos(2*x+pi/3));
>> title('Figur 1');
>> xlabel('x');
>> saveas(h,'figur1.fig')
>> saveas(h,'figur1.eps')
```

Filtyper:
bmp: Windows bitmap
emf: Enhanced metafile
eps: EPS Level 1
fig: MATLAB figur
jpg: JPEG billede
m: MATLAB M-fil
tif: TIFF billede, komprimeret

7.2 – Matlab Plots

- 7.2.1 – Subplot
- 7.2.2 – Hold funktion
- 7.2.3 – Terningekast
- 7.2.4 – Plot limit

Fil I/O

Fil I/O

- Matlab har eget format til at gemme/hente variable: *.mat
- Brug funktionerne `load` og `save` til at hente/gemme data

```
>> save var1 var2 var3
>> load savedvars.mat
```

- Desforuden kender Matlab rigtig mange filformater. Søg efter "File Formats" i hjælpen for yderligere info.

CSV filer

- Meget udbredt
  - Alle programmer (stort set) kan læse CSV filer
- Opbygning
  - En linie pr. post
  - Værdier adskilt af komma eller semikolon:

```
65;19;46;69
55;19;30;37
45;64;88;74
65;44;55;23
```

- `dlmwrite(filename, M, 'D')`
  - M = data
  - 'D' = Delimiter
    - Typisk:
      - Komma
      - Semikolon
      - Tabulator (\t)

Direkte fil-adgang

```
>> m = randi(100,5,2)
m =
    65    19
    46    69
    55    19
    30    37
    75    63
>> fid=fopen('d.txt','w');
>> fprintf(fid,'%i\t%i\n',m');
>> fclose(fid);

>> fid=fopen('d.txt','w');
>> fprintf(fid,'%i;%i\n',m');
>> fclose(fid);
```

Flere formater:
`doc fprintf`

Tekstfil-indlæsning

- importdata
  - `A = importdata(filename, delimiter, nheaderlines)`
- textscan
  - `C = textscan(fid, 'format', 'param', value)`
- xlsread
  - `[num,txt,raw] = xlsread(filename,sheet,range)`
- dlmread
  - `M = dlmread(filename, delimiter)`
- fscanf
  - `A = fscanf(fileID, format)`

File ID

- Nogle funktioner kræver, at der først oprettes adgang til filen via et File ID:
  - `fid = fopen('filnavn.csv')`
- Hvis ikke der specificeres yderligere åbnes filen med læseadgang
  - Skriveadgang (overskriv eksisterende fil)
    - `fid = fopen('filnavn.csv','w')`
  - Append data (tilføj data til eksisterende fil)
    - `fid = fopen('filnavn.csv','a')`
- Når operationerne er færdige
  - `fclose(fid)`

7.3 – Fil IO

- 7.3.1 – Marias Mad
- 7.3.2 – Tonometri data