Guide til det basale i MATLAB

Jens E. Wilhjelm

Ørsted•DTU, Ørsteds Plads, Bygning 349

Danmarks tekniske universitet

2800 Kgs. Lyngby

Forord

Denne guide er primært tiltænkt studerende i kurset 31654 Introduktion til medikoteknik, men alle der ønsker at sætte sig ind i MATLAB kan anvende den. Yderligere aspekter om MATLAB findes i de til kurset hørende opgaver i databaren. Al tekst som MATLAB skriver eller som man selv skal skrive i MATLAB er fortrinsvis angivet i skriftsnittet `Courier`. Bemærk at strenge (tekst i stedet for tal) ikke behandles særskilt, men listes ind hist og pist.

Bemærk at MATLAB bruger amerikansk notation for tal, så mens vi bruger decimal*komma* på dansk, så bruger MATLAB decimal*punktum*.

1 Indledning

MATLAB er et interaktivt, matrix-baseret program til videnskabelig og ingeniørmæssig numerisk beregning og visualisering. Man kan løse komplekse numeriske problemer på en brøkdel af den tid som det ville tage med konventionelle programmeringssprog som Fortran eller C. Navnet MATLAB er en forkortelse af MATrix LABoratory.

Læsevejledning: Hvis MATLAB skal kunne anvendes til løsning af tekniske opgaver, er det vigtigt at nærværende guide forstås fuldstændigt. Det er derfor en god ide at have MATLAB åben ved gennemgang af denne guide og prøve samtlige eksempler og øvelser af.

MATLAB har et godt hjælpesystem. Hvis du vil have flere oplysninger om en kommando som for eksempel kvadratroden (engelsk: square root), vil

```
>> help sqrt
```

give disse. `Help` alene giver en oversigt over hovedgrupperne af MATLAB-kommandoer, og `help general` giver en liste over generelle kommandoer.

Man starter MATLAB fra en PC ved enten at klikke på Matlabikonen på skærmen, eller ved at klikke på Start og derefter Programmer: Her vælges MATLAB 7.0. Fra en UNIX-terminal skrives kommandoen `matlab` i et kommandovindue (shell).

Når MATLAB åbnes fremkommer et vindue med flere undervinduer. `Command Window` er det centrale. De andre er kun hjælpevinduer og det anbefales i første omgang at lukke disse, for at give et bedre overblik (de kan altid åbnes igen via menuen "Desktop"). Kommandovinduet svarer med at skrive:

```
To get started, select MATLAB Help or Demos from the Help menu.

>>
```

">>" er MATLABs prompt, og man skriver MATLAB-kommandoerne efter den. I de følgende eksempler på kommandoer, skal >> ikke skrives. Man stopper MATLAB og returnerer til operativsystemet med kommandoen

```
>> quit
```

Aritmetik bruger den sædvanlige notation, som for eksempel

```
>> 2 + 3*5^2
```

MATLAB vil så svare (idet `5^2` læses som 5^2):

```
ans= 77
```

Variable tilskrives værdier (engelsk: *assignment*) med lighedstegnet "=". For eksempel

```
>> x = 7;

>> y = x^2;
```

Hvis man så vil udregne y/x skrives:

```
>> y / x
```

I de tre foregående eksempler er nogle sætninger afsluttet med semikolon (;), hvilket gør at MATLAB ikke viser resultatet af sin beregning. Det er en god ide at se svaret i begyndelsen, men som man opnår rutine, bliver det hurtigt overflødigt. I denne guide vil svarene fra MATLAB ikke nødvendigvis være angivet.

MATLAB skelner mellem store og små bogstaver; `X` og `x` er altså to forskellige variabelnavne. Hvis man ikke selv specificerer et variabelnavn, gemmer MATLAB det sidste resultat i variablen `ans`. Kommandoen

```
>> whos
```

giver en liste over de definerede variable og deres størrelse (`who` giver kun navnet). Listen viser med andre ord hvad der ligger i MATLABs såkaldte "Workspace" ("arbejdsrum", om man vil) . Disse kan slettes med kommandoen

```
>> clear
```

Hvis man vil slette en enkelt variabel, for eksempel `x`, skriver man `clear x`.

MATLAB har en række predefinerede variable. For eksempel er `i` = $\sqrt{-1}$, `j` = $\sqrt{-1}$ og `pi` ≅ 3,14, med mindre man selv har brugt `i`, `j` eller `pi` som et variabelnavn (hvilket sjældent er en god ide!). `i` og `j` har samme værdi og det er et spørgsmål om tradition og smag, hvilken man vil bruge. I denne guide bruges kun `i`. Den permanente variabel `eps` (epsilon) giver *machine unit roundoff,* ca. 10^{-16} på de fleste maskiner. Det er nogenlunde den mindste værdi MATLAB kan repræsentere.

MATLAB regner med komplekse tal. For eksempel definerer

```
>>s = 22e2 + j*2
```

det komplekse tal $22 \cdot 10^2+2i$. Bemærk, at der ikke må være ophold i et tal; $22 \cdot 10^2$ må altså ikke skrives som "`22 e 2`". Alle regninger foregår som standard[1] i dobbelt præcision (med ca. 16 betydende cifre), selv om ikke alle cifre i resultatet vises. Med ordren `Format` kan man ændre standardformatet og få resultater vist med flere cifre. Se enten under `help format` eller hængemenuerne (`File/Preferences.../Command Window/Numeric format`).

Man kan få tidligere afgivne kommandoer frem med piletasten (↑). Disse kan så eventuelt rettes og afgives igen ved at trykke på Enter-tasten. Dette letter skrivearbejdet meget. Skriver man selv de(t) første bogstav(er) i den ønskede kommando, inden man trykker på ↑, viser MATLAB kun de kommandoer, der begynder med disse bogstaver.

Når MATLAB kører, "står" programmet i en given folder (eller direktorie). Man kan se hvor, ved blot at skrive `cd` (*change directory*). Man kan ændre dette til et hvilketsomhelst andet, ved blot at give et argument til `cd`. For eksempel: `cd u:\users\blabla`, hvor folderen `u:\users\blabla` skal eksistere. Når data og programmer senere skal skrives på disk, er det en god ide at "stå" på sit DTU netværksdrev. Mere herom senere. Hvis man iøvrigt vil sende kommandoer til operativsystemet kan dette gøres direkte fra MATLABs prompt ved at foranstille et udråbstegn (!). En liste over filer fås for eksempel med:

```
>> !dir
```

Inden det næste afsnit bør der ryddes op ved at skrive

```
clear all;
clc;
```

hvorved alle variable slettes og al tekst i kommandovinduet slettes.

2 Matricer

Som navnet antyder er MATLAB matrix-baseret. En matrice med seks elementer,

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \tag{1}$$

dannes med

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

hvor det bemærkes at

- Elementerne er omsluttet af `[` og `]`
- Først angives første række, dernæst anden række o.s.v.
- Rækker adskilles med semikolon ";" (eller ny linie). Elementerne i en række adskilles af kommategnet "," eller mellemrum. (Pas på overflødige mellemrum).

Øvelse: Skriv `A` (uden semikolon) ved prompten og se om resultatet er rigtigt.

Dimensionen af `A` er vigtig. Ved at skrive

1. Der findes i dag et væld af dataformater, som det ses af kommandoen `help datatypes`.

```
>> size(A)
```

ses svaret at være:

```
3   2
```

hvilket er en vektor med to elementer. Der er således tre rækker (nedad) og to søjler (henad) i `A`. Hvis vi vil udtrække værdien af det nederste højre element af `A`, skal vi således skrive

```
>> A(3,2)
```

hvilket giver 6. Det er vigtigt at kunne dette udenad.

Øvelse: Hvad sker der, hvis vi skriver `A(2,3)`? Det er vigtigt at forstå MATLABs svar, for dette svar kan nemt komme igen inden dagen er omme. Hvad er `size(size(A))`?

Med kolon-operatoren, ":", kan man definere lister. For eksempel definerer

```
>>  t = -1 : 0.01 : 1;
```

rækkevektoren t = (-1 -0,99 -0,98 -0,97 .... 0,97 0,98 0,99 1); Der står altså "startværdi : tilvækst : slutværdi". Hvis tilvæksten er 1, kan den udelades og man kan skrive: `t = -1 : 1;`

Øvelse: Hvad er resultatet af `t = -1 : 1`? Resultatet af `size(t)`? Resultatet af `length(t)`?

Øvelse: Hvad er forskellen på `size` og `length`?

Kommandoerne `zeros(M,N)` og `ones(M,N)` definerer en M×N-matrix fyldt med henholdsvis 0-er og 1-er. Disse kan naturligvis kombineres, for eksempel giver

```
>> A = [zeros(1,3), ones(1,2); 1 2 3 4 5; ones(1,5)];
```

matricen

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

Øvelse: Ovenstående er ikke helt simpelt. For at lette forståelsen, er det en god ide at udføre sætningen lidt af gangen; start for eksempel med `A = [zeros(1,3), ones(1,2)]` og byg så langsomt op med en linie ad gangen.

Øvelse: En anden standardmatrix er `eye`. Prøv for eksempel `eye(4,4)`. Hvad er størrelsen?

Mellem på den ene side en matrix, `A`, eller skalar `a` og på den anden side en skalar, `b`, kan man anvende de normale regneoperationer:

```
+   addition (A+b)
-   subtraktion (A-b)
*   multiplikation (A*b)
/   division (A/b)
.^  potensopløftning af alle elementer (A.^b)
```

Øvelse: Brug den store matrix ovenfor, og lad `b` = 4. Anvend derefter de fem regnearter.

Hvis man har to matricer af samme størrelse, `A` og `B`, kan man tilsvarende bruge

```
+   addition
-   subtraktion
.*  multiplikation
./  division
```

hvor operationerne er elementvise (punktummet angiver elementvise operationer). Det bemærkes at `size(A)=size(B)`.

Øvelse: Lad `A=ones(2,3)` og `B = 3*ones(2,3)`. Prøv nu ovenstående fire regnearter på `A` og `B`. Lad derefter `B = 3*ones(3,2)`; hvorfor virker dette ikke?

Denne samme udregning kan udføres på mange måder. For eksempel vil

```
>> a = [1 2 3]

>> b = a .* a
```

give resultat b = (1, 4, 9). Det samme ville

```
>> b = a .^ 2;
```

Hvis man bruger en af de almindelige funktioner såsom `sin` og `cos` på en matrix, anvendes funktionen på hvert element for sig. For eksempel giver

```
>> b = [0 pi/2 pi];

>> c = sin(b)
```

resultatet c = (0, 1, 0). Andre simple funktioner kan findes med `help elfun`.

Der findes yderligere to operatorer

`.'`	transponering `(A.')`
`'`	konjugering og transponering (berøres ikke i denne guide).

Transponering drejer matricen `90°`. Lad `A = [ 1 2 3; zeros(2,3)]`. Sammenlign `A` og `A.'`.

Slutteligt en simpel ting: Man kan indsætte kommentarer ved brug af "%". Dét der er skrevet fra % og linien ud vil blive ignoreret af MATLAB. Eksempel:

```
>> b = [0 pi/2 pi]; % vinkel i radianer
```

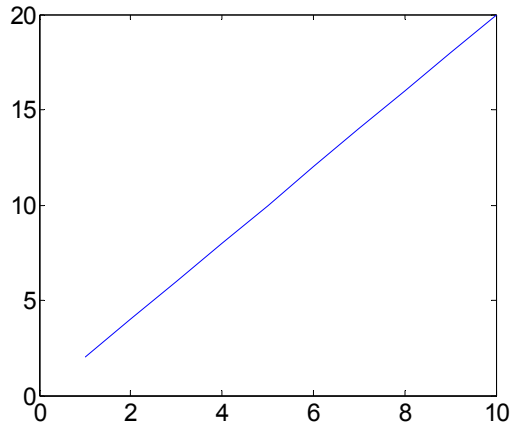
3 Matricer med komplekse tal

MATLAB tillader komplekse tal i alle sine operationer og funktioner. To praktiske måder at indtaste komplekse matricer er:

A= [1 2; 3 4] + i*[5 6; 7 8];

A= [1+5i 2+6i; 3+7i 4+8i];

Når man opfører komplekse tal (for eksempel 2+6i) i en matrix, skal man undgå mellemrum mellem `i` og tallet, da mellemrum bruges til at adskille elementerne i matricen. Bemærk, at normalt skal man altid



Figur 1 Resultatet af plot(A). Bemærk at den vandrette akse angiver index til A, som starter ved 1 og slutter ved antallet af elementer.

anvende gangetegnet når to variable skal ganges sammen, men for den imaginære enhed er dette ikke nødvendigt: `5i` virker, men det gør `i5` ikke, da MATLAB tror at `i5` er en variabel.

Inden der fortsættes, bør man rydde op med `clear all; clc;`.

4 For-sætningen

For-end-sløjfen går ud på at gentage en kommando (eller gruppe af kommandoer) og den kan bedst illustreres med et eksempel:

```
N = 10;
for n = 1:N,
    disp( n);
    A(n) = 2*n;
end;
```

Denne lille programstump består af 5 linier. Alle kommandoerne mellem `for`-linien og `end`-linien gentages det antal gange der er specificeret i `for`-linien. De to linier ind imellem udføres i dette tilfælde 10 gange (altså: `n=1:N` svarer til `n=1:10` som svarer til `n = 1 2 3 4 5 6 7 8 9 10`). Hver gang skrives `n` ud og 2`n` sættes ind i vektoren `A` på `n`'te plads. Bemærk iøvrigt, at når de fem linier indtastes med "enter" efter hver, er MATLAB så snilt indrettet, at den først udfører løkken, når den møder `end`. Længden af A øges iøvrigt med én, i hver iteration.

Ovenstående lille programstump kan også indtastes i en editor, og kopieres derfra til MATLABs kommandovindue, hvis man syntes det er mere praktisk.

Øvelse: Undersøg værdierne af `A` og størrelsen af `A`.

Hvis man nu kom til at tildele N værdien 100.000, så ville MATLAB skrive 100.000 tal ud, og det kunne jo være at man blev lidt træt af det inden MATLAB blev færdig. Heldigvis kan man "dræbe" en process som får kommandovinduet til at "løbe løbsk" ved at trykke Ctrl-C (både i Windows og i LINUX). Man stopper ikke MATLAB derved.

Inden der fortsættes, bør man rydde op med `clear all; clc;`.

5 if-sætningen

Med en sætning af typen if-else-end kan man foretage valg i MATLAB:

```
n = 1;
if n < 2,
    A(n) = n;
else
    A(n) = n.^2;
end;
```

hvor det logiske udtryk (der enten kan være sand, 1, eller falsk, 0) er "`n < 2`". Ovenstående programkode kan godt være svært at forholde sig til den første gang den ses, men tricket er, at læse hver linie grundigt og danne sig et mentalt billede af hvad den laver inden man går videre: I den første linie sættes `n` lig 1. I den næste stilles spørgsmålet om 1 er mindre end 2. Svaret er "sand" og sætningen umiddelbart nedenfor udføres. Linie 4 og 5 ignoreres. Slutteligt er `if`-sætningen færdig i linie 6. Der er følgende logiske operatorer:

```
<   mindre end
>   større end
<=  mindre eller lig med
>=  større eller lig med
==  lig med
~=  ikke lig med
```

Bemærk her, at `=` bruges ved tildeling af en værdi til en variabel, mens `==` altså bruges som logisk operator.

Øvelse: Sæt `n = 3` ovenfor og se hvad `A` bliver. Hvad bliver `A(1)` og `A(2)` og hvorfor?

6 Grafik

I eksemplet med `for`-løkken ovenfor, bliver `A` tildelt værdierne 2, 4, 6, 8, 10, 12, 14, 16, 18, 20. Disse værdier kan plottes med

```
>> plot( A);
```

hvorved et grafisk vindue åbner sig. Resultatet ses i figur 1. Da der er 10 elementer i A, løber den vandrette akse fra 1 til 10. Den lodrette akse løber fra 2 til 20.

Nu til et mere omfattende eksempel, som bruger en del af kommandoerne ovenfor. Først dannes en tidsakse:

```
>> t = 0 : 0.08 : 10; % s
```

som løber fra 0 til 10 sekunder i spring af 0,08 sekunder. Skriv `t(1:5)` for at se at tallene "springer" med 0,08. Bemærk, at det er op til brugeren selv at holde styr på enheder, så her er kommentarer værdifulde. Derefter udregnes et sinus-signal med frekvensen 1 Hz:

```
>> f = 1; % Hz
>> g = sin( 2 * pi * f * t); % V
```

Dette signal plottes nu

```
>> plot( t, g);
```

og resultatet ses i figur 2. Bemærk forskellen til `plot(A)` ovenfor, hvor `plot` blev kaldt med kun eet argument. Det bemærkes at den vandrette akse løber fra 0 til 10 (for det var sådan tidsaksen blev

defineret) og at den lodrette akse løber mellem –1 og 1, idet dette er udfaldsrummet for sinusfunktionen. Der kan nu tilføjes passende benævnelse af akserne og grafen:

```
>>xlabel('t (s)');  ylabel('g(t) (V)');  title('g(t) = sin(2\pift), f = 1 Hz');
```

hvor det ses at man sagtens kan have mere end en sætning per linie. Resultatet er de røde tekster i figur 2. Kommandoerne skal stå lige efter plot-kommandoen. Bemærk `\pi` ovenfor. Denne giver det tilsvarende græske bogstav, π. Man kan rigtig meget af den slags, hvorfor det kan virke endog meget overvældende (skriv `helpdesk` og søg efter funktionen `text`).

Hvis man undersøger længden af `t`, ses det at den er 126 elementer lang. Det vil `g` også være, da den er afledt af `t`. De 126 værdier i g forbindes med lige streger i plottet, og det er grunden til at det ser lidt "kantet" ud. Hvis man nu vil være sikker på at kunne se hvor hvert punkt er, kunne man blot udnytte at plot kan plotte mange kurver samtidig:

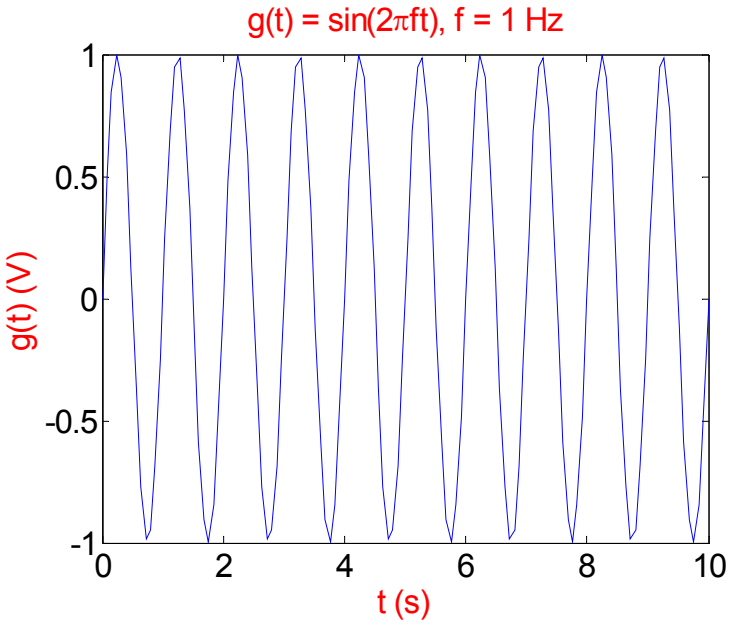
```
>> plot( t, g, t, g, '.');
```

Parret t-g er gentaget to gange og efter det er angivet en yderligere detalje om hvordan den sidste kurve skal fremstå, nemlig blot som prikker (der så er tegnet ovenpå den forrige kurve). De fremkomne farver er bestemt af MATLAB. Man kan dog bestemme det hele selv, hvis man vil:

```
>> plot( t, g, ':k', t, g, '.k');
```

hvor `':'` betyder stiplet og `'k'` betyder sort. Hele repertoiret af muligheder ses ved at skrive `help plot`. Hvis det ovenstående var uklart, så prøv at udføre disse seks linier i MATLAB (de kan kopieres direkte fra denne pdf-fil til MATLABs kommandovindue):

```
>> t = 0 : 0.08 : 10;
>> f = 1; % Hz
>> g = sin( 2 * pi * f * t);
>> subplot(3,1,1);  plot( t, g);  title( 'plot(t,g)' );
```



Figur 2 Plot af g(t). De tre røde tekster er resultatet af en række senere kommandoer.

```
>> subplot(3,1,2);  plot( t, g, '.');  title( 'plot( t, g, ''.'')' );
>> subplot(3,1,3);  plot( t, g, t, g, '.');  title( 'plot( t, g, t, g, ''.'')' );
```

hvor det også ses, at ved flere kurver bliver den første automatisk blå, den næste grøn. Flere detaljer kan findes i `help plot`.

`Plot` vælger selv længden af akser. De kan ændres med kommandoen `axis([xmin, xmax, ymin ymax])`, hvor `xmin, xmax, ymin` og `ymax` er de ønskede endepunkter for akserne.

Kommandoen subplot (m,n,p) opdeler det aktuelle grafiske vindue i en mxn-matrix af mindre grafiske vinduer, og vælger det p'te af disse til det aktuelle plot.

```
>> subplot(2,1,1), plot(1:100); title('subplot(2,1,1)');
>> subplot(2,1,2), plot(1:50); title('subplot(2,1,2)');
```

7 m-filer

Det ses af ovenstående kode til figur 2, at der hurtigt bliver brug for en længere række kommandoer, før man får det ønskede resultat. Det er naturligvis uhensigtsmæssigt at skrive disse direkte i MATLAB hver gang man vil have sin figur vist. Hvis kommandoerne til frembringelse af figur 2 skrives i filen `miteksempel.m` , kan man få udført dem ved at taste ordren

```
>> miteksempel
```

i MATLAB. `miteksempel.m` er nu et såkaldt *script*. MATLAB skal kunne "se" filen `miteksempel.m`, så i første omgang skal filen ligge i det direktorie som MATLAB "står" i. Man kan også danne *funktioner* på denne måde, se næste afsnit.

8 Funktioner

En funktion giver mulighed for at løse en afgrænset opgave for sig selv. Det foregående script kunne *kalde* en funktion, lad os fx kalde den `mypower`. I selve scriptet (som jo er en fil) står der så fx

```
b = mypower(a)
```
[1]

hvor `b` er det output som funktionen giver og som man kan arbejde videre med, mens `a` er input til funktionen og selvfølgelig kendt på det tidspunkt ovenstående kommando i scriptet afvikles. I *en anden fil* laver man så sin funktion. Filens navn skal være `mypower.m`. Indholdet vil så se sådanne ud:

```
function y = mypower(x)
y = x.^2;
```

Der sker nu følgende: scriptet kommer til `b = mypower(a)`, hvor `a` er kendt. `a` sendes til funktionen, og så snart den er der, bliver den kendt ved navnet `x`. Her udregnes så `y`, som sendes tilbage til scriptet og overføres til værdien `b`. `x` og `y` er altså med andre ord *lokale* variable, helt uafhængigt af scriptet. Man kunne således sagtens have opereret med `x` og `y` både i scriptet og funktionen (men de vil altid være lokale i funktionen). Funktioner er beskrevet nærmere i den åbne opgave oa_fun i kursus 31654.

1. Hvis man skriver denne linie nu, så går det galt, for funktionen er endnu ikke defineres!

9 Save and load

Man kan gemme sit *workspace* eller enkelte variable med `save` og senere indlæse dem i MATLAB igen med load. Hjælpen til disse to funktioner er ret overskuelig, prøv `help save`.

Prøv følgende: Definer en tekstvariabel, *e.g.,*

```
MyText = 'MATLAB is fun';
```

Skriv så

```
save MineData MyText;
```

Nu bliver variablen `MyText` gemt i filen `MineData.mat`. Skriv så `clear` og se med `whos` at *workspace* er tomt. Brug så slutteligt

```
load MineData;
```

for at indlæse `MineData.mat` og se med `whos` at `MyText` er kommet tilbage. Husk at slette filen bagefter, hvis den ikke skal bruges.

10 Dokumentation

Hvis man har brug for teksten i MATLABs kommandovindue, så bruger man bare de normale kopi-funktioner i Windows. Hvis man skal bruge en figur i for eksempel en rapport, kan man lagre figuren på disk med

```
>> print -dmeta plot_of_a;
```

I dette tilfælde bliver figuren lagret i Windows Metafile-format. Hvis der er tale om større figurer, er det ofte en fordel at bruge PostScript:

```
>> print -depsc2 plot_of_a;
```

Slutteligt er der en diary funktion, hvormed man kan få skrevet hele MATLABs kommandovindue ned i en fil. Se `help diary`.

11 Fejlmeddelelser

Den der laver noget, laver typisk også fejl. Fejl i MATLAB er normalt trælse i starten, så her kommer et par eksempler. Skriv:

```
blabla
```

hvorefter MATLAB normalt vil svarer:

```
??? Undefined function or variable 'blabla'.
```

Det skyldes at `blabla` ikke er kendt af MATLAB. `blabla` er altså hverken en indbygget funktion eller variabel og ej heller en brugerdefineret variabel eller fil med navnet `blabla.m` i de(t) direktorie(r) MATLAB søger i. Næste eksempel:

```
>> zeros(1,2) + ones( 2,2)
```

hvorefter MATLAB svarer

```
??? Error using ==> plus
Matrix dimensions must agree.
```

Her er problemet at brugen af "+" kræver at de to matricer der skal lægges sammen, *er lige store*. Det sidste eksempel:

```
a = [1 2; 3 4]
a(3,2)
```

giver

```
??? Index exceeds matrix dimensions.
```

idet der ikke er noget element i tredie række og anden søjle. Slutteligt findes yderligere en funktion, hvis begrundelse forfatteren aldrig rigtig har fattet: `why`.

12 Toolbokse

MATLAB har en lang række værktøjskasser (toolboxe) med funktioner, som købes seperat. Det vil sige at ikke alle funktioner er tilgængelige i en given installation.

13 Litteratur og ressourcer

MATLAB har et meget veludviklet web-baseret hjælpesystem. Det nås med:

```
>> helpdesk
```

Iøvrigt ligger der på MATLABs installationsfolder en række pdf-filer med mere udførlig hjælp. Der eksisterer vistnok stadig trykte manualer, men disse må antages at blive udfaset inden for nær fremtid. På internettet findes mange vejledninger, hvoraf nogle af dem er angivet på hjemmesiden til kurset 31654.

14 Taksigelser

Tak til studerende på Med-Tek Laura Frisenfeldt Horn, Aðalheiður Hanna Björnsdóttir og Michael J. Pihl for kritisk gennemlæsning af denne guide.

Denne guide er inspireret af tidligere guider til MATLAB, blandt andet B. Guldbrandsen: Kort introduktion til MATLAB, forår 1998, DTU.

15 Appendiks

Nedenstående tabeller viser de fleste af de funktioner man får brug for i det indledende arbejde med MATLAB.

Tabel 1: Liste over praktiske kommandoer

MATLAB apsekt	Beskrivelse	Eksempel
`clc`	Tømmer kommandovinduet for tekst	
`clear`	Fjerner alle variable	

Tabel 1: Liste over praktiske kommandoer

MATLAB apsekt	Beskrivelse	Eksempel
`whos`	Viser indhold af workspace	`clear; a=2; whos; clear; whos` (indtast dem een ad gangen)
`close all`	Lukker alle grafiske vinduer	
`%`	Kommentar	Al tekst fra % til slutningen af linien ignoreres af MATLAB

Tabel 2: Liste over de matematiske operationer

MATLAB apsekt	Beskrivelse	Eksempel
`"a = b";`	Assignment (*tildeling*)	`a = 2; b = a; b` giver `2`
`[]`	Forming vectors, etc.	`c = [ 3 2 1 5 ];`
`, og end`	Indeksering	`c(1)` giver `3`. `c(end)` giver `5`
`size`	giver size of variables	`size(c)` giver `[1 4]`
`length`	Length of vectors	`length(c)` giver `4`
`.* ./ .+ .-`	Se afsnit 2	`a+c` giver `[ 5 4 3 7 ]`
`* / + -`	Se afsnit 2	
`.^ and .`	Potensopløftning	`a.^2` giver `4`. `a^2` giver også `4`. Der er først forskel når `a` er komplex.

Tabel 3: Liste over programstrukturer

MATLAB apsekt	Beskrivelse	Eksempel

`if else end`	Valg	`if a == 1,`
		`  disp('a is 1');`
		`else`
		`  disp('a is not 1');`
		`end`

`for end`	Gentagelser	`for n = 1:N`
		`  disp('n = ', n);`
		`end`

Tabel 4: Vektorfunktioner (det vil sige at størrelsen af output matrix/vektor er mindre end størrelsen af input matrix/vektor)

MATLAB apsekt	Beskrivelse	Eksempel
`min, max`	Minimum og maximum	`c = [ 3 2 1 5 ];` `min(c)` giver `1`
`mean, median, std`	Middelværdi, median og standard-savigelse	
`any, all`	Blot eet element eller alle elementer	
`sum`	Calculating sum of numbers	`sum(c)` giver `11`
`prod`	Produktet af alle tal	

Tabel 5: Skalarfunktioner (det vil sige at størrelsen af output er lig størrelsen af input)

MATLAB apsekt	Beskrivelse	Eksempel
`sin, cos, tan`	Trigonometriske funktioner	`sin(pi)`
`exp, log10`	Eksponentialfunktion, 10-talslogar-itme	
`abs`	Absolut værdi	`abs(-1)` giver `1`
`round`	Afrunding	`round(-0.5)` giver `-1`
`floor, ceil`	Afskæring opad og nedad	`ceil(0.5)` giver 1 `floor(0.5)` giver 0
`sign`	Fortegn	`sign(-6)` giver `-1`
`sort`	Sortering	`c = [ 3 2 1 5 ];` `sort(c)` giver `[ 1 2 3 5 ]`
`rem`	Rest efter division	`rem( 3, 2)` giver `1`
`sqrt`	Kvadratrod	`sqrt(a)` giver `1.41`

Tabel 6: Specielle matricer

MATLAB apsekt	Beskrivelse	Eksempel
`zeros, ones`	Matrix med 0 eller 1	
`eye`	"Identify matrix"	
`rand`	Rektangulært fordelte tilfældige tal	
`randn`	Normalfordelte tilfældige tal	

Tabel 7: Specielle variable

MATLAB apsekt	Beskrivelse	Eksempel
`ans`	Svar på sidste udregning	
`eps`	Nogenlunde mindste tal der kan repræsenteres	
`pi`	3,14.....	
`i, j`	Imaginære enhed	
`inf`	Uendelig	
`NaN`	Ikke et tal (Not a number)	
`computer`	Computertype	

Tabel 8: Tekstfunktioner

MATLAB apsekt	Beskrivelse	Eksempel
`strings`	Tekst, imodsætning til tal	`d = 'medicin'; d(2)` giver `'e'`
`[]`	"Limer" strenge sammen	`a = 'Hej'; b = 'Mona'; c = [ a ' ' b '!']` giver `Hej Mona!`
`blanks`	Mellemrum	
`num2str`	Konverterer fra tal til tilsvarende tekst	`['c = ' num2str(c)]`
`disp`	Skriver tekst ud til kommandovinduet	`disp(['c = ' num2str(c(1))])` **giver c = 3** `disp([' ']);` **giver en tom linie**
`str2num`	Konverterer fra tekst til tilsvarende tal	
`upper, lower`	Konverterer til henholdsvis store og små bogstaver	

Tabel 9: Plot

MATLAB apsekt	Beskrivelse	Eksempel
`plot, bar`	Plotter data	
`xlabel, ylabel`	Navngiver akser	
`title`	Placerer titel på plot	
`grid`	Sætter hjælpelinier på plot	

Tabel 9: Plot

MATLAB apsekt	Beskrivelse	Eksempel
`text`	Skriver tekst på plot	
`hold`	Fryser eksisterende plot, så man kan skrive oven på	
`axis`	Justerer længden af akser	
`image, imagesc`	Viser en matrix som et farvebillede	
`colorbar`	Viser farveskalaen, som bruges	
`colormap`	Ændre farveskalaen	

Tabel 10: Håndtering af figurvinduer

MATLAB apsekt	Beskrivelse	Eksempel
`figure`	Laver en ny figur	
`gcf`	Returnerer "handle" til aktuel figur (= *current figure*)	
`clf`	Sletter indholdet af aktuel figur	
`close`	Lukker aktuel figur	
`subplot`	Inddeler en figur i mindre plotvinduer	
`print`	Skriver en figur ud eller ned på disk	