IMM
INSTITUT FOR MATEMATISK MODELLERING

Danmarks Tekniske Universitet
2800 Lyngby

10.9.1996

INTRODUKTION TIL MATLAB

ved IMM's kurser

1. udgave

Lars Henrik Hansen
Hans Bruun Nielsen

IMM

Forord

Formålet med denne note er at give nye brugere af MATLAB® en hurtig men fyldig introduktion til dette sprog, som bruges ved grupperegning i en række IMM-kurser.

Læseren forudsættes bekendt med computer-faciliteterne ved IMM og/eller G-databaren.

Denne note er 1. udgave. Ethvert forslag med henblik på rettelser eller behov for yderligere introduktion modtages med tak.

LHH, HBN

Indhold

1. Indledning

1.1. Baggrund

MATLAB er et interaktivt beregningsprogram. På grund af den fleksible opbygning, anvendes MATLAB som beregnings-/simuleringsværktøj på mange fagområder.

Før den nye bruger går i gang med introduktionen til MATLAB's interaktive miljø, er det måske en god idé at have lidt kendskab til programmets opbygning.

1.2. Hvad er MATLAB

MATLAB dækker over et større program-kompleks, bestående af et hovedprogram samt et antal TOOLBOX'e. I hovedprogrammet findes et stort antal standard funktioner, se appendiks A.

En TOOLBOX indeholder MATLAB funktioner (hjælpe programmer) til et specielt fagområde. Som eksempel kan Kontrol-toolboxen nævnes. Den indeholder programmer til beregning af bodeplot, frekvenskarakteristikker, etc., samt forskellige typer regulatorer.

Ved IMM er der adgang til MATLAB fra `serv1`. I G-databaren kan MATLAB startes på: `bohr`, `erlang`, `hald`, `lillep`, `lorenz`, `oersted`, `roemer` og `tycho`.

I tabel 1.1 er der givet en oversigt over TOOLBOX'ene. I tabellen er TOOLBOX'enes engelske navn angivet og *kort-navn* angiver TOOLBOX navnet i MATLAB (dvs. `help` ordren virker med dette navn). Stjernen ved Wawelet-TOOLBOX'en indikerer, at denne tænkes anskaffet i efteråret 1996. Det bør lige bemærkes, at SIMULINK faktisk er en slags grafisk udgave af MATLAB.

TOOLBOX	kort-navn	serv1	GBAR
Control System	*control*	ja	ja
ODE-suite	*odebox*	nej	ja
Optimization	*optim*	nej	ja
Rudimentary statistics	*stixbox*	ja	ja
Signal Processing	*signal*	ja	ja
Splines	*spline*	nej	ja
System Identification	*ident*	ja	ja
Wawelets	*wawelet*	nej	nej*
SIMULINK	*simulink*	ja	nej

Tabel 1.1. *TOOLBOX'e*

Næste afsnit indeholder introduktionen til MATLAB. Appendiks A indeholder en liste med basale MATLAB-funktioner. I appendiks B findes en liste over de enkelte IMM-kursers brug af MATLAB.

2. Introduktion til MATLAB

Dette afsnit giver en hurtig men fyldig introduktion til MATLAB. Som grundlag for afsnittet ligger en tidligere note udarbejdet af HBN.

2.1. Indledning

MATLAB aktiveres i et sædvanligt UNIX-vindue ved kommandoen:

`matlab` eller `nice matlab`

Eller på PC'ere ved at dobbelt-klikke på MATLAB iconen.

MATLAB er et udtrykssprog. Den typiske ordre har formen:

variabel = *udtryk*

Udtrykket tolkes og udregnes af MATLAB, hvorefter værdien tilskrives variablen. Hvis der i forvejen eksisterer en variabel med dette navn, erstattes dens værdi med den nye. Ellers bliver variablen oprettet samtidig med tilskrivningen. Eksempel:

```
>> a = 5/4
```

MATLAB kvitterer med

```
a =
   1.2500
```

Derefter kan man f.eks. skrive udtrykket

```
>> sin(pi*a)
```

og MATLAB kvitterer med

```
ans =
   -0.7071
```


Bemærk, at `pi` er en prædefineret variabel med værdien af π, at `sin` er en indbygget funktion (jvf. afsnit 2.2), – og at hvis man blot skriver *udtryk*, bliver værdien tilskrevet en *variabel* med navn `ans` (for "answer").

En ordre sendes afsted ved tryk på $\boxed{\text{enter}}$ eller $\boxed{\text{return}}$. Man kan skrive flere ordrer per linie. De skal da adskilles med "," (komma) eller ";" (semikolon). Sidstnævnte bevirker også, at udskriften undertrykkes. Eksempel:

```
>> a = 5/4,  sin(pi*a)
```
(2.1)

resulterer i udskriften

```
a =
   1.2500
ans =
   -0.7071
```

mens

```
>> a = 5/4;  sin(pi*a)
```

blot giver

```
ans =
   -0.7071
```

MATLAB er velegnet til beregninger med *talsæt* (vektoror og matricer). Eksempel:

```
>> x = [1  2  3  4];  y = sin(0.5*pi*x)
```

giver

```
y =
   1.0000  0.0000 -1.0000   0.0000
```

I den kantede parentes i udtrykket for `x` kan man erstatte mellemrummene med "," (komma).

Hvis man skriver

```
>> z = [1; 2; 3; 4],  u = sin(0.5*pi*z)
```

får man

```
z =
    1
    2
    3
    4

u =
     1.0000
     0.0000
    -1.0000
     0.0000
```

Man siger, at talsættene **x** and **z** er henholdsvis en *række-vektor* og en *søjle-vektor*. To talsæt er af samme type, hvis 1° de har lige mange elementer, og 2° de begge er enten søjle-vektorer eller række-vektorer.

Et talsæt med kun ét element kaldes en *skalar*. Skalaren $\mathtt{x}_k$, det k-te element i talsættet **x**, fås som `x(k)`. Eksempel:

```
>> v = [z(3), x]                                        (2.2)
```

giver

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

Ordren (2.2) illustrerer også en måde til at udvide talsæt. Mere generelt kan man i stedet for `z(3)` skrive en række-vektor. Se også (2.3) nedenfor.

I princippet er alle MATLAB-variable *matricer*. Således siges

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

at være en matrix med 2 rækker og 3 søjler. Videre er elementerne i princippet komplekse tal. De prædefinerede variable `i` og `j` indeholder begge værdien af den imaginære enhed $\sqrt{-1}$. Hvis alle elementer i en variabel er reele, "glemmes" imaginær-delene i udskriften.

Bemærk forskellen mellem udskriften af **z** og **u** ovenfor: **z** indeholder lutter heltal, hvilket fremgår af udskriften. Ideelt skulle også **u** være heltallig ($\mathtt{u}_k = \sin\frac{k\pi}{2}$), men talrepræsentation og beregningsnøjagtighed er begrænset; jvf. Nielsen (1996). I MATLAB er *maskinnøjagtigheden* givet ved $\epsilon_M = 2^{-52} \approx 2.22{\cdot}10^{-16}$; denne værdi er indlagt i den prædefinerede variabel **eps**.

Udskriftsformatet kan ændres ved

Ordre	Virkning
`format compact`	blanke linier undertrykkes
`format loose`	blanke linier genindsættes
`format short`	5 cifre skaleret fast komma
`format long`	15 cifre skaleret fast komma
`format short e`	5 cifre eksponentiel notation
`format long e`	15 cifre eksponentiel notation

Eksempel: Ovenstående resultater er udskevet med **format compact** og **format short**. Ordrerne

```
>> format short e,  y
```

giver

```
y =
   1.0000e+00  1.2246e-16  -1.0000e+00  -2.4493e-16
```

I tabellen på næste side vises de operatorer, der er til rådighed. Når intet andet er fremhævet, forudsættes det, at **x** og **y** er talsæt af samme type, og at de har n elementer.

Eksempel: Vi vil undersøge om 1-tallerne i **y** er eksakte. Dette kan gøres ved

```
>> y([1 3]) - [1 -1]                                    (2.3)
```

som giver

```
ans =
   0   0
```

Operator	Typisk kald	Virkning
`+`	`z = x+y`	$z_k = x_k + y_k$, k=1, ..., n
`-`	`z = x-y`	$z_k = x_k - y_k$, k=1, ..., n
`.*`	`z = x.*y`	$z_k = x_k \cdot y_k$, k=1, ..., n
`./`	`z = x./y`	$z_k = x_k / y_k$, k=1, ..., n
`.^`	`z = x.^y`	$z_k = (x_k)^{y_k}$, k=1, ..., n. Specielt kan `x` eller `y` være en skalar og den anden operand et talsæt.
`'`	`z = x'`	Hvis `x` er en række-vektor, er `z` en søjle-vektor og omvendt. $\mathtt{z}_k = \overline{\mathtt{x}}_k$ (komplekst konjugeret).
`.'`	`z = x.'`	Som `x'` på nær, at $\mathtt{z}_k = \mathtt{x}_k$.

1-tallerne er således ("tilfældigvis") eksakte. Bemærk, at man kan udtage en delmængde af et talsæt `y` ved at skrive `y(v)`, hvor `v` er et talsæt, hvis elementer har værdier mellem 1 og n (længden af `y`).

Brug af f.eks. `*` og `.*` kan illustreres ved `x = [1 2]` og `y = [3 4]`, idet `a = x*y'` giver skalaren `a = 11`, og `b = x.*y` giver vektoren `b = 3  8` .

Apostrof `'` bruges også til at omslutte *strenge*. Således vil

```
>> s = 'Vel'
```

resultere i

```
s =
   Vel
```

hvorefter (jvf. konstruktionen (2.2))

```
>> t = 'til IMM'; s=[s, 'kommen ', t]
```

giver

```
s =
   Velkommen til IMM
```

Skulle der specielt være behov for en apostrof i sætningen, gøres dette ved en "dobbelt apostrof":


```
>> s = [s,'''s spændende note om MATLAB']
```

resulterer i

```
s =
   Velkommen til IMM's spændende note om MATLAB
```

Små bemærkninger:

- Hvis `x` er et talsæt, vil `x(:)` aflevere en søjle-vektor (med de samme elementer).
- En variabel `x` gøres tom ved ordren `x = [ ]` og den fjernes ved `clear x`. Ordren `clear` (uden argument) sletter alle variable på nær de prædefinerede.
- Ordren `who` giver en liste af de oprettede variable. Ved ordren `whos` får man yderligere deres størrelse, dvs. antallet af rækker og søjler i de matricer, hvori de er lagret.
- Et MATLAB program kan standes ved at taste **c**, mens control holdes nede.

MATLAB forlades ved ordren `quit` (eller `exit`).

2.2. Funktioner I

MATLAB har en lang række indbyggede funktioner og nem mulighed for selv at definere nye funktioner (se afsnit 2.4). Det typiske funktionskald er

[variabel_liste] = funktion(argument_liste)

hvor *funktion* er navnet på den aktuelle funktion. *argument_liste* kan være tom (i dette tilfælde udelades parenteserne), eller der kan være et eller flere argumenter, adskilt af "," (komma). *variabel_liste* har to specialtilfælde (se evt. 2.1): tom (hvor "`[ ]`" skal undlades) eller én variabel (hvor "`[ ]`" kan undlades). I appendiks A gives en liste over en række af de indbyggede funktioner. Man får information om funktionen *fun* ved ordren

```
>> help fun
```

Nedenfor anføres et mindre udpluk af funktionerne. Symbolet # markerer, at hvis argumentet er et talsæt, så vil resultatet være et talsæt af samme type

Navn	Typisk kald	Resultat
`sin`	`sin(x)`	sinus. #
`cos`	`cos(x)`	cosinus. #
`tan`	`tan(x)`	tangens. #
`atan`	`atan(x)`	arctangens. #
`atan2`	`atan2(y,x)`	Vinkel mellem 1. akse og linen fra (0,0) til (x,y).
`abs`	`abs(x)`	Absolut (numerisk) værdi. #
`sqrt`	`sqrt(x)`	Kvadratrod. #
`exp`	`exp(x)`	Exponentialfunktion, e^x. #
`log`	`log(x)`	Naturlig logaritme (grundtal e). #
`log10`	`log10(x)`	Logaritme med grundtal 10. #
`round`	`round(x)`	Afrund til nærmeste heltal. #
`fix`	`fix(x)`	Afrund mod nul (svarende til *trunkering* i Nielsen (1992)). #
`rem`	`rem(x,y)`	Resten ved division af `x` med `y`; dvs. rem(x,y) = x - y·fix(x/y) .
`sign`	`sign(x)`	Fortegn: sign(x) $= -1,\ 0,\ 1$ når hhv. $x < 0,\ x = 0,\ x > 0$. #
`size`	`size(x)`	Række-vektor $[m, n]$, hvor m og n er antallet af hhv. rækker og søjler i `x`. Kan kaldes ved `[m,n] = size(x)`.
`length`	`length(x)`	Længden af x: det største element i `size(x)`.
`sum`	`sum(x)`	$x_1 + \ldots + x_n$, hvor n er længden af `x`.
`prod`	`prod(x)`	$x_1 \cdot \cdots \cdot x_n$, hvor n er længden af `x`.

2.3. Betingede ordrer og løkker

Ved betingede ordrer er den typiske konstruktion

if betingelse1
  ordrer
elseif betingelse2
  ordrer
  ⋮
else
  ordrer
end

I betingelserne kan der indgå *relations-operatorerne*

MATLAB:	`<`	`<=`	`==`	`>`	`>=`	`~=`
"normal":	$<$	$\leq$	$=$	$>$	$\geq$	$\neq$

og de *logiske operatorer*

MATLAB:	`&`	|	$\sim$
"normal":	logisk og	logisk eller	negation

Eksempel: Antag at `x` og `y` er talsæt af samme type. Ordren

```
>> L=(x>0) & (x==y)
```

giver et talsæt `L` af samme type og med værdierne $\mathtt{L}_k$=1, hvis både $\mathtt{x}_k > 0$ og $\mathtt{x}_k = \mathtt{y}_k$, ellers $\mathtt{L}_k = 0$. (Dvs. værdierne *sand* og *falsk* repræsenteres ved hhv. 1 og 0).

Antag nu, at `x` og `y` er skalarer. Følgende programstump udfører det samme som ordren `t = tan2(y,x)`. Bemærk, at der her er tale om betingelser i flere niveauer. Hvert niveau skal afsluttes med `end`.

```
if y==0
   if x<0, t=pi
   else, t=0, end
else
   if x>0, t=atan(y/x)
   elseif x==0, t=sign(y)*pi/2
   else, t=atan(y/x) + sign(y)*pi, end
end
```

Et sæt af ordrer kan gentages ved *løkke*-konstruktionen:

for k = udtryk
   ordrer
end

hvor *udtryk* f.eks. har formen:

$$k0 : dk : k1 \tag{2.3}$$

Ordrerne udføres da successivt for $k = k0, k = k0+dk, \ldots, k = k1$. Ofte er $dk = 1$, og *udtryk* kan da reduceres til `k0:k1`.

Eksempel: Med `x` fra afsnit 2.1 (`x=[1 2 3 4]`) kan `y` beregnes ved

```
>> for i=1:length(x),  y(i) = sin(.5*pi*x(i)); end,  y
```

Semikolon før `end` er anbragt for at undertrykke udskrifter under udførelse af løkken. Man bør dog **ikke** benytte `for`-løkker til sådanne formål, idet kald af `sin` med et talsæt som argument giver det samme resultat langt hurtigere.[1)]

Konstruktionen i (2.3) kan også benyttes til at generere vektorer. Således fremkommer `x` fra afsnit 2.1 ved ordren `x = 1:4`. Talsættet `w` bestående af `x`-elementerne i omvendt rækkefølge fremkommer ved `w = x(4:-1:1)`.

Ved konstruktionen

while betingelse
   ordrer
end

1) Generelt kan det siges om MATLAB, at man får de simpleste udtryk og korteste beregningstider, hvis man vænner sig til at tænke i vektorer og matricer.

gentages *ordrer* sålænge *betingelse* er sand.

Eksempel: Hvad er det mindste naturlige tal n, for hvilket $n!$ har mindst 100 cifre? Svaret kan findes ved følgende MATLAB order

```
>> n = 1;  while prod(1:n) < 1e100, n = n+1; end,  n
```

2.4. Brug af filer og egne funktioner

MATLAB ordren

```
>> diary dagbog
```

bevirker, at der oprettes en fil `dagbog`, hvori alle inddata fra tastaturet og (det meste af) uddata kopieres. Man kan da senere skrive filen ud (evt. efter en redigering). Under MATLAB sessionen vil `diary off` afbryde kopieringen og `diary on` genoptager den.

Et vigtigt hjælpemiddel i MATLAB er de såkaldte *M-filer*, som dels bruges til at lagre (større) MATLAB programmer (såkaldte *script files*), dels til at konstruere egne funktioner (såkaldte *function files*). I begge tilfælde skal filen have ".m" som den sidste del af navnet.

En *script file* indeholder som nævnt blot en sekvens af MATLAB ordrer. Eksempel: Lad filen `middel.m` indeholde

```
% Program til beregning af middelværdi
n = length(x);
mid = sum(x)/n;
```

Ordren `middel` bevirker, at ordrerne i filen `middel.m` udføres. Tegnet "%" benyttes foran kommentarer – dvs. den resterende del af linien ignoreres.

I forbindelse med en *script file* er alle variable *globale*. Det betyder, at man i filen kan referere til variable, som er (eller skal være) oprettet før kaldet, og efter kaldet er eventuelt nye variable (som `n` og `mid`) kendt i resten af MATLAB sessionen (med mindre de fjernes med ordren `clear`).

En *function file* er karakteriseret ved, at første *kommando* linie begynder med ordet `function`.

Eksempel: Hvis vi ændrer filen middel.m til at indeholde

```
% Funktion til beregning af middelværdi
function mid = middel(x)
n = length(x);
mid = sum(x)/n;
```

er den nu en *function file*, og middelværdien af tallene $1,\cdots,99$ kan f.eks. fås ved ordren

```
>> y = middel(1:99)
```
(2.4)

I forbindelse med en *function file* skelnes der skarpt mellem *globale* og *lokale* variable. Alle variable, som indgår i beregningerne, skal enten overføres via *argument_listen* eller fremkomme som mellemresultater (hhv. `x` og `n` i `middel`). Lokale variable (som `n`) er ukendte efter kaldet.

Lad `t` være en streng. Ordren

```
>> eval(t)
```

bevirker, at teksten i `t` tolkes som et sæt af MATLAB ordrer og de tilsvarende beregninger udføres. Således er `y = eval('middel(1:99)')` ækvivalent med (2.4). Denne såkaldte *macro facilitet* åbner mulighed for, at man kan have en funktion som argument til en funktion.

De følgende to funktioner har samme specielle syntaks som `clear` i afsnit 2.1: Ordren

```
>> save gem x y z
```

bevirker, at `x`, `y` og `z` gemmes i filen gem.mat. Ved ordren

```
>> load gem
```

genindlæses de. Ved ordren `save gem` lagres alle de aktuelle variable i gem.mat. Denne facilitet har man bl.a. glæde af, når man kan forudse, at man i en senere MATLAB session skal arbejde videre med de opnåede resultater.

Skriver man blot `save`, lagres alle de aktuelle variable i filen matlab.mat.

2.5. Grafik

Funktionen `plot` kan benyttes til grafisk uddata. Lad `t` og `y` være talsæt med lige mange elementer (m). Ordren

```
>> plot(t,y)
```

genererer en graf, hvor punkterne $(\mathtt{t}_k, \mathtt{y}_k)$, $k = 1, \ldots, m$, er forbundet med rette liniestykker. Grafen indlægges i et koordinatsystem med passende skalering af akserne. Kald af `plot` kan varieres på mange måder (prøv `help plot`). Vi nøjes med at betragte et eksempel:

Lad

$$y(x) = \frac{1}{(x-0.3)^2+0.02} + \frac{1}{(x-0.9)^2+0.03} - 20 \ .$$

Vi ønsker en graf af denne funktion for $0 \le x \le 1$, og vi ønsker også at se tangenten til kurven gennem punktet $(x_0, y_0) = \big(0.45, y(0.45)\big)$; dvs. linien med ligningen

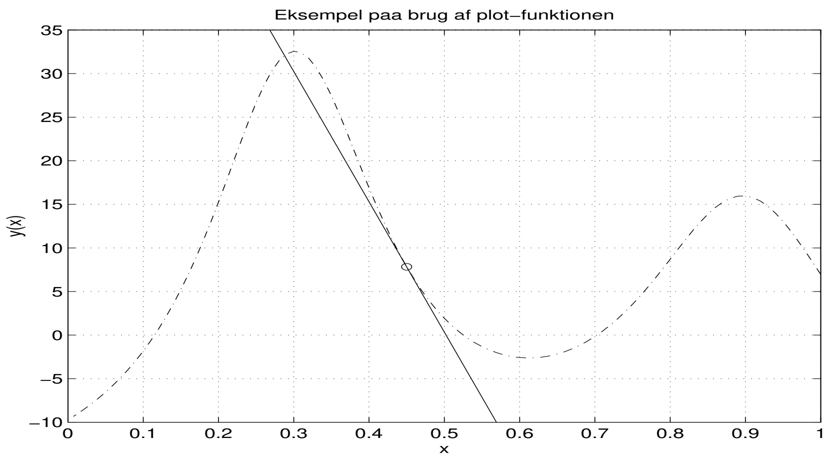
$$z(x) = y_0 + ym{\cdot}(x - x_0) \ ,$$

hvor $ym = y'(x_0)$. Nedenfor vises et program, som giver dette.

```
n = 100;  x = [0:n]/n;
y = 1./((x-.3).^2 + .02) +1./((x-.9).^2 + .03) - 20;
plot(x,y,'-.'),  axis(axis),  hold on
x0 = .45;   N1 = (x0-.3)^2 + .02;  N2 = (x0-.9)^2 + .03;
y0 = 1/N1 + 1/N2 - 20;
ym = -2*((x0-.3)/N1^2 + (x0-.9)/N2^2);
yy = y0*[1 1] + ym*([0 1] - x0*[1 1]);
plot([0 1],yy,'-', x0,y0,'o'),  grid
title('Eksempel paa brug af plot-funktionen')
xlabel('x'),  ylabel('y(x)')
```

Argumenterne `'-.'`, `'-'` og `'o'` i de to kald af `plot` bevirker, at linierne tegnes hhv. stiplet og med fuld streg, og punktet (x_0, y_0) markeres med en bolle . Ved ordresættet `axis(axis), hold on` opnås, at efterfølgende kald af `plot` tegner i samme koordinatsystem. `grid` genererer et net af punkterede akseparallelle linier. `title`, `xlabel` og `ylabel` kan anvendes til at beskrive figur-indholdet.

Resultatet af ovenstående program kan ses nedenfor.



Når der er opnået et tilfredstillende *plot*, kan resultatet (dvs. plottet) lagres i en Postscript fil til senere brug i f.eks. LaTeX. Følgende ordre:

```
>> print mitplot
```

gemmer det aktuelle plot i filen mitplot.ps. Bemærk at .ps ikke er medtaget i ordren. MATLAB understøtter også andre grafik-formater. Prøv `help print`.

2.6. Matricer

En matrix som f.eks. $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ kan indlægges i MATLAB ved ordren

```
>> A = [1 2 3; 4 5 6]
```
(2.5)

Bemærk, at semikolen svarer til et lineskift, dvs. en ny række i matricen.

Antag, at **A** er oprettet, og vi ønsker at indlægge matricen $\mathbf{B} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$. Dette kan ske ved ordren

```
>> B = [A; 7 8 9]
```

mens

```
>> C = [A [7; 8]]
```

bevirker, at **A** udvides med en ekstra søjle til $\mathbf{C} = \begin{bmatrix} 1 & 2 & 3 & 7 \\ 4 & 5 & 6 & 8 \end{bmatrix}$.

Ordren

```
>> D = B(1:2,2:3)
```

udtager den øverste højre 2×2 delmatrix af **B**, $\mathbf{D} = \begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$, mens

```
>> E = A(:,[3 1 2])
```

ombytter søjlerne i **A**: $\mathbf{E} = \begin{bmatrix} 3 & 1 & 2 \\ 6 & 4 & 5 \end{bmatrix}$. Endelig vil ordren

```
>> A(3,2) = -1
```

udvide ovenstånde **A** til matricen $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 0 & -1 & 0 \end{bmatrix}$.

I forbindelse med opbygning af større matricer kan man udnytte

```
>> I = eye(n),  O = ones(m,n),  Z = zeros(m,n)
```

som giver henholdsvis $n \times n$ enhedsmatricen **I**, og $m \times n$ matricerne **O** og **Z** med henholdsvis lutter 0-er og lutter 1-taller. Videre skal vi nævne

```
>> D = diag(x)
```

hvor $\mathtt{x}$ er en $m \times n$ matrix. Hvis $\mathtt{x}$ er en række- eller søjle-matrix ($m = 1$ eller $n = 1$), er **D** en diagonal matrix med $\mathtt{d}_{ii} = \mathtt{x}_i$, $i = 1, \ldots,$ `length(x)`. Ellers er **D** en søjlematrix med $\mathtt{d}_i = \mathtt{x}_{ii}$, $i = 1, \ldots, \min(m, n)$.

Ligesom i afsnit 2.1 forudsætter de *binære operatorer:* `+`, `-`, `.*` og `./` at de to operander er af samme type. Effekten af $\mathbf{C} = \mathbf{A} \diamond \mathbf{B}$ svarer til elementvis udførelse af operationen $\diamond$. Ordren

```
>> C = A*B
```

svarer til matrixmultiplikationen $\mathbf{C} = \mathbf{AB}$ (og forudsætter, at antallet af søjler i $\mathbf{A}$ er lig med antallet af rækker i $\mathbf{B}$). Ordren

```
>> G = A.'
```

transponerer matricen ($\mathbf{G} = \mathbf{A}^T$). Hvis alle elementer i $\mathbf{A}$ er reelle, kan man nøjes med at skrive `G = A'`, jvf. side 11.

2.7. Løsning af lineære ligningssystemer

Lad $\mathbf{A}$ og $\mathbf{b}$ være hhv. en $m \times n$ og en $m \times q$ matrix, repræsenteret i MATLAB ved hhv. `A` og `b`. Ved ordren

```
>> x = A\b
```

fås $n \times q$ matricen $\mathbf{x}$, som repræsenterer "løsningen" til matrixligningen $\mathbf{Ax} = \mathbf{b}$. Ved "løsning" forstås:

`m = n = rank(A)` :	`x` repræsenterer den entydigt bestemte løsning til $\mathbf{Ax} = \mathbf{b}$.
`m > n, rank(A) = n` :	`x` repræsenterer den entydigt bestemte *mindste kvadraters løsning* til det *overbestemte system* $\mathbf{Ax} \approx \mathbf{b}$.
`rank(A) < n` :	`x` repræsenterer en såkaldt *"basis løsning"*, som svarer til, at frie variable i den fuldstændige løsning er sat til nul.

2.8. Funktioner II

De fleste af funktionerne fra tabellen i afsnit 2.2 kan umiddelbart kaldes med en matrix som argument. Hvis `f` er en af de funktioner, der er markeret med #, vil `B = f(A)` være af samme type som `A` og $\texttt{b}_{ij}$=`f(`$\texttt{a}_{ij}$`)`.

Hvis $\mathbf{A}$ repræsenter en $m \times n$ matrix, giver ordren `sum(A)` en række-vektor

med n elementer. Det j-te element er summen af elementerne i den j-te søjle af $\mathbf{A}$. Funktionerne: `prod`, `max`, `min` og `sort` opfører sig tilsvarende; prøv `help` på dem.

For en kvadratisk (`n`$\times$`n`) matrix $\mathbf{A}$ vil ordren

```
>> L = eig(A)
```

give en søjle-vektor med egenværdierne for $\mathbf{A}$: `L(j)` $= \lambda_j$, $j = 1, \ldots, n$. Ordren

```
>> [V, D] = eig(A)
```

giver en diagonalmatrix $\mathbf{D}$, som repræsenter $\mathbf{D} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, mens den j-te søjle i $\mathbf{V}$ er en egenvektor hørende til egenværdien λ_j: `A*V(:,j) = D(j,j)*V(:,j)`. Der gælder $\mathbf{AV} = \mathbf{VD}$.

3. Mere hjælp

Efter at have læst ovenstående introduktion, er man rimeligt udrustet til at kunne gå i gang med at bruge MATLAB. I tilfælde af problemer gælder der

Den hurtigste hjælp opnås i reglen via `help`.

Er denne ordre ikke tilstrækkelig, har vi følgende vink:

- Ved hjælp af ordren `path`, kan man få oplyst, hvad MATLAB "kan se". Man kan udvide MATLAB's synsfelt – prøv at skrive
  ```
  >> help path
  ```
- Hvis man ikke kan huske, hvad en bestemt ordre hedder, kan man prøve at anvende ordren `lookfor`. Har man glemt ordren `diary`, kunne man eksempelvis søge på ordet *save* – idet funktionen af `diary` er at *gemme* det interaktive samspil (se evt. afsnit 2.4). Eksempel:
  ```
  >> lookfor save
  ```
  søger efter ordet "save" i den første kommentarlinie i alle m-filer i de direktorier, som er defineret i `path`.
- Har man for eksempel glemt hvilket filnavn, man angav, i forbindelse med ordren `save` (se afsnit 2.4), kan ordren `what` anvendes. `what` lister alle filer i det aktuelle direktorie af typen: *.m, *.mat og *.mex.
- Har man glemt, hvor man har gemt en m-fil, kan ordren `which` muligvis hjælpe. Eksempel:
  ```
  >> which middel
  ```
  søger efter filen middel.m i de direktorier, som er defineret i `path`.

- Endnu engang sætter vi fokus på ordren `who`. `who` lister, hvilke variable MATLAB aktuelt har kendskab til (se afsnit 2.1).

I appendiks A findes et lille udsnit af MATLAB's standard funktioner. Til den mere erfarne/krævende MATLAB-bruger kan vi henvise til følgende litteratur:

- Haugen (1996) er en norsk bog om MATLAB. Findes på DTV.
- MathWorks (1995) er en amerikansk bog om MATLAB. Findes på DTV og i PF-boghandlen. Med bogen følger en diskette med en "studenter–udgave" af MATLAB. Pris ca. 480 kr.

Desuden henvises *web*-folket til:

- MATLAB's homepage: http://www.mathworks.com
- MATLAB in Education: http://education.mathworks.com
- The MathWorks MATLAB Digest, email news letter. Man kan abonnere ved at sende en email til: subscribe@mathworks.com. Der går i reglen noget tid inden man modtager noget.
- The MATLAB Primer. Introduktionsnote til MATLAB (på engelsk). Findes i GBAR'en på lærerdisken til kursus C0201 og C0204. På IMM-systemet findes den som
  /usr/local/matlab/doc/primer.ps
- MATLAB news group. I GBAR'en kan man via "tin" komme i kontakt med en MATLAB news group på adressen:
  comp.soft-sys.matlab
  Denne news group udgør et forum, hvor man kan diskutere, få og/eller yde hjælp angående MATLAB.

Litteratur

F. Haugen (1996): *Lær MATLAB.* Avdeling for Teknologiske fag, Høgskolen i Telemark.

MathWorks (1995): *The Student Edition of MATLAB®.* Prentice-Hall, Englewood Cliffs, NJ, USA.

H.B. Nielsen (1996): *Numerisk Lineær Algebra.* IMM, DTU, Lyngby.

Appendiks A. Udvalgte MATLAB funktioner

Følgende MATLAB funktioner er relevante i forbindelse med et indledende kursus i numerisk analyse. Oplysning om funktionen `fun` fås ved ordren `help fun`.

abs	all	any	acos	asin	atan	atan2
axis	break	chol	clear	cond	conj	cos
det	diff	disp	echo	error	eval	exp
fix	feval	find	finite	flops	fmin	fzero
grid	hold	imag	input	inv	isnan	isempty
length	linspace	load	log	log10	lu	max
mean	median	min	norm	null	num2str	ones
orth	pause	plot	poly	polyval	prod	qr
rand	rank	rcond	real	rem	return	roots
round	save	sin	sign	size	sort	sprintf
sqrt	sum	tan	title	trace	tril	triu
what	who	whos	xlabel	ylabel	zeros	

Appendiks B. MATLAB i IMM-kurser

Dette appendiks giver en oversigt over de enkelte IMM-kursers brug af MATLAB. For at lette notationen og bevare overblikket, anvendes følgende forkortelses koder:

Kode	Program-pakke
α	MATLAB
β	SIMULINK

Kode	TOOLBOX
A	Control System
B	ODE-suite
C	Optimization
D	Rudimentary statistics
E	Signal Processing
F	Splines
G	System Identification
H	Wavelets

I følgende tabel er de enkelte IMM-kursers brug af MATLAB listet.

Kursus	α	β	A	B	C	D	E	F	G	H
C0201	x									
C0204	x									
C0205	x									
C0211	x			x						
C0254	x							x		x
C0261	x				x					
C0310	o									
C0414	x		x			x			x	
C0417	o					o			o	
C0422	x		o			o			o	

I tabellen angiver: "o" at der er potentiel brug af MATLAB og "x" markerer direkte brug. Simulink er anvendt i forbindelse med nogle eksamensprojekter.