Introduktion til

MATLAB

anden udgave

Udarbejdet af
Johnny Ottesen & Thomas Frommelt

IMFUFA, RUC,
Juni 2000

Indhold

Forord

MATLAB er en programpakke til numeriske beregninger og visualiseringer. Navnet MATLAB står for *Matrix Laboratory*, idet programpakken oprindelig var udviklet til kun at understøtte arbejde med matricer. MATLAB er siden udvidet til også at omfatte andre områder. Kommunikationen mellem bruger og program foregår i et letanvendeligt sprog, hvor problemer og deres løsninger opskrives matematisk.

Dette notat er tilsigtet nybegyndere med MATLAB, der for første gang ønsker at anvende programmet. Ønsker man en mere avanceret manipulation men diverse filer og grafik henvises til MATLABs egne manualer. Notatet er fortrinsvis skrevet til brug af MATLAB under Microsoft Windows NT, men da MATLAB også kan køres under LINUX, er der enkelte steder, hvor der er specificeret hvordan kommunikationen i LINUX versionen foregår. Det forudsættes at brugeren er bekendt med Windows-verden (rullemenuer, knapper og betjening af mus).

Denne anden udgave er en revideret version af *Introduktion til* MATLAB - *første udgave* af BETTINA SØRENSEN og JOHNNY OTTENSEN, IFUFA, RUC, December 1994

Ris og ros til dette notat kan gives på email-adresserne: frommelt@dirac.ruc.dk og johnny@ruc.dk.

Kapitel 1

Opstart af MATLAB

Det er en god ide at læse denne første del af notatet inden man går igang med MATLAB. Her vil blive præsenteret det helt grundlæggende i at starte og stoppe programmet, samt under hvilke forhold kommunikationen mellem bruger og MATLAB foregår.

1.1 Opstart

På IMFUFA er MATLAB installeret så det både kan køre under Microsoft Windows NT og under LINUX. Programmet opstartes fra Windows ved at dobbeltklikke på programmets ikon, som er placeret på skrivebordet, mens det under LINUX kun kan køres fra Dirac.

Den del af computerens hukommelse, der af MATLAB bruges til at huske de variable som oprettes, kaldes Workspace. Dette lager er kun aktivt, mens MATLAB programmet er åbent. Når MATLAB lukkes ned efter brug, vil der ikke automatisk blive gemt oplysninger om de aktuelle variable. Der er dog mulighed for at de variable kan blive lagret, hvis det ønskes. Se afsnit 1.4 side 3. Det er vigtigt at man befinder sig det rigtige sted i katalogtræet når man arbejder i MATLAB, så eksempelvis de gemte variable bliver gemt det rigtige sted. Ved at skrive `pwd` efterfulgt af RETURN retuneres der, hvilket katalog man befinder sig i. Man skifter rundt mellem katalogerne med den normale DOS kommando `cd`.

1.2 Kommandolinie

Kommunikationen mellem bruger og program forgår via kommandolinien i MATLAB Command-vinduet (eller via M-filer; se kapitel 3 side 33). Herfra oprettes variable og herfra kaldes de indbyggede hjælpefunktioner til behandling af de variable.

Eksempel

Det ønskes at oprette en 3×3 matrix A bestående af elementerne 9 til 1. På kommandolinien skrives :

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

Herefter tastes RETURN og MATLAB vil nu behandle udtrykket. Behandlingen resulterer i outputet:

```
A =
     9     8     7
     6     5     4
     3     2     1
```

Workspace husker nu A som en variabel, der kan bruges til videre beregninger. Hvis der nu skrives A og tastes RETURN vil matricen A igen blive skrevet ud.

□

MATLAB skelner mellem store og små bogstaver. De indbyggede funktioner skrives (med enkelte undtagelser) alle med små bogstaver, men både store og små bogstaver kan anvendes i variabelnavne. Det anbefales som hjælp for brugeren selv, at bruge små bogstaver til vektorer og skalar og store bogstaver til matricer. Dette er dog ikke en nødvendighed og brugeren er frit stillet til at gøre, hvad hun finder bedst.

Ligesom de variable huskes af Workspace, huskes også alle de kommandoer, der er tastet ind på kommandolinien. Det er muligt at bladre mellem disse tidligere kommandoer ved hjælp af piletasterne. På den måde kan kommandoer genbruges og brugeren behøver ikke taste hele kommandoen igen. Når MATLAB programmet lukkes ned glemmes kommandoerne igen.

1.3 Hjælpefaciliteter

Der er flere muligheder for at søge hjælp i programmet. Dels via det indbyggede menupunkt **Help** og dels direkte via kommandolinien.

Hjælpemenu

Hjælpeprogrammet tilbyder flere indgange til at søge hjælp. **Table of contents** giver en oversigt over de områder indenfor MATLAB, hvortil der ydes hjælp. **Index** viser en alfabetisk liste over alle de funktioner der er indbygget i MATLAB, hvortil der er hjælp og eksempler. **Help Selected** søger efter hjælp til det ord brugeren har fremhævet med musen i vinduet. **Examples and demos** giver adgang til en række step by step demonstationer af hvordan MATLAB fungerer

Kommandolinie

Der er to kommandoer til at søge efter hjælp med. En specifik kommando `help`[1], der kan anvendes til at søge hjælp på de indbyggede funktioner. Derimod kan `lookfor` med fordel anvendes, hvis navnet på en given indbygget funktion ikke kendes, men giver istedet mulighed for at søge på ord fra alle hjælpeteksterne. F.eks. giver `help` negativt svar ved søgning på `minimum`, da dette ikke er navnet på en funktion. Søgninger som `help min` eller `help fmin` vil give oplysninger om de to forskellige funktioner. Derimod vil en søgning som `lookfor minimum` efter længere søgetid resultere i foreslag om at søge hjælp på funktionerne `COLMMD` og `SYMMMD` m.fl.

Med kommandoen `demo` startes et demonstrationsprogram, som giver en række eksempler på hvordan MATLAB fungerer. Endelig findes en HTML-baseret version af dokumentationen, som kaldes frem med kommandoen `doc`, eller med kommandoen `helpdesk`

1.4 Variable

Det er muligt at liste de variable, der er aktuelle i Workspace ved at skrive `who` på kommandolinien. Dette vil give navne på de variable som Workspace kender. En udvidet oversigt over de variable fås ved at skrive `whos`, hvor efter der for hver variabel listes oplysninger om navn, størelse, antal bytes og klasse. Det fremgår af denne oversigt, at alle variable opfattes som matricer. Skalar variable er blot 1×1 matricer.

Eksempel

Udover matricen A oprettes variablene z og x. Dette gøres ved at taste `z = 4 + 5i` og `x = [1;3;5;4;2]`. Når der herefter skrives:

```
who
```

fås svaret:

```
your variables are:
A        x        z
```

Hvis der nu istedet skrives

```
whos
```

[1] Eller kommandoen `helpwin` der giver adgang til en videresøgning på internettet

fås svaret:

```
Name         Size      Bytes     Class

 A           3 x 3      72       double array
 x           5 x 1      40       double array
 z           1 x 1      16       double array (complex)

 Grand total is 15 elements using 128 bytes
```

□

Ved ingen af disse oversigter ses de variables aktuelle værdier. Disse fås ved blot at taste variabelnavnene på kommandolinien og taste RETURN.

Eksempel

Værdien af variablen x, kan kaldes frem ved at taste:

```
x
```

hvorved der fremkaldes:

```
x =
    1
    3
    5
    4
    2
```

□

Inden man forlader MATLAB er det muligt at gemme de variable man har oprettet, så man kan arbejde videre med disse størrelser ved en senere lejlighed. Dette gøre ved at skrive `save` (efterfulgt af et RETURN som altid). MATLAB gemmer nu alle variablene i en defaultfil ved navn `matlab.mat` og fra denne fil kan de variable igen hentes ind i Workspace. Variablene hentes ved at taste `load` på kommandolinien. De to kommandoer `save` og `load` kan også anvendes til andre filer.

Eksempel

Det er kun variablene A og x som ønskes gemt til næste gang. Disse ønskes gemt i en anden fil end MATLABs defaultfil. Derfor skrives

```
save minfil A x
```

MATLAB opretter nu en fil ved navn `minfil.mat` i brugerens hjemmekatalog, hvor variablene A og x lagres. Husk at adskille variable med mellemrumstegn. MATLAB giver filen det specielle efternavn (`.mat`), så programmet kan se, næste gang når variablene ønskes hentet ind i Workspace, at denne fil er oprettet af MATLAB. Variablene hentes nu ved at skrive:

```
load minfil
```

□

Man forlader MATLAB ved at vælge **Exit** MATLAB under menupunktet **File**, eller ved at lukke MATLAB vinduet (ved at klikke på øverste højre hjørne). Bemærk, at hvis der er blevet oprettet filer ved hjælp af en editor, vil disse ikke lukke ned, da editoren ikke styres af MATLAB, men er et uafhængigt program der anvendes. Filerne skal lukkes og gemmes særskilt.

1.5 Sprogbrug i dette notat

En variabel betegner en størrelse med et navn og en værdi f.eks.:

```
B =
     1     0     1
     2     1     0
     3     1     3
```

Der som nævnt i afsnit 1.4 oprettes ved at taste `B=[1,0,1;2,1,0;3,1,3]` på kommandolinien.

En matematisk funktion f.eks. $f(x) = \sqrt{3x^2 + x}$ eller blot en funktion, skal først repræsenteres som en funktionsfil (Se afsnit 3.2 side 35). Derefter kan funktionen behandles, f.eks. ved at finde ekstrema for funktionen. En diffenretialligningsfunktion er special typer af sådanne funktioner.

Hjælpefunktioner er operationer, der behandler de matematisk størrelser, det vil sige variable eller funktioner. F.eks. er `inv` en hjælpefunktion, der finder den inverse (hvis det er muligt) af en kvadratiske matrix.

På kommandolinien skrives kommandoer. Det er udtryk der indeholder størrelser (variable eller funktioner) og en specifikation af hvilke hjælpefunktioner, der skal behandle størrelserne. F.eks. `C = inv(B)` er en kommando, der fortæller at der skal beregnes den inverse til matricen B og resultatet skal lægges i en variabel C. Ved tast af RETURN vil kommandoen blive udført af MATLAB.

En standardfunktion er en matematisk funktion, som allerede er kendt af MATLAB. Alle andre funktioner skal oprettes som funktionsfil af brugeren selv. Et eksempel er `pi`, der er standardfunktion for π og et andet eksempel er `log`, der er standardfunktion for den naturlige logaritme.

Kapitel 2

Basal brug af MATLAB

Her vil blive gennemgået og vist eksempler på den mest basale brug af MATLAB. Denne del af notatet er dog tænkt som en opslagsdel, hvor det ikke er nødvendigt at læse alle afsnit.

2.1 Matrixoperationer

I det indledende afsnit blev det vist, hvorledes en matrix oprettes via kommandolinien. Der er flere andre metoder til at indføre matricer i Workspace på. Ved hjælp af de indbyggede hjælpefunktioner er det muligt at oprette specielle typer af matricer. Derudover kan matricer indføres via M-filer eller externe datafiler. Disse muligeheder vil blive uddybet senere i kapitel 3. Den simpleste metode er at taste matricerne ind på kommandolinien. Matricen indtastes rækkevis og:

- de enkelte elementer adskilles af blanktegn eller komma
- elementerne omklammes med firkantede parenteser, `[ ]`
- afslutningen af en række (alle pånær den sidste) markeres med et semikolon `;`

Notationer mv. specielt til vektorer gives ikke, idet vektorer opfattes som specialtilfælde af matricer.

Transponering

Den matrix, der ønskes transponeret markeres med en apostrof `'`.

Eksempel

Matricen A er igen defineret som `A = [9 8 7;6 5 4;3 2 1]`. Der ønskes en matrix B, der skal være den transponerede af matricen A. Der skrives:

```
B = A'
```

Herved fremkommer:

```
B =
     9     6     3
     8     5     2
     7     4     1
```

Hvis det ligeledes ønskes at transponere søjle vektoren `x=[1; 3; 5; 4; 2]`, skrives

```
p = x'
```

og svaret bliver

```
p =
     1     3     5     4     2
```

□

Addition og subtraktion

Som for skalar anvendes symbolerne `+` og `-` for henholdsvis addition og subtraktion. Det er vigtigt, at de indgående størrelser har samme dimension, for at operationen kan finde sted. Når `A` og `x` er defineret som i ovenstående eksempel vil udtrykket `A + x` således ikke give nogen mening og vil resultere i en fejlmeddelelse. Denne vil typisk fortælle at operationen ikke kan udføres, fordi de indgående størrelser ikke har samme dimension.

Eksempel

Der oprettes en ny matrix C, der er summen af matricerne A og B.

```
C = A + B
```

og svaret bliver

```
C =
    18   14   10
    14   10    6
    10    6    2
```

□

Multiplikation

Som ved skalarmultiplikation bruges stjernesymbolet `*` ved matrixmultiplikation. Den indre dimension af de to faktorer skal være ens for at operationen giver mening. For matricer gælder det specielt, at første faktors 2. dimension skal svare til anden faktors 1. dimension. Der er både defineret indre og ydre produkter.

Eksempel

Givet vektorene y og z, hvor `y = [1 2 3]` og `z = [-1 0 1]`. Det indre produkt beregnes ved at skrive:

```
y*z'
```

Der resultere i svaret:

```
ans =
      2
```

Tilsvarende kan `z*y'` beregnes. Forkortelsen "ans" står for answer og er standardsvaret, hvor resultatet ikke er specificeret til et bestemt variabelnavn. `ans` indgår herefter som en variabel. Derimod er

```
y'*z
```

et ydre produkt og vil resultere i svaret:

```
ans =
     -1    0    1
     -2    0    2
     -3    0    3
```

□

Produkter mellem matricer og vektorer, kræver ligeledes overenstemmelser i dimensionerne.

Eksempel

Her benyttes matrix A og vektor z fra tidligere eksempler.

```
b = A*z'
```

resulterer i svaret:

```
b =
    -2
    -2
    -2
```

Det er vigtigt, at det er den transponerede af z, som anvendes. `A*z` opfylder ikke kravet til dimensionerne og vil resultere i en fejlmeddelelse.

□

Division

Der er to symboler for division `/` og `\`. Det er udelukkende i de tilfælde, hvor A er en ikke-singulær n $\times$ n matrix (det(A) $\neq$ 0), at disse symboler korresponderer med henholdsvis højre og venstre multiplikation. Det vil sige at

- `X = A\B` er løsning til `A*X = B`
- `X = B/A` er løsning til `X*A = B`

Inden division med matricer er det således relevant at undersøge matricens determinant. Er A en matrix, skrives `det(A)` på kommandolinien og determinanten beregnes.

Singulære eller næsten singulære matricer er et generelt problem ved numerisk behandling af matricer. Det kan således være hensigtmæssigt at beregne en matrixs konditionstal, der er et udtryk for, hvor "singulær" en matrix er. MATLAB foretager selv den slags undersøgelser af matricer inden udførelsen af visse funktioner. Konditionstallet af en matrix A beregnes ved at skrive `cond(A)`. Svaret er et meget stort tal (med en stor 10-potens), hvis matricen er singulær og meget mindre hvis den ikke er singulær. Funktionen `rcond` beregner den reciprokke værdi af konditionstallet. A er langt fra at være singulær hvis svaret er nær ved 1.0, mens A er næsten singulær hvis svaret er tæt ved 0.0 (nul).

Eksempel

Først et eksempel på en ikke-singulær matrix F og en søjlevektor g. Lad `F = [5 0 1; -2 4 0; 3 2 1]` og `g = [2 ;-1 ;1]`. Først udberegnes produktet

```
h = F*g
```

hvilket resulterer i svaret:

```
h =
    11
    -8
     5
```

Dette kan testes ved:

```
F\h
```

Der gives svaret:

```
ans =
      2.0000
     -1.0000
      1.0000
```

□

Eksempel

MATLAB kan dog give overraskende løsninger. I tilfældet, hvor `A = [9 8 7, 6 5 4; 3 2 1]` er determinanten af A nul. Tidligere blev `b` sat lig med `A*z'`, hvor `z = [ -1 0 1]`. For at efterprøve dette skrives:

```
w = A\b
```

Istedet for at give en advarsel omkring A's singularitet, beregnes følgende svar, som ikke svarer til det `z'` man måske forventer:

```
w =
    0.4998
   -2.9996
    2.4998
```

I dette tilfælde er løsningsrummet én-dimensionalt, så der findes uendelig mange løsninger. MATLAB finder blot en af disse. Havde man istedet spurgt efter `b'/A` ville man derimod få advarslen:

```
Warning: Matrix is singular to working precision.
```

□

Anvendelse af potens

Symbolet ^ anvendes til at angive potenser. MATLAB tillader udtryk af typen `A ^ p`, hvor A er en matrix og p en skalar. Dog er der også indbygget muligheder for tilsvarende udtryk, hvor A er en skalar og p en matrix. Hvor både A og p er matricer kan operationen ikke finde sted og der gives en fejlmeddelelse.

Eksempel

Matricen `A = [9 8 7;6 5 4;3 2 1]` opløftes til 2. potens og benævnes B.

```
B = A^2
```

som giver

```
B =
    150    126     102
    96     81      66
    42     36      30
```

□

2.2 Tabeloperationer

Tabeloperationer dækker en række yderligere operationer, hvor der opereres enkeltvis på elementerne. En matrix eller vektor, hvorpå der opereres elementvis kaldes her for en tabel. For regneoperationerne addition og subtraktion er der ingen ændring i syntaksen, men for de resterende operationer tilføjes et punktum til operationssymbolet for at indikere, at der er tale om en tabeloperation og ikke en matrixoperation.

Multiplikation og division

Symbolet `.*` anvendes til elementvis multiplikation. Det er stadigvæk en forudsætning at tabellerne, der indgår som faktorer har ens dimension. Dette skal forståes på den måde at de to faktorer f.eks. begge skal være 1×3 vektorer for at operationen lykkes. Igen findes der to udgaver af division. Hvor `X` og `Y` er to tabeller findes `X./Y` og `X.\Y`

Eksempel

Fra tidligere eksempler kendes y og z, hvor y = [1 2 3] og z = [-1 0 1]. Multiplikation af element for element er givet ved:

```
w = y.*z
```

og der præsenteres svaret:

```
w =
    -1    0    3
```

□

Eksempel

Tilsvarende gælder for division. Med de y og z variable, der er givet tidligere vil kun den ene divisionoperation give mening:

```
v = y.\z
```

Hvilket resulterer i:

```
v =
    -1.0000       0       0.3333
```

□

Anvendelse af potenser for tabeller

Elementvis potensbehandling angives ved `.^`

Eksempel

Her kommer en række kombinationer, hvor der kan anvendes potenser. Variablen y givet ved `y =[1 2 3]` som i forrige eksempel og variablen `x = [3 1 2]` tilføjes.

```
w  = y.^x
```

Herved fås svaret

```
w  =
     1   2   9
```

Der kan ligeledes indgå en skalar:

```
w  = y.^3
```

som giver svaret

```
w  =
     1   8   27
```

Endelig den omvendte anvendelse af skalar

```
w  = 3.^x
```

Hvilket resulterer i svaret

```
w  =
     27   3   9
```

□

Relationsoperationer

For skalarer og matricer giver MATLAB mulighed for 6 relationsoperationer. Disse operationer giver mening under forudsætning, at matricerne er af samme dimension. En skalar kan dog sammenlignes med en matrix, da vil skalaren blot blive sammenlignet med hvert element i matricen. Bemærk at et enkelt stående lighedstegn allerede anvendes som en tildelingsoperator (f.eks. i forrige eksempel tildeles variablen `w` bl.a. resultatet af operationen `y.^ 3`), derfor benyttes det dobbelte lighedstegn som sammenligning operatoren. MATLAB sammenligner matricerne elementvis og resultatet er en matrix, hvis elementer udgøres af 0'er og 1'er, hvor 0 repræsentere 'falsk' og 1 'sandt'.

$<$ mindre end

$<=$ mindre end eller lig med

$>$ større end

$>=$ større end eller lig med

$==$ lig med

$\sim=$ forskellig fra

Eksempel

Lad tabellen y være defineret ved `y = [5 2 1]` og da `pi` (den indbygget standardfunktion for π) er en skalar, kan hvert element i y sammenlignes med denne. En kommando af form:

```
y <= pi
```

vil resultere i svaret:

```
ans =
      0    1    1
```

□

Matematiske funktioner

MATLAB inkluderer en del matematiske funktioner, heriblandt diverse trigonometriske funktioner. Hvor funktionen anvendes på tabeller, sker dette elementvis.

- Trigonometriske funktioner. Her præsenteres et udvalg
  - `sin` står for sinusfunktionen. Tilsvarende for `cos` og `tan`.
  - `asin` er arcsinus. Ligeledes gives der `acos` og `atan`.
  - `sinh` står for funktionen sinus hyberbolsk. Igen findes der varianterne `cosh` og `tanh`
- Udvalg af elementære funktioner
  - `abs` Den absolutte værdi.
  - `sqrt` Kvadratrod.
  - `exp` Eksponential funktionen med base e.
  - `log` Den naturlige logoritme.
  - `log10` Logoritme med base 10.

Eksempel

Variablen z er stadigvæk defineret som `z = [-1 0 1]`. Ved anvendelse af en trigonometrisk funktion på z vil elementerne blive behandlet enkeltvis. Kommandoen

```
sin(z)
```

vil resultere i

```
ans  =
       -0.8415    0    0.8415
```

□

Eksempel

Ganske tilsvarende vil det gøre sig gældende for de elementære funktioner. Her bliver vektoren z givet i eksemplet ovenfor, igen anvendt.

```
sqrt(z)
```

giver

```
ans =
      0 + 1.0000i     0    1.0000
```

Bemærk, at MATLAB, ikke giver en fejlmeddelelse for at skulle tage kvadratroden af et negativt tal. I stedet indenføres komplekse tal i løsningen. Dette må brugeren selv være opmærksom på, hvis de komplekse løsninger ikke ønskes.

□

2.3 Yderligere matrixoperationer

Tidligere er det vist, hvorledes det er muligt at oprette vektorer. MATLAB giver mulighed for nemt at oprette visse vektorer uden at taste hvert element ind for sig. Til dette anvendes kolonsymbolet i udtrykkene.

Eksempel

Udtryk af typen:

```
y = 1:7
```

giver vektoren

```
y =
     1  2  3  4  5  6  7
```

Det vil sige, at der oprettes en rækkevektor, hvis elementer løber fra 1 til 7. Her er afstanden 1 mellem hvert af elementerne.

□

Andre intervallængder end enheden 1 kan anvendes, ligesom en negativ ændring er mulig. Mellem to kolontegn angives intervallængde og hvis denne ønskes negativ markeres dette med en minustegn.

Eksempel

En ny vektor løbene fra 3 til 1.75 med intervallængden 0.25 ønskes oprettet. Derfor skrives der:

```
y = 3 : -0.25 : 1.75
```

og der gives svaret

```
y =
    3.0000  2.7500  2.5000  2.2500  2.0000  1.7500
```

□

Funktioner

Der tilbydes rigtig mange forskellige slags hjælpfunktioner til behandling af matricer. Her vil kun et udvalg blive præsenteret. Hvis A er en matrix gives der følgende funktioner.

- `poly(A)` finder det karakteriske polynomium for matricen A. Svaret gives som en vektor, hvis elementer er polynomiets koefficienter, startende med højestegradsleddet.

- `eig(A)` finder egenværdierne til matricen A. Dette svarer til at rødderne i det karakteriske polynimium beregnes. Svaret præsenteres ligeledes som en vektor, hvis elementer er matricens egenværdier. ønskes det ligeledes at bestemme matricens egenvektorer skal der anvendes en udvidet notationsform: `[V,D] = eig(A)` Her bliver D en diagonal matrix, der indeholder egenværdierne og V bliver en matrix, hvis søjler er de tilhørende egenvektorer.
- `det(A)` beregner determinanten til matricen A.
- `size(A)` returnerer en vektor på to elementer, der udgør række- og søjledimension for matrix A.
- `length(A)` giver længden af en vektor eller hvis A er en matrix returnes den største værdi af række og søjledimensionerne. (`max(size(A))`).
- `inv(A)` er den inverse matrix til matricen A, hvor A er invertibel.
- `cond(A)` beregner konditionstallet for matrix A. Dette tal er et udtryk for, hvor singulær det lineære system er.
- `rcond(A)` beregner den reciprokke værdi af konditionstallet for A.

Som ved division af matricer er disse funktioner afhængige af om A er singulær eller ej. Der kan derfor forekomme fejlmeddelelser og vurderinger af matricens rank og konditionstal som kommentar til resultatet, hvor rank er et mål for hvor tæt på singulær matricen er. (Se MATLABs User's Guide for flere detaljer omkring MATLABs rank-funktion).

Eksempel

En matrix A er givet ved `A = [2 -3 -1;-1 1 0;-3 0 1]`. Egenværdierne bestemmes ved at skrive:

```
x = eig(A)
```

og der gives svaret

```
x =
     4.0000
    -1.0000
     1.0000
```

Ligeledes er det muligt at bestemme egenvektorerne i kombination med en diagonal matrice indeholdende egenværdierne.

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

Her er svaret de to matricer V og D:

```
V =
    -0.6882    0.5345    0.0000
     0.2294    0.2673   -0.3162
     0.6882    0.8018   -0.9487

D =
     4.0000         0         0
        0     -1.0000         0
        0         0    1.0000
```

De egenvektorer der angives er ikke nær så ”pæne” som de der findes ad analystisk vej. Dette skyldes at MATLAB angiver egenvektorerne normeret, således at de alle har længden 1.

□

ADVARSEL: Brugeren må selv forholde sig til løsningen. Her gives et eksempel der tilsyneladende går godt:

Eksempel

Matricen B angives som `B = [1 2 2;2 -1 2;2 -2 1]`, hvilket er en ikke-singulær matrix. Ved kommandoen `[V, D] = eig(B)` gives følgende svar:

```
V =
     0.8018   -0.7071    0.7071
     0.5345    0.0000    0.0000
     0.2673    0.7071   -0.7071

D =
     3.0000         0         0
        0     -1.0000         0
        0         0    -1.0000
```

Men B er ikke diagonaliserbar, idet den geometriske multiplicitet er forskellig fra den algebraiske multiplicitet. Alligevel indikerer notationen i MATLAB at det kan lade sig gøre. Dette må brugeren selv være opmærksom på. Det er selvfølgelig et symptom at der er to ”ens” søjler i V.

□

Udvælgelse af dele og elementer i en matrix

Det kan være ret nyttigt at kunne udvælge dele af matricer, til speciel behandling. MATLAB giver mulighed for at specificere elementer, søjler, rækker og submatricer i matricer. Dette gælder også for vektorer, der blot behandles som matricer.

Eksempel

Der vil her blive givet eksempler, på nogle af de forskellige måder at foretage udvælgelser i en matrix på. Matricen A er givet som i forrige eksempel. Skal der referers til matricens indre bliver det nu på formen `A(`*række*`,`*søjle*`)` Hvis nu element nummer 2 i søjle nummer 3 ønskes udvalgt skrives `A(2,3)`.

Er der ønske om hele søjle 3 skrives i stedet `A(:,3)`. Her står kolontegnet således for alle rækkerne i søjle 3, altså hele søjlen. Ønskes kun de to første elementer i søjle nummer 3 skrives `A(1:2,3)`. Hvis der ønskes række 2 element nummer 2 og 3 skrives `A(2,2:3)`. Ønskes 2×2 submatricen bestående af øverste højre hjørne af A skrives

```
X = A(1:2,2:3)
```

der giver svaret

```
X =
      2    2
     -1    2
```

□

Specielle matricer

Der gives flere funktioner til nemt at generere matricer. Vektorer kan også oprettes ved følgende funktioner. Her præsenteres et udvalg:

- `zeros(n)` genererer en nul-matrix af dimensionen n $\times$ n. `zeros(n,m)` giver tilsvarende en n $\times$ m matrix.
- `ones(n)` og `ones(n,m)` generer henholdsvis en n $\times$ n og en n $\times$ m matrice, hvis elementer alle er 1-taller.
- `rand(n)` eller `rand(n,m)` opretter tilsvarende matricer, hvis elementer er genereret tilfældigt fra intervalet $[0,1]$.

- `randn(n)` eller `randn(n,m)` opretter matricer, hvis elementer er normalfordelt omkring 0 og med spredningen 1.

hvor `n` og `m` betegner naturlige tal.

De to sidste funktioner `rand` og `randn` kan også benyttes som talgenerator for enkeltvis tal ved blot at skrive `rand` eller `randn`.

Eksempel

Her oprettes en matrix X, hvis elementer er tilfældigt genererede udfra intervallet [0,1].

Der skrives

```
X = rand(3,4)
```

og der gives f.eks. svaret

```
X =
     0.2190   0.6793   0.5194   0.0535
     0.0470   0.9347   0.8310   0.5297
     0.6789   0.3835   0.0346   0.6711
```

□

2.4 Databehandling

I de tidligere afsnit er matricer blevet behandlet som matematiske størrelser, men der er også blevet set på matricer som tabeller, hvorpå man ønsker at behandle elementerne enkeltvis. MATLAB tilbyder muligheder for flere operationer med henblik på matricer. Det kan f.eks. være relevant at beregne gennemsnitsværdier eller summen af elementerne i en kolonne, svarende til en søjle i matricen.

2.4.1 Funktioner

Følgende er en oversigt over de væsentligste funktioner, der kan anvendes på matricer og på vektorer. Funktionerne er afhængige af om der opereres på en matrix eller en vektor. Efter præsentationen følger enkelte eksempler på anvendelse af nogle af funktionerne.

- `max(x)` Hvis x er en vektor vil resultatet af denne funktion være det største element i vektoren. Når x er en matrix vil funktionen returnere en række-vektor, hvis elementer er de største elementer fra hver af søjlerne i x.
- `min(x)` Hvis x er en vektor vil resultatet af denne funktion være det mindste element i vektoren. Hvis x er en matrix vil funktionen returnere en række-vektor, hvis elementer er de mindste elementer fra søjlerne i x.
- `mean(x)` Funktionen beregner gennemsnittet af elementerne i vektoren x. Hvis x er en matrix er resultatet en række-vektor, hvis elementer er gennemsnitsværdien for hver søjle i x.
- `median(x)` Her beregnes medianen for elementerne i vektoren x. Er x en matrix vil funktionen returnere en rækkevektor hvis elementer er medianen for hver søjle i x. Anvendelsen af denne funktion indbefatter en sortering af elementerne i x, derfor kan udførelsen af denne være ret omfattende for store matricer.
- `std(x)` Standard afvigelsen eller spredningen er af elementerne i vektor x beregnes med denne funktion. I det tilfælde at x er en matrix, vil funktionen give en rækkevektor som svar. Elementerne i denne vil være den beregnede spredning for hver af søjlerne i matricen x.
- `sort(x)` Sorteringsfunktionen sorterer elementerne i vektoren x i voksende orden. Er x en matrix vil hver søjle blive sorteret i voksende orden. Der returneres således en vektor eller en matrix, der er sorteret.
- `sum(x)` Funktionen beregner summen af elementerne i vektoren x. Når x er en matrix vil der blive returneret en vektor, hvis elementer er summen af hver søjle i x.
- `prod(x)` Svarende til sumfunktionen beregner denne funktion produktet af elementerne i vektoren x eller returnere en rækkevektor med produkterne af elementerne i hver søjle af matricen x.
- `hist(x)` Denne funktion beregner og plotter histogramer. Der konstrueres et 10 stk. bjælke-histogram af elementerne i vektoren x.

Præsentationen af funktionerne her er kun den skrabede udgave af deres anvendelse. Det kan anbefales at konsultere MATLABs egen manual ("Reference Guide"), eller benytte MATLABs `doc` kommando, hvis der er behov for en mere udvidet anvendelse og/eller for at se flere eksempler på anvendelser.

Eksempel

Først anskaffes en matrix x, ved at anvende `magic`-funktionen, der konstruerer tilfældige blandede n $\times$ n matricer af tallene 1 til n^2. Her er n = 4.

```
x = magic(4)
```

Der f.eks. giver følgende resultat:

```
x =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

Nu kan sorteringsfunktion anvendes på matricen x. Der skrives :

```
sort(x)
```

Og svaret der fremkommer bliver:

```
ans =
     4     2     3     1
     5     7     6     8
     9    11    10    12
    16    14    15    13
```

□

Eksempel

Her startes med at skaffe en vektor x bestående af 100 elementer. Disse genereres ved hjælp af talgeneratoren `randn`. Derfor skrives først:

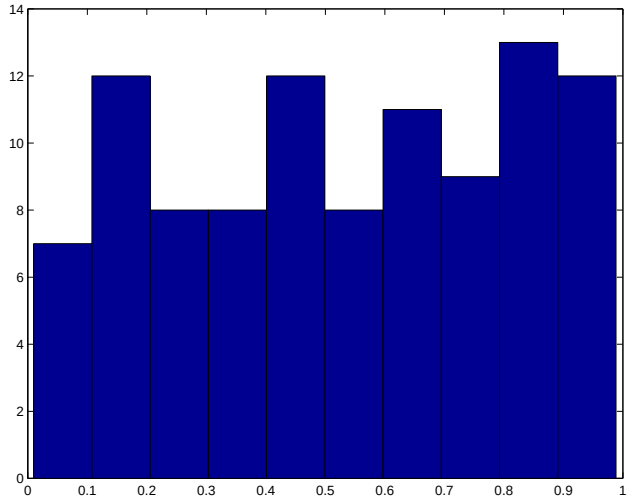
```
x = randn(100,1);
```

Hvorved en 100×1 matrix konstrueres. Semikolonnet, der står efter udtrykket fortæller MATLAB, at resultatet ikke ønskes udskrevet på skærmen. Dette er tit at foretrække fordi udskrivning til skærmen er tidskrævende. Den eksakte værdi af vektorens komponenter er uinteressant i denne forbindelse. Nu kan histogramfunktionen anvendes.

```
hist(x)
```

Plottet der nu konstrueres vil ikke blive vist i kommandovinduet, men i et specielt figurvindue, der automatisk bliver åbent, hvis det ikke allerede er åbent. Man kan eventuelt bladre hen til dette nye vindue, ved hjælp af **Windows**-menuen. Under **Windows**-menuen vises titlerne på de vinduer der er åbne. Ved at vælge **1 Figure** punktet vil figurvinduet komme i forgrunden af de åbne vinduer. Man kan også bladre imellem alle åbne vinduer ved hjælp af ALT TAB tasterne. Histogrammet er vist på figur 2.1.

□



Figur 2.1 Dette er histogrammet, der plottes i figurvinduet ved anvendelse af funktionen `hist`.

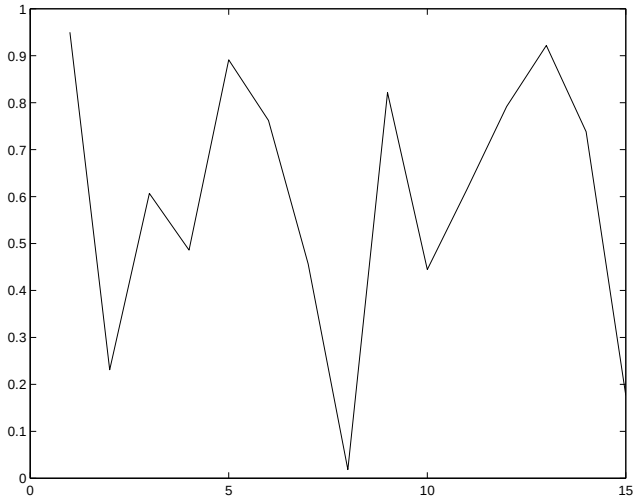
2.4.2 Manglende elementer

Når man arbejder med datamateriale på matrixform forekommer det, at der mangler data på enkelte pladser i matricen. Disse "tomme" felter kan det være hensigtsmæssigt at have med i matricen. Man kan vælge at benytte en udvalgt værdi til at markere hullerne i matricen, f.eks. 0 , -1 eller et stort tal (opfattet som ∞). Men dette vil medføre et misvisende resultat, hvor der skal beregnes på værdierne, f.eks. når der beregnes gennemsnittet for en søjle. I stedet kan man anvende betegnelsen `NaN` (Not-a-Number). Valget af `NaN`, løser imidlertid ikke problemerne ved beregningerne. I de fleste beregninger, hvor resultatet afhænger af dette element vil resultatet blive `NaN`. Af de ovenfor nævnte funktioner kan kun `sort` funktionen uden problemer anvendes selvom matricerne indeholder NaN elementer.

At anvende NaN til de manglende dataelementer betyder således, at de fleste hjælefunktioner er ubrugelige. Til gengæld undgås de misvisende resultater ved databehandlinger, der ikke afslører, at der er huller i datamaterialet.

2.5 Grafik

En del af MATLABs funktioner tjener til at konstruere grafiske præsentationer. I dette afsnit vises kun nogle enkle muligheder for 2-D og 3-D grafik. Ved mere avanceret brug anbefales det, at konsultere MATLABs egne manualer.



Figur 2.2 Dette er plottet af en vektor, hvis element er genereret udfra `randn`-funktionen

2.5.1 2-Dimensional grafik

Den mest elementære funktion til at tegne grafer med er `plot` funktionen. Denne tager en vektor som inputparameter og producerer en stykvis lineære graf af elementerne i vektoren versus indeks af elementerne i vektoren. F.eks. en vektor hvis elementer svarer til et datasæt. `plot` kan også tage to vektorer x og y til input og funktionen vil producere en graf af vektor y versus vektor x.

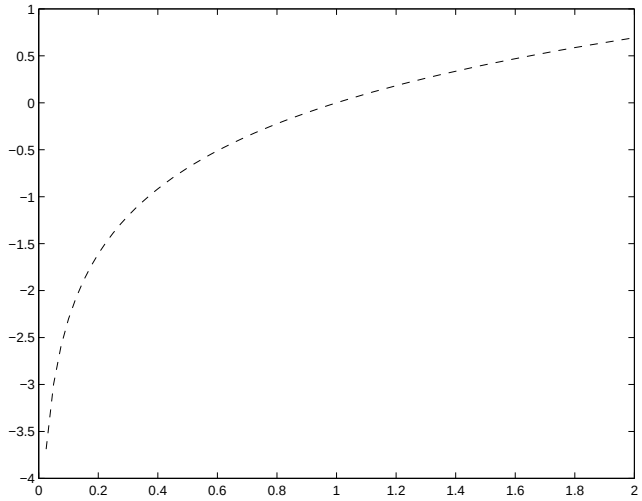
Eksempel

Først kunstrueres en vektor x og man vælger f.eks. at anvende `randn`-funktionen. Der skrives `x = randn(15,1)` på kommandolinien. Herefter plottes vektoren x ved at taste:

```
plot(x)
```

Resultatet vil nu vises i figurvinduet, se på figur 2.2. Som standard, forbinder `plot` de enkelte punkter med en ret linie.

□



Figur 2.3 Dette er plottet af den naturlige logaritme, hvor grafen er tegnet med en stiplet linie.

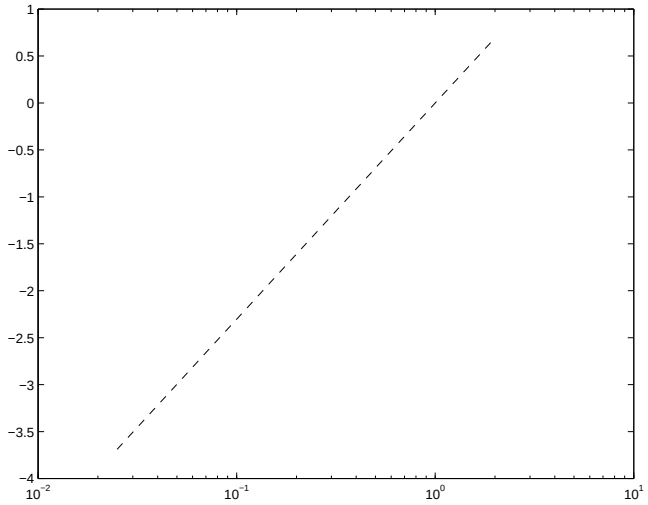
Eksempel

En matematisk funktion kan plottes ved at anvende plot funktionen med to vektorer som input[1]. Vektoren t skal fungere som første akse. Denne angiver således et interval og en intervalinddelling mellem de punkter hvorudfra grafen konstrueres. F.eks. skrives: `t = 0.025: 0.025 : 10` på kommandolinien. Intervallet er nu [0.025,10] og skridtlængden er 0.025. Herefter konstrueres vektoren y, der indeholder funktionensværdierne. Funktionen er her den naturlig logaritmefunktion og for vektoren y skrives `y = log(t)`. Det er muligt at angive om grafen skal bestå af adskilte punkter, eller være en sammenhængende kurve. Om det skal være krydser eller stjerne som punkter eller om det skal være en fuld optrukken eller stiplet linie. Samlet kan dette opskrives som:

```
t = 0.025: 0.025 : 2
y = log(t)
plot(t,y,'--')
```

Her er angivet at grafen skal være en stiplet linie ved at taste to bindestreger mellem to enkel citationstegn. Citationstegnene er vigtige, fordi de fortæller MATLAB, at her kommer en angivelse af grafens udseende. Bemærk, at rækkefølgen af t og y viser, at t ønskes ud af 1-aksen og y ud af 2-aksen. Resultatet vises i figurvinduet og ses på figur 2.3. Fordi den matematiske funktion her er en logaritmisk funktion, kunne det have interesse med akser, der er logaritmisk inddelt. Dette kan let gøres ved at anvende hjælpefunktionen `semilogx` istedet for `plot`. Der tastes

[1] Inputvektorerne skal adskilles med et komma



Figur 2.4 Dette er plottet af den naturlig logaritme funktion, hvor første aksen er inddelt logaritmisk. Kun de positive data plottes

```
semilogx(t,y,'--')
```

Og figuren i figurvinduet ændres til de nye akser. Resultatet kan ses på figur 2.4

□

Udover at kunne angive hvilke typer af linier og punkter, der ønskes anvendt i grafen kan eventuelle farver også specificeres. Hvis f.eks. der ønskes en rød graf skrives 'r-' istedet for '-'. Er der ingen angivelse af farve og form er MATLABs default indstilling en fuld optrukken blå linie. I tabellen 2.1 ses en oversigt over de angivelser, der kan anvendes.

Det er muligt at plotte to eller flere grafer i samme figur. Dette kan gøres på to forskellige måder. Her vil dette blive vist ved at udvide forrige eksempel.

Eksempel

I forrige eksempel konstrueres vektorerne `t = 0.025: 0.025 :2` og `y = log(t)`. Her generes en ny vektor x ved at skrive `x = t .* log(t)`. Graferne for y og x begge versus t ønskes nu i samme figur. Grafen for y skal være en fuld optrukken rød linie, mens x ønskes markeret som en blå stiplet linie. Der ønskes igen anvendt lineære akser. På kommandolinien tastes nu:

```
plot(t,y,'r-',t,x,'b--')
```

Bemærk, at det er vigtigt at angive t igen som 1-akse ved grafen for x. Den anden løsning er at benytte de specielle kommandoer `hold on` og `hold off`.

Tabel 2.1 Oversigt over de symboler, der kan anvendes til at angive grafens stil.

symbol	farve/linie	symbol	punkt stil
y	gul (yellow)	.	prik
m	magentarød	o	cirkel
c	cyan	x	kryds
r	rød	+	plus
g	grøn	*	stjerne
b	blå	s	firkant
w	hvid (white)	d	diamant
k	sort (black)	v	trekant (op)
-	fuldt optrukken	<	trekant (venstre)
:	prikket	>	trekant (højre)
-.	stiplet/prikket	p	pentagram
–	stiplet	h	hexagram

`hold on` kommandoen sørger for, at MATLAB ikke fjerner tidligere grafer, når en ny graf skal tegnes. Det kan dog være nødvendigt at omskalere akserne, men den oprindelige graf vil da blive overført til de nye akser. Når `hold on` effekten ikke ønskes mere skrives `hold off`. Graferne for y og x kan således blive tegnet ganske tilsvarende ved at skrive:

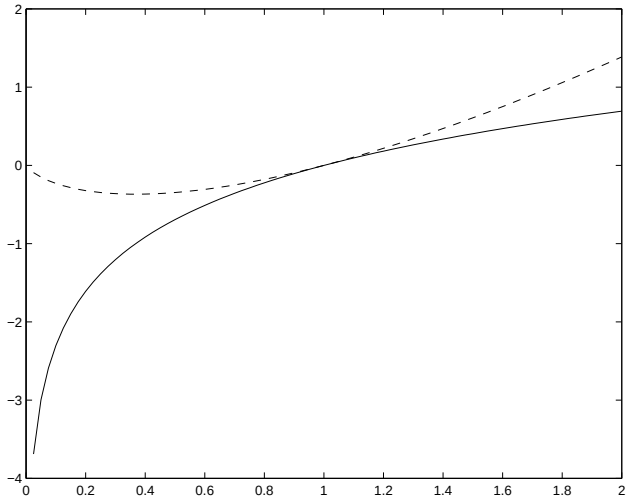
```
hold on
plot(t,y,'r-')
plot(t,x,'b--')
hold off
```

På figur 2.5 ses resultatet. I dette sort-hvide tryk vil farverne af graferne naturligvis ikke kunne gengives.

□

Når grafen for det ønskede er plottet i figurevinduet, kan der tilføjes læsevejledende tekster til vinduet. Der er flere hjælpefunktioner her til.

- `title('`*tekst*`')` Denne funktion sætter den titel, der er angivet mellem de to citationstegn, på figurvinduet.
- `xlabel('`*tekst*`')` Her sættes en tekst på første aksen af figuren.
- `ylabel('`*tekst*`')` Med denne funktion anbringes en tekst på anden aksen af figuren.
- `text(x,y,'`*tekst*`')` Her placeres teksten ind på grafen på det punkt, der er angivet ved koordinaterne x og y.



Figur 2.5 Her er plottet de to funktioner log(t) (fuld optrukken) og t*log(t) (stiplet).

- `gtext('`*tekst*`')` Nu placeres teksten ind på grafen der, hvor man markerer med musen.
- `legend('`*tekst1*`','`*tekst2*`,...)` Laver en box med tekst til de enkelte grafter. Boxen kan flyttes rundt med musen

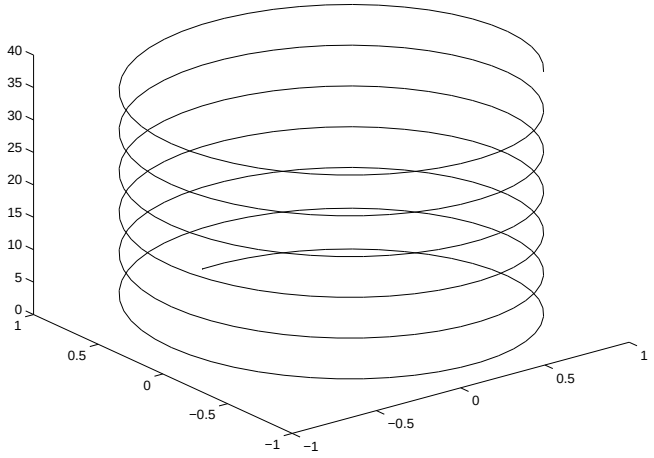
2.5.2 3-Dimensional grafik

Af de talrige 3-dimensionale grafik muligheder som MATLAB byder på er det kun kurver og simple flader, der vil blive gennemgået. Den 3-dimensionale udgave af `plot` hedder `plot3`. Denne funktion tager tre vektorer adskilt af kommaer, som input og rækkefølgen angiver, hvilke der er 1. 2. og 3. akse.

Eksempel

Som i det 2-dimensionale tilfælde angives her vektorerne f.eks. x y og z og MATLAB konstruerer en 2-dimensionel projektion af denne 3-D figur. Først angives her vektoren x og herefter de funktioner som y og z skal repræsentere. Samlet kan dette skrives som:

```
x = 0: pi/25 : 40
y = cos(x)
z = sin(x)
plot3(z,y,x)
```



Figur 2.6 Denne graf viser en 2-D projektion af kurven $(t, \cos(t), \sin(t))$.

Ved 3-D plot kan ligeledes angives farve samt linie- og punktstil, som for 2-D grafer. Her i eksemplet anvendes blot defaultværdien. Tilsvarende kan der tilføjes titel og akse-tekster, samt indføres tekster i grafen. Til den tredje akse anvendes funktionen `zlabel('`*tekst*`')` og for tekster i grafen angives punktet med 3 koordinater `text(x,y,z,'`*tekst*`')`. Grafen for dette eksempel kan ses på figur 2.6.

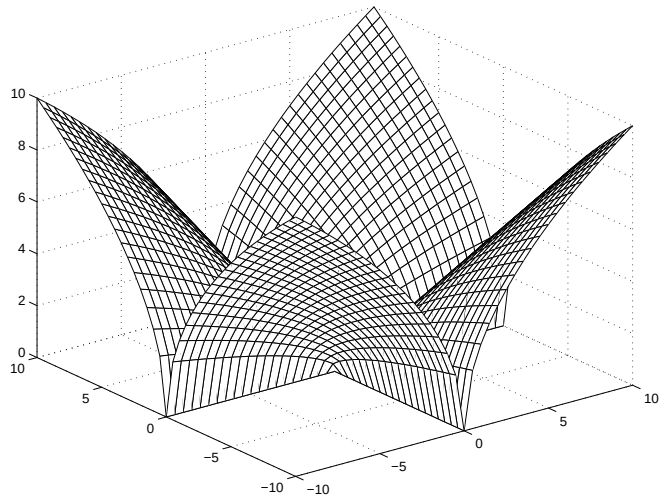
□

Ved gitter-grafer plottes punkter af z koordinater over et rektangulær gitter i xy-planen. Nabopunkter forbindes med en lige linie og derved fremkommer effekten af en flade. Funktionerne af typen $z = f(x,y)$ kan tegnes ved sådanne grafer. Domænet for funktionen z kan opfattes som et gitter i xy-planen. Dette gitter reprænsenteres ved hjælp af 2 matricer X og Y. X matricen indeholder alle planens x-koordinater og tilsvarende indeholder Y alle y-koordinaterne. Disse matricer X og Y anvendes til at beregne værdierne for z-funktionen, samt til at konstruere grafen for z.

Eksempel

Den graf, der skal konstrueres er funktionen for $z = f(x,y) = \sqrt{|xy|}$. Hjælpefunktionen `meshgrid(x,y)` tager to vektorer som input og producerer de to matricer X og Y. Rækkerne i X er kopier af vektor x og søjlerne i Y er kopier af vektoren y. Dette gøres ved at skrive:

```
x = -10 : 0.5 : 10
y = x
[X,Y] = meshgrid(x,y)
```



Figur 2.7 Eksempel på en gitter graf.

Her vil domænet blive kvadratisk, idet x og y er identiske. Der defineres nu funktionen $r = |xy|$. Ved den videre behandling af z-funktionen opfattes denne som $z = \sqrt{r}$. Når z er defineret som en matrix Z anvendes hjælpefunktionen `mesh` til at plotte denne matrix. Dette kan skrives ved:

```
R = abs(X.*Y)
Z = sqrt(R)
mesh(X,Y,Z)
```

Her anvendes hjælpefunktionen `abs` den absolutte funktion. Grafen for funktionen z kan ses på figur 2.7.

□

Kapitel 3

M-filer

Den mest umiddelbare anvendelse af MATLAB er via kommandolinien i Command-vinduet, men man kan vælge at samle kommandoerne i en fil og så udføre alle kommandoerne ved at udføre en kørsel af filen. Filer af denne type kaldes script-filer. Ligeledes kan funktioner respræsenteres som en fil der derved kan manipuleres med. Disse filer benævnes som funktions-filer. Tilsammen kaldes disse filer for M-filer, fordi de begge har endelsen `.m` som fortæller MATLAB, at disse filer er oprettet specielt til brug for MATLAB. M-filer er almindelige ASCII tekst filer.

I kommando-vinduets **File** menu findes et underpunkt **New**. Under dette punkt findes endnu en opdeling i underpunkterne **M-file**, **Figure**, og **Model**. Hvis punktet **Figure** vælges vil MATLAB åbne et nyt vindue specielt til grafik og hvis **Model** vælges åbnes et vindue til modeller (Dette er kun aktuelt, når man ønkser at anvende de dele af MATLAB der falder indunder SIMULINK pakken. Denne er en pakke med special værktøjer til arbejde med modeller. SIMULINK er installeret på IMFUFA, men vil ikke blive yderligere omtalt i dette hæfte). Det er naturligvis **M-file**, der skal vælges når en ny M-fil ønskes oprettet. MATLAB åbner nu et vindue til en default-editor, hvori M-filernes indhold kan skrives. Hvis brugeren ønsker en anden default-editor kan dette præciseres overfor MATLAB ved underpunktet **Editor Preference** under **Options** menupunktet. Ønsker man at rette i en allerede oprettet M-fil vælges **Open M-file**[1].

Arbejder man med MATLAB under LINUX findes der ikke en default-editor. Her står det frit for brugeren at benytte den editor han er mest tryk ved. Dog kan det anbefales at bruge **xemacs** idet den understøtter med farvekoder, som gør programmeringen mere overskuelig.

[1] Alternativt kan man skrive MATLAB kommandoen `edit` i workspace (eller `edit` '*filnavn*'), hvorved editorvinduet automatisk åbnes.

3.1 Scriptfiler

En scriptfil er en samling af MATLAB kommandoer i en fil. Denne fil har endelsen `.m`. Kommandoerne kan operere på de variable der (globalt) er oprettet i Workspace og på de variable, der (lokalt) er oprettet i scriptfilen. Efter udførelsen af en M-fil vil de lokalt oprettede variable også indgå i Workspace som globale variable[2].

Eksempel

En M-fil kunne indholde kommandoerne til at plotte grafer for funktionerne $f(x) = \sin(x)$ og $g(x) = \cos(x)$ i intervalet $[0, 2\pi]$. Filen indeholder da følgende tekst:

```
%Denne scriptfil plotter graferne for sin(x) og cos(x);
x = 0: pi/25: 2*pi;
f = sin(x);
g = cos(x);
hold on
plot(x,f,'r-');
plot(x,g,'g-');
hold off
title('Cosinus og Sinus');
```

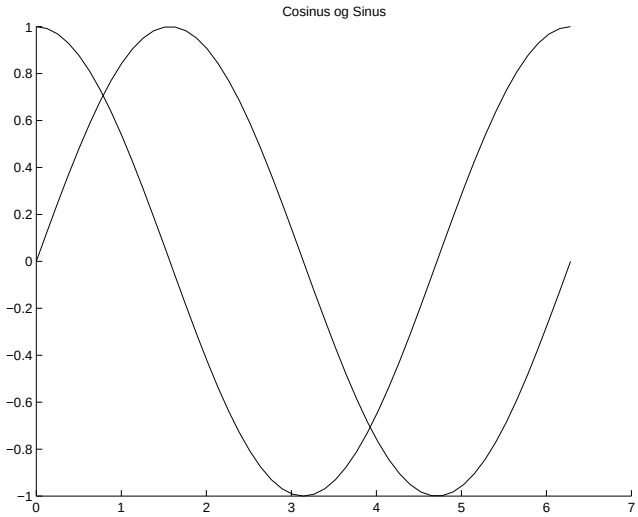
Den øverste linie i filen er allerførst markeret med et procenttegn. Dette betyder, at alt hvad der følger efter dette tegn på denne linie er en kommentar, som MATLAB blot skal ignorere. Det vil sige, at programmet ikke skal prøve at udføre teksten som en kommado. Det er tit en stor hjælp for den der skriver filen (og især for andre, der senere ønsker at læse filen) at indsætte hjælpetekster, der i almindeligt sprog forklarer, hvad der sker i filen.

Hver kommando afsluttes med et semikolon. Dette er ikke en nødvendighed, men fortæller MATLAB, at der ikke skal udskrives resultatet af kommandoen i Command-vinduet. Anvendelsen af plot funktionen vil medføre at graferne plottes i et figurvindue. For overskuelighedens skyld er der kun skrevet en kommando på hver linie, men MATLAB godtager også, at der står flere kommandoer på hver linie, de skal dog adskilles med komma eller semikolon.

Når filen er skrevet skal filen gemmes og kaldes et unikt navn, med endelsen `.m`. Det er vigtigt, at filen ligger i brugerens hjemmekatalog i forbindelse med andre MATLAB filer. Filen, der er oprettet gemmes nu under navnet `sincos.m`[3]. Dette gøres via den valgte editor. Det er nu muligt at udføre en kørsel af filen. Dette kan

[2] En scriptfil adskiller sig fra en funktionsfil ved, at der ikke eksplicit optræder input variable ved kørsel af en scriptfil.

[3] Visse navne er reserveret til indbyggede MATLAB funktioner og scripts, andre navne er forbudte, f.eks. må et filnavn ikke starte med et tal. Envidere bør de specielle danske bogstaver æ, ø og å også undgås.



Figur 3.1 Plot af funktionerne cosinus og sinus.

gøres på to måder: via kommadolinien i Commandvinduet eller via menupunktet **Run M-file** i **File** menuen. På kommandolinien skrives nu `sincos` hvorefter der tastes RETURN. Filen vil nu blive udført. Bemærk at endelsen (.m) ikke skal skrives. Vælges **Run M-file** menupunktet giver MATLAB mulighed for brugeren kan bladre mellem de oprettede M-filer og derved udvælge den der skal udføres[4]. Udførelsen af `sincos.m` filen bevirker, at der åbnes et figurvindue, hvori graferne plottes. Resultatet kan ses på figur 3.1.

□

Der er dog lidt problemer i kommunikationen mellem NT, MATLAB og NOVEL. Dette gør at MATLAB ikke kan 'se' en m.fil, der er oprettet efter MATLAB er opstartet. Dette problem kan 'løses' på flere måder, den nemmeste er at oprettte en m-fil af navn `startup.m`, med indholdet:

```
sysstem_dependent RemotePathPolicy Reload;
```

Hvis denne ligges i det katolog MATLAB starter op i, køres den automatisk, ellers skal den køres som det første man gør når MATLAB startes op.

3.2 Funktionsfiler

Funktionsfiler adskiller sig fra scriptfilerne ved at kunne tage argumenter til filen. Kun variable oprettet lokalt i filen kendes og der er ikke umiddelbar tilgang til

[4]Dele af en M-fil kan køres fra commandvinduet, ved først at markere det ønskede i M-filen og derefter trykke F9.

Workspaces globale variable. Det er via argumenter at eventulle variabelværdier overføres.

En funktionsfil består i hovedtræk af 4 dele: En funktion definitionslinie, en række hjælpeliner, selve funktionen og en række kommentare. Det allerførste er en definitionslinie, der starter med ordet `function`, som fortæller MATLAB at m-filen indeholder en funktion og dens specifikationer. Herefter følger en række linier med kommentarer, der alle er markeret med et procenttegn forest. Som hjælp vil disse kommentarlinier umiddelbart efter funktionslinien, blive indlemmet som en hjælpetekst. Hvis der søges om hjælp (med `help` '*filnavn*'), så udskrives disse kommentarlinier til commandvinduet. Ved søgning med `lookfor` kommandoen vises kun den første hjælpelinie. Derfor skrives der på disse hjælpelinier kort, hvad funktionsfilen kan og gør. Det er ikke et krav med disse hjælpelinier, det er udelukkende en service til brugere af funktionen. Generelt er det sådan, at linier i funktionsfiler, der starter med et procenttegn af MATLAB vil blive opfattet som en kommentar og ikke som en del af funktionen. Det er derfor muligt at fortælle en bruger (evt. sig selv nogle uger efter at funktionsfilen er lavet) hvad de forskellige dele af funktionen gør. Endelig er der specifikationen af selve funktionen.

Eksempel

Der ønskes en funktionsfil til at repræsentere polynomier af 3. grad. I en M-fil med navnet `pol3.m` oprettes følgende funktionsfil.

```
function y = pol3(x)
global A B C D;
y = A .* x.^3 + B .* x.^2 + C .* x + D;
```

Når A, B, C og D er sat som globale vil MATLAB søge ude i Workspace efter parametrene A, B, C og D. Da der fra en funktionsfil ikke er direkte adgang til de størrelser, der er defineret i workspace, er det nødvendigt at definere disse som globale. Bemærk, at de forskellige variable der erklæres globalt er adskilt af mellemrum. I denne fil er der ikke indlagt hjælpetekst til andre brugere.

Alternativt kan programmet skrives uden brug af globalt defineret størrelser, ved i stedet at skrive:

```
function y=pol3ny(x)
A=3;
B=-2;
C=1;
D=7;
y=A.*x.^3+B.*x.^2+C.*x+D
```

Hvis man ønsker at variere parametrene A, B, C og D, skal dete gøres i selve M-filen (som så skal gemmes på ny).

Efter ordet function står der et variabelnavn `y`. Denne variabel vil i filen få tildelt den værdi som er funktionens resultat. Det vil sige `y` er det resultat der returneres og f.eks. udskrives som svar i Commandvinduet. Når filen anvendes kaldes filen, i dette tilfælde med `plo3`. Det ses at `pol3`-funktionen tager et argument `x`, hvilket er den vektor funktionen opererer på. Det er muligt at lade funktionsfiler returnere mere end kun en værdi. Til sidst følger selve funktionen, der er et generel 3. grads polynomium.

Nu kunne funktionsfilen `pol3.m` anvendes til at konstruere grafen for $f(x) = 4x^3 - 2x^2 + x - 3$. På kommandoelinien (eller i en scriptfil) skrives nu:

```
t = -5 : 5;
A = 4; B = -2; C = 1; D = -3;
y = pol3(t);
plot(t,y);
```

Bemærk, at variablen y her ikke er den samme som i `pol3`-filen.

□

De MATLAB funktioner, der er præsenteret her i manualen er for manges vedkommende programmeret som funktionsfiler. Ved at brugeren selv konstruerer funktionsfiler i MATLAB vil denne hele tiden kunne udvide MATLAB til sit eget behov. Ved at kikke på funktionsfilerne for nogle af de indbyggede hjælpefunktioner, kan man få en et indtryk af hvordan mere kompliceret funktionsfiler opbygges.

F.eks. kunne man åbne funktinsfilen for `std`, der beregner spredningen af elementerne i en vektor eller matrix. Funktionen tager således en vektor eller matrix som input til filen. Indholdet af `std`-filen listes i Command-vinduet ved at taste `type std.m`. Herefter listes:

```
function y = std(x)
%STD  Standard deviation.
%     For vectors, STD(x) returns the standard deviation.
%     For matrices, STD(X) is a row vector containing the
%     standard deviation of each column.
%
%     STD computes the "sample" standard deviation, that
%     is, it is normalized by N-1, where N is the sequence
%     length.
%
%     See also COV, MEAN, MEDIAN.

%     J.N. Little 4-21-85
%     Revised 5-9-88 JNL
%     Copyright (c) 1984-93 by The MathWorks, Inc.
```

```
[m,n] = size(x);
if (m == 1) + (n == 1)
    m = max(m,n);
    y = norm(x-sum(x)/m);
else
    avg = sum(x)/m;
    y = zeros(size(avg));
    for i=1:n
        y(i) = norm(x(:,i)-avg(i));
    end
end
if m == 1
    y = 0;
else
    y = y / sqrt(m-1);
end
```

Som det fremgår af `std`-filen er det selvfølgelig muligt at anvende andre funktionsfiler i denne fil. `std`-filen gør brug af funktionerne `size, max, norm, sum, zeros` og `sqrt`. Bemærk, at efter alle kommandoerne står et semikolon, der indikerer at resultatet af denne kommando ikke skal skrives ud i kommando-vinduet. Hvis de ikke stod der, ville alle mellemregningerne blive skrevet ud i kommandovinduet. I filen anvendes den specielle if-sætning flere gange til at kontrollere udførelsen af beregningen.

Som det allersidste i filen, tildeles variablen `y` resultatværdien. Bemærk, at y både kan være en skalar (hvis x argumentet er en vektor) og en vektor (hvis x er en matrix).

I `std`-filen anvendes variablene `m, n, avg` og `y` som lokale variable. Men for funktioner der f.eks. skal repræsentere matematiske funktioner, hvori der indgår en række parametre kan det være hensigtsmæssigt at erklære disse parametre for globale, således at værdierne kan blive tildelt udenfor funktionsfilen.

3.3 Profiler

Når en M-fil bliver tilpas stor kan den tid MATLAB bruger på at udføre M-filens ordre blive så lang, at det bliver et problem. For at finde ud af hvor MATLAB bruger tiden, findes der et profileringsværktøj, som giver mulighed for at finde tidsmæssige flaskehalse i funktioner. Værktøjet giver informationer om, hvor stor en del af det totale tidsforbrug, der forbruges på hver enkelt linie i en M-fil.

Profileringsværktøjet startes med `profile on`, derefter køres funktionen. Kommandoen `profile report` genererer et HTML dokument. I dette er der både

en kort oversigt hvor der bla. står det totale tidsforbrug, og en detaljeret profil af funktionen, hvor der specificeres hvor meget tid der bruges på hver enkelt programlinie. Profileringen afsluttes med `profile off`.

Eksempel

Funktionen herunder generer en tabel med funktionsværdierne for funktionen $y = 1 + x^3$ i intervallet $[-2 : 2]$ i step af 0.001 og gemmer x- og y værdierne i hhv. r og s.

```
function profiex
r=[];s=[];
for x=-2:0.001:2
    y=1+x^3;
    r=[r x];
    s=[s y];
end
```

Først fortæller man MATLAB at funktionen ønskes profileret:

```
profile on
```

Dernæst køres funktionen:

```
profiex
```

Og tilsidst genereres en tidprofil af funktionen:

```
profile report, profile off
```

HTML dokumentet startes nu automatisk op og i dette tilfælde bla. viser:

```
Total recorded time: 4,46s
```

mens den detaljeret profil viser:

```
100% of the total time in this function was
     spend on the following lines:


                    3:     for x=-2:0.001:2
0.121s,       3 %   4:         y=1+x^3;
2.253s,      51 %   5:         r=[r x];
2.052s,      46 %   6:         s=[s y];
0.030s,       1 %   7:     end
```

Som det fremgår er det række 5 og 6, der er flaskehalse i `profiex`. På disse linier udføres en omfattende dynamisk allokering af hukommelse. Hver gang løkken gennemføres skal der allokeres plads til yderligere to elementer. En god vane er derfor at allokere pladsen på forhånd. En alternativ løsning kan så komme til at se således ud:

```
function profiex
x=-2:0.001:2;
r=zeros(size(x));s=zeros(size(x));
i=0;
    for x=-2:0.001:2
        i=i+1;
           y=1+x^3;
           r(i)=x;
           s(i)=y;
    end
```

Hvis samme procedure gentages fås nu en anderledes profil:

```
Total recorded time: 0,17s
```

mens den detaljeret profil viser:

```
100% of the total time in this function was
     spend on the following lines:


                 3:  r=zeros(size(x));s=zeros(size(x));
0.01s,     5%    4:  i=0;
                 5:  for x=-2:0.001:2
0.04s,    19%    6:      i=i+1;
0.11s,    53%    7:         y=1+x^3;
0.30s,    14%    8:         r(i)=x;
0.01s,     5%    9:         s(i)=y;
0.030s,    5%   10:  end
```

Som det se er tidsforbruget til den dynamiske allokering betydeligt nedsat, da den kun udføres en gang.

Det mest optimale program udnytter, at MATLAB er vektorielt:

```
function profiex
x=-2:0.001:2;
y=1+x.^3;
r=x;
s=y;
```

Profilen for dette program er:

```
   Total recorded time: 0.01s
```

med den detaljeret profil:

```
  100% of the total time in this function was
       spend on the following lines:


                    3:  x=-2:0.001:2;;
  0.01s,   100%     4:  y=1+x.^3;
                    5:  r=x;
                   6:  s=y;
```

□

Kapitel 4

Hjælpefunktioner til matematiske funktioner

En række af MATLABs funktioner er ikke beregnet til brug ved matricer, men til matematiske funktioner. Funktioner repræsenteres i MATLAB ved en M-fil, se afsnit 3.2 side 35 . I det følgende forudsættes det, at der er oprettet en fil ved navn `funkfil.m`, der repræsenterer følgende matematiske funktion:

$$y = \frac{4}{1+x^2}$$

4.1 Numerisk integration

Der findes to metoder til numerisk integration af en funktion $f(x)$ over et givet interval. Denne beregning kaldes for kvadratur. Der løses problemer, der matematisk formuleres som

$$q = \int_a^b f(x)\, dx$$

De to metoder er:

- `quad('`*fname*`',a,b)` - Funktionen finder en Simpson approksimation som løsningen. Filnavnet i kursiv skal erstattes med navnet på den M-fil, der repræsenterer funktionen $f(x)$. De enkelte citationstegn er vigtige og fortæller MATLAB, at dette er navnet på en M-fil og ikke en indbygget funktion i MATLAB. Intervalet er angivet her fra a til b.

- `quad8('`*fname*`',a,b)` - Her benyttes en rekursiv Newton-Cotes metode til løsningen. Der gælder de samme forhold for `quad8` som for `quad` omkring funktionsfilen og intervalangivelser.

Antallet af rekursioner for begge hjælpfunktioner er begrænset og det kan ske, at dette antal ikke er nok til at en løsning kan findes. MATLAB vil da give en fejlmeddelelse. Det er muligt at ændre loftet for det maksimale antal iterationer en numerisk løsningsmetode skal gennemløbe. Ændringen skal ske som en ændring af en specifik vektor, der er defineret af MATLAB. Se MATLABs egne manualer vedrørende optionsvektor.

Eksempel

Arealet under en positiv funktion defineret med funktionsfilen `funkfil.m` i intervalet $[0, 1]$ ønskes bestemt. På kommandolinien skrives nu:

```
quad('funkfil',0,1)
```

Hvorefter at MATLAB beregner resultatet til :

```
ans =
      3.1416
```

Den anden metode vil ved kommandoen `quad8('funkfil',0,1)`i dette tilfælde producere tilsvarende:

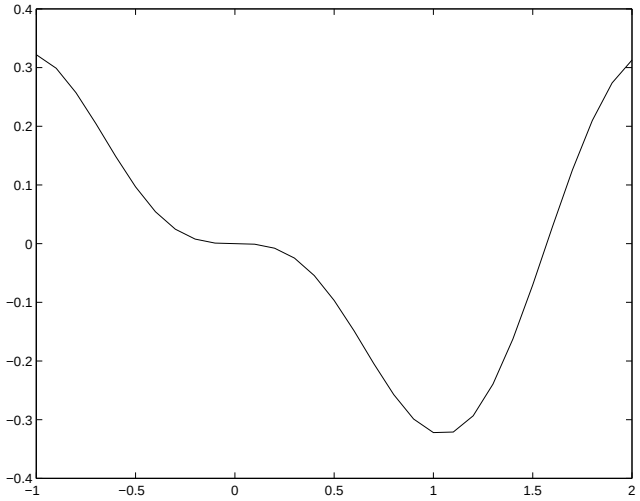
```
ans =
      3.1416
```

□

4.2 Ikke-lineær ligninger og optimeringsfunktioner

I standardpakken af MATLAB gives der 3 hjælpefunktioner for ikke-lineære funktioner. Disse er som følger:

- `fminbnd('`*fname*`',x1,x2)` Denne funktion beregner minimum for funktionen angivet i en M-fil. Funktionen må kun være afhængig af en variabel. Intervalet, hvori minimum ønskes fundet, angives ved grænserne `x1` og `x2`.
- `fminsearch('`*fname*`',x0)` `fminsearch` finder minimum for funktioner af flere variable. Der angives et begyndelsesgæt `x0`, hvori metoden tager udgangspunkt. `fminsearch` anvender en simplex-søgemetode udarbejdet af Nelder og Mead.
- `fzero('`*fname*`',x0)` Der findes et nulpunkt for funktioner af en varibel, nær ved begyndelsesgættet `x0`.



Figur 4.1 Funktionen $f(x) = -\sin^3(x)\cos(x)$

Eksempel

En M-fil ved navn `trifil.m` er oprettet for at repræsentere funktionen:

$$f(x) = -\sin^3(x)\cos(x)$$

På figur 4.1 ses en skitse af denne funktion. Først ønskes det at finde minimum i intervalet $[0, \frac{\pi}{2}]$. Der skrives:

```
m = fminbnd('trifil', 0, pi/2)
```

og der gives svaret

```
m =
      1.0472
```

Bemærk at dette kun er et lokalt minimum. Herefter søges der efter et nulpunkt udfra et begyndelsesgæt på værdien $\frac{\pi}{8}$.

```
n = fzero('trifil', pi/8)
```

Det svar der beregnes er :

```
n =
    1.5884e-16
```

Dette er ikke det præcise nulpunkt (som er i nul), men nulpunktet forekommer i en vendetangent og nulpunktsalgoritmen har derfor svært ved finde nulpunktet. Se figur 4.1. Bemærk, at MATLAB kun finder et nulpunkt ad gangen. Det er brugeren, der selv må overskue om der kunne være flere og hvor det er hensigtsmæssigt at gætte på en løsning. I dette eksempel kan der således bl.a. findes endnu et nulpunkt ved:

```
p = fzero('trifil', 2)
```

der giver svaret:

```
p =
    1.5708
```

□

4.3 Løsning af differentialligningssystemer

MATLAB tilbyder flere hjælpefunktioner til numerisk løsning af et system af ordinære differentialligninger, bl.a. `ode23` og `ode45`, der benytter en 2. og 3. hhv 4. og 5. ordens Runge-Kutta-Fehlberg integrations algoritme med (automatisk genererede) variabel skridtlængde. Nærværende afsnit vil koncentrere sig om disse to metoder. Benyt hjælpefunktionerne til at undersøge, hvilde andre integrations algoritmer MATLAB kan tilbyde.

Et system af (ikke-lineære) differentialligninger kan altid omformes til et system af første ordens (ikke-lineære) differentialligninger:

$$\dot{x} = f(x,t)$$

hvor $x = x(t) \in \mathbf{R}^n$ er tilstandsvariablen, t er (sædvanligvis) tiden, og f er en funktion der giver den tidsafledede $\dot{x}$ af x som funktion af x og t. Det aktuelle system af (ikke-lineære) første ordens differentialligninger specificeres i en M-fil (se kapitel 3 om M-filer og evt nedestående eksempel).

Med kommandoen

```
[t,x] = ode23('name', tspan, xstart);
```

startes integrationen, af det differentialligningssystem der er specificeret i M-filen ved navn `name`, ved hjælp af ovennævte 2. og 3. ordens integrations algoritme. Hvis man ønsker at benytte en 4. og 5. ordens integrations algoritme skrives `ode45` istedet for `ode23`. Begge integrations algoritmer skal (mindst) have tre input argumenter. Disse er navnet på en M-fil (som allerede beskrevet), det tidsinterval

`[tstart tslut]`, hvori systemet skal løses[1], og en startbetingelse `xstart`. Det skal nævnes, at der er mulighed for at specificere yderligere input argumenter, såsom regnenøjagtighed (se MATLABs egen manual for flere detaljer).

Eksempel

Betragt Van der Pol ligningen, der er en anden ordens differentialligning,

$$\ddot{x} + (x^2 - 1)\dot{x} + x = 0$$

Sættes $x_2 = x$ og $x_1 = \dot{x}$ kan Van der Pol ligningen skrives som følgende system af første ordens differentialligninger:

$$\begin{aligned} \dot{x}_1 &= x_1(1 - x_2^2) - x_2 \\ \dot{x}_2 &= x_1 \end{aligned}$$

Det første skridt i simuleringen er at skabe en M-fil indenholdende disse diferentialligninger. Denne kaldes `vdpol.m` Dette gøres ved at klikke på **File** menuen, derefter vælge undermenuen **New**, og endelig undermenupunktet **M-file**. I denne M-fil skrives:

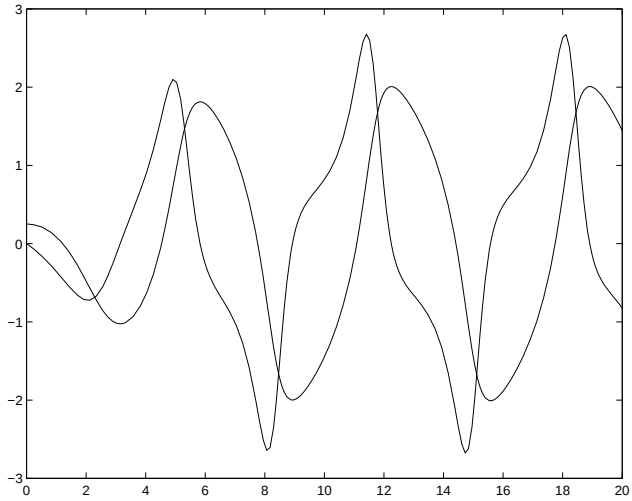
```
function xdot =  vdpol(t,x)
xdot          =  zeros(2,1);
xdot(1)       =  x(1) .* (1 - x(2).^2) - x(2);
xdot(2)       =  x(1);
```

I første linie erklæres funktionen og samtidig fastlægges, at det er argumentet til `vdpol` der returneres. I anden linie nulstilles vektoren `xdot` samtidig med at den oprettes som en 2×1 matrix. I tredie linie skrives den første differentialligning og i fjerde linie skrives den anden differentialligning. Når indholdet i M-filen er indskrevet i denne skal filen gemmes. Dette sker ved at klikke på **File** menuen og derefter klikke på undermenuen **Gem som ...**, herefter åbnes et vindue, hvori filenavnet skrives. I dette eksempel skrives `vdpol.m`. Da er filen gemt og vinduet kan lukkes[2]. Dette er proceduren for at gemme filer i default editoren 'Notepad'. Ved valg af anden editor, kan proceduren være en anden.

For at simulere Van der pol differentialligningen, som er beskrevet i M-filen `vdpol.m` over et tidisinterval $0 \leq t \leq 20$ vha. `ode23` skrives

[1] Ønskes systemet kun løst til bestemte tidspunkter $t_1, t_2, \ldots, t_n$ sættes `tspan` til $[t_1\ t_2, \ldots, t_n]$.

[2] Man behøver ikke lukke vinduet for at kunne benytte M-filen, den skal blot gemmes før brug. Ændringer i en M-fil registreres ikke før filen er gemt på ny.



Figur 4.2 Plot af Van der Pol-systemmet. x(1) og x(2) som funktion af tiden.

```
tspan     = [0 20];     %Starttid og sluttid
                        %for integrationen
xstart    = [0;0.25];   %startbetingelse
[t,x]     = ode23('vdpol',tstart,tslut,xstart);
                        %simuleringen udføres kurverne
plot(t,x)               %x(1) og x(2) plottes versus t
```

På figur 4.2 ses resultatet af dette plot.

De tre første linier giver sig selv. I fjerde linie kaldes `ode23` algoritmen, som mindst skal have fire indput argumenter. Bemærk at navnet på den M-fil der indenholder differentialligningerne optræder uden 'efternavn' og imellem enkelcitationstegn, her `'vdpol'`. Den femte linie åbner et vindue, hvori de numerisk beregnet løsningskurver plottes (se afsnittet om grafik for videre detaljer). Bemærk, at hver beregningslinie, både her og i M-filen, slutter med et semikolon, hvilket betyder, at de beregnede værdier ikke løbende udskrives på skærmen. Dette er meget tidsbesparende. Ønskes nogle eller alle data og beregninger udskrevet løbende på skærmen undlades de tilsvarende semikolon tegn.

Hvis vi istedet for Van der Pol ligningen ønsker at simulerer følgende modificerede system:

$$\begin{aligned} \dot{x_1} &= x_1(\alpha - x_2^2) - x_2 \\ \dot{x_2} &= \beta x_1 \\ \dot{x_3} &= \gamma \sin(t) \end{aligned}$$

for en række forskellige værdier af parameterne α, β og γ, skal disse parameter erklæres som værende globale på følgende måde. (Se kapitel 3 om M-filer for mere information om gloable og lokale variable.) M-filen modificeres til:

```
function xdot  =  vdpol(t,x)
global ALPHA BETA GAMMA
xdot           =  zeros(3,1);
xdot(1)        =  x(1) .* (ALPHA - x(2).^2)-x(2);
xdot(2)        =  BETA * x(1);
xdot(3)        =  GAMMA * sin(t);
```

Parameterne ALPHA, BETA og GAMMA kan nu ændres interaktivt, f.eks. skrives

```
tspan    = [0 20];      %Starttid og sluttid
                        %for integrationen
xstart   = [0;0.25;0];  %startbetingelse
global ALPHA BETA GAMMA %parameterne erklæres globale
ALPHA    = 1.5;         %ALPHA tildeles en værdi
BETA     = 0.85;        %BETA tildeles en værdi
GAMMA    = 0;           %GAMMA tildeles en værdi
[t,x]    = ode23('vdpol',tstart,tslut,xstart);
                        %simuleringen udføres
plot(t,x)               %kurvene x(1) og x(2)
                        %plottes versus t
```

Bemærk, at den modificerede M-fil stadig, noget misvisende, kaldes `vdpol.m` Herefter kan simuleringer for andre værdier af ALPHA, BETA og GAMMA køres ved blot at tildele disse nye værdier interaktivt efterfulgt af en ny integration og af plot funktionen[3].

Ønskes et plot af f.eks. `x(1)` erstattes sidste linie ovenfor blot af

```
plot(t,x(:,1));        %kurven x(1) plottes versus t
```

Bemærk, at den returnerede variabel `x` er en matrix med tre søjler og ligeså mange rækker som antallet af skridt i den aktuelle numeriske integration (hvilket ofte er et relativt stort antal, her 185). Ønsker man at kende antallet af rækker skrives blot `whos`, hvorved oplysninger om bl.a. `x` matricens dimension udskrives. Når der i ovenstående plot funktion står `(t,x(:,1))` som argument, er inputtet til plot funktionen `t`-vektoren og `x(1)`-vektoren. Disse har samme dimension og plot funktionen plotter da punkterne `(t(1),x(1,1)),(t(2),x(2,1)),...,(t(m),x(m,1))`, hvor `m` er antallet af rækker i `x` matricen. Da punkterne ligger meget tæt opnås en illusion om at de danner en glat kurve.

Ønskes et faseplot af f.eks. `x(2)` versus `x(1)` erstattes sidste linie med

```
plot(x(:,1),x(:,2))     %faseplot af x(2) versus x(1)
```

[3] MATLAB registrerer ikke ændringerne i M-filer før filen er gemt på ny, enten ved at klikke gem under **file** på menubjælken eller ved at bruge genvejstasten `CTR S`.

□

Det skal nævnes at her blot er medtaget nogle få af de muligheder MATLAB byder på med hensyn til simulering af ordinære (ikke lineære) differentialliningssystemer, for videre brug henvises til manualen.

Kapitel 5

Input- og output filer

MATLAB indeholder er række redskaber til at håndtere filer, der ikke er konstrueret i MATLAB (import af data). Ligeledes kan der være behov for at eksportere data fra MATLAB til et andet program til videre behandling. F.eks kunne det være en mere relevant måde at opbevare data som binære tal end som ASCII tekst filer. Her vil kun blive gennemgået behandlingen af binære filer, se om nødvendig MATLABs egne manualer for yderligere behandling af filer.

5.1 Åbne og lukke filer

Inden det er muligt at læse eller skrive i en fil skal filen formelt åbnes. Efter brug skal filen igen formelt lukkes. At en fil skal åbnes består i at computeren får lokaliseret filen og sat rettighederne for tilgangen til filen. Skal der udelukkende skrives eller læses fra filen? Eller begge dele? Hvis filen er åbnet med rettigheder til at læse og der gøres forsøg på at skrive i filen vil det resultere i en fejlmeddelelse. Det er en beskyttelse af filen at disse rettigheder sættes. En fil kan kun åbnes af en bruger af gangen. Det kan lyde trivielt, men på flerbrugersystemer (netværk), er det en yderligere beskyttelse, at filen kun kan åbnes af en ad gangen. Ligelede hvis den samme bruger forsøger at åbne den samme fil to gange vil beskyttelsen igen træde i kraft og resultere i en fejlmeddelelse.

Eksempel

Der er oprettet en datafil med navnet `minfil.dat`. På denne fil ønskes data lagt ud, hentet ind fra og overført til en matrix i MATLAB. Først skal `minfil.dat` åbnes. På kommandolinien skrives:

```
fid = fopen('minfil.dat', 'r+');
```

De enkelte citationstegn er vigtige og fortæller MATLAB at her er tale om en fil og ikke en indbygget kommando. Det i citationstegn angivne `'r+'` betyder, at filen skal åbnes for at læse OG skrive i. For kun at skrive i filen tastes `'w'` og for kun at læse tastes `'r'`. Variabelen `fid` er en såkaldt fil-identifikator. Man kan tænke på denne som en pil, der hele tiden peger på filen i lageret. Denne identifikator kan benyttes som argument for bl.a læse, skrive og lukke funktionerne. Men identifikatoren tildeles også en værdi, der returneres af `fopen`-funktionen. Hvis åbning lykkes gives værdien 0 (sand) og -1 (falsk) hvis det ikke lykkedes. Efter åbningen er filen klar til at blive læst fra.

Efter endt brug skal filen nu lukkes. Dette gøres ved `fclose`-funktionen. På kommandolinien skrives :

```
fclose(fid)
```

Der gælder de samme forhold for `fclose` som for `fopen`. Hvis lukningen lykkes returneres værdien 0 og hvis ikke værdien -1. Både for `fopen` og `fclose` indikerer værdien -1, at der er en fejl. Den ønskede kommando kunne ikke udføres. Man kan være heldig, at der gives en fejlmeddelelse, som man med held kan blive klogere af. Denne fejlmeddelelse kan kaldes frem ved at taste `ferror(fid)`. En fejl kunne f.eks. være, at man prøvede at åbne en fil, der ikke eksisterer.

Eksemplet fortsætter på side 54

□

5.2 Læse og skrive i filer

De to funktioner `fwrite` og `fread` anvendes til henholdvis at skrive og læse binært fra filer. Skrivefunktionen opskrives efter formen:

```
c = fwrite(fid, A,'præcision')
```

Ved denne kommando vil A matricens elementer blive skrevet ud på filen med identifikator `fid`, med den nøjagtighed af tallene som angives i citationstegn istedet for ordet præcision. Det vil sige om der f.eks er tale om heltal eller reelle tal og om der er anvendes 8, 16, 32 eller 64 bit til at repræsentere tallet.

`fwrite` returnere en værdi svarende til antallet af elementer, der med succes er blevet skrevet ud på filen. Her er variablen `c` sat til at opbevare denne værdi.

Der er en del forskellige måder, hvorpå en computer kan repræsentere tal. Præciseringen af hvordan de binære tal skal repræsenteres er således vigtig for den computer, hvor MATLAB er installeret. Her følger en liste over forskellige angivelser af præcision og hvad de står for. Disse angivelser skrives i stedet for *præcision*

i kommandoen. I MATLABs egen manual ("Reference Guide") under `fread` ses en udvidet liste af præcisionsangivleser. Nøjagtighed af tallene har også afgørende betydning for læse-funktionen.

- `'char'` Tegn, 8 bit. Det vil sige bogstaver eller andre tegn. Et ciffer vil også blive opfattet som et tegn og ikke som en værdi.
- `'short'` Heltal, 16 bit.
- `'long'` Heltal, 32 bit.
- `'float'` Flydende tal, 32 bit. En computer er ikke istand til korrekt at repræsentere reelle tal. I stedet er der indført en bestemt notation hvor et reelt tal repræsenteres ved eksponentiel notation. På den måde kan en del af de reelle tal repræsenteres med en vis nøjagtighed.
- `'intN'` Heltal, N bit. N kan være præcisionstallene 8, 16, 32 eller 64. Heltallet angives med fortegn. Det vil sige at en af bittene anvendes til at angive fortegnet.
- `'uintN'` Heltal, N bit. Igen kan N være præcisionstallene 8, 16, 32 eller 64. Her er tallet repræsenteret uden speciel fortegnsbit.

Læse-funktionen `fread` kan anvendes på to måder, der kan skrives som[1]

```
A = fread(fid);
[A, c] = fread(fid, size, 'præcision');
```

I den simple udgave forsøger `fread` at læse elementer fra filen med identifikator `fid` til en matrix A. Der er ikke sat en præcision for hvordan tallene er repræsenteret på filen, men default-præcisionen er `uchar`, det vil sige tegn uden fortegnsangivelser.

I den udvidede udgave, læses der stadigvæk fra filen med identifikator `fid`, men her præciseres det yderligere at der skal læses elementer ind i en matrix af størrelse `size`. Der er tre mulige `size` former:

- `n` Der læses n elementer ind i en søjlematrix.
- `Inf` Der læses til slut på filen. Elementer indsættes i en søjlematrix.
- `[m,n]` Der læses elementer indtil de fylder en m $\times$ n matrix. Matricen fyldes søjlevis. Hvis filen ikke indeholder elementer nok fyldes op med 0 (nul).

[1] Se også textread i MATLAB manualen.

Når `size` ikke specielt er angivet er defaultværdien `inf`. I den udvidede udgave præciseres yderligere på hvilken form tallene er repræsenteret på filen. `fread` returnere udover en matrix, en værdi der svarer til antallet af elementer, der med succes er læst ind. Denne værdi tildeles variablen c.

For begge funktioner `fread` og `fwrite` returneres således en kontrolværdi, der er et udtryk for, hvor godt transmissionen til og fra filen gik. Svarer antallet af læste elementer ikke med det man regnede med at læse ind, kan der være problemer. Måske er filen ikke så lang som man troede, muligvis er der fejl i filen så MATLAB ikke kan læse alle elementerne. Måske er tallene repræsenteret på en anden måde (i det tilfælde kan problemet løses ved at andre præcisionen og så forsøge at læse ind igen). Er der fejl i filen er det nødvendigt at gå tilbage til konstruktionen af filen og løse problemet ad den vej.

Eksempel

I det forrige eksempel blev filen `minfil.dat` åbnet med rettigheder til at skrive og læse fra filen. I dette eksempel ønskes at lægge en matrix B ud på filen, hvorefter dele af denne matrix vil blive læst ind igen. Dette er en øvelse og giver ikke i sig selv mening (at læse en matrix ud for derefter at læse den ind igen). B matricen skaffes ved hjælp af `magic` funktionen, der genererer tilfældige heltal til matricer. B sættes til en 7×7 matrix ved at skrive: `B = magic(7)`

For at holde øje med transmission tildeles variablen c kontrolværdien af `fwrite`-funktionen. B's elementer skal lagres som heltal repræsenteret ved 16 bit. På kommandolinien skrives nu:

```
c = fwrite(fid, B, 'short')
```

Her skrive MATLAB matricen B ud på filen og returnerer værdien c = 49. Ialt 49 elementer er læst ud på `minfil.dat`. Fordi filen er åbnet med rettigheder 'r+' til både at skrive og læse, kan man allerede nu læse ind igen fra filen. Ellers ville det være nødvendigt at lukke filen og åbne den igen med rettighederne sat til at læse. Der skrives nu:

```
[A, c] = fread(fid,9,'short')
```

I matricen A indlæses nu en søjlematrix på 9 elementer. Det forklares at formatet for filen er 'short'. Værdierne for variablene A og c bliver nu:

```
A =
    30
    38
    46
```

```
     5
    13
    21
    22
    39
    47

c =
    9
```

Herefter skal filen `minfil.dat` lukkes som beskrevet i det forrige eksempel.

□

Kapitel 6

Diverse

Enkelte vigtige ting vedrørende arbejde med MATLAB er endnu ikke berørt. I denne del vil små og mere løsrevne elementer blive taget op.

6.1 Udskrivning

MATLAB har kun få udskrift faciliteter. Direkte fra MATLAB er det kun muligt at udskrive de figurer, der konstrueres i figurvinduet. Udskrift af filer sker via den teksteditor, hvori filerne er oprettet. Det er ad omveje muligt at udskrive listen af kommandoer fra Commandvinduet, samt resultater af disse kommandoer, som brugeren har anvendt i arbejdet med MATLAB.

Udskrift af kommandoer

Det er muligt at gemme Commandvinduets indhold ved kommandoen `diary` efterfulgt af et filnavn (med endelse '.m'). Denne kommando lister alle de aktuelle kommandoer og resultaterne (dog ikke eventuelle figurer) i en fil, der så bagefter kan udskrives.

Udskrift af filer

Fra MATLABs default-editor Notepad er det muligt at udskrive indholdet af de filer, der er oprettet via editoren. Under **File** menupunktet vælges **Print** underpunktet og det åbne vindue, hvorfra **File** menuen er valgt, vil blive skrevet ud på den printer, der er sat som standard printer.

Udskrift af figurer

Blandt menupunkterne i det åbne figurvindue vælges **File** menuen. Herunder findes punktet **Print**. Når **Print** vælges udskrives figuren, der er plottet i det åbne vindue, på den printer, der er valgt som MATLABs default printer. Der vil fremkomme en dialog box, hvor der ønskes en bekræftigelse (klik på OK) på, at figuren skal udskrives. Bemærk, at hvis printeren ikke er en farveprinter, vil de forskellige farver, som er anvendt i figuren ikke fremgå.

Det er også muligt at udskrive fra kommandolinien. Ved af skrive `print` efterfulgt af et RETURN, vil det aktuelle figurvindue blive udskrevet på default printeren. Men denne kommando kan også anvendes hvis figuren skal sendes til en fil i stedet for en printer. I den forbindelse er der en lang række options eller flag der kan sættes, som fortæller MATLAB i hvilket format, figuren skal sendes. Hvis der eks. skrives `print -deps` *filnavn* sendes figuren til en `.eps` fil, som kan inkluderes i et LaTex dokumment. For øvrige options Se MATLABs manual "Reference Guide" under `print` for disse options.

Valg af printer

Dette kan gøres dels via Windows standard printer og har derfor intet at gøre med MATLAB, og dels lokalt ved at sætte MATLABs default printer. MATLAB sætter sin default printer efter Windows valget af standard printer. Men bemærk, at MATLAB sætter sin printer ved opstarten af MATLAB. Hvis standard printeren ændres efter at MATLAB er åbnet, vil denne ændring ikke ændre MATLABs valg af default printer. Derfor skal MATLAB genstartes hvis Windows standard printeren ændres.

Det er også muligt at ændre MATLABs standard printer lokalt og kun for MATLAB programmet. Dette gøres ved at vælge undermenupunktet **Printer Setup ...**, der findes under **File**. Her under er det muligt at bladre mellem de mulige printer og derved vælge en default printer. Dette valg vil blive husket næste gang, MATLAB opstartes.

Husk, at MATLAB ikke er en del af teksteditoren. En ændring af MATLABs default printer vil ikke ændre teksteditorens valg af printer.

6.2 Format af resultat

Indad til i MATLAB er alle tal repræsenteret med størst mulig nøjagtighed. Dette format kaldes `double` og er en (64 bit) lang flydende tal repræsentation. Denne repræsentation giver et stort antal af betydende cifre og bevirker, at de beregninger som foretages sker så nøjagtig som muligt. Det output som vises i Commandvinduet, efter at en kommando er blevet udført er af en andet format.

Tabel 6.1 Formater for fremvisning af resultat

KOMMANDO	TEKST	EKSEMPEL
`format short`	5 cifret	3.1416
`format long`	15 cifret	3.14159265368979
`format short e`	5 cifre exponent	3.1416e+00
`format long e`	16 cifret exponent	3.141592653689793e+00
`format hex`	hexadecimalt	400921fb54442d18
`format bank`	dollar og cent	3.14
`format rat`	brøk af små heltal	355/113

Default formatet kaldes `short` og er med 5 betydende cifre. Det er muligt at skifte til et andet format f.eks `long`, der har 15 betydende cifre. På kommandolinien skrives : `format long` og tast RETURN. I tabel 6.1 ses en samlet oversigt med repræsentation af tallet π som eksempel på formaterne.

Appendiks A

Sætninger til kontrol af udførelse

MATLABs kommandoer ligner for en stor del høj-niveau programmeringssprog som f.eks Pascal. I arbejdet med script-filer (se kapitel 3 om M-filer) har MATLABs kommandoer karakter af at fungere som programmeringsprog, idet kommandoerne bagefter kan udføres ved en såkaldt kørsel af filen (programmet). MATLAB tilbyder således på linie med andre høj-niveau sprog 3 stærke sætningstyper, der gør det muligt at kontrollere udførelsen af kommadoer.

A.1 If-sætninger

Denne type sætning anvendes, hvor man ønsker specielle kommandoer udført såfremt en given betingelse er opfyldt. Man kan formulere dette som en sætning der lyder:

```
HVIS betingelse
   kommando 1
ELLERS
   kommando 2
SLUT
```

Dette betyder at hvis en given betingelse er opfyldt skal kommando 1 udføres ellers (hvis betingelsen ikke er opfyldt) skal kommando 2 udføres. På intet tidspunkt vil både kommando 1 og 2 blive udført. De ord, der står med store bogstaver er vigtige nøgleord.

Eksempel

I stedet for ordet 'betingelse' kan der f.eks stå `a < 0` hvilket er et udtryk, der kan evalueres til sand eller falsk. Er denne betingelse opfyldt skal kommandoen

`a = a/b` udføres og hvis ikke skal kommandoen `a = b/a` udføres. Nøgleordene skrives på engelsk og i MATLAB kan der nu formuleres:

```
if a < 0
   a = a/b
else
   a = b/a
end
```

Det er vigtigt at afslutte med ordet `end`, fordi på kommando 2's (som på kommando 1's) plads kan der stå vilkårlig mange kommandoer og `end` indikerer, at her slutter hele if-sætningen. Linieskift og indryk er ikke nødvendigt. Man kunne også adskille udtryk og kommandoer med kommaer. MATLAB vil således også kunne forstå sætningen udformet som:

```
if a < 0 , a = a/b, else , a = b/a , end
```

□

Ved mere omfattende sætninger øger det dog læsbarheden væsentligt med linieskift og indryk, hvilket er en meget anvendt måde at opskrive programkode på.

Bemærk, at der i stedet for `kommando` kan komme endnu en if-sætning. Der behøver ikke komme kommandoer efter `else` og i det tilfælde skrives blot `end` i stedet for `else`. Hvis betingelsen ikke er opfyldt udføres således ingen kommandoer og MATLAB går videre med de kommandoer, der eventuelt måtte stå efter `end`.

Eksempel

I MATLAB er der indbygget en speciel udgav af `else`, der kan anvendes hvor flere if-sætninger anvendes indeni hinanden. F.eks:

```
if A
   x = a
else
   if B
      x = b
   else
      if C
         x = c
      else
         x = d
      end
   end
end
```

Det kan være ret forvirrende at følge med i, hvor en if-sætning starter og slutter helt præcist. Variablen x bliver tildelt en værdi afhængig af tre betingelser A, B og C. Mere overskueligt kan det nu skrives:

```
if A
  x = a
elseif B
  x = b
elseif C
  x = c
else
  x = d
end
```

□

A.2 For-løkker

Nogle gange vides det på forhånd, at en given kommando (eller flere) skal udføres et fast antal gange. Man kan da benytte en tællevariabel, der tæller det antal gange løkken af kommandoer er blevet udført. Lad variabel `i` være en tællevariabel. Man kan nu formulere sætningen:

```
FOR i lig med a og op til b
   kommando
SLUT
```

Det er her indforstået at variabel `i` bliver talt en op hvergang løkken er løbet igennem. Ordene med stort er nøgleord. Oversættes de til engelsk vil MATLAB genkende sætningen på netop disse ord.

Eksempel

Det ønskes at nulstille vektoren x. Denne har 10 kompenenter og der kan i MATLAB nu formuleres:

```
for i = 1:10
   x(i) = 0
end
```

Ordet `end` fortæller MATLAB, at her slutter for-løkken. Det er underforstået, at `i` tælles en op (`i = i + 1`) hver gang kommandoen er blevet udført. Efter et antal gennemløb tildeles `i` værdien 10, hvorefter MATLAB går videre til de kommandoer, der eventuelt står efter ordet `end`. Som for if-sætninger er linieskift og indryk ikke nødvendigt og MATLAB vil også forstå sætningen skrevet ved hjælp af kommaer:

```
for i = 1:10 , x(i) = 0, end
```

□

Det er selvfølgelig tilladt at anvende et andet variabelnavn end `i` for tællevariablen og det er også muligt at kontrollere denne, således at der i stedet for tælles en op hver gang, bliver talt f.eks 10 ned hver gang.

Eksempel

Hvis y har værdien 1000 kunne der f.eks skrives:

```
for j = 100:-10:10
   x = y/j
   y = x + j
end
```

Efter denne løkke vil de variable have værdierne `y = 12.1569, j = 10` og `x = 2.1569`.

□

A.3 While-løkker

Den sidste sætningstype er også en løkke. Løkken udføres sålænge et givet udtryk er sandt. Antal gennemløb af løkken kendes ikke på forhånd, men når en given betingelse ændrer tilstand er det tegn på at løkke skal stoppe. Det vil sige at undervejs i løkken må der ske et eller andet med de variable der indgår i betingelsen, ellers vil løkken aldrig stoppe. Man kan formulere sætningen som:

```
SÅLÆNGE betingelse
   kommando
SLUT
```

Nøgleordene der står med stort her oversættes til engelsk og er reserverede ord, som MATLAB anvender til at genkende sætninger af denne type. Der kan komme vilkårlig mange kommandoer inde i while-løkken, f.eks andre løkker.

Eksempel

Et eksempel på en kort while-løkke: Først sættes f.eks variblene `a` og `b` til henholdsvis 10 og 5, derefter følger while-løkken:

```
a = 10
b = 5
while b < a
   a = a - 1
end
```

Det er let at overskue, at løkken her vil løbe 5 gange. Bemærk hvorledes variablen `a` ændres, således at betingelsen ender op med at blive falsk og løkken standser. Som for de andre sætningstyper er indryk og linieskift en hjælp til en overskuelig opsætning, men MATLAB til også kunne forstå:

```
a = 10, b = 5
while b < a, a = a + 1, end
```

Det burde fremgå af dette eksempel, at det er let at konstruere while-løkker, der aldrig standser. Man kan dog afbryde den uendelige løkke med `Ctrl c`

□

Litteratur

[1] MATLAB, Reference Guide.
The Mathworks inc, 1992.

[2] MATLAB, User's Guide.
The Mathworks inc, 1993.

[3] EDWARDS, C. H., PENNY, D. E.
Calculus and analytic geometry.
Prentice-Hall, 1982.

Stikord