Matlab - en kort handledning

Matematiska Institutionen, Chalmers & GU *

Innehåll

*Synpunkter och förslag kan skickas till vilhelm vid chalmers pynkt se (här strös alltså ett villospår för 'sökmaskiner'!)

1 Introduktion

Denna lilla skrift om matlabanvändande är mycket kortfattad och elementär; för mer information kan man konsultera http://www.math.chalmers.se/∼jorgen/texter/matlabHandl.pdf eller *MATLAB-beräkningar inom teknik och naturvetenskap*, Per Jönsson, Studentlitteratur eller *MATLAB for Engineers*, Holly Moore, Pearsson Education.
Matlab är en förkortning för *Mat*rix *lab*oratory och är ett datorprogram framtaget historiskt i första hand för att lösa (stora) ekvationssystem, som enklast formuleras m h a matriser. Det betyder bl a att alla de vanliga standardoperationerna som t ex gånger, $*$, anses, om inget annat sägs, verka på matriser och är alltså *matrismultiplikation*. Detta är ngt som ni kommer att läsa i lp III i kursen *Linjär algebra*. Vi kommer här att använda oss av olika matriser (eller *teckenscheman**talscheman*); t ex radmatriser (eller $1 \times n$-matriser)

$$(a_1 \ a_2 \ \dots \ a_n)$$

kolonnmatriser (eller $m \times 1$-matriser)

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{pmatrix}$$

och $m \times n$-matriser

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Speciellt kan reella tal ses som 1×1-matriser. Elementet a_{ij} står i rad i och kolonn j och denna position kallas ofta *fack* eller plats ij. En matris består alltså av olika *element*, oftast, men inte nödvändigtvis, reella eller komplexa tal i de olika facken i matrisen. Om det är reella eller komplexa tal kan man addera, multiplicera etc matriserna; utföra matrisoperationer.
Matlab fungerar bl a ungefär som en enkel 'räknare' så att kommandona för addition etc är $+ \ - \ * \ /$ och $\hat{}$ som vanligt. Dessa är alltså matriskommandon. Motsvarande *fackvisa* kommandon är desamma men föregås av en punkt: $.*$ $./$ och $.\hat{}$ (addition, subtraktion och multiplikation med ett tal är även för matriser fackvisa operationer och behöver alltså inte föregås av en punkt).
En (riktig) matrismultiplikation, $*$, av två matriser A och B av typ $m \times n$ respektive $k \times \ell$ kräver för att vara definierad att $n = k$. För fackvis multiplikation krävs ju naturligtvis att $m = k$ ock $n = \ell$; alltså att A och B är av samma typ.

Exempel 1 $(1\ 2\ 3) * (4\ 5\ 6)$ *(matrismultiplikation; är ej definierad).*

Exempel 2 $(1\ 2\ 3).*(4\ 5\ 6) = (4\ 10\ 18)$ *(fackvis multiplikation).*

Exempel 3 $(1\ 2\ 3) * \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} = 4 + 10 + 18 = 32$ *(matrismultiplikation).*

Exempel 4 $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 4 & 5 & 3 \\ 3 & 4 & 5 \\ 5 & 3 & 4 \end{pmatrix}$ *(matrismultiplikation).*

Exempel 5 *Som 'vanligt' ger även i matlab kommandona* $5+2$, $5*2$, *4^2,* $\cos(pi)$ *och sqrt(4) resultaten* 7, 10, *16, -1, respektive 2*

Matriserna i de föregående exemplen skrivs av matlab utan paranteser; och kommandon för att ange en matris är:

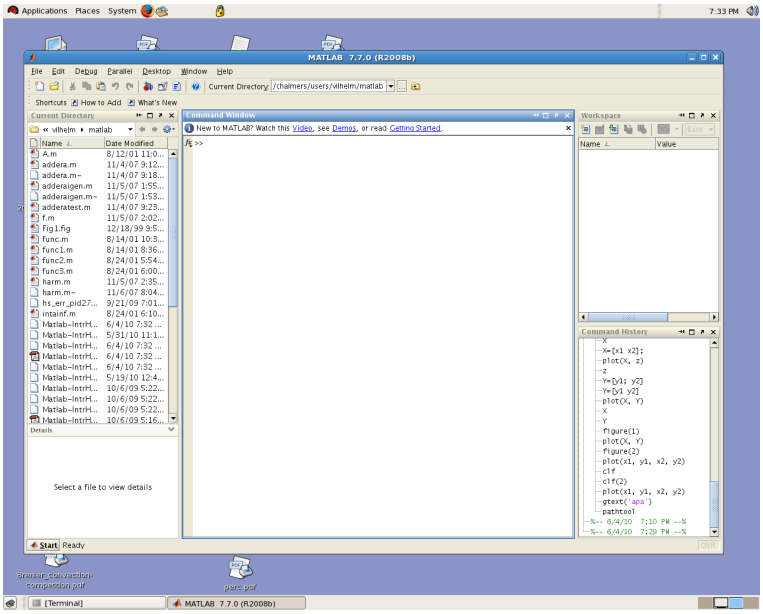
Exempel 6 $A = [1\ 2\ 3;\ 4\ 5\ 6;\ 7\ 8\ 9]$ *som ger* $A = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{matrix}$ *; rader separeras alltså av ; och fack av 'space'.*

2 Matlabmiljön

Efter dessa inledande steg startar vi vårt *matlab*-användande.

I fortsättningen kommer vi låta >> betyda prompten på kommandoraden i matlab och ↩ betyda 'Enter'. Vidare använder vi konventionen att vinkelparanteser markerar sådant som ska anges som input av matlab-användaren och inte ska anges i det faktiska matlabkommandot; t ex help <...> där användaren kan ange t ex cos, för att få upplysningar och hjälp med kommandot cosinus.

Det första man ska göra är att skapa sig ett bibliotek, directory, lämpligen kallat matlab, med kommandot (innan man startat matlab) mkdir matlab. Sedan går man ned i detta directory och här startar man nu matlab med just kommandot matlab. Vi kommer här anta att vi använder version 7.5.0 av matlab. Vi får då upp följande matlabfönster:



Matlabfönstret består av olika delar: *Command Window*, *Command History*, *Work Space* och *Current Directory* är dess huvuddelar. Vidare finns Stänga fönstret-knappen (× i övre högra hörnet), Start-knappen (i nedre

vänstra hörnet) för start av diverse olika processor såsom Editor, Toolboxes (Verktygslådor) och andra hjälpverktyg. Dessutom finns som på en vanlig 'hemsida' diverse 'rullmenyer' längst upp där man kan komma åt vanliga kommandon och funktioner och även funktioner som kan 'kommas åt' på andra sätt (som t ex Editor). Observera att Current Directory finns på två ställen; i adressfönstret och i ett fönster som delas med Work Space (man byter i det fönstret mellan Work Space och Current Directory genom en 'rullmeny' ovanför fönstret) och där innehållet i Current Directory visas.
Dessutom finns (bl a) document windows, graphic windows och editing windows som kan öppnas (eller öppnas automatiskt) vid behov.
Innan vi går vidare tittar vi på ett par enkla sätt att få hjälp i matlab. Det finns oxå externa hjälpfunktioner som man kan utnyttja om behov uppstår.

1. Ge i Command Window kommandot help ger en lista med hjälp-rubriker där man kan klicka sig vidare; t ex så kan i början av matlab-användandet matlab/elmat, matlab/elfun och matlab/funfun (för elementary matrices, elementary functions respektive function functions) vara till nytta.

   Exempel 7 >> *help tan* *för att få hjälp med tangensfunktionen.*

2. Öppna help-fönstret genom att i rullmenyn Help ange Product Help (eller ev annan rubrik) eller använd F1-knappen, för samma resultat. Häe finns (längst upp) de två rubrikerna Functions: By Category, Alphabetical list vilka brukar vara mycket användbara.

   Man kan oxå nå denna typ av hjälp direct från Command Window genom kommandot *doc*:

   Exempel 8 >> *doc tan*

Innehållet i hjälpen för dessa två hjälpvarianter är (för samma ämne; *tan* i detta fall) olika så att gå till 'den andra varianten' kan löna sig.

Exempel 9 *Säg att du vill hitta ngt om avrundning av decimaltal:*

1. *Öppna hjälpfönstret med Help i rullmenyn och där Product Help; eller använd F1-knappen*
2. *välj Functions By category*
3. *öppna länken Mathematics*
4. *öppna länken Elementary Math*

och där ser man en rubrik som innehåller Rounding; som man alltså klickar på. Man kan oxå pröva Functions: Alphabetical list.

3 Matlabfönstren

Vi kommenterar helt kort de olika fönstren.

3.1 Command Window

Kommandon i Command Window 'läses tillbaka' till fönstret (så att användaren ser och kan kontrollera riktigheten i givna kommandon) som default; om man avslutar kommandot med ; så läses inte kommandot tillbaka. Man kan upprepa kommandon genom att använda ↑ och ↓ pilarna på tangentbordet eller genom att kopiera (klicka-dra) från Command History-fönstret. Command Window kan rensas med *clc*; som bara rensar detta fönster och ej andra t ex

3.2 Workspace Window

Här finns alla variabler som matlabkörningen har använt/använder, och dessas egenskaper. Detta fönster (och Command Window) rensas med *clear*.

3.3 Current Directory Window

innehåller alla filer i Current Directory. Katalogen kan ändras i rullmenyn eller browse-knappen.

3.4 Document Window

Olika slags Document Windows kan öppnas för olika applikationer och startas t ex genom att påbörja applikationen.
Så startas t ex genom att dubbelklicka på variabelnamnen i Workspace, en editor, *array editor*, där man kan editera variabeln; lägga till rader och/eller kolumner eller ändra värden i de olika 'facken'. Alternativt kan man klicka på/markera en variabel och klicka på ikonen för array editor i rubrikmenyn för fönstret (Workspece). I rubrikmenyn kan man oxå skapa en ny variabel genom symbolen *New variabel* (längst till vänster) i rubrikmenyn för fönstret.
Graphics window, graf-fönster, startas automatiskt vid användande av kommandot *plot*, men ett graf-fönster kan oxå startas genom huvud-rullmenyn eller genom graf/plot-ikonen i rubrikmenyn för Workspace window.
En editor öppnas t ex genom att i huvud-rullmenyn öppna File och klicka på New och där starta *M-file*.

3.5 Allmänt om fönster

Observera att huvudrullmenyn som presenteras beror på vilket som är aktivt (vilket fönster man 'står' i).
Vidare kan man, bl a för att skapa plats, 'lägga undan' till höger- eller vänster-marginalen ett fönster man inte just för tillfället använder, genom att klicka på symbolen $|\leftarrow$ eller $\rightarrow|$ (ungefär) i det fönstrets rubrikmeny.
Genom kommandona $\nearrow$ och $\searrow$ (ungefär) i rubrikmenyerna för de olika fönstren kan dessa frigöras respektive återanslutas (dockas) till matlabs huvudfönster. De olika fönstrens position i matlabs huvudfönster kan ändras genom att 'klicka och dra' till lämplig/möjlig position som presenteras genom 'siluetter/gränsmarkeringar'.
De olika fönstren kan oxå stängas var för sig; som vanligt med stängsymbolen $\times$ längst upp till höger. Ursprungskonfigurationen kan återfås genom att i huvudrullmenyn konsekutivt välja Desktop, Desktop Layout och Default.

3.6 Startknappen

Längst ner till vänster ger denna knapp en alternativ åtkomst för många av funktionerna nämnda ovan. Dessutom nås här t ex Matlabs hemsida och andra Internetprodukter, Demos och Toolboxar ('verktygslådor') för ytterligare funktionalitet och specifika applikationer.

4 Grundläggande Matlab-kommandon

1. x=x+1 är som vanligt att förstå som ett tilldelningskommando; x tilldelas det nya värdet x+1.

2. Variabelnamn måste starta med en bokstav och kan ha vilken längd som helst; men bara de 63 första karaktärerna räknas.

3. Kommandot

   >> isvarname <variabelnam>

   testar om ett tilltänkt variabelnamn är tillåtet; svar 1 betyder OK och 0 betyder otillåtet.

4. Vissa namn är reserverade och används av matlab:

   >> iskeyword

   listar de reserverade namnen. Man skulle kanske tro att vanliga funktionsnamn som t ex sin är reserverade namn, men så är inte fallet. Funktionen sin kan alltså omdefinieras:

   >> sin = 4

   gör sin till en variabel/konstant med värdet 4. För att kontrollera status kan man använda *which*:

   >> which sin

   Ett omdefinierat namn kan återställas med

   >> clear sin

5. Matlab läser inte tomrum, (space), så långa uttryck kan göras mer läsbara genom att lägga in extra tomrum.

5 Ytterligare lite om matris-hantering

Den grundläggande datatypen i matlab är matris (eng: matrix). Den enklaste matristypen 1×1-matriser; detsamma som tal.
Vi nämnde ovan hur man lägger in matriser i matlab; t ex

>> A=[1 2 3 4; 4 5 6 7; 7 8 9 10] ↩

ger från matlab utskriften

$$A = \begin{matrix} 1 & 2 & 3 & 4 \\ 4 & 5 & 6 & 7 \\ 7 & 8 & 9 & 10 \end{matrix}$$

(och avslutas kommandot med ; så ger, som nämnts ovan, alltså matlab ingen utskrift).
För att lägga till fler fack i en matris eller ändra värdena i facken o dyl så är det smidigt att använda *array editor*, som startas genom att dubbelklicka (snabbt!) på ngt variabelnamn i workspace eller new variable ikonen i workspace rubrik/huvud.
Kolonnvektorer/matriser

$$x = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

läggs in som
>> x=[1; 2; 3; 4]
eller
>> y=[1 2 3 4]
>> x=y'
Det finns oxå flera kommandon för att lägga in vissa enkla typer av matriser/vektorer. T ex ger
>> b=1:5
eller
>> b=[1:5]
utskriften $b = [1\ 2\ 3\ 4\ 5]$ som alltså är en vektor med tal från 1 till 5 med steglängd 1. Annan steglängd fås med
>> c=[1:2:5]
som ger $c = [1\ 3\ 5]$ och
>> c=[1:3:5]

ger $c = [1\ 4]$. Kommandot *linspace* ger en indelning av ett talintervall i ett valt antal punkter:
>> d=linspace(1,10,3)
ger $d = 1.0000\ 5.5000\ 10.0000$
Motsvarande kommando med 'logaritmiskt spridda' tal är *logspace*.

6 Format

Matlab använder sig av olika utskriftsformat. Grundformerna är *format short* (fyra decimaler) och *format long* (fjorton decimaler). Vidare finns *format bank* med två decimaler och (endast) realdelen för komplexa tal. Dessutom ges utskriften från kommandot *clock* en enklare form; ngt som oxå kan fås med *fix(clock)*. Ytterligare format är *format short e*, *format long e* för s k scientific notation som skriver tal 10-potenser (t ex är 6.022e23 matlabs sätt att skriva $6{,}022 \cdot 10^{23}$), *format rat* (skriver som rationella tal), *format long eng* och *format short eng* använder engineering notation som är som scientific notation men kräver att 10-potenser är en multipel av tre motsvarande millimeter, micrometer, nanometer, picometer etc. Matlab kan själv välja det format som passar utskriften bäst med kommandot *format short g* respektive *format long g*. Formatet *format +* anger för en matris vilka fack som är positiva/innehåller positiva tal, vilka som är negativa och vilka som är noll.

Observera att t ex e^3 skrivs som exp(3) i matlab.

7 Spara arbete

Kommandot *diary* ger möjlighet att spara/spela in en hel matlab-körning och återanvända den senare. Ge kommandot diary igen eller diary off avslutar 'inspelningen' och då sparas en fil med namnet diary. Denna fil öppnas i en editor genom att dubbelklicka på filnamnet eller genom att starta en editor och där öppna filen. Om man vill använda diary igen så läggs senare matlab-pass till efter tidigare i filen diary. För att skilja passen åt kan man spara med >> diary <filnamn>

eller
>> diary('<filnamn>')

För att spara de variabler som finns i workspace (som en binär MAT-fil, med default-namn matlab.mat) används
>> save <filnamn>
och för att ladda in variablerna igen används
>> load <filnamn>
Det finns möjlighet att spara som ascii-fil och även i 'sexton'-format, genom flaggorna *-ascii* respektive *-double*. Ascii används som standard vid utbyte av data mellan olika program/plattformar; så detta är formatet att använda vid fildelning och då ska filextensionen vara .dat eller .txt istället för .mat.

Exempel 10 >> *save <variabellista.dat> -ascii -double*

Kommandot load återskapar data från nuvarande directory; rullmenyn går ju som vanligt oxå att använda för att återskapa data från andra directories.

8 M-filer

En script M-fil är en lista med matlabkommandon som kan köras genom att trycka på knappen Run and Save i Editor-fönstret.

Man kan se vilka M-filer och MAT-filer som finns i current directory genom kommandot *what* i kommandofönstret.
Det är praktiskt att arbeta med en uppgift/ett projekt och spara listan med kommandon i projektet som en M-fil för att fortsätta vid ett senare tillfälle.
Man kan köra en delmängd av kommandona i en delfil genom att markera dessa och sedan högerklicka på Evaluate section.
Det finns oså s k Funktionsfiler som är M-filer och de är ett sätt att skapa egna användardefinierade funktioner. Mer om det i en rubrik nedan.
Dessförinnan studerar vi:

9 Inbyggda funktionsfiler i matlab

Många av de inbyggda funktionerna och/eller funktionsnamnen är desamma som återfinns i t ex C-programspråk, Fortran och Java.
Vi kommer lista ett antal av dessa inbyggda funktioner nedan (utan speciellt mycket kommentarer) för att bara upplysa om att de finns; vidare info fås via help-funktioner, t ex helt enkelt bara

>> help <funktionsnamn>

9.1 Vanliga matematikfunktioner

1. *sqrt a* där a är en vektor, beräknar roten fackvis för elementen i a
2. *rem(x,y)* beräknar resten vid division av x med y
3. *size(d)* där d är en vektor/matris ger typen av d. Ex: [rows, cols]=size(d)
4. *abs(x)*
5. *nthroot(x,n)* Ex: nthroot$(-2, 3)$ ger $(-2)^{1/3}$
6. *sign(x)*
7. *rem(x,y)* Ex: rem(25,4) ger 1
8. *exp(x)*
9. *log(x)*
10. *log10(x)*

9.2 Avrundningsfunktioner

1. *round(x)*
2. *fix(x)* avrundar nedåt
3. *floor(x)*
4. *ceil(x)* Ex: ceil(-8.6) ger -8

9.3 Trigonometriska funktioner

1. sin(x), cos(x), tan(x), asin(x), sinh(x), asinh(x), sind(x), (x anges i grader), asind(x).

9.4 Diskret matematik

1. *factorial(x)*
2. *factor(x)*
3. *gcd(x,y)*
4. *lcm(x,y)* Ex: lcm(2,5); ans = 10
5. *rats(x)* Ex: rats(1.5) ger 3/2
6. *nchoosek(n,k)*
7. *primes(x)* anger primtal mindre än x
8. *isprime(x)* anger om x är ett primtal
9. *max(x)* största värdet i vektorn x eller i de oliks kolonnerna i matris x
10. *[a,b]=max(x)* maximum och position för maximum för vektor eller kolonner i matris
11. *max(x,y)* finner för två matriser av samma typ, största värdet fackvis
12. *min(x), [a,b]=min(x), min(x,y)*
13. *mean(x)* medelvärdet för vektor eller kolonnvis i matris
14. *median(x)* medianvärdet för vektor eller kolonnvis i matris
15. *mode(x)* vanligaste värdet i x
16. *sum(x)* summerar elementen i x/kolonnvis för matris x
17. *prod(x)*
18. *cumsum(x)* kumulativ summa
19. *cumprod(x)*
20. *sort(x), sort(x, 'descend')*
21. *sortrows(x)* sorterar hela raden baserat på första kolonn
22. *sortrows(x, k)* sorterat baserat på kolonn k i matrisen
23. *sortrows(x, -k)* i avtagande ordning

9.5 Matrisstorlek

1. *size(x)*
2. *[a,b]=size(x)* a är antalet rader, b är antalet kolonner
3. *length(x)* största dimensionen av x

9.6 Statistik-funktioner

1. *std(x)* standard deviation ('kolonnernas')
2. *var(x)* variansen

9.7 Slumptal

1. *rand(n)* $n \times n$-matris med fack innehållande slumptal mellan 0 och 1
2. *rand(m,n)* $m \times n$-matris
3. *randn(n)* $n \times n$-matris med normalfördelade slumptal
4. *randn(m,n)* $m \times n$-matris

9.8 Komplexa tal

1. *abs(x)*
2. *angle(x)* ges i radianer
3. *complex(x,y)* kan oxå anges som x+iy
4. *real(x)* ger realdelen
5. *imag(x)* ger imaginärdelen
6. *isreal(x)*
7. *conj*

9.9 Speciella funktioner

pi; i; j; Inf (kalkyl utöver kapacitet eller division med 0); NaN (för en odefinierad beräkning); clock (aktuell tid, fix(clock) ger lättare utläst resultat); date.

10 Matriser

Följande är valida sätt att skapa matriser:

>> C=[-1 0 1; 1 1 0; 1 -1 0; 0 0 -1]

eller

```
>> C=[-1 0 1
1 1 0
1 -1 0
0 0 -1]
```

Långa matriser som behöver delas upp över flera rader skapas med:

>> F=[1 2 3 4 5 ... 6 7 8 9 10]

Matriser kan användas för att 'bygga' större matriser:

```
>> B=[1 2]
>> S=[3 B]
```

ger att

S=
3 1 2

och

>> T=[1 2 3; S]

ger

T=
1 2 3
3 1 2
Man kan oxå 'indicera in i en matris':
>> S(2)=-1

ger

S=
3 -1 2

och

>> S(7)=9 ger

S=
3 -1 2 0 0 0 9

där alltså icke-definierade fack i matrisen sätts till 0.
Vidare kan man ange matriser genom att ange en bestämd steglängd; default är steglängd 1:

>> H=1:8

ger

H=
1 2 3 4 5 6 7 8

Annan steglängd anges genom
>> J=0:0.5:2

som ger

J=
0.0 0.5 1.0 1.5 2.0

Låt

>> M=[1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7]

Då ger

>> X=M(:,1)

första kolonnen i M:

X=
1
2
3

Analogt ger

>> M(:,4)

fjärde kolonnen,

>> M(1,:)

första raden och

>> M(2:3,:)

ger rad 2 och 3. Vidare ger

>> M=(2:3,4:5)

elementen i rad 2 och 3 och kolonn 4 och 5; dvs

$$\text{ans}= \begin{matrix} 5 & 6 \\ 6 & 7 \end{matrix}$$

Kommandot >> M(2,3) ger elementet i rad 2, kolonn 3, dvs

ans=
4

Kommandot >> M(2,3)=5 ändrar detta element i M till 5.

Slutligen ger kommandot >> M(:) alla kolonnerna efter varandra i en lång kolonn; dvs

$$\text{ans}= \begin{matrix} 1 \\ 2 \\ 3 \\ 2 \\ 3 \\ 4 \\ 3 \\ \vdots \end{matrix}$$

och t ex >> M(10) ger ans = 4.
Kommandot end är av värde då man inte vet dimensionen på matrisen. T ex så ger >> M(1, end) svaret ans = 5; och >> M(end, end) svaret ans = 7 och det gör oxå >> M(end).

10.1 Speciella matriser

Kommandot >> zeros(m,n) ger en $m \times n$-matris med nollor i alla facken. På samma sätt ger kommandona >> zeros(m),>> ones(m,n) och >> ones(m) de naturliga motsvarigheterna.
Kommandot >> diag(A) extraherar för en kvadratisk matris A diagonalen och för en vektor A skapar det en matris med A som diagonal. Den extra parametern k, positivt/negativt heltal, i >> diag(A,k) gör samma för diagonal parallell med huvuddiagonalen, flyttad uppåt eller nedåt k steg.

Exempel 11 *Radvektorn A=[1 2 3]; ger att kommandot*

>> *diag(B,1)*

resulterar i

$$ans = \begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

Ytterligare specialmatriser ges med kommandona >> magic(m) (en $m \times m$-matris där summan av elementen i rader är densamma och även för kolonner och diagonaler), >> fliplr (speglar matrisen från höger till vänster), >> flipud (speglar upp mot ner).

11 Plottning

Låt x och y vara vektorer av samma längd, eller dimension. Kommandot >> plot(x,y) ritar kurvan genom punkterna (x_i, y_i) i xy-planet.
Efter plot-kommandot kan diverse info (den måste läggas till **efter** kommandot) läggas till den ritade grafen; t ex

>> plot(x,y), title('Laborationsexperiment 1'), xlabel('Tid, sek'), ylabel('Distans, meter'), grid

Observera att text anges inom 'fnuttar'; behöver man skriva apostrof i texten anges denna genom två enkelapostrofer, ''. Grid ger ett rutnät i koordinatsystemet.
Vid användande av plot-kommandot ritas grafen i ett nytt grafikfönster; default-grafikfönstret. Genom kommandon >> figure(n) kan man generera flera grafikfönster. Alla grafer ritas i det senast öppnade fönstret; något som alltså även väljs med figure(n). Vid ritandet av en graf raderas tidigare ritade grafer. Genom kommandot >> pause eller >> pause(n) kan man fördröja ritandet respektive fördröja i n sekunder. Man kan behålla/inte behålla tidigare ritade grafer i grafik-fönstret genom >> hold on/off.
Låt

```
>> x=0:pi/100:2*pi;
>> y1=cos(x*4);
>> y2=sin(x);
```

och

```
>> Y=[y1;y2];
```

Då ger >> plot(x,Y) grafen för y1 och y2 i samma fönster och med olika färg. Kommandot >> plot(x, y1, x, y2) ger samma graf med eventuellt olika färg om man dessutom anger det. Det finns olika plot-tecken och olika färger, t ex -,:,¨,.,o,x,+,* och b (blue), g (green), r (red), y (yellow), k (black). För användande och fler möjligheter; se

help plot

Skala på axlarna ändras med
>> axis([xmin, xmax, ymin, ymax])
och
>> legend('line1', 'line2', 'line3')
ger beskrivning av vilken graf som är vilken. Kommandot
>> text(xkoordinat, ykoordinat, 'string') lägger till texten *string* med början vid (xkoordinat, ykoordinat).
Kommandot >> clf raderar innehållet i ett figurfönster; och
>> close stänger figurfönstret.
Kommandot >> gtext('text') är likanande som kommandot text men istället för att ange koordinaterna för positionen där angiven text ska placeras, ges ett 'sikte' som framträder på skärmen och med vars hjälp man genom att klicka kan positionera angiven text.
Delfönster skapas med >> subplot(m, n, p) där p anger position i en $m \times n$-matris av delfönster, av det delfönster det kommer skrivas i härnäst, räknat radvis åt höger.
Andra sätt att presentera grafer ges t ex med >> bar(x) (ger stapeldiagram), >> pie(x) (ger cirkel- eller tårtdiagram), >> hist(x) (ger histogram). Det finns ett antal andra kommandon för andra sätt att presentera data. Se help-funktionen.
Funktionsgrafer kan oxå plottas direkt m h a, t ex kommandot

>> fplot('sin(x)', [-2*pi, 2*pi])

I detta kommando kan oxå *anonyma funktioner* (se avsnitt nedan) användas och då anges den anonyma funktionens funtionshandtag ('handle') som funktionsnamn istället (för sin(x) i exemplet ovan).
Vidare ger >> plot3(x,y,z) en tredimensionell plot,>> surf(x,y,z) ger en tvådimensionell funktionsyta i $\mathbf{R}^3$ och det snarlika >> surfc(x,y,z) ger en plot av konturen för grafen surf(x,y,z).
Plottar kan sparas med *Save As* i fil-menyn.

12 Funktioner definierade m h a M-filer

I en M-fil måste första raden börja med en funktionsdefinition som

1. som börjar med ordet *function*

2. en variabel som är funktionsresultatet; är i värdemängden

3. ett funktionsnamn

4. en variabel som används som det som funktionen opererar på; är i definitionsmängden.

Dessutom måste funktionsnamnet börja med en bokstav och får bara innehålla bokstäver, siffror och 'underscore'. Namn reserverade av Matlab får ej användas.
Kommentarer som följer omedelbart efter den första raden är det som ges som information vid help-förfrågan för funktionen.

Exempel 12 *function result = poly(x)*
% This function calculates the
% value of a third-order polynomial.
*output = 3*x.∧+5*x.∧2-2*x+1;*

Obs: Innan funktionen kan användas måste filen sparas i 'current directory' med samma filnamn som funktionsnamnet.

Om y är en vektor så blir poly(y) oxå motsvarande vektor.

Funktioner kan ha flera 'inputs' och 'outputs'.

Exempel 13 (dist, vel, accel) *= motion(t,w)*
% Denna funktion beräknar sträcka, hastighet, och acceleration
% vid tiden t för en bil med vikten w.
*accel = w.*t;*
*vel = accel.*t;*
*dist = vel.*t;*

13 Lokala variabler

Variablerna som används i en funktionsfil är 'inre' variabler bara åtkomliga för funktionsprogrammet självt och ej åtkomligt från workspace. Omvänt är det enda sättet ett funktionsprogram kan kommunicera med workspace, via input och output.

Exempel 14 *function result = distance(t)*
% This function calculates the distance a falling object
% travels due to gravity.
g=9.8 % meteres per second squared
*result = (1/2)*g*t.∧2;*

Observera att värdet på g måste ges inne i funktionsprogrammet; eller ges ges till programmet som en input som i följande exempel:

Exempel 15 *function result = distance(g,t)*
% This function calculates the distance a falling object
% travels due to gravity.
g=9.8 % meteres per second squared
*result = (1/2)*g*t.∧2;*

Dessutom kan man på ytterligare ett sätt ange g, m h a s k

14 Globala variabler

Globala variabler är tillgängliga för alla delar av ett datorprogram; så generellt är det inte en bra idé att låta storheter vara globala för de kan ju då ändras av ngn del av programmet utan att andra delar vet det.

Exempel 16 *Samma funktionsfil igen men nu med en global variabel som vi kallar G.*
function result = distance(t)
% This function calculates the distance a falling object
% travels due to gravity.
global G % meteres per second squared
*result = (1/2)*G*t.∧2;*

Man måste ju då innan man kör funktionsfilen, definiera G i 'command window':

>> global G
>> G=9.8;

Då kan man oxå enkelt ändra värdet på G, direkt i 'command window'.

Om man istället vill behålla en variabel som lokal för ett program och behöver ändra dess värde i programmet, så kan man lätt få upp M-filen med

>> type<fil-namn>

eller

>> type('filnamn')

som visar filen i 'command window'.

15 Vektygslåda med funktioner

Default för en funktionsfil som sparas är att den sparas i current directory.
Om man vill spara till kommande Matlab-körningar så kan man skapa ett directory, en mapp, kallad t ex My-functions.
M h a kommandot i rullmenyn,

File → Set Path

eller kommandot

>> pathtool

i command window, kan ordningen hur matlab söker filer att använda, ändras. Ny sökvägar kan oxå adderas till sökrutinen; se

help addpath

16 Anonyma funktioner och funktionshantag

Om man i en körning eller i ett program, bara mer tillfälligt vill använda en funktion, kan man enkelt 'skriva in' den utan en funktionsfil, som en anonym funktion, genom att använda 'snabela'-kommandot.

Exempel 17 *ln=@(x) log(x)*
ger för kommandot
ln(10)
resultatet
ans = 2.3026

Denna funktion ln försvinner när workspace raderas. Vill man ha tillgång till den även senare, kan man precis som med variabler etc, spara denna anonyma funktion som en .mat-fil

>> save my-ln-function ln

och sedan få den 'inladdad' igen med kommandot

>> load my-ln-function

Man kan tänka på anonyma funktioner angivna med @, som smeknamn för 'riktigt' angivna funktioner.

17 Funktionsfunktioner

Vissa funktioner, inbyggda eller definierade av av användaren, tar som argument, andra funktioner; t ex fplot. Om man t ex vill plotta log(x) på intervallet [0.1, 10] så kan det göras genom att använda 'fnuttar' som i

>> fplot('log(x)', [0.1, 10])

eller oxå genom att använda vår i förra avsnittet egendefinierade funktion ln, anonym eller sparad, genom

>> fplot(ln, [0.1, 10])

Man ser då en poäng med att använda anonyma funktioner, funktioner som man då slipper skriva in i andra funktionskommandon: t ex

>> poly5=@(x) -5*x.^5+400*x.^4+3*x.^3+20'x.^2-x+5;

som ju mycket enklare plottas med

>> fplot(poly5,[-30, 90])

än med 'fnuttar' som i det första kommandot ovan.

Många frågeställningar **om** funktioner kan besvaras av program som är funktionsfunktioner, dvs tar funktioner som input: t ex

fzero

som har bl a en funktion som input och ger en grov uppskattning av ett nollställe till funktionen.

18 Vidare studier

Detta avslutar en snabb genomgång av Matlab. Genom att använda help-kommandot o dyl så kan mer inhämtas. Googla och/eller läsa i inledningen angiven litteratur tar en vidare i Matlab-världen.

That's all folks!