

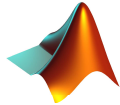
# Introduktionskurs i MATLAB

## Datorlektion 1

Denna version: 11 januari 2010

### 1 Välkommen till MATLAB

Denna lektion introducerar några viktiga delar i MATLAB— men du kommer inte långt om du inte går in på fler detaljer. Använd därför hjälpsystemet eller boken [1] för att ta reda på mer.

I ISY:s datorlab loggar du in med ditt studentkonto och -lösenord. Du kan starta MATLAB<sup>1</sup> genom ikonen . Ett MATLAB-fönster dyker då

upp, som innehåller tre delar:

- **Workspace:** Här kan man se vilka variabler man matat in (inga än så länge).
- **Command History:** De senast körda kommandona.
- **Command Window:** Det viktigaste fönstret. Det är här man kör själva MATLAB.

### 2 Förberedelseuppgifter

Innan första lektionen bör du göra följande förberedelseuppgifter:

<sup>1</sup>Beroende på vilket operativsystem som körs kan det finnas flera olika sätt att starta MATLAB. Fråga din lektionsassistent om du får problem!

1. Läs igenom introduktionskapitlet: ”What is MATLAB”, som du hittar en länk till om du öppnar helpdesk (skriv `helpdesk` vid kommandoraden — se vidare i avsnitt 3).
2. Gå igenom avsnitt 3–4 och gör så mycket du kan av uppgifterna. Se till att du hela tiden förstår vad som händer. Samla ihop de frågor som uppstår så att du kan ställa dem till lektionsassistenten.

### 3 Hjälpsystemet

MATLABs hjälpsystem får man fram genom att skriva

```
>> helpdesk
```

i MATLABs kommandofönster.

Ett hjälpfönster dyker då upp, med ett navigationsfönster till vänster och textfönster till höger. Från hjälpfönstret hittar man dokumentationen till MATLAB, och det finns flera sätt att få hjälp här. En grundlig introduktion till MATLAB finns i ”Getting started with MATLAB”. Den omfattar det som behandlas i den här kursen (och lite till). Om du inte vill läsa den direkt på skärmen kan du också ladda ner och skriva ut den.

Förutom introduktionen ”Getting started ...” finns i hjälpsystemet ett index över kommandon, en sökfunktion och demos.

Kommandot

```
>> help kommando
```

skriver ut en kort hjälptext om kommandot kommando (kommando ska alltså bytas ut mot det kommando du vill veta mer om). Ta för vana att köra det på alla nya kommandon. Prova det t ex på sig själv!

Ett annat väldigt användbart hjälpkommando är doc. Ta reda på vad det gör.

Mer hjälp finns även i boken [1].

Den bästa dokumentationen är ofta den man själv gjort. Gör därför en lista med användbara kommandon efter hand som du stöter på dem under lektioner och laborationer.

## 4 Matlab som miniräknare

Genom att skriva in numeriska uttryck kan MATLAB användas som en vanlig miniräknare. Prova

```
>> 5+3*2-4/7
```

Vad gör

```
>> round(10*ans)/10
```

.....

Andra avrundningskommandon är floor och ceil. Ta reda på hur de fungerar (Ledning: Använd help och prova även lite olika exempel!).

.....

.....

.....

Vad är ans för något? .....

Tangentbordets upp- och ner-pilar kan användas för att bläddra i de kommandon man tidigare skrivit. Prova det för att beräkna

```
>> 5+3*2-4/7
>> round(100*ans)/100
```

Man kan också skriva det första tecknet (eller fler) i något man tidigare skrivit. Pilarna bläddrar då mellan de kommandon som börjar på det eller de tecknet.

### Elementära funktioner

Beräkna:

```
cos(pi/3) = ..... sin(pi/4) .....
```

I vissa fall får man vara uppmärksam på att MATLAB har begränsad precision. Vad bör resultatet bli av:

```
>> sin(pi)
```

### Andra relaterade kommandon:

Upphöjt till görs med ^ . Vill man upphöja  $e$  till något använder man kommandot exp.

Komplexa tal kan anges med i eller j som den imaginära enheten. Några kommandon som kan användas är imag, real, abs och angle.

## Utmatningsformat

MATLAB visar normalt 4 decimaler för flyttal. Vill man se fler decimaler kan man använda

```
>> format long
```

Man ändrar tillbaka med `format short`.

Om man inte vill se svaret på ett kommando kan man avsluta med semikolon.

## Variabler

Vid längre beräkningar är det oftast smidigt att spara delresultat i variabler:

```
>> a=3
>> b=6;
>> c=b*a
```

De variabler man använt syns i fönstret "Workspace". Man kan också använda kommandot `whos`.

Använder man samma variabel igen får den ett nytt värde. Man kan t ex skriva

```
>> a = a+1
```

Vill man ta bort variabler finns kommandot `clear`. Att rensa bort variabler man inte använder minskar risken för misstag. Vid mer omfattande beräkningar kan det också behövas för att få bättre plats i datorns minne.

## 5 Skript och dokumentation

Det händer ofta att man vill köra en hel sekvens med kommandon flera gånger, kanske med lite olika värden. Det gör man genom så kallade MATLAB-skript, eller m-filer. En m-fil är helt enkelt en textfil med MATLAB-kommandon, som har filändelsen `.m` (samma filändelse har även funktioner, som vi kommer till senare). Filens namn blir ett kommandonamn, som kan användas i MATLAB precis som vilket kommando som helst.

För att minnas vad skriptet gör, och därmed ha glädje av det en annan dag, bör man ha med kommentarer till koden. Kommentarer börjar med ett `%`-tecken.

Starta MATLABs editor med kommandot `edit`. (Man kan också använda andra texteditorer, välj din favorit.) Skriv lite kod i din m-fil:

```
disp('Nu körs skriptet!');
stud1 = 3;      % Betyg som student 1 och 2 hade på en tenta
stud2 = 5;
medel = (stud1 + stud2)/2; % Här beräknas deras medelbetyg
disp('Resultat:');
display(medel);
```

Spara skriptet genom att välja `Save` eller `Save As` i `File`-menyn. Döp det t ex till `mittskript.m`. Skriptet sparas då som filen `mittskript.m`. Om man sparat skriptet i en särskild katalog (mapp) kan man behöva byta till den katalogen i MATLAB för att kunna köra det. Kommandot `dir` skriver ut innehållet i den katalog man är, och kommandot `cd` kan användas för att byta katalog. `pwd` talar om vilken katalog man befinner sig i.

Du kan nu använda det på samma sätt som andra MATLAB-kommandon:

```
>> mittskript
```

Anta att student 1 klagade på rättningen och fick 4 i betyg istället. Ändra i skriptet och kör skriptet igen. Glöm inte att spara efter ändringen.

## 6 Matriser

Matriser är den grundläggande datatypen i MATLAB och MATLAB har mycket kraftfulla funktioner för matrishantering och matrisberäkningar.

Skapa några matriser:

```
>> A = [1 2 5;3 8 10]
>> b = [7; 4; 5]
>> c = [3 2 1]
```

Vad har semikolon för funktion här? .....

Vad gör följande kommandon:

```
>> b(3) ? .....
>> A(2,3) ? (Vad står 2 och 3 för?) .....
>> A(:,2) ? (Här står : för "alla rader".) .....
>> A(1,:) ? .....
```

Att plocka ut element ur en matris på detta sätt kallas ibland för *indexering*.

Vi kan förstås räkna med matriser också:

```
>> A*b .....
```

Varför fungerar inte A\*c? .....

Använd .' för att transponera en matris. (Enbart ' ger komplexkonjugerat transponat.) Transponera någon matris och kontrollera att det stämmer.

Det finns andra sätt att skapa vissa speciella matriser. Vad gör följande kommandon:

```
>> H = ones(3,2) ? .....
>> B = eye(3) ? .....
>> y = 3:9 ? .....
>> x = 1:4:21 ? (Vad står 4 för?) .....
>> z = 10:-0.5:7 ? .....
```

Några fler exempel på indexering — vad händer här?

```
>> ind = [1 4 5]
>> y(ind)
>> A(2,2:3)
```

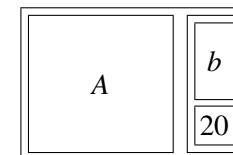
Indexering av delmatriser fungerar även för att tilldela värden till enstaka element:

```
>> A(1,1) = 9
>> A(2,3) = 4
>> A(2,1:3) = [7 3 9]
```

Skapa en större matris från andra matriser och vektorer:

```
>> A = ones(4,4)
>> D = [A [b;20]]
```

Här används klamrarna till att gruppera ihop ett antal matriser till en ny matris. Liksom i de tidigare matriserna betyder semikolonet att vi börjar på en ny rad. Se bilden nedan!



Varför fungerar inte D = [A b;20]? .....

Hur gör du för att lägga till en rad ettor under D? Dvs skapa matrisen

$$E = \begin{bmatrix} D \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

### Matrisfunktioner

De flesta elementära funktioner fungerar också på matriser, och verkar då komponentvis. För de operationer som har en speciell matrismening (främst multiplikation, division och upphöjt till) får man komponentvis verkan genom att sätta en punkt framför operatören. Prova

```
>> x = 0:0.1:1
>> cos(x).^2./x
```

Vad är skillnaden på

```
>> A*B
```

och

```
>> A.*B
```

(hitta på lämpliga matriser att prova på)? .....

.....

Gör ett skript som verifierar trigonometriska ettan med hjälp av komponentvisa operationer för  $x = 0.5, 1, 1.5, 2, 2.5, 3, 3.5$  och 4. Hur ser din kod ut?

.....

.....

.....  
.....  
.....

En vanlig användning av matriser är att representera ekvationssystem. Ekvationssystemet  $Ax = b$ , där  $A$  och  $b$  är matriser och  $x$  obekanta kan som bekant lösas genom multiplikation från vänster med  $A$ 's invers,  $x = A^{-1}b$ . I MATLAB, liksom när man räknar för hand, är dock detta inte det bästa sättet att lösa ekvationssystemet.

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

tar fram lösningen på ett snabbare och numeriskt mer stabilt sätt. Om  $A$ 's invers existerar ger det samma lösning som  $A^{-1}b$ . Bakåstrecket kan även användas för över- och underbestämda ekvationssystem, och ger då lösningen i minsta kvadratmening.

Lös ekvationssystemet

$$\begin{aligned} 5x_1 + 2x_2 &= 3 \\ 3x_1 + 2x_2 &= 5 \end{aligned}$$

Svar: .....

### Fler matriskommandon

Dessa kommandon bör du testa samt läsa dokumentationen till, då de kan vara användbara senare. Detta gäller särskilt de s.k **Databehandlings**kommandona. Notera att en del av kommandona kan ta fler än ett inargument och returnera mer än vad namnet antyder.

**Speciella matriser:** zeros, ones, rand, randn, eye, diag.

**Databehandling:** sort, max, min, find, sum.

## 7 Skriva egna funktioner

Förutom m-filer i form av skript, som man kan använda för att kunna köra en hel kommandosekvens om igen för lite andra variabelvärden, så kan man i MATLAB också skriva funktioner. Till funktionerna kan man skicka med inargument precis som till ett vanligt MATLAB-kommando, och man kan få ut ett eller flera utargument.

Funktioner börjar alltid med raden

```
function ut = filnamn(in1,in2,in3)
```

ut anger vilken variabel som ska returneras från funktionen. `filnamn` är namnet på funktionen, och den ska sparas i filen `filnamn.m`. `in1`, `in2` och `in3` är funktionens inargument, dvs det som användaren av funktionen behöver ge värden på för att funktionen ska kunna utföra sina beräkningar. Man kan ha godtyckligt många in- och utargument. Vill man t ex ha två utargument skriver man

```
function [ut1,ut2] = filnamn(in1,in2,in3)
```

*Exempel 1.* Vi kan skriva om vårt skript där vi beräknade medelbetyget för två studenter som en funktion:

```
function medel = medelbetyg(stud1,stud2)
% Funktion som beräknar medelbetyg för två studenter.
% Inargument: Betygen som student 1 och 2 hade på en tenta.
% Utargument: Medelbetyget.

disp('Nu körs funktionen!');
medel = (stud1 + stud2)/2; % Här beräknas deras medelbetyg
disp('Resultat:');
display(medel);
```

Spara den nya funktionen som `medelbetyg.m`.

Om vi jämför funktionen `medelbetyg` med motsvarande skript kan vi se att i skriptet skrev vi in direkt vilka betyg studenterna hade. Här låter vi istället betygen vara inargument, vilket innebär att vi lämnar till den som ska använda funktionen att bestämma vilka betyg han/hon vill beräkna medel på. Om man t ex vill beräkna medelbetyget för två studenter med betygen 3 och 5 kan man skriva

```
>> m = medelbetyg(3,5)
```

Funktioner och skript har två huvudsakliga skillnader:

1. Funktioner kan ha in- och utargument.
2. Ett skript delar variablerna med Workspace därifrån skriptet anropas. Funktioner däremot skapar sitt eget workspace, där till en början bara inargumenten är definierade. I funktionen kan sedan nya variabler definieras. När alla kommandon i funktionen har körts skickas utargumentet tillbaka till MATLABs Workspace och funktionens eget workspace (med alla variabler som definierats där) försvinner.

Många (men inte alla) av de interna funktionerna i MATLAB är skrivna på samma sätt som i exemplet ovan. För att se hur t ex funktionen `mean` ser ut kan vi köra kommandot

```
>> type mean
```

*Exempel 2.* Pelle har tagit ett lån för att köpa något stort och fint (en bil, en hemmabio eller en segelbåt). Han har tänkt betala tillbaka det med 10% av sin lön varje månad. Lånet har 1% ränta per månad. Skriv en funktion där Pelle kan mata in lånebeloppet och lönen, och få ut hur mycket han är skyldig nästa månad (alltså efter en månads ränta och betalning). Ett förslag på kod är:

```
function ny_skuld = lanberakning(skuld,lon)
% ny_skuld = lanberakning(skuld,lon)
% Beräknar skulden nästa månad.
% Mata in skuld och lön

ranta = 0.01;           % Ränta per månad
betalningsgrad = 0.1;  % Betalningsgrad i förhållande
                        % till månadslönen
```

```
ny_skuld = skuld + skuld*ranta - betalningsgrad*lon;
```

Skriv in koden i en fil som du sparar med namnet lanberakning.m. (Filen kommer att användas även under nästa lektion.) Funktionen anropas med

```
>> lanberakning(pelles_skuld,pelles_lon)
```

där pelles\_skuld och pelles\_lon är numeriska värden på Pelles skuld och lön, eller variabler som innehåller dessa värden.

Kolla hur stor Pelles skuld blir nästa månad, om lånet är på t ex 10 000 kr, och Pelle tjänar 25 000 kr/månad.

.....

## 8 Grafik

En bild säger mer än tusen ord. Vad ritas av följande exempel?

```
>> t = 0:0.01:6*pi;
>> y = cos(t);
>> plot(t,y)
```

.....

Vad är skillnaden jämfört med

```
>> plot(y)
```

Varför blir denna figur inte lika bra?

```
>> t = 0:1:10;
>> y = cos(t);
>> plot(t,y)
```

.....

Om man vill rita flera funktioner i samma figur kan man t ex använda

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

Man kan också använda kommandot hold. Ta reda på hur det fungerar och rita cos och sin i samma diagram med hjälp av det. Din kod:

.....  
.....  
.....  
.....

**Uppgift:** Följande uppgift är hämtad från kompletterande kursmaterial till Analys A:

Rita kurvan  $f(x) = \frac{x^3}{x^2 - 2|x - 2|}$ . Ange alla extremvärden samt största och minsta värden, om de existerar.

Gör en MATLAB-funktion som löser uppgiften numeriskt. Du måste generera en vektor  $x$  över lämpligt intervall och med tillräcklig upplösning, evaluera funktionen i dessa punkter, och sedan använda ett lämpligt kommando för att hitta extremvärdena. Om du vill kan du börja med att göra ett skript och sedan göra om det till en funktion. Kalla funktionen för `kurvplot`. Att tänka på:

- Vilka  $x$ -värden är intressanta? Hur tätt ska de ligga?
- Man kan zooma in i bilden med kommandot `zoom`, eller genom zoom-verktyget (förstoringsglaset) högst upp i figuren. Ett annat användbart kommando för att fokusera på det intressanta i bilden är `axis equal` ger axlarna samma skala, så att `text` en cirkel ser ut som en cirkel.
- Kommandot `grid` ger ett rutnät i figuren.
- Vad är för- och nackdelen med att undersöka en funktion på det här sättet jämfört med en traditionell funktionsundersökning à la Analysen (där man tar fram derivatans nollställen osv)?

Sätt namn på figuren och axlarna genom att använda kommandona `title`, `xlabel` och `ylabel`. Du kan skriva ut din figur med kommandot `print`.

Lämpliga inargument kan vara det intervall man vill rita kurvan i, och lämpliga utargument största och minsta värdet i det intervallet. Första raden i funktionen kan alltså se ut så här:

```
function [fmin,fmax] = kurvplot(xmin,xmax)
```

Funktionen kan sedan anropas på följande sätt:

```
>> [fmin,fmax] = kurvplot(0,10)
```

(Inargumenten — värdena på `xmin` och `xmax` — kan naturligtvis ändras beroende på vilket intervall man vill undersöka.)

Att inkludera en MATLAB-figur i `tex` en rapport kan vara mycket illustrativt. Figurerna kan sparas genom att välja `Save As` i `File`-menyn, eller genom att ange fler argument till kommandot `print`. Vilket format man ska välja beror på vilket program man vill inkludera figuren i. Några exempel på format är `jpg` för bilder som ska publiceras på internet, `eps` för filer som ska inkluderas i  $\text{\LaTeX}$  och `meta` för filer som ska inkluderas i `Word` (fungerar dock bara i `Windows`).

## 9 Hemuppgifter

### Allmänt

1. Gör färdigt det du inte hann under lektionen!

### Matlab som miniräknare

1. Förutom `round`, `floor` och `ceil` finns ytterligare ett avrundningskommando, nämligen `fix`. Vad är skillnaden på `floor` och `fix`? Ledtråd: Jämför deras hjälptexter! Hur avrundas negativa tal?  
.....
2. Använd `help log` eller någon annan hjälpfunktion för att ta reda på hur du räknar ut följande med MATLAB:  
 $\ln(34) = \dots \log_2(8) = \dots$
3. Använd `format long` för att svara på frågan: Är  $\ln(203.8)/\sqrt{9\pi}$  större än 1? .....  
Finns det andra sätt att få svar på frågan?

## Dokumentation

1. Ibland kan man vilja spara sina variabler i en fil (t ex när man har gjort en lång komplicerad beräkning och vill minnas resultatet till ett annat tillfälle. Det kan du göra med kommandot `save`. Du kan sedan ladda tillbaka dem med `load`. Variablerna sparas automatiskt i MATLABs eget format<sup>2</sup> (MAT-filer), men du kan också spara dem som text. Kommandot

```
>> save min_fil a b c
```

sparar variablerna a, b och c i en MAT-fil. Prova det på några variabler!

## 10 Fördjupning

Fördjupa dig i två eller flera av följande områden. Använd boken [1] eller MATLABs egen dokumentation.

### Mer om MATLAB som miniräknare och matriser/linjär algebra

**Förslag på fördjupningsuppgifter ur boken [1]:** 1, 3, 5, 7, 9, 15, 17, 34

Se även hjälpavsnittet ”Matrices and Linear Algebra”.

---

<sup>2</sup>Formatet har ändrats mellan version 5, 6 och 7 av MATLAB, så om du ska ladda in variabler i en annan version än du sparar dem i bör du läsa dokumentationen till `save`.

## Mer om grafik

Fler ritkommandon: `bar`, `pie`, `stem`, `hist`.

Illustrera flera saker: `subplot`, `figure`

Dokumentera figurerna: `text`, `legend`

Undersök också hur man kan variera färg och plotsymboler.

**Förslag på uppgifter:** 29, 32

Några kommandon för 3D-grafik: `mesh`, `surf`, `contour`, `meshgrid`, `colormap`

**Förslag på uppgifter:** 39, 41

MATLAB-dokumentationen har ett avsnitt om grafik. Det finns dessutom flera demos.

## Symboliska beräkningar

Symbolic toolbox kan användas för att göra symboliska beräkningar i MATLAB. Se s 92 i boken [1].

I MATLAB-dokumentationen finns ett avsnitt om Symbolic Math Toolbox, samt demos.

## Numerik och noggrannhet

MATLAB som numeriskt verktyg har god noggrannhet i allmänhet, men i vissa fall bör man se upp. Mer om MATLABs numeriska noggrannhet och representation av tal finns i kapitel 20 i boken [1].

## Referenser

- [1] F. Gustafsson och N. Bergman, *Matlab for Engineers Explained*, Springer Verlag, 2003.