Chalmers tekniska högskola

En introduktion till MatLab

Författare:

Gustafsson Gabriel
gabgus@student.chalmers.se

Norell Pontus
npontus@student.chalmers.se

Johansson Việt Simon
simoj@student.chalmers.se

Staničić Ivana
stanicic@student.chalmers.se

18 augusti 2014

Innehåll

1 Inledning

Det här dokumentet innehåller övningsuppgifter och exempel som ska användas till MatLabintroduktionen för nyantagna teknologer på sektionerna K och KfKb. Materialet är framtaget av äldre teknologer på dessa sektioner. MatLabintroduktionen är frivillig och kunskaper inom MatLab är inte något förkunskapskrav. Därför ligger fokusen främst på grundläggande begrepp och kända svårigheter från tidigare års undervisning.

Introduktionen är uppdelad i två 1,5 timmars lektioner och förhoppningsvis ska dessa uppgifter vara i lagom omfattning för den tiden. Tanken är att studenterna ska få göra så många uppgifter som möjligt där man får prova sig fram och därmed själva få se och experimentera med hur MatLab fungerar. Syftet är att ge en introduktion till MatLab och att jämna ut kunskapsskillnader hos de nya teknologerna för att underlätta den ordinarie undervisningen och inlärningen.

2 Lektion 1 - Grunderna i MatLab

Målet med lektion 1 är att lära sig hur MatLab fungerar på en grundläggande nivå. Studenterna ska förstå hur man gör enklare beräkningar, vad man använder de olika MatLab-fönsterna till, hur MatLab läser och tolkar kommandon, hur man sparar sin kod samt diverse tips och trix. Funktioner, if-sats, for- och while-loopar introduceras också.

2.1 Uppgift 1

a) Öppna MatLab. Desktop: Markera "Command Window" och "Workspace".
I Command window: skriv 1 och klicka enter. → ans=1. (Variabelns ans har tilldelats värdet 1) Skriv ans och klicka enter för att kontrollera detta → ans=1. Gå in i Workspace och observera att variabeln ans har sparats (Name: ans, value: 1 m.m.)

b) Ans är standardnamnet för variabler i MatLab. Om du skriver en ny siffra i Command Window kommer ans tilldelas ett nytt värde. Skriv 2 klicka enter → ans=2. Kolla vad som hänt i Workspace.

c) Du kan även hitta på egna variabler. Skriv A=1 klicka enter→ A=1. A har tilldelats värdet 1. Kontrollera att variabeln A har sparats i Workspace. Gå tillbaka till Command window: Skriv A och klicka enter→ A=1.

d) Om man gör många beräkningar i Command window kan det lätt bli lite väl mycket siffror och bokstäver. För att rensa Command Window: Skriv clc och klicka enter. (clc står för clear command)

e) Gå in i Workspace och notera att ans fortfarande är 2 och att A fortfarande är 1. Detta kan du även se om du går in i Command window: Skriv A och klicka enter→ A=1

f) För att rensa Workspace skriv clear i Command Window och klicka enter. Kontrollera att Workspace är tom. Skriv A i Command Window och klicka enter → En röd varningstext kommer upp och berättar att A är odefinerad, d.v.s. variabeln A har inte längre något värde.

g) Command window fungerar precis som en miniräknare. Prova själv följande beräkningar eller hitta på egna:

$I) 1 + 1$
$II) 1 - 1$
$III) 4/2$
$IV) 2 * 2$
$V) \sin(0)$
$VI) A = 2$
$VII) B = 2$
$VIII)\ A + B$
$IX)\ C = 1 + 3$
X) sqrt(4)
XI) `2^2`
osv..

2.2 Uppgift 2

I de kommande uppgifterna ska vi arbeta i Editorn istället för i Command window. För att kunna köra sin kod i Editorn måste den vara sparad som en m-fil (filnamnet slutar med .m) Vi börjar med att skapa en mapp där ni ska spara ert arbete. Skapa en mapp i z: och döp den till "MatLab-introduktion". Nu återgår vi till MatLab. Gå in under fliken desktop och markera Current folder (om den inte redan är markerad). Nu kommer fönstret Current folder finnas i MatLab-miljön. Leta reda på mappen du precis skapade och markera den. Se alltid till att mappen du sparar din kod i är markerad i Current folder annars kommer MatLab inte kunna hitta din kod sen när den ska köras.

Längst upp i Matlabfönstret finns fliken "Editor". Markera denna och tryck sedan på "New" för att öppna ett nytt fönster. I denna kan du nu skriva ett script. Ett script är en lista av instruktioner som du vill att Matlab skall göra i en bestämd ordning. Kommandon som du skriver in i ditt script är likadana som de du kan använda i Command Window. Fördelen med ett script är att du inte längre är begränsad till en rad kod för att utföra en handling.

a) Skriv en instruktion åt MatLab så att den rensar "Workspace" och samtidigt tilldelar x värdet 10. Skriv instruktionerna på två olika rader. Spara koden i mappen MatLab-introdutionsom du nyligen skapade och döp filen till "introduktionsuppgifter". Notera att filen sparas som en m-fil, vilket är helt i sin ordning. Se till att rätt mapp är markerad i Current folder. Tryck sedan "Run" för att köra scriptet.

b) MatLab läser alltid koden uppifrån och ner. Detta är väldigt viktigt att ha i bakhuvudet när man skriver och när man kör en kod. Prova detta genom att köra följande kod.

```
A=1
B=2
C=3
```

Notera att svaren skrivs ut i samma ordning som i koden. Ändra plats på variablerna (enligt nedan) och kör.

```
C=3
B=2
A=1
```

Notera att svaren skrivs ut i den nya ordningen när du kör.

c) Om något är fel kommer MatLab avsluta körningen och ge en röd varningstext. Testa detta genom att ändra till en felaktig kod enligt nedan.

```
A=1
B=*
C=3
```

Notera att ingen av operationerna utfördes.

d) När man gör många beräkningar vill man inte att MatLab skriver ut alla svar utan bara de svar man är intresserad av. Då använder man semikolon. Till exempel om man endast vill att B ska skrivas ut skriver man på följande vis.

```
A=1;
B=2
C=3;
```

Kör och se vad som händer.

e) Det är viktigt att ha bra struktur i sin kod, särskilt när det blir mycket kod. I MatLab använder man procent-tecken för att skriva kommentarer. Då kan man till exempel skriva rubriker till delavsnitt eller beskriva vad en viss kod beräknar. Skriv in den nedanstående koden och kör för att se vad som händer.

```
% I denna MatLab-kod kommer C beräknas med hjälp av A och B

A=1 % Variabeln A har värdet 1
B=2 % Variabeln B har värdet 2
C=A+B % C beräknas genom att A och B adderas
```

f) Ett ytterligare sätt att skapa ordning och reda i sin kod är att dela in olika avsnitt i så kallade celler. Detta görs med dubbla procent-tecken. Skriv in följande exempel och se hur det ser ut.

```
% Denna kod beräknar C, D och E.
%% Konstanter
A=2;
B=4;
%% Beräkning av C
C=A+B
%% Beräkning av D
D=B/A
%% Beräkning av E
E=A*B
```

Kör och se vad som skrivs ut och vad som inte skrivs ut.

2.3 Uppgift 3

a) Öppna ett nytt script. Skapa en funktion genom att högst upp i scriptet skriva:

```
function y=Dubblering(x)

y=2*x;  % Definierar den nya variabeln y, dubbelt så stor som x.

end
```

och spara funktionen som en m-fil med namnet ”Dubblering” i den aktuella mappen. Det är viktigt att funktionen sparas med samman namn som sedan kommer användas för att anropa den, annars kommer MatLab inte kunna hitta den.

b) Funktionen som vi nyligen skapade i uppgift a) fungerar enligt följande: Man lägger in ett värde,t.ex. 2, till variabeln x genom att skriva Dubblering(2). Då kommer funktionen att utföras och man får ut en ny variabel y med ett nytt värde som är 4.

Kolla om funktionen fungerar genom att använda funktionen i Command Window

```
>> Dubblering(5)
```

c) Värdet x som du använder funktionen på behöver inte vara en siffra utan kan vara en variabel den med. Testa att använda din funktion på ditt förra utvärde och se vad som händer. Notera att värdet i Workspace skall ändra på sig.

```
>> Xny=5

>> Dubblering(Xny)
```

d) Konstruera en funktion `function y=Celsius(x)` som konverterar Celsius till Farenheit och kontrollera om funktionen fungerar genom att i Command Window skriva Celsius(18). Förhållandet mellan systemen är

$Farenheit = Celsius * 1.8 + 32.$

2.4 Uppgift 4

a) if-sats. En if sats är uppbygd på följande vis

Det börjar med ett konstaterande:

“Om A är mindre än 1”

Sedan fortsätter man med:

Så gäller att B=2

Slutligen avslutas loopen

“end”

I Matlab-kod ser det ut så här:

```
if A<1
B=2
end
```

if-satsen kommer endast att “köras” om konstaterandet gäller. Testa själv:

```
clear
clc
A=0
if A<1
B=2
end
```

Kör och notera att B=2

Ändra till A=2, kör igen och notera att B är odefinerad.

Man kan lägga till en “else”- som talar om vad som gäller om if-satsen inte är uppfylld.

Testa detta med följande kod

```
clear
clc
A=2
if A<1
B=2
else
B=1
end
```

Prova och ändra värdet på A och notera att om A är mindre än 1 kommer if-satsen köras och om A inte är mindre än 1 kommer "else-satsen" att köras.

b) for loop

En for loop är uppbygd på följande vis

Det börjar med att man definerar hur många "varv" for-loopen ska gå samt vilka värden som ska användas i respektive varv:

"for i=1,2,3 (Detta betyder att tre loopar ska genomföras, i loop 1 är i=1 i loop 2 är i=2 och i loop 3 är i=3)

Sedan skrivs vad som ska beräknas:

A=i

Slutligen avslutas loopen

"end"

I Matlab-kod ser det ut så här:

```
for i=[1 2 3]
A=i
end
```

Testa själv

```
clear
clc
for i=[1 2 3]
A=i
end
```

Kör och notera att tre uträkningar av A görs, en för varje loop. A=1, A=2 och A=3

c) while loop

En while loop liknar en for-loop, men har framförallt en betydande skillnad; I en for-loop definerar du exakt hur många loopar som matlab ska utföra. I en while-loop repeteras loopen tills ett visst kriterie som du har definerat är uppfyllt.

En while loop är uppbygd på följande vis

Det börjar med att man skriver ett kriterie som definerar när loopen är färdig:

while $i < 3$ (Detta betyder att så länge i är mindre än 3 ska loopen fortsätta köras)

Sedan skrivs vad som ska beräknas:

i+1 (I varje loop adderas 1 till variablen i)

Slutligen avslutas loopen

"end"

I Matlab-kod ser det ut så här:

```
while i<1
i=i+1
end
```

Testa själv

```
clear
clc
i=0
while i<3
i+1
end
```

Kör och notera att tre uträkningar av i görs, i=1, i=2, i=3, d.v.s. loopen har repeterats tre gånger. Efter den tredje beräkningen har i antagit värdet 3 och är därmed inte längre mindre än 3. Då avslutas loopen.

3 Lektion 2 - Vektorer, loopar och grafer

Vektorer är en samling nummer och skrivs enligt följande i Matlab.

```
>> v=[1 2 3 ... 10]
```

Testa själva! Om man vill göra det enkelt för sig kan man skriva på följande sätt:

```
>> v=1:1:10
```

Skriv ut v i Command Window och se vad som händer.

I MatLab kan dessa samlingar användas till att rita ut punkter i grafen eller figurer (man kan välja dessa nummer så att de bildar en kvadrat eller triangel exempelvis). Har man en väldigt lång vektor `v` kan man ta reda på elementet på plats nummer `N` genom att skriva `v(N)`. Om man skulle vilja anropa fler element, till exempel element två till fem, så kan man göra det genom att skriva `v(2:5)`.

3.1 Uppgift 1

a) Gör en egen vektor med 5 valfria siffror. Anropa det tredje elementet i vektorn.

b) Multiplicera vektorn med 10 och anropa det tredje elementet igen. Skriv vektorns namn i Command Window. Skedde allt som du trodde att det skulle?

c) Gör en ny vektor med 5 valfria siffror (Använd ett annat namn!). Addera den föregående vektorn med den nya. Anropa element 3 till 5. Skriv ut alla tre vektorer och kolla att allt stämmer.

Ett annat kommando att skapa vektorer med är linspace. Genom att skriva `linspace(startvärde,slutvärde,antal värden)` får man ut en vektor med dessa egenskaper. Om antalet värden inte skrivs ut använder kommandot värdet 100 som standard, det vill säga `linspace(1,10)` är samma sak som `linspace(1,10,100)`.

d)Testa `a=linspace(1,10,10)` och jämför med de andra vektorerna.

3.2 Uppgift 2

MatLab är mycket användbart när man vill visualisera data av olika slag. Datan som används är oftast i vektorform. Det kommando som vanligen används är `plot(x,y)`, där x är en vektor med alla x-värden och y en vektor med alla y-värden. y-vektorn har man i många fall räknat ut tidigare med hjälp av x-vektorn.

Det plotkommandot gör är att rita punkter med koordinaterna `(x1,y1), (x2,y2)`... och sedan dra ett streck mellan dessa.

a) Skapa två vektorer med 4 värden. Plotta dina värden med `plot(x,y)` och se om du kan se hur vektorerna hänger ihop med punkterna i grafen.

b) Använd linspace för att skapa en vektor `x` på intervallet `0` till 2π med fyra värden. Skapa en y-vektor genom att i ditt script skriva: `y=sin(x);`

Plotta dessa vektorer med hjälp av `plot(x,y)`.

c) Testa att öka antalet värden i x-vektorn. Testa med 7, 10 och 50 värden.

Tips: Om ni vill jämföra bilderna kan ni plotta de i olika figurfönster. Detta görs genom att skriva figure(1), figure(2) etc. innan plotkommandot.

d) Se om du kan plotta en triangel.

3.3 Uppgift 3

Ibland behöver man göra samma sak många gånger och då kan MatLab vara till stor hjälp. Man använder sig av så kallade loopar. De två vanligaste looparna är for- och while-looparna. For-looparna kan användas till att summera olika talföljder, medan while-looparna kan användas till att utföra en beräkning under en begränsning. Om man exempelvis vill summera talen 1 till 10 kan man göra det på följande sätt med en for-loop.

```
s=0
for i=1:10
s=s+i
end
s
```

a) Summera följande tal $1^2 + 2^2 + ... + 50^2$.

b) För att få MatLab att skriva ut text kan du använda `disp('text')`, notera enkelparenteserna. Skriv "I like fluffy unicorns" 10 gånger. Använd en for-loop.

Ett annat användningsområde för for-loopar är att räkna ut samma sak fast för olika fall. Om man exempelvis hade velat konvertera celsius till farenheit, som i uppgiften från förra lektionen, vid temperaturerna 10, 40, 50 och 100. Så hade man kunnat göra det såhär:

```
for T=[10 40 50 100] % Där T står för temperaturen
Farenheit=T*1.8+32
end
```

c) Konvertera svenska kronor till dollar för artiklar med priserna 20 kr, 49 kr, 112 kr, 169 kr och 500 kr. I skrivande stund är en svensk krona värd 0.150525 dollar.

3.4 Uppgift 4

Ibland vill man att MatLab ska göra olika saker från fall till fall. Ett av de enklaste sätten att göra detta är med hjälp av if-satser. En av de enklaste formerna på en if-sats kan se ut på följande sätt:

```
if T>100    % Där T står för temperaturen
disp('För varmt!!')
end
```

En if-sats på den här formen fungerar på så sätt att om det villkor som skrivs direkt efter if-kommandot är uppfyllt så gör MatLab det som står mellan `if` och `end`, men inte annars.

Operator	Betydelse
$>$	Större än
$<$	Mindre än
$==$	Lika med
$>=$	Större än eller lika med
$<=$	Mindre eller lika med
$\sim$	Inte

Om if-satsen utökas med ett else-kommando kan det se ut enligt följande:

```
if T>100    % Där T står for temperaturen
    disp('För varmt!!')
else
    disp('Åh va bra')
end
```

Den här if-satsen fungerar som den föregående men med tillägget om villkoret inte är uppfyllt så gör MatLab det som står efter `else`.

Den sista utökningen av en if-sats fås genom att lägga till ett eller flera `elseif` som nedan.

```
if T>100    % Där T står for temperaturen
    disp('För varmt!!')
elseif T<10
    disp('För kallt!!')
else
    disp('Åh va bra')
end
```

I denna if-sats utvärderar MatLab först villkoret vid if-kommandot, om detta inte är uppfyllt går det vidare till nästa villkor, om inget av villkoren är uppfyllt går MatLab vidare till `else` som i föregående if-sats.

Om man använder flera `elseif` får man tänka på vilken ordning MatLab läser dessa. Så fort Matlab hittar ett villkor som är uppfyllt nöjer den sig. Exempelvis om det hade stått `elseif T<4` efter det första `elseif` i exemplet ovan hade det aldrig blivit läst för 4<10 och då är det första villkoret alltid uppfyllt.

a) Skriv in den sista if-satsen i ditt script. Testa vad som händer om T=5, T=50, T=120. Skriv `T="nummer"` innan if-satsen i scriptet.

b) Kommandot `randi(6)` ger slumpmässiga tal mellan 1 till 6. Liknande det du får om du slår en tärning. Gör med hjälp av en if-sats ett script som slår en tärning och säger i ord vad resultatet blev. Exempelvis "Du fick en trea", "Du fick en fyra" etc.

Kom ihåg: Om du använder >= eller <= är det viktigt att tänka på ordningen. Se stycket ovan. Du kan även använda ==.

c) Oscar och Kalle kastar var sin tärning. Gör ett script som säger vem som får högst. Exempelvis "Kalle vinner, wihoo!", "Oscar vinner, tjodelihejsan!" eller valfritt utrop.

4 Extrauppgifter

4.1 Lektion 1 - extrauppgifter

För de som är klara erbjuds ett par svårare uppgifter som bland annat innebär att man får tar reda på ännu fler kommandon själv med hjälp av hjälpavsnitten och sedan använda dessa i praktiken. Vissa kommandon gås kort igenom här.

a) Läs i hjälpavsnittet om kommandot `integral` genom att skriva

```
>> help integral
```

i Command window. Använd sedan kommandot `integral` till att beräkna integralen av

```
y=@(x)exp(x)
```

på intervallet 0 till 5.

b) Skriv in följande i Commando window:

```
penny
```

```
spy
```

```
life
```

```
why
```

4.2 Lektion 2 - extrauppgifter

För de som är klara erbjuds ett par svårare uppgifter som bland annat innebär att man får tar reda på ännu fler kommandon själv med hjälp av hjälpavsnitten och sedan använda dessa i praktiken. Vissa kommandon gås kort igenom här.

a) Ibland vill man plotta flera bilder i samma fönster. Då kan man använda kommandot `subplot`. Genom att skriva `subplot(a,b,c)` kan man välja med a hur många rader man vill ha, med b hur många kolonner man vill ha och med c vilken man ska plotta i. Plotta nu två olika funktioner i samma fönster. Prova att plotta genom att skriva `plot(x,y,'g')` och variera g genom att byta ut den mot m, b, r eller y exempelvis.

b) Genom att använda `xlabel` kan man märka sin x-axel och med hjälp av `ylabel` märka sin y-axel. Gör detta för två av dina plottar. Vanligtvis vill man även ha en titel till sin graf. Sök i hjälpavsnittet om kommandot `title` och använd den på två av plottarna.

```
>> help title
```

c) Plotta funktionen `y=sin(x)` över intervallet 0 till 60 genom att använda `linspace`. Skriv in

```
>> axis equal
```

i Command Window och lista ut vad som händer.

d) Om man bara skulle vilja titta på bara en period kan man använda sig av kommandot `axis([x1 x2 y1 y2])`, se hjälpavsnittet. Skala nu om bilden så att man bara ser en period.

e) Plotta funktionen `y=@(x)(x-1).*exp(-(x-1).^2)+0.2;`

i intervallet -7 till 7. Skriv `grid on` i Command window. Vad är nollställena till funktionen? Svara på ett ungefär, använd ginput(2).