Laboration 1

Introduktion

Efter den här laborationen ska du kunna använda Matlabs olika fönster och hjälpfunktioner. Du ska kunna skapa, manipulera och använda variabler som är skalära, vektorer och matriser. Du ska kunna plotta t.ex. funktioner, använda if-satser och villkor samt skriva script.

I. Matlabs fönster, hjälp, variabler och enkla beräkningar

Dessa uppgifter löses i Matlabs kommandofönster.

När du startar Matlab öppnas ett antal fönster bland annat kommandofönstret, på engelska Command Window. I detta fönster kan man skriva in olika kommandon och exekvera dem. Till exempel kan man beräkna *cosinus* för *pi* genom att skriva $cos(pi)$ på kommandoraden >> och trycka på enter/retur-tangenten. Matlab exekverar kommandot dvs tolkar vad som står, utför beräkningen och skriver ut svaret -1.

```
>> cos(pi)
ans =
    -1
>>
```

Den tomma kommandoraden >> visar att Matlab är redo för ett nytt kommando.

1. Att hitta hjälp

Prova Matlabs inbyggda hjälpfunktioner genom att skriva *help* följt av ett sökord t.ex.

```
>> help cos
```

Vad heter den *cosinus*-funktion som tar grader som indata?
Försök gärna med andra sökord.

Ett alternativ till att söka hjälp via kommandoraden är att använda *Help* via Matlabs meny eller ikonen ?. Här kan man hitta lite mer och utförligare beskrivningar av olika saker. Klicka på ? och vid *Search for* : skriver du in *getting started*. Titta igenom det som du tycker är intressant.

Lägg gärna lite tid på att söka och använda Matlabs hjälp-funktioner! Ett tips är att titta på olika demos, gör gärna *help demo*!

2. Att ge variabler namn och värden.

Vi ska nu börja med att använda variabler i Matlab. Läs i Matlab-häftet om variabler.

Innan du börjar skriva in kommandon tittar du i fönstret som kallas arbetsarea (på engelska Workspace). Hur ser det ut? Efter varje kommando du skriver in tittar du i arbetsarean och ser hur den förändras. Varför förändras den?

Skriv nu in följande på kommandoraden

```
>> x=2;
>> w=pi;
```

Vad händer om man dubbelklickar på w i arbetsarean?

3. Att arbeta med variabler.

Nu har vi definierat två variabler som vi kan använda. Fortsätt nu med att skriva in kommandona nedan och ta reda på vad som händer och varför. Vad händer i arbetsarean och varför? Vilket värde har w respektive x, förändras det och i sådana fall när?

```
>> w
>> format long
>> w
>> format short
>> cos(w*x)
>> ans*2
>> x^2+x-5
>> x = x + 3
```

Vad är *ans*? Hur kommer du att använda arbetsarean och *ans*? Titta i Matlab-häftet eller använd dig av *help*.

Vad är det för skillnad då man skriver ; i slutet av kommandona? Titta i Matlab-häftet kapitel 3.

4. Vilka variabelnamn är tillåtna/otillåtna och varför?

(a) $hund1$

(b) $1hund$

(c) $antal\_hundar$

(d) $\_ras$

(e) $rasRen?$

Tänk på att du alltid använder variabelnamn som är beskrivande. Det är ett enkelt sätt att göra program lättlästa.

5. Beräkningar

Gör nedanstående matematiska beräkningar i Matlab och använd dig av att $x = -2$ och $y = \pi$. Tänk igenom vad beräkningarnas resultat blir innan du kör dem i Matlab. Leta i t.ex. Matlab-häftet hur man skriver de olika matematiska funktionerna.

Skriv variablerna på kommandoraden och därefter beräkningarna.

Blir resultaten som du tänkt dig?

(a) $cos(xy)$

(b) $sin(x)/cos(x) - tan(x)$

(c) $ln(x)$

(d) $sin^2(xy)$

(e) $\sqrt{(x^2 + y^2)}$

II. Vektor, matris och plot

Dessa uppgifter löses i kommandofönstret.

1. Att skapa och arbeta med vektorer

För att skapa en vektor kan man göra på ett flertal olika sätt, se i Matlab-häftet. Skapa en vektor med heltalen 1, 2, 3, 4 och lägg den i variabeln x. Prova olika sätt att skapa vektorn. Tänk på att kontrollera vektorn så att du säkert vet att du gjort och skrivit rätt. Resonera dig fram till vad följande beräkningar kommer att resultera i. Vad händer och varför? Rita och förklara!

(a) $x + x$

(b) x'

(c) x^2

(d) $x' * x$

(e) $\sqrt{x}$

(f) $x(5) = x(3) * x(1)$

(g) $x.\hat{}2$

(h) $plot(x)$

(i) $x.\hat{}2 + x - 5$

Skriv in beräkningarna i kommandofönstret, överensstämmer dessa resultat med dina? Hur ser arbetsarean ut? Matlab kommer att ge felmeddelande för en del av beräkningarna, varför? Vad är felet?

En likartad beräkning som den sista beräkningen ovan finns i uppgiften I. 3, vilken är skillnaden mellan de olika beräkningarna?

2. Att skapa och arbeta med matriser

Skapa matrisen

$$A = \begin{pmatrix} 3.1 & 2.5 & 4 & -1.3 \\ 5 & 2 & 3 & 45 \\ -1 & 3 & 7.1 & 2 \end{pmatrix}$$

För att skapa en matris kan man göra på ett flertal olika sätt, se i Matlab-häftet. Prova olika sätt att skapa matrisen. Tänk på att kontrollera matrisen så att du säkert vet att du gjort och skrivit rätt. Resonera dig fram till vad följande beräkningar kommer att resultera i. Vad händer och varför? Rita och förklara!

(a) $A + A$

(b) A'

(c) A^2

(d) $A' * A$

(e) $A(2 : end, 2)$

(f) $A(:)$

(g) $A(4, 3)$

Skriv in beräkningarna i kommandofönstret, överensstämmer dessa resultat med dina? Hur ser arbetsarean ut? Matlab kommer att ge felmeddelande för en del av beräkningarna, varför? Vad är felet?

3. Beräkna och plotta funktionen $y(x) = x^2 + x - 5$

Skapa en vektor x med talen 0, 0.01, 0.02, 0.03, ... 2.97, 2.98, 2.99, 3. Utför därefter beräkningen av funktionen $y(x) = x^2 + x - 5$ för att sedan plotta y som en funktion av x med hjälp av *plot*. Titta i Matlab-häftet hur du kan göra!
Lägg till titel, axeletiketter och rutnät (eng. *grid*) i plotten. Använd hjälp-funktionerna eller Matlab-häftet som hjälp.

4. Att kunna välja.

För att kunna välja mellan två olika alternativ kan man använda en if-sats. Vi ska nu kontrollera vilken variabel av x och y som har det största värdet.
Använd dig av att $x = -2$ och $y = 4$. Skriv in följande i kommandofönstret:

```
>> if x>y
disp(' x är större än y ')
else
disp('x är inte större än y ')
end
```

Stämmer texten som skrivs ut? Vad händer och varför?

Vi förändrar if-satsen och skriver in följande i kommandofönstret:

```
>> if x>y
disp(' x är större än y ')
y=x;
else
disp('x är inte större än y ')
x=y;
end
```

Vad händer och varför? Vilka värden har x och y före och efter if-satsen? Om man skriver in if-satsen en gång till får man samma resultat då? Stämmer texten som skrivs ut?

III. Script och programmeringsteknik.

All programkod läggs i en m-fil, ett s.k. script.

1. Skriva ett script

Öppna en ny fil genom att i meny-raden välja File/New/M-File eller klicka på ikonen som ser ut som ett tomt ark. Skriv in raden:

```
clear all, close all, clc
```

först i filen. Ta reda på vad dessa gör! Varför är det en bra ide' att börja ett program med dessa kommandon? Är det alltid en bra ide'?
Därefter ska du kopiera de kommandon du använde för att lösa plot-uppgiften II.3 ovan till filen. Du kan återfå de kommandon du tidigare har gett i kommandofönstret genom att använda ↑ och ↓ och bläddra fram och tillbaka mellan dem. En alternativ väg är att använda kommadohistorien (på engelska Command History).
När du kopierat in alla kommandon i rätt ordning sparar du filen via menyns File/Save och anger namnet $uppgift1.m$. För att köra programmet, scriptet skriver du in filens namn (utan filändelsen m) på kommandoraden, dvs

```
>> uppgift1
```

Ett alternativt sätt att spara och köra programmet är att i editorn klicka på pilikonen.

Blir resultatet detsamma som förra gången? Vad händer när man kör ett script? När är det bättre att använda script än kommandoraden och vice versa? Vad händer med arbetsarean?

2. Programmeringsteknik

Börja med att skriva in följande program i filen *reverseOrder.m*.

```
clear all, close all, clc

% grunddata
x = 1; y = 2; z = 3;

disp('     x     y     z')
disp([x y z])        % Skriver ut talen i variablerna x, y och z

if (x<y) && (y<z)
    x = z;           % Byter plats på det minsta och största talet
    z = x;
elseif (x>y) && (y>z)
    x = z;           % Byter plats på det största och minsta talet
    z = x;
else
    disp('Talen är osorterade och ingen förändring görs!')
end

disp('     x     y     z')
disp([x y z])        % Skriver ut talen i variablerna x, y och z
```

Uppgifterna nedan är tänkt att visa vilka steg man kan ta för att förbättra programmet ovan. Det är också ett exempel på hur man i varje steg arbetar med ett litet och väl avgränsat problem samt hur man löser och testar det. En bra hjälp för att lyckas med att förbättra ett program är att följa programflödet och också kontrollera variablers värde genom att skriva ut dem.

OBS! Följ instruktionerna nedan!

1. Tanken med programmet

Tanken med programmet är att det ska vända (reversera) ordningen på tre sorterade tal. Talen kan vara skrivna i stigande eller fallande ordning.

Talen 1 2 3 (= grunddata) är sorterade i stigande ordning och programmet ska då ändra till avtagande ordning och skriva ut resultatet 3 2 1. På liknande vis är det för tal i avtagnade ordning som då reverseras och skrivs ut i stigande ordning. För tre osorterade tal skriver programmet endast ut en textsträng och de tre talen.
Titta igenom koden noga och försök förstå vad som är tänkt (vilket tyvärr inte är samma sak som det som verkligen händer när programmet körs.) Ändra inte i koden ännu! Glöm inte bort att använd Matlabs hjälp och Matlabhäftet vid oklarheter!

2. Programflöde och variabelutskrift

Kör programmet! Det önskade resultatet är 3 2 1 men programmet ger 3 2 3. Varför och hur har det gått till?
Då man följer programflödet ser man att det första if-villkoret är sant eftersom talen är sorterad i stigande ordning. Det är alltså i första if-satsen som felet uppstår.
Undersök noggrannare och gör variabelutskrifter genom att ta bort semikolonen i de två tilldelningssatserna. Kör programmet igen och se nu hur variablerna x och z uppdateras. Korrigera felet, behöver du tips så finns det skrivet på allra sista sidan. Provkör efter korrigering och kontrollera att du får önskat resultat. Om du inte får önskat resultat; gå tillbaka och se över vad du gjort. Har du skrivit rätt eller är din ide' felaktig, behöver du tänka om?

3. Åtgärda fel

Programmet skall även fungera för tal i avtagande ordning. Ändra därför värden på variablerna till $x = 3$, $y = 2$ och $z = 1$ (= grunddata i filen). Talen är nu sorterade i fallande ordning. Kör programmet! Det önskade resultatet är 1 2 3 men programmet ger 1 2 1. Vad händer?
Arbeta på samma vis som du gjort i punkt 2 ovan.

4. Koduppprepning

De båda if-satserna har identisk kod. Detta kallas koduppprepning och är en källa till fel varför det ska tas bort. Ersätt de två if-satserna med **en** där det nya villkoret är en kombination av de tidigare. Tips finns på allra sista sidan om du behöver! Provkör efter korrigering och kontrollera att du får önskat resultat. Om du inte får önskat resultat; gå tillbaka och se över vad du gjort. Har du skrivit rätt eller är din ide' felaktig, behöver du tänka om?

5. Användarvänlighet

För att göra programmet mer användarvänligt ska programmet ändras så att användaren kan mata in tal till variablerna. Ett tips är att använda *input* till varje inmatning, använd *help* för att ta reda på hur den fungerar. Provkör efter korrigering och kontrollera att du får önskat resultat. Om du inte får önskat resultat; gå tillbaka och se över vad du gjort. Har du skrivit rätt eller är din ide' felaktig, behöver du tänka om?

6. Testkör och korrigera!

Programmet fungerar nu för sorterade tal men vad händer om man skriver in följande värden $x = 2$, $y = 2$ och $z = 1$ och $x = 2$, $y = 2$ och $z = 2$? Fungerar det för en osorterad följd? Testkör med egen data och korrigera när du upptäcker fel! Provkör efter korrigering och kontrollera att du får önskat resultat. Om du inte får önskat resultat; gå tillbaka och se över vad du gjort. Har du skrivit rätt eller är din ide' felaktig, behöver du tänka om?

IV. Redovisning och ytterligare uppgifter

Vid redovisningen ska du visa att du:

* förstår hur variabler skapas, uppdateras och används
* kan skapa, uppdatera och hämta data ur vektorer och matriser.
* kan beräkna och plotta t.ex. funktioner.
* kan skriva script och förstå hur de fungerar.
* förstår hur if-satser och villkor fungerar.

Vid redovisningen väljer du själv en uppgift från del I och en från del II som du vill redovisa. Handledaren kommer också att välja en uppgift från del I och en från del II som du ska redogöra för. Hela del III redogör du för.

Dokumentera dina erfarenheter noggrant så att du lätt kan använda detta om du behöver repetera dina Matlabkunskaper.

Idag den ..... har .....

med personnummer .....

godkänts av .....

1. Rekommenderade uppgifter

Ändra i programmet *reverseOrder.m* så att man läser in tre tal direkt i en vektor istället för till variablerna x, y och z. Gör nödvändiga ändringar i programmet.

Ändra i programmet *reverseOrder.m* så att man läser in tre tal direkt i en vektor. Programmet sorterar de tre talen. Den osorterade och den sorterade vektorn skrivs ut. Gör nödvändiga ändringar i programmet.

2. Extra uppgifter

Skriv om programmet *reverseOrder.m* så att man kan skriva in ett godtyckligt antal tal. Du bestämmer hur inläsningen av talen ska avslutas. Talen lagras i en vektor. Programmet ska inte sortera talen. Matlabs funktioner som reverserar ordningen får inte användas!

3. Tips

Uppgift III. 2 och 3.

Använd en variabel till. Med hjälp av den kan man göra en mellanlagring.

Uppgift III. 4.

Tänk på att talen antingen är sorterade i stigande *ELLER* avtagande ordning, dvs det finns två villkor som sätts samman.

Allmänt.

Bli mästare på att hitta i Matlabs hjälp och att läsa och tolka felmeddelanden. Detta spar tid och minskar frustration!