DATORÖVNING 1 — BLI BEKANT MED MATLAB

1. Instruktioner

Den här laborationen innehåller de absoluta grunderna i Matlabanvändning. Det är saker som man verkligen måste kunna i fortsättningen. Se därför till att du förstår vad du gör. Ta alla frågor på allvar. Tänk efter vad du tror kommer hända före du trycker enter. Varje gång. Se till att förstå vad som hände om något annat hände. I den här laborationen står det hela tiden vad du ska göra utom i de allra sista uppgifterna men i de kommande laborationerna kommer problemlösning att vara en viktig del. Då kommer de här kunskaperna hjälpa.Det gäller att få grepp om den struktur som Matlab är uppbyggd av samt att lära sig så mycket kommandon som möjligt. För att bli en effektiv Matlabprogrammerare så behöver man ha en stor mängd utantillkunskap. Men man kan inte kunna allt och på fyra laborationer kommer vi bara nosa i Matlabs digra filkataloger. Därför hoppas vi kunna ge er verktyg för att gå vidare på egen hand när så behövs. Det gör de övningar som handlar om att söka och läsa hjälpfiler oerhört viktiga.

Det ni inte hinner med ska ni göra hemma till nästa laboration. Det går bra att komma till Fredrik Lindgrens rum MV:L2104 och fråga om ni stöter på problem.

2. Uppgifter

2.1. **Lär känna Matlab.**

2.1.1. *Starta Matlab.* Börja med att försöka lista ut hur man startar Matlab på den datorn du sitter och gör sedan det.

2.1.2. *Skapa kataloger.*

- Titta vilken katalog du befinner dig i under *current folder*.
- Flytta dig till din användares rot.
- Skapa en kurskatalog om du inte redan har det genom att skriva `mkdir('InlAnalys')` eller vad du nu vill kalla din katalog. Använd inte mellanrum. Det kan ställa till problem i Matlab. Det kan möjligen även å, ä och ö. Låt alltid katalognamn liksom filnamn börja med bokstäver.
- Flytta dig till katalogen du just ställde dig i genom att skriva `cd InlAnalys` vid prompten.
- Skapa en katalog med namnet Matlab på samma sätt som ovan. Flytta dig till den katalogen och skapa sedan en katalog med namnet Lab1. Ställ dig i den.

2.1.3. *Matlabmiljön.* Studera kapitel 2-3 i *Matlab - En kort handledning* som finns att ladda ned på kurshemsidan. Bekanta dig med Matlabmiljön. Se till att du vet var du hittar *Command Window*, *Command History*, *Work Space* och *Current Directory*. Titta igenom rullistor och allt som nu kan vara av intresse.

2.1.4. *Kommandofönstret.* I kommandofönstret kan man vid prompten skriva olika kommandon som *exekveras* när man trycker enter. Prompten är helt enkelt där man skriver in ett kommando och ser ut så här

```
>>
```

i Matlab. Om man skriver `sin(2)` och trycker enter så räknas $sin(2)$ ut samtidigt som variabeln *ans deklareras* (skapas) och *tilldelas värdet* $sin(2) \approx 0,9093$. Om man istället skriver `y=sin(2)` så deklareras en variabel y som får det angivna värdet.

- Skriv `clear` vid prompten. Kommandot `clear` tar bort alla variabler ur minnet och samtidigt även från arbetsytan (Work Space). Titta på arbetsytan och se att den är tom.
- Skriv `sin(2)` vid prompten och tryck enter.

Date: 2010–09–13, Fredrik Lindgren, Matematiska vetenskaper, Chalmers tekniska högskola.

- Titta på arbetsytan. Har något förändrats?
- Skriv *ans* vid prompten och tryck enter. Vad händer?
- Skriv `y=sin(2)`. Titta på arbetsytan igen. Något nytt?
- Skriv `z=sin(2);` vid prompten och tryck enter. Vad gör semikolonet?
- Titta vad som har hänt i fönstret kommandohistoria!
- Skriv `clear` och dubbelklicka på någon av raderna i kommandohistorien. Vad händer?
- Se till attt kommandofönstret är aktivt och tryck ”pil upp”. Vad hände? Tryck ”pil upp” igen. Tryck enter. Skriv y och tryck ”pil upp”. Det senare skriver ut det senaste kommandot som börajde på y. Trycker man på pilen igen så får man det näst senaste o.s.v. Det här tricket underlättar ganska mycket i många fall!
- Skriv nu `minVar1=[3 5 8]` och tryck enter. Skriv därefter `min` och tryck på tab-tangenten. Nu får du upp en lista med alla inbyggda funktioner som börjar på min samt de variabler som deklarerats och har ett namn som börjar på min. Bläddra ner till `minVar` och tryck enter två gånger.

2.2. **Matlab som miniräknare.**

2.2.1. *De grundläggande räkneoperatorerna.* De fyra räkensätten i Matlab utförs med /, *, + och -. Även \ är definierat. Genomför följande:

```
>>a=4
>>b=2
>>a+b
>>a-b
>>a*b
>>a/b
>>a\b
```

2.2.2. *Matriser.* Tills vidare så ska vi bara se *matriser* som listor med tal eller annat men de är också viktiga och användbara matematiska objekt med definierade räkneregler som ni kommer lära er i kursen i linjär algebra under läsperiod 3.

- Läs kapitel 1 och kapitel 5 i *Matlab - en kort handledning*.
- Skriv A=[1 2 3;4 5 6; 7 8 9] vid prompten och tryck enter.
- Genomför följande sekvens och försök att förutsäga vad som kommer att hända.

  ```
  >>A(2)
  >>A(3)
  >>A(4)
  >>A(7)
  >>A(1,3)
  >>A(2,2)
  >>A(:,2)
  >>A(1,:)
  >>A(2:3,2:end)
  >>A(3,4)
  >>A(10)
  ```

  Vad händer? Det är uppenbarligen så att Matlab har två sätt att numrera elementen i matriser. Dels med syntaxen `A(n,m)` som ger elementet i rad n och kolumn m och dels med `A(k)` där element 1 är det högst upp till vänster och sedan räknar man nedåt tills man kommer till slutet varvid nästa element är det som ligger högst upp i den andra kolumnen o.s.v.
- Ett specialfall av matriser är vektorer. Radvektorer med punkter fördelade över ett intervall kommer att vara viktiga för oss. Ett sett att skapa sådana är med syntaxen *vänstergräns:steglängd:högergräns* eller för den delen *högergräns:-steglängd:vänstergräns*. Testa följande och försök förutsäga vad som kommer att hända:

```
>>0:1:10
>>10:-1:5
>>-pi:0.5:pi
```

Ett annat sätt att skapa sådana matriser är med kommandot `linspace`.

```
>>linspace(a,b,N)
```

ger N st punkter på jämnt avstånd där det första är a och det sista är b. Testa följande:

```
>>v1=linspace(0,1,50)
>>size(v1)
>>v2=linspace(0,1.5,10)
>>size(v2)
>>v3=linspace(0,1.5)
>>size(v3)
```

Vad bör dt vara för att `0:dt:pi` ska göra samma sak som `linspace(0,pi)`? När vill man använda den ena syntaxen respektive den andra?

Varför är då dessa kommandon viktiga? Testa följande:

```
>>x=linspace(-pi,pi);
>>y=sin(x);
>>plot(x,y)
```

När man ger en vektor som argument till `sin` så räknar den ut sinusvärdet för varje element i den vektorn och returnerar en ny vektor med samma storlek där `y(i)=sin(x(i))`. Vi ska titta mer på plot-kommandot i nästa laboration!

- Några olika speciella matriser man kan skapa:

```
>>ones(2,3)
>>zeros(3,7)
>>zeros(5)
>>rand(2,2)
>>eye(5)
```

Ändra eventuellt på siffrorna och se till att du förstår vad de gör!

Man kan och vill ofta ändra i matriser. Skapa en 3×4-matris, A, med bara nollor. Testa följande kommandosekvens. Tänk igenom vad du tror ska hända innan du trycker enter och vad som egentligen hände om det du trodde skulle hända inte hände. Om du får ett felmeddelande, ändra kommandot så lite som det går så att det fungerar!

```
>>A(1,2)=7
>>A(1,:)=[3 8 1 2]
>>A(end,3:4)=1:2
>>A(:,end)=[]
>>A(1,:)=[3 8 1 2]
>>A(1,:)=10
>>A(1,2:1:3)=[2,4,5]
```

matrisen A är nu kvadratisk. Tänk att vi vill räkna ut kvadraten på varje elemet a_{ij} i A. Testa följande sätt:

```
>>A*A
>>A.*A
>>A^2
>>A.^2
```

Vi kan också tänkas vilja dela varje element i matrisen med något. Skapa två slumpmässiga matriser

```
>>A=rand(4)
>>B=rand(4)
```

Testa också:

```
>>A/B
>>A./B
>>A./A
>>A/A
```

När vi vill utföra elementvisa operationer i Matlab båste vi alltid använda punkt före operatorerna multiplikations-, divisions- och potensoperatorn. Det beror på att dessa operationer är definerade på annat sätt för matriser. Se *Matlab - en kort handledning*.

2.3. **Söka och få hjälp.** Det finns massor av färdiga funktioner i Matlab och det är en bra idé att tidigt lära sig att hitta bland dessa och att kunna läsa sig til hur de ska användas. Vi har tidigare sett att det finns en funktion som heter `rand`. Om vi vill ta reda på vad denna funktion gör så skriver man

```
>>help rand
```

vid kommandoprompten. Gör det och försök att förstå det första stycket i hjälptexten. Testa också att skriva `doc rand`. Notera i synnerhet att dokumentationen slutar med några exempel. Försök förstå det första!

Antag nu att vi inte vet hur man ska beräkna tangens av ett tal. Vi vet att tangens heter *tangent* på engelska. Med kommandot `lookfor` kan vi söka efter alla Matlabfunktioner som innehåller ett visst ord eller en textsnutt i den första raden i hjälptexten. Så för att söka efter tangens skriver vi

```
>>lookfor('tangent')
```

eller eftersom det bara är ett ord

```
>>lookfor tangent
```

Gör detta. Skulle man inte hitta något kan man testa med

```
lookfor 'ord1 ord2 ...' -all
```

Då söks hela första stycket igenom efter den sträng man givit inom de enkla citationstecknen. Testa samtliga skrivsätt. En av alla träffar är funktionen `tan`. Eftersom vi känner igen det som en förkortning av tangens så verkar det som det skulle kunna vara den funktionen som gör vad vi vill. Därför läser vi mer om den genom att skriva `help tan`.

2.3.1. *Lite uppgifter.*

- Testa alla kommandona ovan och läs hjälptexten för `tan`.
- Hur gör man om man vill räkna ut tangens av ett tal som är givet i grader?
- Kolla också dokumentationen!
- Varför står det +0.01 och -0.01 på den första raden kod i exemplet?
- Kontemplera dessutom lite över det som står under rubriken **Algorithm**. Det är obegripligt men antyder att sättet som tangens räknas ut på är viktigt. Varför då tror du?

2.3.2. *Bläddra i dokumentationen.* Ett annat sätt att leta reda på nya saker är att bläddra i dokumentationen. Gör Exempel 9 i *Matlab - en kort handledning*. Istället för att klicka dig fram till dokumentationen kan du skriva `doc` vid prompten.

2.3.3. *Olika kataloger/bibliotek.* Om man enbart skriver `help` så får man en lista över olika bibliotek. Titta på de som heter något på formen `matlab/biblioteksnamn`. Du kan se vilka filer dessa innehåller genom att klicka på namnet i listan. Alternativt kan man skriva `help biblioteksnamn`. Titta på biblioteken `ops`, `elfun`, `elmat`, `polyfun` och `matfun`. Notera funktionen `rref` i den sista katalogen. Den är mycket användbar!

2.4. **Några nyttiga blanduppgifter.**

2.4.1. *Kryss- och skalärprodukt.* Skapa två vektorer i $\mathbb{R}^3$ som vare sig är parallella eller ortogonala. Leta reda på funktioner som beräknar kryssprodukter och skalärprodukter. Testa dem på dina vektorer. Bekräfta den antikommutativa lagen för kryssprodukt samt det faktum att kryssprodukten är ortogonal mot båda de ursprungliga vektorerna. Detta är egenskaperna (ii) och (vi) på sidan 579 i RA.

2.4.2. *Lösa linjära ekvationssystem.* Använd funktionen `rref` för att lösa något linjärt ekvationssystem med fyra eller fler obekanta och lika många ekvationer. Skapa först en *koefficientmatris* $A \in \mathbb{R}^{4\times 4}$ och ett högerled $b \in R^3$. Skapa sedan en *utökad koefficientmatris* genom

```
>>C=[A b]
```

Testa också kommandot `A\b`. Vad gör detta?

2.5. **Extrauppgifter.**

2.5.1. *Trigonometriska funktioner.* Rita de trigonometriska funktioner du känner till på intervallet $[-\pi, \pi]$ i samma figur. Tips:

```
>>plot(x,y1)
>>hold on
>>plot(x,y2,'r')
```

Sätt dessutom ut en figurtitel med `title` samt förklaringar av graferna med `legend`. Skriv x på x-axeln och y på y-axeln med hjälp av `xlabel` och `ylabel`.

2.5.2. $A\sin(\omega(t-\phi))$. Gör samma sak som i ovanstående uppgift men med sinusfunktioner med olika fas, frekvens och amplitud.

2.5.3. *Monom.* Rita några olika monom i samma figur, dvs

$$y_n(x) = x^n, \quad n = 0, 1, 2, \ldots, N$$

på intervallet $[-2, 2]$.

2.5.4. *n:te rötter.* Gör om det hela men för n:te rötterna av x, dvs

$$y_n(x) = x^{1/n}, \quad n = 0, 1, 2, \ldots, N$$

Pass på här! Något är lurt!