Laboration 1
Matematisk statistik för K, TMA074

1 Introduktion

Syftet med den här laborationen är att ge en introduktion till hur vi kan använda Matlab för att utföra många beräkningar vi än så länge gjort för hand i kursen. Vi kommer också att titta på simulering av slumptal och centrala gränsvärdessatsen.

Nedan kommer en mycket kort introduktion till Matlab, om du vill ha en mer grundläggande introduktion finns rekommenderad litteratur på kurshemsidan. Om du redan är bekant med Matlab kan du hoppa över resten av den här sektionen och gå direkt till Sektion 2 där vi börjar med Matlabs statistikverktyg.

1.1 Matlab

Matlab, som är ett program för numeriska beräkningar och visualisering, kommer att användas under alla laborationer i kursen. Till att börja med kan vi använda Matlab precis som en miniräknare:

```
>> (5+7^2 - 15)/6
ans =
    6.5000
```

Vi kan även spara variabler i minnet:

```
>> a = 5;
>> b = 7;
>> c = 15;
>> d = (a+b^2 - c)/6
d =
    6.5000
```

Om vi inte vill att tilldelningen eller uträkningen skrivs ut så avslutar vi raden med ';'. Vi kan även använda kommandot `who` för att se vilka variabler vi har sparade i minnet och kommandot `clear` för att radera variabler ur minnet. Till exempel ser vi vilka variabler vi har i minnet och raderar variabeln `d` genom att skriva

```
>> who
Your variables are:
a  b  c  d
>> clear d;
```

En av Matlabs styrkor är dess förmåga att enkelt hantera vektorer och matriser. Vi konstruerar matriser och vektorer genom att använda hakparenteser. Semikolon används för att separera rader i matriser och ' används för att transponera en vektor eller matris:

```
>> x = [1 2]
x =
     1     2

>> y = [3 4 5]'
y =
```

```
     3
     4
     5
>> z = [1 2;3 4]
z =
     1     2
     3     4
```

En annan styrka med Matlab är dess visualiserings-funktioner. Vi plottar till exempel en vektor mot en annan genom att skriva

```
>> x = [1 2 3 4 5];
>> y = x.^2 - 1;
>> plot(x,y)
```

Här betyder `x.^2` att vi tar varje element i `x` upphöjt till 2. Vi kan även specificera hur vi vill att figuren ska se ut:

```
>> plot(x,y,'r.')
```

plottar röda punkter istället för den blå kurvan som är standard. För fler valmöjligheter, se

```
>> help plot
```

Slutligen kan vi precis som i andra programmeringsspråk skriva egna funktioner. Detta görs i stort sett genom att man skriver de vanliga Matlab kommandona i en så kallad m-fil. Om vi skapar en m-fil som heter `lab1.m` som innehåller

```
x = [1 2 3 4 5];
y = x.^2 - 1;
plot(x,y)
```

Kan vi sedan köra de tre raderna direkt genom att bara skriva

```
>> lab1
```

i Matlab. Om vi vill kunna köra denna plot-rutin för olika vektorer `x` kan vi istället göra en funktion av filen. Vi skriver då i filen `lab1.m`

```
function y = lab1(x)
y = x.^2 - 1;
plot(x,y)
```

Funktionen tar `x` som argument och returnerar `y`. Vi kan nu använda funktionen för olika vektorer genom att skriva

```
>> x = [1 4 7 8];
>> y = lab1(x);
```

2 Statistics toolbox

Statistics toolbox är en programpaket till Matlab som innehåller många användbara statistikfunktioner. För att se alla kan du skriva

```
>> help stats
```

De funktioner vi framförallt kommer använda är de som heter ...pdf, ...cdf och ...inv, vilka används för att beräkna täthetsfunktioner, fördelningsfunktioner och kvantiler till fördelningar. Vi kommer även använda ...rnd som används för att simulera slumptal från givna fördelningar. Dessa funktioner finns implementerade för en mängd olika fördelningar, några vanliga anges i tabellen nedan.

fördelning	$f(x)$ eller $p(k)$	$F(x)$	$F^{-1}(x)$	slumptal
$\mathsf{N}(\mu, \sigma^2)$	`normpdf`(x, μ, σ)	`normcdf`(x, μ, σ)	`norminv`(x, μ, σ)	`normrnd`(μ, σ)
$\exp(\mu)$	`exppdf`(x, μ)	`expcdf`(x, μ)	`expinv`(x, μ)	`exprnd`(μ)
$\mathrm{Bin}(n, p)$	`binopdf`(k, n, p)	`binocdf`(k, n, p)	`binoinv`(k, n, p)	`binornd`(n, p)

Använd `help` följt av funktionsnamnet för att få mer information om funktionerna. Till exempel kan vi specificera hur många slumptal vi vill ha i ...rnd-funktionerna.

Här följer några enkla uppgifter som ni ska lösa genom att använda funktionerna ovan. Kom ihåg att använda `help` om ni är osäkra på något!

1. Om $X \sim \mathsf{N}(5, 10)$, vad är $\mathsf{P}(X < 3)$?
2. Om $X \sim \exp(5)$, vad är $\mathsf{P}(X < 3)$?
3. Om $X \sim$Bin$(100, 0.5)$, vad är $\mathsf{P}(X > 49)$?
4. Plotta täthetsfunktionen och fördelningsfunktionen för en $\mathsf{N}(5, 2)$-fördelad slumpvariabel. Tips: x kan vara en vektor i argumentet till ...pdf och ...cdf, funktionen `linspace` är användbar.
5. Använd `norminv` för att beräkna 5%, 10%, 90% och 95%-kvantilerna för en $\mathsf{N}(45, 2)$-fördelad slumpvariabel.

3 Centrala gränsvärdessatsen

Vi ska nu undersöka centrala gränsvärdessatsens påstående genom att simulera slumpvariabler. Vi gör detta genom att använda exemplet vi tog upp under föreläsning 5, där vi kastar en tärning n gånger och tittar på fördelningen för medelvärdet av tärningarnas värden.

Vi kan använda funktionen `randi` för att simulera tärningskast (se hjälptexten till funktionen!):

```
>> randi(6)
```

drar likformigt ett tal från $\{1, 2, 3, 4, 5, 6\}$. Vi kan beräkna medelvärdet av 10 tärningskast via

```
>> m = mean(randi(6,10,1))
```

Vi vill nu upprepa detta N gånger så att vi kan titta på fördelningen för m. Vi kan göra detta genom att skapa en vektor $\mathbf{m}$ som innehåller N medelvärden enligt

```
>> m = mean(randi(6,10,1000));
```

där vi har valt att upprepa försöket $N = 1000$ gånger.

Vi visualiserar nu fördelningen genom att räkna antalet medelvärden som hamnade på varje möjligt värde av tärningskasten:

```
>> x = (10:60)/10;
>> for(i = 1:length(x)); f(i) = sum(m==x(i)); end
>> stem(x,f/N);
```

Fördelningen liknar en normalfördelning. Detta kan också ses om vi istället plottar ett histogram:

```
>> histogram(m,'Normalization','probability')
```

Här anger `'Normalization','probability'` att höjden av boxarna ska skalas med antalet observationer, så att vi får något som liknar en sannolikhetsfördelning. Om detta kommando inte anges blir höjden istället lika med antalet observationer i varje box.

För att se att fördelningen verkligen liknar en normalfördelning kan vi samtidigt som vi ritar ut histogrammet också rita ut motsvarande täthetsfunktion för normalfördelningen. Detta görs med kommandot `histfit`:

```
>> histfit(m,20,'Normal')
```

Här anger `20` att 20 boxar ska användas i histogrammet, och `'Normal'` att täthetsfunktionen för en normalfördelning ska ritas ut. I hjälptexten till funktionen listas en rad andra fördelningar som också kan användas. Slutligen kan vi också testa om normalfördelningen passar bra genom att rita ett normalfördelningsdiagram:

```
>> normplot(m)
```

Gör om försöket för andra antal tärningskast, det vill säga byt 10 kast mot både fler och färre och se hur resultaten ändras. Hur många kast måste vi göra för att en normalfördelning ska vara en bra approximation av fördelningen?

4 Projektuppgift

Utför den första delen på projektet.