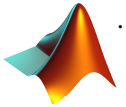
TSRT04: Introduktionskurs i Matlab Datorlektion 1

Version: 29 augusti 2017

1 Välkommen till MATLAB

Denna lektion introducerar några viktiga delar av MATLAB – men du kommer inte långt om du inte söker upp mer information på egen hand. Använd hjälpsystemet i MATLAB och sök på internet för att hitta mer information.

Observera: Syftet med denna datorlektion är att du ska lära dig grunderna i MATLAB genom att prova dig fram. Det finns mängder av frågor i detta dokument och vi rekommenderar att du skriver ner dina svar och reflektioner. Då blir det lätt för dig att gå tillbaka om du glömt något. Däremot behöver du inte presentera resultatet för någon. Assistenterna är här för att svara på frågor och ge återkoppling på din kod. Passa på – det finns inga dumma frågor!

För att logga in på ISYs datorer så behövs ett studentkonto och lösenord. Kontakta support (`support@isy.liu.se`) ifall du behöver ett nytt konto. När du loggat in så startar du MATLAB[1] genom att klicka på ikonen  .

Då öppnas ett MATLAB-fönster. Det innehåller olika delar som kan ha olika ordning beroende på inställningar. Följande är de tre viktigaste delarna:

- **Workspace**: Här listas de variabler som du skapat (inga än så länge).
- **Command History**: Här finns en historik över dina kommandon.
- **Command Window**: Det här är det viktigaste fönstret. Det är här som du skriver in de kommandon som du vill MATLAB ska köra.

[1] Beroende på vilket operativsystem som körs kan det finnas flera olika sätt att starta MATLAB. Fråga din lektionsassistent om du får problem.

2 Förberedelseuppgifter

Innan första lektionen ska du göra följande:

1. Närvara vid den första föreläsningen eller läsa igenom föreläsningsbilderna noggrant.
2. Se några ”MATLAB video tutorials”:
   `http://youtu.be/tqjZ80PwqBU?list=PL7CAABC40B2825C8B`

   Det rör sig om 8 korta videor (totalt 45 minuter).
3. Börja arbeta på avsnitt 3–4 i detta dokument. Försök att slutföra så mycket som möjligt av övningarna. Se till så att du förstår vad du gör och vad de olika kommandona gör. Samla på dig frågor som du kan ställa till lektionsassistenten under lektionen!

3 Hjälpsystemet i MATLAB

MATLABs dokumentation når du genom att skriva kommandot

```
>> doc
```

i Command window. Prova själv och se vad som händer – varje gång som ett kommando nämns i detta dokument så ska du prova det själv!

Om du vill ha en dokumentation med ett mer grafiskt gränssnitt så prova online-versionen:

`http://www.mathworks.se/help/matlab/index.html`

Kommandot

```
>> help command
```

skriver ut en kort hjälptext om kommandot `command` (`command` ska alltså bytas ut mot det kommando som du vill veta mer om). Ta för vana att köra `help` på alla nya kommandon. Prova det till exempel på sig självt: `help help`

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

.....

.....

.....

.....

.....

4 MATLAB som räknedosa

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

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

Undersök skillnaden mellan följande rader:

```
>> 14/7*2
>> 14/(7*2)
```

Varför ger de olika resultat?

.....

Vad (och varför) blir resultatet av kommandot

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

.....

Andra avrundningskommandon är `floor` och `ceil`. Ta reda på hur de fungerar med hjälp av `help`.

.....

.....

.....

Vad är `ans` för något? .....

Tangentbordets upp- och ner-pilar kan användas för att bläddra i de kommandona som man tidigare skrivit, för att köra de igen och kanske köra mindre förändringar. 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 de tecknen.

Du kan också leta upp ”Command History” i MATLAB-fönstret. Där listas alla dina senaste kommandon. Du kan dra ett eller flera kommandon till Command Window, för att köra de igen och kanske ändra i några av de.

Elementära funktioner

Beräkna

$\cos(\pi/3) =$ ..... $\sin(\pi/4)$ .....

I vissa fall får man vara uppmärksam på att MATLAB har begränsad beräkningsprecision. Vad blir resultatet av

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

och hur ska detta tolkas?

.....

.....

Andra relaterade kommandon

Exponentialfunktioner (alltså ”upphöjt till”) kan skrivas med hjälp av `^` :

```
>> 2^4
```

Det finns en speciell funktion `exp` för att använda det naturliga talet *e* som bas:

```
>> exp(4)
```

Andra elementära funktioner är kvadratroten, `sqrt`, och olika logaritmer `log`, `log2`, `log10`:

```
>> sqrt(4)
>> log10(1000)
```

Komplexa tal kan skapas genom att använda `1i` eller `1j` som det imaginära talet. Andra relevanta kommandon för komplexa tal är `imag`, `real`, `abs` och `angle`. Undersök vad dessa gör!

Utmatningsformat

MATLAB visar normalt 4 decimaler för flyttal, men har betydligt fler decimaler i minnet. 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 kommandot med ett 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`.

Om du använder samma variabelnamn igen så får variabeln ett nytt värde. Exempelvis kan man skriva

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

Vill man ta bort variabler finns kommandot `clear`. Det kan användas som `clear a` för att ta bort variabeln `a` eller så skriver man `clear` för att ta bort alla variabler i Workspace. Att rensa bort variabler man inte använder minskar risken för att göra misstag. Vid mer omfattande beräkningar kan det också behövas för att få mer ledigt utrymme 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 på variablerna. Detta gör man smidigast genom så kallade MATLAB-skript, eller m-filer. En m-fil är helt enkelt en textfil med ett antal MATLAB-kommandon, som har filändelsen `.m` (samma filändelse används även för funktioner, som vi kommer till senare). Filnamnet blir ett kommandonamn som kan användas i MATLAB precis som vilket annat 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, men dessa är mindre smidiga eftersom MATLABs editor har en ”run”-knapp, kan hjälpa dig att hitta buggar i koden och kan rekommendera kodförbättringar.)

Skriv lite kod i din m-fil:

```
disp('Nu körs skriptet!');
student1 = 3;     % Betyg som student 1 och 2 hade på en tenta
student2 = 5;
medel = (student1 + student2)/2;  % Beräkna medelbetyget
disp('Resultat:');
display(medel);
```

Spara skriptet genom att välja Save eller Save As i File-menyn. Döp det 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 skriptet. Kommandot `dir` skriver ut innehållet i den katalog man är i, kommandot `cd` kan användas för att byta katalog och `pwd` talar om vilken katalog man befinner sig i. Du kan även byta katalog genom att klicka på en knapp i det övre vänstra hörnet av MATLAB-fönstret.

Du kan använda ditt skript 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 det igen. Glöm inte att spara efter ändringen.

Leta efter knappen ”run” i MATLABs editor. Tryck på den och se vad som händer.

6 Matriser

Matriser är den grundläggande datatypen i MATLAB. Faktum är att alla skalära siffor hanteras som matriser med dimensionen 1×1. Det finns mycket kraftfulla inbyggda funktioner för matrishantering och matrisberäkningar i MATLAB.

Skapa några matriser:

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

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

Undersök dimensionerna på dessa matriser med hjälp av funktionerna `size` och `length`. Vad är skillnaden mellan dessa båda funktioner?

.....

.....

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 även räkna med matriser:

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

Varför fungerar inte `A*c`? .....

Använd `.'` för att transponera en matris. (Enbart `'` ger ett 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)` ? .....
`>> G = zeros(1,4)` ? .....
`>> 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?

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

.....

.....

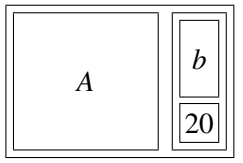
Indexering av delmatriser fungerar även för att tilldela värden till bara några av elementen:

```
>> A = ones(4,4)
>> A(1,1) = 9
>> A(2,3) = 4
>> A(3,2:4) = [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. att 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 operatorn. Prova

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

Vad är skillnaden på

```
>> A*B
```

och

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

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

. . . . . .

Gör ett skript som verifierar den trigonometriska ettan ($\sin^2(x) + \cos^2(x) = 1$ för alla x) med hjälp av komponentvisa operationer för olika x-värden, t.ex. $x = 0.5, 1, 1.5, 2, 2.5, 3, 3.5$ och 4. Hur ser din kod ut?

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

En vanlig användning av matriser är att representera linjära ekvationssystem. Dessa skrivs vanligtvis som $Ax = b$ där A är en känd matris och b är en känd vektor. Den obekanta vektorn x kan då beräknas genom att multiplicera från vänster med A:s invers, det vill säga $x = A^{-1}b$. I MATLAB kan detta beräknas som

```
>> x = inv(A)*b
```

Detta är dock inte det bästa sättet att lösa linjära ekvationssystem i MATLAB, eftersom inversen `inv(A)` kommer beräknas först och sedan multipliceras med `b`. Den numeriska osäkerhet som uppstår när `inv(A)` beräknas kan påverka lösningen. Dessutom tar det onödigt lång tid att beräkna `inv(A)` när `A` är en stor matris.

I grundkurser i linjär algebra så brukar vi lösa $Ax = b$ genom Gausselimination, utan att behöva beräkna matrisinversen. Vi kan låta MATLAB göra samma sak genom att skriva

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

Detta är båda snabbare och ger en mer pålitlig lösning till ett linjärt ekvationssystem.

Det är också värt att notera att beräkningen $x = A^{-1}b$ kräver att A är en kvadratisk och inverterbar matris. I andra fall (t.ex. för över- och underbestämda

ekvationssystem) kan systemet lösas i minsta kvadratmening. Detta sker automatiskt om man skriver `x = A\b`, medan `x = inv(A)*b` inte ens fungerar i dessa fall.

Lös följande ekvationssystem i MATLAB:

$$5x_1 + 2x_2 = 3$$
$$3x_1 + 2x_2 = 5$$

Svar: .....

.....

Fler matriskommandon

Du ska även testa följande kommandon, eftersom de kan bli användbara senare. Kommandona för dataprocessning är särskilt viktiga. Observera att flera av dessa kommandon kan ta flera inargument och returerna mer än bara en matris.

Speciella matriser: `rand`, `randn`, `diag`, `linspace`, `inf`.
Dataprocessning: `sort`, `max`, `min`, `find`, `sum`, `prod`.

7 Skriva egna funktioner

Förutom att spara skript som m-filer, för att kunna upprepa en hel kommandosekvens, så kan man också skapa egna funktioner i MATLAB. Till funktionerna kan man skicka med inargument precis som till en vanlig MATLAB-funktion, och funktionen kan returnera ett eller flera utargument.

Funktioner börjar alltid med en rad på formen

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

där `ut` anger vilken variabel som ska returneras från funktionen. `filnamn` är namnet på funktionen och den måste alltså sparas i en fil med namnet `filnamn.m`. `in1`, `in2` och `in3` är funktionens inargument. Detta är den information som användaren behöver ge till funktionen för att den ska kunna utföra sina beräkningar. Man kan ha godtyckligt många in- och utargument. Vill man ha två utargument skriver man

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

Uppgift 1. Nu ska vi skriva om vårt skript (där vi beräknade medelbetyget för två studenter) som en funktion:

```
function medel = medelbetyg(student1,student2)
% 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 = (student1 + student2)/2;  % Här beräknas medelbetyget
disp('Resultat:');
display(medel);
```

Spara den nya funktionen som `medelbetyg.m`.

Om vi jämför funktionen `medelbetyg` med motsvarande skript i avsnitt 5 så 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 som medelvärdet ska beräknas över. Om man 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 variabler med det workspace som skriptet anropats från. Funktioner skapar däremot sitt eget workspace och till en början innehåller detta workspace bara inargumenten. 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 exempelvis funktionen `mean` ser ut kan vi köra kommandot

```
>> type mean
```

Hur skulle vi kunna använda `mean` för att göra samma beräkning som i `medelbetyg`?

.....

.....

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

```
function ny_skuld = laneberakning(skuld,lon)
% ny_skuld = laneberakning(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 `laneberakning.m`. (Filen kommer att användas även under nästa lektion.) Funktionen anropas med

```
>> laneberakning(emmas_skuld,emmas_lon)
```

där `emmas_skuld` och `emmas_lon` är numeriska värden på Emmas skuld och lön, eller variabler som innehåller dessa värden.

Kolla hur stor Emmas skuld blir nästa månad, om lånet är på 10 000 kr, och Emma tjänar 25 000 kr/månad.

.....

8 Visualisering

En bild säger mer än tusen ord (som ordspråket lyder). Vad ritas av följande kod?

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

.....

Vad är skillnaden jämfört med

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

Varför blir följande figur inte lika bra och vad kan man göra för att kurvan ska se mer ut som kosinus?

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

.....

.....

Om man vill rita flera funktioner i samma figur kan man exempelvis använda

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

En annan möjlighet är att använda kommandot `hold`. Ta reda på hur det fungerar och plotta cosinus och sinus i samma bild. Din kod:

.....

.....

.....

.....

Uppgift 3. Följande uppgift är hämtad från en grundkurs i analys.

Rita kurvan $\frac{x^3}{x^2-2|x-2|}$ för en lämpligt val of x-värden. Ange alla extremvärden samt största och minsta värden, om de existerar. Du kan använda `max` och `min` för detta.

Gör en MATLAB-funktion som löser uppgiften numeriskt. Du måste skapa en vektor `x` över ett lämpligt intervall och med *tillräcklig* upplösning, evaluera funktionen i dessa punkter, plotta kurvan och sedan använda ett lämpligt kommando för att hitta extremvärdena.

Det är rekommenderat att du först skriver koden som ett skript (för att enkelt komma åt variablerna i Workspace ifall det blir buggar i koden) och därefter gör om det till en funktion. Kalla funktionen för `kurvplot`.

Några saker att tänka på:

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

.....

.....

.....

Sätt namn på figuren och axlarna genom att använda kommandona `title`, `xlabel` och `ylabel`.

Lämpliga inargument till `curveplot`-funktionen kan vara det intervall man vill rita kurvan för. Lämpliga utargument är 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 en rapport (eller liknande) kan vara mycket illustrativt. Detta kanske kommer vara det huvudsakliga skälet för att använda MATLAB i senare kurser.

Figurerna/bilderna kan sparas genom att välja Save As i File-menyn. Vilket format man ska välja beror på vilket program man vill använda figuren i. Några exempel på format är JPG för bilder som ska publiceras på internet eller användas i MS Word, medan EPS och PDF är användbara i LaTeX. Dessutom har MATLAB ett eget FIG-format så gör att man kan öppna upp figuren i MATLAB igen. Det är bra att alltid spara i FIG, utöver andra format, så att du senare kan göra ändringar i bilden (zooma, lägga till fler kurver etc.).

9 Hemuppgifter: Vecka 1-2

Allmänt

1. Installera MATLAB på din egen dator (om du har en).
2. Slutför de delar av detta dokument som du inte gjorde klart under första datorlektionen!
3. Nedan följer ett antal hemuppgifter som vi rekommenderar att gör under det första kursveckorna.

MATLAB som räknedosa

1. Utöver `round`, `floor` och `ceil`, så finns det ännu ett kommando för avrundningar: `fix`. Vad är skillnaden mellan `floor` och `fix`? Ledning: Jämför hjälpinformation för kommandona! Hur hanterar de negativa tal?

   .....

2. Använd `help log` eller någon annan hjälpfunktion för att ta reda på hur man beräknar följande:
   $\log(34) =$ ..... $\log_2(8) =$ .....

3. Använd `format long` för att besvara följande fråga: Är $\log(203.8)/\sqrt{9\pi}$ större än 1? .....
   Finns det andra sätt att ta reda på detta? .....

4. Hur många decimaler i π kan du hitta genom att ändra utmatningsformatet?

   .....

5. Vad gör kommandot `eps`? Beräkna `eps^2`, `1+eps` och `1+eps^2`. Förklara det oväntade beteendet.

   .....

   .....

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 (.MAT-filer), men du kan också spara de som text. Kommandot

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

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

   Töm Workspace och använd `load` för att återfå variablerna från din .MAT-fil.

Matriser

1. Skapa en diagonalmatris där diagonalen innehåller sifforna 1 1 2 3 5 8 13 och alla andra element är noll. Vilken funktion kan du använda för detta?

   .....

2. Skapa följande matris med hjälp av `reshape` och/eller `repmat`:

$$\begin{bmatrix} +1 & +1 & +1 \\ -1 & -1 & -1 \\ +1 & +1 & +1 \\ -1 & -1 & -1 \end{bmatrix}$$

   .....

   .....

3. Verifiera att determinanten uppfyller $\det(I+AB) = \det(I+BA)$ för alla matriser A och B med kompatibla dimensioner. Matriserna behöver inte vara kvadratiska. Du kan prova att låta A ha dimensionerna 4×3 medan B har dimensionerna 3×4. Du kan skapa slumpmässiga matriser av valfria dimensioner med kommandona `rand` and `randn`.

Visualisering

1. Verifiera att $\sin(t) = \frac{e^{it}-e^{-it}}{2i}$, för alla t, genom att rita $\sin(t)$ and $\frac{e^{it}-e^{-it}}{2i}$ i samma figur. Notera att i är det imaginära talet. Använd olika färger för att särskilja kurvorna.

2. Skapa 10000 slumpmässiga siffror (dvs. en matris med dimensionen 10000×1) med hjälp av `rand` eller `randn`. Studera fördelningen på de slumpmässiga numren genom att göra ett histogram med hjälp av `hist`. För att få en rimlig upplösning så kan du sätta antalet staplar till 50. Vad är skillnaden i slumpmässighet mellan `rand` och `randn`?

   .....

   .....

3. Skapa en 3D-visualisering av funktionen $\mathrm{sinc}(x^2+y^2)$ för $-10 \leq x \leq 10$ och $-10 \leq y \leq 10$. Du kan exempelvis använda funktionerna `sinc` och `meshgrid` för att evaluera funktionen. Rita resultaten med hjälp av `mesh` och `surf`. Vad är skillnaden mellan dessa kommandon? Hur kan du välja färgschema på grafen?

   .....

   .....