Introduktionskurs i MATLAB (TSRT04)

VT1 2020

Emil Björnson

Division of Communication Systems
Department of Electrical Engineering (ISY)
Linköping University, Sweden

www.commsys.isy.liu.se/en/student/kurser/TSRT04

Målet med kursen

Kursen ska ge grundläggande praktiska kunskaper om programmeringsspråket Matlab, som används till tekniska beräkningar. Kursen kan ses dels som en förberedelse till ett större antal kurser där Matlab används, dels som ett tillfälle att lära sig behärska ett kraftfullt ingenjörsverktyg.

Efter fullgjord kurs förväntas studenten kunna:

- *använda Matlab som miniräknare för både skalärer och matriser: Använda elementära funktioner och definiera variabler.*
- *konstruera enkla skript och funktioner.*
- *använda kontrollstrukturer (if-satser, for- och while-loopar) i enkla exempel.*
- *presentera resultat av beräkningar och åskådliga datamängder genom grafiska figurer.*
- *använda hjälpsystemet för att lära sig nya funktioner.*

Kursupplägg

- 1 föreläsning, 2×2 h lektioner, 5×2 h laborationer
  - Notera skillnaderna: 2 hp $\sim$ 53 timmar av studier!
  - 37 h av självstudier – betoning på ”egen utforskning”.
  - Kursmaterialet finns på kurshemsidan.
  - Ladda ner MATLAB från Studentportalen!
  - Ingen kursbok behövs – några bokförslag och videolänkar på hemsidan

- Lektioner och laborationer
  - Lära genom att prova – bästa sättet att lära MATLAB.
  - Arbeta i par: Hitta någon med liknande programmeringsbakgrund.
  - Vi förväntar oss att båda skriver och förstår koden.
  - Lektioner är frivilliga, laborationer är obligatoriska!

Anmälan till lektioner och laborationer

3 grupper

- Grupp A: Elin Jirskog
- Grupp C: Sara Rejgård
- Grupp D: Emil Boström

Anmälan

- Välj din grupp – detta bör du göra omedelbart!
- Anmälan till en lektionsgrupp (omfattar tvålektioner) och en labbgrupp (omfattar fem labbar)
- Anmälan görs i Lisam

Examination

1) Quiz

- Individuellt test om MATLABs grunder.
- Sker vid första labben. Examinerar lektionsmaterialet.

2) Plottuppgift

- Plotta och visualisera en datamängd på olika sätt.

3) ”Mini-projekt”

- Lös ett litet problem och visualisera lösningen.
- Obligatoriskt deltagande i laborationer tills projektet är godkänt (vi kommer föra närvaro!)

Generella riktlinjer

- Flera olika plottuppgifter och projekt (välj 1!)
- All undervisning är tillfällen att ställa frågor.
- Mellan schemalagd tid: Självstudier (5-6h/vecka)!
- Redovisning på engelska, endast vid labbtillfälle 3 och 5.

Svårighetsgrad

- Förkunskaper: Linjär algebra, Programmering
- Universitetskurser har *lärandemål*.
- Svårighetsgraden beror *alltid* på tidigare erfarenhet.
- Skillnaderna i programmeringserfarenhet är särskilt stor!

Glöm ej:

- MATLAB kommer vara användbar i många kurser.
- Det kanske det enda programmingsspråket du behöver!
- Om du är bra på Java/C/Python programmering – se till att du också blir en bra MATLAB-programmerare!

Vad är MATLAB?

MATrix LABoratory (MATLAB)

- Avancerade räknedosa för tekniska beräkningar
- Enkelt men kraftfullt programmeringsspråk
- Numeriska beräkningar (ej symboler som Mathematica)
- Tillgängligt för Windows, Mac, Unix, Linux, etc.
- Nya versioner två gånger per år: 2018a, 2018b, 2019a

- Bra: Enkelt att komma igång, enkelt att visualisera resultat
- Bra: Många exempel och verktygslåder för olika områden (t.ex. matematik, statistik, optimering, telekommunikation, reglerteknik, biologi, finans).
- Dåligt: Inte snabbast – men vanligtvis tillräckligt snabb!

- Bra för att testa idéer, lösa forskningsproblem, utveckla/validera algoritmer

- **Octave:** Öppen källkod – kompatibelt med MATLAB

Min forskning: 5G trådlösa kommunikation

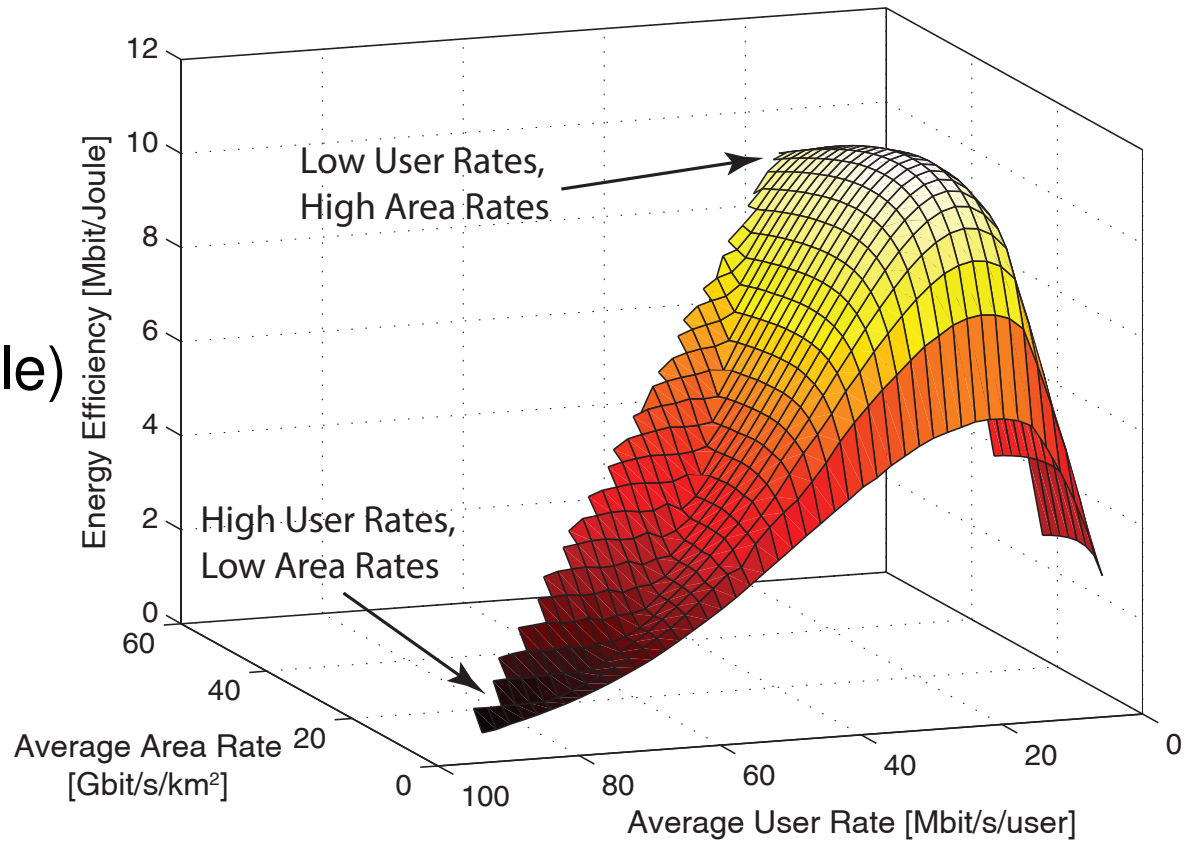
Mål: Utveckla designprinciper för nästa generations mobilnät.

Första samspelet mellan

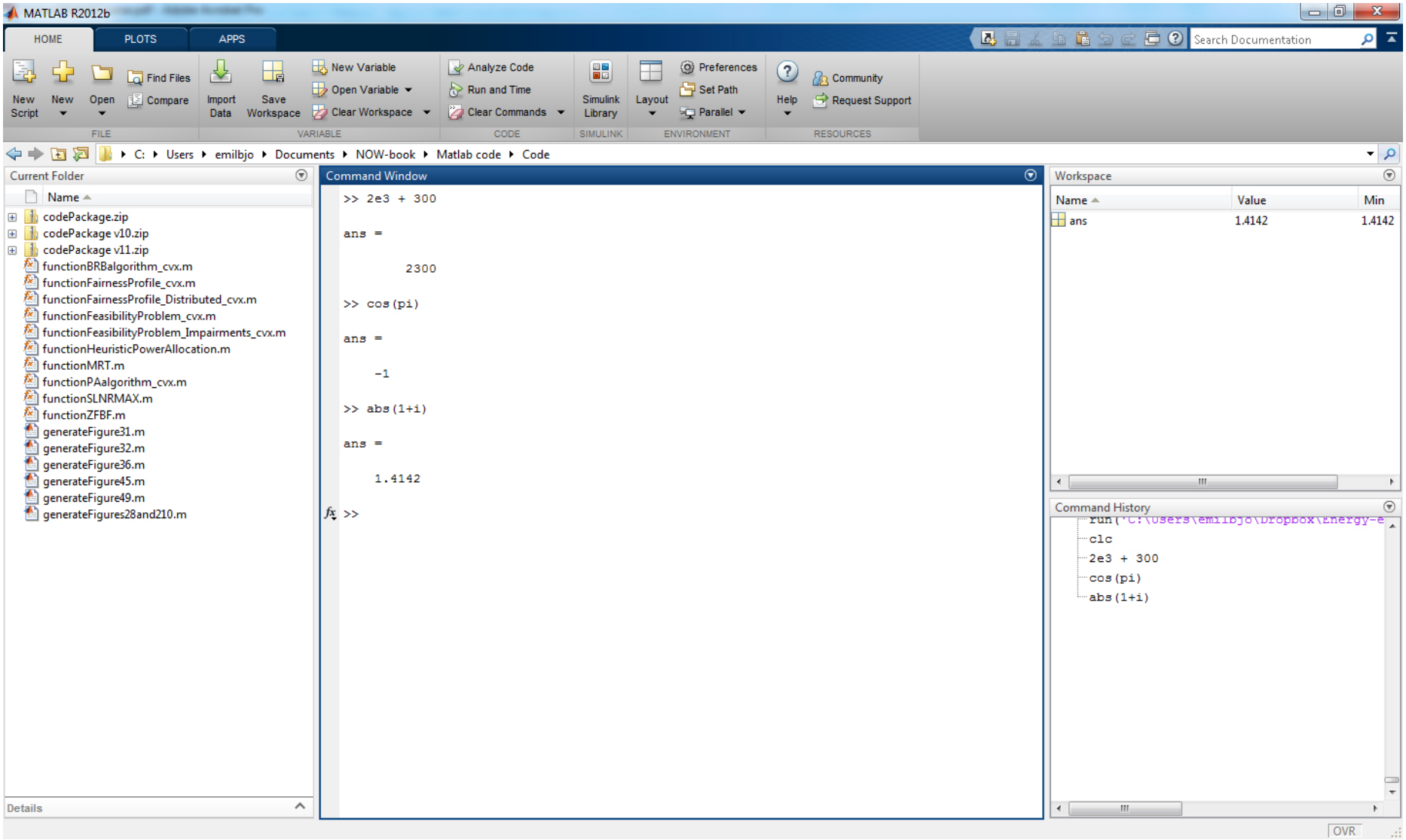
- Datatakt per användare (bit/s/user)
- Datatakt per areaenhet (bit/s/km^2)
- Energieffektivitet (bit/Joule)

Rollen för MATLAB:

- Testa modeller
- Utveckla algoritmer
- Visualisera avvägningar



MATLABs gränssnitt



MATLAB som en räknedosa

Använd *Command Window* som en avancerad räknedosa

- Vanliga nummer: `30`, `pi` (π), `1e2` ($1 \cdot 10^2$)
- Vanliga symboler: `+ - / *`
- Vanliga funktioner: kosinus ($\cos()$), absolutbelopp ($\mathrm{abs}(\cdot)$)

Exempel:

```
>> 2e3 + 300

ans = 2300

>> cos(pi)

ans = -1

>> abs(1+1i)

ans = 1.4142
```

Variabler

- En ”container” att spara nummer i.
- Har ett namn och ett värde.

```
>> a = 5

a = 5


>> b = a + 3

b = 8
```

(Det till höger om = beräknas först och resultatet sparas i `b`.)

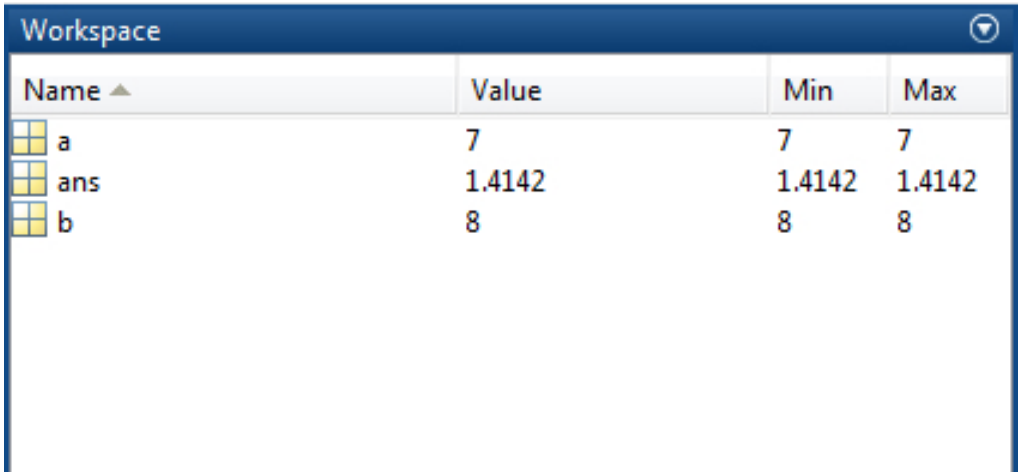
Vad blir resultatet av:

```
>> a = a + 2

a = 7
```

Workspace

Variabler sparas i ”Workspace”, som ett förvaringsfack.



Undersök ditt workspace

- Om du inte väljer ett variabelnamn: Resultatet sparas i `ans`
- Du kan klicka på variabler i Workspace för att få veta mer.
- Du kan lista alla variabler med kommandot `>>whos`.

Vektorer och matriser

Vektorer och matriser är grunden i MATLAB.

- $a = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$ skrivs som `>>a = [4 5 6]`
  (or `[4, 5, 6]`)
- $b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ skrivs som `>>b = [1; 2; 3]`
- $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ skrivs som `>>A = [1 2; 3 4]`

Dessa sparas i Workspace – precis som alla andra variabler:

Vektorer och matriser

- Semikolon skippar utskrift av resultatet av ett kommando:

```
>> a = [4 5 6];
```

- Matristransponat beräknas med `.'`:

```
>> a.'

ans =
       4
       5
       6
```

Skapa speciella matriser och vektorer:

- `>>C = eye(2)` ger $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.
- `>>x = 3:6` ger $x = \begin{bmatrix} 3 & 4 & 5 & 6 \end{bmatrix}$.
- `>>y = 2:3:11` ger $y = \begin{bmatrix} 2 & 5 & 8 & 11 \end{bmatrix}$.

Matrisoperationer

Ursprungliga syftet med MATLAB: Matrisoperationer

- Skapa matriser:

```
>> A = [1 2; 3 4];
>> B = eye(2);
```

- Beräkna multiplikation:

```
>> A*B

ans =
     1   2
     3   4
```

dvs. $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

(vanlig matrismultiplikation)

```
>> A.*B

ans =
     1   0
     0   4
```

dvs. $\begin{bmatrix} 1 \cdot 1 & 2 \cdot 0 \\ 3 \cdot 0 & 4 \cdot 1 \end{bmatrix}$

(elementvis multiplikation)

- Liknande: `^2` resp. `.^2`, och `/` resp. `./`

Matrisoperationer

Det finns massor av funktioner som hanterar matriser:

- Klassiska: `exp() log() sin() cos() tan()`
- Ordningsfunktioner: `min() max() mean() sort()`

Vissa funktioner arbetar elementvis:

```
>> x = 0:(pi/2):(2*pi)

x = 0    1.5708    3.1416    4.7124    6.2832

>> y = sin(x)

y = 0    1.0000    0.0000   -1.0000   -0.0000
```

Vissa funktioner processar alla element gemensamt:

```
>> z = max(x)

z = 6.2832
```

Indexering av matriser

Hur kommer man åt vissa element i vektorer och matriser?

```
>> y = [0 1 0 -1 0];
>> y(4)

ans = -1

>> A = [3 5 2; 7 8 6];
>> A(1,2)

ans =  5
```

$$A = \begin{bmatrix} 3 & 5 & 2 \\ 7 & 8 & 6 \end{bmatrix}$$

Hur hittar man en funktion?

Om du letar efter en funktion:

- Hur vet du om den existerar i MATLAB?
- ”lookfor term” söker i dokumentationen efter textsträngen ”term”
- Exempel: `lookfor determinant` letar efter en matris-determinant-funktion

Hur tar du reda på hur den fungerar?

- ”help command” visar hjälptext för ”command”
- ”doc command” ger mer ingående information

Generell dokumentation:

- ”doc” öppnar Matlab-dokumentationen
- ”help” visar en lista över ”toolboxes” (samlingar av funktioner för specifika ändamål)

Bortom räknedosan: Skript

- Ett sätt att köra flera kommandon flera gångar.
- Spara några kommandon i en m-fil (filnamnet måste sluta med `.m`) och kör alla genom att skriva namnet på filen som ett kommando.
- `>>edit` startar en redigerare som lämpare väl för att skriva m-filer.
- Dokumentation: Skriv kommentarer som `% Comment`

Stark rekommendation:

- Använd alltid skript!
- Enkelt att reproducera resultat och skriva dokumentation.
- Enkelt att göra små ändringar och köra om.

Exempel: Skript

Lina har sprungit 5 km på 23 min och 15 s.

- Hon vill beräkna tiden per km.
- Hon vill göra samma sak nästa vecka.

m-fil computeRunPace.m

```
distance = 5; % Distance in km
minutes = 23; % Total time expressed in
seconds = 15; % minutes and seconds

% Compute time per km in minutes:
totalminutes = minutes + seconds/60;
minperkm = totalminutes/distance
```

Skript respektive funktioner

Vad är skript?

- Bara en uppsättning kommandon.
- Använder MATLABs generella Workspace.
- Kan skriva över gamla variabler (överlappande namn).
- Kan råka använda gamla variabler (kodningsfel).
- Enklaste lösningen: Börja skript med `clear` som tömmer Workspace.

Vad är funktioner?

- Ett annat koncept: Har sina egna lokala Workspaces.
- Fungerar precis som MATLABs egna funktioner.
- Utmärkt sätt att återanvända samma kod flera gånger.

Exempel: Funktion

m-fil computeRunPace.m

```
function minperkm = computeRunPace(dist, min, s)
% Computes the time per km in minutes, given
% the distance and the total time expressed
% in minutes and seconds.

  totalMinutes = min + s/60;
  minperkm = totalMinutes/dist;

end
```

- `function` – indikerar början på en funktion
- *funktionsnamn* – ska vara samma som m-filen namn
- *inparametrar* – data som funktionen behöver

Exempel: Använda funktionen

```
>>mpkm=computeRunPace(5,23,15)
```

Workspace: MATLAB

m-fil computeRunPace.m

```
function minperkm = computeRunPace(dist, min, s)
% Computes the time per km...

  totalMinutes = min + s/60;
  minperkm = totalMinutes/dist;
end
```

Exempel: Använda funktionen

```
>>mpkm=computeRunPace(5,23,15)
```

Workspace: MATLAB

m-fil computeRunPace.m

```
function minperkm = computeRunPace(dist, min, s)
% Computes the time per km...

  totalMinutes = min + s/60;
  minperkm = totalMinutes/dist;
end
```

Workspace: computeRunPace

```
dist = 5
min = 23
s = 15
```

Exempel: Använda funktionen

```
>>mpkm=computeRunPace(5,23,15)

mpkm = 4.65
```

Workspace: MATLAB

```
mpkm = 4.65
```

m-fil computeRunPace.m

```
function minperkm = computeRunPace(dist, min, s)
% Computes the time per km...

  totalMinutes = min + s/60;
  minperkm = totalMinutes/dist;
end
```

Workspace: computeRunPace

```
dist = 5
min = 23
s = 15
totalMinutes =
23.25
minperkm = 4.65
```

Exempel: Använda funktionen

```
>>mpkm=computeRunPace(5,23,15)

mpkm = 4.65
```

Workspace: MATLAB

```
mpkm = 4.65
```

m-fil computeRunPace.m

```
function minperkm = computeRunPace(dist, min, s)
% Computes the time per km...

  totalMinutes = min + s/60;
  minperkm = totalMinutes/dist;
end
```

Kombinera skript och funktioner

Funktioner

- Skapa funktioner varje gång en viss ”algoritm” eller flerradig beräkning används flera gånger
- MATLABs inbyggda funktioner är skriva på detta vis (skriv `type funktionsnamn` för att se)

Skript

- Definiera några variabler/parametrar
- Anropa olika funktioner
- Processa and visualisera resultaten från funktioner

Så här arbetar jag

- Kolla på min MATLAB-kod: https://github.com/emilbjornson/
- Jag publicerar forskningskod på nätet – enkelt att reproducera

Visualisering

Säg att vi vill plotta (visualisera) den matematiska funktionen $y = \sin(x)$ för $0 \leq x \leq 10$:

m-fil plotSine.m

```
x = 0:0.1:10; % The x for which y should be computed
y = sin(x);

figure; % Open a new figure ready for plotting
plot(x,y) % Plot y as a function of x
xlabel('x') % Give a name to the horizontal axis
ylabel('y = sin(x)') % Give a name to the vertical axis
title('My first plot') % Give a name to the whole figure
```

Visualisering: Många typer

Många funktionern för att plotta data:

- 2D linjegrafer: `plot`, `semilogx` (horizontal log-scale)
- 2D stapeldiagram: `bar`, `histogram`
- 3D linjegrafer: `plot3`
- 3D staplar eller nät: `bar3`, `mesh`
- 3D ytor: `surf`, `sphere`, `ellipsoid`

Använd `help` för att läsa mer!

Anpassa plottar:

- Nästan allt kan skräddarsys.
- Använd ”Property Editor” i menyn ”View” för en funktion.

Kontrollstrukturer

Vissa ”beteenden” beror på inparametrar:

- Har ditt bankkonto tillräckligt med pengar?

Vissa kodsnuttar upprepas:

- Behöver du köra samma kodrader flera gånger?
- Vet du i förväg hur många gånger?

MATLAB har flera *kontrollstrukturer*:

- if-satser
- while-loopar
- for-loopar

Dessa liknar andra programmeringsspråk.

if-satser

Generell syntax:

```
if villkor
  % kommandon om villkoret är uppfyllt
else
  % kommandon om villkoret inte är uppfyllt
end
```

Skriv villkor med logik:

- Använder operatorer som: `>` `>=` `==` `&&` `||` `~=` `<` `<=`
- Säg att `savings` är en variabel med saldot på ditt bankkonto.
- Exempel: `savings >= 0`, `(savings >= 0) || (salary > 35000)`

Exempel: if-satser

Example

Ett bankkonto ger 2% i ränta på sparade pengar och debiterar 14% i ränta på krediter. Skriv en funktion som beräknar räntan för ett viss belopp.

m-fil computeBankInterest.m

```
function interest = computeBankInterest(amount)
% Computes annual interest for a given amount

if amount >= 0
  interest = 0.02*amount;
else
  interest = 0.14*amount;
end

end
```

Skydda mot fel

If-satser kan användas för att undvika felaktiga beteenden

- Exempel: `computeBankInterest(amount)` kan inte hantera komplexa tal
- Detta kan kontrolleras och hanteras så här:

```
if imag(amount) ~= 0
  error('There is no imaginary money!');
end
```

- `imag()` ger den imaginära delen av en skalär/vektor/matris
- `error()` visar ett felmeddelande
- Textsträngar kan skriva som `'meddelande'`
- Alternativ: `disp()` visar ett icke-felrelaterat meddelande

while-loopar

- Upprepa samma beräkningar *så länge (while)* ett villkor är uppfyllt
  - Villkoret kontrolleras i början av varje loop
  - Var säker på att villkoret till slut blir falskt – annars oändlig loop!

- Generell syntax:

```
while villkor
   % kommandon som ska upprepas
end
```

Exempel: while-loopar

Example

Anta att du har lånat 1 miljon kr från en bank. Banken debiterar 0.25% i ränta per månad. Du amorterar 5000 kr per månad. Hur många månader tar det att återbetala hela lånet?

m-fil predictLoan.m

```
currentLoan = 1e6; % The initial loan is 1,000,000 kr
monthlyPayment = 5000; % You pay 5000 kr each month
montlyInterest = 0.0025; % The bank charges 0.25% per month
monthNumber = 0; % Keep track of month number

while currentLoan >= 0
  currentLoan = currentLoan + currentLoan*montlyInterest; %Apply interest rate
  currentLoan = currentLoan - monthlyPayment; %Reduce loan by monthly payment
  monthNumber = monthNumber + 1;
end

% monthNumber will now contain the month when you have repaid your loan
% Be sure that monthlyPayment > currentLoan*montlyInterest, otherwise it never stops!
```

for-loopar

- Vet hur många gångar som kommandon ska upprepas?
  - Mer kompakt att använda `for`-loopar istället för `while`

- Generell syntax:

```
for var = vektor med värden
  % kommandon som ska upprepas
end
```

Exempel: for-loopar

Example

Anta att du börjar spara 500 kr per månad när ditt barn föds. Den månatliga räntan är 0.17% (2% per år). Hur mycket kommer barnet ha vid 18 års ålder?

m-fil predictSavings.m

```
currentSaving = 0; % Bank account is empty in advance
monthlySaving = 500; % You save 500 kr per month
montlyInterest = 0.0017; % The bank interest is 0.17% per month

numberOfMonths = 12*18; % Compute number of months before turning 18

for index = 1:numberOfMonths
  currentSaving = currentSaving + currentSaving*montlyInterest; %Apply interest rate
  currentSaving = currentSaving + monthlySaving; % Add monthly saving
end

% currentSaving will now contain the savings at the age of 18
```

Sammanfattning

- MATLAB är användbart för många typer av beräkningar
- Standardverktyg på universitet och många företag – mer än 1 miljon användare
- Välj variabelnamn noggrant – skriv kommentarer
- Använd skript och funktioner, det *kommer spara dig tid*
- Kontrollstrukturer:
  - `if`-satser – gör olika saker baserat på ett villkor
  - `for`-loopar – upprepa beräkningar ett förbestämt antal gånger
  - `while` loops – upprepa beräkningar tills ett villkor inte är uppfyllt längre
- Använd hjälpsystemet för att utveckla dina kunskaper!!!

Lycka till med kursen!

Ha kul med MATLAB!

Lär er genom att utforska!