Quick Start Tutorial

Modelling, Simulation & Control

Figure 1

Bode Diagram

MATLAB
The Language of Technical Computing

Hans-Petter Halvorsen, M.Sc.
What is MATLAB?

• MATLAB is a tool for technical computing, computation and visualization in an integrated environment.

• MATLAB is an abbreviation for MATrix LABoratory

• It is well suited for Matrix manipulation and problem solving related to Linear Algebra, Modelling, Simulation and Control Applications

• Popular in Universities, Teaching and Research
Lessons

1. Solving Differential Equations (ODEs)
2. Discrete Systems
3. Interpolation/Curve Fitting
4. Numerical Differentiation/Integration
5. Optimization
6. Transfer Functions/State-space Models
7. Frequency Response
Lesson 1

Solving ODEs in MATLAB
- Ordinary Differential Equations

\[ \ddot{x} = -\frac{k}{m} x - \frac{c}{m} \dot{x} + \frac{1}{m} F \]
Example:

\[ \dot{x} = ax \]

Where \( a = \frac{-1}{T} \)

T is the Time constant

The Solution can be proved to be (will not be shown here):

\[ x(t) = e^{at} x_0 \]

Use the following:

\[ T = 5 \]
\[ x(0) = 1 \]
\[ 0 \leq t \leq 25 \]

Note!

\[ \dot{x} = \frac{dx}{dt} \]

\[ a = -\frac{1}{T}; \]
\[ x_0 = 1; \]
\[ t = [0:1:25]; \]
\[ x = \exp(a*t)*x_0; \]
\[ \text{plot}(t,x); \]
\[ \text{grid} \]

Students: Try this example
Differential Equations

\[ x(t) = e^{at} x_0 \]

\[ T = 5 \quad a = -\frac{1}{T} \]

\[ x(0) = 1 \]

\[ 0 \leq t \leq 25 \]

Problem with this method: We need to solve the ODE before we can plot it!!

T = 5;
a = -1/T;
x0 = 1;
t = [0:1:25];
x = exp(a*t)*x0;
plot(t,x);
grid
Using ODE Solvers in MATLAB

Example: \( \dot{x} = ax \)

**Step 1:** Define the differential equation as a MATLAB function (`mydiff.m`):

```matlab
function dx = mydiff(t,x)
T = 5;
a = -1/T;
dx = a*x;
```

**Step 2:** Use one of the built-in ODE solver (ode23, ode45, ...) in a Script.

```matlab
clear
clc
tspan = [0 25];
x0 = 1;
[t,x] = ode23(@mydiff,tspan,x0);
plot(t,x)
```

Students: Try this example. Do you get the same result?
Higher Order ODEs

Mass-Spring-Damper System

Example (2.order differential equation):

\[
\ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + \frac{1}{m}F
\]

\(x\) – position, \(\dot{x}\) – speed/velocity, \(\ddot{x}\) – acceleration

\(c\) – damping constant, \(m\) – mass, \(k\) – spring constant, \(F\) – force

In order to use the ODEs in MATLAB we need reformulate a higher order system into a system of first order differential equations.
Higher Order ODEs

Mass-Spring-Damper System:

\[ \ddot{x} = -\frac{k}{m}x - \frac{c}{m}\dot{x} + \frac{1}{m}F \]

In order to use the ODEs in MATLAB we need reformulate a higher order system into a system of first order differential equations

We set:

\[
\begin{align*}
x_1 &= x \\
x_2 &= \dot{x} = \dot{x}_1
\end{align*}
\]

This gives:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \ddot{x}
\end{align*}
\]

Finally:

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= -\frac{k}{m}x_1 - \frac{c}{m}x_2 + \frac{1}{m}F
\end{align*}
\]

Now we are ready to solve the system using MATLAB
**Step 1:** Define the differential equation as a MATLAB function

(mass_spring_damper_diff.m):

```matlab
function dx = mass_spring_damper_diff(t,x)

k = 1;
m = 5;
c = 1;
F = 1;

dx = zeros(2,1); %Initialization

dx(1) = x(2);
dx(2) = -(k/m)*x(1)-(c/m)*x(2)+(1/m)*F;
```

**Step 2:** Use the built-in ODE solver in a script.

Students: Try with different values for k, m, c and F

Students: Try this example

```matlab
clear
clc
tspan = [0 50];
x0 = [0;0];
[t,x] = ode23(@mass_spring_damper_diff,tspan,x0);
plot(t,x)
```
\[ [t, x] = \text{ode23}(@\text{mass\_spring\_damper\_diff}, t\text{span}, x0); \]
\[
\text{plot}(t, x) 
\]

\[ [t, x] = \text{ode23}(@\text{mass\_spring\_damper\_diff}, t\text{span}, x0); \]
\[
\text{plot}(t, x(:,2)) 
\]
For greater flexibility we want to be able to change the parameters $k$, $m$, $c$, and $F$ without changing the function, only changing the script. A better approach would be to pass these parameters to the function instead.

**Step 1:** Define the differential equation as a MATLAB function (mass_spring_damper_diff.m):

```matlab
function dx = mass_spring_damper_diff(t,x,param)

k = param(1);
m = param(2);
c = param(3);
F = param(4);

dx = zeros(2,1);

dx(1) = x(2);
dx(2) = -(k/m)*x(1) - (c/m)*x(2) + (1/m)*F;
```

Students: Try this example
clear
clc
close all

tspan = [0 50];
x0 = [0;0];

k = 1;
m = 5;
c = 1;
F = 1;
param = [k, m, c, F];

[t,x] = ode23(@mass.spring_damper_diff,tspan,x0, [], param);
plot(t,x)

Step 2: Use the built-in ODE solver in a script:

Students: Try this example
Whats next?
Learning by Doing!

Self-paced Tutorials with lots of Exercises and Video resources

Do as many Exercises as possible! The only way to learn MATLAB is by doing Exercises and hands-on Coding!!!
Lesson 2

- Discrete Systems

\[ \dot{x} \approx \frac{x(k+1) - x(k)}{T_s} \]
Discrete Systems

- When dealing with computer simulations, we need to create a discrete version of our system.
- This means we need to make a discrete version of our continuous differential equations.
- Actually, the built-in ODE solvers in MATLAB use different discretization methods.
- Interpolation, Curve Fitting, etc. is also based on a set of discrete values (data points or measurements).
- The same with Numerical Differentiation and Numerical Integration.
- etc.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Discrete Systems

Discrete Approximation of the time derivative

Euler backward method:
\[
\dot{x} \approx \frac{x(k) - x(k-1)}{T_s}
\]

Euler forward method:
\[
\dot{x} \approx \frac{x(k+1) - x(k)}{T_s}
\]
Discrete Systems

Discretization Methods

Euler backward method:
\[ \dot{x} \approx \frac{x(k) - x(k-1)}{T_s} \]
More Accurate!

Euler forward method:
\[ \dot{x} \approx \frac{x(k+1) - x(k)}{T_s} \]
Simpler to use!

Where \( T_s \) is the sampling time, and \( x(k+1), x(k), x(k-1) \) are discrete values.

Other methods are Zero Order Hold (ZOH), Tustin’s method, etc.
Different Discrete Symbols and meanings

Previous Value:
\[ x(k - 1) = x_{k-1} = x(t_{k-1}) \]

Present Value:
\[ x(k) = x_k = x(t_k) \]

Next (Future) Value:
\[ x(k + 1) = x_{k+1} = x(t_{k+1}) \]

Note! Different Notation is used in different literature!
Discrete Systems

Given the following continuous system (differential equation):

\[
\dot{x} = -ax + bu \quad \quad x(k + 1) = ?
\]

Where \( u \) may be the Control Signal from e.g., a PID Controller

We will use the Euler forward method:

\[
\dot{x} \approx \frac{x(k + 1) - x(k)}{T_s}
\]

Students: Find the discrete differential equation (pen and paper) and then simulate the system in MATLAB, i.e., plot the Step Response of the system. Tip! Use a for loop

Set \( a = 0.25 \) and \( b = 2 \)
Solution:

Discrete Systems

Given the following continuous system:

\[
\dot{x} = -ax + bu
\]

\[x(k + 1) = ?\]

We will use the Euler forward method:

1. \[
\frac{x(k + 1) - x(k)}{T_s} = -ax(k) + bu(k)
\]

2. \[x(k + 1) = x(k) + T_s[-ax(k) + bu(k)]\]

3. \[x(k + 1) = x(k) - T_s ax(k) + T_s bu(k)\]

4. \[x(k + 1) = (1 - T_s a)x(k) + T_s bu(k)\]
Solution:

Discrete Systems

MATLAB Code:

```matlab
% Simulation of discrete model
clear, clc, close all

% Model Parameters
a = 0.25; b = 2;

% Simulation Parameters
Ts = 0.1; %s
Tstop = 20; %s
uk = 1; % Step in u
x(1) = 0; % Initial value

% Simulation
for k=1:(Tstop/Ts)
    x(k+1) = (1-a*Ts).*x(k) + Ts*b*uk;
end

% Plot the Simulation Results
k=0:Ts:Tstop;
plot(k, x)
grid on
```

Students: Try this example

Students: An alternative solution is to use the built-in function `c2d()` (convert from continuous to discrete). Try this function and see if you get the same results.
Solution:

Discrete Systems

MATLAB Code:

```matlab
% Find Discrete model
clear, clc, close all

% Model Parameters
a = 0.25;
b = 2;
Ts = 0.1; %s

% State-space model
A = [-a];
B = [b];
C = [1];
D = [0];

model = ss(A,B,C,D)
model_discrete = c2d(model, Ts, 'forward')
step(model_discrete)
grid on
```

Euler Forward method

Students: Try this example
Whats next?
Learning by Doing!

Self-paced Tutorials with lots of Exercises and Video resources

Do as many Exercises as possible! The only way to learn MATLAB is by doing Exercises and hands-on Coding!!!
Lesson 3

• Interpolation
• Curve Fitting
Interpolation

Given the following Data Points:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

(Logged Data from a given Process)

```
x=0:5;
y=[15, 10, 9, 6, 2, 0];
plot(x,y , 'o')
grid
```

Students: Try this example

Problem: We want to find the interpolated value for, e.g., \( x = 3.5 \)
Interpolation

We can use one of the built-in Interpolation functions in MATLAB:

```matlab
x=0:5;
y=[15, 10, 9, 6, 2, 0];
plot(x,y ,'-o')
grid on
new_x=3.5;
new_y = interp1(x,y,new_x)
```

MATLAB gives us the answer 4.
From the plot we see this is a good guess:

Students: Try this example
Curve Fitting

• In the previous section we found interpolated points, i.e., we found values between the measured points using the interpolation technique.
• It would be more convenient to model the data as a mathematical function \( y = f(x) \).
• Then we can easily calculate any data we want based on this model.
Example:

**Curve Fitting**

**Linear Regression**

Given the following Data Points:

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Based on the Data Points we create a Plot in MATLAB:

```matlab
x=0:5;
y=[15, 10, 9, 6, 2, 0];
plot(x,y ,'o')
grid
```

Based on the plot we assume a linear relationship:

\[ y = ax + b \]

We will use MATLAB in order to find a and b.
Example

Based on the plot we assume a linear relationship:

\[ y = ax + b \]

We will use MATLAB in order to find \( a \) and \( b \).

```matlab
clear
clc
x=[0, 1, 2, 3, 4, 5];
y=[15, 10, 9, 6, 2, 0];
n=1; % 1.order polynomial
p = polyfit(x,y,n)
```

\[ p = \begin{bmatrix} -2.9143 & 14.2857 \end{bmatrix} \]

Next: We will then plot and validate the results in MATLAB.

\[ y \approx -2.9x + 14.3 \]

Students: Try this example
**Example**

**Curve Fitting**

\[ y \approx -2.9x + 14.3 \]

**Linear Regression**

We will plot and validate the results in MATLAB

Students: Try this example

```
clear
clc
close all

x=[0, 1, 2, 3, 4 ,5];
y=[15, 10, 9, 6, 2 ,0];
n=1; % 1.order polynomial
p=polyfit(x,y,n);

a=p(1);
b=p(2);

ymodel = a*x+b;

plot(x,y,'o',x,ymodel)
```

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>
Curve Fitting

Problem: We want to find the interpolated value for, e.g., \( x=3.5 \)

3 ways to do this:

- Use the `interp1` function (shown earlier)
- Implement \( y=-2.9+14.3 \) and calculate \( y(3.5) \)
- Use the `polyval` function

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

... (see previous examples)

\[
\text{new}_x=3.5;
\]

\[
\text{new}_y = \text{interp1}(x,y,\text{new}_x)
\]

\[
\text{new}_y = a*\text{new}_x + b
\]

\[
\text{new}_y = \text{polyval}(p, \text{new}_x)
\]
Curve Fitting

**Polynomial Regression**

\[ y(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n \]

1. order: \[ y(x) = ax + b \]

2. order: \[ y(x) = ax^2 + bx + c \]

3. order: \[ y(x) = ax^3 + bx^2 + cx + d \]

etc.

Students: Try to find models based on the given data using different orders (1. order – 6. order models).
Plot the different models in a subplot for easy comparison.
Curve Fitting

```
clear
clc
close all

x = [0, 1, 2, 3, 4, 5];
y = [15, 10, 9, 6, 2, 0];

for n=1:6 % n = model order
    p = polyfit(x, y, n)
    ymodel = polyval(p, x);
    subplot(3, 2, n)
    plot(x, y, 'o', x, ymodel)
    title(sprintf('Model order %d', n));
end
```

- As expected, the higher order models match the data better and better.
- Note! The fifth order model matches exactly because there were only six data points available.
- $n > 5$ makes no sense because we have only 6 data points available.
What's next?
Learning by Doing!

Self-paced Tutorials with lots of Exercises and Video resources

Do as many Exercises as possible! The only way to learn MATLAB is by doing Exercises and hands-on Coding!!!
Lesson 4

• Numerical Differentiation
• Numerical Integration
A numerical approach to the derivative of a function \( y=f(x) \) is:

\[
\frac{dy}{dx} = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}
\]

Note! We will use MATLAB in order to find the numeric solution – not the analytic solution.
Example:

**Numerical Differentiation**

\[ y(x) = x^2 \]

We know for this simple example that the exact analytical solution is:

\[ \frac{dy}{dx} = 2x \]

Given the following values:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>4</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

\[ \frac{dy}{dx}(x = -2) = -4 \]

\[ \frac{dy}{dx}(x = -1) = -2 \]

\[ \frac{dy}{dx}(x = 0) = 0 \]

\[ \frac{dy}{dx}(x = 1) = 2 \]

\[ \frac{dy}{dx}(x = 2) = 4 \]
Example: Numerical Differentiation

\[ y(x) = x^2 \quad \text{and} \quad \frac{dy}{dx} =? \]

MATLAB Code:

```matlab
x=-2:2;
y=x.^2;

% Exact Solution
dydx_exact = 2*x;

% Numerical Solution
dydx_num = diff(y)./diff(x);

% Compare the Results
dydx = [dydx_num, NaN]', dydx_exact'
plot(x,[dydx_num, NaN]', x, dydx_exact')
```

Students: Try this example.
Try also to increase number of data points, \(x=-2:0.1:2\)
Numerical Differentiation

The results become more accurate when increasing number of data points

\[
\frac{dy}{dx}
\]

\(x = -2:0.1:2\)
Numerical Integration

An integral can be seen as the area under a curve. Given $y=f(x)$ the approximation of the Area ($A$) under the curve can be found dividing the area up into rectangles and then summing the contribution from all the rectangles (trapezoid rule):

$$A = \sum_{i=1}^{n-1} (x_{i+1} - x_i) \cdot \frac{(y_{i+1} + y_i)}{2}$$

$$A = y_0 \Delta x + \frac{1}{2}(y_1 - y_0)\Delta x = \frac{(y_0+y_1)\Delta x}{2}. $$
Example:

Numerical Integration

We know that the exact solution is:

\[ y(x) = x^2 \quad \rightarrow \quad \int_a^b y(x) \, dx = ? \quad \rightarrow \quad \int_0^1 x^2 \, dx = \frac{a^3}{3} \]

\[ \int_0^1 x^2 \, dx = \frac{1}{3} \approx 0.3333 \]

We use MATLAB (trapezoid rule):

```matlab
x=0:0.1:1;
y=x.^2;
plot(x,y)

% Calculate the Integral:
avg_y=y(1:length(x)-1)+diff(y)/2;
A=sum(diff(x).*avg_y)
```

A = 0.3350

Students: Try this example
Example:

We know that the exact solution is:

\[ y(x) = x^2 \quad \Rightarrow \quad \int_a^b y(x) \, dx =? \quad \Rightarrow \quad \int_0^a x^2 \, dx = \frac{a^3}{3} \]

In MATLAB we have several built-in functions we can use for numerical integration:

```matlab
clear
clc
close all
x=0:0.1:1;
y=x.^2;
plot(x,y)

% Calculate the Integral (Trapezoid method):
avg_y = y(1:length(x)-1) + diff(y)/2;
A = sum(diff(x).*avg_y)

% Calculate the Integral (Simpson method):
A = quad('x.^2', 0,1)

% Calculate the Integral (Lobatto method):
A = quadl('x.^2', 0,1)

Students: Try this example. Compare the results. Which gives the best method?
```
Whats next?
Learning by Doing!

Self-paced Tutorials with lots of Exercises and Video resources

Do as many Exercises as possible! The only way to learn MATLAB is by doing Exercises and hands-on Coding!!!
Lesson 5

• Optimization
Optimization

Optimization is important in modelling, control and simulation applications. Optimization is based on finding the minimum of a given criteria function.

Example: $y(x) = 2x^2 + 20x - 22$

We want to find for what value of $x$ the function has its minimum value.

Students: Try this example

```
clear
clc
x = -20:0.1:20;
y = 2.*x.^2 + 20.*x - 22;
plot(x,y)
grid
i=1;
while ( y(i) > y(i+1) )
    i = i + 1;
end
x(i)
y(i)
```

The minimum of the function is $(-5, 72)$. ```
Optimization

Example:

\[ y(x) = 2x^2 + 20x - 22 \]

```matlab
clear
clc
close all

x = -20:1:20;
f = mysimplefunc(x);
plot(x, f)
grid

x_min = fminbnd(@mysimplefunc, -20, 20)
y = mysimplefunc(x_min)
```

Note! if we have more than 1 variable, we have to use e.g., the `fminsearch` function

Students: Try this example

```matlab
function f = mysimplefunc(x)
f = 2*x.^2 + 20.*x - 22;

x_min = -5

y = -72
```

We got the same results as previous slide
What's next? Learning by Doing!

Self-paced Tutorials with lots of Exercises and Video resources

Do as many Exercises as possible! The only way to learn MATLAB is by doing Exercises and hands-on Coding!!!
Lesson 6

• Transfer Functions
• State-space models

\[ H(s) = \frac{y(s)}{u(s)} = \frac{2}{s^2 + 4s + 3} \]

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{bmatrix} = \begin{bmatrix}
1 \\
3
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} + \begin{bmatrix}
2 \\
4
\end{bmatrix} u
\]

\[ y = \begin{bmatrix}
1 \\
c
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} \]
Transfer functions

\[ H(S) = \frac{x(s)}{u(s)} \]

A Transfer function is the ratio between the input and the output of a dynamic system when all the others input variables and initial conditions is set to zero.

Example:

\[ H(s) = \frac{x(s)}{u(s)} = \frac{3}{0.5s + 1} \]

\[ \dot{x} = ax + bu \]
Transfer functions

1. order Transfer function:

\[ H(s) = \frac{y(s)}{u(s)} = \frac{K}{Ts + 1} \]

Step Response:

\[ u(s) = \frac{U}{s} \]

1. order Transfer function with Time Delay:

\[ H(s) = \frac{K}{Ts + 1} e^{-\tau s} \]
Transfer functions

\[ H(s) = \frac{x(s)}{u(s)} = \frac{4}{2s + 1} \]

MATLAB:

clear
clc
close all

% Transfer Function
num = [4];
den = [2, 1];
H = tf(num, den)

% Step Response
step(H)

Students: Try this example
Transfer functions

2. order Transfer function:

\[ H(s) = \frac{K}{as^2 + bs + c} = \frac{K \omega_0^2}{s^2 + 2\zeta \omega_0 + \omega_0^2} = \frac{K}{\left(\frac{s}{\omega_0}\right)^2 + 2\zeta \frac{s}{\omega_0} + 1} \]

Example: \[ H(s) = \frac{y(s)}{u(s)} = \frac{2}{s^2 + 4s + 3} \]

MATLAB:

```matlab
clear
clc
close all

% Transfer Function
num = [2];
den = [1, 4, 3];
H = tf(num, den)

% Step Response
step(H)
```

Students: Try this example.
Try with different values for \( K, a, b \) and \( c \).
State-space models

A set with linear differential equations:

\[
\begin{align*}
\dot{x}_1 &= a_{11}x_1 + a_{21}x_2 + \ldots + a_{n1}x_n + b_{11}u_1 + b_{21}u_2 + \ldots + b_{n1}u_n \\
& \vdots \\
\dot{x}_n &= a_{1n}x_1 + a_{2n}x_2 + \ldots + a_{nn}x_n + b_{1n}u_1 + b_{2n}u_2 + \ldots + b_{nn}u_n \\
& \vdots
\end{align*}
\]

Can be structured like this:

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_n
\end{bmatrix} =
\begin{bmatrix}
a_{11} & \cdots & a_{n1} \\
\vdots & \ddots & \vdots \\
a_{1m} & \cdots & a_{nm}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n
\end{bmatrix} +
\begin{bmatrix}
b_{11} & \cdots & b_{n1} \\
\vdots & \ddots & \vdots \\
b_{1m} & \cdots & b_{nm}
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_n
\end{bmatrix}
\]

Which can be stated on the following compact form:

\[
\begin{align*}
\dot{x} &= Ax + Bu \\
y &= Cx + Du
\end{align*}
\]
State-space models

Example:

\[
\begin{align*}
\dot{x}_1 &= x_1 + 2x_2 \\
\dot{x}_2 &= 3x_1 + 4x_2 + u \\
y &= x_1
\end{align*}
\]

MATLAB:

```matlab
clear
clc
close all

% State-space model
A = [1, 2; 3, 4];
B = [0; 1];
C = [1, 0];
D = [0];
ssmodel = ss(A, B, C, D)

% Step Response
step(ssmodel)

% Transfer function
H = tf(ssmodel)

Note! The system is unstable
```

Students: Try this example
State-space models

Mass-Spring-Damper System

Example:

\[ \ddot{x} = -\frac{k}{m} x - \frac{c}{m} \dot{x} + \frac{1}{m} F \]

\( x \) – position, \( \dot{x} \) – speed/velocity, \( \ddot{x} \) – acceleration

\( c \) – damping constant, \( m \) – mass, \( k \) – spring constant, \( F \) – force

Students: Find the State-space model and find the step response in MATLAB. Try with different values for \( k \), \( m \), \( c \) and \( F \). Discuss the results
State-space models

We set:

\[
\begin{align*}
    x_1 &= x \\
    x_2 &= \dot{x} = \dot{x}_1
\end{align*}
\]

This gives:

\[
\dot{x}_1 = x_2 \\
\dot{x}_2 = \ddot{x}
\]

Finally:

Note! we have set \( F = u \)

\[
\begin{bmatrix}
    \dot{x}_1 \\
    \dot{x}_2
\end{bmatrix} = \begin{bmatrix}
    0 & 1 \\
    -\frac{k}{m} & -\frac{c}{m}
\end{bmatrix} \begin{bmatrix}
    x_1 \\
    x_2
\end{bmatrix} + \begin{bmatrix}
    0 \\
    1/m
\end{bmatrix} u
\]

\[
 F = u \\
 k = 5; \\
 c = 1; \\
 m = 1; \\
 A = [0 1; -k/m -c/m]; \\
 B = [0; 1/m]; \\
 C = [0 1]; \\
 D = [0]; \\
 sys = ss(A, B, C, D) \\
 step(sys)
\]
Whats next?
Learning by Doing!

Self-paced Tutorials with lots of Exercises and Video resources

Do as many Exercises as possible! The only way to learn MATLAB is by doing Exercises and hands-on Coding!!!
Lesson 7

• Frequency Response
The frequency response of a system expresses how a sinusoidal signal of a given frequency on the system input is transferred through the system.
The frequency response of a system is defined as the **steady-state response** of the system to a sinusoidal input signal.

When the system is in steady-state, it differs from the input signal only in **amplitude/gain** (A) ("forsterkning") and **phase lag** (ϕ) ("faseforskyvning").
The frequency response is an important tool for analysis and design of signal filters and for analysis and design of control systems.
Whats next?
Learning by Doing!

Self-paced Tutorials with lots of Exercises and Video resources

Do as many Exercises as possible! The only way to learn MATLAB is by doing Exercises and hands-on Coding!!!
Hans-Petter Halvorsen, M.Sc.

University College of Southeast Norway

www.usn.no

E-mail: hans.p.halvorsen@hit.no

Blog: http://home.hit.no/~hansha/