



Høgskolen i Telemark

Telemark University College

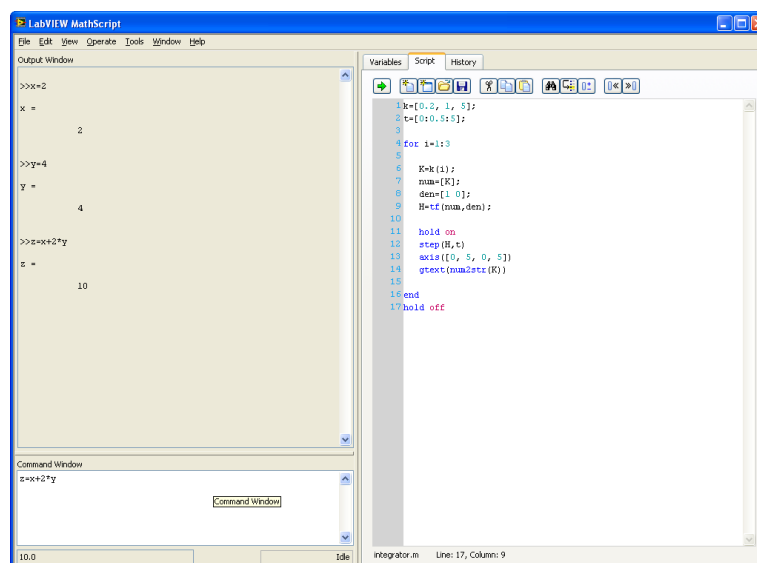
Department of Electrical Engineering, Information Technology and Cybernetics

So You Think You Can

MathScript

HANS-PETTER HALVORSEN, 2011.09.21

Part II: Dynamic Systems



Faculty of Technology, Postboks 203, Kjølnes ring 56, N-3901 Porsgrunn, Norway. Tel: +47 35 57 50 00 Fax: +47 35 57 54 01

Preface

Purpose with this Lab



In this lab you will learn how to use a tool like MathScript (which has a similar syntax as MATLAB) to solve control and simulation problems.

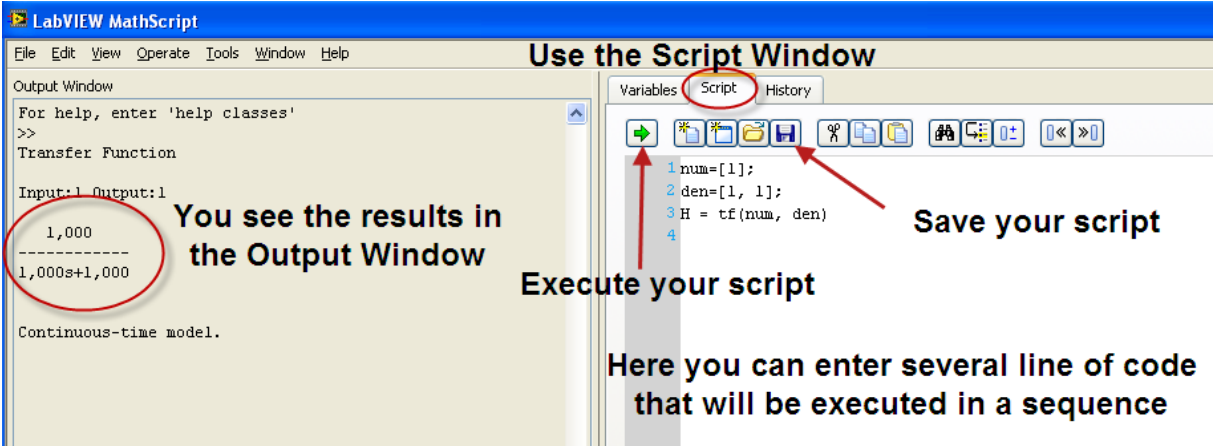
In this assignment you will define and simulate dynamic systems using:

- Block Diagrams
- Transfer functions
- State-space models
- Time delay and Pade' approximations

For additional information and resources:

<http://home.hit.no/~hansha/?lab=mathscript>

Note! For all the tasks in this document you should use the **Script Window** (Not the Command Window). When you use the Script Window you can save  the code as an .m file. In the Script Window we can enter several commands in a sequence and save them as a file. You execute these script files by clicking the green arrow  in the toolbar. This way you can easily save each task as an separate .m file (e.g., task1.m, task2.m, etc.).



LabVIEW MathScript

File Edit View Operate Tools Window Help

Use the Script Window

Output Window

For help, enter 'help classes'

>>

Transfer Function

Input:1 Output:1

1,000

1,000s+1,000

Continuous-time model.

Variables Script History

Execute your script

Save your script

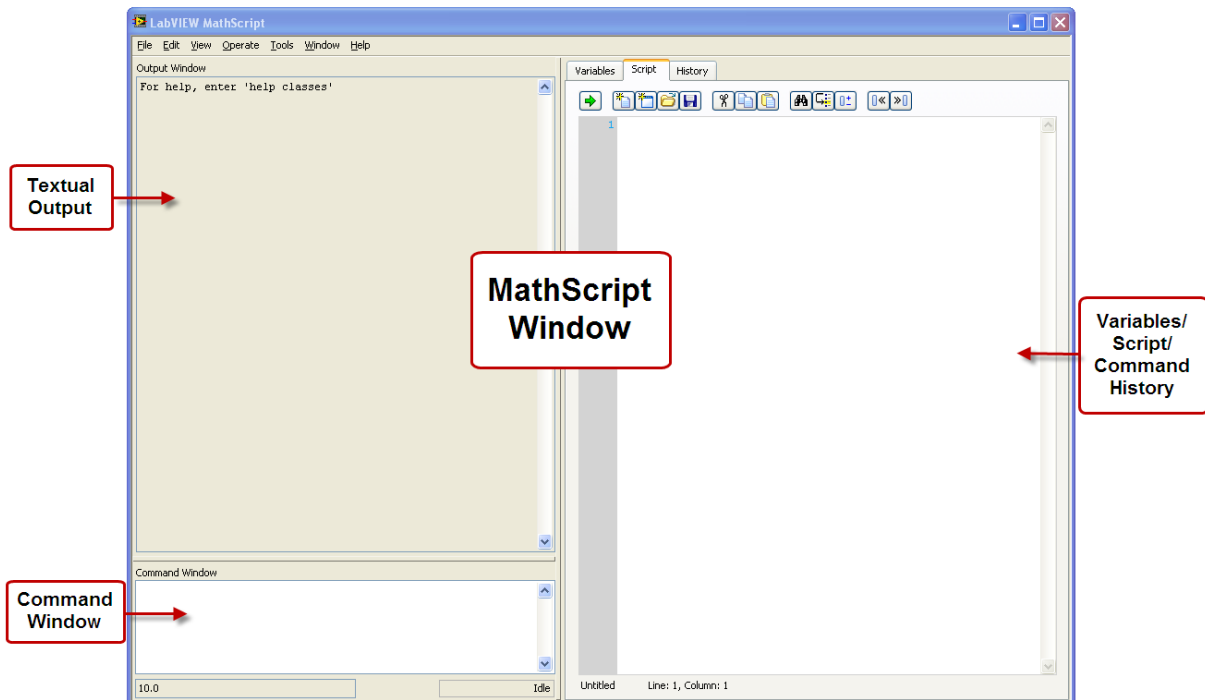
Here you can enter several line of code that will be executed in a sequence

```
1 num=[1];
2 den=[1, 1];
3 H = tf(num, den)
4
```

MathScript

MathScript is a high-level, text-based programming language. MathScript includes more than 800 built-in functions and the syntax is similar to MATLAB. You may also create custom-made m-file like you do in MATLAB.

MathScript is an add-on module to LabVIEW but you don't need to know LabVIEW programming in order to use MathScript.



What is LabVIEW?

LabVIEW (short for **L**aboratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) is a platform and development environment for a visual programming language from National Instruments. The graphical language is named "G".

What is MATLAB?

MATLAB is a tool for technical computing, computation and visualization in an integrated environment. MATLAB is an abbreviation for MATrix LABoratory, so it is well suited for matrix manipulation and problem solving related to Linear Algebra.

MATLAB offers lots of additional Toolboxes for different areas such as Control Design, Image Processing, Digital Signal Processing, etc.

What is MathScript?

MathScript is a high-level, text- based programming language. MathScript includes more than 800 built-in functions and the syntax is similar to MATLAB. You may also create custom-made m-file like you do in MATLAB.

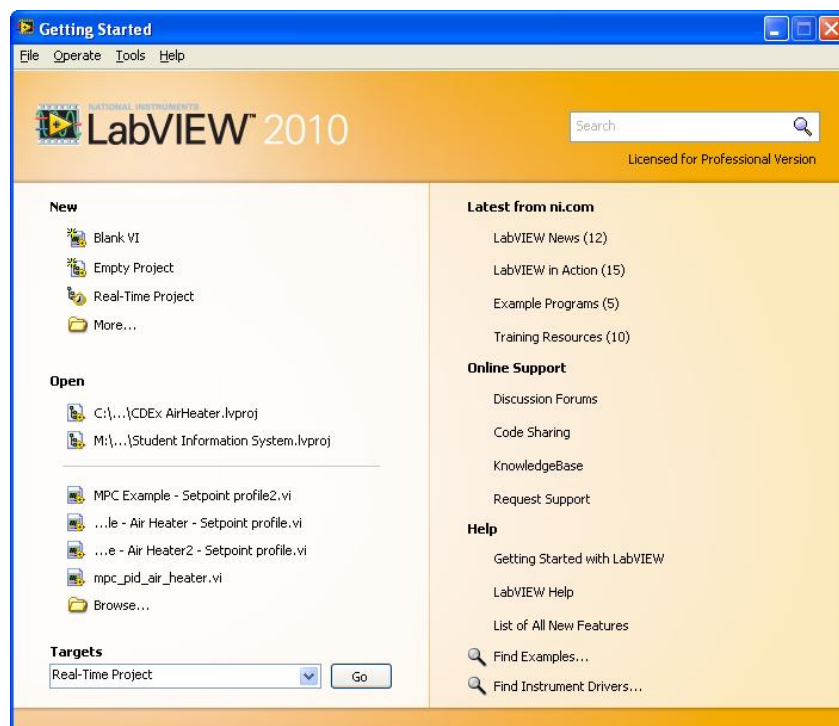
MathScript is an add-on module to LabVIEW but you don't need to know LabVIEW programming in order to use MathScript. If you want to integrate MathScript functions (built-in or custom-made m-files) as part of a LabVIEW application and combine graphical and textual programming, you can work with the **MathScript Node**.

In addition to the MathScript built-in functions, different add-on modules and toolkits installs additional functions. The **LabVIEW Control Design and Simulation Module** and **LabVIEW Digital Filter Design Toolkit** install lots of additional functions.

You can more information about MathScript here: <http://www.ni.com/labview/mathscript.htm>

How do you start using MathScript?

You need to install **LabVIEW** and the **LabVIEW MathScript RT Module**. When necessary software is installed, start MathScript by open LabVIEW:



In the **Getting Started** window, select **Tools -> MathScript Window...**:

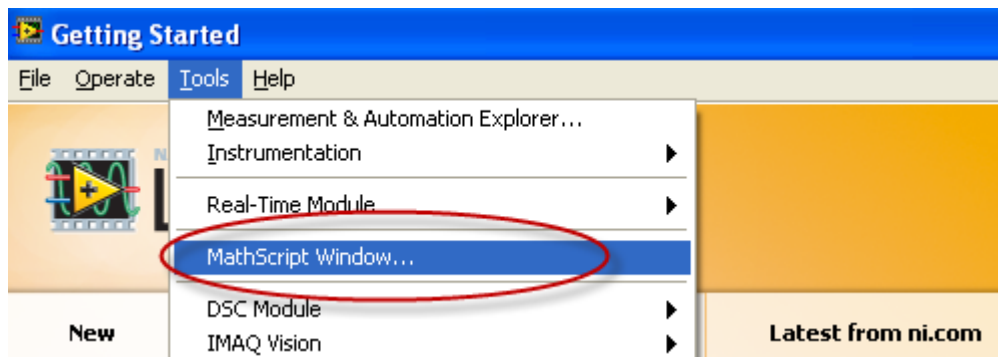


Table of Contents

Preface.....	ii
Purpose with this Lab.....	ii
MathScript.....	iii
Table of Contents	vi
1 Control Design in MathScript.....	8
2 Transfer Functions	9
2.1 Introduction.....	9
2.2 First order Transfer Functions	11
2.3 Second order Transfer Functions	13
2.4 Block Diagrams	15
2.5 PID	17
2.6 Analysis of Standard Functions	18
3 State-space Models	22
3.1 Introduction.....	22
3.2 Tasks	23
4 Time-delay and Pade'-approximations.....	26
4.1 Introduction.....	26
4.2 Tasks	31
5 Stability Analysis	33
5.1 Introduction.....	33
5.2 Poles	34
5.3 Tasks	35
5.4 Feedback Systems	36

6	Additional Tasks.....	39
	Appendix A – MathScript Functions.....	40

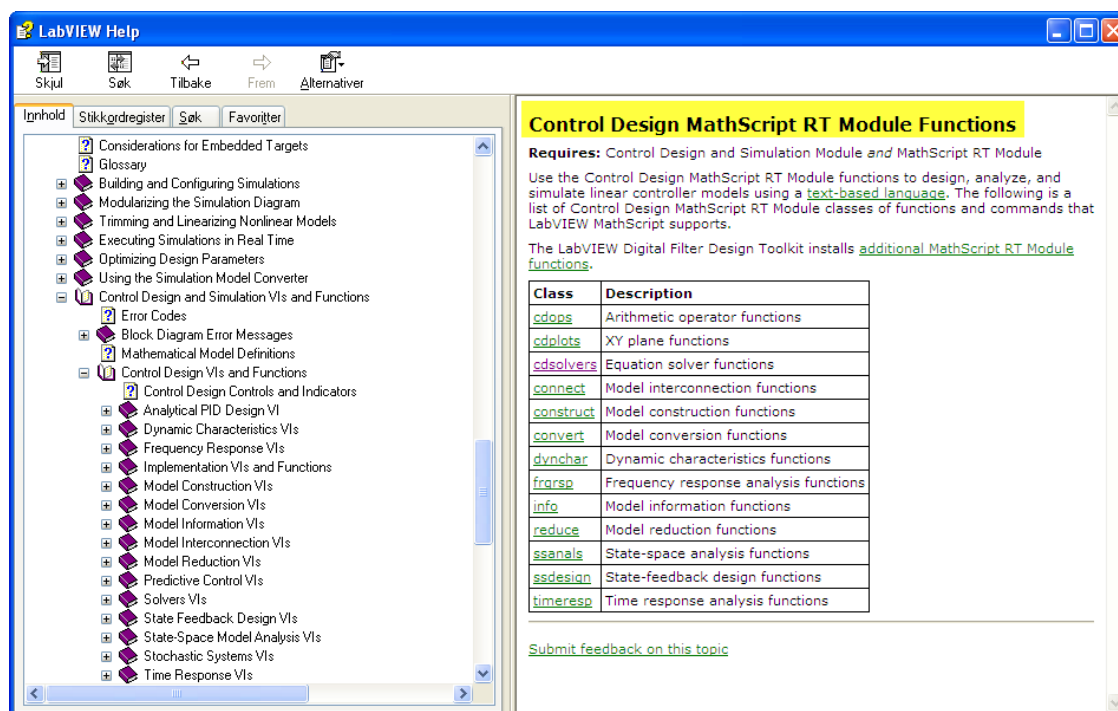
1 Control Design in MathScript

In this task you will learn how to use MathScript for Control Design and Simulation. We will learn to create transfer functions and state-space models and simulate such systems. We will also learn how to implement systems with time-delay using Pade' approximations.



Note! Using LabVIEW MathScript for Control Design purposes you need to install the **“Control Design and Simulation Module”** in addition to the **“MathScript RT Module”** itself.

Type **“help cdt”** in the Command Window in the MathScript environment and the LabVIEW Help window appears:



Use the Help window and read about some of the functions available for control design and simulation.

2 Transfer Functions

2.1 Introduction

Transfer functions are a model form based on the Laplace transform. Transfer functions are very useful in analysis and design of linear dynamic systems.

A general Transfer function is on the form:

$$H(s) = \frac{y(s)}{u(s)}$$

Where y is the output and u is the input.

MathScript has several functions for creating transfer functions:

Function	Description	Example
tf	Creates system model in transfer function form. You also can use this function to state-space models to transfer function form.	<pre>>num=[1]; >den=[1, 1, 1]; >H = tf(num, den)</pre>
Sys_order1	Constructs the components of a first-order system model based on a gain, time constant, and delay that you specify. You can use this function to create either a state-space model or a transfer function model, depending on the output parameters you specify.	<pre>>K = 1; >tau = 1; >H = sys_order1(K, tau)</pre>
Sys_order2	Constructs the components of a second-order system model based on a damping ratio and natural frequency you specify. You can use this function to create either a state-space model or a transfer function model, depending on the output parameters you specify.	<pre>>dr = 0.5 >wn = 20 >[num, den] = sys_order2(wn, dr) >SysTF = tf(num, den)</pre>
pid	Constructs a proportional-integral-derivative (PID) controller model in either parallel, series, or academic form. Refer to the LabVIEW Control Design User Manual for information about these three forms.	<pre>>Kc = 0.5; >Ti = 0.25; >SysOutTF = pid(Kc, Ti, 'academic');</pre>

A general transfer function can be written on the following general form:

$$H(s) = \frac{\text{numerator}(s)}{\text{denominator}(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0}$$

The Numerators of transfer function models describe the locations of the zeros of the system, while the Denominators of transfer function models describe the locations of the poles of the system.

In MathScript we can define such a transfer function using the built-in **tf** function as follows:

```
num=[bm, bm_1, bm_2, ... , b1, b0];
den=[an, an_1, an_2, ... , a1, a0];
```

```
H = tf(num, den)
```

Example:

1. Given the following transfer function:

$$H(s) = \frac{2s^2 + 3s + 4}{5s + 9}$$

MathScript Code:

```
num=[2, 3, 4];
den=[5, 9];
H = tf(num, den)
```

2. Given the following transfer function:

$$H(s) = \frac{4s^4 + 3s + 4}{5s^2 + 9}$$

MathScript Code:

```
num=[4, 0, 0, 3, 4];
den=[5, 0, 9];
H = tf(num, den)
```

Note! If some of the orders are missing, we just put in zeros. The transfer function above can be rewritten as:

$$H(s) = \frac{4s^4 + 0 \cdot s^3 + 0 \cdot s^2 + 3s + 4}{5s^2 + 0 \cdot s + 9}$$

3. Given the following transfer function:

$$H(s) = \frac{7 + 3s + 2s^2}{5s + 6s^2}$$

We need to rewrite the transfer function to get it in correct orders:

$$H(s) = \frac{2s^2 + 3s + 7}{6s^2 + 5s}$$

MathScript Code:

```
num=[2, 3, 7];
den=[6, 5, 0];
H = tf(num, den)
```

[End of Example]

Below we will learn more about 2 important special cases of this general form, namely the 1.order transfer function and the 2.order transfer function.

2.2 First order Transfer Functions

A first order transfer function is given on the form:

$$H(s) = \frac{K}{T_s + 1}$$

Where

K is the Gain

T is the Time constant

Example:

Given the following transfer function:

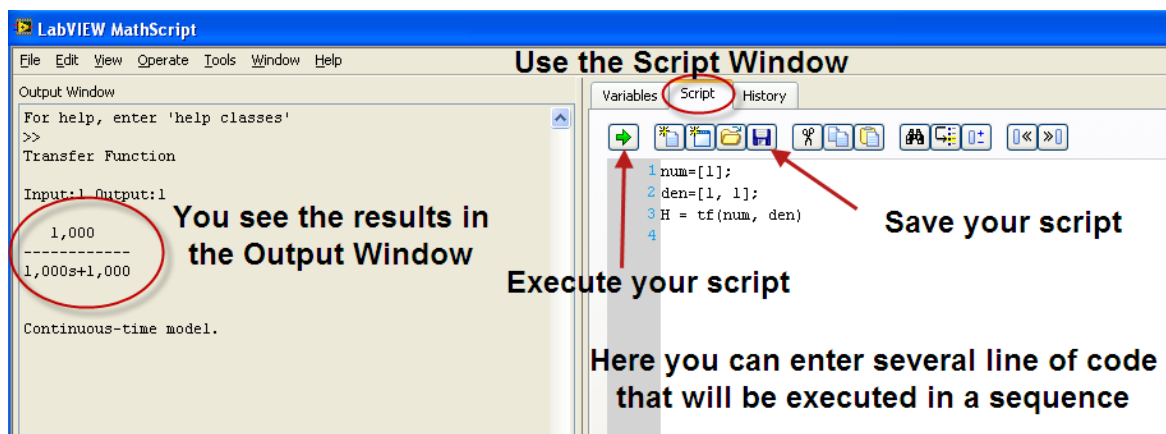
$$H(s) = \frac{1}{s + 1}$$

In MathScript we will use the following code:

```
num=[1];
den=[1, 1];
H = tf(num, den)
```

We divide the transfer function in numerator and denominator, and then we use the built-in **tf** function.

We enter the code shown above in the Script window as shown below:



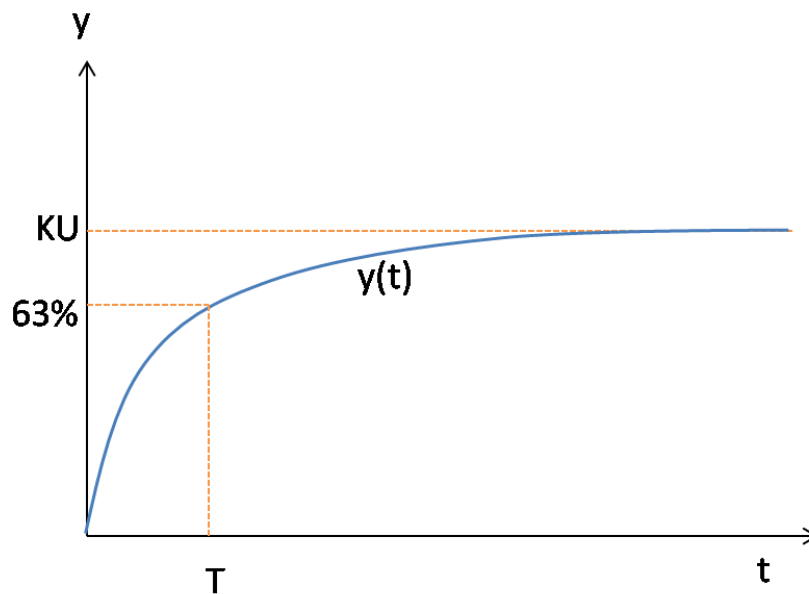
We can also use the `sys_order1` function:

```
K = 1;
T = 1;
H = sys_order1(K, T)
```

[End of Example]

Step Response:

The step response for a 1.order transfer function is as follows (a step U at $t = 0$):



The time constant T is defined as the time where the response reaches 63% of the steady state value.

Task 1: Transfer function

→ Use the `tf` function in MathScript to define the transfer function below:

$$H(s) = \frac{K}{Ts + 1}$$

Set $K = 3$ and $T = 5$.

→ Define the same function using the `sys_order1` function.

→ Find also the step response for the system using the built-in `step` function.

Note! In this task and the subsequent tasks, you should use the **Script Window** (Not the Command Window). When you use the Script Window you can save the code as an `.m` file. In the Script Window we can enter several commands in a sequence and save them as a file.

[End of Task]

2.3 Second order Transfer Functions

A second order transfer function is given on the form:

$$H(s) = \frac{K}{\left(\frac{s}{\omega_0}\right)^2 + 2\zeta \frac{s}{\omega_0} + 1}$$

Where

K is the gain

ζ zeta is the relative damping factor

ω_0 [rad/s] is the undamped resonance frequency.



Theory: [2.order Systems](#)

Example:

Given the following system:

$$H(s) = \frac{\text{numerator}}{\text{denominator}} = \frac{s^2 + 2s + 3}{4s + 1}$$

MathScript Code:

```
num=[1 2 3];
den=[4 1];
H=tf(num,den)
```

This gives the following output in MathScript:

```
>>
Transfer Function

Input:1 Output:1

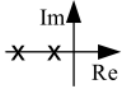
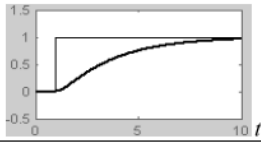
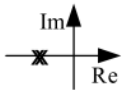
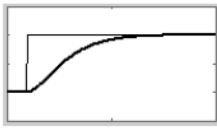
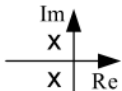
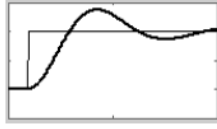
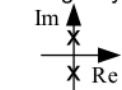
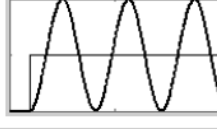
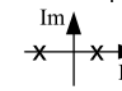
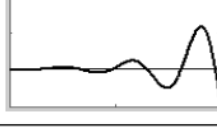
1,000s^2+2,000s+3,000
-----
4,000s+1,000

Continuous-time model.
```

[End of Example]

Step Response:

For a 2.order system we have the following step responses depending on ζ :

Value of ζ	Poles p_1 and p_2	Type of step response $y(t)$
$\zeta > 1$	Real and distinct 	Overdamped 
$\zeta = 1$	Real and multiple 	Critically damped 
$0 < \zeta < 1$	Complex conj. 	Underdamped 
$\zeta = 0$	Imaginary 	Undamped 
$\zeta < 0$	Pos. real part 	Unstable 

Task 2: 2.order

→ Define the transfer function below using the **tf** and the **sys_order2** functions (2 different methods that should give the same results).

$$H(s) = \frac{K}{\left(\frac{s}{\omega_0}\right)^2 + 2\zeta \frac{s}{\omega_0} + 1}$$

Set $K = 1, \omega_0 = 1$

→ Plot the step response (use the **step** function in MathScript) for different values of ζ . Select ζ as follows:

$$\zeta = 0.2$$

$$\zeta = 1$$

$$\zeta = 2$$

→ Explain the results.

Do you get the same results using `tf()` and `sys_order2()`?

[End of Task]

Task 3: Step Response

Given the following system:

$$H(s) = \frac{s + 1}{s^2 - s + 3}$$

→ Plot the time response for the transfer function using the `step` function. Let the time-interval be from 0 to 10 seconds, e.g., define the time vector like this:

```
t=[0:0.01:10]
```

and use

```
step(H, t)
```

→ Find poles and zeros for the system. Plot these into the complex plane. **Tip!** Use the built-in functions `poles`, `zero` and `pzgraph`.

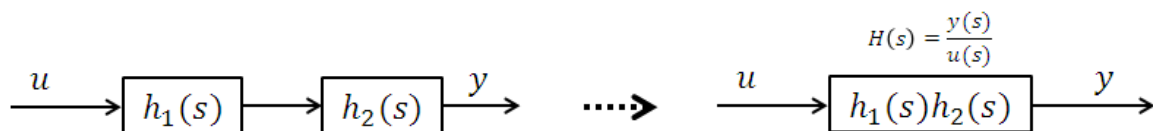
→ Discuss the results

[End of Task]

2.4 Block Diagrams

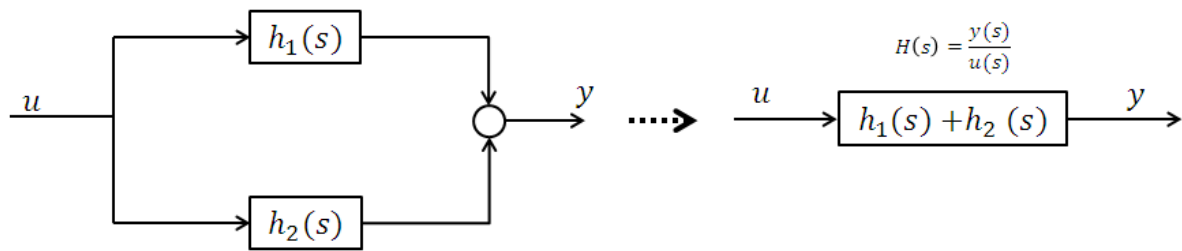
MathScript have built-in functions for manipulating block diagrams and transfer functions.

Serial:



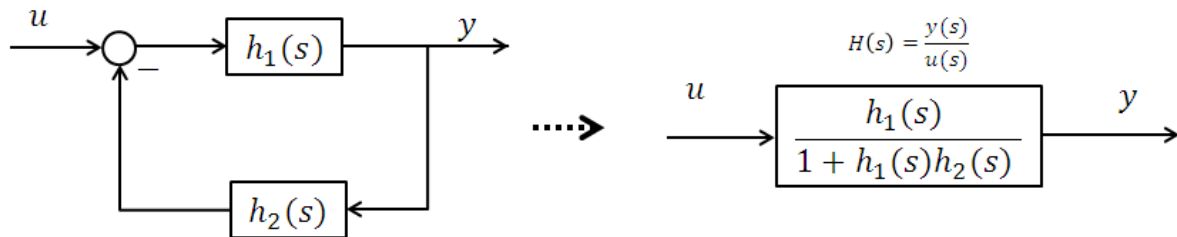
MathScript:

```
H = series(h1, h2)
```

Parallel:

MathScript:

```
...
H = parallel(h1, h2)
```

Feedback:

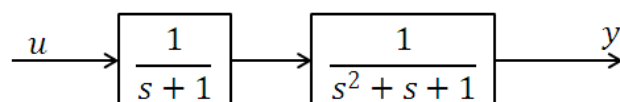
MathScript:

```
...
H = feedback(h1, h2)
```

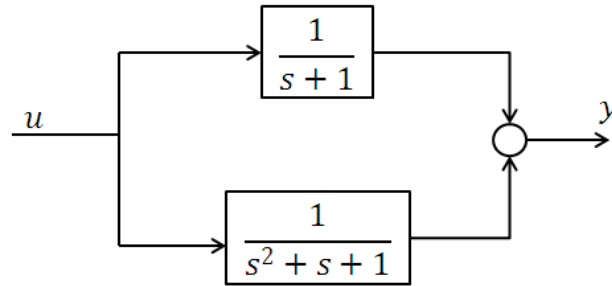
Task 4: Transfer functions and Block Diagrams

Use the **series**, **parallel** and **feedback** functions in MathScript on the block diagrams below:

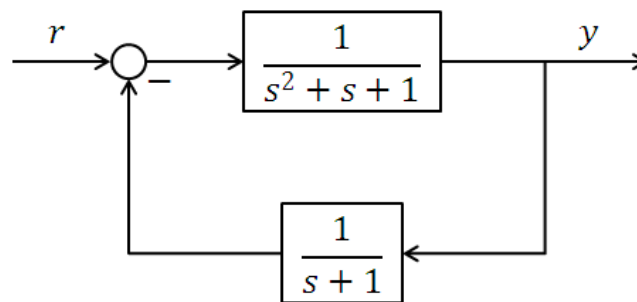
→ Find the transfer function $H(s) = \frac{y(s)}{u(s)}$ from the following block diagram:



→ Find the transfer function $H(s) = \frac{y(s)}{u(s)}$ from the following block diagram:



→ Find the transfer function $H(s) = \frac{y(s)}{r(s)}$ from the following block diagram:



→ Find the step response for these systems.

[End of Task]

2.5 PID

Currently, the Proportional-Integral-Derivative (PID) algorithm is the most common control algorithm used in industry.

The PID controller calculates the controller action, $u(t)$:

$$u(t)_{PID} = u_0 + K_p e + \frac{K_p}{T_i} \int_0^t e d\tau + K_p T_d \dot{e}$$

Task 5: PI Controller

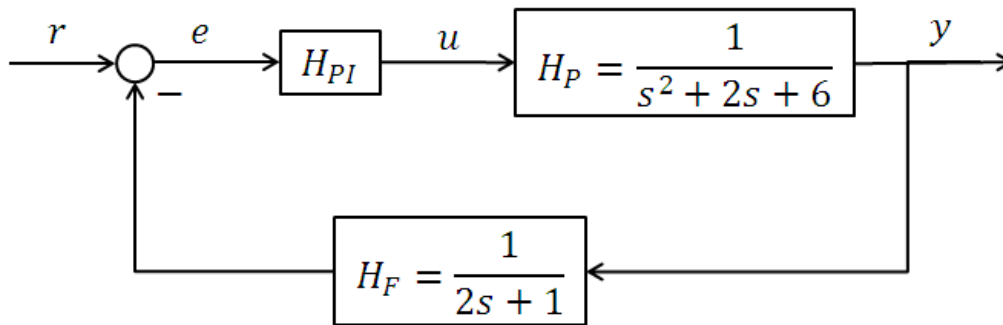
→ Create a transfer function for a PI controller using both the built-in `pid` function and the `tf` function in MathScript.

Do you get the same results?

Tip! When using the **tf** function you need to find the transfer function for a PI controller using Laplace on the equation:

$$u(t)_{PI} = u_0 + K_p e + \frac{K_p}{T_i} \int_0^t e d\tau$$

Given the following system:



Where H_{PI} is the PI controller, H_P is the process and H_F is a low-pass filter.

→ Use the **step** function in MathScript in order to plot the step response of the system.

Try with different values for K_p and T_i in order to get a good result.

[End of Task]

2.6 Analysis of Standard Functions

Here we will take a closer look at the following standard functions:

- 1. Order system
- 2. Order system

Task 6: 1.order system

1.order system:

The transfer function for a 1. order system is as follows:

$$H(s) = \frac{K}{Ts + 1}$$

→ Find the pole(s)

→ Plot the Step response. Use the **step** function in MathScript.

- Step response 1: Use different values for K , e.g., $K = 0.5, 1, 2$. Set $T = 1$
- Step response 2: Use different values for T , e.g., $T = 0.2, 0.5, 1, 2, 4$. Set $K = 1$

Discuss the result

→ (optional) Find the mathematical expression for the step response ($y(t)$). Use “Pen & Paper” for this Assignment.

$$y(s) = H(s)u(s)$$

Where

$$u(s) = \frac{U}{s}$$

Tip! Use inverse Laplace and find the corresponding transformation pair in order to find $y(t)$.

Use the mathematical expression you found for the step response ($y(t)$) and Simulate it in MathScript using, e.g., For Loop. Compare the result with the result from the **step** function.

→ Create a simple sketch of step response where you mark K , U and $T (= T_{63})$

Discuss the result

[End of Task]

Task 7: 2.order system

2.order system:

The transfer function for a 2. order system is as follows:

$$H(s) = \frac{K\omega_0^2}{s^2 + 2\zeta\omega_0s + \omega_0^2} = \frac{K}{\left(\frac{s}{\omega_0}\right)^2 + 2\zeta\frac{s}{\omega_0} + 1}$$

Where

- K is the gain
- ζ zeta is the relative damping factor
- ω_0 [rad/s] is the undamped resonance frequency.

→ Find the pole(s)

→ Plot the Step response: Use different values for ζ , e.g., $\zeta = 0.2, 1, 2$. Set $\omega_0 = 1$ and $K = 1$. Use the **step** function in MathScript.

Discuss the results

[End of Task]

Task 8: 2.order system – special case

Special case: When $\zeta > 0$ and the poles are real and distinct we have:

$$H(s) = \frac{K}{(T_1s + 1)(T_2s + 1)}$$

We see that this system can be considered as two 1.order systems in series.

$$H(s) = H_1(s)H_2(s) = \frac{K}{(T_1s + 1)} \cdot \frac{1}{(T_2s + 1)} = \frac{K}{(T_1s + 1)(T_2s + 1)}$$

→ Find the pole(s)

→ Plot the Step response. Set $K = 1$. Set $T_1 = 1$ and $T_2 = 0$, $T_1 = 1$ and $T_2 = 0.05$, $T_1 = 1$ and $T_2 = 0.1$, $T_1 = 1$ and $T_2 = 0.25$, $T_1 = 1$ and $T_2 = 0.5$, $T_1 = 1$ and $T_2 = 1$. Use the **step** function in MathScript.

Tip! Use the **conv** or the **series** together with the **tf** function in order to define the system.

Compare and discuss the results.

→ (optional) Find the mathematical expression for the step response ($y(t)$). Use “Pen & Paper” for this Assignment.

$$y(s) = H(s)u(s)$$

Where

$$u(s) = \frac{U}{s}$$

Tip! Use inverse Laplace and find the corresponding transformation pair in order to find $y(t)$.

→ Use the mathematical expression you found for the step response ($y(t)$) and Simulate it in MathScript using, e.g., For Loop. Compare the result with the result from the **step** function.

Discuss the results

[End of Task]

3 State-space Models

3.1 Introduction

A state-space model is a structured form or representation of a set of differential equations. State-space models are very useful in Control theory and design. The differential equations are converted in matrices and vectors, which is the basic elements in MathScript.

A general linear State-space model is on the form:

$$\dot{x} = Ax + Bu$$

$$y = Dx + Eu$$

MathScript has several functions for creating state-space models:

Function	Description	Example
ss	Constructs a model in state-space form. You also can use this function to convert transfer function models to state-space form.	<pre>>A = [1 2; 3 4] >B = [0; 1] >C = B' >ssmodel = ss(A, B, C)</pre>
Sys_order1	Constructs the components of a first-order system model based on a gain, time constant, and delay that you specify. You can use this function to create either a state-space model or a transfer function model, depending on the output parameters you specify.	<pre>>K = 1; >T = 1; >H = sys_order1(K, T)</pre>
Sys_order2	Constructs the components of a second-order system model based on a damping ratio and natural frequency you specify. You can use this function to create either a state-space model or a transfer function model, depending on the output parameters you specify.	<pre>>dr = 0.5 >wn = 20 >[A, B, C, D] = sys_order2(wn, dr) >ssmodel = ss(A, B, C, D)</pre>

Example:

Given the following state-space model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

The MathScript code for implementing the model is:

```
% Creates a state-space model
A = [1 2; 3 4];
B = [0; 1];
C = [1, 0];
```

```
D = [0];
model = ss(A, B, C, D)
```

[End of Example]

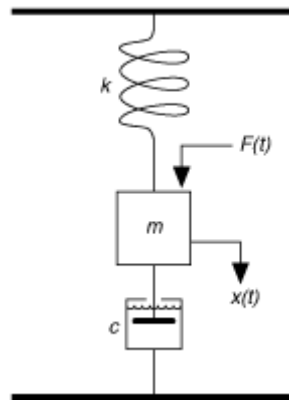


Theory: [State-Space Models](#)

3.2 Tasks

Task 9: State-space model

Given a **mass-spring-damper** system:



Where c =damping constant, m =mass, k =spring constant, $F=u$ =force

The state-space model for the system is:

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

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

→ Define the state-space model above using the **ss** function in MathScript.

Step Response:

→ Apply a step in u and use the **step** function in MathScript to simulate the result.

Start with $c = 1$, $m = 1$, $k = 1$, then explore with other values.

Conversion:

→ Convert the state-space model defined in the previous task to a transfer function using MathScript code.

[End of Task]

Task 10: Equations

→ Implement the following equations as a state-space model in MathScript:

$$\dot{x}_1 = x_2$$

$$2\dot{x}_2 = -2x_1 - 6x_2 + 4u_1 + 8u_2$$

$$y = 5x_1 + 6x_2 + 7u_1$$

→ Find the transfer function(s) from the state-space model using MathScript code.

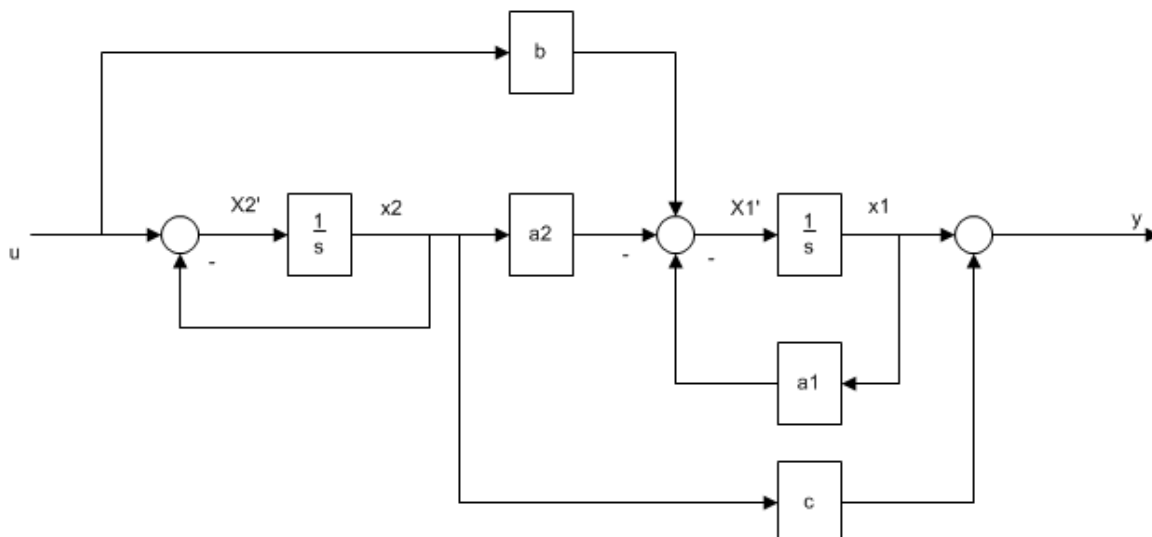
→ Plot the Step Response for the system

Discuss the results.

[End of Task]

Task 11: Block Diagram

→ Find the state-space model from the block diagram below and implement it in MathScript.



Note! $x1' \rightarrow \dot{x}_1$ and $x2' \rightarrow \dot{x}_2$.

Set

$$a_1 = 5$$

$$a_2 = 2$$

And $b = 1$, $c = 1$

→ Simulate the system using the **step** function in MathScript.

[End of Task]

4 Time-delay and Padé'-approximations

4.1 Introduction

Time-delays are very common in control systems. The Transfer function of a time-delay is:

$$H(s) = e^{-\tau s}$$

In some situations it is necessary to substitute $e^{-\tau s}$ with an approximation, e.g., the Padé'-approximation:

$$e^{-\tau s} \approx \frac{1 - k_1 s + k_2 s^2 + \dots \pm k_n s^n}{1 + k_1 s + k_2 s^2 + \dots + k_n s^n}$$



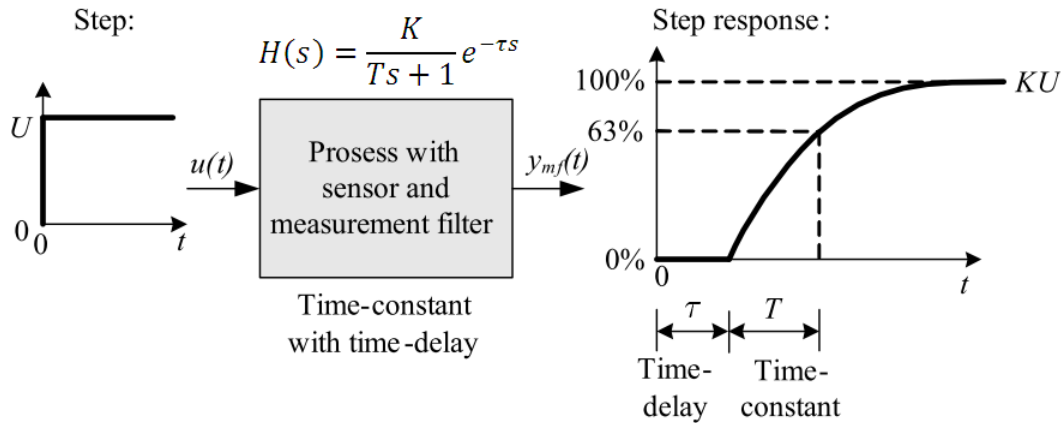
Theory: [Padé'-approximation of a time-delay](#)

A 1.order transfer function with time-delay may be written as:

$$H(s) = \frac{K}{T_s + 1} e^{-\tau s}$$

Step Response:

A step response for a 1.order system with time delay have the following characteristics:



MathScript has a built-in function called **pade** for creating transfer functions for time-delays:

Function	Description	Example
pade	Incorporates time delays into a system model using the Pade approximation method, which converts all residuals. You must specify the delay using the set function. You also can use this function to calculate coefficients of numerator and denominator polynomial functions with a specified delay.	<pre>>[num, den] = pade(delay, order) >[A, B, C, D] = pade(delay, order)</pre>
Sys_order1		<pre>>K=4; T=3; delay=5; >H = sys_order1(K, T, delay)</pre>
set		<pre>>H = set(H1, 'inputdelay', delay);</pre>
series		<pre>>H = series(H1,H2);</pre>

MathScript has a built-in function called **pade** for creating transfer functions for time-delays.

Example:

This example shows how to use the **pade** function in MathScript.

Given the following system:

$$H(s) = e^{-2s}$$

We want to create a Pade' approximation with order 3:

```
delay = 2
order = 3
[num, den] = pade(delay, order);
H = tf(num, den)
```

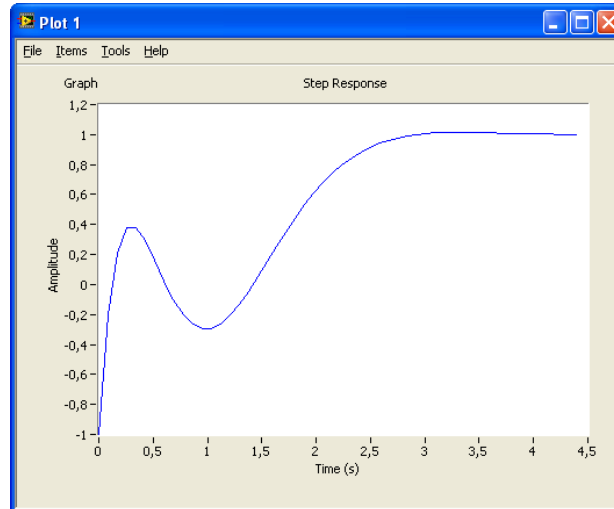
This gives the following transfer function:

$$\frac{-1,000s^3 + 6,000s^2 - 15,000s + 15,000}{1,000s^3 + 6,000s^2 + 15,000s + 15,000}$$

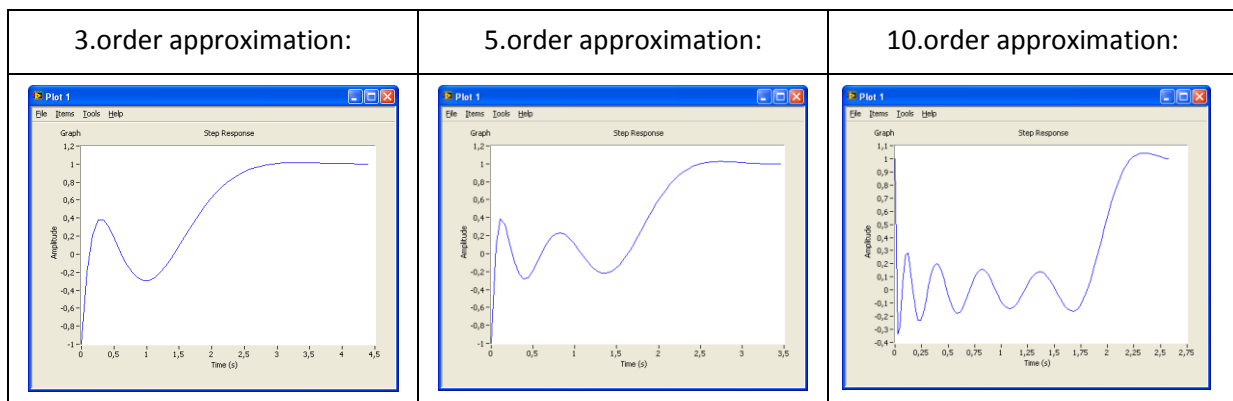
We can also plot the step response:

```
step(H)
```

This gives the following plot:



We can also try with other orders in the approximation:



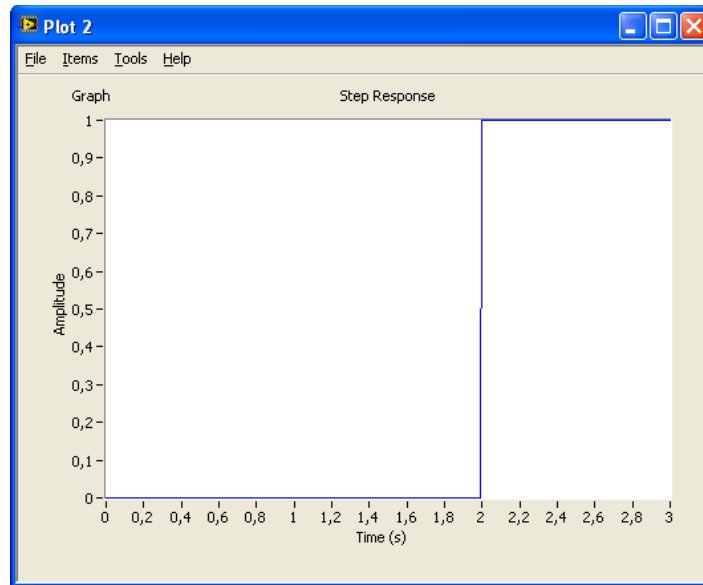
To get the “exact” step response for a time delay, let’s try to use the `sys_order1` function:

```
...
K=1;
T=0;
delay = 2

H2 = sys_order1(K,T,delay)

step(H2)
```

The step response becomes:



→ As you can see, with a higher order in the approximation, we get closer to the “exact” result. But a drawback, the approximation gets very complex.

A higher order results in a more accurate approximation of the delay but also increases the order of the resulting model. A large order can make the model too complex to be useful.

[End of Example]

Example:

Given a 1.order transfer function with time-delay:

$$H(s) = \frac{K}{Ts + 1} e^{-\tau s}$$

Where $K = 1, T = 4, \tau = 6$, i.e.:

$$H(s) = \frac{1}{4s + 1} e^{-2s}$$

We want to find the step response for this system.

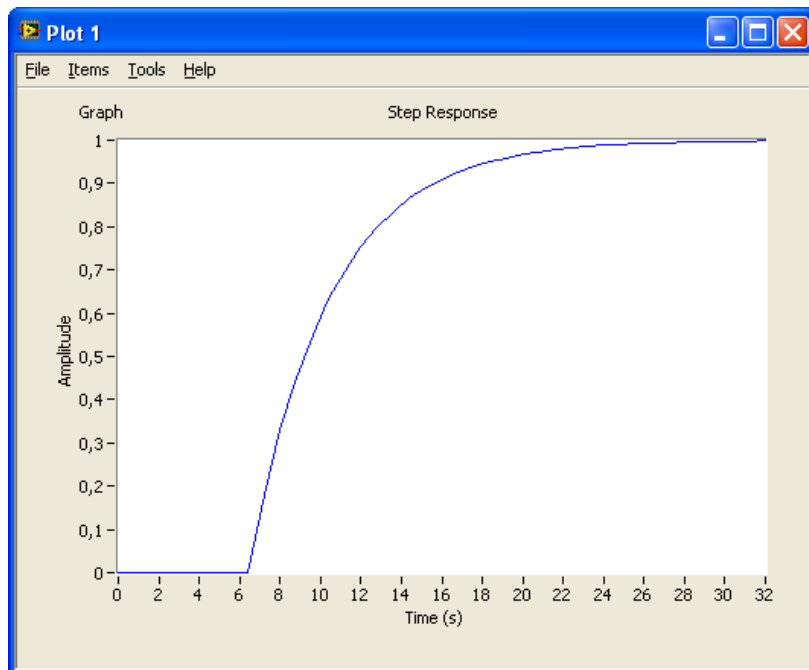
Method 1:

We use the `sys_order1` function in order to get the exact solution:

```
K=1;
T=4;
delay=2;

H = sys_order1(K,T,delay)
step(H)
```

The Step response becomes:



Method 2:

Let's try with a 5.order Pade' approximation:

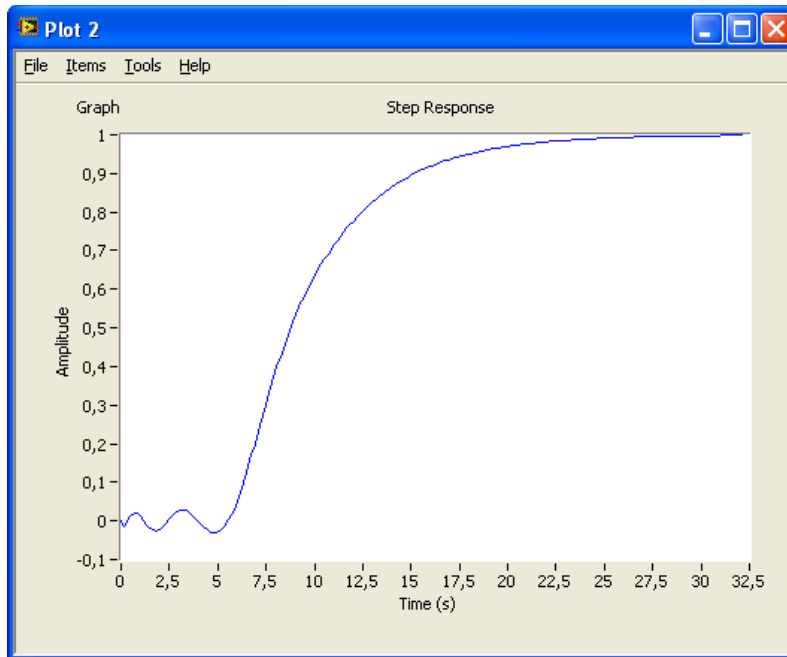
```
% Define Transfer function without delay:
num = [K];
den = [T, 1];
H1 = tf(num, den);

% Define the delay:
order = 5;
H2 = pade(delay, order)

% Put them together:
H = series(H1, H2)

The Step Response for the sytem:
step(H)
```

The step response becomes:



Method 3:

We can also implement the same function using the `tf` function in combination with the `set` function, like this:

```
...
s = tf('s');
H1 = tf(K/(T*s+1));
H2 = set(H1,'inputdelay',delay);
step(H2)
```

This gives the exact solution as shown in method 1.

[End of Example]

4.2 Tasks

Task 12: Pade'

→ Create a pade'-approximation for the time-delay:

$$H(s) = e^{-\tau s}$$

Set the time-delay $\tau = 3$ and find the pade'-approximation for different orders, e.g., 1, 2, 3, 4, 10. Use the **pade** function in MathScript.

→ Use the **step** function to plot the responses for different orders.

Discuss the results.

[End of Task]

Task 13: Pade' approximation

Note! In this task we shall note use the built-in pade function, but create our own approximation using the definition itself:

$$e^{-\tau s} \approx \frac{1 - k_1 s + k_2 s^2 + \dots \pm k_n s^n}{1 + k_1 s + k_2 s^2 + \dots + k_n s^n}$$

where:

$n = 1$	$n = 2$
$k_1 = \frac{\tau}{2}, \text{ other } k_i = 0$	$k_1 = \frac{\tau}{2}, k_2 = \frac{\tau^2}{12}, \text{ other } k_i = 0$

→ Set up the mathematical expressions, i.e, find the transfer functions for a 1.order and 2.order Pade'-approximation (Pen & Paper).

→ Define the transfer function for a 1.order and 2.order pade'-approximation using the **tf** function in MathScript. Set the time-delay $\tau = 5$.

→ Use the **step** function to plot the responses.

Do you get the same results using the **pade** function? Discuss the results.

[End of Task]

Task 14: Transfer function with Time delay

Define the following transfer function in MathScript:

$$H(s) = \frac{4}{2s + 1} e^{-3s}$$

And plot the step response.

Try both the **sys_order1** and the **pade** functions to see if you get the same results. Use e.g., a 5.order approximation.

[End of Task]

5 Stability Analysis

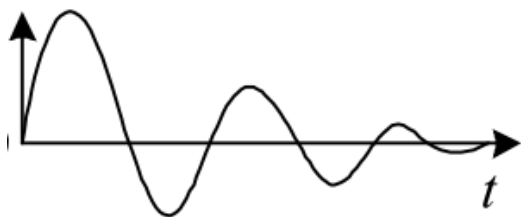
5.1 Introduction

A dynamic system has one of the following stability properties:

- Asymptotically stable system
- Marginally stable system
- Unstable system

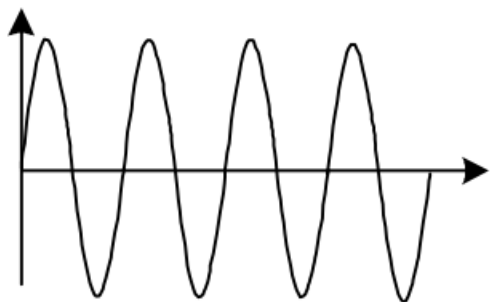
Below we see the behavior of these 3 different systems after an impulse:

Asymptotically stable system:



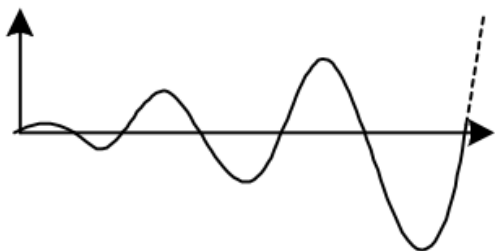
$$\lim_{t \rightarrow \infty} h(t) = 0$$

Marginally stable system:



$$0 < \lim_{t \rightarrow \infty} h(t) < \infty$$

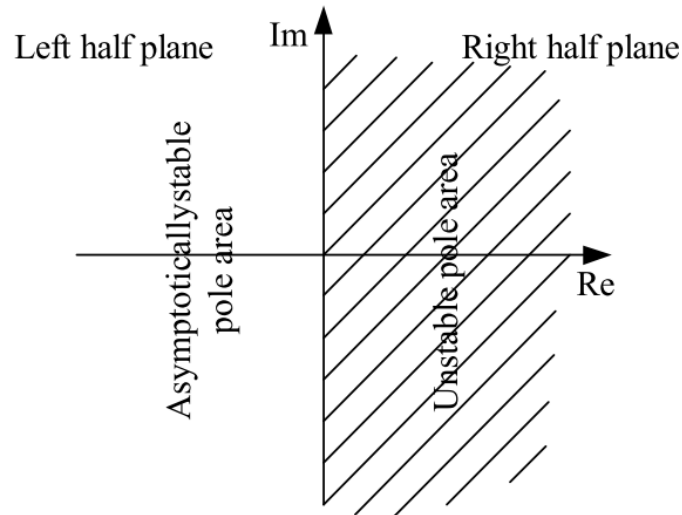
Unstable system:



$$\lim_{t \rightarrow \infty} h(t) = \infty$$

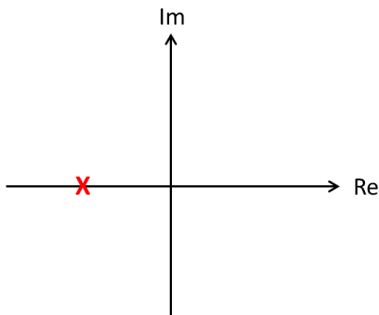
5.2 Poles

The poles is important when analysis the stability of a system. The figure below gives an overview of the poles impact on the stability of a system:



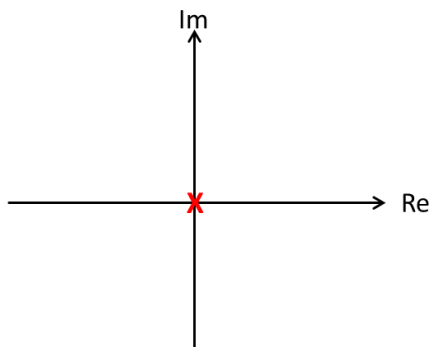
Thus, we have the following:

Asymptotically stable system:

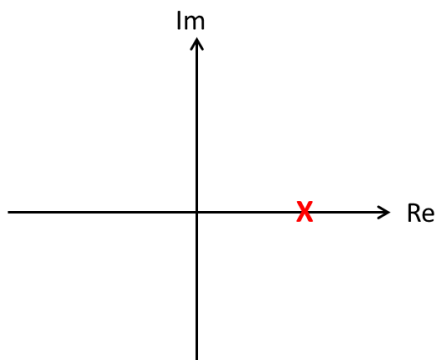


Each of the poles of the transfer function lies strictly in the left half plane (has strictly negative real part).

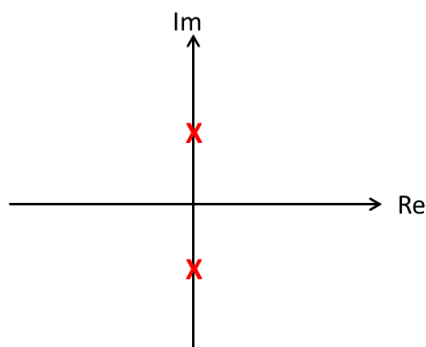
Marginally stable system:



One or more poles lies on the imaginary axis (have real part equal to zero), and all these poles are distinct. Besides, no poles lie in the right half plane.

Unstable system:

At least one pole lies in the right half plane (has real part greater than zero).



Or: There are multiple poles on the imaginary axis.

5.3 Tasks

Task 15: Stability Analysis

Given the following transfer functions:

$$H(s) = \frac{1}{s + 1}$$

$$H(s) = \frac{1}{s}$$

$$H(s) = \frac{1}{s^2}$$

$$H(s) = \frac{1}{s - 1}$$

→ Find the **poles** for the different transfer functions above using MathScript. Plot the poles in the imaginary plane. What are the stability properties of these systems (Asymptotically stable system, Marginally stable system or Unstable system)? Discuss the results.

Tip! Use the built-in functions **poles** and **pzgraph**.

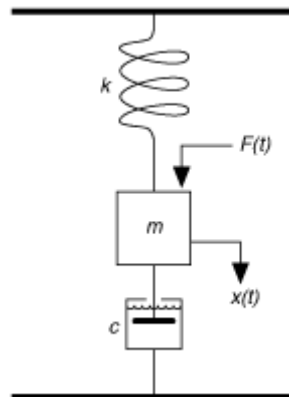
→ Plot the **impulse responses** of these systems. Discuss the results.

Tip! Use the built-in function **impulse**, which is similar to the **step** function we have used before.

[End of Task]

Task 16: Mass-spring-damper system

Given a **mass-spring-damper** system:



Where c =damping constant, m =mass, k =spring constant, $F=u$ =force

The state-space model for the system is:

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

$$y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Case 1: Set $k = 2$, $m = 20$, $c = 4$

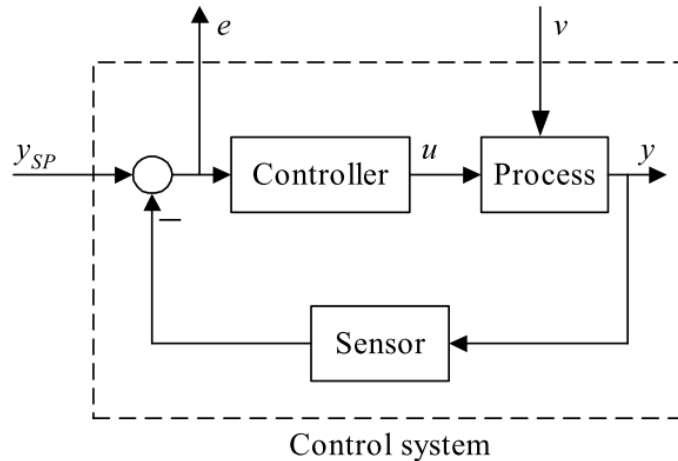
Case 1: Set $k = 2$, $m = 20$, $c = 0$

→ Investigate the stability properties of the system (Impulse response and poles).

[End of Task]

5.4 Feedback Systems

Here are some important transfer functions to determine the stability of a feedback system. Below we see a typical feedback system.



Loop Transfer function:

The **Loop transfer function** $L(s)$ (Norwegian: “Sløyfetransferfunksjonen”) is defined as follows:

$$L(s) = H_c(s)H_p(s)H_m(s)$$

Where

$H_c(s)$ is the Controller transfer function

$H_p(s)$ is the Process transfer function

$H_m(s)$ is the Measurement (sensor) transfer function

Note! Another notation for L is H_0

Tracking transfer function:

The **Tracking transfer function** $T(s)$ (Norwegian: “Følgeforholdet”) is defined as follows:

$$T(s) = \frac{y(s)}{r(s)} = \frac{H_c H_p H_m}{1 + H_c H_p H_m} = \frac{L(s)}{1 + L(s)} = 1 - S(s)$$

The **Tracking Property** (Norwegian: “følgeegenskaper”) is good if the tracking function T has value equal to or close to 1:

$$|T| \approx 1$$

Sensitivity transfer function:

The **Sensitivity transfer function** $S(s)$ (Norwegian: “Sensitivitetsfunksjonen/avviksforholdet”) is defined as follows:

$$S(s) = \frac{e(s)}{r(s)} = \frac{1}{1 + L(s)} = 1 - T(s)$$

The **Compensation Property** is good if the sensitivity function S has a small value close to zero:

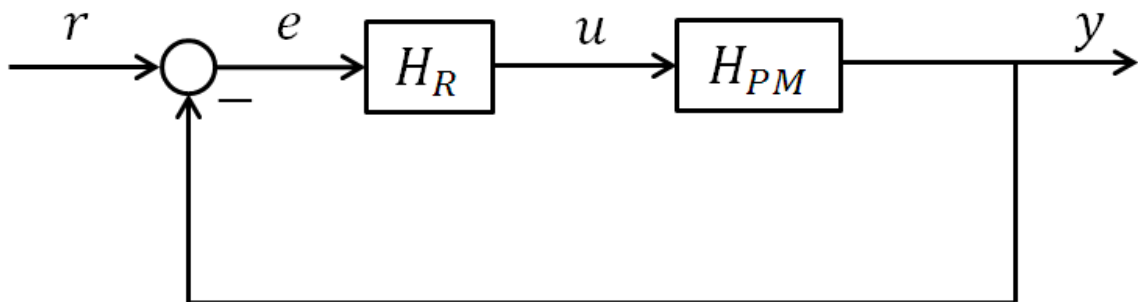
$$|S| \approx 0 \text{ or } |S| \ll 1$$

Note!

$$T(s) + S(s) = \frac{L(s)}{1 + L(s)} + \frac{1}{1 + L(s)} \equiv 1$$

Task 17: Stability Analysis of Feedback Systems

Given the following feedback system:



The transfer function for the process (including the measurement/sensor) is:

$$H_{PM}(s) = \frac{1}{(s + 1)s}$$

The transfer function for the controller is:

$$H_R(s) = K_P$$

→ Find $L(s)$, $T(s)$ and $S(s)$ for the system.

→ Plot the step response $T(s)$ and plot the poles in the imaginary plane for $T(s)$.

Is the system stable or unstable?

Start with $K_P = 1$, then explore with different values.

[End of Task]

6 Additional Tasks

Task 18: Integrator

Integrator:

The transfer function for an Integrator is as follows:

$$H(s) = \frac{K}{s}$$

→ Find the pole(s)

→ Plot the Step response: Use different values for K, eg., K=0.2, 1, 5. Use the **step** function in MathScript.

→ Discuss the result

→ Find the mathematical expression for the step response ($y(t)$). Use “Pen & Paper” for this Assignment.

$$y(s) = H(s)u(s)$$

Where

$$u(s) = \frac{U}{s}$$

Tip! Use inverse Laplace and find the corresponding transformation pair in order to find $y(t)$.

Use the mathematical expression you found for the step response ($y(t)$) and Simulate it in MathScript using, e.g., For Loop. Compare the result with the result from the **step** function.

[End of Task]

Appendix A – MathScript Functions

Here are some descriptions for the most used MathScript functions used in this Lab Work.

Function	Description	Example
plot	Generates a plot. <code>plot(y)</code> plots the columns of <code>y</code> against the indexes of the columns.	<pre>>X = [0:0.01:1]; >Y = X.*X; >plot(X, Y)</pre>
tf	Creates system model in transfer function form. You also can use this function to state-space models to transfer function form.	<pre>>num=[1]; >den=[1, 1, 1]; >H = tf(num, den)</pre>
poles	Returns the locations of the closed-loop poles of a system model.	<pre>>num=[1] >den=[1,1] >H=tf(num,den) >poles(H)</pre>
tfinfo	Returns information about a transfer function system model.	<pre>>[num, den, delay, Ts] = tfinfo(SysInTF)</pre>
step	Creates a step response plot of the system model. You also can use this function to return the step response of the model outputs. If the model is in state-space form, you also can use this function to return the step response of the model states. This function assumes the initial model states are zero. If you do not specify an output, this function creates a plot.	<pre>>num=[1,1]; >den=[1,-1,3]; >H=tf(num,den); >t=[0:0.01:10]; >step(H,t);</pre>
lsim	Creates the linear simulation plot of a system model. This function calculates the output of a system model when a set of inputs excite the model, using discrete simulation. If you do not specify an output, this function creates a plot.	<pre>>t = [0:0.1:10] >u = sin(0.1*pi*t) >lsim(SysIn, u, t)</pre>
sys_order1	Constructs the components of a first-order system model based on a gain, time constant, and delay that you specify. You can use this function to create either a state-space model or a transfer function model, depending on the output parameters you specify.	<pre>>K = 1; >tau = 1; >H = sys_order1(K, tau)</pre>
sys_order2	Constructs the components of a second-order system model based on a damping ratio and natural frequency you specify. You can use this function to create either a state-space model or a transfer function model, depending on the output parameters you specify.	<pre>>dr = 0.5 >wn = 20 >[num, den] = sys_order2(wn, dr) >SysTF = tf(num, den) >[A, B, C, D] = sys_order2(wn, dr) >SysSS = ss(A, B, C, D)</pre>
damp	Returns the damping ratios and natural frequencies of the poles of a system model.	<pre>>[dr, wn, p] = damp(SysIn)</pre>
pid	Constructs a proportional-integral-derivative (PID) controller model in either parallel, series, or academic form. Refer to the LabVIEW Control Design User Manual for information about these three forms.	<pre>>Kc = 0.5; >Ti = 0.25; >SysOutTF = pid(Kc, Ti, 'academic');</pre>
conv	Computes the convolution of two vectors or matrices.	<pre>>C1 = [1, 2, 3]; >C2 = [3, 4]; >C = conv(C1, C2)</pre>
series	Connects two system models in series to produce a model SysSer with input and output connections you specify	<pre>>Hseries = series(H1,H2)</pre>
feedback	Connects two system models together to produce a closed-loop model using negative or positive feedback connections	<pre>>SysClosed = feedback(SysIn_1, SysIn_2)</pre>
ss	Constructs a model in state-space form. You also can use this function to convert transfer function models to state-space form.	<pre>>A = eye(2) >B = [0; 1] >C = B' >SysOutSS = ss(A, B, C)</pre>

ssinfo	Returns information about a state-space system model.	<pre>>A = [1, 1; -1, 2] >B = [1, 2]'; >C = [2, 1] >D = 0 >SysInSS = ss(A, B, C, D) >[A, B, C, D, Ts] = ssinfo(SysInSS) >[num, den] = pade(delay, order) >[A, B, C, D] = pade(delay, order)</pre>
pade	Incorporates time delays into a system model using the Pade approximation method, which converts all residuals. You must specify the delay using the set function. You also can use this function to calculate coefficients of numerator and denominator polynomial functions with a specified delay.	

For more details about these functions, type “[help cdt](#)” to get an overview of all the functions used for Control Design and Simulation. For detailed help about one specific function, type “[help <function_name>](#)”.



Høgskolen i Telemark

Telemark University College

Faculty of Technology

Kjølnes Ring 56

N-3914 Porsgrunn, Norway

www.hit.no

Hans-Petter Halvorsen, M.Sc.

Telemark University College

Department of Electrical Engineering, Information Technology and Cybernetics

Phone: +47 3557 5158

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

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

Room: B-237a

