The purpose of this experiment is to become acquainted with the tools installed on your station. In this experiment you will use the software packages MATLAB and SIMULINK. These two packages are among the most popular tools that control engineers use.

Include all the printouts in your report!

1. Run MATLAB and open the Demo window by clicking on Start button (icon in the bottom-left corner), Demos. Go through the demos and get familiar with basic matrix functions that MATLAB offers.

   (a) Generate two random nonsingular 3×3 matrices A and B. Compute the inverse of A and the transpose of A. Print the results.

   (b) Using A and B, compute $A * B$ and $A. * B$. Print the result and explain what is the difference between the two operations.

   (c) Describe what does the function $eig()$ do.

   (d) Plot a graph of function $f(x) = e^{-2x} \sin(3x + 1)$ on the interval $x \in [0, 10]$.

2. Explore Simulink by clicking on Start $\rightarrow$ Simulink $\rightarrow$ Demos. You can find lots of information in the video demos.

3. In the Simulink Demos window, scroll to General Applications (or use the explorer window on the left) and open Thermal Model of a House demo. Open the model and this will start a demo of the thermodynamics of a house in SIMULINK. Explore the demo by clicking on different blocks.

   (a) Double click on the Scope block labeled PlotResults. Start the simulation by using **Start** command on the **Simulation** menu, or by clicking the Play button. Let the simulation run for some time and then stop the simulation by choosing **Stop**. Print the plots and explain what they represent.

   (b) The constant block labeled Set Point sets the desired temperature. Change the value to 80F. Run the simulation again and see how the indoor temperature and the heating cost changes. Print the results.

   (c) Adjust the daily temperature variation by opening the Sine Wave block labeled Daily Temp Variation and changing the **Amplitude** parameter. Repeat the simulation, print the results and comment what happened.

4. Now we want to build a simple model in SIMULINK. We would like to integrate a sine wave and display the result along with the sine wave.

   Open a new model in Simulink. To create the model you will need to copy blocks into the model from the following SIMULINK block libraries:

   - Sources library (the Sine Wave block).

- Sinks library (the Scope block).
- Continuous library (the Integrator block).
- Signals & Systems library (the Mux block).

Find all these blocks and drag the appropriate icons into the Model window. You will see that the icons have little > parts. These represent input and output ports.

Connect the output of the Sine Wave block with the top input of the Mux block (click the left mouse button on the appropriate port and while pressing the button move to the port you want connected). Similarly, connect the output of the Integrator and the bottom input of the Mux, and the output of the Mux and the input of the Scope. Now we would also like to connect the output of Sine Wave with the input of the Integrator. To do this:

(a) Position the pointer *on the line* connecting Sine Wave with the Mux.

(b) Press and hold down the **Ctrl** key (or click the right mouse button) and drag the line to the input of the Integrator.

(c) Release the button.

Print out the drawing of the model when you are done.

Now open the Scope block to view the simulation output. Set SIMULINK to run for 10 seconds by adjusting **Stop time** in the box **Simulation Parameters** under the **Simulation** menu.

Start the simulation and watch the traces on the Scope. Print the results and explain what the curves represent.

Save the model by choosing **Save** from the **File** menu. Exit MATLAB.

Transfer functions are a convenient and powerful way to model systems. Most transfer functions are rational functions – quotients of polynomials.

The objective of this experiment is twofold:

- To learn how to use MATLAB to: (1) generate polynomials; (2) manipulate polynomials; (3) generate transfer functions; (4) manipulate transfer functions; and (5) perform partial-fraction expansions.
- To learn to use MATLAB and the Symbolic Toolbox to: (1) find Laplace transforms for time functions; (2) find time functions from Laplace transforms; (3) create LTI transfer functions from symbolic transfer functions; and (4) perform solutions of symbolic simultaneous equations.

Include all the printouts in your report!

1 Polynomials

You can find a quick tutorial on basic operations with polynomials in Matlab at `http://www.matrixlab-examples.com/polynomials.html` and another one at `http://www.engin.umich.edu/group/ctm/basic/basic.html#polynomial`.

1. Using Matlab, compute:
   (a) The roots of $P_1 = s^6 - 8s^5 + 4s^4 + 82s^3 - 125s^2 - 74s + 120$.
   (b) The roots of $P_2 = s^5 - 2s^4 + s^3 + 2s^2 - 2s$.
   (c) $P_3 = P_1 + P_2$; $P_4 = P_1 - P_2$; $P_5 = P_1 P_2$; $P_6 = P_1/P_2$.
2. Find a Matlab command that can compute the coefficients of the polynomial:

$$P_7 = (s+3)(s-2)(s+1)(s+5)(s-6)(s+7).$$

2 Transfer Functions

A nice tutorial on how to use Matlab for transfer function operations can be found at `http://www.regpro.jku.at/downloads/autpr/Tutorials/MatlabSimulink/DefiningTransferFunctionsinMatlab.htm`. You can also obtain help on a Matlab command by using "`doc` *`command`*".

1. Use Matlab function "`zpk`" to create a representation "`sys1`" of the following transfer function:

$$G_1 = \frac{20(s+1)(s+3)(s+6)(s+8)}{s(s+7)(s+9)(s+10)(s+15)}.$$

Apply the function "`tf`" to "`sys1`″ to find G_1 represented as a numerator polynomial divided by a denominator polynomial.

2. Now find how the transfer function

$$G_2 = \frac{s^4 + 17s^3 + 99s^2 + 223s + 140}{s^5 + 32s^4 + 363s^3 + 2092s^2 + 5052s + 4320},$$

can be represented as a product of factors in the numerator divided by a product of factors in the denominator.

3. Compute $G_3 = G_1 + G_2$, $G_4 = G_1 - G_2$ and $G_5 = G_1 G_2$, and represent the results as factors divided by factors, and as polynomials divided by polynomials. Do G_1 and G_2 need to be expressed in the same form to perform these operations?

4. Use the function "`residue`" and compute partial-fraction expansion of the following transfer functions:

   (a) $G_6 = \frac{5(s+2)}{s(s^2+8s+15)}$,

   (b) $G_7 = \frac{5(s+2)}{s(s^2+6s+9)}$,

   (c) $G_8 = \frac{5(s+2)}{s(s^2+6s+34)}$.

3 System Models

For this part, you might want to go over tutorials for the Matlab Symbolic Toolbox. The following Matlab functions will be useful for the problems: "`poly2sym`", "`sym2poly`", "`numden`", "`tfdata`", "`laplace`", "`ilaplace`", and "`solve`".

1. Define a symbolic function

$$f(t) = \frac{5\ (\cos(2\,t) + 5\ \sin(2\,t))}{8\,\mathrm{e}^{3\,t}} - \frac{5}{8\,\mathrm{e}^{t}},$$

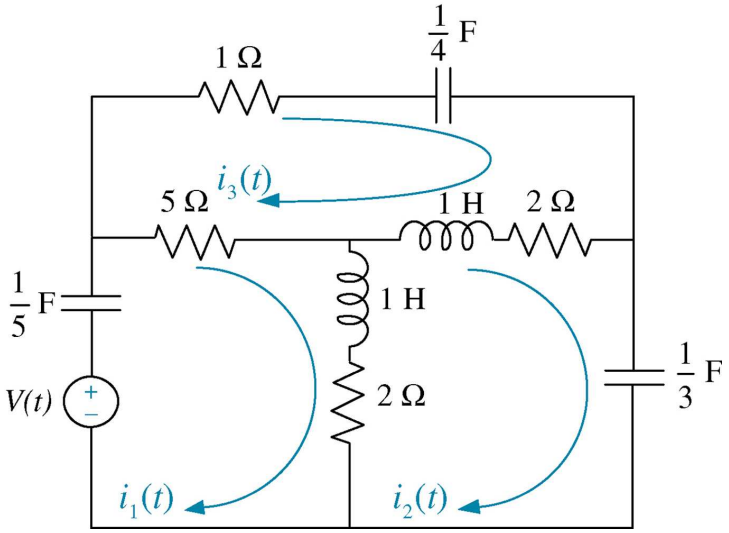
and compute its Laplace transform.

2. Define a symbolic function

$$F(s) = \frac{2(s+3)(s+5)(s+7)}{s(s+8)(s^2+10s+100)},$$

and find its inverse Laplace transform. Also generate a transfer function model corresponding to $F(s)$ both as factors divided by factors, and as polynomials divided by polynomials.

3. Using Symbolic Toolbox in Matlab, find the three loop currents for the following circuit:


1/4 F
1 Ω
i3(t)
5 Ω
1 H
2 Ω
1/5 F
1 H
V(t)
2 Ω
1/3 F
i1(t)
i2(t)

Many engineering systems exhibit nonlinear behavior. Since there are many more tools available to design and study linear systems than nonlinear systems, it is customary to approximate a nonlinear system with its linearization around an operating point. In this experiment you will use Matlab and Simulink to compare the behavior of a nonlinear system with its linearization.

1. Consider a pendulum, that is a point mass m swinging on a massless rod of length l:

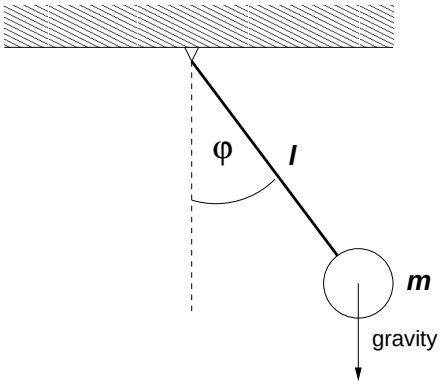


Figure 1: A pendulum.

(a) Derive the differential equation in φ describing the motion of the mass m. Now introduce the following two variables:

$$x_1 = \varphi$$
$$x_2 = \dot{\varphi}$$

We clearly have the relation $\dot{x}_1 = x_2$. Determine the expression for $\dot{x}_2$ using the differential equation you derived before.

Let $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. You can write the equations for the pendulum in the form:

$$\dot{x} = f(x)$$

where $f(x)$ is a 2×1 vector function. Determine the function $f(x)$. The model in this form is called a **state space** model.

(b) If you performed the calculations correctly, the function $f(x)$ should be nonlinear. Write a first order Taylor series of $f(x)$ around the point $x_1 = 0$, $x_2 = 0$. Recall that a Taylor series of a function of $x = (x_1, x_2, \ldots, x_n)$ around the point $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$ is:

$$f(x_1, x_2, \ldots, x_n) \approx f(x^0) + \frac{\partial f}{\partial x_1}|_{x=x^0}(x_1 - x_1^0) + \frac{\partial f}{\partial x_2}|_{x=x^0}(x_2 - x_2^0) + \cdots + \frac{\partial f}{\partial x_n}|_{x=x^0}(x_n - x_n^0)$$

Using the Taylor series, you can write a **linearized model** of the pendulum in the form:

$$\dot{x} = Ax$$

where A is a constant 2×2 matrix. Determine A.

(c) We would like to build a simulation of the pendulum in Simulink. Schematically, a simulation of the system described with a state-space model looks like Figure 2.

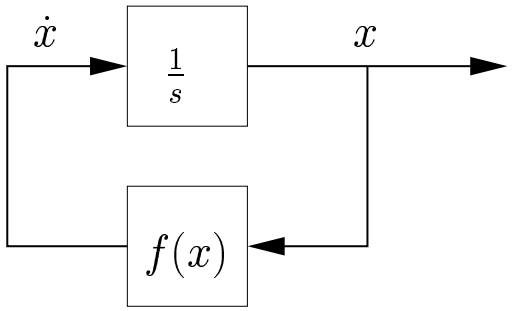


Figure 2: Block diagram for the simulation.

Use two integrators for x_1 and x_2 and combine the signals using elements from the Math library to build the model of the pendulum in Simulink. For the simulation, use the values $m = 1kg$ and $l = 1m$. In the same Simulink window, build the model of the linearized system. Using the **Scope** block and the **Mux** block, display the x_1 variable for the pendulum and for the linearized system.

(d) We would like to compare how well does the linearized system describe the pendulum. First, you will need to set the initial conditions for the simulation by setting the initial conditions of the integrators. To do that, click on the **Integrator** block, specify the **Initial condition source** as **internal** and enter the value in the **Initial condition** parameter field.

Run the simulation for $10s$ for the initial conditions:

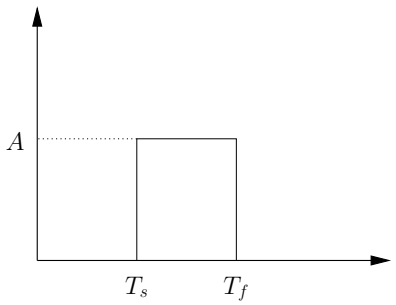
i. $x_1 = 5°$, $x_2 = 0$.
ii. $x_1 = 10°$, $x_2 = 0$.
iii. $x_1 = 30°$, $x_2 = 0$.
iv. $x_1 = 50°$, $x_2 = 0$.

Print the results in each case. What can you say about the relation between the behavior of the pendulum and that of the linearized system?

One of the most important components of a control system is the actuator. Many systems rely on an electrical actuator, and among these, DC motor continues to be the most popular choice. In this experiment you will explore the behavior of a DC motor and learn how it is used in control systems.

Remember to save all your work on portable media.

1. A typical input to an electrical motor is a pulse:



If $1(t)$ is a unit step function, a pulse in the figure can be defined as:

$$p(t) = A\,(1(t - T_s) - 1(t - T_f))\,.$$

Use the Simulink to implement a pulse as an input function. Use the **Step** block in the **Sources** library and the **Sum** block in the **Math** library. Assume that $A = 1$, $T_s = 1$ and $T_f = 2$. Verify your design with a simulation using a **Scope** block. Print the result of the simulation.

Hint: If you click on the **Step** block you can set the amplitude and the location of the step.

2. We would like to see how a field-controlled motor responds to a pulse. Implement the following block diagram:

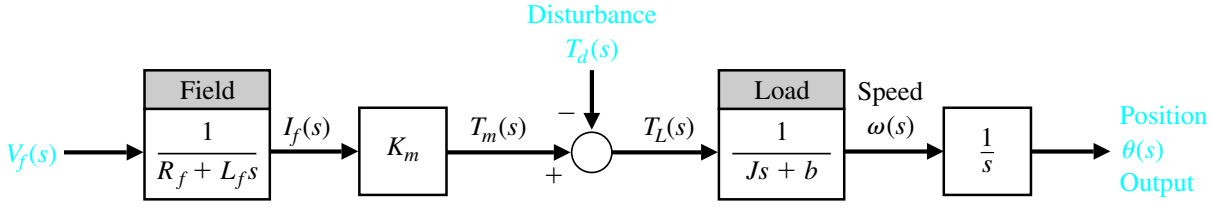


Figure 1: Field-controlled DC motor.

The parameters of the motor are:

$$R_f = 1 \qquad J = 2 \qquad K_m = 10$$
$$L_f = 0.1 \qquad b = 0.5$$

(a) Use the **Transfer Fcn** block in the **Continuous** library to implement each transfer function. You can change the coefficients of the transfer function by clicking on the block.

Let the input V_f be the pulse you implemented above. Set the disturbance to be a **Constant** block in the **Sources** library. Set the value of the constant to 0.

(b) Using the **Mux** block, connect the output of the motor θ, the torque T_m and the input V_f to a scope. Set the Stop time for the simulation to 40. Open the scope window and run the simulation. Click the right button on the scope window to Autoscale the plot. Print the result.

(c) Now set the value of the disturbance torque T_d to 0.1. Run the simulation again, Autoscale the plot and print the result. Comment on the difference from the previous simulation.

3. We would like to compare the performance of the field-controlled motor with that of an armature-controlled motor:

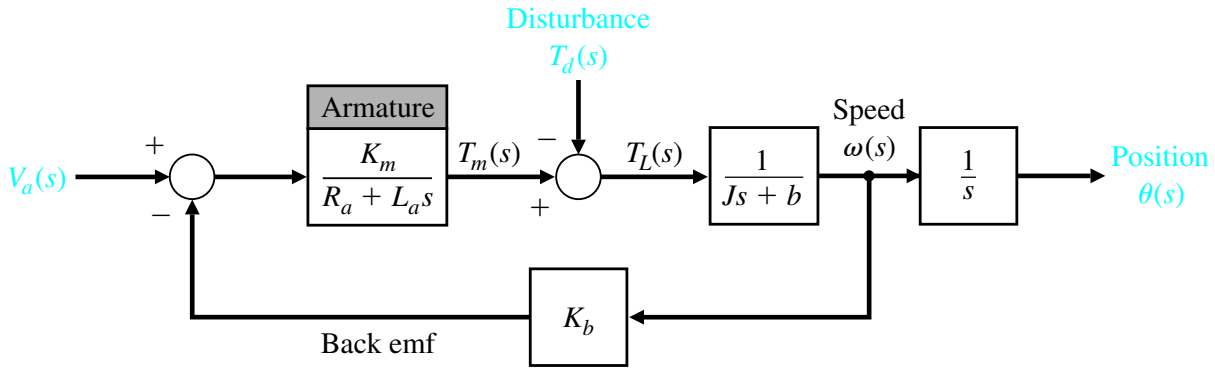


Figure 2: Armature-controlled DC motor.

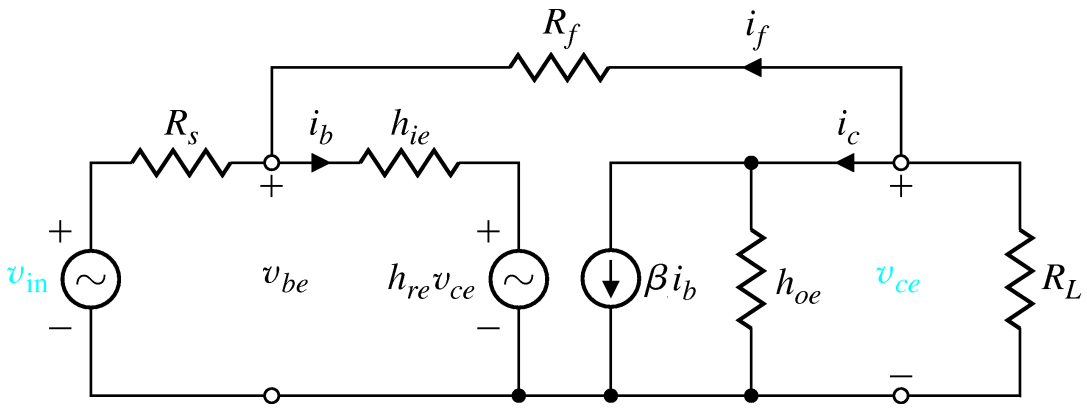
(a) For the same parameters as above and assuming $R_a = R_f$, $L_a = L_f$, $K_b = K_m$, draw the block diagram of the armature-controlled motor in Simulink. Use the pulse for the input and 0 for the disturbance torque T_d.

(b) Using the **Mux** block, connect the output of the motor θ, the torque T_m and the input V_a to a scope. Set the Stop time for the simulation to 40. Open the scope window and run the simulation. Click the right button on the scope window to Autoscale the plot. Print the result.

(c) Compare the performance of the armature-controlled motor with that of the field-controlled motor. In particular, comment on the torque response, T_m. Based on the shape of T_m, which configuration seems simpler to use?

In this exercise you will see how signal-flow graphs and Mason's formula can be used to analyze a function of a transistor. This way of modeling can be considered an alternative to standard circuit analysis.

The following figure shows the small signal circuit representation of a common-emitter transistor amplifier:



1. Draw a signal-flow graph representation of the circuit. Use the following nodes:

$$\begin{array}{ccc} v_{in} & v_{be} & v_{ce} \\ i_b & i_f & i_c \end{array}$$

and the following nodal equations:

$$\begin{aligned} i_b &= \frac{v_{in} - v_{be}}{R_s} + i_f \\ i_f &= \frac{v_{ce} - v_{be}}{R_f} \\ i_c &= \frac{v_{ce}}{h_{oe}} + \beta i_b \\ v_{be} &= h_{re}v_{ce} + h_{ie}i_b \\ v_{ce} &= -(i_f + i_c)R_L \end{aligned}$$

2. Using Mason's formula determine the transfer function between v_{in} and v_{ce}.

3. Represent the signal-flow graph with a block diagram in Simulink. Do not use numerical values, use variables to represent the transfer functions (i.e., use $1/R_f$ instead of 0.1 even if you know that $R_f = 10$). Print the block diagram. Use the **Signal Generator** from the **Sources** library as the input v_{in}.

4. In Matlab, create an M-file and define the following variables (use the names you used in Simulink):

$$R_s = 1\Omega \quad R_f = 100\Omega \quad R_L = 10k\Omega$$
$$h_{re} = 0.1 \quad h_{oe} = 50\Omega \quad h_{ie} = 2k\Omega$$
$$\beta = 1500$$

Run the simulation when v_{in} is a sawtooth waveform with the amplitude $5V$ and frequency $10Hz$. Plot v_{in}, v_{ce}, and $\frac{v_{ce}}{v_{in}}$. How can you check whether your derived transfer function is correct?

Signal-flow graphs are a good way to simulate a transfer function. In this exercise you will learn how to implement signal-flow graphs in Simulink and use them to study system response.

1. Find a signal-flow graph that corresponds to the controller canonical form (phase-variable canonical form) realization of the following transfer function:

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

   Write also the matrices that describe this system in the state-space form.

2. Using the `ss` function in Matlab define the state space model of the system. Use the `tf` function to convert the state-space representation into a transfer function and verify that your state space model corresponds to the transfer function $T(s)$.

3. Implement the signal-flow graph above in Simulink. Plot the response of the system to the unit step input. What can you say about this system?

4. To improve the performance of the system we can design a feedback controller. If $u(t)$ is the input signal and x is the state, the so-called **state feedback controller** is described by equation:

$$u = v - Kx$$

   where v is the new input and K is a 1×3 vector called **the gain vector**. Modify the signal-flow graph above to implement a state feedback controller and plot the step response of the resulting system for:

   (a) $K = [8\ 8\ 4]$

   (b) $K = [5\ 5\ 3]$

   (c) $K = [12\ 14\ 6]$

5. Based on your analysis, suggest the best controller for the system.

Print all the plots and submit them with your report.

Open and closed loop systems respond in different ways to disturbances and effects of unmodeled dynamics. In this exercise you will compare open and closed systems using Matlab and Simulink. The purpose of the exercise is to show you how having numerical tools allows a designer to quickly evaluate different design alternatives.

A remotely controlled vehicle has the transfer function:

$$G(s) = \frac{1}{s^2 + 2s + 4}$$

We want to compare an open loop controller with a closed loop controller.

1. Design a proportional open loop controller:

$$H(s) = K.$$

Figure 1 shows the diagram of the overall system. On the figure, $R(s)$ is the input and $D(s)$ is a disturbance.

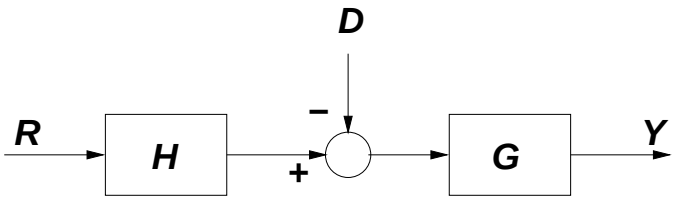


Figure 1

(a) Plot the response of the system to a step input (assuming $D(s) = 0$) for $K = 1, 4, 10$. How does a change in K affect the response? Which K would you choose to get a steady-state error equal to zero?

(b) Plot the response of the system to a unit step disturbance (assuming $R(s) = 0$). Evaluate the steady-state error for a disturbance numerically (from the plot) and analytically. Does K affect the disturbance response of the system?

(c) What is the sensitivity of the open-loop transfer function to changes in K?

2. Now design a closed-loop controller:

$$H(s) = \frac{K(s+2)}{(s+1)}.$$

The overall system is shown in Figure 2.

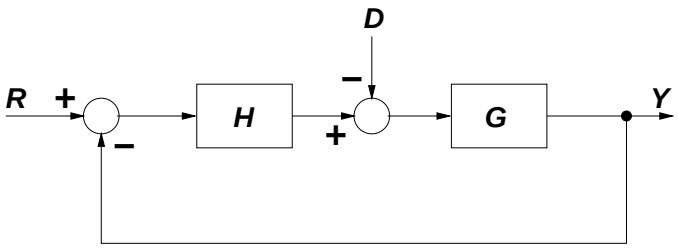


Figure 2

(a) Plot the response of the system to a step input (assuming $D(s) = 0$) for $K = 4.5, 10, 20$. Create a table and compare the three responses with respect to the:

- The time at which the response reaches its peak.
- The steady-state error.
- Percentage the system overshoots the steady-state value. Compute this value as $\frac{M_p - F_s}{F_s}$ where M_p is the peak value and F_s is the steady-state value.

Which K would you choose for the system based on your table?

(b) Plot the response of the system to a unit step disturbance (assuming $R(s) = 0$) for the same values of K. In each case, determine the steady-state error of the system. Which K gives the smallest error?

(c) Evaluate the sensitivity of the closed-loop transfer function to changes in K. Using the nominal value $K = 10$, plot a graph of the sensitivity function on the complex plane for frequencies $\omega = 0.01 rad/s$ to $\omega = 100 rad/s$.

Hint: Figure 4.33 in Dorf & Bishop shows you how to do this.

3. Based on your analysis, suggest the best design for the system.

Print all the plots and submit them with your report.

In this exercise you will explore a typical response of first and second order systems. All other systems can be usually approximated with first or second order systems so it is important to be well acquainted with these typical responses.

1. Consider the first order system:
$$G(s) = \frac{p}{s+p}$$
where p is a parameter. On the same figure, plot the step response of the system for $p = 1, 2, 10$. How does p affect the response?

2. Now consider a second order system:
$$G(s) = \frac{K}{s^2 + ps + K}$$
where K and p are the parameters of the system.

   (a) For $K = 1$, plot the step response of the system for $p = 0, 1, 2, 5$. Plot all the responses on the same figure! Comment on the effect of p on the response. On one figure, sketch the location of the poles of the system in the complex plane for each value of p (you can use the function **pzmap** to find the location of the poles).

   (b) On one figure, plot the step response of the system for $K = 1, 2, 10$ and $p = 0.8K$. How does the response change in this case? Sketch the location of the poles of the system in the complex plane for each value of K on one figure.

3. Now consider a second order system with a zero:
$$G(s) = \frac{(\gamma s + 1)}{s^2 + 1.2s + 1}$$
On the same figure, plot the step response of the system for $\gamma = 0, 0.1, 1, 10$. Note that $\gamma = 0$ corresponds to the case when the system has no zeros. Identify the location of the zero for each value of γ. On one figure, sketch the location of the poles and the zero for each value of γ. How does a zero affect the response of a second order system?

4. Consider the third order system:
$$G(s) = \frac{1}{(s^2 + s + 1)(\gamma s + 1)}$$
On the same figure, plot the step response of the system for $\gamma = 0, 0.05, 0.1, 1, 10$. Note that $\gamma = 0$ corresponds to the case when the system has no additional pole. Identify the location of the pole for each value of γ and sketch the location of all the poles on the complex plane (again, do this on one figure). How does a third pole affect the response of a second order system?

ECE 350 Laboratory 9 Spring 2013

High order systems can be usually well approximated as a first or a second order system. This is important for design of controllers since it significantly simplifies the problem. However, care needs to be taken with approximation so that all the important properties of the system are preserved. This exercise will introduce you to some of these issues.

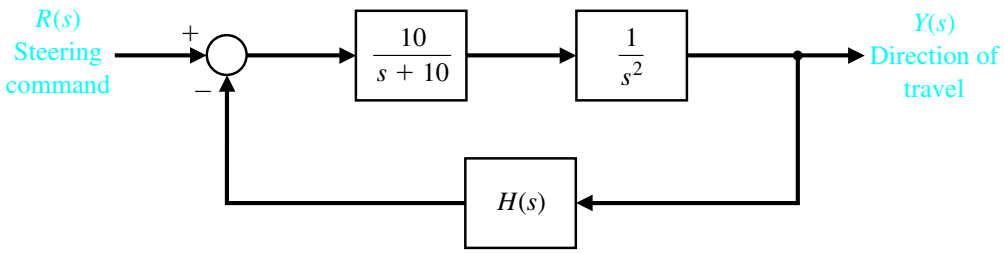
1. Consider the system:
$$G(s) = \frac{15}{(s^2 + 2s + 2)(s + 8)}$$
   (a) Plot the step response of the system. Determine the percent overshoot, the peak time, and the steady-step error for the step response.
   (b) We would like to approximate the system with a second-order system
$$\widehat{G}(s) = \frac{K}{s^2 + 2s + 2}.$$
   Determine the value K that will result in the same steady-state error as the original system. Plot the step response of $\widehat{G}$ for this value of K. Determine the percent overshoot, the peak time and the steady-state error and compare them with those of the original system. What can you say about this approximation?

2. Consider the system:
$$G(s) = \frac{0.1(s + 8)}{(s^2 + 1.5s + 1)}$$
   (a) Plot the step response of the system. Determine the percent overshoot, the peak time, and the steady-step error.
   (b) We would like to approximate the system with a system without a zero:
$$\widehat{G}(s) = \frac{K}{s^2 + 1.5s + 1}.$$
   Determine the value K that will result in the same steady-state error as the original system. Plot the step response of $\widehat{G}$ for this value of K. Determine the percent overshoot, the peak time and the steady-state error and compare them with those of the original system. What can you say about this approximation?

3. Now consider the following system:
$$G(s) = \frac{12(s + 4)}{(s^2 + 3s + 4)(s + 12)}$$
   Find a second order system that best approximates this system and has the minimal number of zeros. Explain how you arrived at your solution and provide plots that support your decision.

Stability is the most important property that a closed-loop system should posses. The first step in the design of a control system is thus to identify the values of the parameters for which the system is stable. Within the region of stability, the parameters can be subsequently adjusted for optimal performance. The purpose of this exercise is to introduce you to this two-level process. **Note that there are typically many choices of the parameter values that will achieve the desired performance.**

1. An automatically guided vehicle on Mars is represented by the following system:



The system has a steerable wheel in both the front and back of the vehicle, and the design requires that $H(s) = Ks + 1$. Determine

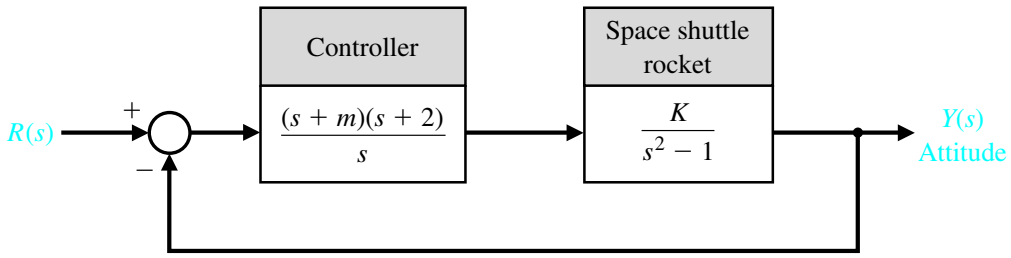
(a) The value of K required for stability.

(b) The value of K when one root of the characteristic equation is equal to $s = -5$.

(c) The value of the two remaining roots for the gain selected in (b).

(d) Find the step response for the gain selected in part (b).

2. The following figure shows the attitude control system of a space shuttle rocket:



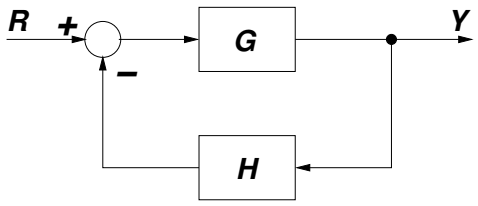
(a) Determine the range of gain K and parameter m so that the system is stable. Plot the region of stability on the K vs. m plot.

(b) Select the gain and parameter values so that the steady-state error to a ramp input is less than or equal to 10% of the input magnitude.

(c) Determine the percent overshoot for a step input for the system selected in part (b).

Design of controllers is usually an iterative process. A controller is typically designed for an approximation of the system. Subsequently, it is applied to the full system and refined until all the specifications are met. In this exercise you will go over these steps. **Note that there are typically many choices of the parameter values that will achieve the desired performance.**

1. Consider the closed-loop system:



where

$$G(s) = \frac{K_1(s+2)}{s(s+1)},$$

and

$$H(s) = 1 + K_2 s.$$

We want to select K_1 and K_2 so that the peak time is 0.5 second and the overshoot for a step input is less than 2%.

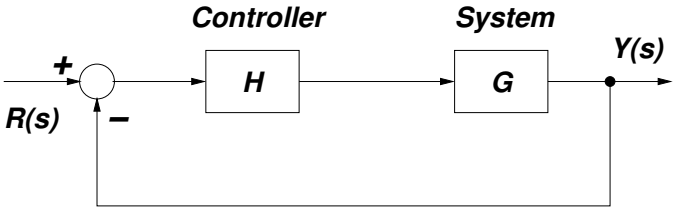
(a) Approximate $G(s)$ as a system without a zero. Make sure that the approximation has the same steady-state error as the original system.

(b) Find K_1 and K_2 for the approximated system that would satisfy the above specifications.

(c) Plot the step response of the original system with the chosen K_1 and K_2 and determine the peak time and the percent overshoot.

(d) Tune K_1 and K_2 so that the specifications are met for the original system. Explain how you arrived at your values (why did you increase or decrease the values chosen in 1b).

2. Consider the closed-loop system:



where

$$G(s) = \frac{10}{(s+1)(s+9)},$$

and

$$H(s) = \frac{K}{s+90}.$$

(a) Determine a second-order model for the closed-loop system by approximating $H(s)$ with its DC gain (the steady-state value of the step response).

(b) Using the second-order model, select a gain K so that percent overshoot is less than 15% and the steady-state error to a step is less than 12%.

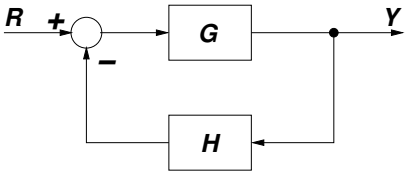
(c) Verify your design by determining the actual performance of the third-order system (submit the step response plot with your report).

(d) Tune the value of K so that the specifications are met. Plot the step response of the system to show that your design works. Explain how you arrived at your new value.

The Root Locus procedure was derived to help in the design of control systems. Matlab provides two functions for working with root locus: **rlocus** and **rlocfind**. The first sketches the root locus while the second enables the user to find the value of the parameter at a particular point on the root locus. But although one might be tempted to completely rely on the Matlab to draw the root locus, it is important to understand the steps that are involved in drawing an approximate root locus in order to effectively use it for control system design.

1. Using Matlab's Help system, read about the functions **rlocus** and **rlocfind**.

2. Consider the closed-loop system:



where

$$G(s) = \frac{s(3s^2 + 5s - 20)}{(s^2 - 2s + 10)(s + 3)(s + K)}$$

and

$$H(s) = 1.$$

(a) Find the characteristic polynomial of the closed-loop system.

(b) Transform the characteristic polynomial into the root-locus form.

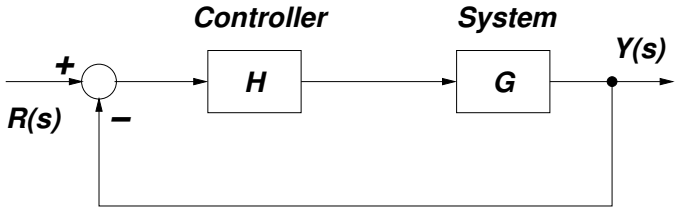
(c) Plot the root locus for the parameter K using Matlab.

(d) Using **rlocfind** find the value of K and the location of the roots where the root-locus breaks off from the real axis.

(e) Find the value of K and the location of the roots where the root-locus merges to the real axis.

(f) Find the value of K at which the system becomes unstable. Identify all the roots of the characteristic equation for this value of K.

3. Root locus is especially useful in design of controllers. Consider the following configuration:



where the system transfer function is

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

and $H(s)$ is a controller.

(a) Let $H(s) = K$. This is a so-called *proportional* controller. Using Matlab, plot the root locus for K. Can you stabilize the system using this controller?

(b) Now consider the controller in the form:

$$H(s) = K_1 + \frac{K_2}{s} + K_3 s. \tag{1}$$

This is so-called *PID controller*. In order to effectively use the root locus for designing the PID controller, we can represent the PID controller in the form:

$$H(s) = K\frac{(s+z_1)(s+z_2)}{s} \tag{2}$$

i. Find the relation between the location of the zeros z_1 and z_2 in Equation (2) and the parameters K_1, K_2, and K_3 in Equation (1).

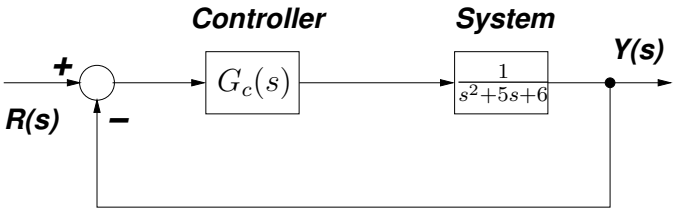
ii. Place the zeros of the controller at $-2 \pm 3j$ (that is, $z_{1,2} = 2 \mp 3j$). Plot the root locus of the system for K (using the form of $H(s)$ in Equation (2)). What can you say about the ability of this controller to stabilize the system?

iii. For the PID controller above, find the value of K for which the system's complex conjugate poles have their real part equal to -1. For this value of K, identify what are the corresponding parameters K_1, K_2, and K_3 in Equation (1).

PID controllers are widely used in industrial process control due to their good performance in a wide range of operating conditions and their functional simplicity. In this exercise you will explore how the PID controllers can be designed using the root locus procedure.

You are encouraged to take the advantage of the Matlab functions **rlocus** and **rlocfind** for this exercise.

1. Consider the following feedback control system:



We have three potential controllers for our system:

(i) $G_c(s) = K$ (proportional controller)

(ii) $G_c(s) = K/s$ (integral controller)

(iii) $G_c(s) = K(1 + 1/s)$ (proportional, integral (PI) controller)

The design specifications are $T_s \leq 10s$ and $PO \leq 10\%$ for a unit step input.

(a) For the proportional controller, sketch the root locus, and determine the value of K such that the design specifications are satisfied.

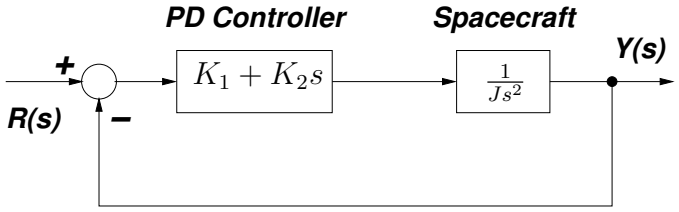
(b) Repeat part 1a for the integral controller.

(c) Repeat part 1a for the PI controller.

(d) Co-plot the unit step responses for the closed-loop systems with each controller designed in parts 1a-1c.

(e) Compare and contrast the three controllers obtained in parts 1a-1c, concentrating your discussion on the steady-state errors and transient performance.

2. Consider the spacecraft single-axis attitude control system:



The controller is known as proportional-derivative (PD) controller.

(a) Sketch the root locus for K_1 when $K_2 = 0$.

(b) Suppose that we require the ratio of $K_1/K_2 = 5$. Sketch the root locus for the system and discuss how it differs from the previous case.

(c) Using the root locus, find the values of K_2/J and K_1/J in part (b) such that $T_s \leq 4s$ and $PO \leq 10\%$ for a unit step input.

This laboratory exercise is optional. If you hand it in you will receive extra credit.

Control Toolbox in Matlab provides a powerful tool for helping with the design of the control systems called "`sisotool`". You can read about the tool at `http://www.mathworks.com/help/toolbox/control/getstart/f2-1037701.html`. In this exercise you will use "`sisotool`" to solve two problems requiring design of controllers.

Before using "`sisotool`", define systems in the main Matlab window. You can then import these systems directly into "`sisotool`" as explained in the tutorial (click **Architecture** $\rightarrow$ **System Data** in "`sistool`" window). For example, to define the transfer function:

$$G_1(s) = \frac{1}{(s+2)(s+4)(s+6)(s+8)}$$

you can either use

```
g1=zpk([],[-2 -4 -6 -8],1)
```

or

```
s=tf('s');
g1=1/((s+2)*(s+4)*(s+6)*(s+8))
```

Use "`sisotool`" to solve the following problems:

1. Take a unity feedback system with

$$G(s) = \frac{K}{(s+2)(s+4)(s+6)(s+8)}.$$

   First, find K so that the uncompensated system will have a damping ratio of 0.5. Then find the transfer function of a lead-lag compensator that will yield a settling time 0.5 second shorter than that of the uncompensated system, will also result in the damping ratio of 0.5, and will improve the steady-state error by a factor of 30. The compensator zero should be at -5. Find the compensated system's gain. Justify any second-order approximations and verify the design through simulation.

2. Take a unity feedback system with

$$G(s) = \frac{K}{(s+1)(s+4)}.$$

Design a PID controller that will yield a peak time of 1.047 seconds and a damping ratio of 0.8, with zero error for a step input. Remember that you can rewrite the PID controller as:

$$K_p + \frac{K_i}{s} + K_d s = K_c \frac{(s + z_1)(s + z_2)}{s}.$$

One zero and the pole of the compensator can be then designed as a PI compensator, while the other zero can be designed as a PD compensator.