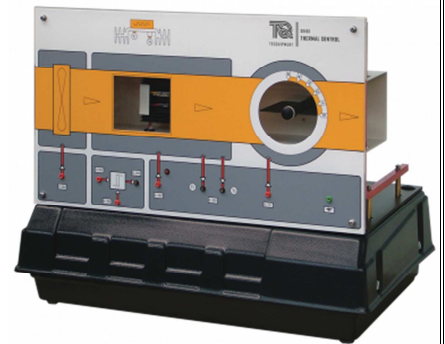
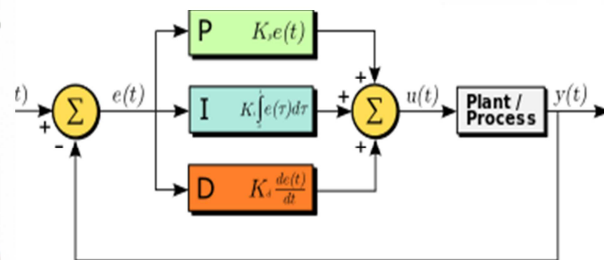
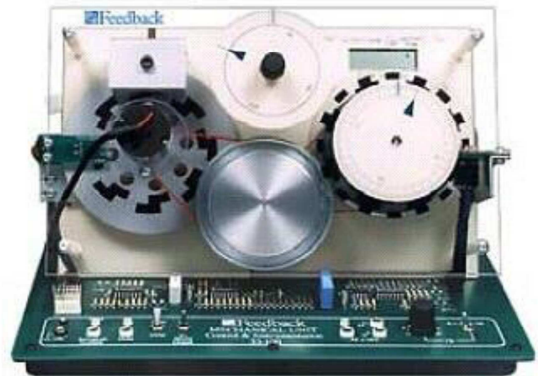




University of Jordan
Mechatronics Engineering Department
Control Laboratory Manual
MX-0908453

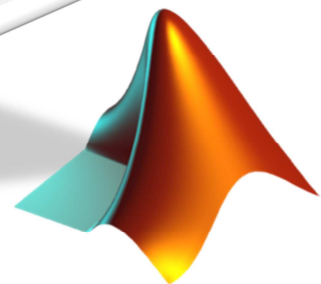
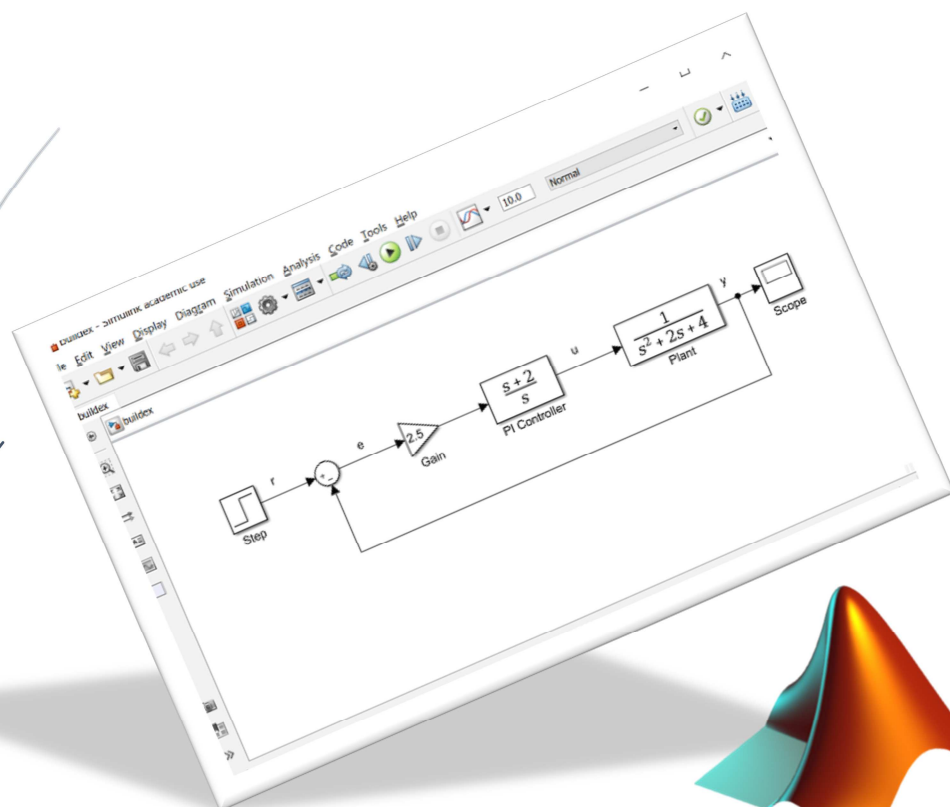


Experiment One

Mathematical Modelling Using Simulink

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



1. Mathematical Model Definition

A **mathematical model** is a description of a system using mathematical concepts and language. The process of developing a mathematical model is termed **mathematical modeling**. Mathematical models are used in the natural sciences (such as physics, biology, earth science, meteorology) and engineering disciplines (such as computer science, artificial intelligence), as well as in the social sciences (such as economics, psychology, sociology, political science).

Physicists, engineers, statisticians, operations research analysts, and economists use mathematical models most extensively. A model may help to explain a system and to study the effects of different components, and to make predictions about behavior.

Mathematical models can take many forms, including but not limited to dynamical systems, statistical models, differential equations, or game theoretic models. These and other types of models can overlap, with a given model involving a variety of abstract structures. In general, mathematical models may include logical models. In many cases, the quality of a scientific field depends on how well the mathematical models developed on the theoretical side agree with results of repeatable experiments. Lack of agreement between theoretical mathematical models and experimental measurements often leads to important advances as better theories are developed.

2. Model Classifications in Mathematics

Mathematical models are usually composed of relationships and variables. Relationships can be described by operators, such as algebraic operators, functions, differential operators, etc. Variables are abstractions of system parameters of interest, that can be quantified. Several classification criteria can be used for mathematical models according to their structure:

- **Linear vs. nonlinear:** If all the operators in a mathematical model exhibit linearity, the resulting mathematical model is defined as linear. A model is considered to be nonlinear otherwise. The definition of linearity and nonlinearity is dependent on context, and linear models may have nonlinear expressions in them. For example, in a statistical linear model, it is assumed that a relationship is linear in the parameters, but it may be nonlinear in the predictor variables. Similarly, a differential equation is said to be linear if it can be written with linear differential operators, but it can still have nonlinear expressions in it. In a mathematical programming model, if the objective functions and constraints are represented entirely by linear equations, then the model is regarded as a linear model. If one or more of the objective functions or constraints are represented with a nonlinear equation, then the model is known as a nonlinear model. Nonlinearity, even in fairly simple systems, is often associated with phenomena such as chaos and irreversibility. Although there are exceptions, nonlinear systems and models tend to be more difficult to study than linear ones. A common approach to nonlinear problems is linearization, but this can be problematic if one is trying to study aspects such as irreversibility, which are strongly tied to nonlinearity.

Experiment 1: Mathematical Modelling Using Simulink

- **Static vs. Dynamic:** A dynamic model accounts for time-dependent changes in the state of the system, while a static (or steady-state) model calculates the system in equilibrium, and thus is time-invariant. Dynamic models typically are represented by differential equations.
- **Explicit vs. Implicit:** If all of the input parameters of the overall model are known, and the output parameters can be calculated by a finite series of computations (known as linear programming, not to be confused with linearity as described above), the model is said to be explicit. But sometimes it is the output parameters which are known, and the corresponding inputs must be solved for by an iterative procedure, such as Newton's method (if the model is linear) or Broyden's method (if non-linear). For example, a jet engine's physical properties such as turbine and nozzle throat areas can be explicitly calculated given a design thermodynamic cycle (air and fuel flow rates, pressures, and temperatures) at a specific flight condition and power setting, but the engine's operating cycles at other flight conditions and power settings cannot be explicitly calculated from the constant physical properties.
- **Discrete vs. Continuous:** A discrete model treats objects as discrete, such as the particles in a molecular model or the states in a statistical model; while a continuous model represents the objects in a continuous manner, such as the velocity field of fluid in pipe flows, temperatures and stresses in a solid, and electric field that applies continuously over the entire model due to a point charge.
- **Deterministic vs. Probabilistic (stochastic):** A deterministic model is one in which every set of variable states is uniquely determined by parameters in the model and by sets of previous states of these variables; therefore, a deterministic model always performs the same way for a given set of initial conditions. Conversely, in a stochastic model—usually called a "statistical model"—randomness is present, and variable states are not described by unique values, but rather by probability distributions.
- **Deductive, Inductive, or Floating:** A deductive model is a logical structure based on a theory. An inductive model arises from empirical findings and generalization from them. The floating model rests on neither theory nor observation, but is merely the invocation of expected structure. Application of mathematics in social sciences outside of economics has been criticized for unfounded models. Application of catastrophe theory in science has been characterized as a floating model.

3. Introduction to Simulink

3.1 Simulink Description:

Simulink® is a block diagram environment for multidomain simulation and Model-Based Design. It supports system-level design, simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink provides a graphical editor, customizable block libraries, and solvers for modeling and simulating dynamic systems. It is integrated with MATLAB®, enabling you to incorporate MATLAB algorithms into models and export simulation results to MATLAB for further analysis.

3.2 Simulink Features:



- Graphical editor for building and managing hierarchical block diagrams
- Libraries of predefined blocks for modeling continuous-time and discrete-time systems
- Simulation engine with fixed-step and variable-step ODE solvers
- Scopes and data displays for viewing simulation results
- Project and data management tools for managing model files and data
- Model analysis tools for refining model architecture and increasing simulation speed
- MATLAB Function block for importing MATLAB algorithms into models
- Legacy Code Tool for importing C and C++ code into models

3.3 Open the Simulink Library Browser:

You need MATLAB® running before you can open the Simulink® Library Browser.

1. In the MATLAB Command Window, enter **simulink**.

A short delay occurs the first time you open the Simulink Library Browser. The figure 1 shows the Library Browser with the Out1 block selected in the Simulink/Commonly Used Blocks sub library. A block description that appears when you hover over it.

2. The Library Browser keeps a repository of all the libraries it shows. If your library has missing repository information, a notification bar appears above the Libraries pane when you refresh the Library Browser. To prevent this notification from appearing again, click on Fix in the notification bar and choose Resave libraries in SLX file format. This saves all libraries in .slx format with Enable LBRepository property set to on. You can also open the Simulink Library Browser from the MATLAB Toolstrip, by clicking  the Simulink Library button. To keep the Library Browser above all other windows on your desktop, in the toolbar, select  the Stay on top button .

Experiment 1: Mathematical Modelling Using Simulink

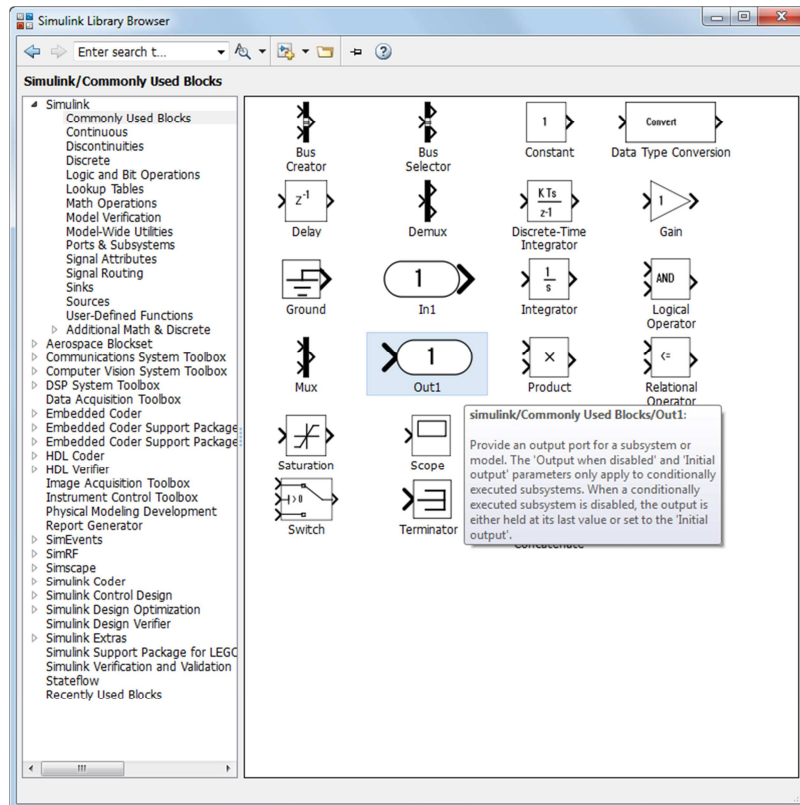



Figure 1.1: Simulink Library

3.4 Create a New Simulink Model

Create a new Simulink model from the Simulink Library Browser.

1. From the Simulink Library Browser toolbar, click the New Model button .  An empty model diagram figure 1.2 opens in the Simulink Editor.

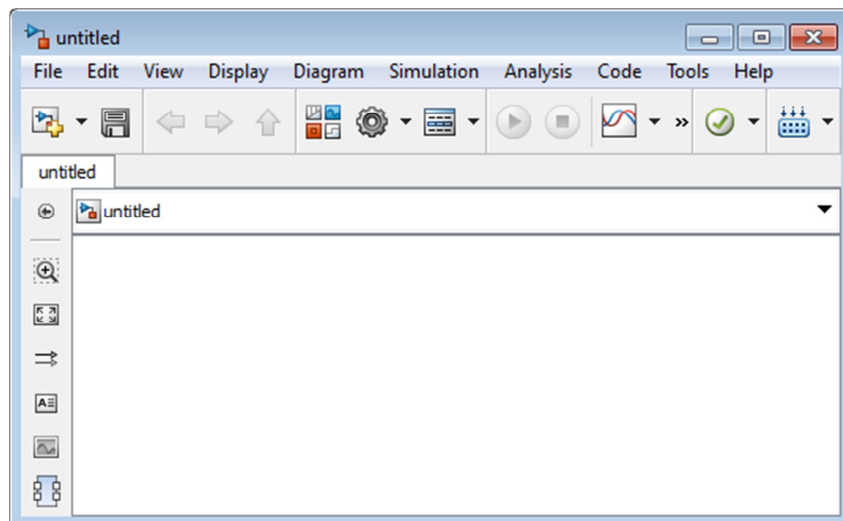


Figure 1.2: Simulink New Model

Example 1: Model the differential Equation (Time Domain)

$$\dot{x}(t) = 2x(t) + u(t) \tag{1}$$

where $u(t)$ is a square wave with an *amplitude of 1 and a frequency of 1 rad/sec*, use an integrator block and a gain block.

The Integrator block integrates its input $\dot{x}(t)$ to produce $x(t)$. Other blocks needed in this model include a Gain block and a Sum block. To generate a square wave, use a Signal Generator block and select the Square Wave form but change the default units to radians/sec. Again, view the output using a Scope block. Gather the blocks and define the gain.

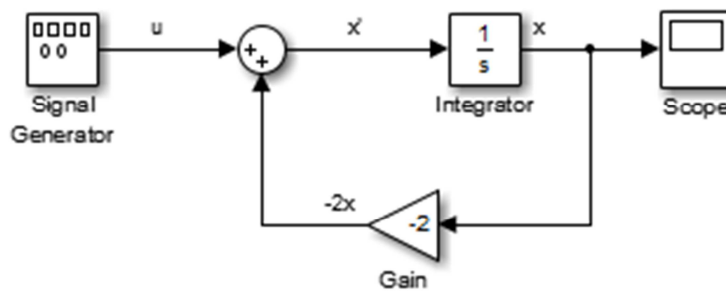


Figure 1.3: Blocks representation of equation 1

An important concept in this model is the loop that includes the Sum block, the Integrator block, and the Gain block.

In this equation, $x(t)$ is the output of the Integrator block. It is also the input to the blocks that compute $\dot{x}(t)$, on which it is based. This relationship is implemented using a loop.

The Scope displays $x(t)$ at each time step. For a simulation lasting 10 seconds, the output shows in figure 4:

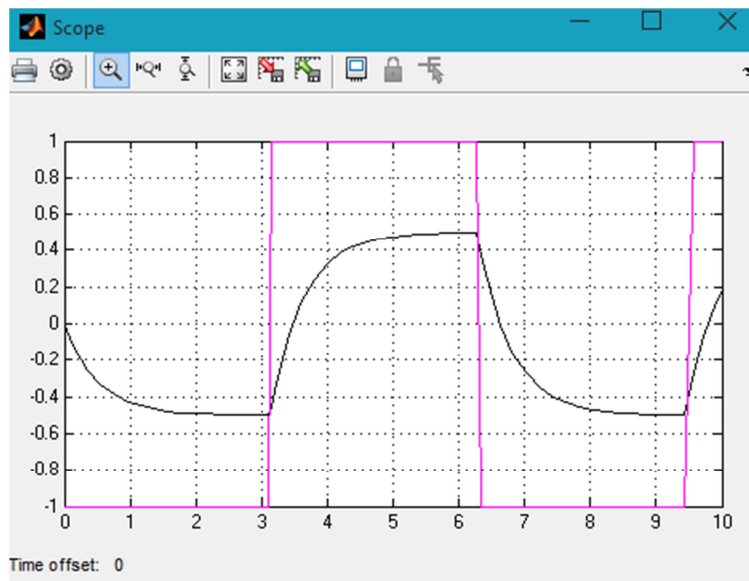


Figure 1.4: The simulation result of $x(t)$

Example .2: Model the differential Equation (Frequency Domain- Transfer Function)

$$\dot{x}(t) = 2x(t) + u(t) \quad (2)$$

The equation you modeled in this example can also be expressed as a transfer function. The model uses the **Transfer Fcn block**, which accepts u as input and outputs $x(t)$.

So, the block implements $(s)/U(s)$. If you substitute $sX(s)$ for $\dot{x}(t)$ in the above equation, you get

$$sX(s) = -2X(s) + U(s)$$

Solving for $X(s)$ gives

$$X(s) = \frac{U(s)}{s + 2}$$

Or

$$\frac{X(s)}{U(s)} = \frac{1}{s + 2}$$

The **Transfer Fcn** block uses parameters to specify the numerator and denominator coefficients. In this case, *the numerator is 1 and the denominator is $s+2$* . Specify both terms as vectors of coefficients of successively decreasing powers of s

In this case the numerator is [1] (or just 1) and the denominator is [1 2].

Experiment 1: Mathematical Modelling Using Simulink

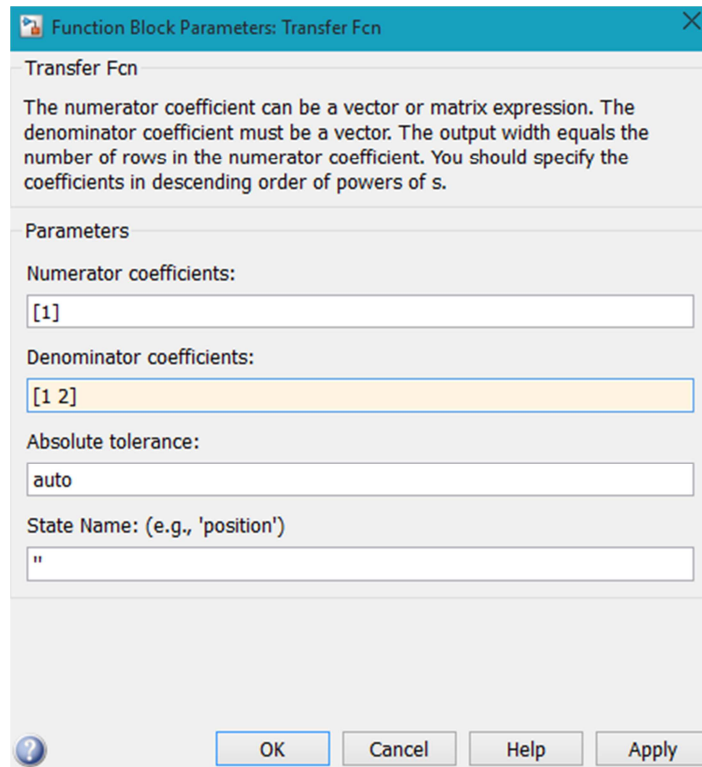


Figure 1.5: Transfer Fcn block parameter

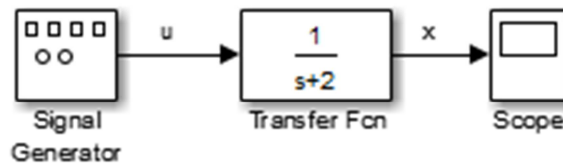


Figure 1.6: Transfer function representation of equation 2

Experiment 1: Mathematical Modelling Using Simulink

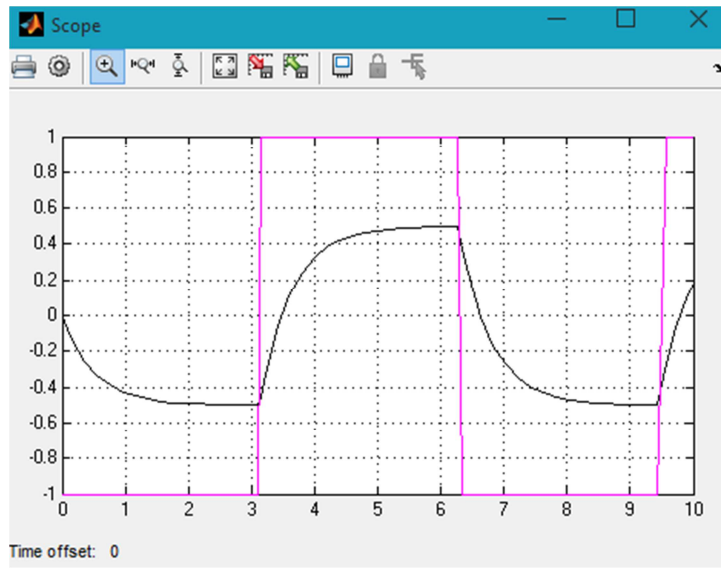


Figure 1.7: The simulation result of $x(t)$

Exercises:

Exercise1-Solving differential equations using SIMULINK:

Find the solution of the following differential equations using MATLAB- Simulink for a step input function:

$$\ddot{y} + 2\dot{y} + 4y = U(t)$$

$$\ddot{y} + 4y = U(t)$$

Exercise2-First order system:

The mathematical model of a first order system is given by:

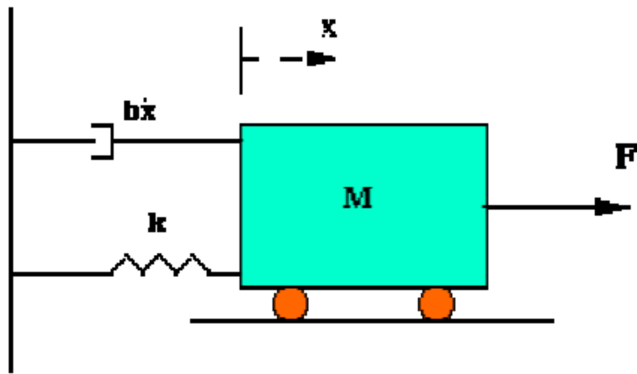
$$\dot{y} + ay = bu$$

1. Obtain the transfer function of the system.
2. Build a Simulink model for the system. With $a=1, b=1$. Run the model for $u=1, 5$ and 10 .
3. Find the time constant and the steady state value from the scope.
4. For $a=0.5, b=1$ and $u=1$, run the model and find the time constant.

Exercise3-Second order system:

A simple mass, spring and damper compose the mechanical system shown below

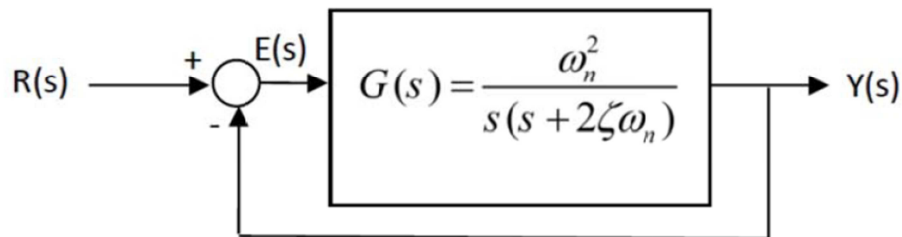
Experiment 1: Mathematical Modelling Using Simulink



where M is the mass, b is the damper, k is spring, $x(t)$ is the position of the cart and $F(t)$ is the force applied to the system.

- Find the dynamic equation (differential equation) of this system.
- Let $M=0.1\text{Kg}$, $b=1\text{ Ns/m}$ and $k=10$. Determine the transfer function of this system $X(S)/F(S)$.
- Calculate (from the transfer function) the %OS, T_r , T_s , T_p and s.s error.
- Build a SIMULINK model and plot the step response of the system and from the curve find %OS, s.s error, T_s , T_r and T_p .

Exercise4-The effect of the damping ratio on the performance of the system:



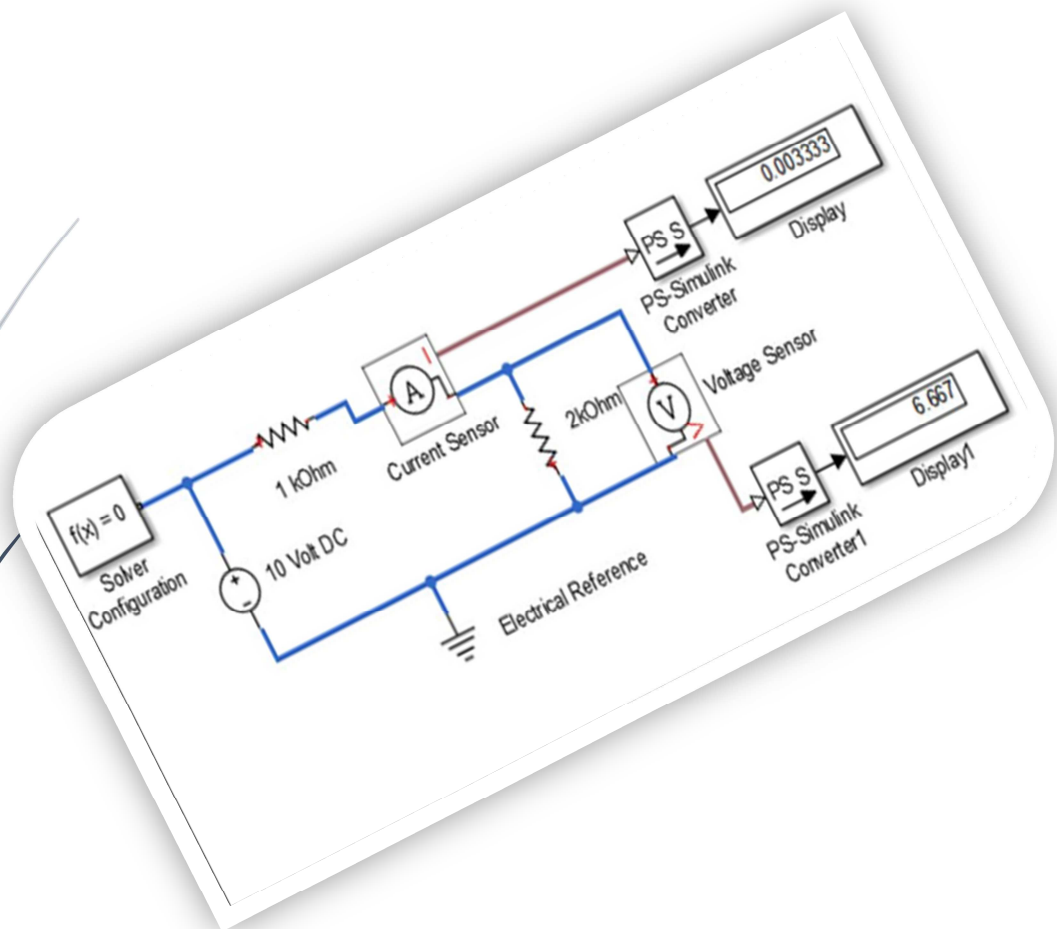
Build the simulink model using matlab, then find the step response of the system for $\omega_n=1$, and $\xi=0.1, 0.5, 0.8, 1$ and 2 .

Experiment Two

Mathematical Modelling Using SimScape (Electrical Systems)

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



1. Model and simulate MultiDomain Physical Systems

Simscape provides an environment for modeling and simulating physical systems spanning mechanical, electrical, hydraulic, and other physical domains. It provides fundamental building blocks from these domains that you can assemble into models of physical components, such as electric motors, inverting op-amps, hydraulic valves, and ratchet mechanisms. Because Simscape components use physical connections, your models match the structure of the system you are developing.

Simscape models can be used to develop control systems and test system-level performance. You can extend the libraries using the MATLAB[®] based Simscape language, which enables text-based authoring of physical modeling components, domains, and libraries. You can parameterize your models using MATLAB variables and expressions, and design control systems for your physical system in Simulink[®]. To deploy your models to other simulation environments, including hardware-in-the-loop (HIL) systems, Simscape supports C-code generation.

2. Key Features

- Single environment for modeling and simulating mechanical, electrical, hydraulic, thermal, and other multidomain physical systems
- Libraries of physical modeling blocks and mathematical elements for developing custom components
- MATLAB based Simscape language, enabling text-based authoring of physical modeling components, domains, and libraries
- Physical units for parameters and variables, with all unit conversions handled automatically
- Ability to simulate models that include blocks from related physical modeling products without purchasing those products
- Support for C-code generation

3. Simscape Block Libraries

Simscape block library contains two libraries that belong to the Simscape[™] product:

- Foundation library — Contains basic hydraulic, pneumatic, mechanical, electrical, magnetic, thermal, thermal liquid, and physical signal blocks, organized into sublibraries according to technical discipline and function performed
- Utilities library — Contains essential environment blocks for creating Physical Networks models

In addition, if you have installed any of the add-on products of the Physical Modeling family, you will see the corresponding libraries under the main Simscape library.

Simscape Foundation libraries contain a comprehensive set of basic elements and building blocks, such as:

- Mechanical building blocks for representing one-dimensional translational and rotational motion
- Electrical building blocks for representing electrical components and circuits
- Magnetic building blocks that represent electromagnetic components
- Hydraulic building blocks that model fundamental hydraulic effects and can be combined to

Experiment 2: Mathematical Modelling Using SimScape

create more complex hydraulic components

- Pneumatic building blocks that model fundamental pneumatic effects based on the ideal gas law
- Thermal building blocks that model fundamental thermal effects
- Thermal liquid building blocks that model fundamental thermodynamic effects in liquids
- Physical Signals block library that lets you perform math operations on physical signals, and graphically enter equations inside the physical network

Using the elements contained in these Foundation libraries, you can create more complex components that span different physical domains. You can then group this assembly of blocks into a subsystem and parameterize it to reuse and share these components.

In addition to Foundation libraries, there is also a Simscape Utilities library, which contains utility blocks, such as:

- Solver Configuration block, which contains parameters relevant to numerical algorithms for Simscape simulations. Each Simscape diagram (or each topologically distinct physical network in a diagram) must contain a Solver Configuration block.
- Simulink-PS Converter block and PS-Simulink Converter block, to connect Simscape and Simulink® blocks. Use the Simulink-PS Converter block to connect Simulink outputs to Physical Signal inputs. Use the PS-Simulink Converter block to connect Physical Signal outputs to Simulink inputs.

For examples of using these blocks in a Simscape model, see the tutorial [Creating and Simulating a Simple Model](#).

You can combine all these blocks in your Simscape diagrams to model physical systems. You can also use the basic Simulink blocks in your diagrams, such as sources or scopes. See [Connecting Simscape Diagrams to Simulink Sources and Scopes](#) for more information on how to do this.

Simscape block libraries contain a comprehensive selection of blocks that represent engineering components such as valves, resistors, springs, and so on. These prebuilt blocks, however, may not be sufficient to address your particular engineering needs. When you need to extend the existing block libraries, use the Simscape language to define customized components, or even new physical domains, as textual files. Then convert your textual components into libraries of additional Simscape blocks that you can use in your model diagrams.

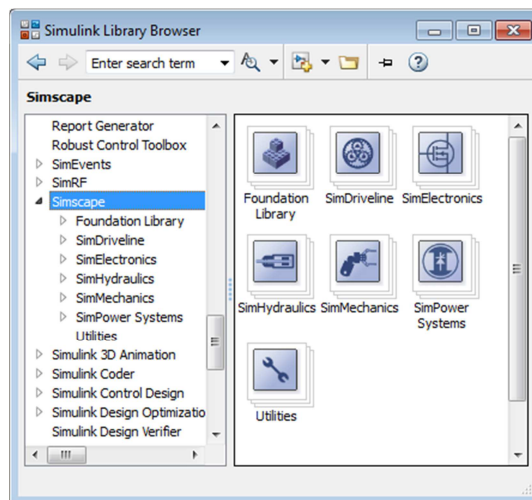


Figure 2.1: SimScape Library

Experiment 2: Mathematical Modelling Using SimScape

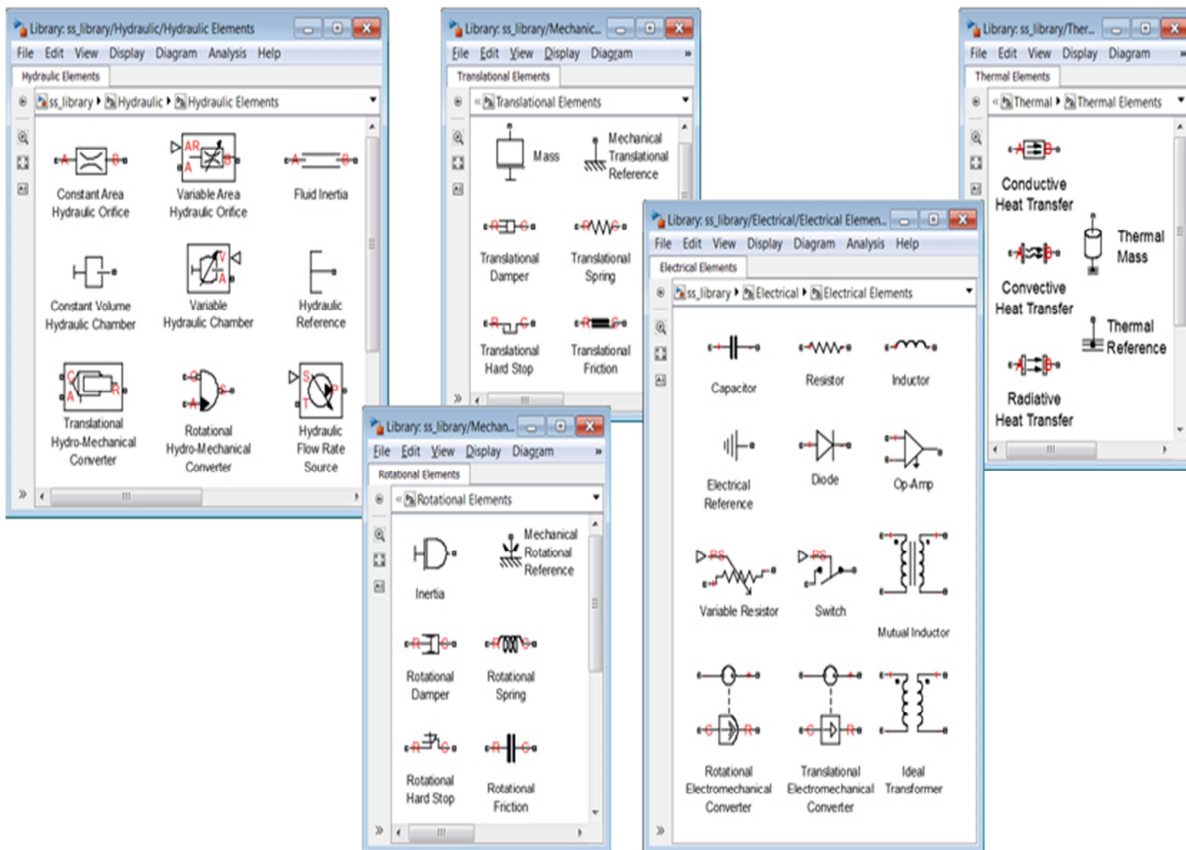


Figure 2.2: SimScape Library

4. How Simscape Simulation Works

Simscape software gives you multiple ways to simulate and analyze physical systems in the Simulink environment. Running a physical model simulation is similar to simulating any Simulink model. It entails setting various simulation options, starting the simulation, and viewing the simulation results. This topic describes various aspects of simulation specific to Simscape models. For specifics of simulating and analyzing with individual Simscape add-on products, refer to the documentation for those individual add-on products.

The flow chart in Figure 2.3 presents the Simscape simulation sequence.

The flow chart consists of the following major phases:

1. Model Validation
2. Network Construction
3. Equation Construction
4. Initial Conditions Computation
5. Transient Initialization
6. Transient Solve

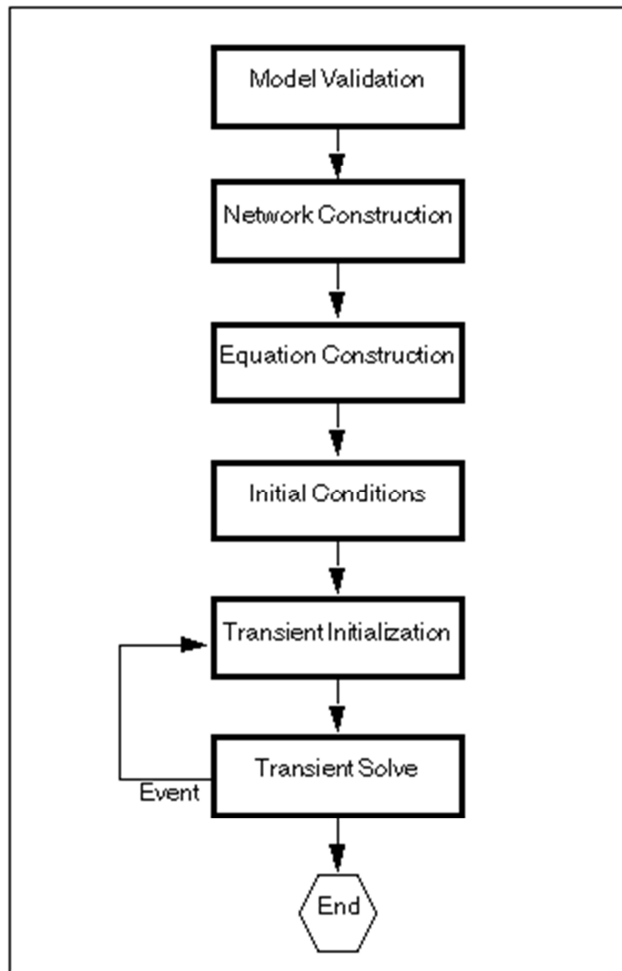


Figure 2.3: flow chart presents the Simscape

4.1 Model Validation

The Simscape solver first validates the model configuration and checks your data entries from the block dialog boxes.

All Simscape blocks in a diagram must be connected into one or more physical networks. Unconnected Conserving ports are not allowed.

- Each topologically distinct physical network in a diagram requires exactly one Solver Configuration block.
- If your model contains hydraulic elements, each topologically distinct hydraulic circuit in a diagram must connect to a Custom Hydraulic Fluid block (or Hydraulic Fluid block, available with SimHydraulics® block libraries). These blocks define the fluid properties that act as global parameters for all the blocks that connect to the hydraulic circuit. If no hydraulic fluid block is attached to a loop, the hydraulic blocks in this loop use the default fluid. However, more than one hydraulic fluid block in a loop generates an error.
- Similarly, if your model contains pneumatic elements, default gas properties for a pneumatic network are for dry air and ambient conditions of 101325 Pa and 20 degrees Celsius. If you attach a Gas Properties block to a pneumatic circuit, you can change gas properties and ambient conditions for all the blocks connected to the circuit. However, more than one Gas

Properties block in a pneumatic circuit generates an error.

- Signal units specified in a Simulink-PS Converter block must match the input type expected by the Simscape block connected to it. For example, when you provide the input signal for an Ideal Angular Velocity Source block, specify angular velocity units, such as rad/s or rpm, in the Simulink-PS Converter block, or leave it unitless. Similarly, units specified in a PS-Simulink Converter block must match the type of physical signal provided by the Simscape block output.

4.2 Network Construction

After validating the model, the Simscape solver constructs the physical network based on the following principles:

- Two directly connected Conserving ports have the same values for all their Across variables (such as voltage or angular velocity).
- Any Through variable (such as current or torque) transferred along the Physical connection line is divided among the multiple components connected by the branches. For each Through variable, the sum of all its values flowing into a branch point equals the sum of all its values flowing out.

4.3 Equation Construction

Based on the network configuration, the parameter values in the block dialog boxes, and the global parameters defined by the fluid properties, if applicable, the Simscape solver constructs the system of equations for the model.

These equations contain system variables of the following types:

- **Dynamic** : Time derivatives of these variables appear in equations. Dynamic, or differential, variables add dynamics to the system and require the solver to use numerical integration to compute their values. Dynamic variables can produce either independent or dependent states for simulation.
- **Algebraic** : Time derivatives of these variables do not appear in equations. These variables appear in algebraic equations but add no dynamics, and this typically occurs in physical systems due to conservation laws, such as conservation of mass and energy. The states of algebraic variables are always dependent on dynamic variables, other algebraic variables, or inputs.

The solver then performs the analysis and eliminates variables that are not needed to solve the system of equations. After variable elimination, the remaining variables (algebraic, dynamic dependent, and dynamic independent) get mapped to Simulink state vector of the model.

4.4 Initial Conditions Computation

The Simscape solver computes the initial conditions only once, at the beginning of simulation

($t = 0$). In the Solver Configuration block dialog box, the default is that the Start simulation from steady state check box is not selected. If it is selected in your model, see Finding an Initial Steady State.

The solver computes the initial conditions by finding initial values for all the system variables that exactly satisfy all the model equations. You can affect the initial conditions computation by block-level variable initialization, that is, by specifying the priority and target initial values on the Variables tab of the block dialog boxes.

The values you specify during block-level variable initialization are not the actual values of the respective variables, but rather their target values at the beginning of simulation ($t = 0$). Depending on the results of the solve, some of these targets may or may not be satisfied. The solver tries to satisfy the high-priority targets first, then the low-priority ones:

- At first, the solver tries to find a solution where all the high-priority variable targets are met exactly, and the low-priority targets are approximated as closely as possible. If the solution is found during this stage, it satisfies all the high-priority targets. Some of the low-priority targets might also be met exactly, the others are approximated.
- If the solver cannot find a solution that exactly satisfies all the high-priority targets, it issues a warning and enters the second stage, where High priority is relaxed to Low. That is, the solver tries to find a solution by approximating both the high-priority and the low-priority targets as closely as possible.

After you initialize the block variables and prior to simulating the model, you can open the Variable Viewer to see which of the variable targets have been satisfied. For more information on block-level variable initialization, see Variable Initialization.

Finding an Initial Steady State

When you select the Start simulation from steady state check box, the solver attempts to find the steady state that would result if the inputs to the system were held constant for a long enough time, starting from the initial state obtained from the initial conditions computation just described. If the steady-state solve succeeds, the state found is some steady state (within tolerance), but not necessarily the state expected from the given initial conditions. Steady state means that the system variables are no longer changing with time. Simulation then starts from this steady state.

A model can have more than one steady state. In this case, the solver selects the steady-state solution that is consistent with the variable targets specified during block-level variable initialization.

4.5 Transient Initialization

After computing the initial conditions, or after a subsequent event (such as a discontinuity resulting, for example, from a valve opening, or from a hard stop), the Simscape solver performs transient initialization. Transient initialization fixes all dynamic variables and solves for algebraic variables and derivatives of dynamic variables. The goal of transient initialization is to provide a consistent set of initial conditions for the next phase, transient solve.

4.6 Transient Solve

Finally, the Simscape solver performs transient solve of the system of equations. In transient solve, continuous differential equations are integrated in time to compute all the variables as a function of time.

The solver continues to perform the simulation according to the results of the transient solve until the solver encounters an event, such as a zero crossing or discontinuity. The event may be within the physical network or elsewhere in the Simulink model. If the solver encounters an event, the solver returns to the phase of transient initialization, and then back to transient solve. This cycle continues until the end of simulation.

5. Electrical System Using SimScape

In this example, you are going to model an electrical system and observe its behavior under various conditions. This tutorial illustrates the essential steps to building a physical model and makes you familiar with using the basic Simscape blocks.

The Figure 2.4 shows a simple model of an electrical circuit. It consists of a Resistance, Capacitor and Inductor, which is supplied by a Voltage source.

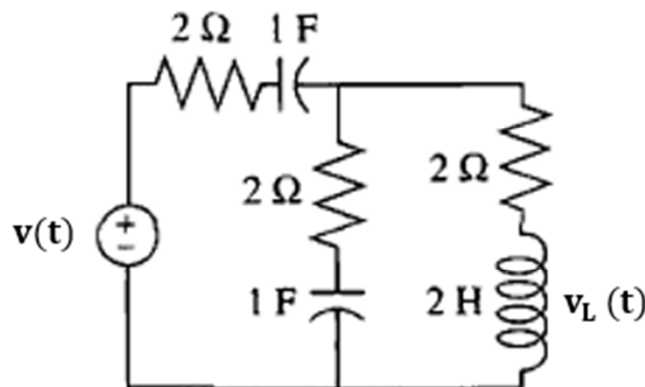


Figure 2.4: Electrical Circuit

- **Modeling of System**

To create an equivalent Simscape diagram, follow these steps:

1. Open the Simulink® Library Browser, as described in Simscape Block Libraries.
2. Create a new model. To do this, from the top menu bar of the Library Browser, select File > New > Model. The software creates an empty model in memory and displays it in a new model editor window.
3. Open the Simscape > Foundation Library > Electrical Element Elements library as shown in Figure 2.5.

Experiment 2: Mathematical Modelling Using SimScape

4. Drag the Mass, Translational Spring, Translational Damper, and two Mechanical Translational Reference blocks into the model window.
5. Orient the blocks as shown in the Figure 2.6. To rotate a block, select it and press Ctrl+R.
6. Connect the Resistor, Capacitor, and Inductor blocks to one of the Electrical Reference Ground blocks as shown in Figure 2.7.

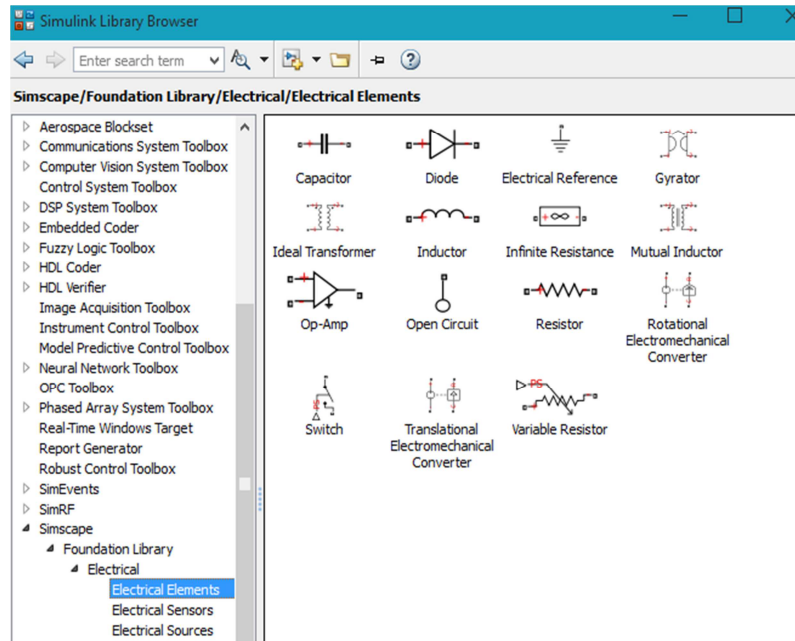


Figure 2.5: Electrical Element, step 3

Experiment 2: Mathematical Modelling Using SimScape

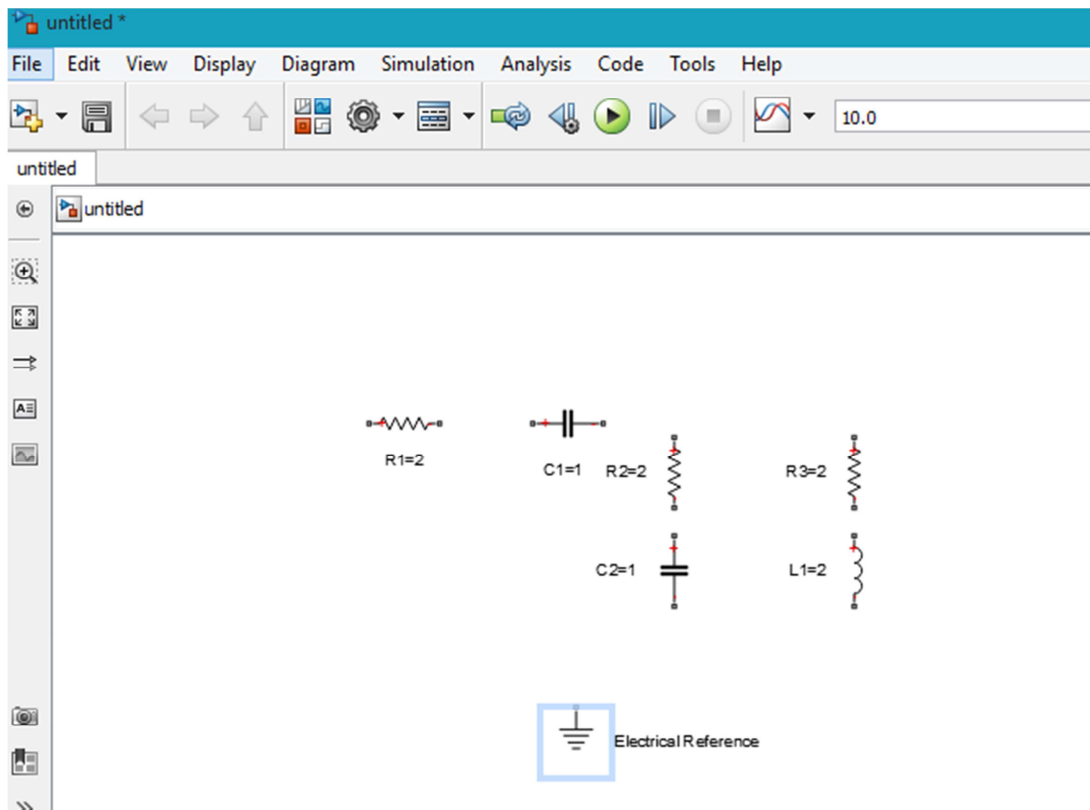


Figure 2.6: Selected Element, step 5

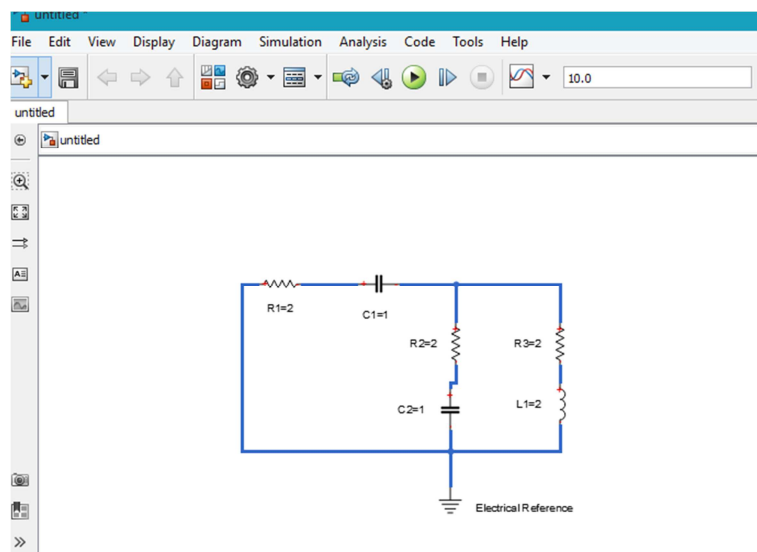


Figure 2.7: Connect the element, step6

7. To add the representation of the force acting on the mass, open the Simscape > Foundation Library > Electrical > Electrical Sources library Figure 2.8, and add the Controlled Voltage Source block to your diagram as Figure 2.9.

Experiment 2: Mathematical Modelling Using SimScope

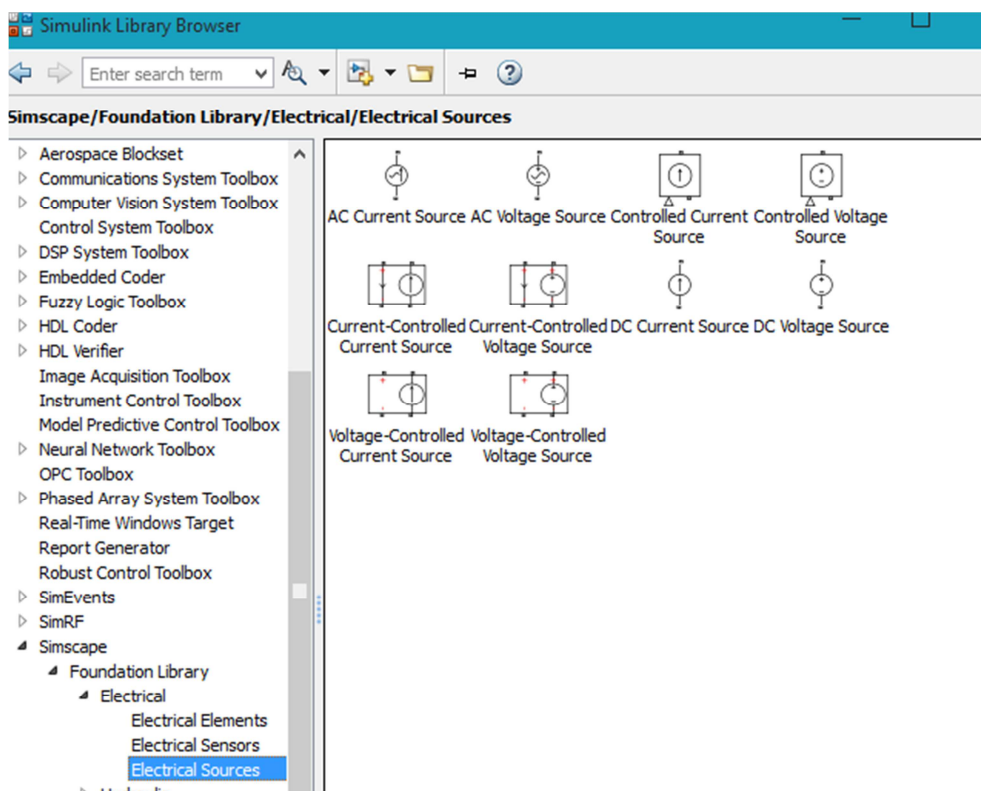


Figure 2.8: Electrical Source, step7

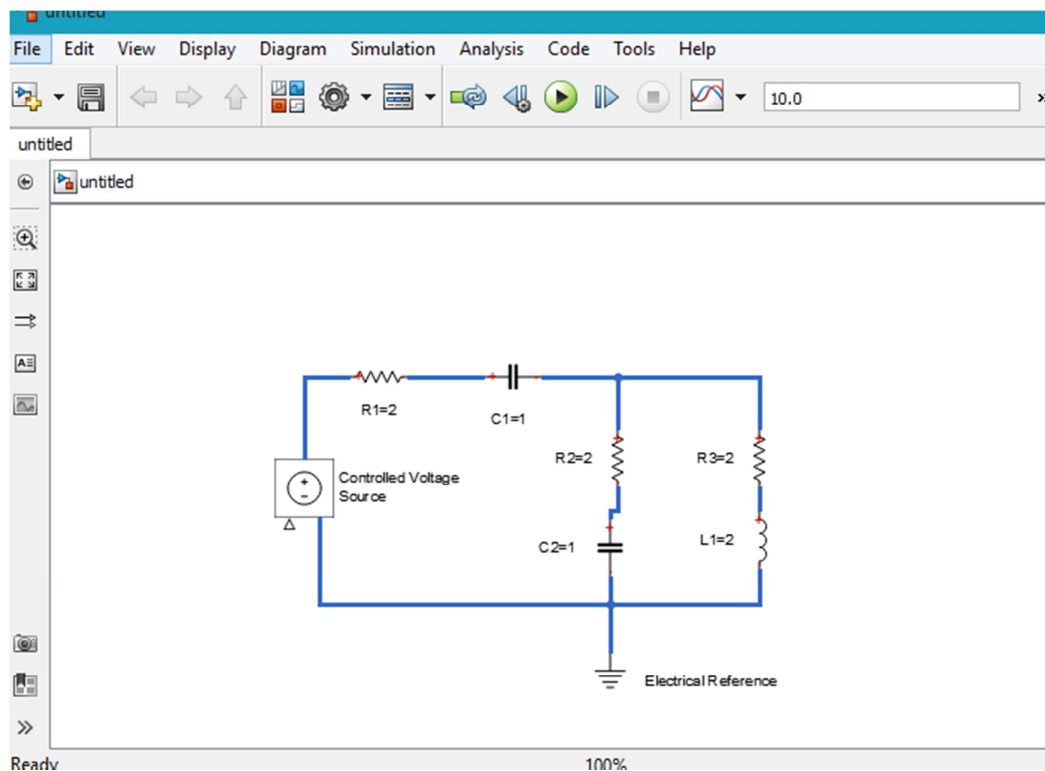


Figure 2.9: The block of circuit after add the voltage source, step7

Experiment 2: Mathematical Modelling Using SimScope

8. Add the sensor to measure speed and position of the mass. Place the Voltage and current Sensors block from the Electrical Sensors library Figure 2.10 into your diagram and connect it as shown in Figure 2.11.

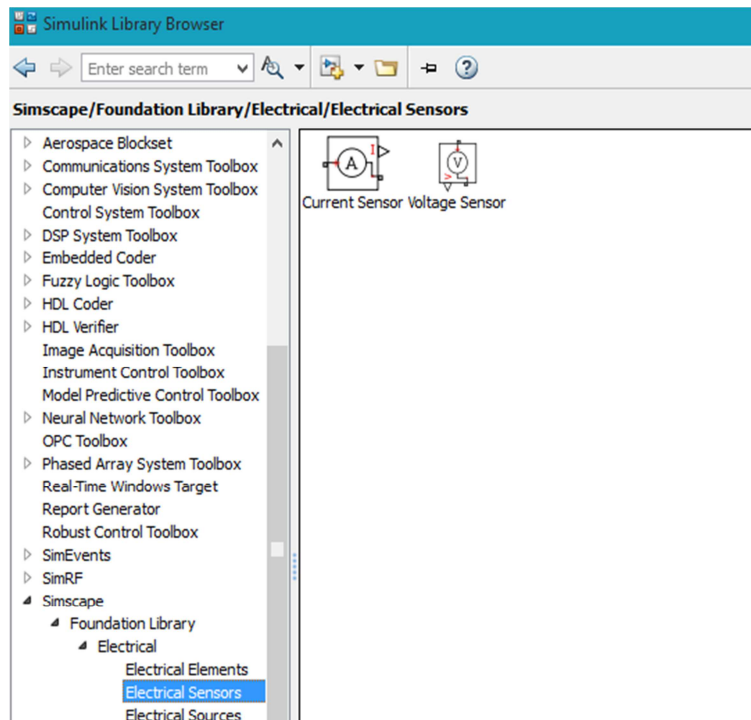


Figure 2.10: Electrical Sensor

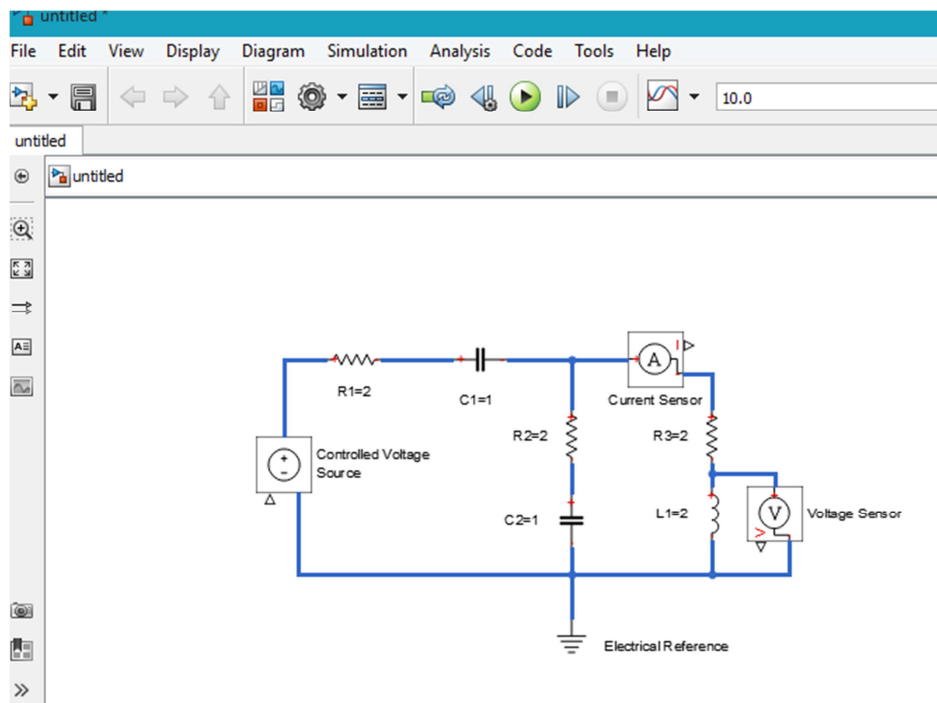


Figure 2.11: Electrical circuit using SimScope

9. Now you need to add the sources and scopes. They are found in the regular Simulink libraries. Open the Simulink > Sources library and copy the Signal Builder block into the

Experiment 2: Mathematical Modelling Using SimScope

model. Then open the Simulink > Sinks library and copy two Scope blocks..

- Every time you connect a Simulink source or scope to a Simscape diagram, you have to use an appropriate converter block, to convert Simulink signals into physical signals and vice versa. Open the Simscape > Utilities library Figure 2.12 and copy a Simulink-PS Converter block and two PS-Simulink Converter blocks into the model. Connect the blocks as shown in Figure 2.13.
- Each topologically distinct physical network in a diagram requires exactly one Solver Configuration block, found in the Simscape > Utilities library Figure 2.12. Copy this block into your model and connect it to the circuit by creating a branching point and connecting it to the only port of the Solver Configuration block. Your diagram now should look like Figure 2.13.

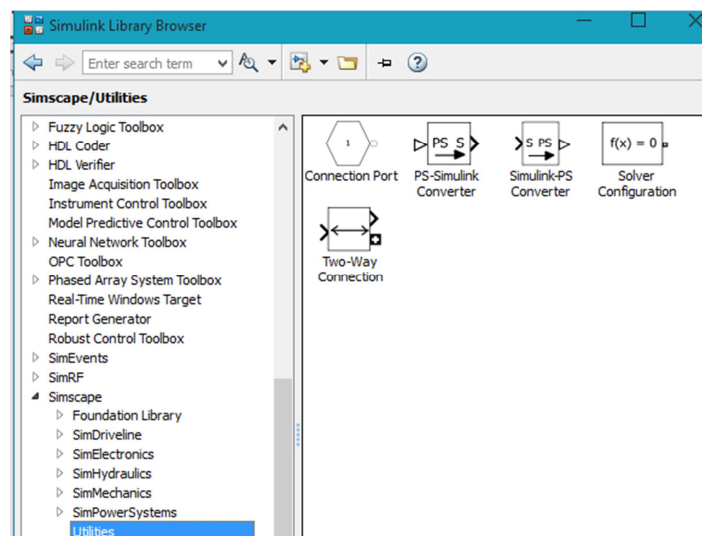


Figure 2.12: Simscape Utilities Library

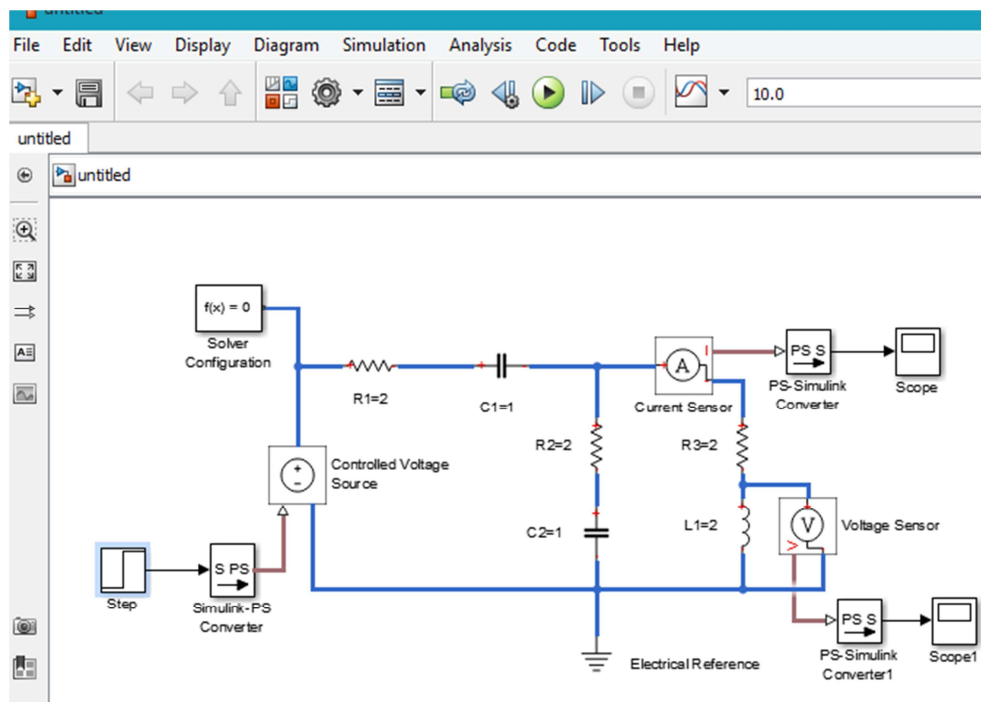


Figure 2.13: Full Simscape Representation for Electrical Circuit

The Simulation Result

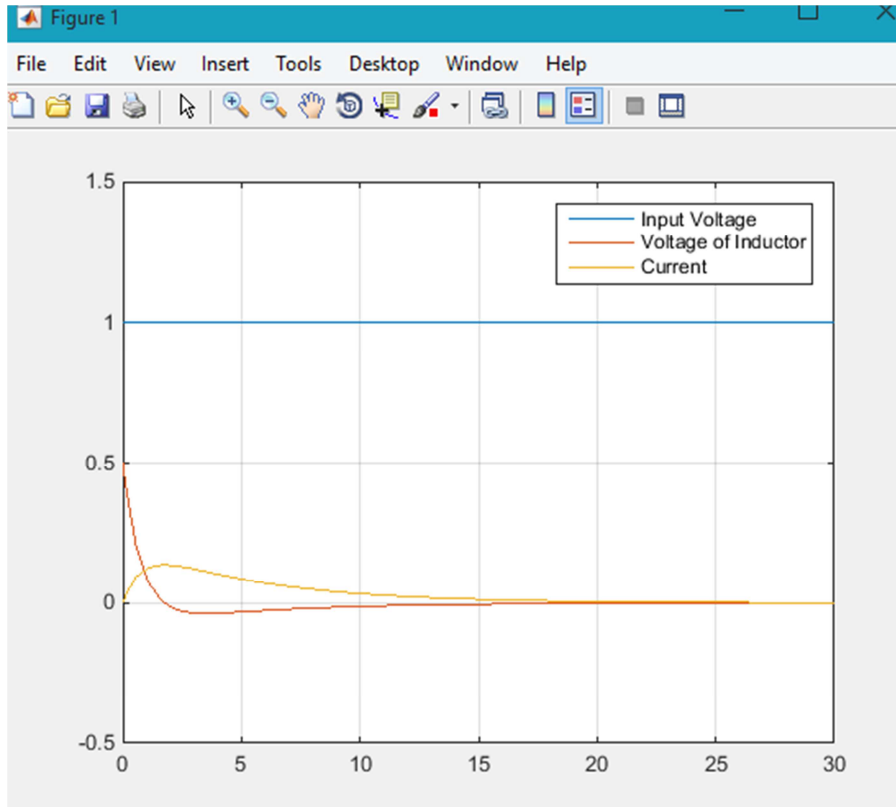
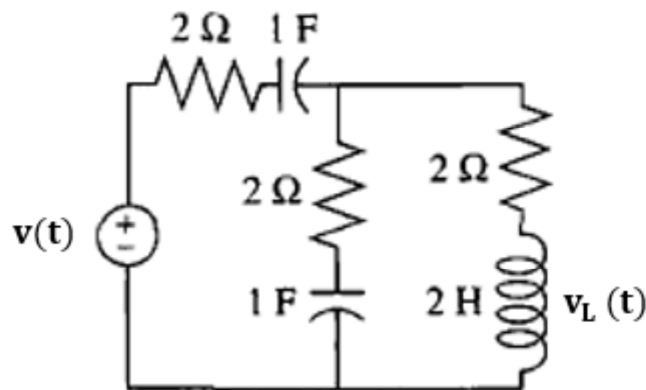


Figure 2.14: The Simulation Result

Exercises:

For the following electrical system answer the questions.



1. Write the differential equations of the system.
2. Find the transfer function, $G(s) = V_L(s)/V_i(s)$.

Experiment 2: Mathematical Modelling Using SimScape

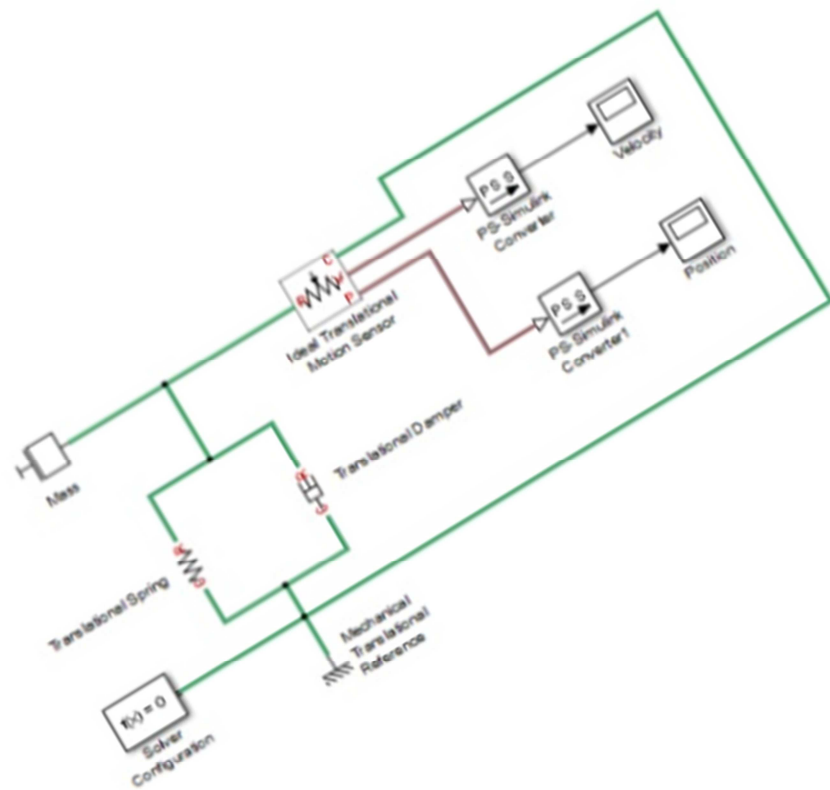
3. Build the system using simscape.
4. For a step input $v_i(t)=5u(t), t>0$ plot the open loop response of:
 - a) The output voltage on the inductor $v_L(t)$
 - b) The current $I_L(t)$.
5. Study the sensitivity of the output voltage due to change the following :
 - a) The value of capacitor c_1 .
 - b) The value of capacitor c_2
 - c) The value of resistor R_1 .
 - d) The value of resistor R_2
 - e) The value of Inductor L_1 .

Experiment
Three

Mathematical Modelling Using SimScape (Mechanical Systems)

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



1. Translational Mechanical System

In this example, you are going to model a simple mechanical system and observe its behavior under various conditions. This tutorial illustrates the essential steps to building a physical model and makes you familiar with using the basic Simscape blocks.

The figure 3.1 shows a simple model of a car suspension. It consists of a spring and damper connected to a body (represented as a mass), which is agitated by a force. You can vary the model parameters, such as the stiffness of the spring, the mass of the body, or the force profile, and view the resulting changes to the velocity and position of the body.

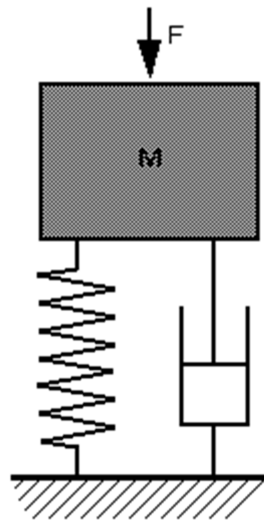


Figure 3.1: A model of Car Suspension

To create an equivalent Simscape diagram, follow these steps:

1. Open the Simulink® Library Browser, as described in Simscape Block Libraries.
2. Create a new model. To do this, from the top menu bar of the Library Browser, select **File > New > Model**. The software creates an empty model in memory and displays it in a new model editor window.
3. Open the **Simscape > Foundation Library > Mechanical > Translational Elements** library as shown in figure 3.2.
4. Drag the Mass, Translational Spring, Translational Damper, and two Mechanical Translational Reference blocks into the model window.
5. Orient the blocks as shown in the figure 3.3. To rotate a block, select it and press Ctrl+R.
6. Connect the Translational Spring, Translational Damper, and Mass blocks to one of the Mechanical Translational Reference blocks as shown in figure 3.4.

Experiment 3: Mathematical Modelling Using SimScope

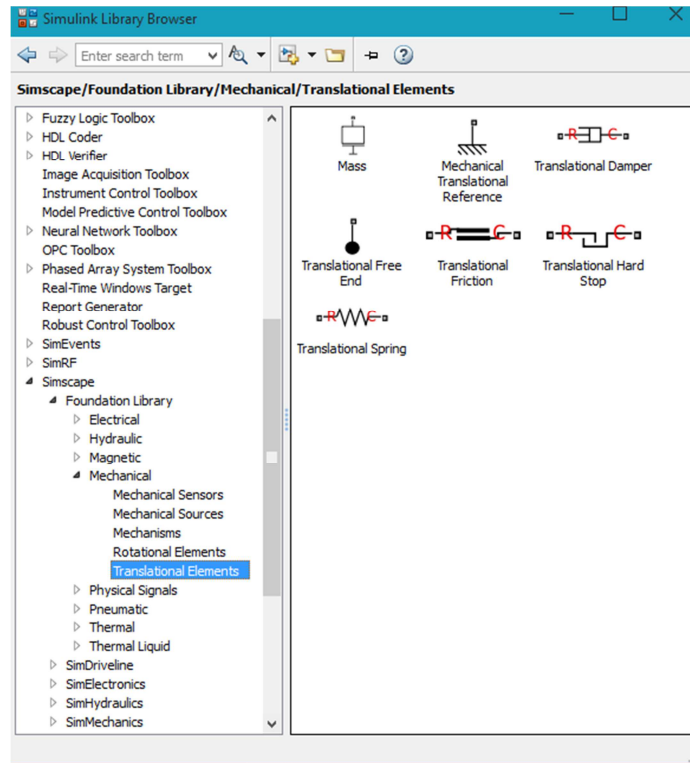


Figure 3.2: Mechanical Translational Elements, step 3

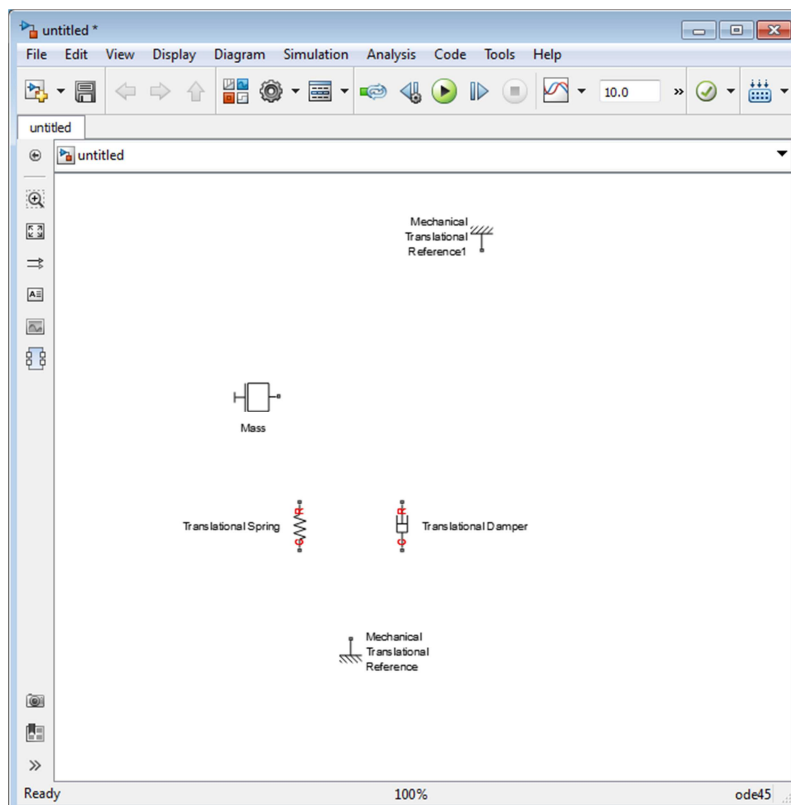


Figure 3.3: Selected Elements, step 5

Experiment 3: Mathematical Modelling Using SimScape

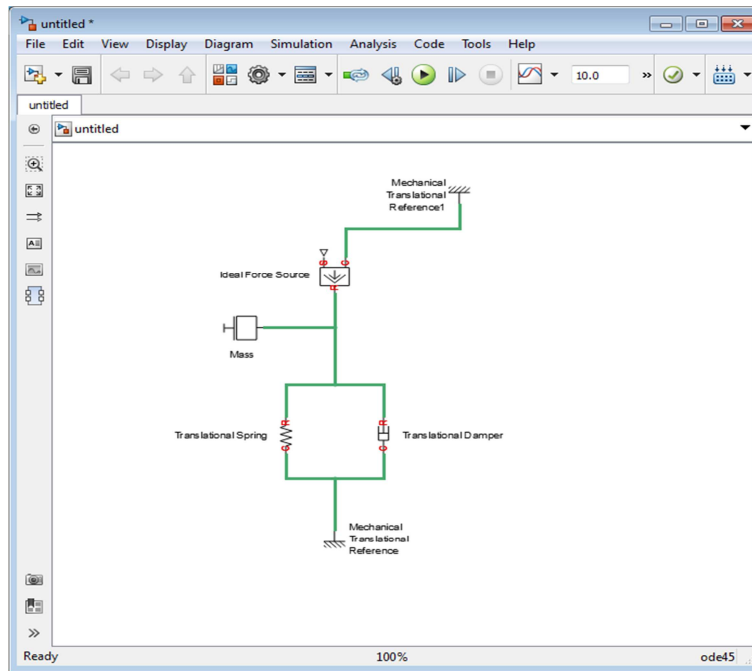


Figure 3.4: Connect the elements to each other, step 6

7. To add the representation of the force acting on the mass, open the **Simscape > Foundation Library > Mechanical > Mechanical Sources** library figure 3.5, and add the Ideal Force Source block to your diagram as figure 3.4.

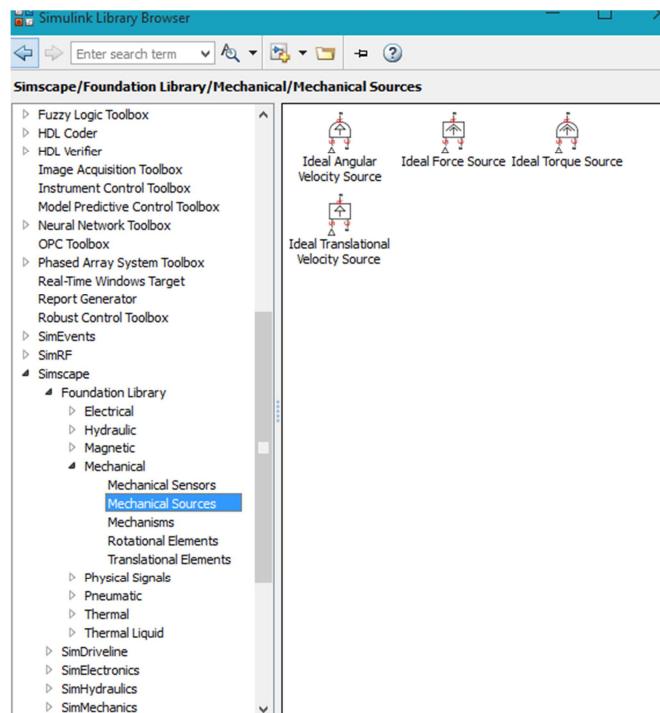


Figure 3.5: Mechanical Force, step 7

Experiment 3: Mathematical Modelling Using SimScape

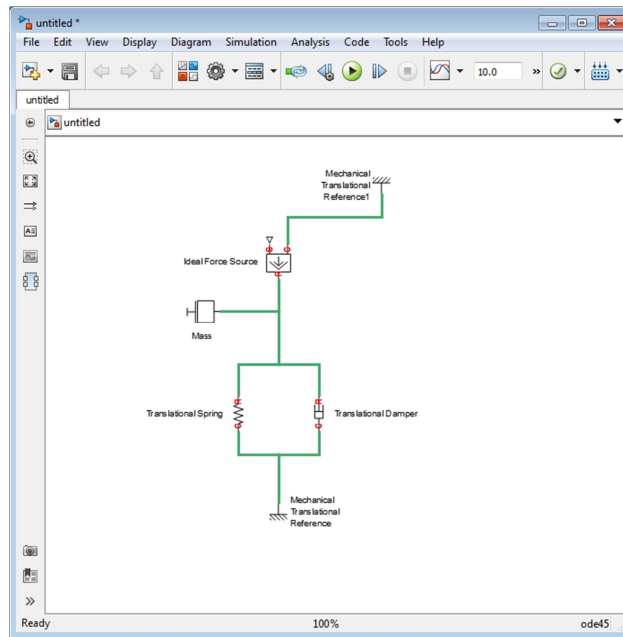


Figure 3.6: The block diagram after add the Force, step 7

8. Add the sensor to measure speed and position of the mass. Place the Ideal Translational Motion Sensor block from the **Mechanical Sensors library** figure 3.7 into your diagram and connect it as shown in figure 3.8.

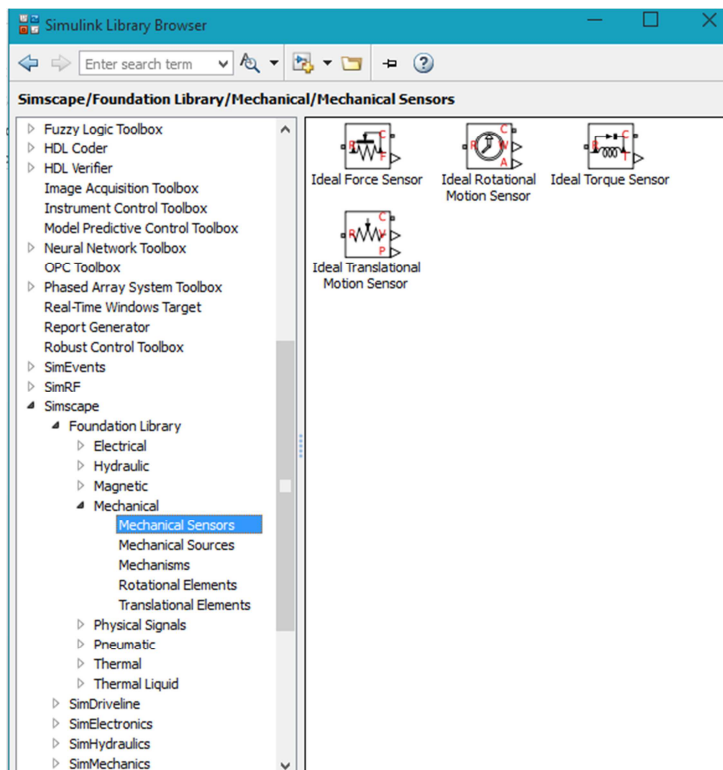


Figure 3.7: Mechanical Sensors Library

Experiment 3: Mathematical Modelling Using SimScape

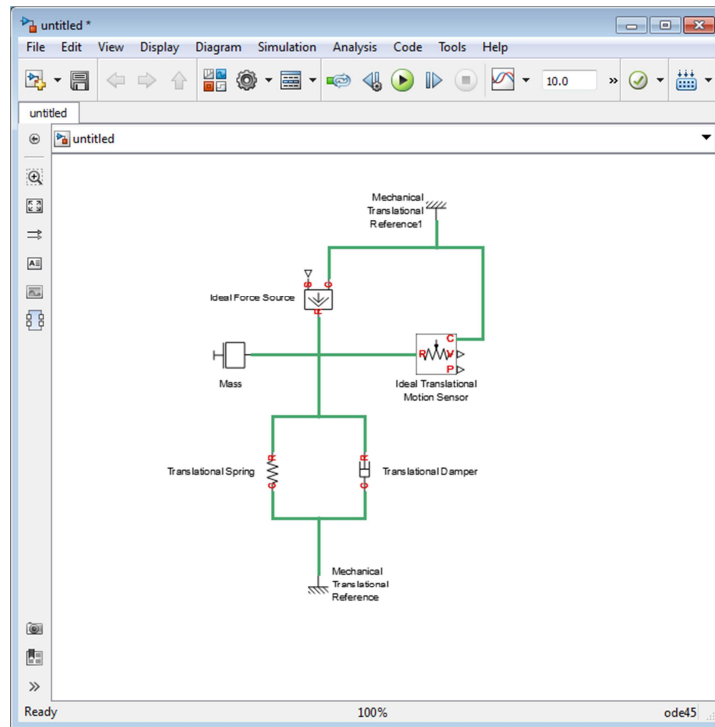


Figure 3.8: Block Diagram

9. Now you need to add the sources and scopes. They are found in the regular Simulink libraries. Open the **Simulink > Sources library** and copy the Signal Builder block into the model. Then open the **Simulink > Sinks library** and copy two Scope blocks. Rename one of the Scope blocks to *Velocity* and the other to *Position*.
10. Every time you connect a Simulink source or scope to a Simscape diagram, you have to use an appropriate converter block, to convert Simulink signals into physical signals and vice versa. Open the **Simscape > Utilities library** figure 3.9 and copy a Simulink-PS Converter block and two PS-Simulink Converter blocks into the model. Connect the blocks as shown in figure 3.9.
11. Each topologically distinct physical network in a diagram requires exactly one Solver Configuration block, found in **the Simscape > Utilities library** figure 3.9. Copy this block into your model and connect it to the circuit by creating a branching point and connecting it to the only port of the Solver Configuration block. Your diagram now should look like figure 3.10.

Experiment 3: Mathematical Modelling Using SimScope

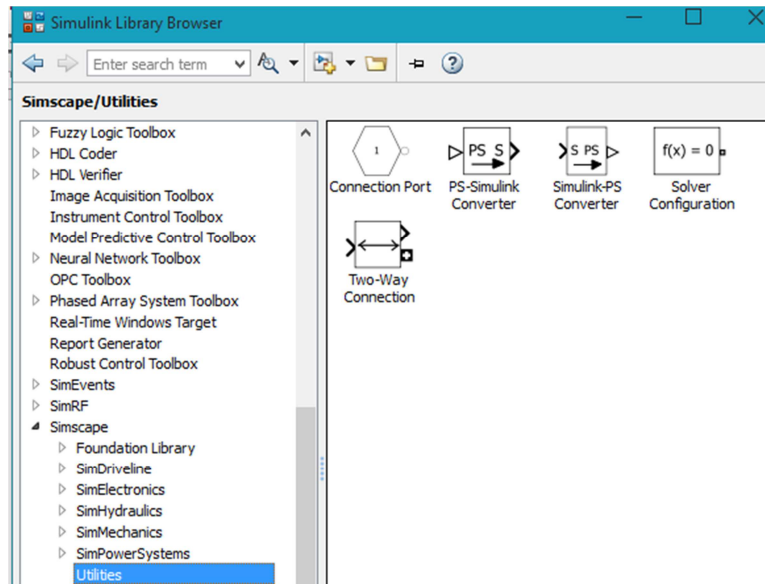


Figure 3.9: Simscape Utilities Library

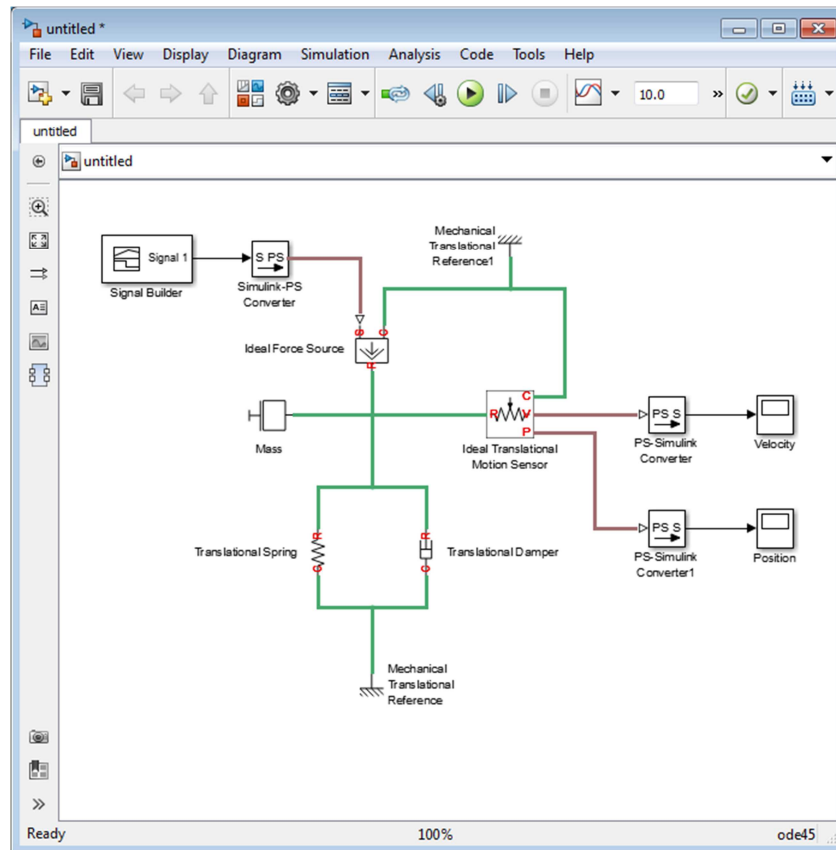


Figure 3.1: Block Diagram of System

- **Modifying Initial Settings**

After you have put together a block diagram of your model, as described in the previous section, you need to select a solver and provide the correct values for configuration parameters.

To prepare for simulating the model, follow these steps:

1. Select a Simulink solver. On the top menu bar of the model window, select Simulation > Model Configuration Parameters. The Configuration Parameters dialog box opens, showing the Solver node.

Under Solver options, set Solver to *ode23t (mod.stiff/Trapezoidal)* and Max step size to 0.2.

Also note that Simulation time is specified to be between 0 and 10 seconds. You can adjust this setting later, if needed.

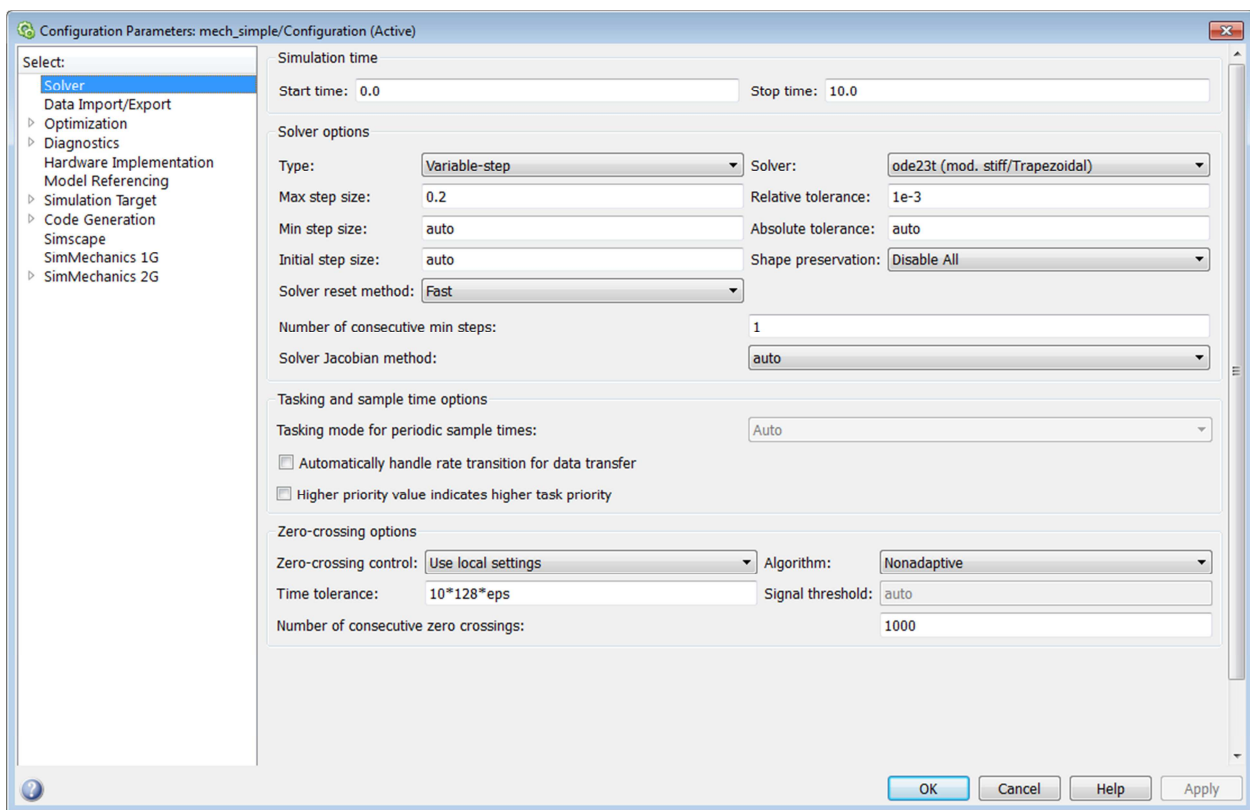


Figure 3.2: Solver Configuration

2. Rotational Mechanical System

In this example, you are going to model a mechanical system and observe its behavior under various conditions. This tutorial illustrates the essential steps to building a physical model and makes you familiar with using the basic Simscape blocks.

The figure 3.12 shows a rotational mechanical system. It consists of a spring and damper connected to a body (represented as an inertia), which is agitated by a torque. You can vary the model parameters, such as the stiffness of the spring, the inertia of the body, or the torque profile, and view the resulting changes to the angular velocity and angular position of the body.

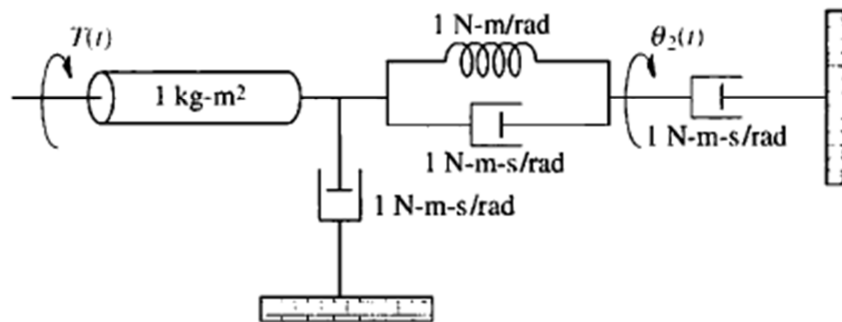


Figure 3.12 A Rotational Mechanical System

- **Modeling of System**

To create an equivalent Simscape diagram, follow these steps:

12. Open the Simulink® Library Browser, as described in Simscape Block Libraries.
13. Create a new model. To do this, from the top menu bar of the Library Browser, select **File > New > Model**. The software creates an empty model in memory and displays it in a new model editor window.
14. Open the **Simscape > Foundation Library > Mechanical > Rotational Elements** library as shown in figure 3.13.
15. Drag the Inertia, Rotational Spring, Rotational Damper, and Mechanical Rotational Reference blocks into the model window.
16. Orient the blocks as shown in the figure 3.14. To rotate a block, select it and press Ctrl+R.
17. Connect the Translational Spring, Translational Damper, and Mass blocks to one of the Mechanical Translational Reference blocks as shown in figure 3.15

Experiment 3: Mathematical Modelling Using SimScape

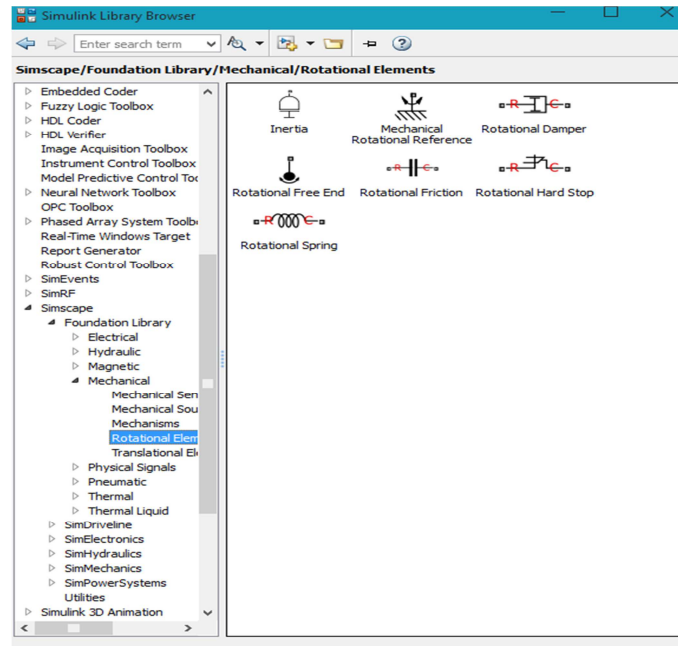


Figure 3.3: Mechanical Rotational Elements, step 3

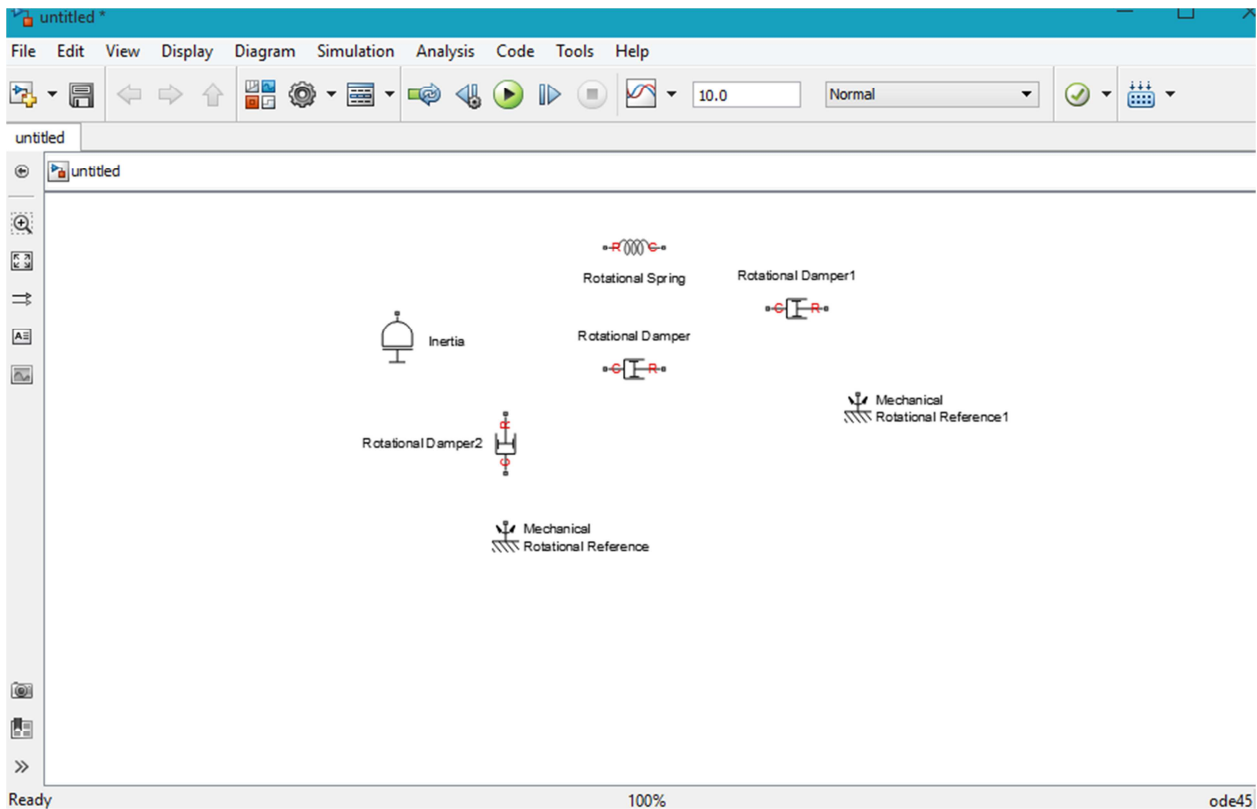


Figure 3.4: Selected Elements, step 5

Experiment 3: Mathematical Modelling Using SimScape

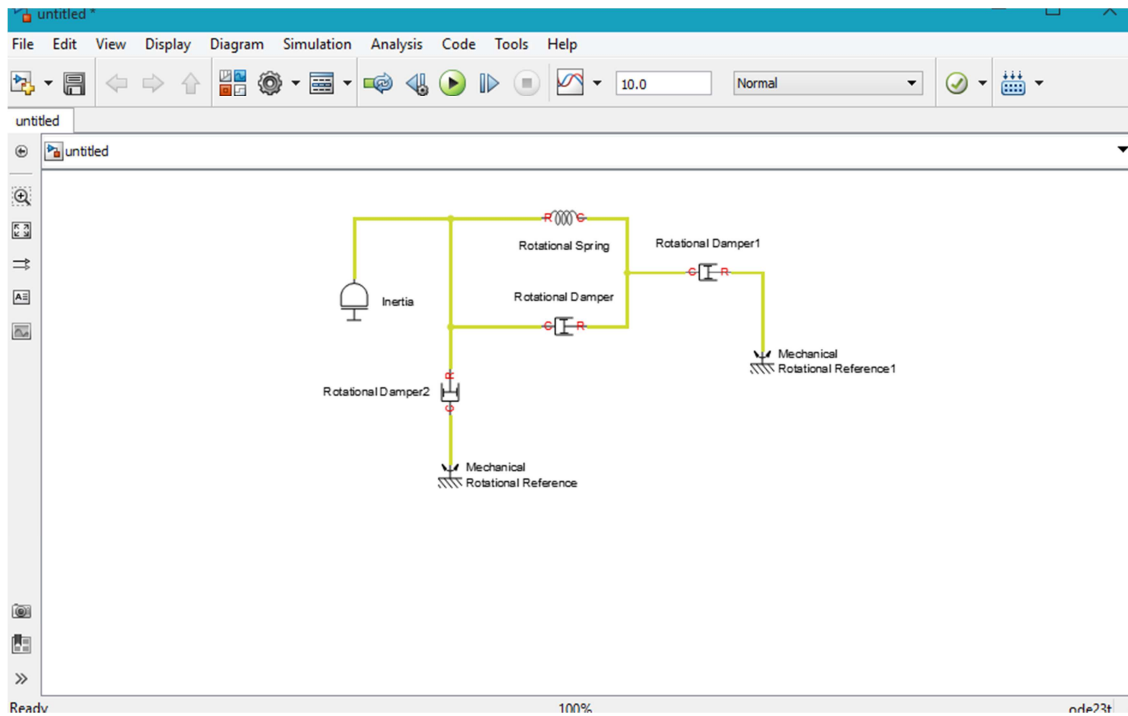


Figure 3.5: Connect the elements to each other, step 6

18. To add the representation of the force acting on the mass, open the **Simscape > Foundation Library > Mechanical > Mechanical Sources** library figure 3.16, and add the Ideal Torque Source block to your diagram as figure 3.17.

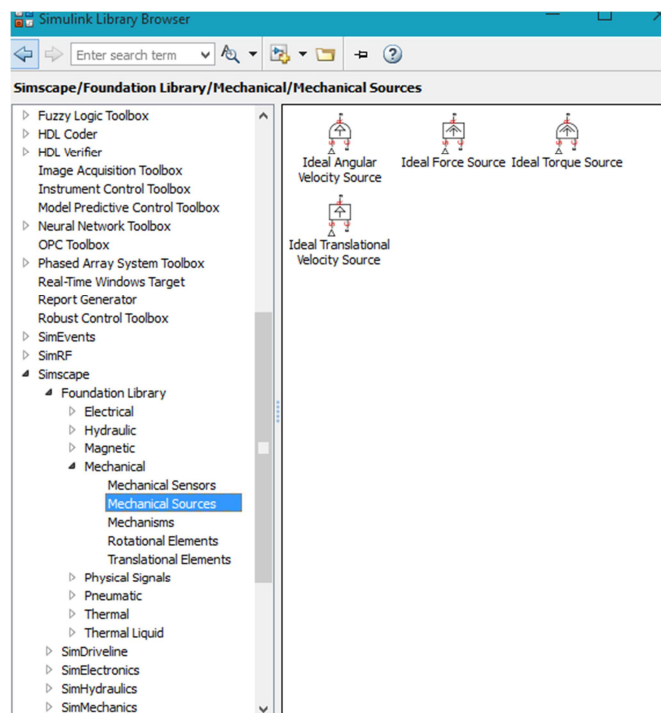


Figure 3.6: Mechanical Force, step 7

Experiment 3: Mathematical Modelling Using SimScape

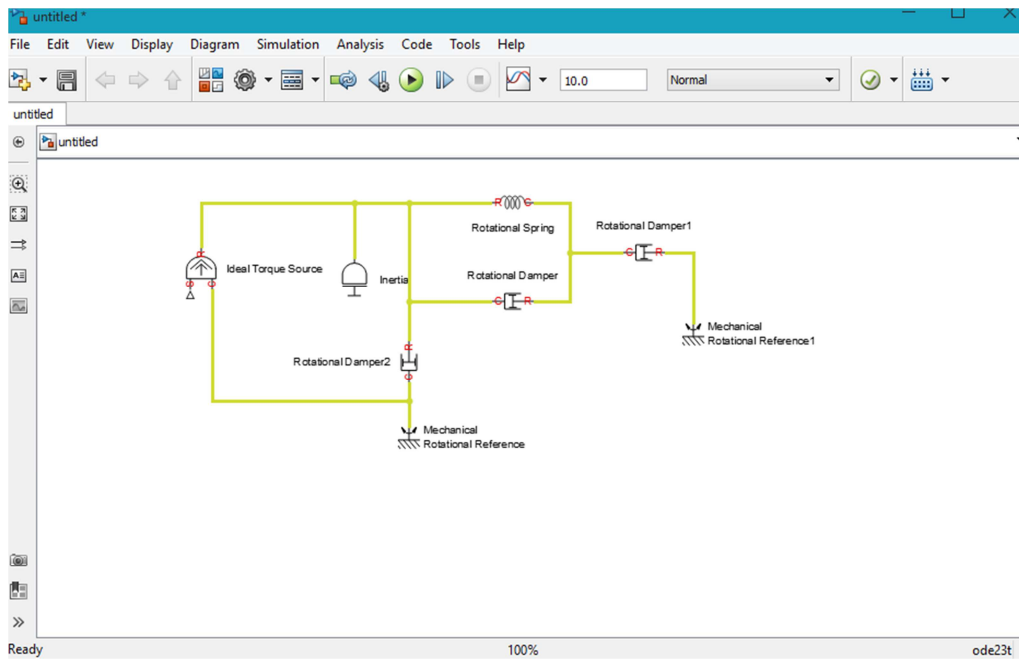


Figure 3.17: The block diagram after add the Force, step 7

19. Add the sensor to measure angular speed and angular position at $\theta_2(t)$. Place the Ideal Rotational Motion Sensor block from the **Mechanical Sensors** library figure 3.18 into your diagram and connect it as shown in figure 3.19.

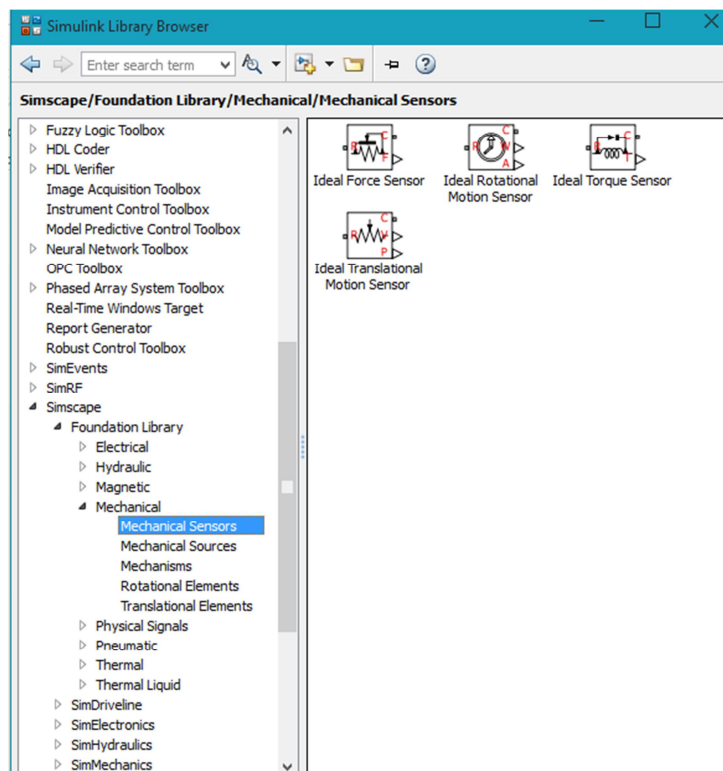


Figure 3.18: Mechanical Sensors Library

Experiment 3: Mathematical Modelling Using SimScape

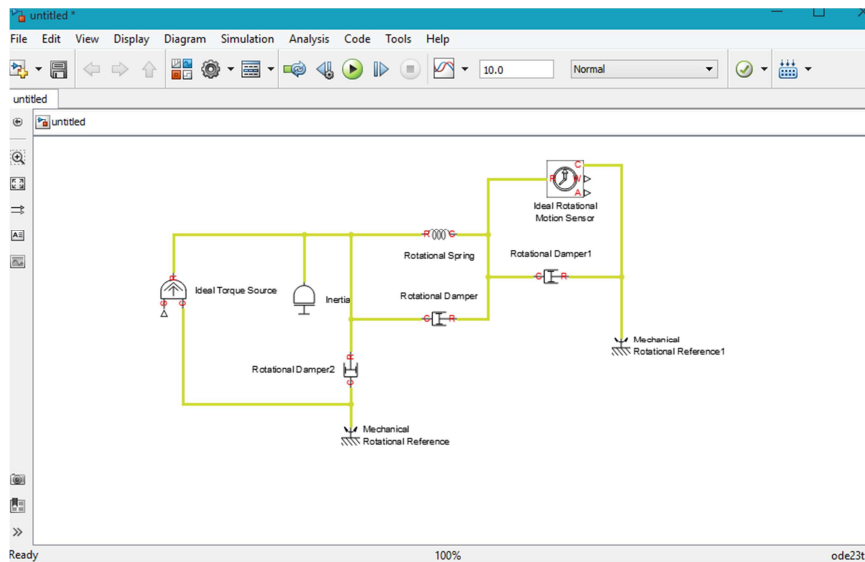


Figure 3.19: Block Diagram

20. Now you need to add the sources and scopes. They are found in the regular Simulink libraries. Open the **Simulink > Sources library** and copy the Signal Builder block into the model. Then open the **Simulink > Sinks library** and copy two Scope blocks. Rename one of the Scope blocks to *Velocity* and the other to *Position*.
21. Every time you connect a Simulink source or scope to a Simscape diagram, you have to use an appropriate converter block, to convert Simulink signals into physical signals and vice versa. Open the **Simscape > Utilities library** figure 3.20 and copy a Simulink-PS Converter block and two PS-Simulink Converter blocks into the model. Connect the blocks as shown in figure 3.20.
22. Each topologically distinct physical network in a diagram requires exactly one Solver Configuration block, found in **the Simscape > Utilities library** figure 3.21. Copy this block into your model and connect it to the circuit by creating a branching point and connecting it to the only port of the Solver Configuration block. Your diagram now should look like figure 3.21.

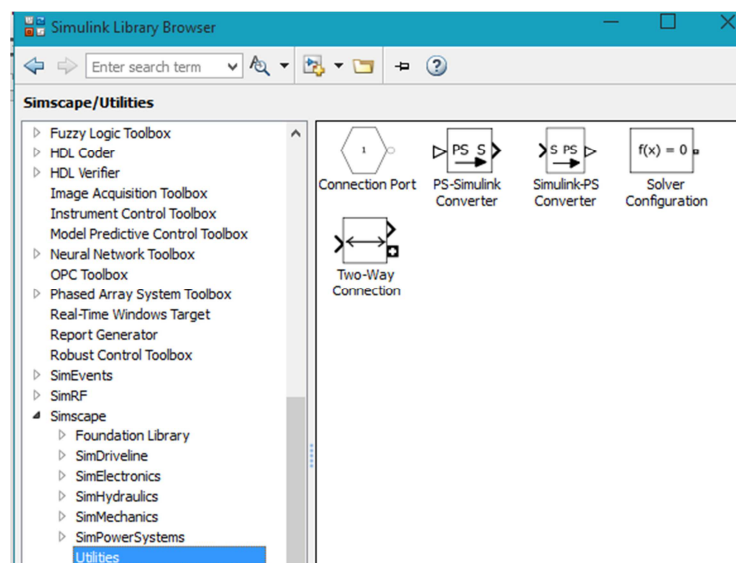


Figure 3.20: Simscape Utilities Library

Experiment 3: Mathematical Modelling Using SimScape

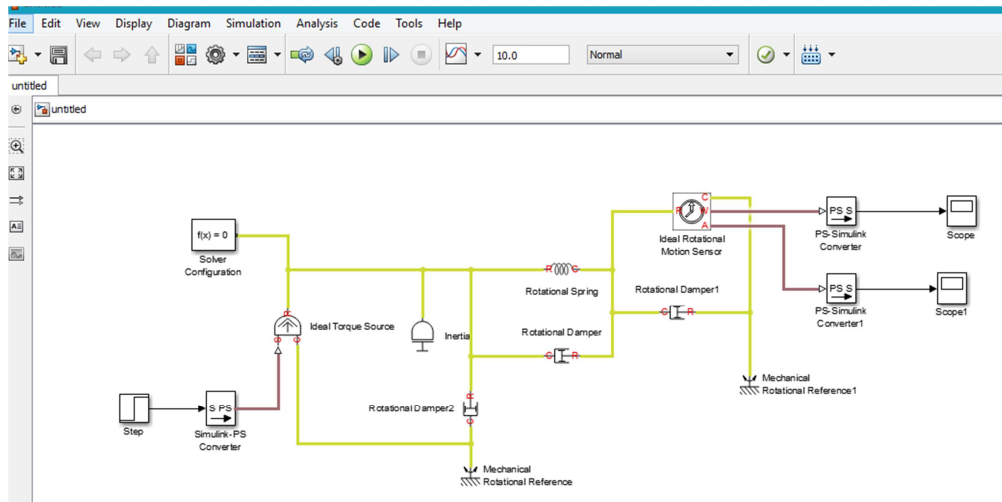
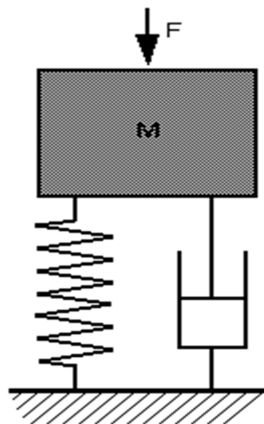


Figure 3.21: Block Diagram of System

Exercises:



For M is the mass, b is the damper, k is spring, $x(t)$ is the position of the cart and $F(t)$ is the force applied to the system.

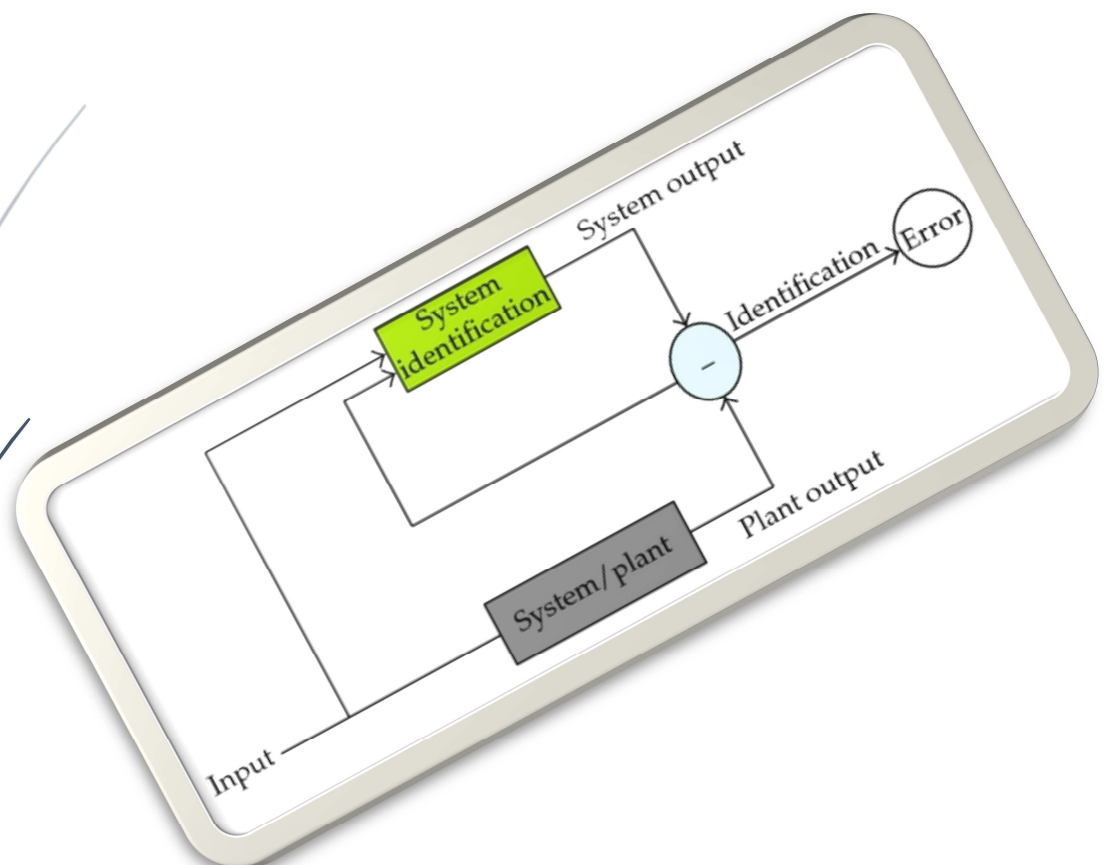
1. Write the differential equations of the system.
2. Let $M=1\text{Kg}$, $b=1\text{Ns/m}$ and $k=1$. Find the transfer function, $G(S)=X(S)/F(S)$.
3. Build the system using simscape.
4. For a step input $f(t)=5u(t), t>0$ plot the open loop response of the position $x(t)$.
5. Find the %OS, s.s output, T_s , T_r and T_p from the previous part.
6. How does the change in the spring stiffness affect the mass displacement?
7. How does the change in the damper viscosity affect the mass displacement?

Experiment
Four

System Identification (First order System)

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



1. Introduction:

First order systems are, by definition, systems whose input-output relationship is a first order differential equation. A first order differential equation contains a first order derivative but no derivative higher than first order – the order of a differential equation is the order of the highest order derivative present in the equation.

First order systems contain a single energy storage element. In general, the order of the input-output differential equation will be the same as the number of independent energy storage elements in the system. Independent energy storage cannot be combined with other energy storage elements to form a single equivalent energy storage element. For example, we previously learned that two capacitors in parallel can be modeled as a single equivalent capacitor – therefore, a parallel combination of two capacitors forms a single independent energy storage element.

First order systems are an extremely important class of systems. Many practical systems are first order; for example, the mass-damper system and the mass heating system are both first order systems. Higher order systems can often be approximated as first order systems to a reasonable degree of accuracy if they have a dominant first order mode.

2. First Order System Model

The first order system has only one pole as shown

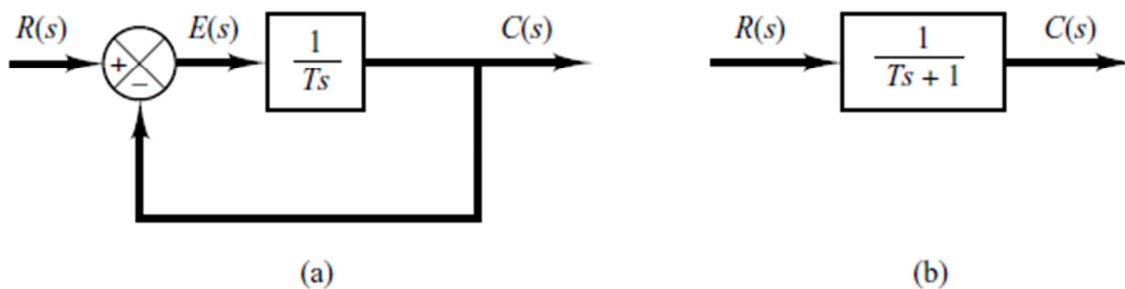


Figure 1: (a) Block Diagram of a first-order system; (b) Simplified block Diagram

$$\frac{C(s)}{R(s)} = K \frac{1}{Ts + 1} \quad (1)$$

- Where K is the DC Gain and T is the time constant of the system.
- Time Constant is a measure of how quickly a 1st order system response to a unit step input.
- DC gain of the system ration between the input signal and the steady state value of output.

Example 1:

For the first order system given below

$$G(s) = \frac{10}{3s + 1}$$

- DC gain (K) is equal 10
- Time Constant (T) is equal 3

Example 2:

Experiment 4: System Identification

For the first order system given below

$$G(s) = \frac{3}{s + 5} = \frac{\frac{3}{5}}{\frac{1}{5}s + 1}$$

- DC gain (K) is equal $\frac{3}{5}$
- Time Constant (T) is equal $\frac{1}{5}$

2.1 Impulse Response of 1st Order System

Consider the following 1st order system in figure 2

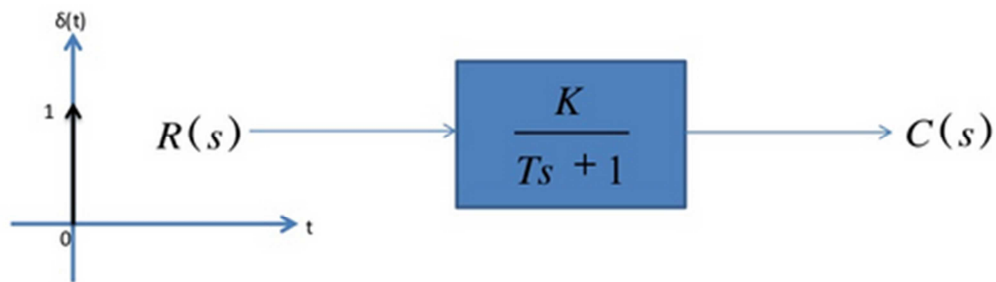


Figure 2: Impulse response of first order system

$$R(s) = \delta(s) = 1$$

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

In order to represent the response of the system in time domain we need to compute the inverse Laplace transform of the above equation, we have

$$c(t) = \frac{K}{T} e^{-\frac{t}{T}} \quad (2)$$

Example 3:

For the first order system given below

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

The impulse response is shown in figure 3

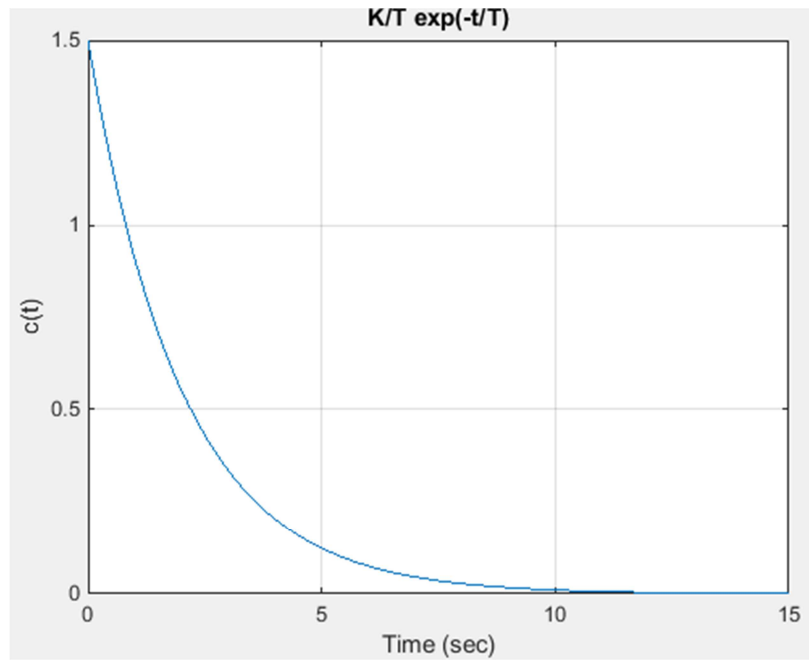


Figure 3: Impulse response of example 3

2.2 Ramp Response of 1st Order System

Consider the first order system in figure 1.

$$R(s) = \frac{1}{s^2}$$
$$C(s) = \frac{K}{Ts + 1} \frac{1}{s^2}$$

In order to represent the response of the system in time domain we need to compute the inverse Laplace transform of the above equation, we have

$$c(t) = Kt - T + Te^{-\frac{t}{T}} \quad (3)$$

Example 4:

For the first order system given below

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

- 1) The ramp response of system, if $K = 1$ and $T = 1$ is shown in figure 4
- 2) The ramp response of system, if $K = 1$ and $T = 3$ is shown in figure 5

Experiment 4: System Identification

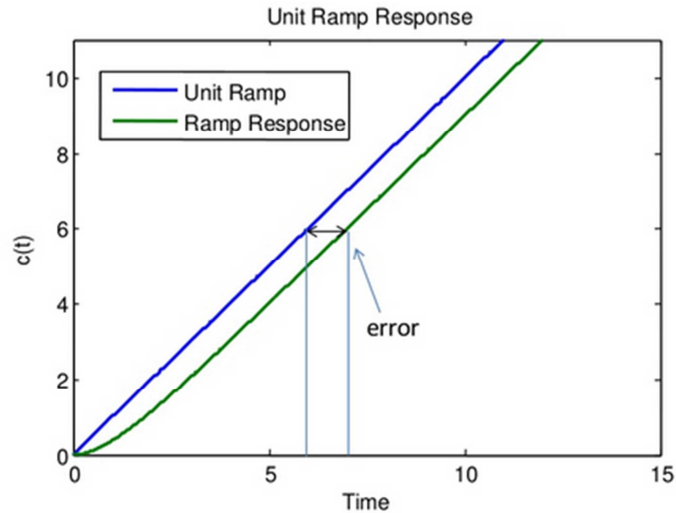


Figure 4: The ramp response for case 1, example 4

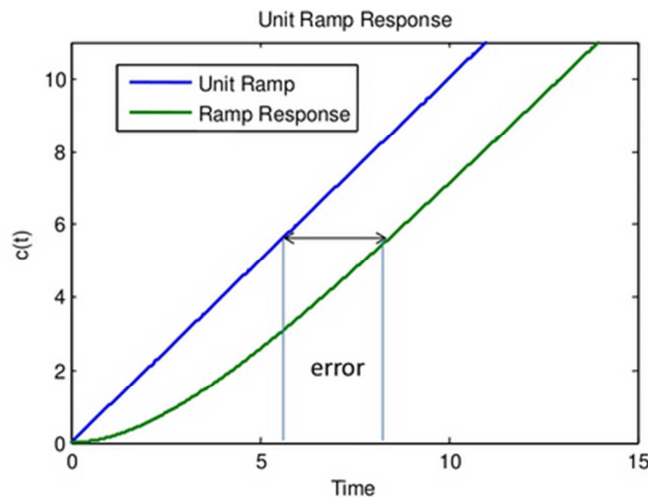


Figure 5: The ramp response for case 2, example 4

2.3 Step Response of 1st Order System

Consider the first order system in figure 1.

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

In order to represent the response of the system in time domain we need to compute the inverse Laplace transform of the above equation, we have

$$c(t) = Ku(t) - e^{-\frac{t}{T}} \quad (4)$$

1) Where $u(t) = 1$

Experiment 4: System Identification

$$c(t) = K - e^{-\frac{t}{T}} \quad (5)$$

2) Where $t = T$

$$c(t) = K - e^{-1} = 0.632K \quad (6)$$

- For example, assume $K = 10, T = 1.5s$

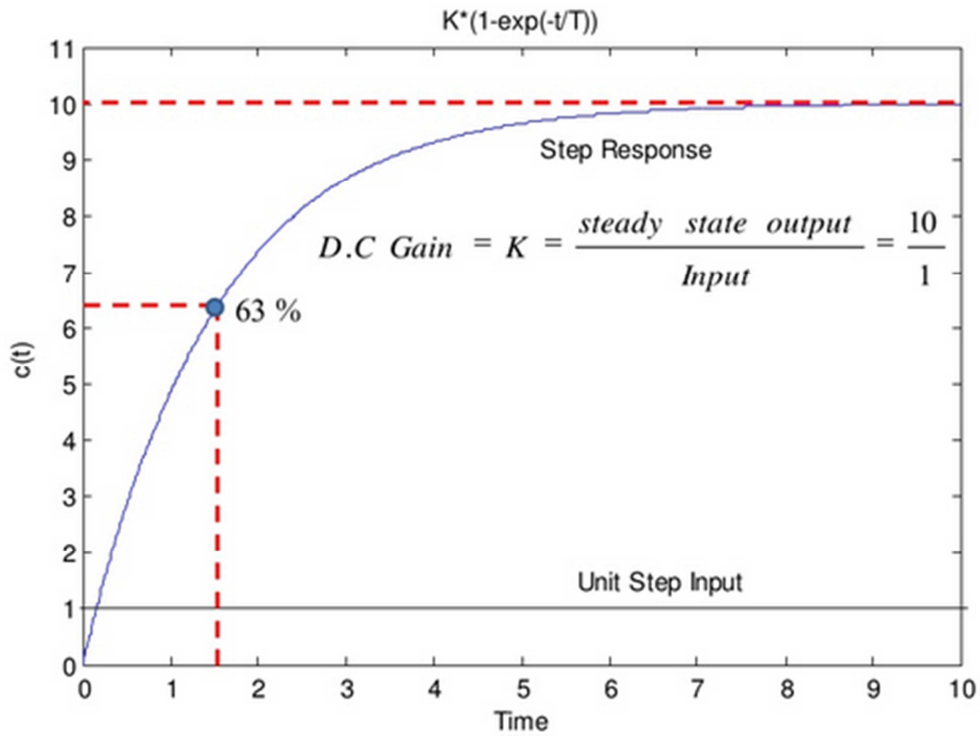


Figure 6: The step response specification of first order system

The step response of the first order system takes five time constants to reach its final value.

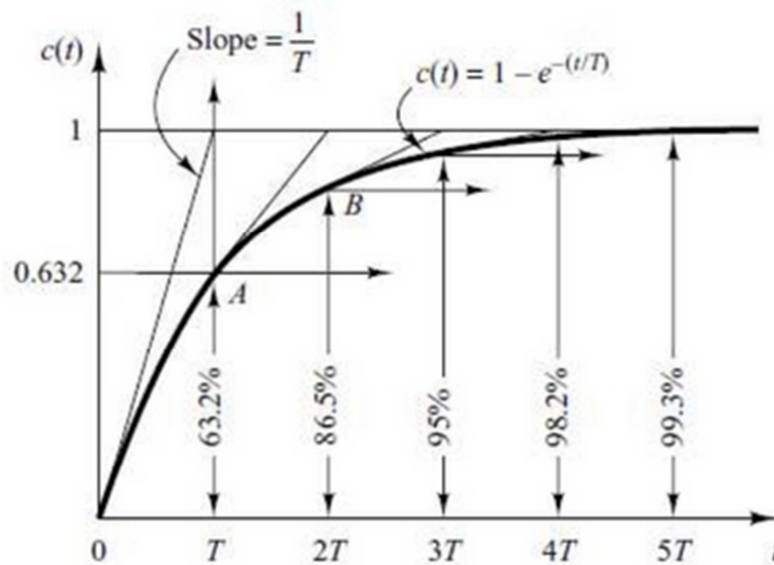


Figure 7: Step response

Experiment 4: System Identification

- $K = 10, T = 1, 3, 5, 7 \text{ s}$

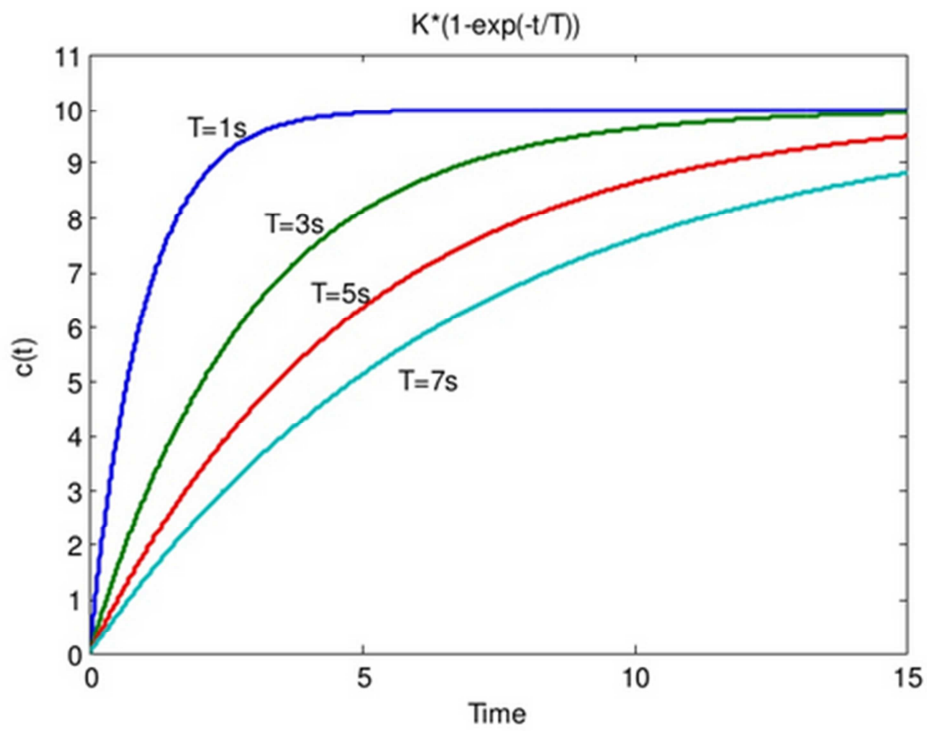


Figure 8: The step response at different value of T

- $K = 1, 3, 5, 10, T = 1 \text{ s}$

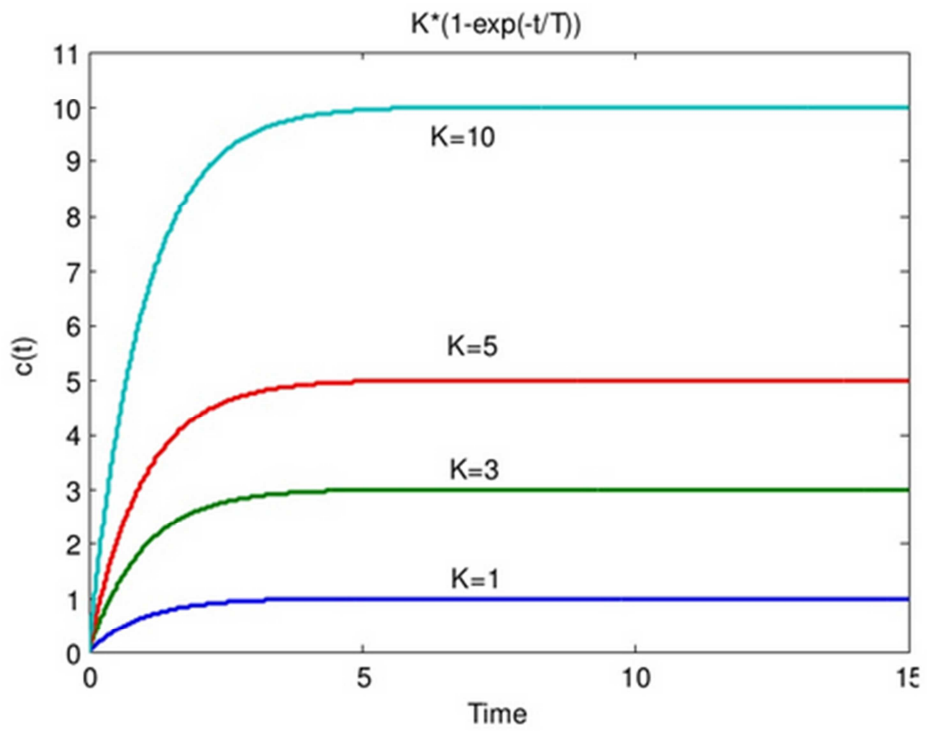


Figure 9: The step response at different value of K

3. First Order System with a Zero

The transfer function of the first order system with a zero can be represented using the form in equation (7)

$$G(s) = \frac{C(s)}{R(s)} = \frac{K(1 + \alpha s)}{Ts + 1} \quad (7)$$

- Zero of the system lie at $s = -1/\alpha$ and pole at $s = -1/T$
- Step response of the system would be:

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

The Laplace inverse transfer is

$$c(t) = K + \frac{K}{T}(\alpha - T) e^{-\frac{t}{T}} \quad (8)$$

- **Case 1: $T > \alpha$**

The shape of the step response is approximately same (with offset added by zero)

Example 5:

Consider the first order system given by

$$\frac{C(s)}{R(s)} = \frac{10(1 + 2s)}{3s + 1}$$

In this system:

1. $K = 10$
2. $T = 3$
3. $\alpha = 2$

$$c(t) = 10 + \frac{10}{3}(2 - 3)e^{-t/3}$$

Experiment 4: System Identification

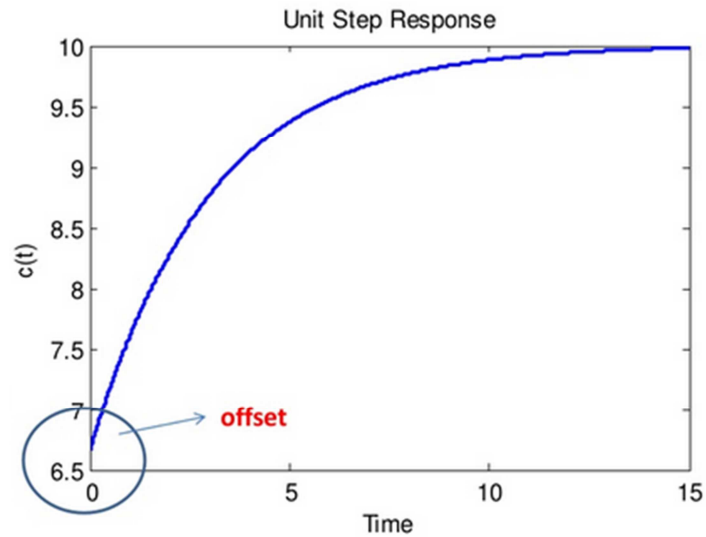


Figure 10: the step response of case 1 $T > \alpha$

$$\text{offset} = K + \frac{K}{T}(\alpha - T)$$

- **Case 2: $T < \alpha$**

Example 6:

Consider the first order system given by

$$\frac{C(s)}{R(s)} = \frac{10(1 + 2s)}{1.5s + 1}$$

In this system:

1. $K = 10$
2. $T = 1.5$
3. $\alpha = 2$

$$c(t) = 10 + \frac{10}{1.5}(2 - 1)e^{-t/1.5}$$

Experiment 4: System Identification

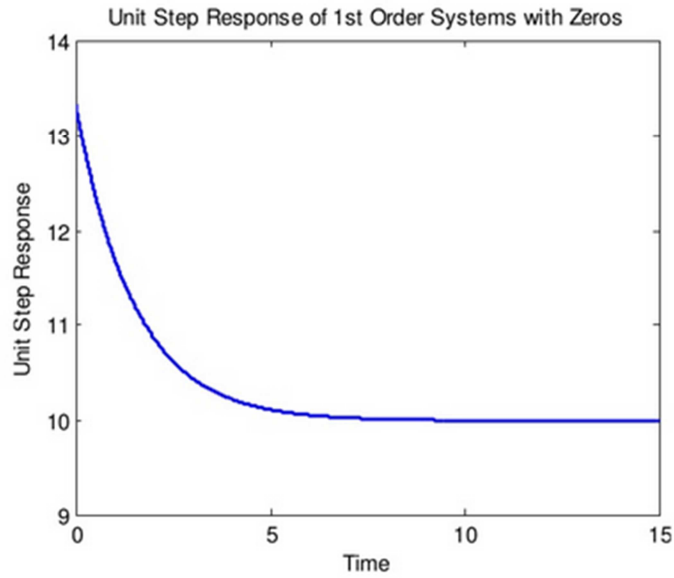


Figure 11: The step response for case 2 $T < \alpha$

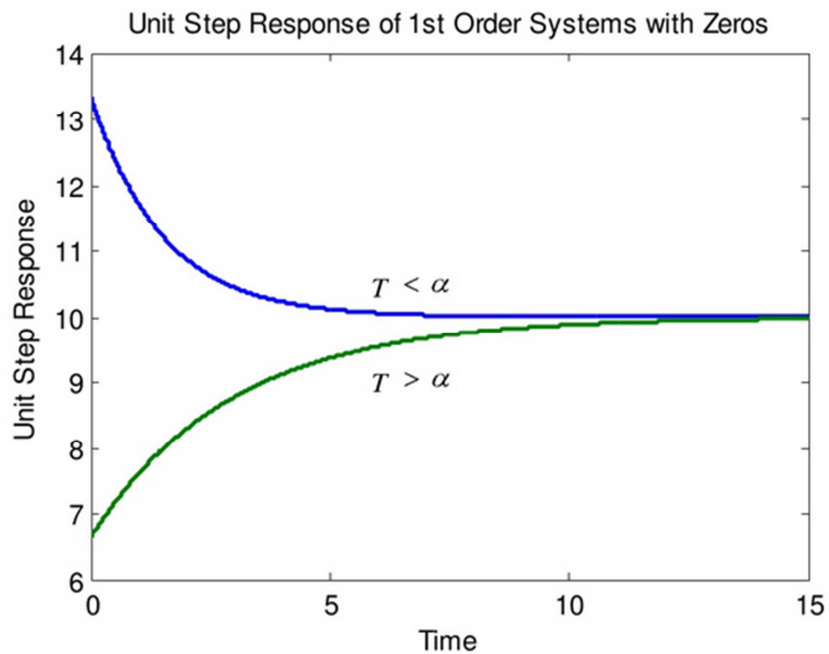


Figure 12: The step response of first order system with zero

4. First Order System with Delays

The first order system with delay time can have the following transfer function

$$\frac{C(s)}{R(s)} = \frac{K}{Ts + 1} e^{-st_d} \quad (9)$$

Where

t_d : is the delay time

Experiment 4: System Identification

The step response of this type of system is shown in figure 13

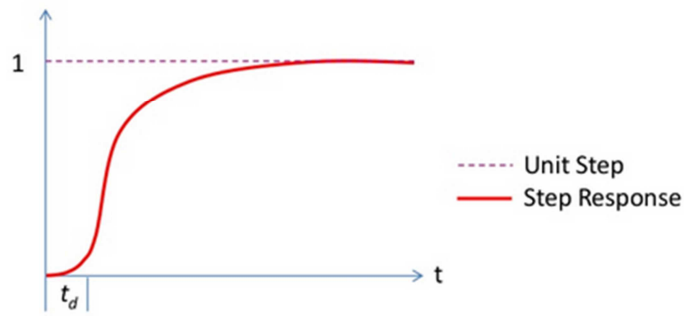


Figure 13: the step response of first order system with delay time

Example 6:

Consider the following first order system

$$G(s) = \frac{10}{s + 1} e^{-2s}$$

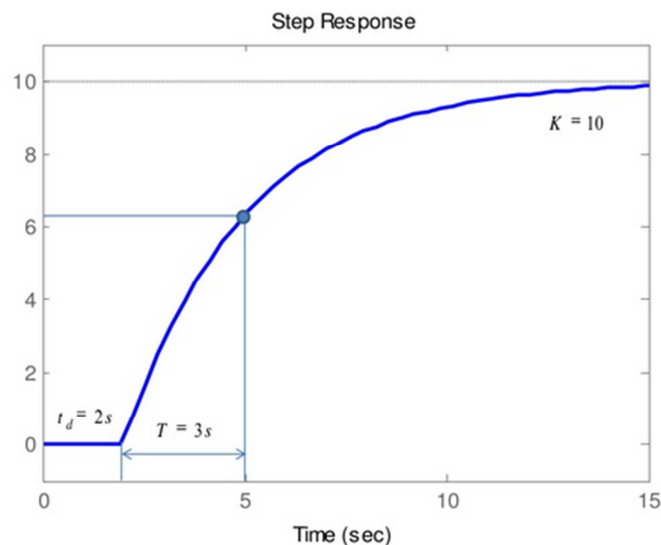


Figure 14: step response for system in Example 6

5. System Identification of Transfer Function of 1st Order Systems

The field of system identification uses statistical methods to build mathematical models of dynamical systems from measured data. System identification also includes the optimal design of experiments for efficiently generating informative data for fitting such models as well as model reduction.

A much more common approach is therefore to start from measurements of the behavior of the system and the external influences (inputs to the system) and try to determine a mathematical relation between them without going into the details of what is actually happening inside the system.

Experiment 4: System Identification

This approach is called system identification. Two types of models are common in the field of system identification:

- 1) **Grey box Model:** although the peculiarities of what is going on inside the system are not entirely known, a certain model based on both insight into the system and experimental data is constructed. This model does however still have a number of unknown free parameters which can be estimated using system identification.

One example, uses the Monod saturation model for microbial growth. The model contains a simple hyperbolic relationship between substrate concentration and growth rate, but this can be justified by molecules binding to a substrate without going into detail on the types of molecules or types of binding. Grey box modeling is also known as semi-physical modeling.

- 2) **Black box Model:** No prior model is available. Most system identification algorithms are of this type.

In science, computing, and engineering, a black box is a device, system or object which can be viewed in terms of its inputs and outputs (or transfer characteristics), without any knowledge of its internal workings. Its implementation is "opaque" (black). Almost anything might be referred to as a black box: a transistor, algorithm, or the human brain.



Figure 15: Black Box

5.1 System Identification of First order system

- Often it is not possible or practical to obtain a system's transfer function analytically.
- Perhaps the system is closed, and the component parts are not easily identifiable.
- The system's step response can lead to a representation even though the inner construction is not known.
- With a step input, we can measure the time constant and the steady-state value, from which the transfer function can be calculated.
- If we can identify T and K from laboratory testing we can obtain the transfer function of the system.

Experiment 4: System Identification

Example 7:

Assume the unit step response given in figure 16;

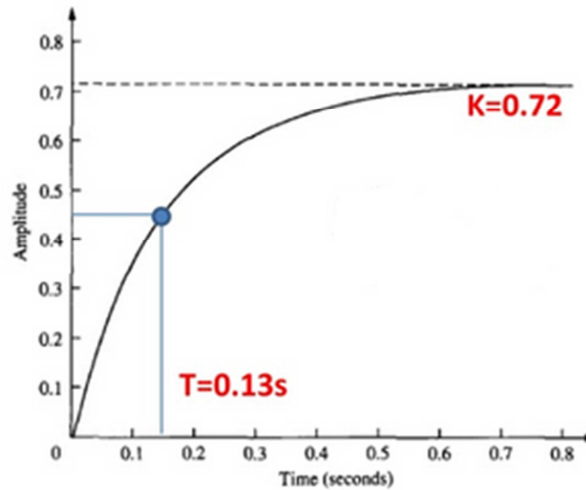


Figure 16: Unit step Response

- From the response, we can measure the **time constant T** , that is the time constant for the amplitude to reach 63% of its final value.

Since the final value is about 0.72 the time constant is evaluated for the curve reaches

$$0.63 \times 0.72 = 0.45, \text{ then } T = 0.13$$

- K is simply the steady state value.
- The transfer function is obtained as:

$$\frac{C(s)}{R(s)} = \frac{0.72}{0.13s + 1} = \frac{5.5}{s + 7.7}$$

6. Simulation of First order system using Simulink

In this section we study a open loop and closed loop system for case a first order system with delay and show the parameter of first order system.

Note: Maybe there more than blocks representation but we discuss and use the most model simulate the practical experiments as shown in the following example

Example 8:

Consider the first order system with delay given by the following transfer function

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

As shown in the previous section

$$K = 2 \quad (\text{DC gain})$$

$$T = 3 \quad (\text{Time Constant})$$

Experiment 4: System Identification

$t_d = 1$ (delay time)

We can simulate this system in Simulink using the basic block diagrams (Transfer Fcn, gain , sum and Transport Delay)

- **Open Loop System**

We assume the system in unit step input.

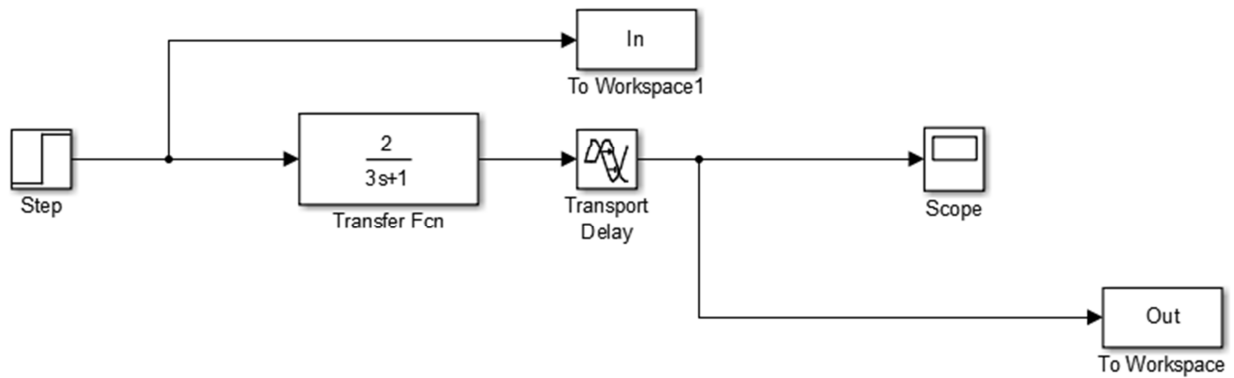


Figure 17: Open loop of system in example 8

The result of step response and the parameter is shown in figure 18

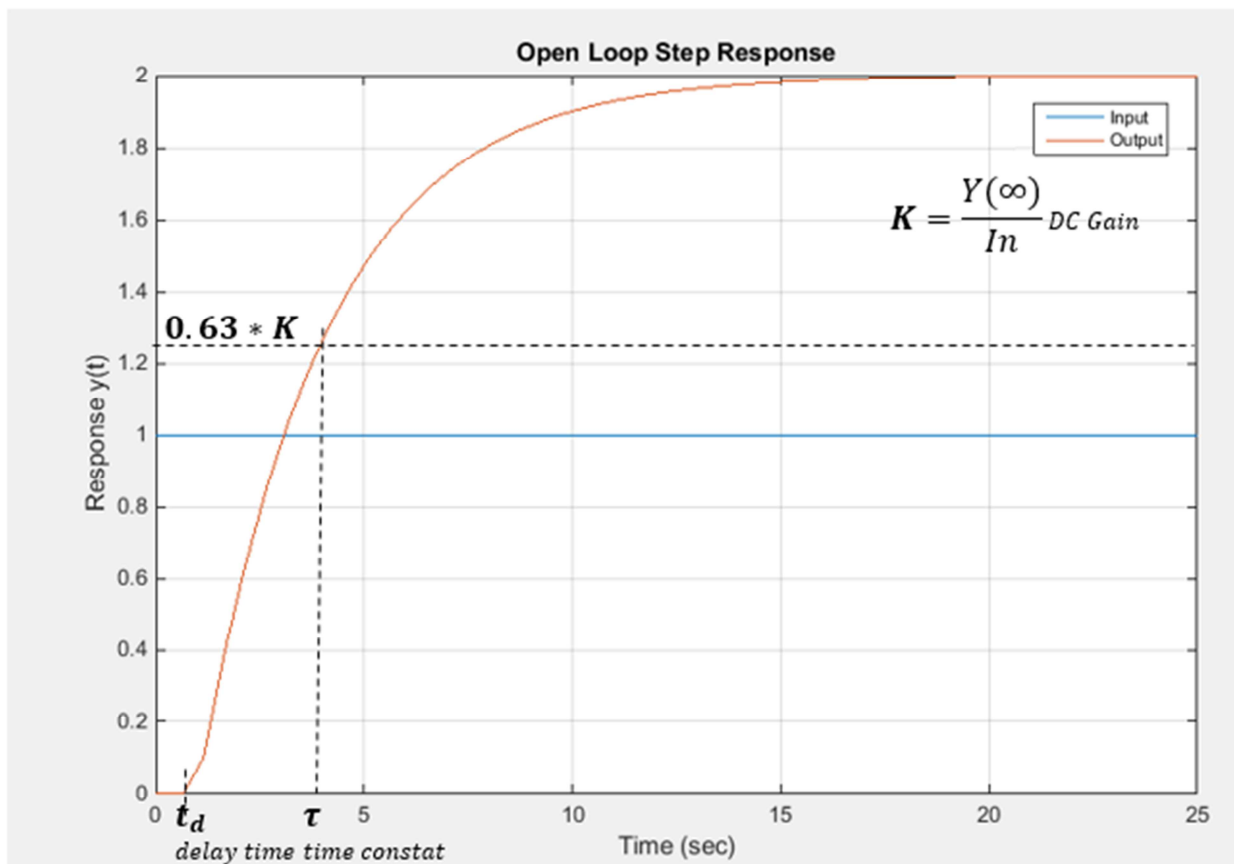


Figure 18: Open loop Response

So if the experimental data for first order system is known we can obtain the transfer function by

Experiment 4: System Identification

determine the parameter (K, τ, t_d) as discussed in the system identification example.

- **Closed loop Response**

The block diagram representation of closed loop system is shown in the figure 19

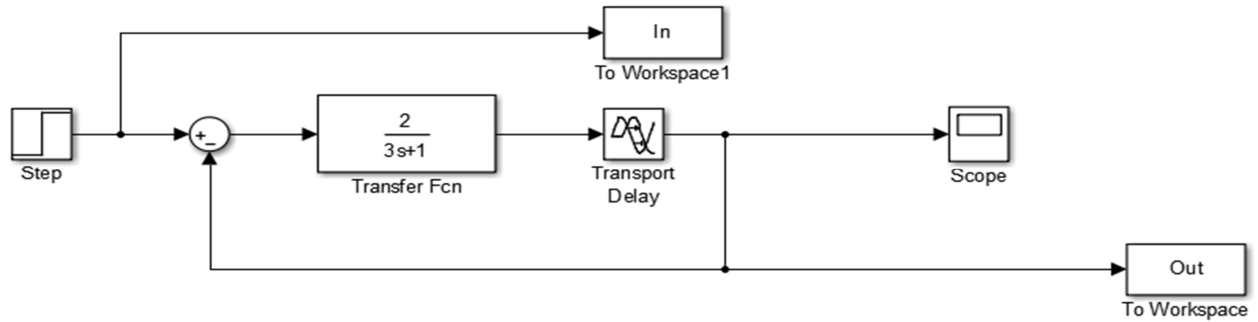


Figure 19: Closed Loop Block Diagram

The step response is shown in figure 20

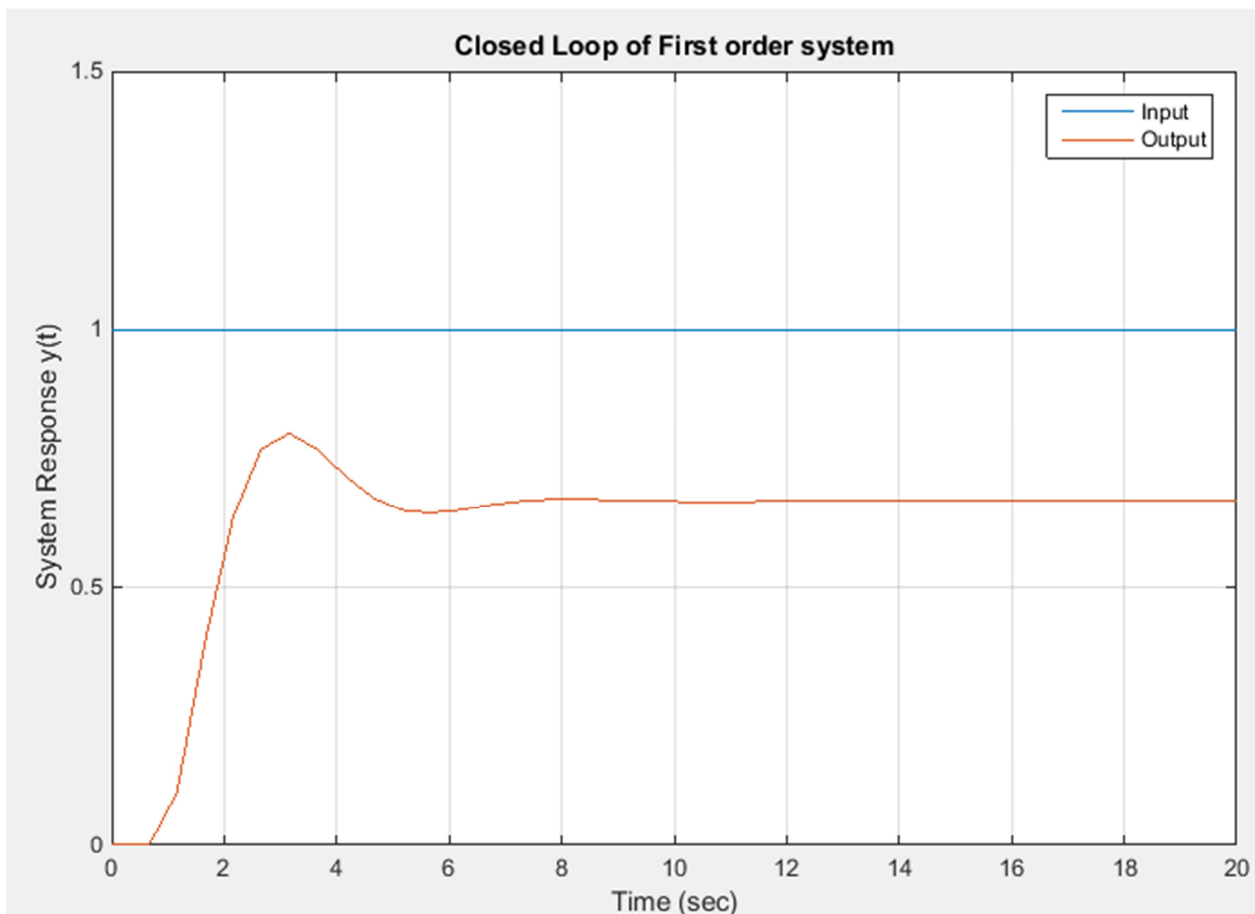


Figure 20: Step Response of closed loop

Depend on the two responses of system, we observe to determine the transfer function of the

Experiment 4: System Identification

system we must get the parameter from the open loop characteristic because in this case we obtain the response and the behavior of the system without any modification or addition effect.

System Identification Toolbox:

System Identification Toolbox provides MATLAB® functions, Simulink® blocks, and an app for constructing mathematical models of dynamic systems from measured input-output data.

It lets you create and use models of dynamic systems not easily modeled from first principles or specifications.

You can use time-domain and frequency-domain input-output data to identify continuous-time and discrete-time transfer functions, process models, and state-space models. The toolbox also provides algorithms for embedded online parameter estimation.

System Identification Toolbox lets you create models from measured input-output data. You can:

1. Analyze and process data
2. Determine suitable model structure and order, and estimate model parameters
3. Validate model accuracy

System identification is open using the command (*ident*)

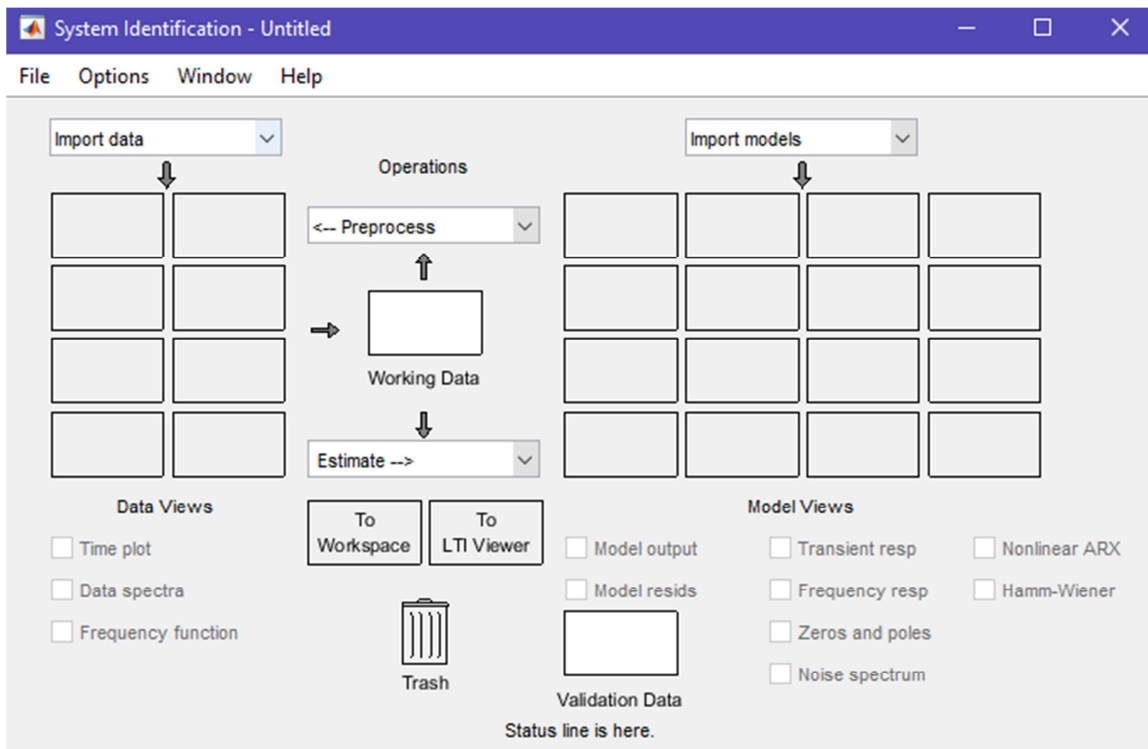


FIGURE 5: SYSTEM IDENTIFICATION TOOLBOX

1. Importing Data

When [preparing data for identifying models](#), you need to specify information such as input-output channel names, sampling time, and intersample behavior. The toolbox lets you attach this

Experiment 4: System Identification

information to the data, which facilitates visualization of data, domain conversion, and various preprocessing tasks.

The procedure of importing data as shown in figure 2:

- 1) Select the time domain data
- 2) Write the variable of input and Output

For example, if the **(Input: Vin, Output: Vc)**

- 3) Write the Starting time, and Sample time and this depend on the experimental data, for example **(Starting time :0, Sample time :0.1)**

Name ▲	Value
lc	501x1 double
t	501x1 double
tout	509x1 double
Vc	501x1 double
Vin	501x1 double
VL	501x1 double

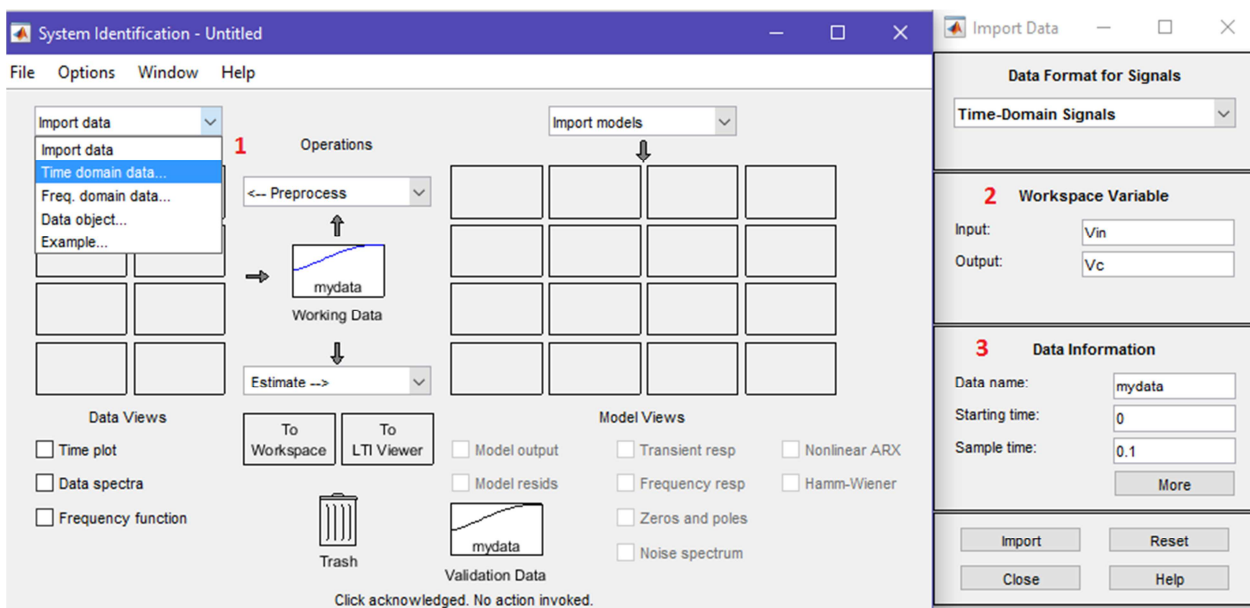


Figure 6: Importing Data

Note: This Testing Data is attached in **Data_test1.mat**

- 4) Click on Import the data will stored in model as figure 3
- 5) We can plot the data by check the box (Time Plot)

Experiment 4: System Identification

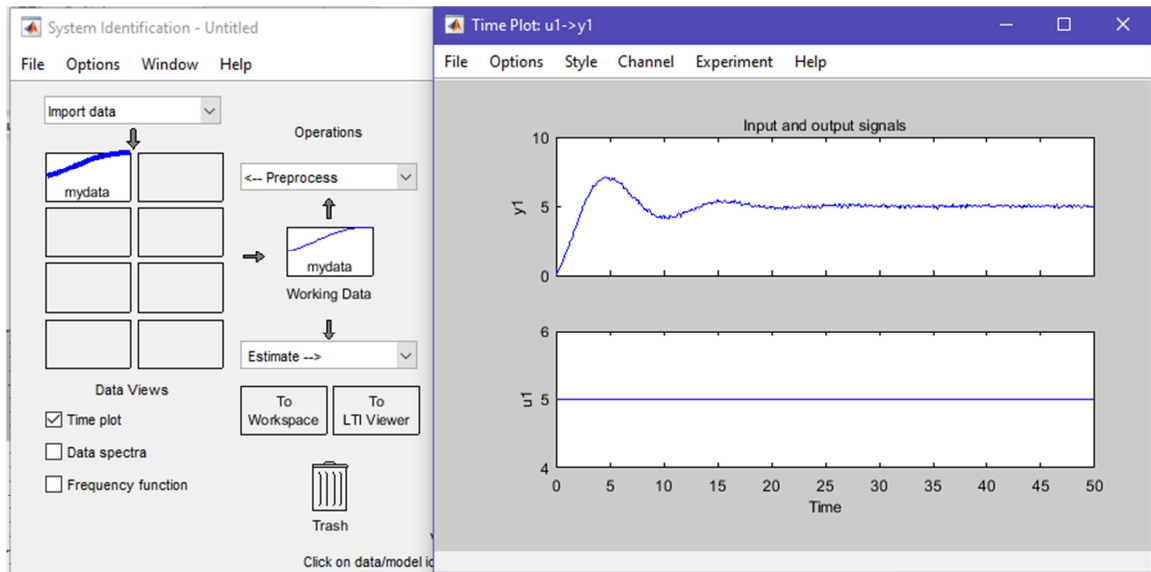


Figure 7: Imported Data

2. Estimating Model Parameters

Parametric models, such as transfer functions or state-space models, use a small number of parameters to capture system dynamics. System Identification Toolbox estimates model parameters and their uncertainties from time-response and frequency-response data. You can analyze these models using time-response and frequency-response plots, such as step, impulse, Bode plots, and pole-zero maps.

As shown in the figure 4, we can choose an estimated model from menu,

Note: the working data block must contain the data that we need to find estimated model for it,

For example, if we choose Transfer model, the new screen will appear as shown in figure 5, form this screen:

- 1) determine the number of poles and number of zeros of estimated transfer function,
- 2) Estimate the transfer function

Note:

- 1) Always started estimation by small number of poles and zeros and increase it by one and repeat the estimation.
- 2) There is no specific rules to select the optimal number of poles and zeros, you need trial and error or experience.

Experiment 4: System Identification

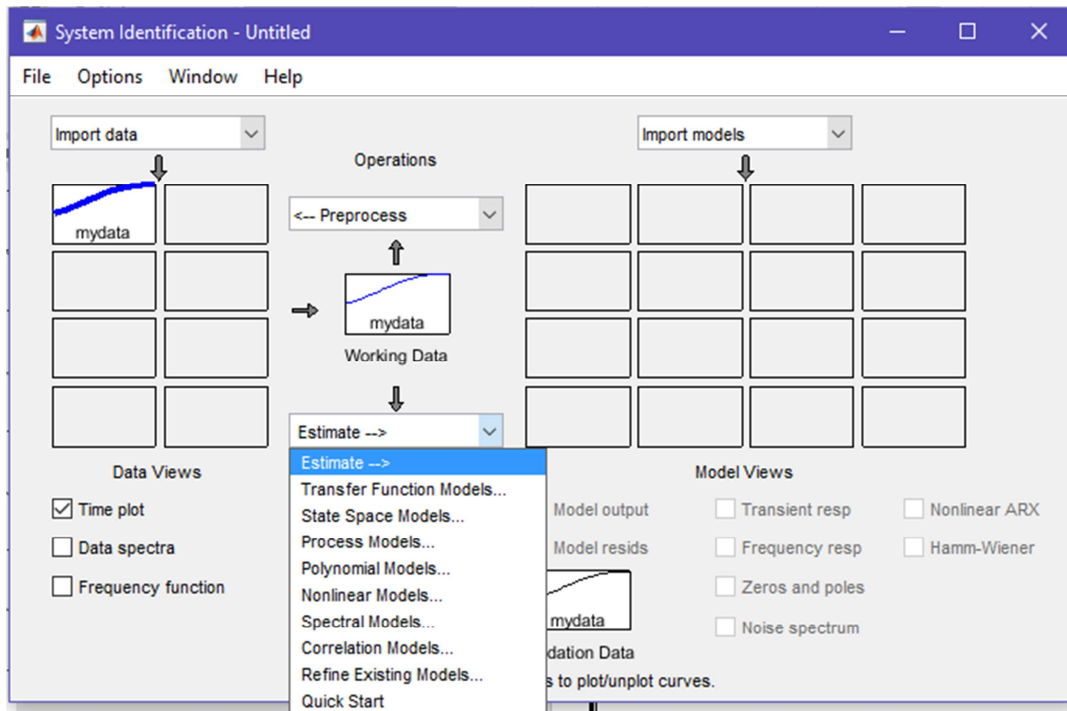


Figure 8: Estimate Model Selection

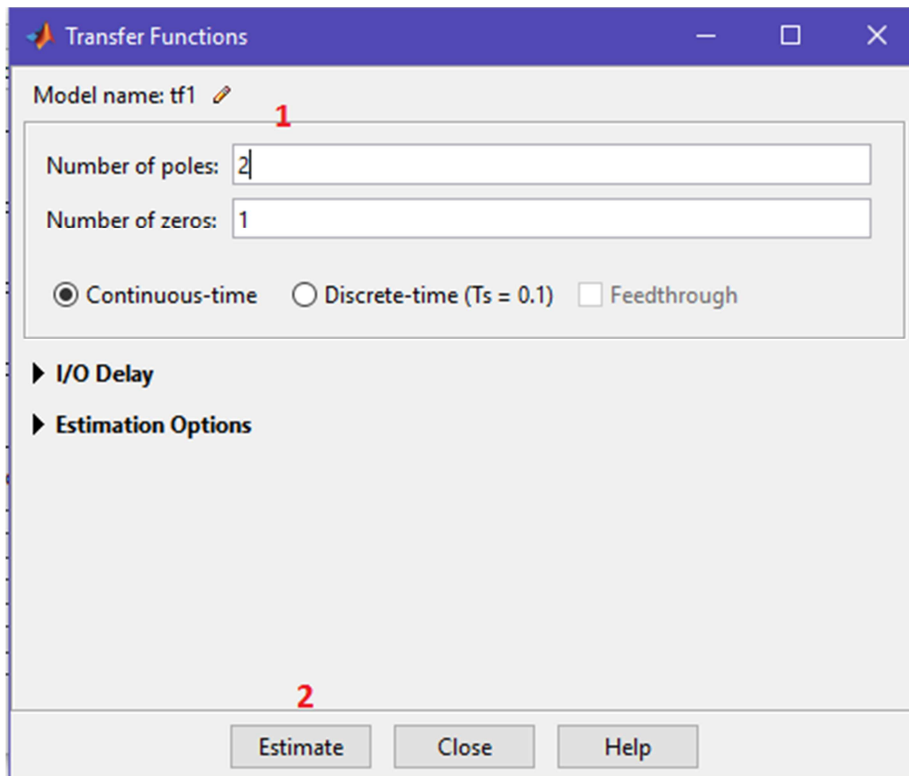


Figure 9: Estimate Transfer Function

3. Validation Model

For a given set of input data, the toolbox computes the output of the identified model and lets you compare that output with the measured output from a real system.

After estimated the model, we can compare the estimated output with measured output and determine the fitness value, as shown in figure 6

The validation Data, must be matching with Importing data, otherwise the validation will be incorrect

Check the box (Model Output) and the response will appear as shown in figure 7

Best fits: is measure the error ratio between estimated and measured output.

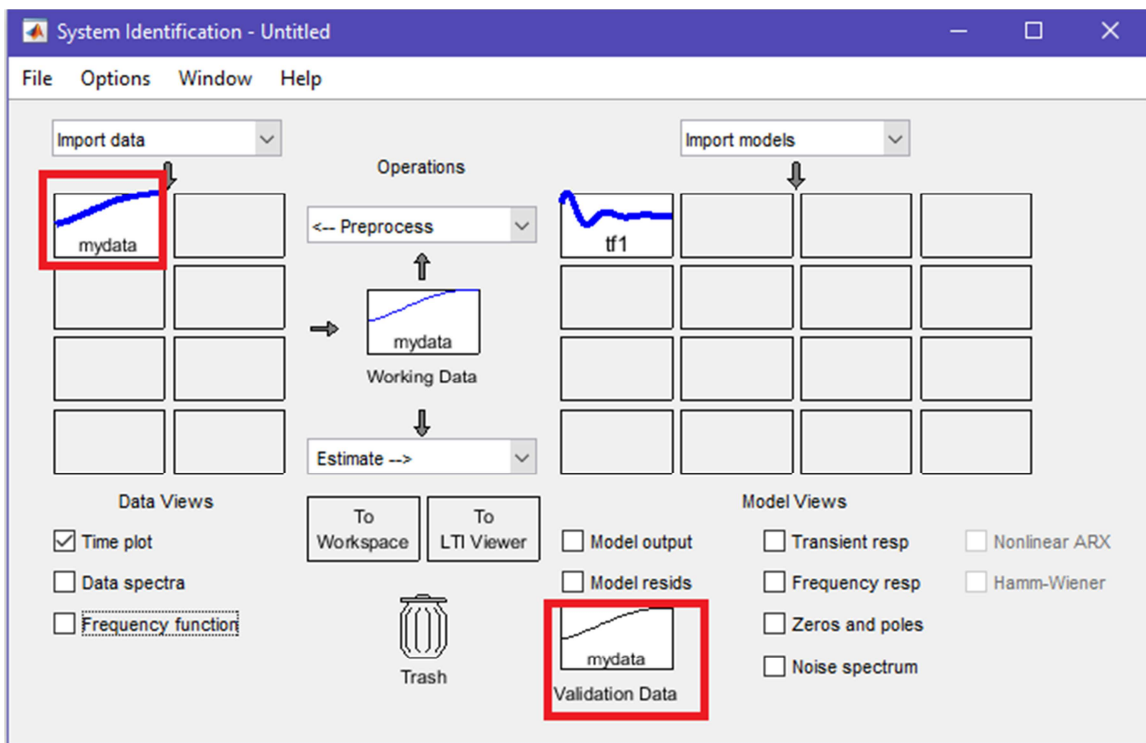
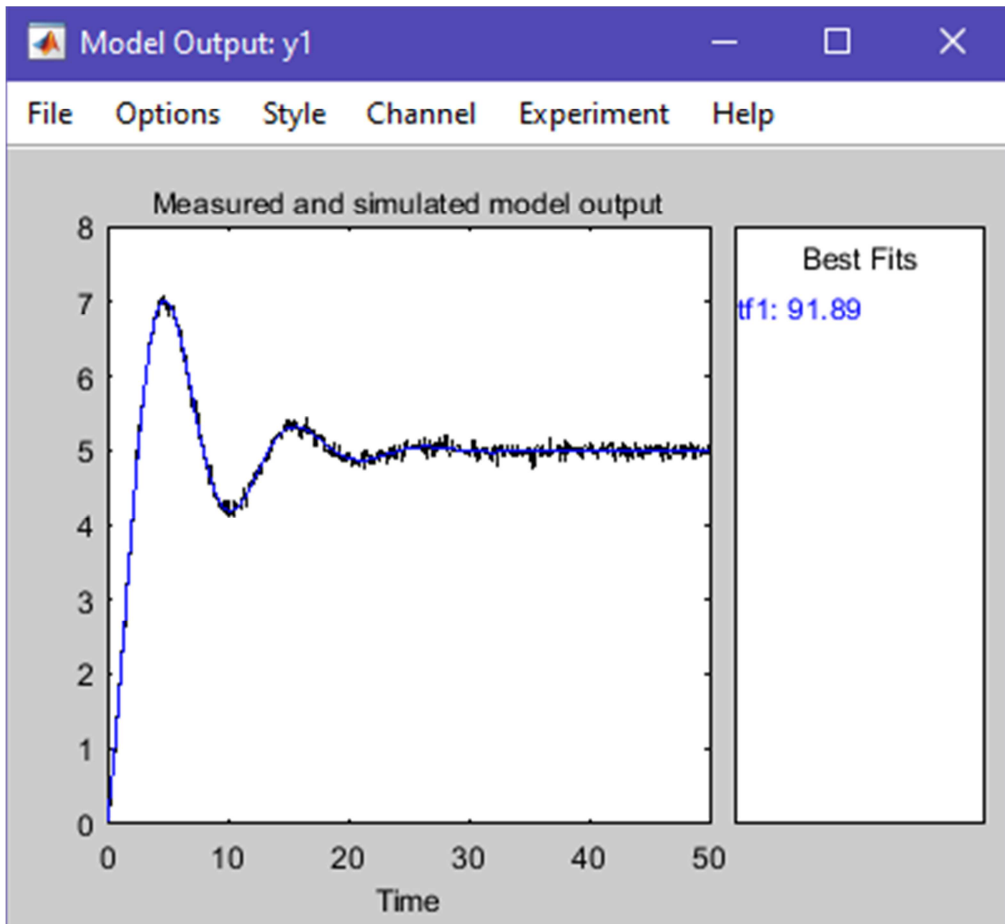
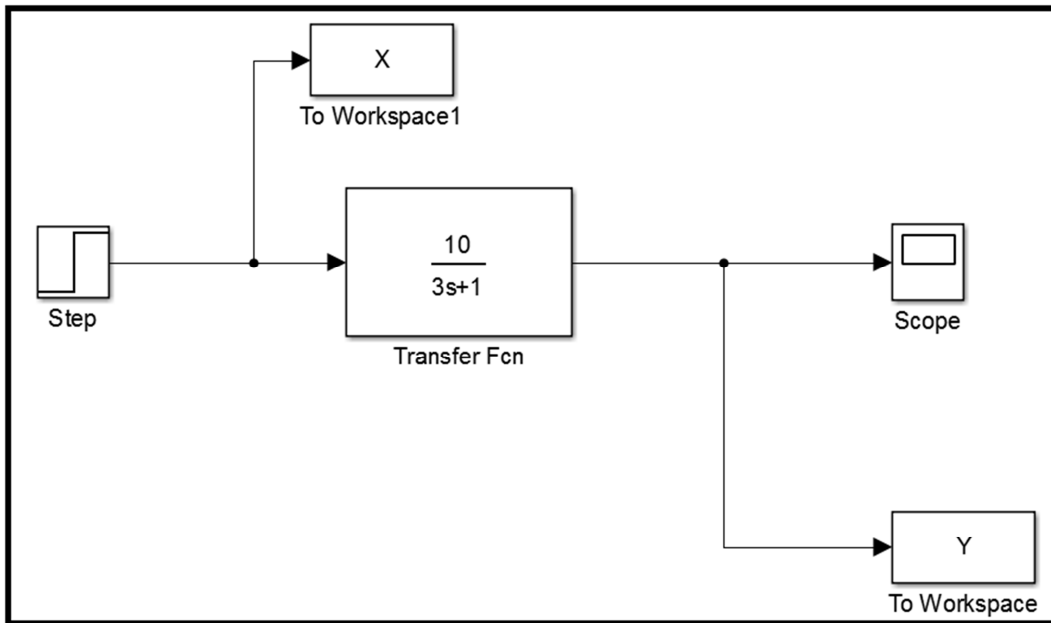


Figure 10: Importing model and validation data

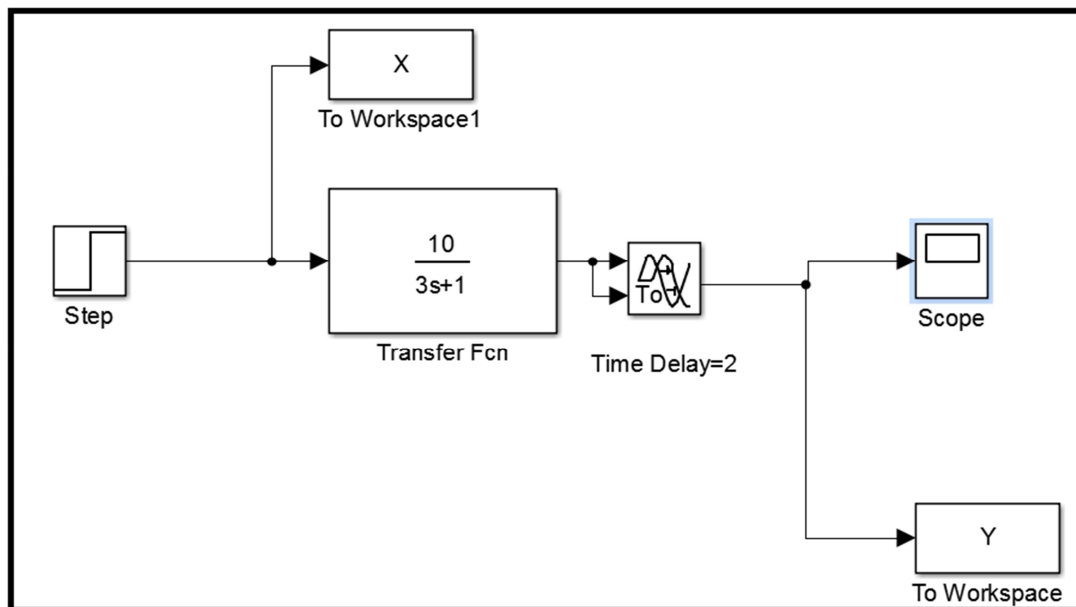


Exercises:

Build a Simulink model for the following systems:

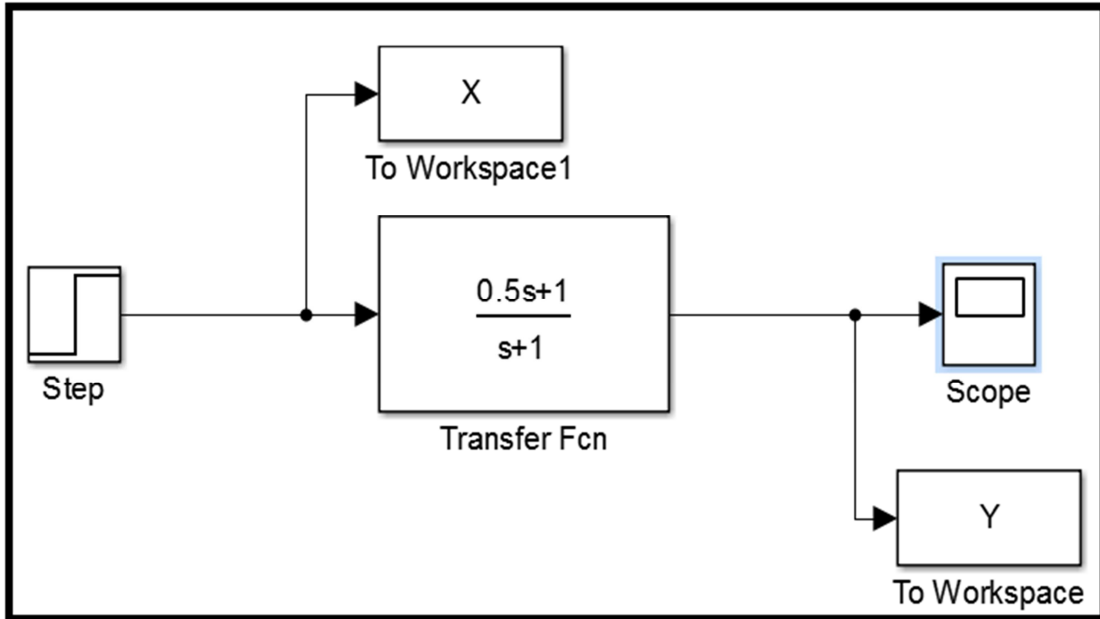


System A

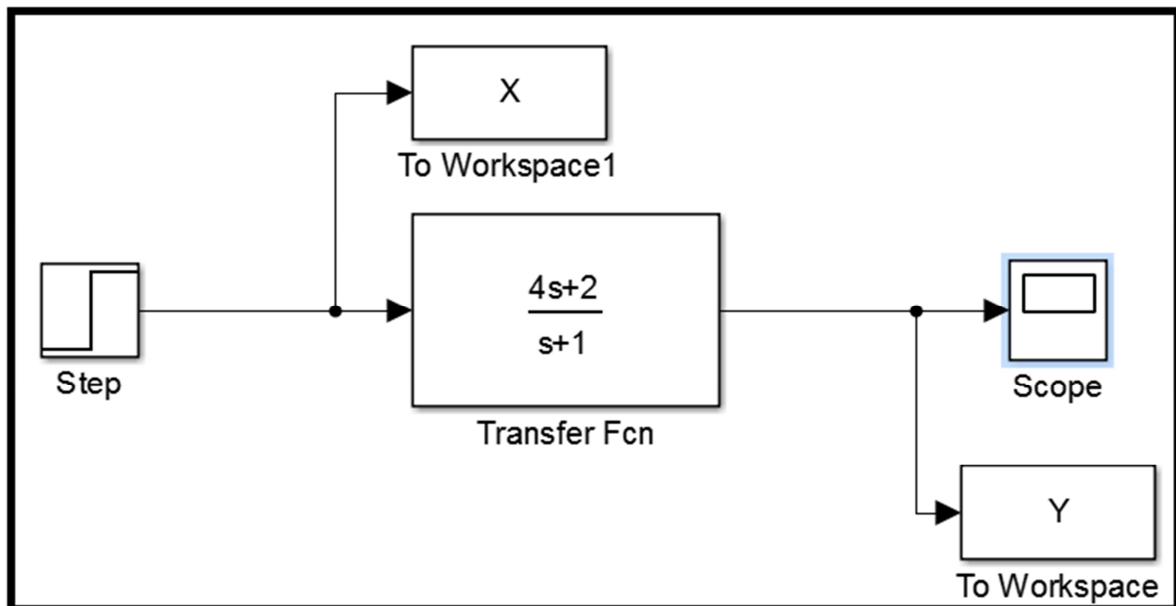


System B

Experiment 4: System Identification



System C



System D

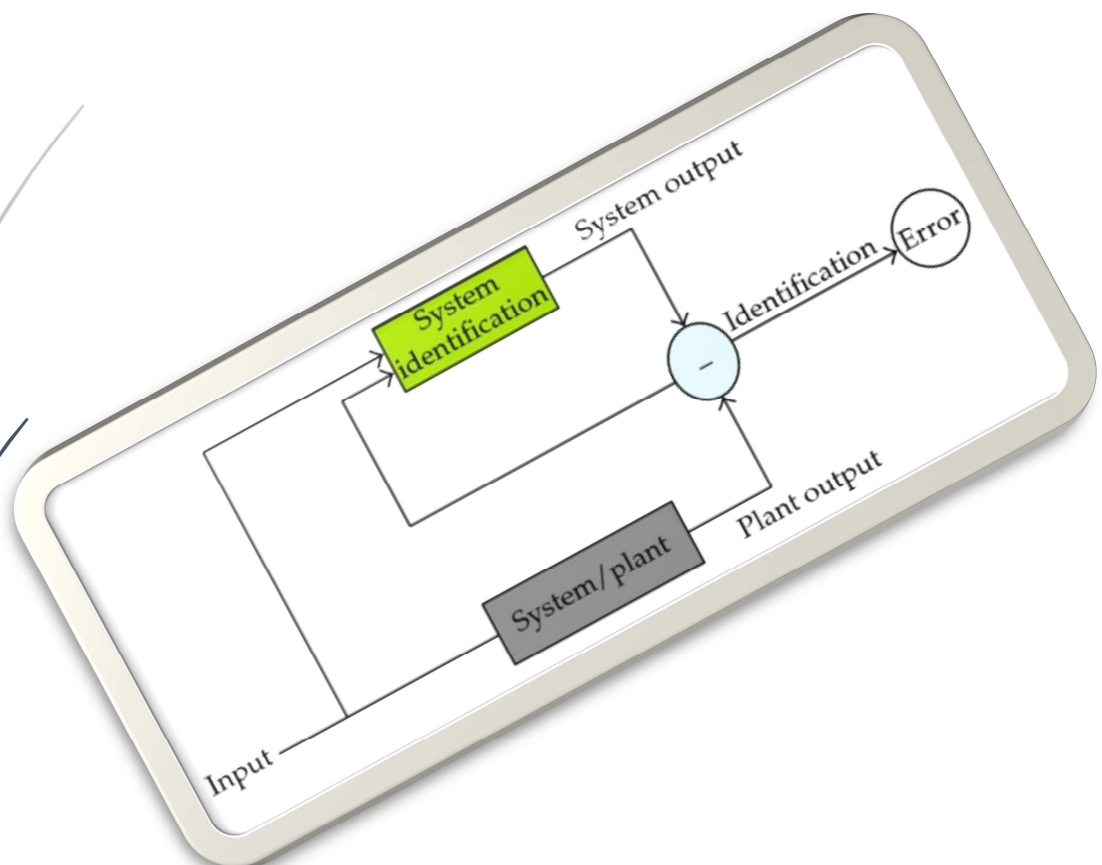
1. From the response on the scope find the transfer function for each system by calculations.
2. From X and Y data in the workspace estimate the transfer function for each system using system identification toolbox in matlab.

Experiment Five

System Identification (Second order System)

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



Second Order System

In this section, we shall obtain the response of a typical second-order control system to a step input.

In terms of damping ratio (ζ) and natural frequency (ω_n), the system shown in figure 1 , and the closed loop transfer function $C(s)/R(s)$ given by the equation 1

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \quad 1$$

This form is called the *standard form* of the second-order system.

The dynamic behavior of the second-order system can then be description in terms of two parameters ζ and ω_n .

We shall now solve for the response of the system shown in figure 1, to a unit-step input. We shall consider three different cases: the underdamped ($0 < \zeta < 1$) , critically damped ($\zeta = 1$), and overdamped ($\zeta > 1$)

1) Underdamped Case ($0 < \zeta < 1$) :

In this case, the closed-loop poles are complex conjugates and lie in the left-half s plane. The $C(s)/R(s)$ can be written as

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{(s + \zeta\omega_n + j\omega_d)(+\zeta\omega_n - j\omega_d)} \quad 2$$

Where $\omega_d = \omega_n\sqrt{1 - \zeta^2}$, the frequency ω_d is called damped natural frequency. For a unit step-input, $C(s)$ can be written

$$C(s) = \frac{\omega_n^2}{s(s^2 + 2\zeta\omega_n s + \omega_n^2)} \quad 3$$

By apply the partial fraction expansion and the inverse Laplace transform for equation 3, the response can give by

$$c(t) = 1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1 - \zeta^2}} \sin\left(\omega_d t + \tan^{-1}\left(\frac{\sqrt{1 - \zeta^2}}{\zeta}\right)\right) \quad 4$$

If the damping ratio ζ is equal to zero, the response becomes undamped and oscillations continue indefinitely. The response $c(t)$ for the zero damping case may be obtained by substituting $\zeta = 0$ in Equation 4, yielding

$$c(t) = 1 - \cos \omega_n t \quad 5$$

2) Critically Damped Case ($\zeta = 1$)

If the two poles of $C(s)/R(s)$ are equal, the system is said to be a critically damped one. For a unit-step input, $R(s) = 1/s$ and $C(s)/$ can be written

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{s(s + \omega_n)^2} \quad 6$$

By apply the partial fraction expansion and the inverse Laplace transform for equation 6, the response can give by

$$c(t) = 1 - e^{-\omega_n t}(1 + \omega_n t) \quad 7$$

3) Overdamped Case ($\zeta > 1$):

In this case, the two poles of $C(s)/R(s)$ are negative real and unequal. For a unit-step input, $R(s) = 1/s$ and $C(s)$ can be written

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{s(s + \zeta\omega_n + \omega_n\sqrt{\zeta^2 - 1})(+\zeta\omega_n - \omega_n\sqrt{\zeta^2 - 1})} \quad 8$$

By apply the partial fraction expansion and the inverse Laplace transform for equation 6, the response can give by

$$c(t) = 1 + \frac{\omega_n}{2\sqrt{\zeta^2 - 1}} \left(\frac{e^{-s_1 t}}{s_1} - \frac{e^{-s_2 t}}{s_2} \right) \quad 9$$

$$s_{1,2} = \zeta \pm \sqrt{\zeta^2 - 1}$$

Thus, the response $c(t)$ includes two decaying exponential terms.

A family of unit-step response curves $c(t)$ with various values of z is shown in Figure1 , where the abscissa is the dimensionless variable $\omega_n t$.

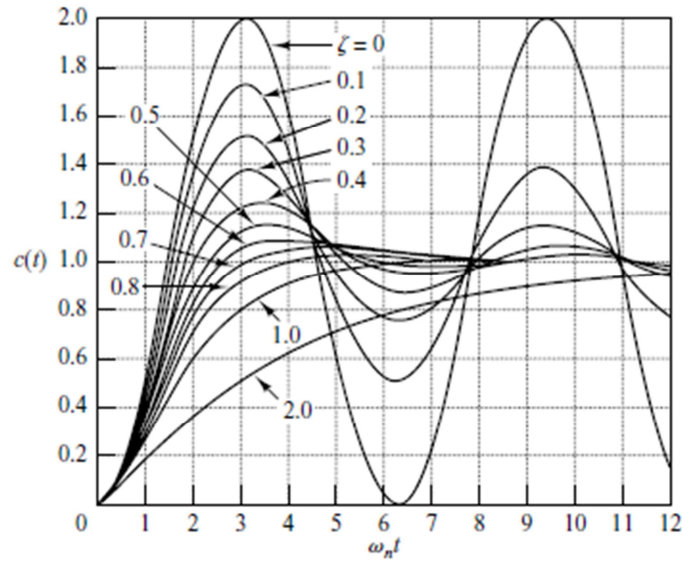


Figure 1: Unit step response curves of the system

1. Definition of Transient-Response Specification:

Frequently, the performance characteristics of a control system are specified in terms of the transient response to a unit-step input, since it is easy to generate and is sufficiently drastic. (If the response unit step input is known, it is mathematically possible to compute the response to any input.)

The transient response of a system to a unit-step input depends on the initial conditions. For convenience in comparing transient responses of various systems, it is a common practice to use the standard initial condition that the system is at rest initially with the output and all time derivatives thereof zero. Then the response characteristics of many systems can be easily compared.

The transient response of a practical control system often exhibits damped oscillations before reaching steady state. In specifying the transient-response characteristics of a control system to a unit-step input, it is common to specify the following:

1. Delay time, t_d
2. Rise time, t_r
3. Peak time, t_p
4. Maximum overshoot, M_p
5. Settling time, t_s

These specifications are defined in what follows and are shown graphically in Figure 2.

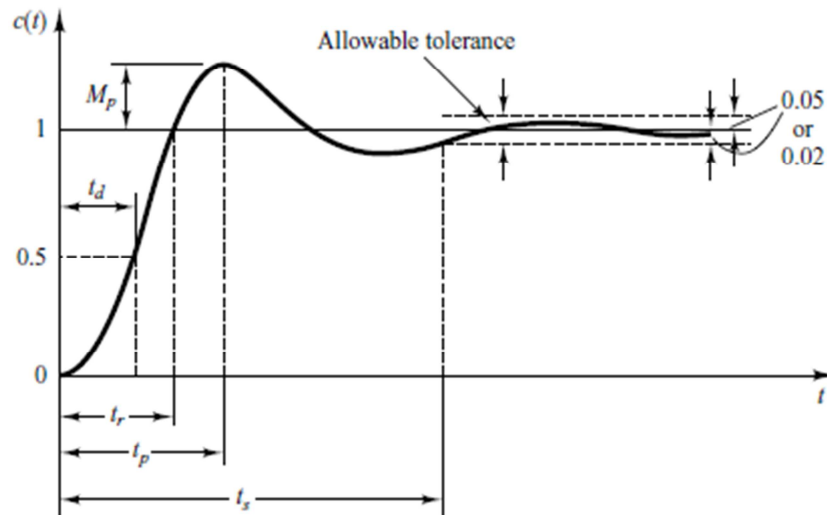


Figure 2: Step response specification

These specifications are defined in what follows and are shown graphically in Figure 2.

1. **Delay time, t_d :** The delay time is the time required for the response to reach half the final value the very first time.
2. **Rise time, t_r :** The rise time is the time required for the response to rise from 10% to 90%, 5% to 95%, or 0% to 100% of its final value. For underdamped second order systems, the 0% to 100% rise time is normally used. For overdamped systems, the 10% to 90% rise time is commonly used.
3. **Peak time, t_p :** The peak time is the time required for the response to reach the first peak of the overshoot.
4. **Maximum overshoot, M_p :** The maximum overshoot is the maximum peak value of the response curve measured from unity. If the final steady-state value of the response differs from unity, then it is common to use the maximum percent overshoot. The amount of the maximum (percent) overshoot directly indicates the relative stability of the system.
5. **Settling time, t_s :** The settling time is the time required for the response curve to reach and stay within a range about the final value of size specified by absolute percentage of the final value (usually 2% or 5%). The settling time is related to the largest time constant of the control system. Which percentage error criterion to use may be determined from the objectives of the system design in question.

The time-domain specifications just given are quite important, since most control systems are time-domain systems; that is, they must exhibit acceptable time responses. (This means that, the control system must be modified until the transient response is satisfactory.)

1.1 Second Order System and Transient- Response Specifications...

In the following, we shall obtain the rise time, peak time, maximum overshoot, and settling time of the second-order system. These values will be obtained in terms of ζ and ω_n . The system is

Experiment 5: System Identification

assumed to be underdamped.

1. Rise time , t_r

$$t_r = \frac{\pi - \beta}{\omega_d}$$

where angle β is defined in figure 3. Clearly, for a small value of t_r , ω_d must be large.

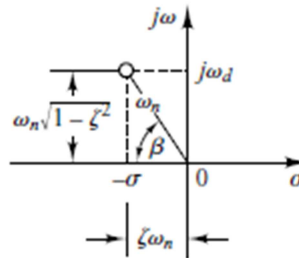


Figure 3

2. Peak time, t_p

Since the peak time corresponds to the first peak overshoot,

$$t_p = \frac{\pi}{\omega_d}$$

The peak time t_p corresponds to one-half cycle of the frequency of damped oscillation.

3. Maximum overshoot, M_p

Assuming that the final value of the output is unity

$$M_p = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}$$

If the final value $c(\infty)$ of the output is not unity, then we need to use the following equation:

$$M_p = \frac{c(t_p) - c(\infty)}{c(\infty)}$$

4. Settling time, t_s

For convenience in comparing the responses of systems, we commonly define the settling time , t_s to be

$t_s = \frac{4}{\zeta\omega_n}$	(2% criterion)
$t_s = \frac{3}{\zeta\omega_n}$	(5% criterion)

2. Higher Order Systems

In this section we shall present a transient-response analysis of higher-order systems in general terms. It will be seen that the response of a higher-order system is the sum of the responses of first-order and second-order systems.

Consider the system shown in Figure4 .The closed-loop transfer function is

Experiment 5: System Identification

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

In general, $G(s)$ and $H(s)$ are given as ratios of polynomials in s , or

$$G(s) = \frac{p(s)}{q(s)} \quad \text{and} \quad H(s) = \frac{n(s)}{d(s)}$$

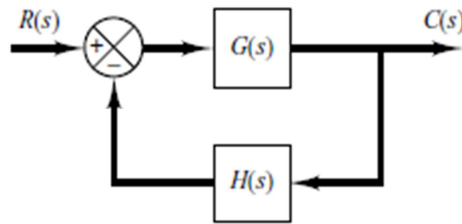


Figure 4: Control System

The closed loop T.F. of any linear invariant system can be expressed as :

$$\frac{C(s)}{R(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0}, \quad m \leq n$$

The factorized form is given as:

$$\frac{C(s)}{R(s)} = \frac{(s + z_1)(s + z_2) \dots (s + z_m)}{(s + p_1)(s + p_2) \dots (s + p_n)}$$

The response of the system for a step input can be:

1. Real distinct roots:

$$C(s) = \frac{a}{s} + \sum_{i=1}^n \frac{r_i}{s + p_i}$$

Where r_i resembles the residue of the i^{th} pole at $s = -p_i$.

$$c(t) = a + \sum_{i=1}^n r_i e_i^{-pt}$$

Notes:

- ✓ For a stable system, the relative magnitudes of the residues determine the relative importance of the corresponding poles.
- ✓ If there is a closed loop zero close to a closed loop pole, then the residue of this pole is small.
 - 1) A pair of closely located poles and zeros will effectively cancel each other.
- ✓ If a pole is located very far from origin, the residue of this pole may be small and its response will last for a short time.
- ✓ Pole having very small residues contribute little to the transient response and correspondingly may be neglected
- ✓ After neglecting the higher order system may be approximated by a

Real poles and pairs of complex conjugate poles: $C(s)$ may be given as:

$$C(s) = \frac{a}{s} + \sum_{j=1}^q \frac{r_j}{s + p_j} + \sum_{k=1}^r \frac{b_k(s + \zeta_k w_k) + c_k w_k \sqrt{1 - \zeta_k^2}}{s^2 + 2\zeta_k w_k s + w_k^2}$$

This means that the factored form of the poles of higher order systems consists of first and 2nd order terms. As a result, the response of the higher order system is composed of a number of terms involving the responses of first order and 2nd order systems. The response is given as:

$$c(t) = a + \sum_{j=1}^q a_j e^{-p_j t} + \sum_{k=1}^r b_k e^{-\zeta_k w_k t} \cos \left[\left(w_k \sqrt{1 - \zeta_k^2} \right) t \right] + \sum_{k=1}^r c_k e^{-\zeta_k w_k t} \sin \left[\left(w_k \sqrt{1 - \zeta_k^2} \right) t \right]$$

which means that for a stable higher order with nonrepeated simple or complex roots, the response is the sum of a number of exponential curves (for real distinct roots) and damped sinusoidal curves (for unrepeated complex poles).

- ✓ For a stable higher order system, the exponential terms and the damped or sinusoidal curves will approach zero as $t \rightarrow \infty$ and the steady state output $y_{ss} = a = y(\infty)$
- ✓ As the real part of the poles moves farther from the origin or |real part| increase then the response of that pole decay rapidly to zero and correspondingly the setting time of that pole decrease. That is,

$$t_s \propto \frac{1}{\left| \begin{array}{c} \text{real} \\ \text{part} \end{array} \right|}$$

- ✓ The type of transient response is determined by the closed loop poles, while the zeros of the close loop T.F. do affect the magnitudes and signs of the residues of the expanded terms.
- ✓ The poles of the input $R(s)$ yield the steady state in the solution while the poles of the closed loop T.F. yield the transient response terms of the solution as they enter exponential transient response terms and/or damped sinusoidal transient response terms.

3.1 Dominant closed loop poles:

The relative dominance of closed loop poles is determined by:

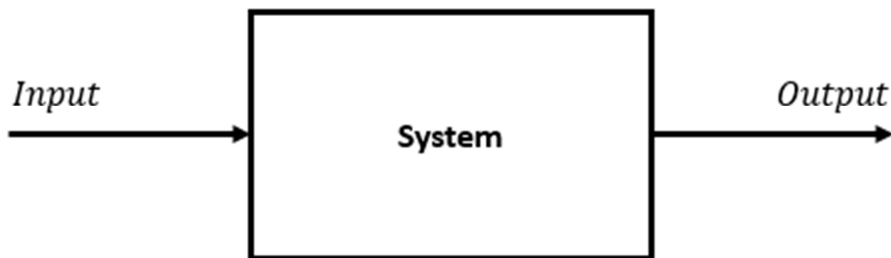
- 1) The ratio of the real parts of closed loop poles

Experiment 5: System Identification

- 2) The relative magnitudes of the poles residues which depend on both the closed loop poles and zeros.
 - ✓ If the real part of the closest pole to the “ $j\omega$ ” axis is (5 - 10) times less than the real part of the closest pole to this pole and there are no zeros nearby, then former pole is called dominant closed loop pole since this pole will dominant the transient response and will decay slowly.
 - ✓ The dominant closest loop poles are the most important among all closed loop poles.

3. System Identification of Second Order and Higher Order System

In the control system, the system identification process is applied to system by assume the input is step response



And the question now, How I can determine the transfer function of the system form measured output data ??

The step response is given in the following figure 5, and to determine the transfer function we follow the following step

- 1) Determine the settling time and the Overshoot of system
- 2) Determine the natural frequency ω_n and the damping ratio ζ

$$M_p = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}}$$

$t_s = \frac{4}{\zeta\omega_n}$	(2% criterion)
$t_s = \frac{3}{\zeta\omega_n}$	(5% criterion)

- 3) The standard form of second order system is given by

$$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

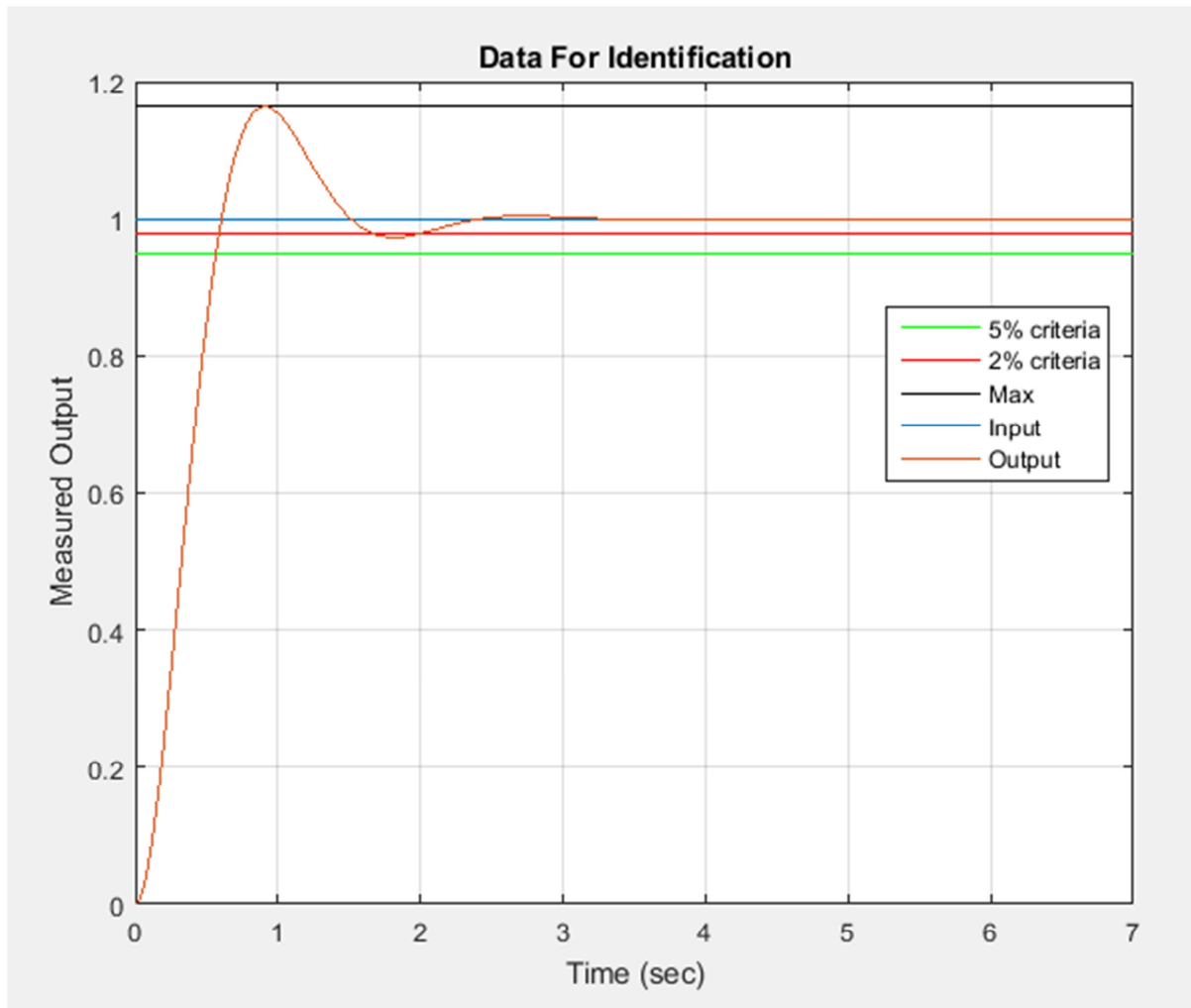


Figure 5

From the over shoot determine ζ

$$M_p = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} = 1.163 - 1 = 0.163 \quad \text{we find } \zeta = 0.5$$

And from the settling time , determine

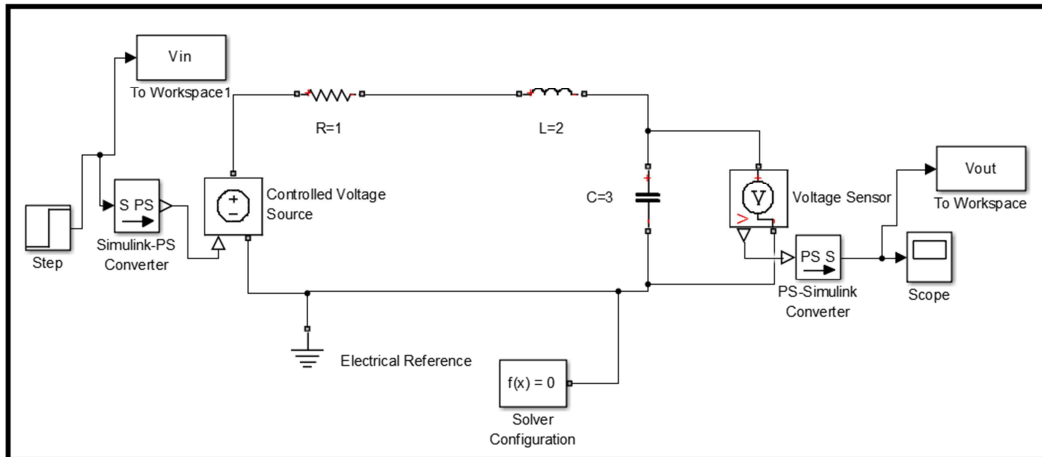
$$t_s = \frac{4}{\zeta\omega_n} = 2 \quad \text{we find } \omega_n = 4$$

$$\frac{16}{s^2 + 4s + 16}$$

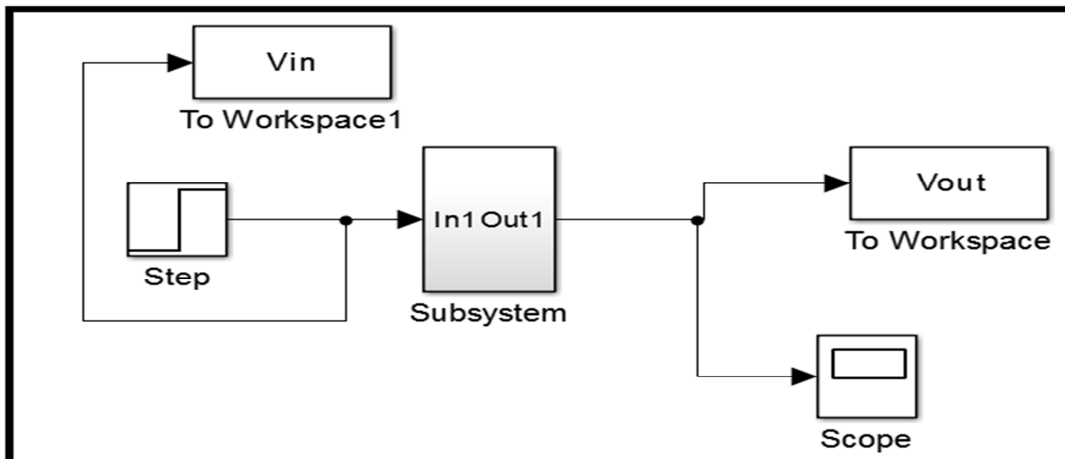
Experiment 5: System Identification

Exercises:

Exercise-1: Build the following system using Simscape:



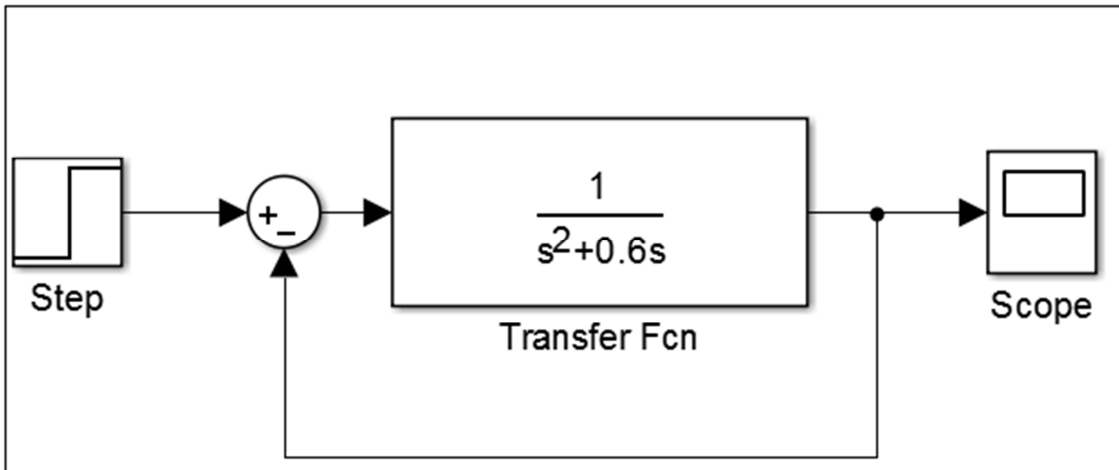
Create subsystem as shown below and suppose it unknown system:



From V_{in} and V_{out} data in the workspace estimate the transfer function for the system using system identification toolbox.

Experiment 5: System Identification

Exercise-2: Build a Simulink model for the following system:



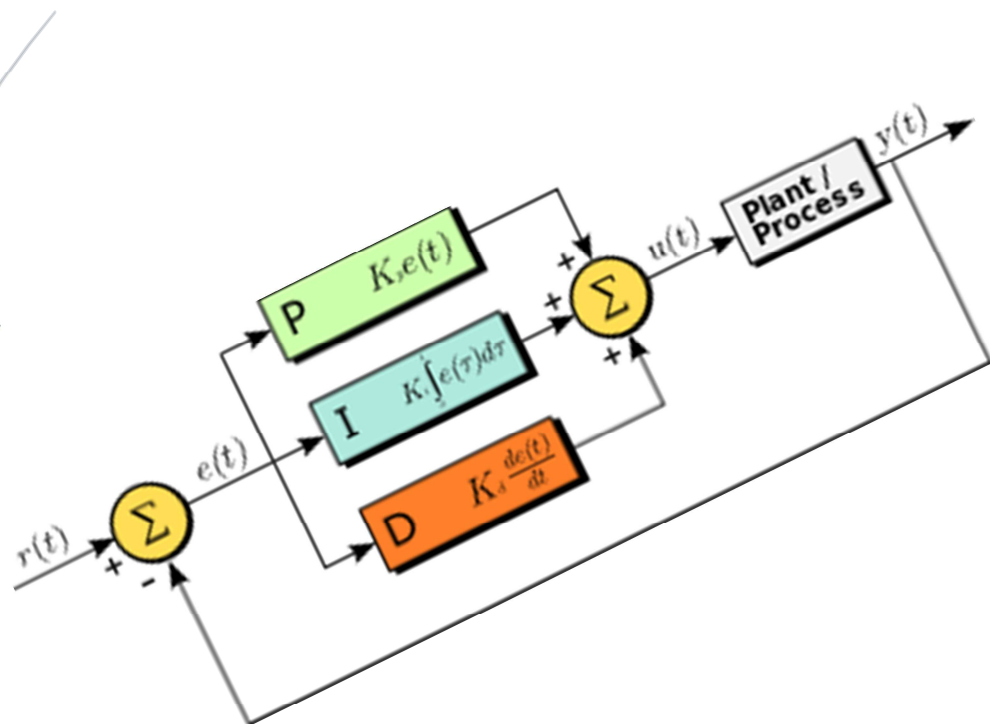
Use the output response data for identification and find the transfer function.

Experiment Six

PID Controller

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



Introduction

A proportional-integral-derivative controller (PID controller) is a control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller calculates an error value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process through use of a manipulated variable.

The PID controller algorithm involves three separate constant parameters, and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted P, I, and D. Simply put, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve, a damper, or the power supplied to a heating element.

In the absence of knowledge of the underlying process, a PID controller has historically been considered to be the most useful controller. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint, and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

Some applications may require using only one or two actions to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are fairly common, since derivative action is sensitive to measurement noise, whereas the absence of an integral term may prevent the system from reaching its target value due to the control action.

1. PID Controller Theory

The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output,

Controller manufacturers arrange the Proportional, Integral and Derivative modes into three different controller algorithms or controller structures. These are called Interactive, Noninteractive, and Parallel algorithms. Some controller manufacturers allow you to choose between different controller algorithms as a configuration option in the controller software. The PID Algorithms are:

Experiment 6: PID Controller

1) Interactive Algorithm

$$u(t) = K_c \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau \right] \times \left[1 + T_d \frac{d}{dt} e(t) \right]$$

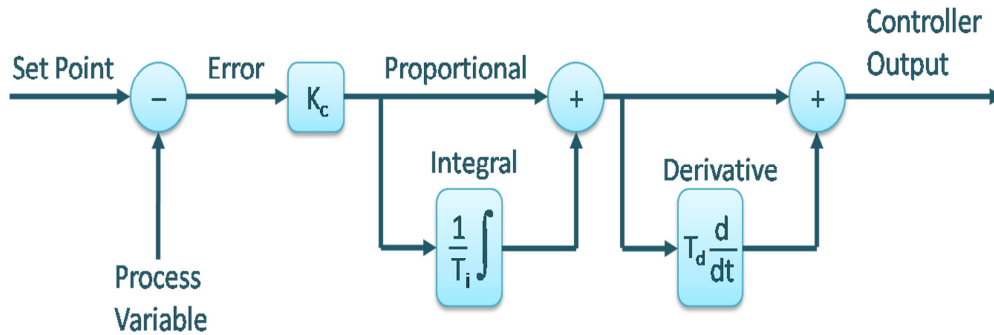


Figure 1: Interactive Algorithm

2) NonInteractive Algorithm

$$u(t) = K_c \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{d}{dt} e(t) \right]$$

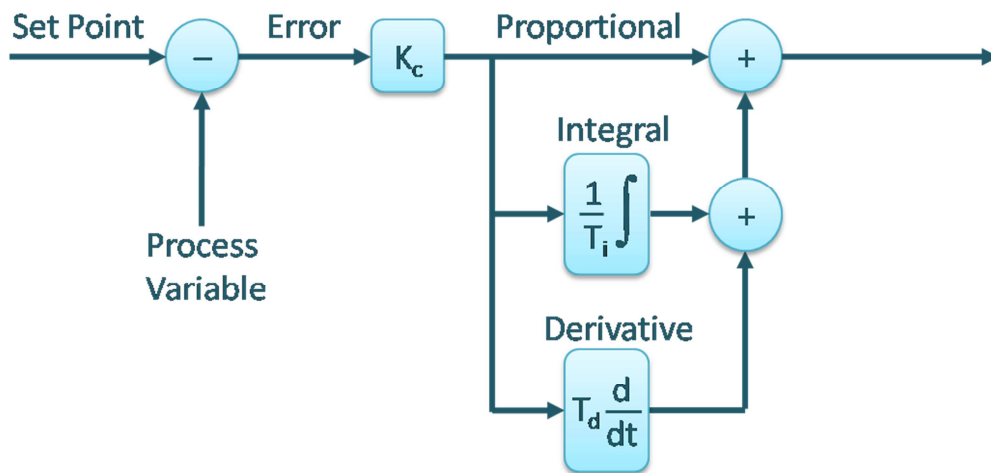


Figure 2: NonInteractive Algorithm

3) Parallel Algorithm

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t)$$

Experiment 6: PID Controller

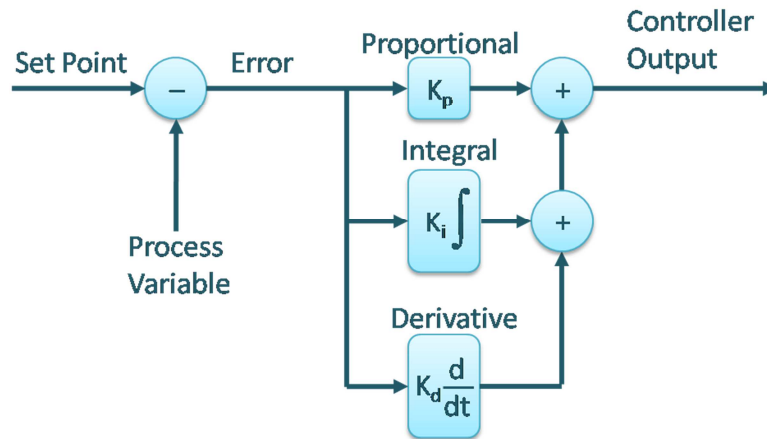


Figure 3: Parallel Algorithm

Where

$K_p = K_c$: Proportional Gain

$K_i = \frac{K_c}{T_i}$: Integral Gain

$K_d = K_c T_d$: Derivative Gain

$e(t) = r(t) - y(t)$

1.1 Proportional Term

The proportional term produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p , called the proportional gain constant. The proportional term is given by:

$$P_{out} = K_p e(t)$$

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive or less sensitive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Tuning theory and industrial practice indicate that the proportional term should contribute the bulk of the output change

Experiment 6: PID Controller

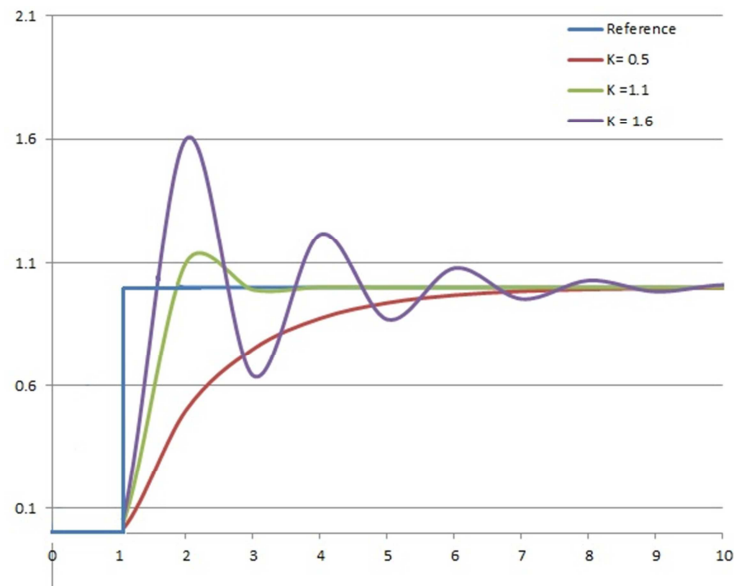


Figure 4: The effect of add K_p (K_i , and K_d) held constant

2.2 Integral Term

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain K_i and added to the controller output.

$$I_{out} = K_i \int_0^t e(\tau) d\tau$$

The integral term accelerates the movement of the process towards set-point and eliminates the residual steady-state error that occurs with a pure proportional controller. However, since the integral term responds to accumulated errors from the past, it can cause the present value to overshoot the set-point value.

2.3 Derivative Term

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d . The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d . The derivative term is given by

$$D_{out} = K_d \frac{d}{dt} e(t)$$

Experiment 6: PID Controller

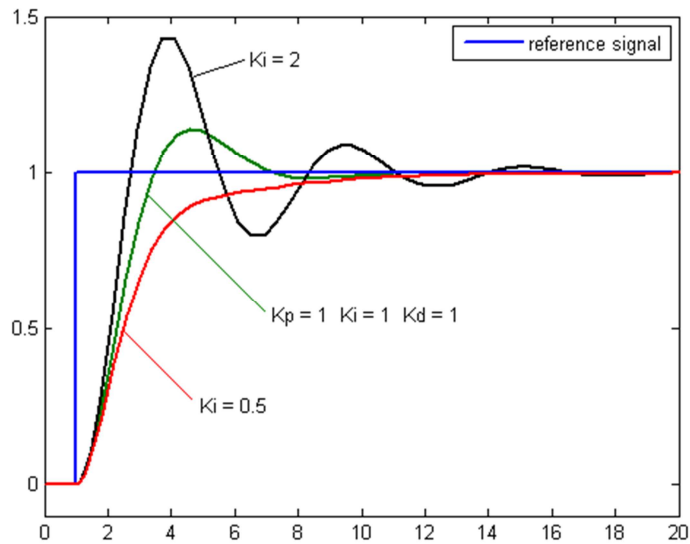


Figure 5: The effect of add K_i (K_p , and K_d) held constant

Derivative action predicts system behavior and thus improves settling time and stability of the system. An ideal derivative is not causal, so that implementations of PID controllers include an additional low pass filtering for the derivative term, to limit the high frequency gain and noise. Derivative action is seldom used in practice though - by one estimate in only 20% of deployed controllers- because of its variable impact on system stability in real-world applications.

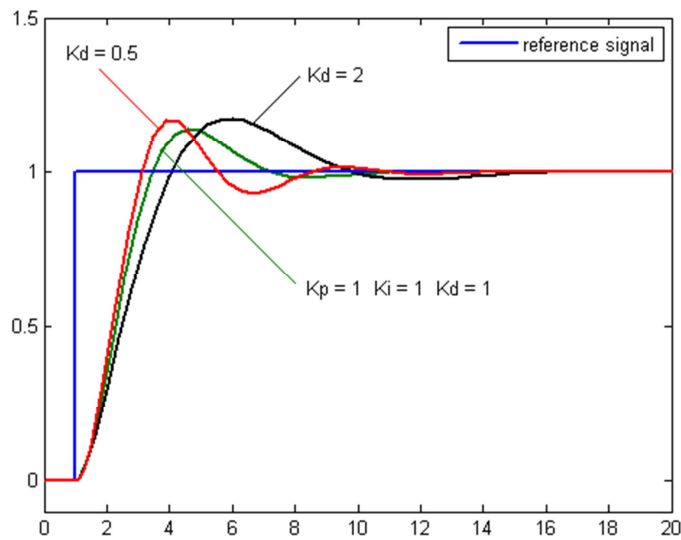


Figure 6 The effect of add K_d (K_p , and K_i) held constant

Table 1: Effect of increasing parameter independently

Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
K_p	Decrease	Increase	Small Change	Decrease	Degrade

Experiment 6: PID Controller

K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor Change	Decrease	Decrease	No Effect	Improve if K_d small

2. Overview of Methods

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively inefficient, particularly if the loops have response times on the order of minutes or longer.

The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and on the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Table 2: Choosing a Tuning Method

Method	Advantages	Disadvantages
Manual Tuning	No math required , Online	Requires experienced personnel
Ziegler-Nichols	Proven Method, Online	Process upset, some trial-and-error, very aggressive tuning
Cohen-Coon	Good process models	Some math; offline; only good for first-order processes
Software Tools	Consistent tuning; online or offline - can employ computer-automated control system design (CAutoD) techniques;	Some cost or training involved

3. Open Loop Method

In these methods, the PID is being tuned in open loop, isolated from the process plant. First a step input is applied to the plant and the process reaction curve is obtained. Using the process reaction curve with one of the First Order Plus Dead Time (FOPDT) estimation methods an approximation of the process is calculated. Knowing K_m , τ_m and t_d the PID parameters can be evaluated from the related correlations according to the method used.

First Order Plus Dead Time (FOPDT) is given by

$$G(s) = \frac{K_m}{\tau_m s + 1} e^{-t_d s}$$

3.1 Ziegler-Nichols Open Loop Method

In the 1940's, Ziegler and Nichols devised two empirical methods for obtaining controller parameters. Their methods were used for first order plus dead time situations, and involved intense manual calculations. With improved optimization software, most manual methods such as these are no longer used. However, even with computer aids, the following two methods are still employed today, and are considered among the most common.

This method remains a popular technique for tuning controllers that use proportional, integral, and derivative actions. The Ziegler-Nichols open-loop method is also referred to as S-shaped curve method, because it tests the open-loop reaction of the process to a change in the control variable output. This basic test requires that the response of the system be recorded, preferably by a plotter or computer. Once certain process response values are found, they can be plugged into the Ziegler-Nichols equation with specific multiplier constants for the gains of a controller with either P, PI, or PID actions.

In this method, we obtain experimentally the open loop response of the FOPDT to a unit step input. This method only applied if the response to a step input exhibits an *s*-shaped curve as shown in figure 7. This means that if the plant involves integrators (like 2nd order prototypes system) or complex-conjugate poles (general 2nd order system), then this method can't be applied since *s*-shaped will not be obtained.

This method remains a popular technique for tuning controllers that use proportional, integral, and derivative actions. The Ziegler-Nichols open-loop method is also referred to as a process reaction method, because it tests the open-loop reaction of the process to a change in the control variable output.

The Tuning Procedure:

To use the Ziegler-Nichols open-loop tuning method, you must perform the following steps:

1. Make an open loop step test
2. From the process reaction curve determine the transportation lag or dead time, t_d , the time constant or time for the response to change, τ_m , and the ultimate value that the response reaches at steady-state, K_m , for a step change of X_o .
3. Determine the loop tuning constants. Plug in the reaction rate and lag time values to the Ziegler-Nichols open-loop tuning equations for the appropriate controller (P, PI, or PID) to calculate the controller constants. Use the table 3.

Experiment 6: PID Controller

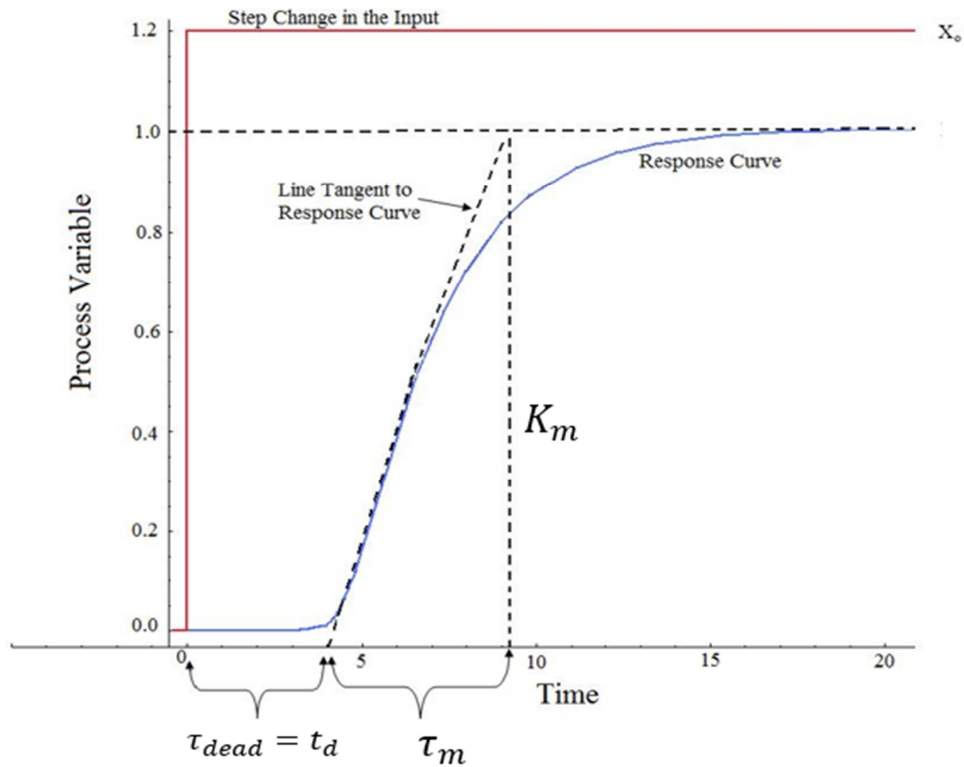


Figure 3: Open Loop of First order system plus dead Time (s-shaped curve)

Table 3: Open-loop Calculation of (K_p, T_i, T_d)

	K_p	T_i	T_d
P- Controller	$\frac{X_o \tau_m}{K_m t_d}$	∞	0
PI- Controller	$0.9 \frac{X_o \tau_m}{K_m t_d}$	$3.3 t_d$	0
PID- Controller	$1.2 \frac{X_o \tau_m}{K_m t_d}$	$2 t_d$	$0.5 t_d$

The PID controller tuned by this method gives (according to the formula shown in Table (8.4)).

$$\begin{aligned}
 G_c(s) &= K_p \left[1 + \frac{1}{T_i s} + T_d s \right] \\
 &= 1.2 \frac{X_o \tau_m}{K_m t_d} \left(1 + \frac{1}{2t_d s} + 0.5t_d s \right) \\
 &= 0.6\tau_m \frac{(s + 1/t_d)^2}{s}
 \end{aligned}$$

- This mean that the controller adds double zero at $s = -\frac{1}{t_d}$, and pole at origin
- Advantages Ziegler-Nichols Open Loop Tuning Methods

Experiment 6: PID Controller

1. Quick and easier to use than other methods
 2. It is a robust and popular method
- Disadvantages Ziegler-Nichols Open Loop Tuning Methods
 1. It depends upon purely t_d to estimate I and D controllers.
 2. Approximations for the K_p , T_i , and T_d values might not be entirely accurate for different systems.
 3. It does not hold for I, D and PD controllers

3.2 Cohen-Coon Method

The Cohen-Coon tuning rules are suited to a wider variety of processes than the Ziegler-Nichols tuning rules. The Ziegler-Nichols rules work well only on processes where the dead time is less than half the length of the time constant. The Cohen-Coon tuning rules work well on processes where the dead time is less than two times the length of the time constant (and you can stretch this even further if required). Also it provides one of the few sets of tuning rules that has rules for PD controllers.

Like the Ziegler-Nichols tuning rules, the Cohen-Coon rules aim for a quarter-amplitude damping response. Although quarter-amplitude damping-type of tuning provides very fast disturbance rejection, it tends to be very oscillatory and frequently interacts with similarly-tuned loops. Quarter-amplitude damping-type tuning also leaves the loop vulnerable to going unstable if the process gain or dead time doubles in value.

In this method the process response curve is obtained first, by an open loop test as shown in figure 8 and then the process dynamics is approximated by a first order plus dead time model, with following parameters:

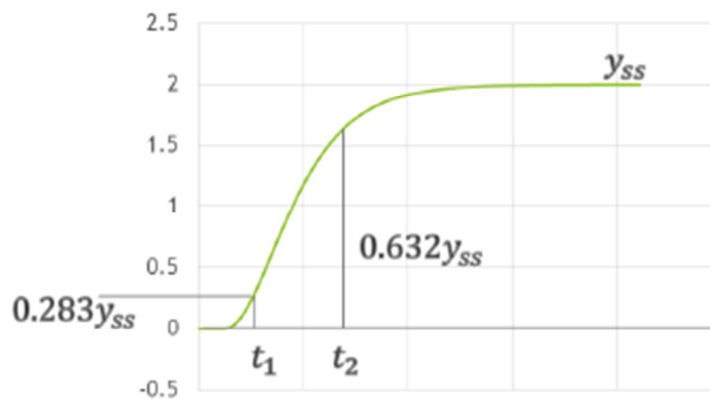


Figure 7: The open loop response of plant

$$\tau_m = \frac{3}{2}(t_2 - t_1)$$

$$t_d = t_2 - \tau_m$$

Again the particular rules for this method are used to calculate the PID parameters. They are

listed in table 4

Table 3: the parameter of Cohen-Coon Method

Controller	K_c	T_i	T_d
P	$\frac{\tau_m}{Kt_d} \left(1 + \frac{t_d}{3\tau_m}\right)$	-	-
PI	$\frac{\tau_m}{Kt_d} \left(0.9 + \frac{t_d}{12\tau_m}\right)$	$t_d \left(\frac{30 + \frac{3t_d}{\tau_m}}{9 + \frac{20t_d}{\tau_m}}\right)$	-
PD	$\frac{\tau_m}{Kt_d} \left(1.25 + \frac{t_d}{6\tau_m}\right)$	-	$t_d \left(\frac{6 - \frac{2t_d}{\tau_m}}{22 + \frac{3t_d}{\tau_m}}\right)$
PID	$\frac{\tau_m}{Kt_d} \left(1.33 + \frac{t_d}{4\tau_m}\right)$	$t_d \left(\frac{32 + \frac{6t_d}{\tau_m}}{13 + \frac{8t_d}{\tau_m}}\right)$	$t_d \left(\frac{4}{11 + \frac{2t_d}{\tau_m}}\right)$

4. Ziegler-Nichols Closed - Loop Tuning Method

The Ziegler-Nichols closed-loop tuning method allows you to use the critical gain value, K_{cr} , and the critical period of oscillation, P_{cr} , to calculate K_p . It is a simple method of tuning PID controllers and can be refined to give better approximations of the controller. You can obtain the controller constants K_p , T_i , and T_d in a system with feedback. The Ziegler-Nichols closed-loop tuning method is limited to tuning processes that cannot run in an open-loop environment.

Determining the ultimate gain value, K_{cr} , is accomplished by finding the value of the proportional-only gain that causes the control loop to oscillate indefinitely at steady state. This means that the gains from the I and D controller are set to zero so that the influence of P can be determined. It tests the robustness of the K_p value so that it is optimized for the controller. Another important value associated with this proportional-only control tuning method is the critical period (P_{cr}). The ultimate period is the time required to complete one full oscillation while the system is at steady state. These two parameters, K_{cr} and P_{cr} , are used to find the loop-tuning constants of the controller (P, PI, or PID). To find the values of these parameters, and to calculate the tuning constants, use the following procedure:

- The Tuning Procedure:
 1. Remove integral and derivative action. Set integral time (T_i) to ∞ or its largest value and set the derivative controller (T_d) to zero.
 2. Create a small disturbance in the loop by changing the set point. Adjust the proportional, increasing and/or decreasing, the gain until the oscillations have constant amplitude.
 3. Record the gain value (K_{cr}) and period of oscillation (P_{cr}).

Experiment 6: PID Controller

4. Plug these values into the Ziegler-Nichols closed loop equations and determine the necessary settings for the controller.

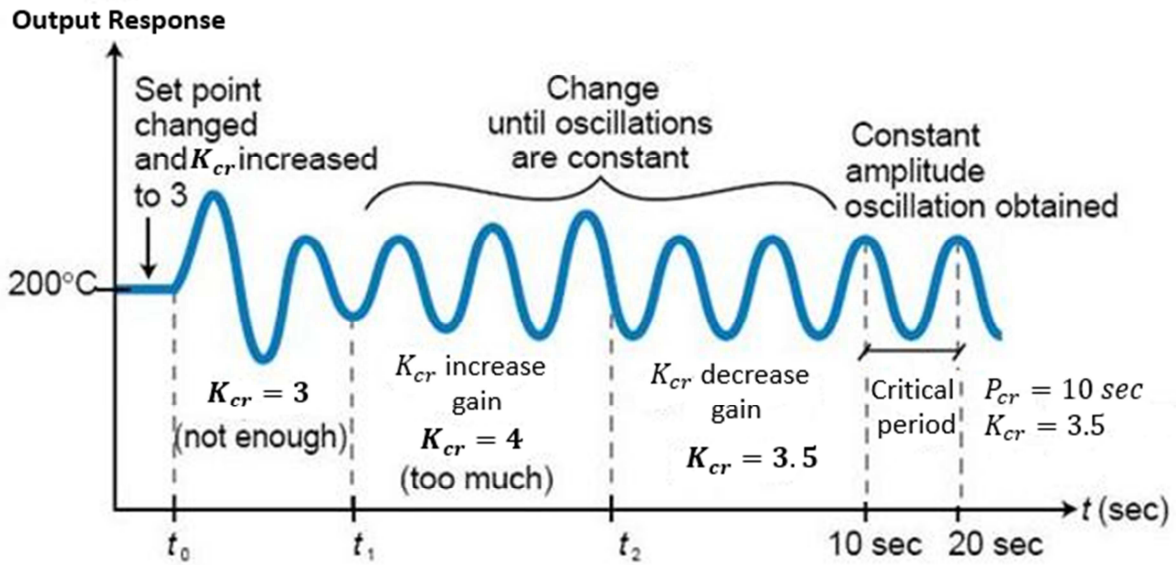


Figure 9: System tuned using the Ziegler-Nichols closed-loop tuning method

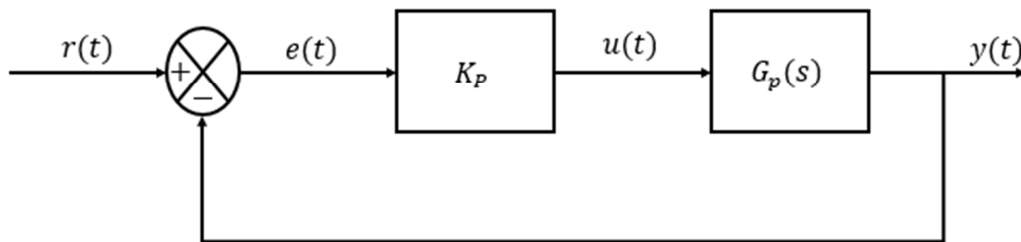


Figure 10: The Control system with Gain K_p

Table 4: Closed-Loop Calculation of (K_p, T_i, T_d)

	K_p	T_i	T_d
P- Controller	$\frac{K_{cr}}{2}$	∞	0
PI- Controller	$\frac{K_{cr}}{2.2}$	$\frac{P_{cr}}{1.2}$	0
PID- Controller	$\frac{K_{cr}}{1.7}$	$\frac{P_{cr}}{2}$	$\frac{P_{cr}}{8}$

The PID controller tuned by this method gives (according to the formula shown in Table 8.5).

Experiment 6: PID Controller

$$\begin{aligned}G_c(s) &= K_p \left[1 + \frac{1}{T_i s} + T_d s \right] \\&= 0.6K_{cr} \left(1 + \frac{1}{0.5P_{cr}s} + 0.125P_{cr}s \right) \\&= 0.075K_{cr}P_{cr} \frac{(s + 4/P_{cr})^2}{s}\end{aligned}$$

Thus the PID Controllers adds a pole at origin and double zeros at $s = -\frac{4}{P_{cr}}$

If the system has a known mathematical model (Transfer function is given), then RL method can be used to find K_{cr} value (critical gain) and the frequency of the sustained oscillations w_{cr} . After that P_{cr} is found from

$$P_{cr} = \frac{2\pi}{w_{cr}}$$

These values can be found from the crossing points of the root locus branches with the jw axis. This method doesn't apply if the root locus doesn't cross the jw axis.

- Advantages Ziegler-Nichols Closed-Loop Tuning Methods
 1. Easy experiment; only need to change the P controller
 2. Includes dynamics of whole process, which gives a more accurate picture of how the system is behaving
- Disadvantages Ziegler-Nichols Closed-Loop Tuning Methods
 1. Experiment can be time consuming
 2. Can venture into unstable regions while testing the P controller, which could cause the system to become out of control

5. Software Method (PID Tuning Toolbox In MATLAB)

5.1 Automatically Tune PID Controller Gains

PID tuning is the process of finding the values of proportional, integral, and derivative gains of a PID controller to achieve desired performance and meet design requirements.

PID controller tuning appears easy, but finding the set of gains that ensures the best performance of your control system is a complex task. Traditionally, PID controllers are tuned either manually or using rule-based methods. Manual tuning methods are iterative and time-consuming, and if used on hardware, they can cause damage. Rule-based methods also have serious limitations: they do not support certain types of plant models, such as unstable plants, high-order plants, or plants with little or no time delay.

You can automatically tune PID controllers to achieve the optimal system design and to meet design requirements, even for plant models that traditional rule-based methods cannot handle well.

An automated PID tuning workflow involves:

Experiment 6: PID Controller

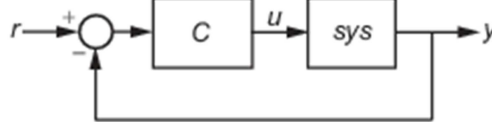
- Identifying plant model from input-output test data
- Modeling PID controllers in MATLAB using PID objects or in Simulink using PID Controller blocks
- Automatically tuning PID controller gains and fine-tune your design interactively
- Tuning multiple controllers in batch mode
- Tuning single-input single-output PID controllers as well as multiloop PID controller architectures

5.2 PID Tuning Toolbox

Can be use the PID tuning toolbox to determine the parameter of controller depend on the system form MATLAB or SIMULINK as following step:

1) MATLAB

Use the PID Tuner to interactively design a SISO PID controller in the feed-forward path of single-loop, unity-feedback control configuration



The PID Tuner automatically designs a controller for your plant. You specify the controller type (P, I, PI, PD, PDF, PID, PIDF) and form (parallel or standard). You can analyze the design using a variety of response plots, and interactively adjust the design to meet your performance requirements.

To launch the PID Tuner, use the pidTuner command:

```
pidTuner(sys,type)
```

where sys is a linear model of the plant you want to control, type is a string indicating the controller type to design

PID Controller Type

The PID Tuner can tune up to seven types of controllers. To select the controller type, use one of these methods:

Provide the type argument to the launch command pidTuner.

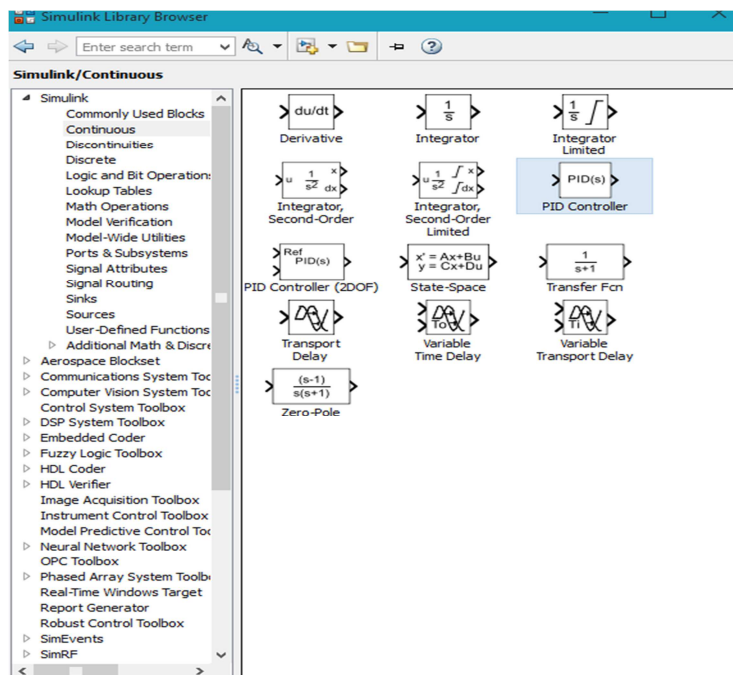
In PID Tuner, use the Type menu to change controller types.

Experiment 6: PID Controller

type input to pidTuner	Entry in Type menu	Controller Type	Continuous-time Controller Formula (parallel form)	Discrete-time Controller Formula (parallel form, ForwardEuler integrator formulas)
'p'	P	Proportional only	K_p	K_p
'i'	I	Integral only	$\frac{K_i}{s}$	$K_i \frac{T_s}{z-1}$
'pi'	PI	Proportional and integral	$K_p + \frac{K_i}{s}$	$K_p + K_i \frac{T_s}{z-1}$
'pd'	PD	Proportional and derivative	$K_p + K_d s$	$K_p + K_d \frac{z-1}{T_s}$
'pdf'	PDF	Proportional and derivative with first-order filter on derivative term	$K_p + \frac{K_d s}{T_f s + 1}$	$K_p + K_d \frac{1}{T_f s + 1} \frac{z-1}{z-1}$
'pid'	PID	Proportional, integral, and derivative	$K_p + \frac{K_i}{s} + K_d s$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{z-1}{T_s}$
'pidf'	PIDF	Proportional, integral, and derivative with first-order filter on derivative term	$K_p + \frac{K_i}{s} + \frac{K_d s}{T_f s + 1}$	$K_p + K_i \frac{T_s}{z-1} + K_d \frac{1}{T_f s + 1} \frac{z-1}{z-1}$

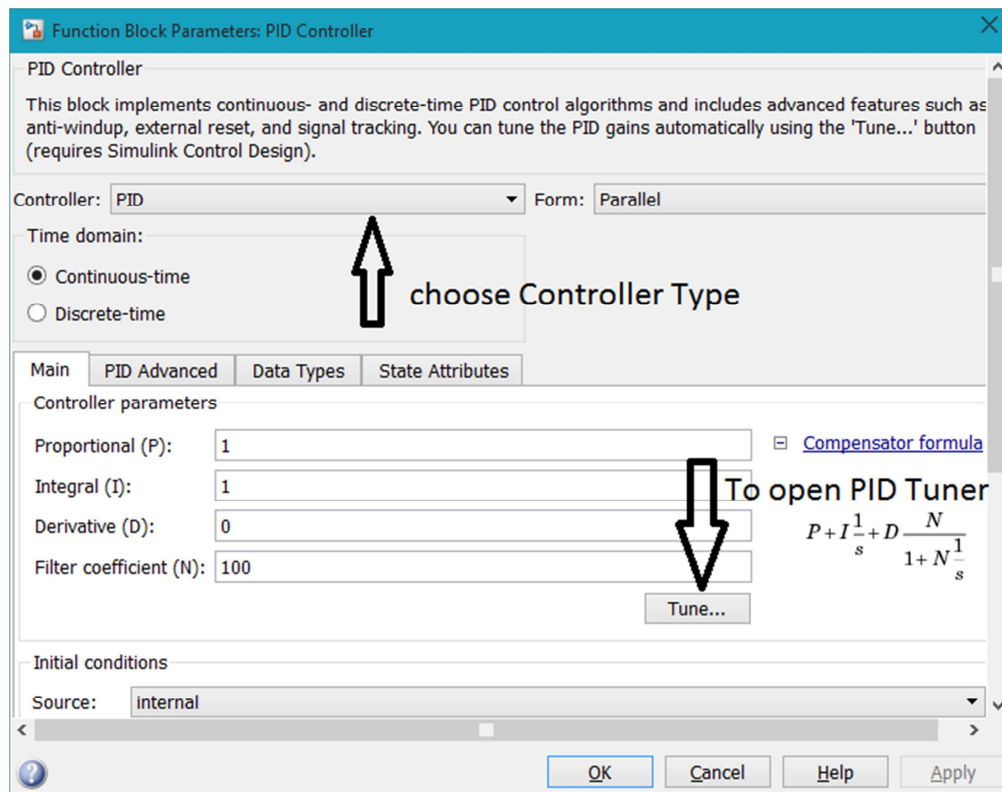
2) SIMULINK

Select the PID controller block form Simulink Library



Drag the PID controller and place in the Simulink model , and double click on block.

Experiment 6: PID Controller



Example 1:

This example shows how to use the PID tuner to design a controller for the plant

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

1. Create the plant model and open the PID Tuner to design a PI controller for a first pass design.

```
>> G=zpk([], [-1 -1 -1], 1)

G =

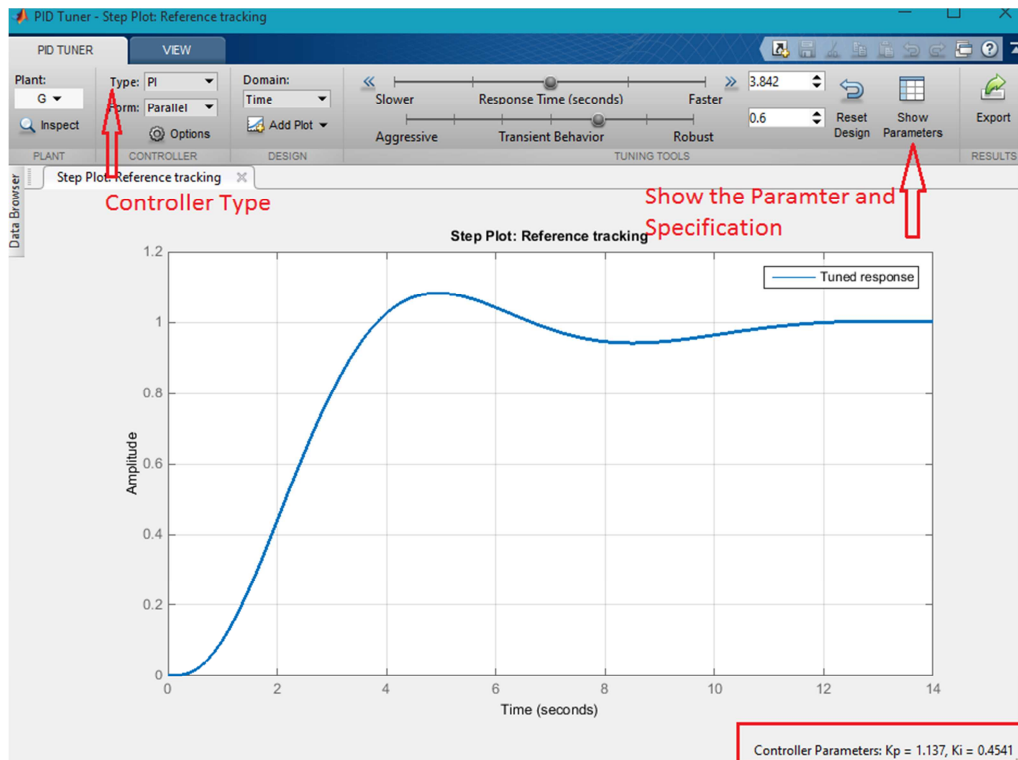
    1
-----
(s+1)^3

Continuous-time zero/pole/gain model.
```

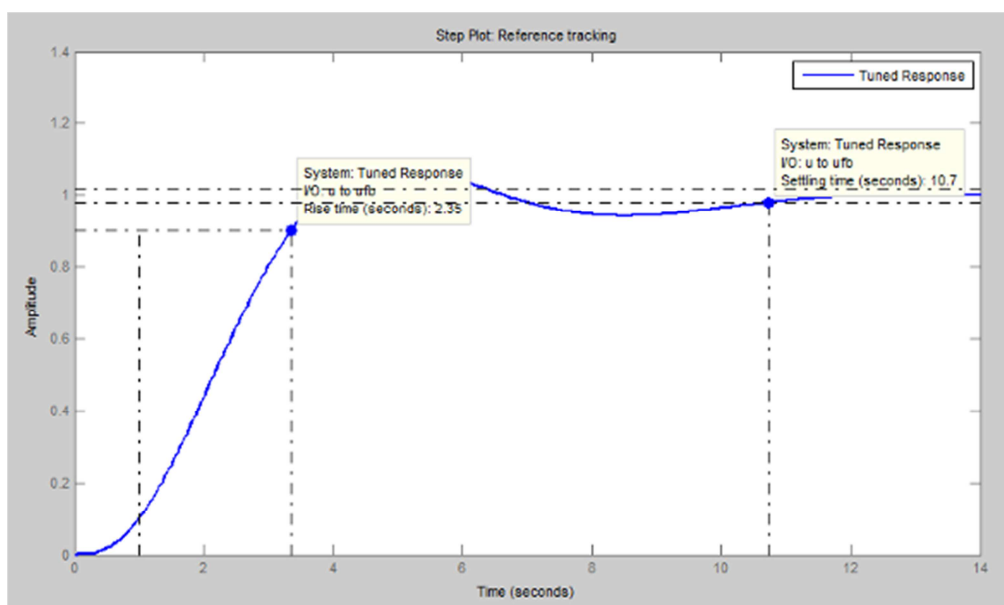
```
Command Window
fx >> pidTuner(G)
```

2. The PIDTuner toolbox

Experiment 6: PID Controller

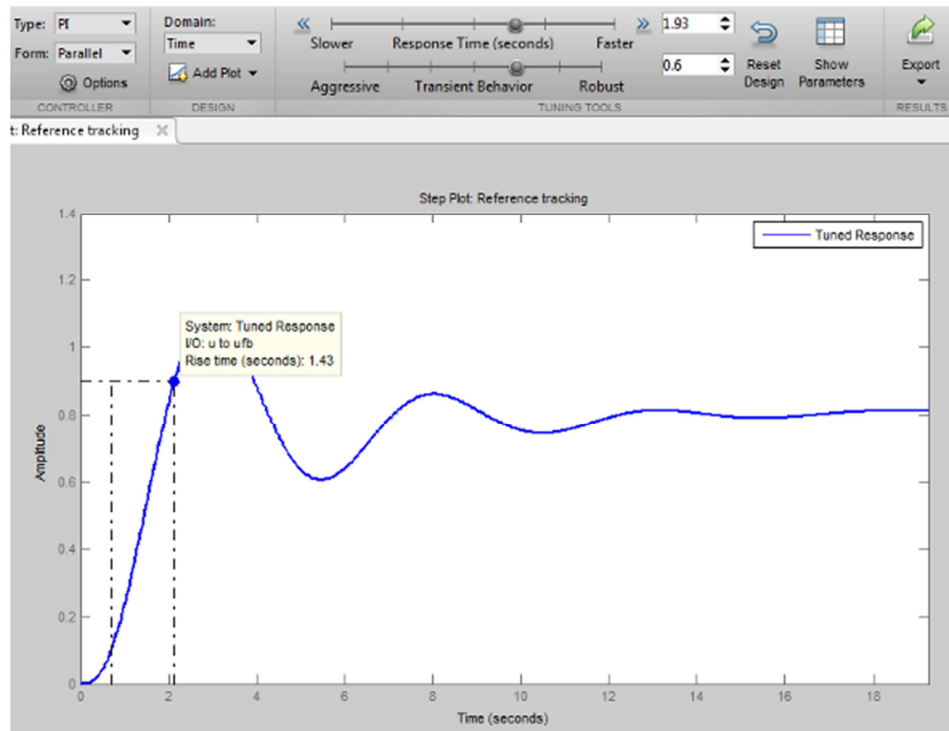


3. Examine the reference tracking rise time and settling time.
4. Right-click on the plot and select Characteristics > Rise Time to mark the rise time as a blue dot on the plot. Select Characteristics > Settling Time to mark the settling time. To see tool-tips with numerical values, click each of the blue dots.



5. Slide the Response time slider to the right to try to improve the loop performance. The response plot automatically updates with the new design

Experiment 6: PID Controller



Moving the Response time slider far enough to meet the rise time requirement of less than 1.5 s results in more oscillation. Additionally, the parameters display shows that the new response has an unacceptably long settling time.

The 'Controller parameters' dialog box shows the following data:

Controller parameters	
	Tuned
Kp	2.9965
Ki	0.054468
Kd	
Tf	

Performance and robustness	
	Tuned
Rise time	1.43 seconds
Settling time	182 seconds
Overshoot	9.42 %
Peak	1.09
Gain margin	8.37 dB @ 1.72 rad/s
Phase margin	40.7 deg @ 1.04 rad/s
Closed-loop stability	Stable

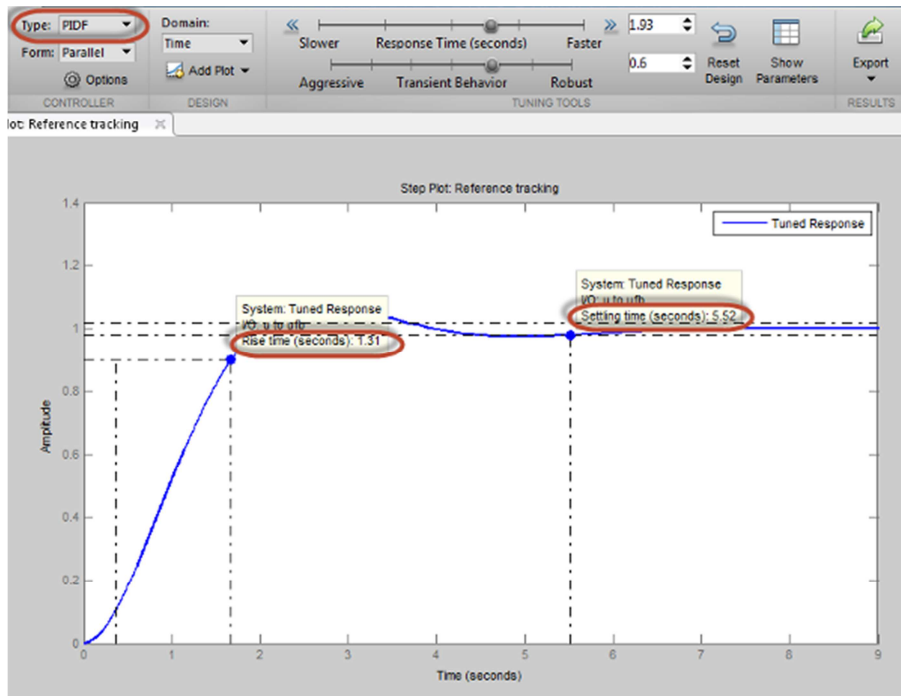
To achieve the faster response speed, the algorithm must sacrifice stability.

6. Change the controller type to improve the response.

Adding derivative action to the controller gives the PID Tuner more freedom to achieve adequate phase margin with the desired response speed.

In the Type menu, select PIDF. The PID Tuner designs a new PIDF controller

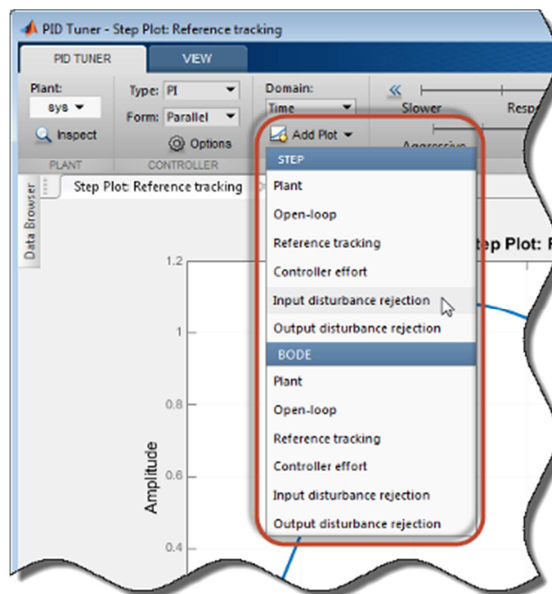
Experiment 6: PID Controller



The rise time and settling time now meet the design requirements. You can use the Response time slider to make further adjustments to the response. To revert to the default automated tuning result, click Reset Design.

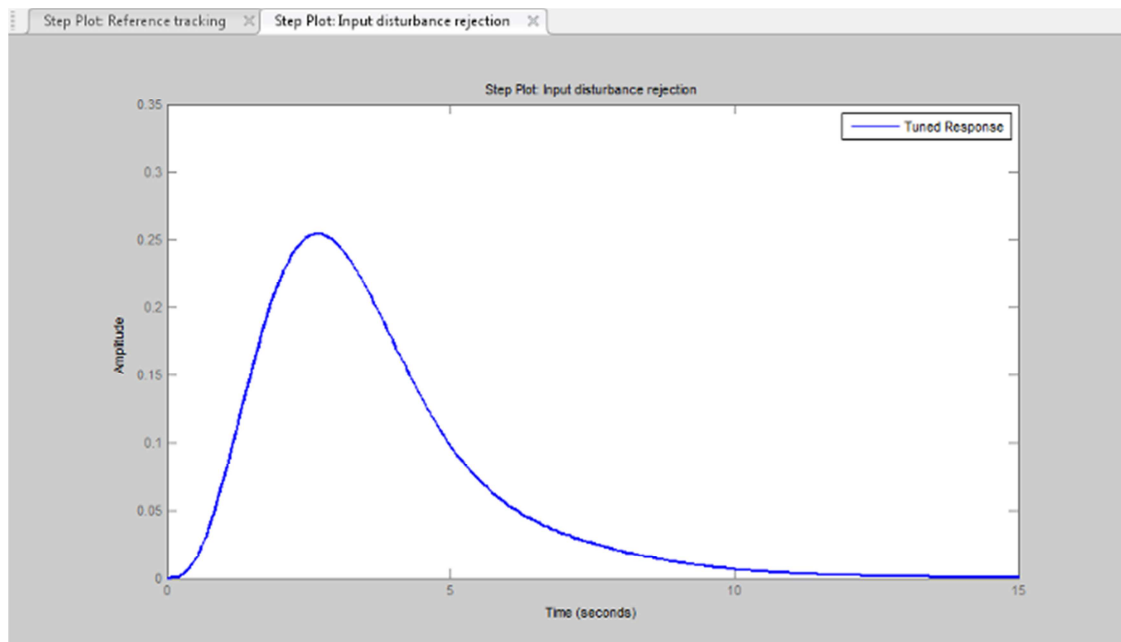
7. Analyze other system responses, if appropriate.

To analyze other system responses, click Add Plot. Select the system response you want to analyze.



For example, to observe the closed-loop step response to disturbance at the plant input, in the Step section of the Add Plot menu, select Input disturbance rejection. The disturbance rejection response appears in a new figure.

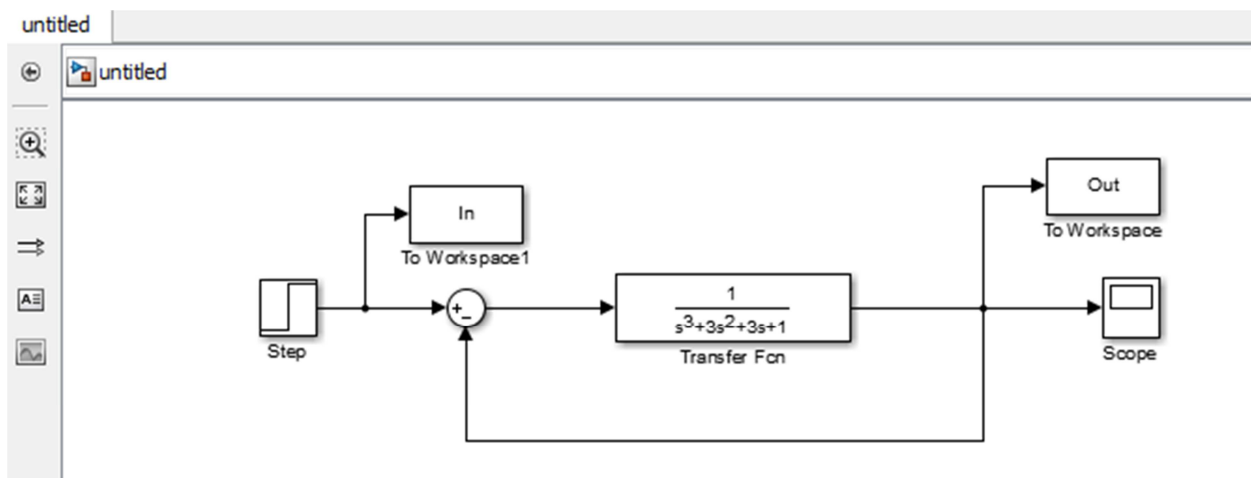
Experiment 6: PID Controller



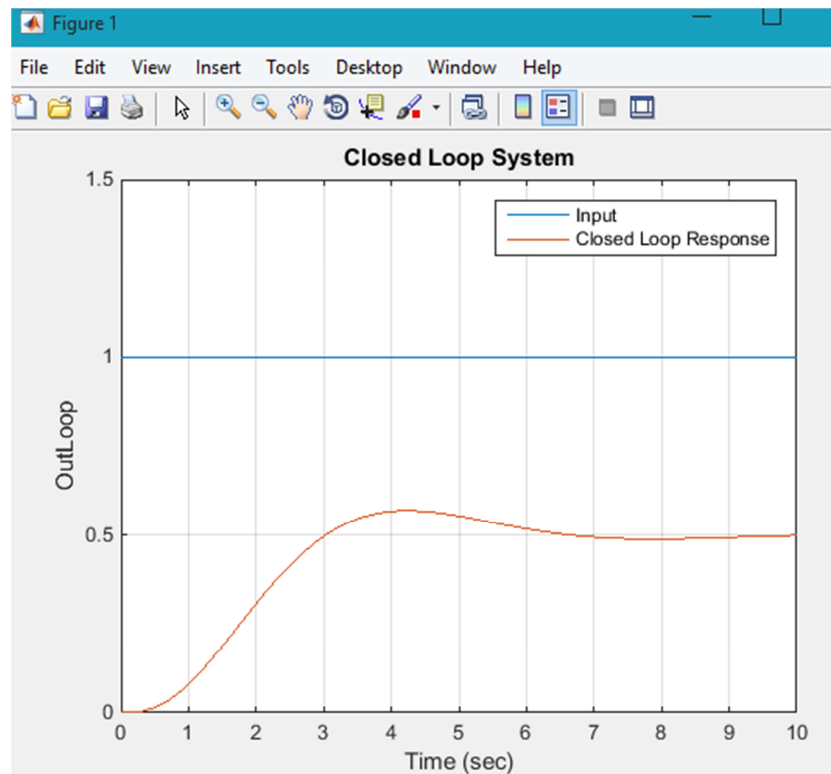
Example 2:

Assume the system in example 1 Bu

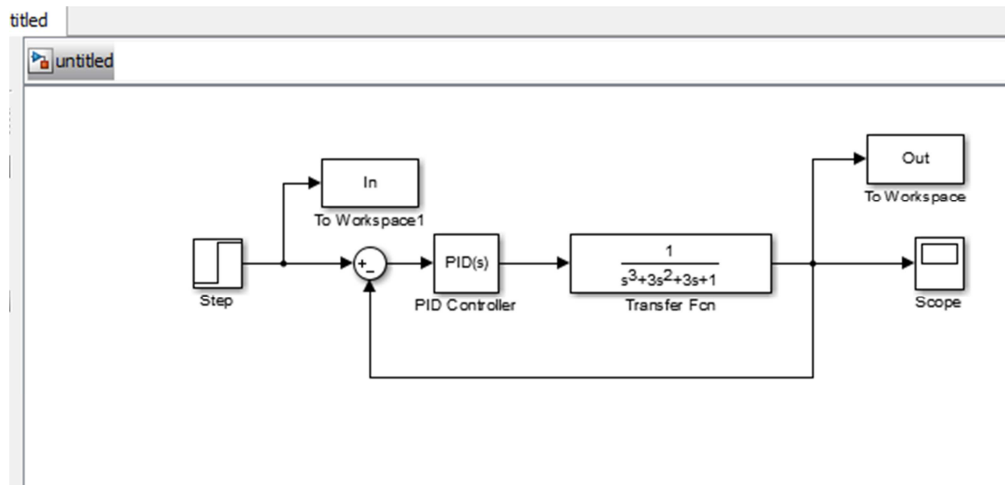
- 1) Build the Simulink block represent the closed loop system



Experiment 6: PID Controller

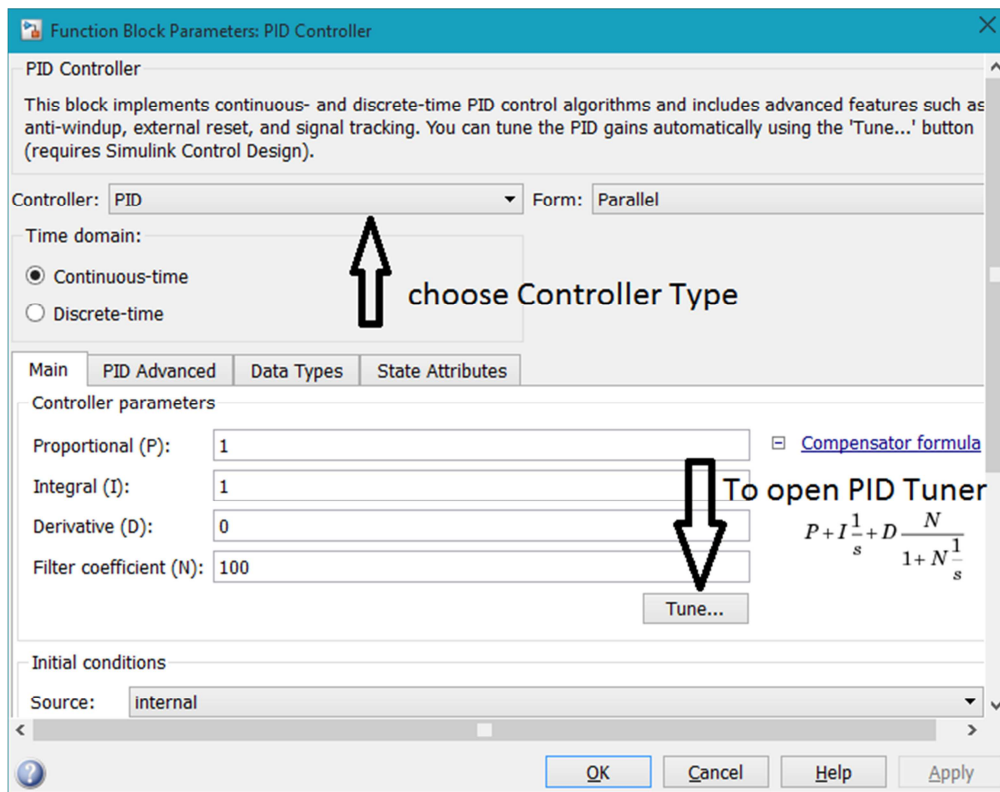


2) Add PID Controller block



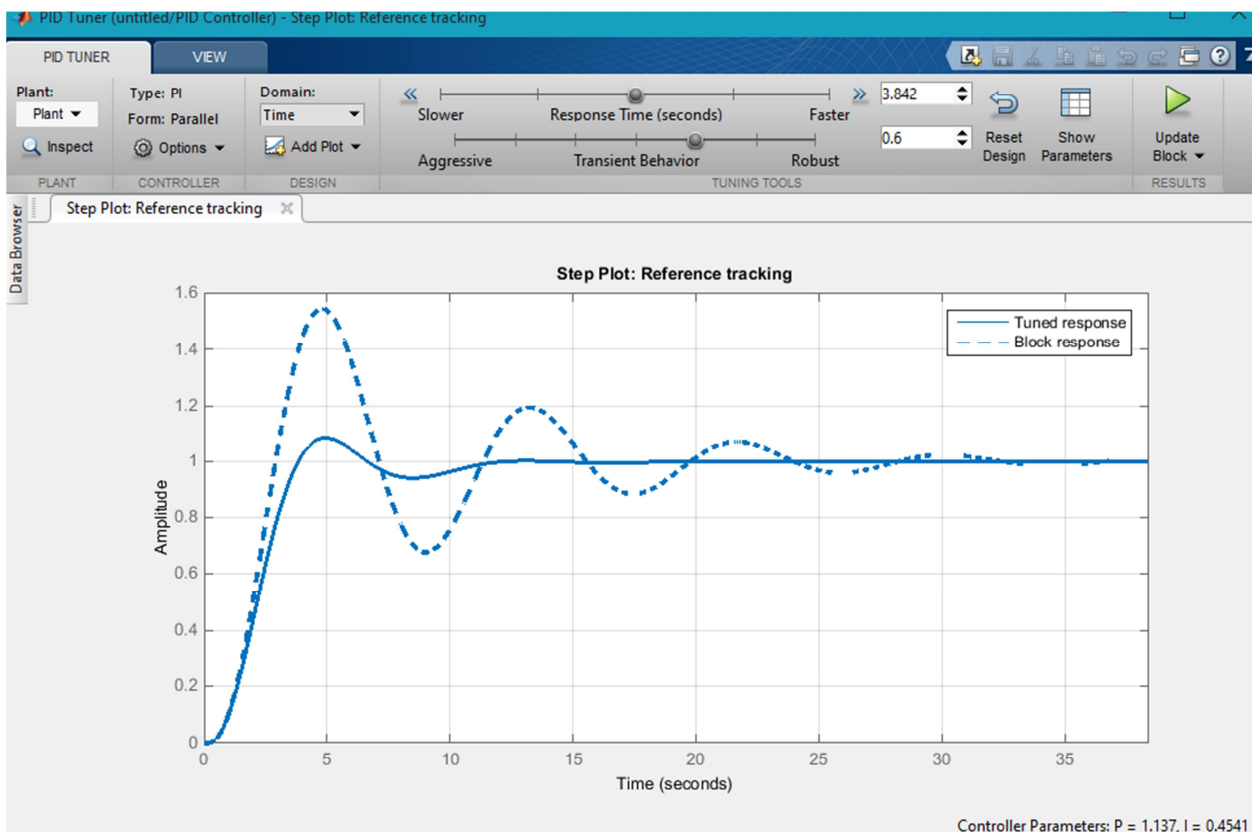
3) Double click on PID controller block

Experiment 6: PID Controller



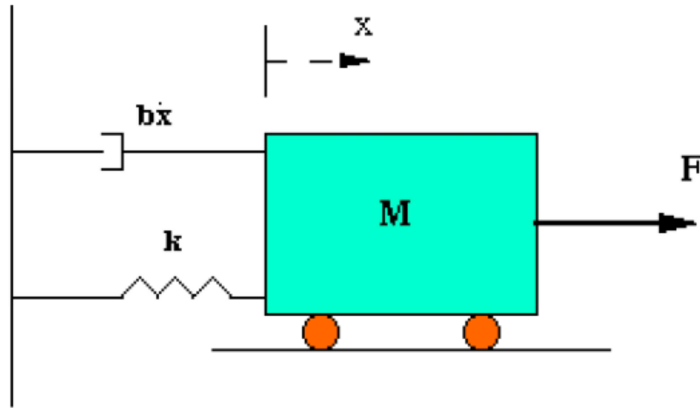
4) Choose the controller type , click apply and click tune

The PID tuner window will appear and can follow the steps as example 1



Exercises:

Exercise-1: Suppose we have a simple mass, spring, and damper problem.

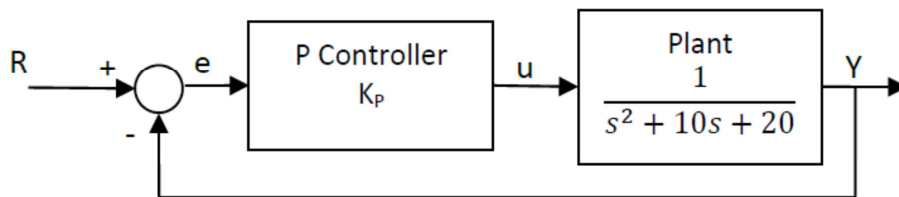


$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2 + bs + k}$$

Let $M = 1\text{kg}$, $b = 10\text{ N.s/m}$, $k = 20\text{ N/m}$, $F(s) = 1$.

The goal of this problem is to show how each of K_p , K_i and K_d affect on the closed loop system.

Add a P-controller ($K_p=300$):

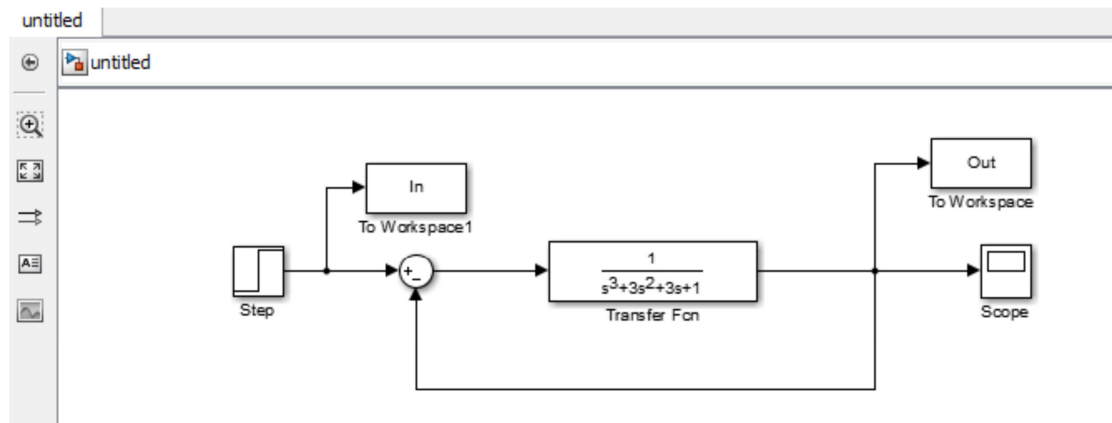


$$\frac{X(s)}{F(s)} = \frac{K_p}{s^2 + 10s + (20 + K_p)}$$

- 1) Add a PD-controller ($K_p=100, K_d=10$).
- 2) Add a PI-controller ($K_p=30, K_i=70$).
- 3) Add a PID-controller ($K_p=350, K_i=300$, and $K_d=50$).

Experiment 6: PID Controller

Exercise-2: for the following closed loop system



- 1) Design a PI Controller to yield a closed-loop step response with overshoot less than 10% and settling time less than 11 sec.
- 2) Design a PD Controller to yield a closed-loop step response with overshoot less than 20% and settling time less than 3 sec.
- 3) Design a PID Controller to yield a closed-loop step response with overshoot less than 5% and settling time less than 5 sec.

**Experiment
Seven**

DC Servo

**Control Laboratory
MX-0908453**

Mechatronics Eng. Dept.

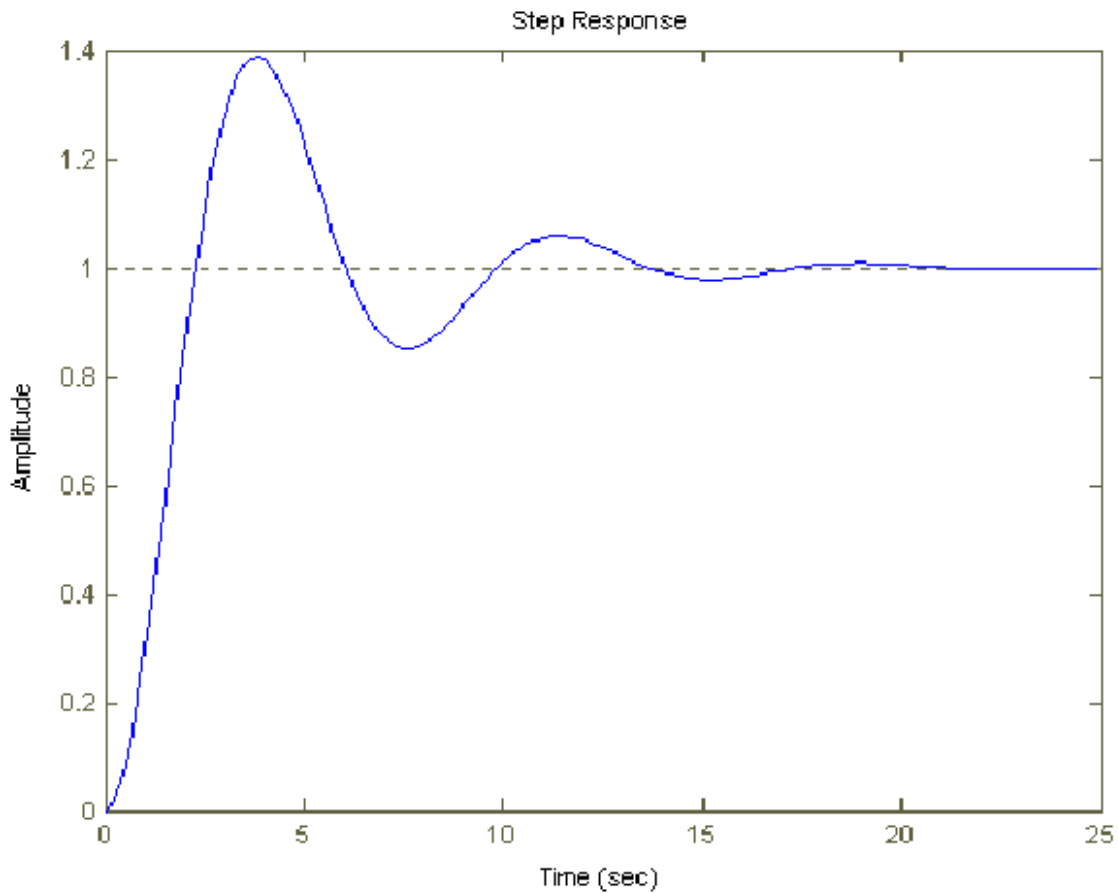


OBJECTIVES:

The aim of this experiment is to provide students with a sound introduction to the principles of analogue servomechanisms, and by extension to those of closed-loop systems more generally.

PRE-LAB:

1. In this experiment, what type of DC motors is used?
2. Draw the speed torque characteristics for DC motor, what is the equation that performs the relationship between torque and speed?
3. From the following step response for a second order system, determine the following:
 - What is the behavior of this system?
 - What is the supposed valu for ζ ?
 - The overshoot.
 - The settling time.
 - The rise time.



EQUIPMENT and APPARATUS:

- 1. Power supply.
- 3. Feedback, 33-002 Servo Fundamentals Trainer, which consists of the following:
 - Mechanical Unit 33-100
 - Analogue Unit 33-110

Mechanical Unit 33-100

Contains a **power amplifier** to drive the motor from an analogue or switched input. The motor drives the output shaft through a **32:1 belt reduction**. The motor shaft also carries a **magnetic brake disc** and an **analogue speed transducer (tachogenerator)**. A two-phase pulse train for digital speed and direction sensing is also derived from tracks on the brake disc.

The output shaft carries **analogue (potentiometer) and digital (64 location Gray code) angle transducers**.

The unit contains a simple signal generator to provide low frequency test signals; sine, square and triangular waves, and requires an external power supply providing:

- +15 V, 0, .15 V at 1.5 A
- +5 V, 0, at 0.5 A

The Mechanical Part

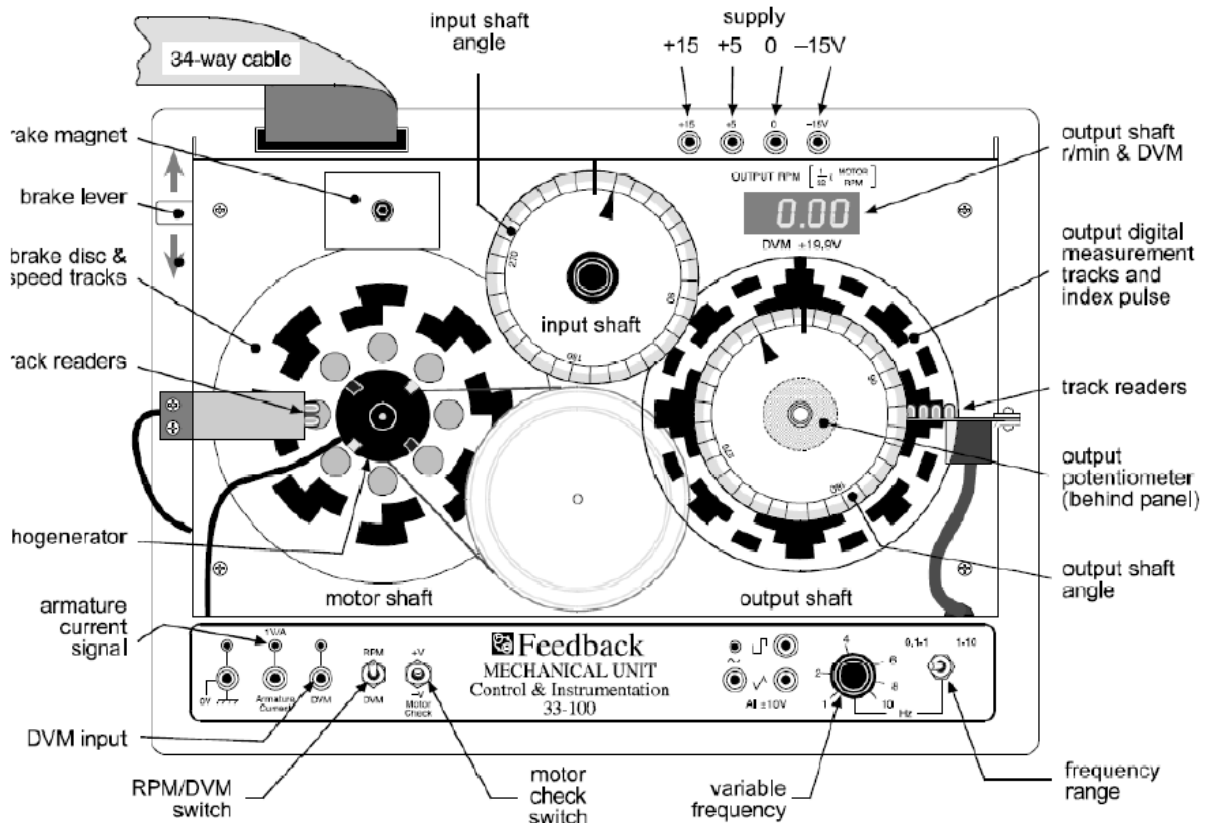


Figure 1. Feedback Mechanical unit 33-100.

Motor shaft This carries the brake disc, together with a 2-phase speed track and tachogenerator.

Experiment 7: DC Servo

Brake disc and magnet	The brake is applied by the lever projecting at the left. The lever scale is provided to enable settings to be repeated.
Speed tracks and readers	These provide two-phase, 0-5 V square waves at 8 cycles per revolution. These signals are available on the 34-way socket but are not used in the Analogue system.
Motor check switch	This enables the motor to be rotated as an initial check.
Armature current signal	This is a voltage waveform indicating the armature current with scale of 1 V/A.
Input shaft	This carries the input potentiometer and scale and gives a signal i in the range ± 10 V.
Test signal frequency range switch	These control the internal oscillator to provide ± 10 V square, and triangular and sine waveforms with nominal frequency 0.1 to 10 Hz in two ranges. The square and triangular waveforms are connected to the 34-way socket.
Output shaft	This carries the output potentiometer and digital angular measurement tracks. The potentiometer provides o in the range ± 10 V.
Digital measurement and information and are read by	The digital tracks give 6 bit Gray code (64 locations) Readers infra-red readers. The 6-bit information is supplied as 0 or 5 V to six pins on the 34-way socket.
Index pulse	At one pulse per revolution this provides an output shaft reference point for incremental control connected to a pin on the 34-way socket.
Output speed display	This provides a direct reading of output shaft speed in r/min in the range 00.0 to 99.9, derived from the tachogenerator. Since the reduction ratio is 32:1, a motor speed of 1000 r/min gives 31.1 r/min at the output shaft.

The Analogue Part

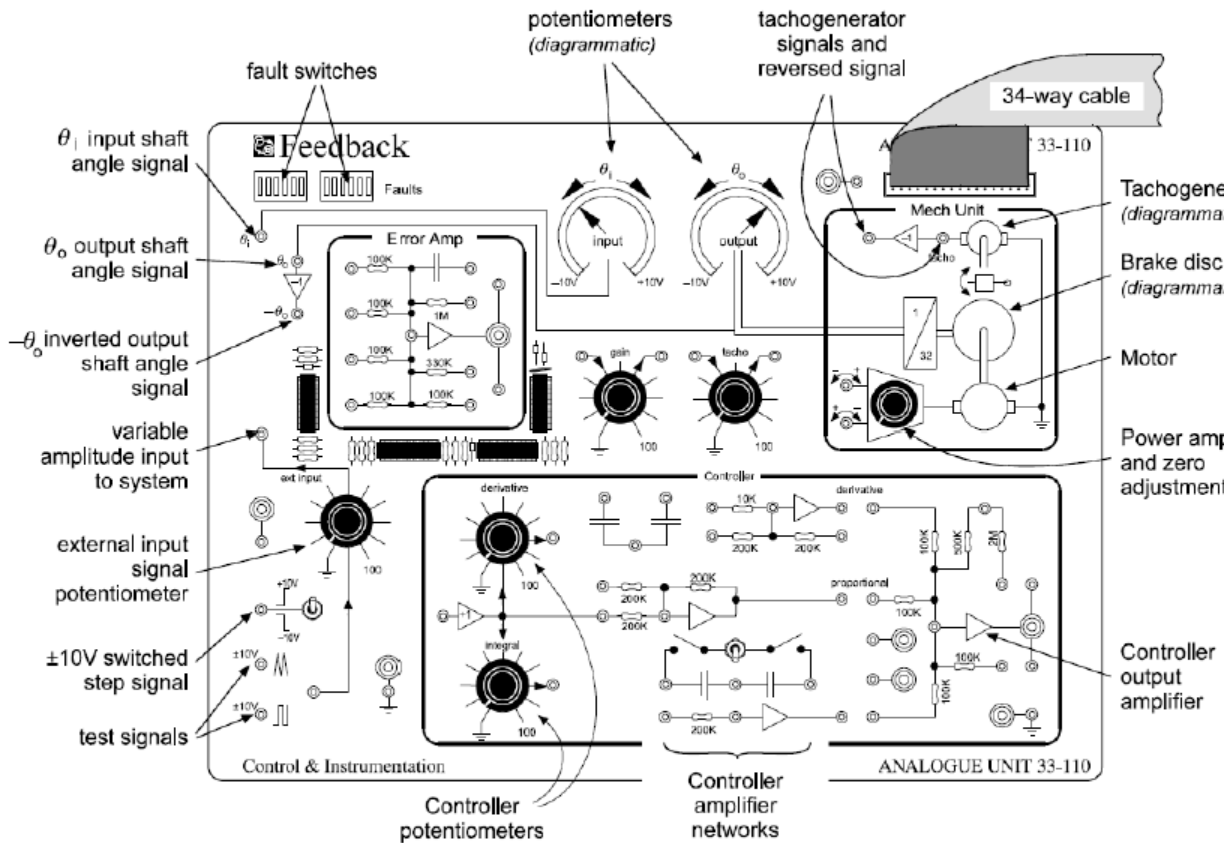


Figure 2. Feedback Analogue Unit 33-110.

Figure 2 shows the general arrangement of the panel, interconnections are made by 2 mm plug leads and there are a few 4 mm sockets for conversion or oscilloscope connections.

<p>Upper portion of panel From left to right θ_1, θ_0</p>	<p>These sockets give the voltage signals from the input and output shaft potentiometers. These are represented diagrammatically in the centre of the panel, the potentiometers themselves being in the Mechanical Unit.</p>
<p>- θ_0</p>	<p>This socket provides a reversed output shaft signal required for certain applications.</p>
<p>Fault switches</p>	<p>These enable faults to be introduced. For normal (no fault) operation all switches should be down.</p>
<p>Error Amplifier</p>	<p>This is used to combine potentiometer signals to provide the error.</p>
<p>Potentiometers P1 and P2</p>	<p>These provide system gain control and tachogenerator signal adjustment.</p>
<p>Power amplifier</p>	<p>This drives the motor. The two inputs drive the motor in opposite</p>

Experiment 7: DC Servo

directions for a given input. The zero adjustment enables the motor to be rotated with no amplifier input.

Motor	This is in the Mechanical Unit and drives the brake disc and tachogenerator directly, and the output shaft through a 32:1 belt reduction.
Brake disc and magnet	These are in the Mechanical Unit and provide an adjustable load for the motor.
Tachogenerator	This is mounted on the motor shaft and provides a voltage proportional to motor speed; the voltage is available with reversed polarity.
Lower portion of panel to right ± 10 V step	This enables a manually switched 10 V step input to be from left obtained.
Test signals	These sockets provide ± 10 V low frequency (nominally 0.1 to 10 Hz) square and triangle waveforms. The frequency control and range switch are on the Mechanical Unit. A sine wave test input is available from the Mechanical Unit.
External input potentiometer P3	This can be linked to any input to provide an adjustable input to the error amplifier.
Controller	This contains operational amplifiers with associated networks to enable various compensating and control circuits to be introduced to improve the performance of a basic system

Introduction:

The servo is an automatic electromechanical device that uses error-sensing feedback to correct the performance of a mechanism. The term applies only to systems where the feedback or error-correction signals help control mechanical position or other parameters.

A common type of servomotors provides position control. Servos are commonly electrical or partially electronic in nature, using an electric motor as the primary means of creating mechanical force. Other types of servos use hydraulics, pneumatics, or magnetic principles. Usually, servos operate on the principle of negative feedback, where the control input is compared to the actual position of the mechanical system as measured by some sort of transducer at the output. Any difference between the actual and wanted values (an "error signal") is amplified and used to drive the system in the direction necessary to reduce or eliminate the error. An entire science known as control theory has been developed on this type of system.

The Closed-Loop Control System

The difference or error signal may be thought of as producing effects which move forward, from the point of comparison to the resulting action. The comparison itself depends on a signal which is fed back from the output of the process to be compared with the reference or input signal. The forward flow and feedback of signals form a loop around which information flows, see Figure 3. Such a system is therefore called a closed-loop system.

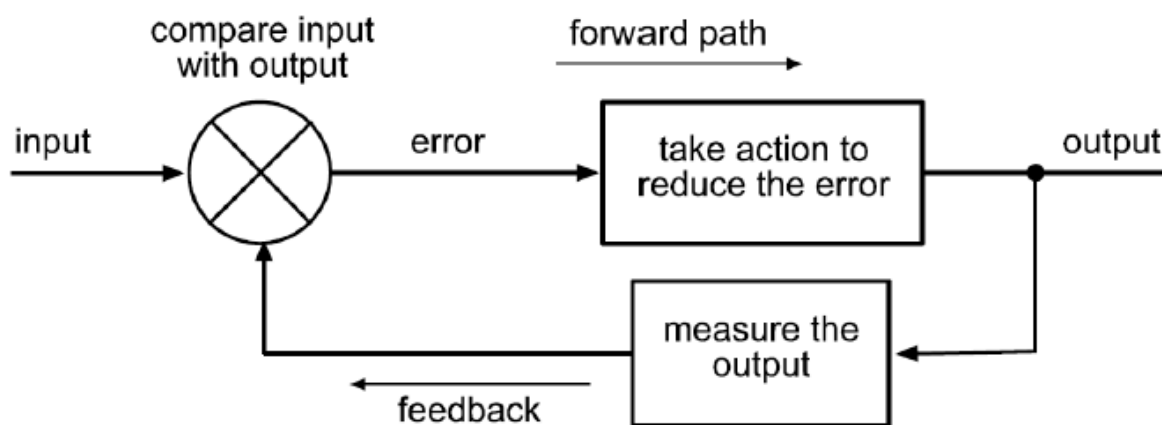


Figure 3. The Closed Control Loop.

Various names are given to the signals in different industrial or other contexts, but the meanings of words in any one of the columns below are much the same:

Difference	Output	Input
Error	Actual value	Reference value
Deviation	Measured value	Set value
	Controlled quantity	Set point
		Desired value
		Demanded value

Experiment 7: DC Servo

Where the system is electrical, the state will normally be represented by signals expressed in volts; in the strip that being rolled in a steel mill it might be, for the width, a signal representing ten inches per volt.

The difference in the comparison will be called the **error signal** and the part of the system that carries out the comparison is the **error channel**.

There is usually a power amplifying device to drive the **Actuator** (which in Figure 4 is the geared motor).

It is usual for control engineers to describe their systems in a block diagram form. The block diagram below describes the type of system we shall be using in this experiment.

Here there is a comparison by the error channel of the input and output, the error is then amplified to drive a motor and gearing in the forward path so that the speed or position of the output shaft can be modified.

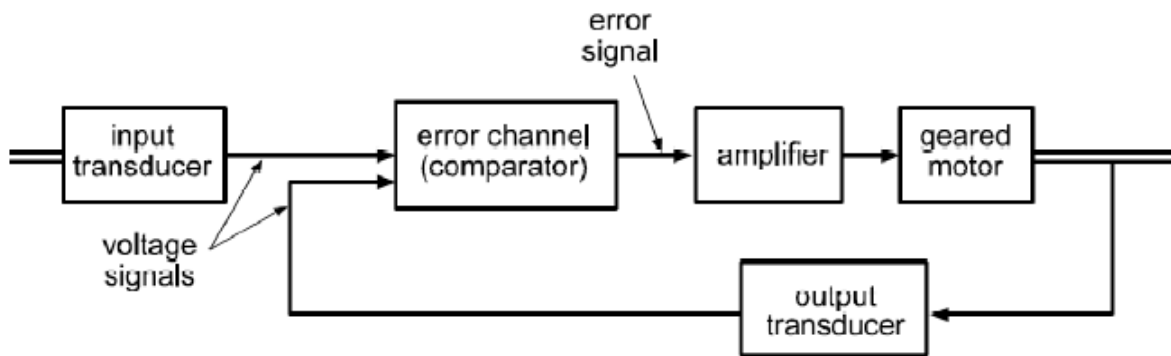


Figure 4. Block Diagram of an Analogue Closed-Loop System.

Motor, Tachogenerator and Brake Characteristics

The motor is a **permanent magnet type** and can be represented in idealized form as in Figure 5, where R_a is the armature resistance and T1, T2 are the actual motor terminals.

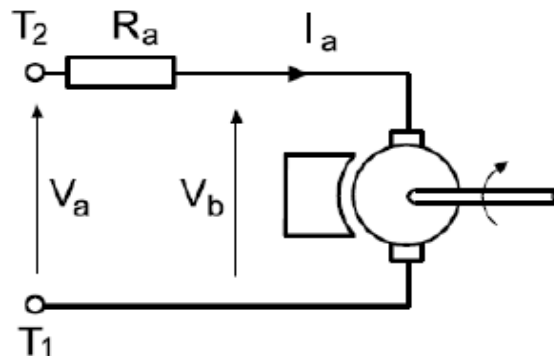


Figure 5. Representation of a Motor in terms of an Ideal Motor.

If the motor is stationary and a voltage V_a is applied, a current I_a flows which causes the motor to rotate. As the motor rotates a back emf V_b is generated. As the motor speeds up, the back emf increases and I_a falls. In an ideal (loss free) motor, the armature current falls to substantially zero and V_b approximately equals V_a . Thus if V_a is varied slowly in either polarity, the motor speed is proportional to V_a , and a plot of motor speed against V_a would have the form of Figure 6.

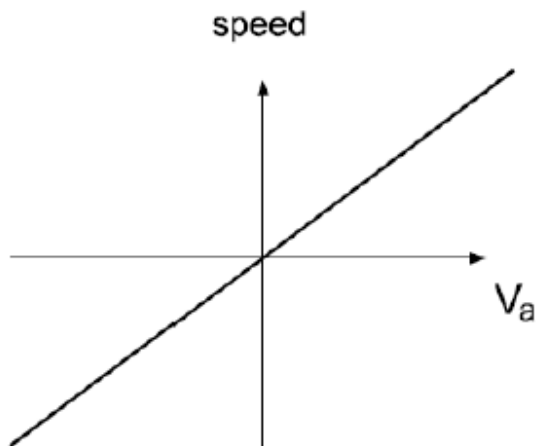


Figure 6. DC Motor characteristics.

In the 33-100 the armature voltage V_a is provided by a **power amplifier**. A power amplifier is necessary, because although the voltages in the error channel may be of the same order as V_a , the motor current may be up to 1 A, while the error channel operates with currents of less than 1 mA and could not drive the motor directly. The amplifier has two input sockets, enabling the motor rotation direction to be reversed for a given input.

The **tachogenerator** is a small permanent magnet machine and hence when rotated produces an emf proportional to speed which can be used as a measure of the rotation speed.

The magnetic brake consists of a permanent magnet which can be swung over an aluminum disc. When the disc is rotated eddy currents circulate in the area of the disc within the magnet gap, and these react with the magnet field to produce a torque which opposes rotation. This gives an adjustable torque speed relation of the form of Figure 7, and provides a very convenient load for the motor.

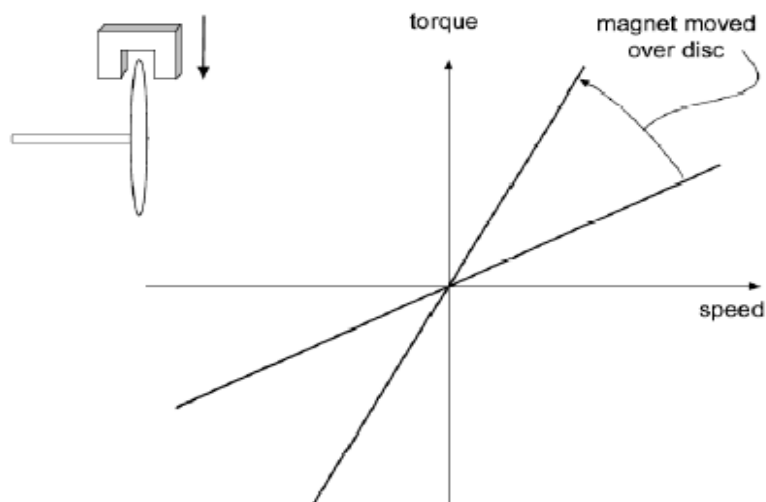


Figure 7. Characteristic of Magnetic Brake.

Experiment 7: DC Servo

The overall characteristics of a motor may be considered from two aspects, both of which can be related to the idealized representation of Figure 8. These aspects are: **Steady-state**, which are concerned with constant or very slowly changing operating conditions, and **transient**, corresponding with sudden changes, both are important in control system applications.

PROCEDURE:

PART I: Steady-State Characteristics

1. In this part, the motor is operated in a range of steady-state conditions.
2. Arrange the system as shown in Figure 8, where P3 enables a voltage in the range ± 10 V to be applied to the power amplifier.

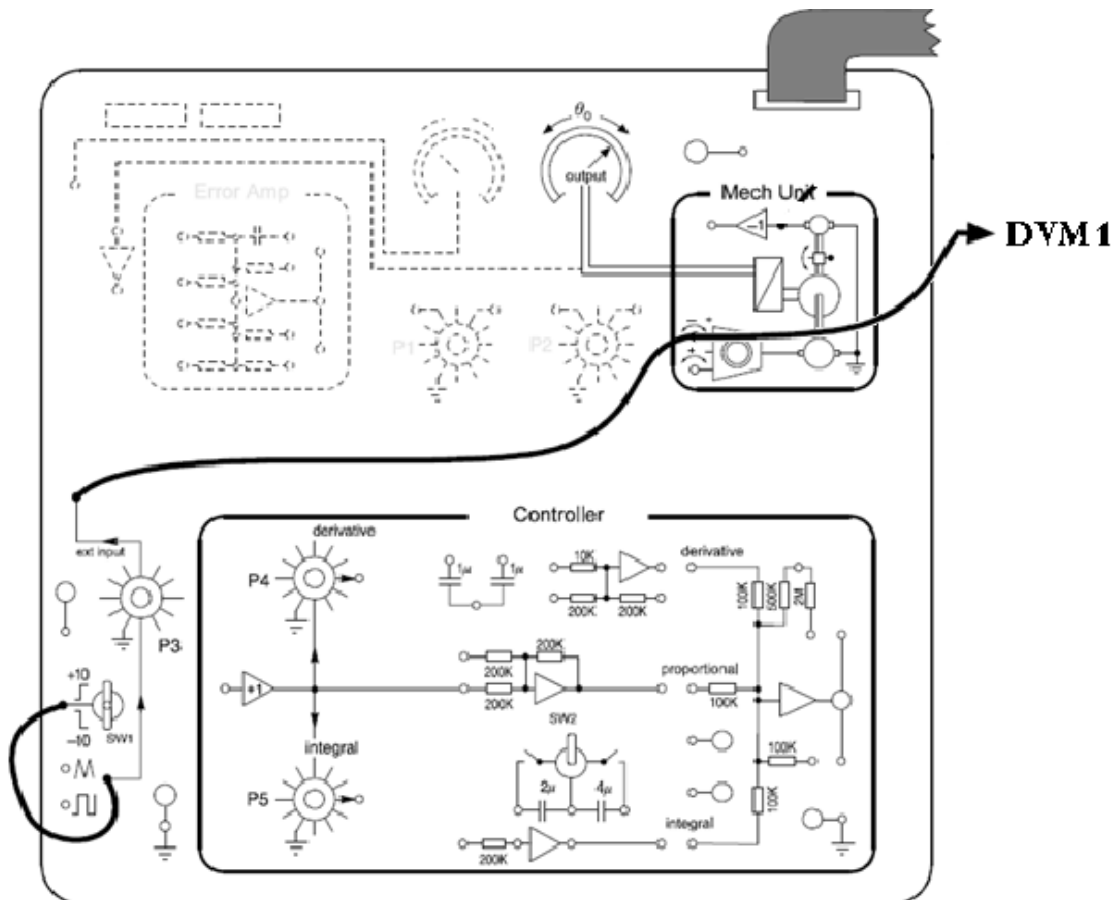


Figure 8. Connections for PART I.

3. Use the DVM on the 33-100 for voltage measurements. For each measurement set up the required steady state then switch between DVM and RPM.
4. By setting SW1 and varying P3, fill in the following table, then make a plot of motor speed against amplifier input.

Note:

Since the reduction to the output shaft is 32:1, the motor speed is calculated by multiplying the r/min reading by 32; eg, a reading of 31.25 = a motor speed of 1000 r/min.

Experiment 7: DC Servo

Amplifier Input (Volts)	Output Shaft Speed (RPM)	Motor Speed (RPM)
+10		
+9		
+7.5		
+5		
+2.5		
+1		
0		
-1		
-2.5		
-5		
-7.5		
-9		
-10		

Brake Load

Considering the idealized motor shown in Figure 9(a), when the motor is unloaded the back emf V_b substantially equals the applied voltage V_a , the armature current being very small. When the motor is loaded the speed falls, the back emf falls, and the armature current increases and the voltage drop in the armature resistance $V_r (= I_a R_a)$ added to V_b matches V_a , that is:

$$\begin{aligned} V_a &= V_r + V_b \\ &= I_a R_a + V_b \end{aligned}$$

Hence, if the motor is loaded so that the speed falls, the armature current increases, the general characteristic being as the solid lines in Figure 9(b). If the armature resistance is low, which is the situation for a normal motor, the current increases greatly, as shown dotted, for a small change in speed. The proper operating range of the motor would be up to a load corresponding with a few percent drop in speed, perhaps to the point when the dotted current line crosses the speed line.

1. Adjust P3 to set the motor speed to 2000 r/min (62.5 r/min at output), with the brake fully upwards. Connect the DVM to the Armature Current (1 volt/amp) output on the Mechanical Unit.
2. Set the brake to each of its six positions in turn and for each setting record the speed and armature current.

Experiment 7: DC Servo

Brake Setting	Armature current I_a (Amps)	Output Shaft Speed (RPM)	Motor Speed (RPM)
0			
1			
2			
3			
4			
5			
6			

3. Plot the speed and armature current against brake setting, the plot should have the general form of Figure 9(c).

4. Comment on your results.

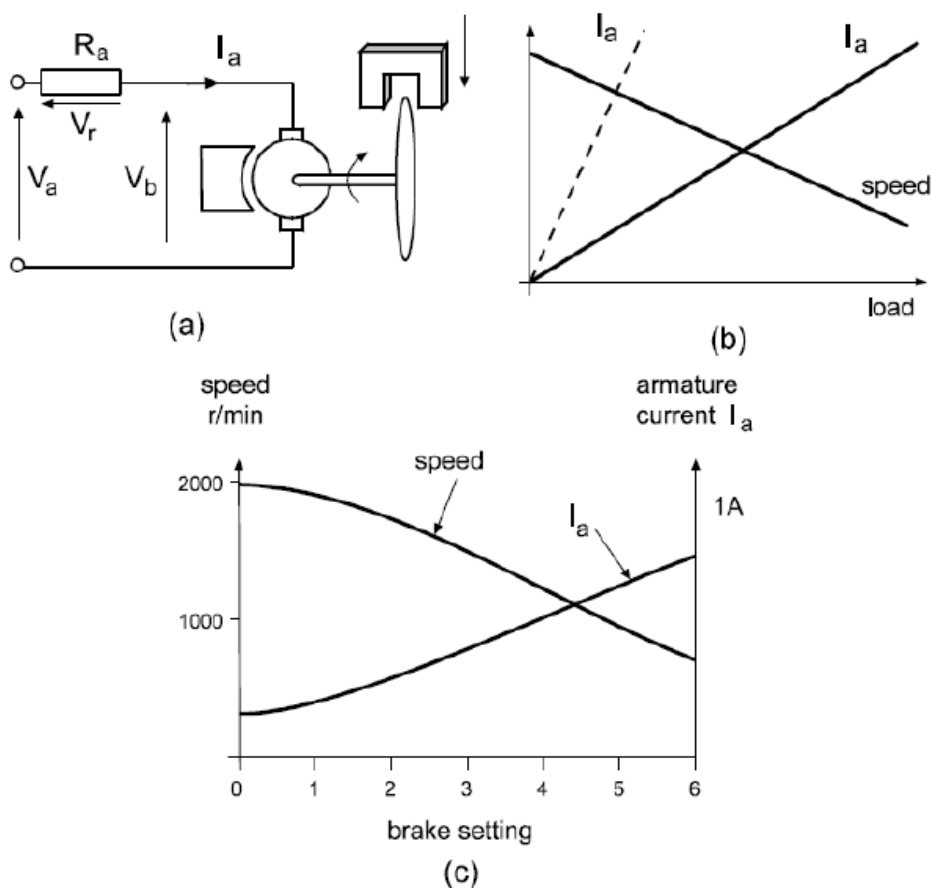


Figure 9. Motor Characteristics Related to Load.

PART II: Speed Control System

An important aspect of closed-loop control is speed control, which has many industrial applications, varying from heavy industrial, such as paper mills or steel rolling mills, to tape or video transport mechanisms.

The essential principle of closed-loop speed control is similar to position control, except that the feedback signal is an output velocity signal V_s , normally from a tachogenerator, which is compared with a reference voltage V_r to give an error

$$V_e = V_r - V_s$$

In operation the reference is set to a required value, which drives the motor to generate V_s , which reduces the error until the system reaches a steady speed.

If the motor is loaded, e.g. with the magnetic brake on the 33-100, the speed falls; this tends to increase the error, increasing the motor drive and thus reducing the speed fall for a given load. Note that this implies negative feedback around the loop.

The speed fall with load, sometimes termed **droop** is a very important characteristic in speed control systems.

The rotation direction can be reversed by reversing the reference voltage, though many industrial speed control systems are required to operate in one direction only.

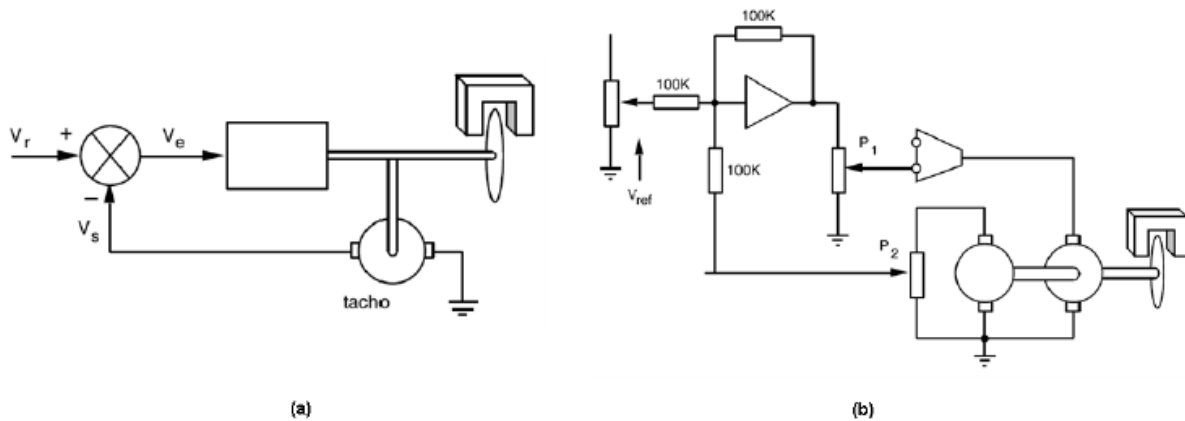


Figure 10. Essential features of a Closed Loop Speed Control.

1. Arrange the system as in Figure 11. Set P2 (tacho) to zero and set the amplifier feedback resistor to 100 k, this gives $G = 1$. Set P1 to 100. Set SW1 up to +10 and adjust P3 to run the motor at 1000 r/min (31.25 r/min at output).

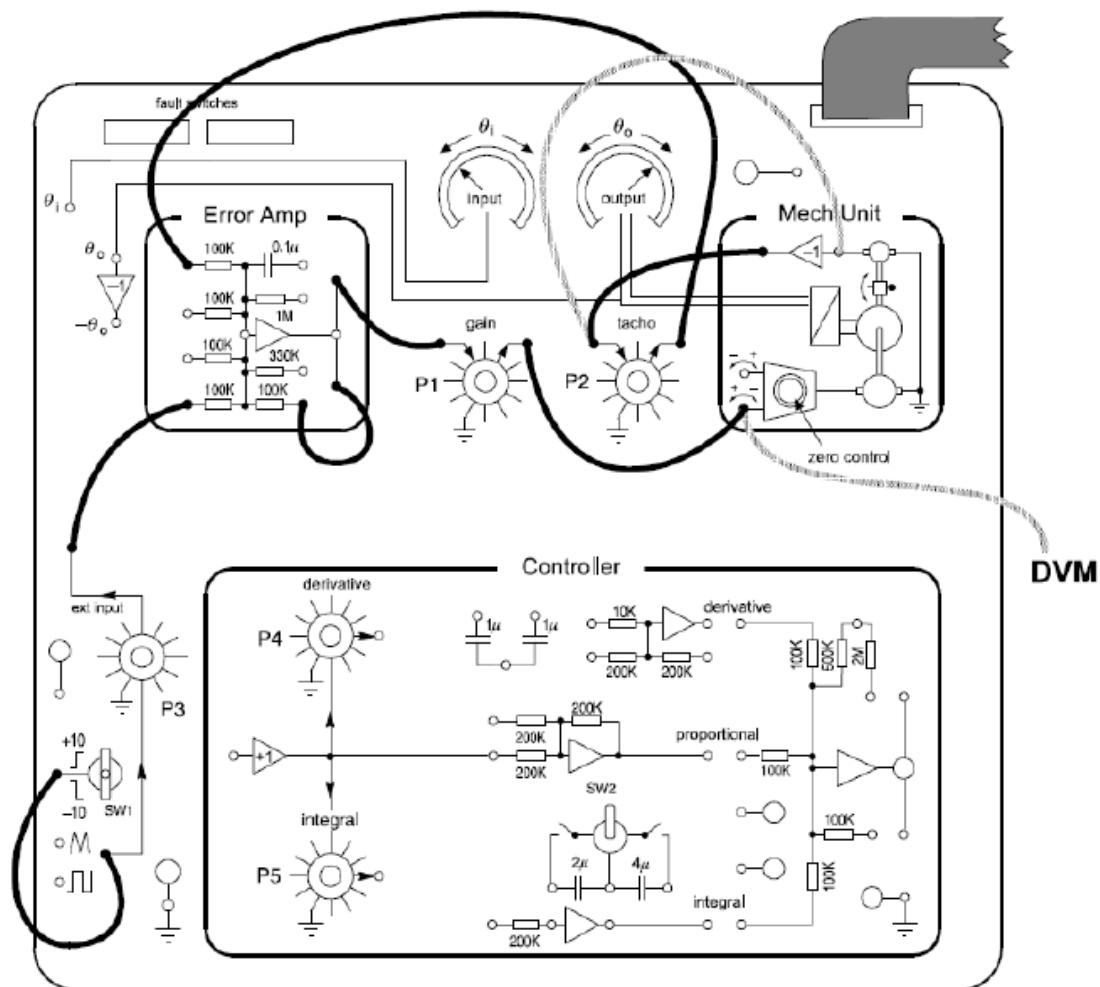


Figure 11. Connections for PART V.

- Turn up P2 slightly, if the speed decreases the loop feedback is negative as required. If the speed increases use the other tachogenerator polarity.

Note that if the system has negative feedback and both the tachogenerator polarity and the power amplifier input are reversed, the system still has negative feedback, but the motor runs in the opposite direction.

- Set P2 to zero and fill in the following table then plot the speed against brake setting to full brake load. The general characteristic should be as in Figure 12.

Experiment 7: DC Servo

Brake Setting	Armature current I_a (Amps)	Output Shaft Speed (RPM)	Motor Speed (RPM)
0			
1			
2			
3			
4			
5			
6			

- Set P2 to 100 and readjust P3 to give 1000 r/min with the brake off.
- Fill in the following table, then plot the speed characteristic and error (Power Amplifier input).

Brake Setting	I_a (Amps)	Output Shaft Speed (RPM)	Motor Speed (RPM)	PA Input (Volts)
0				
1				
2				
3				
4				
5				
6				

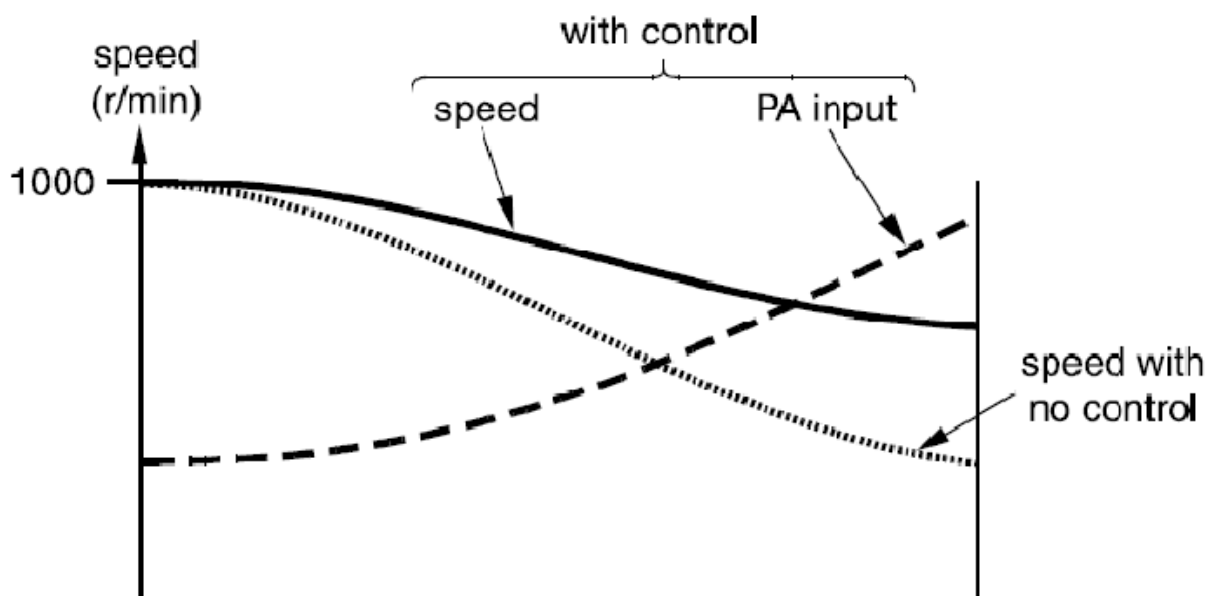


Figure 12. Speed Regulation with and without Closed-Loop Control.

PART III: Position Control System

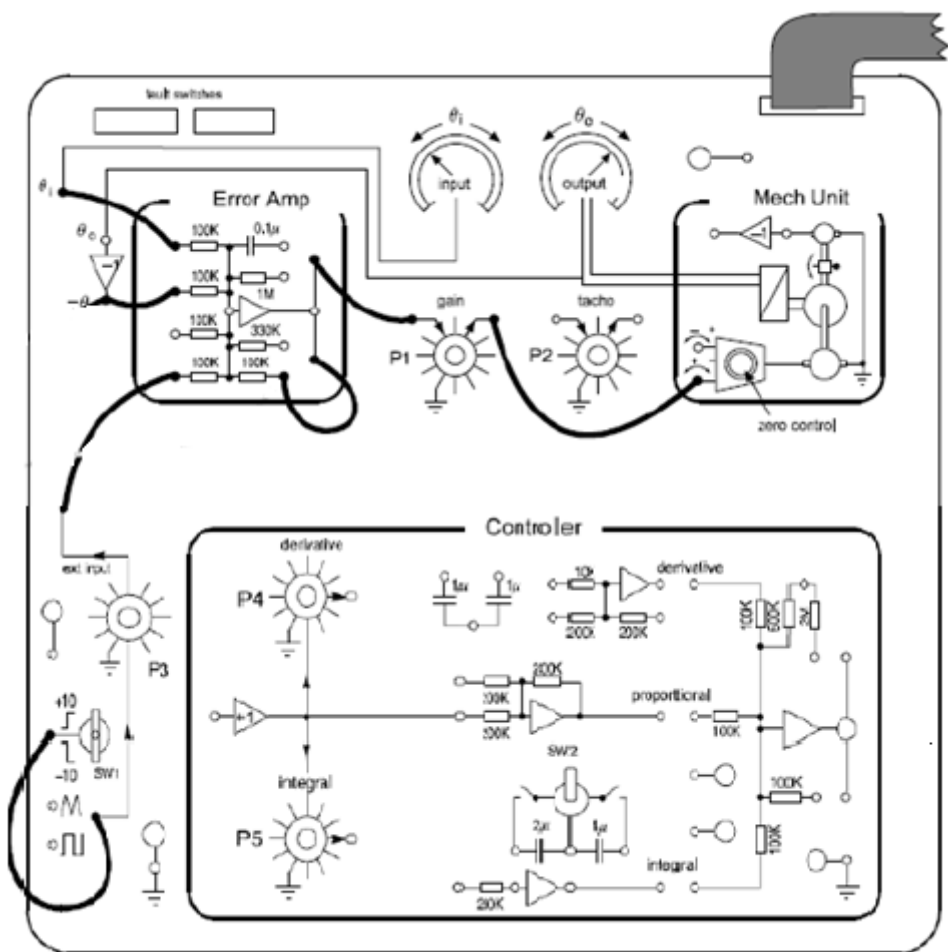


Figure 13. Connection for PART III.

1. Arrange the system with the solid connections of Figure 13, with the error amplifier feedback resistor 100 k, giving $G = 1$.
2. Set the desired angle θ_i at a certain position.
3. Set P1 to zero, and then turn up P1 until the motor just rotates; notice the response.
4. Disconnect the position feedback signal θ_o , and notice the response.
 - **What is the P1 percentage value that makes the system just rotates?**
 - **What is the relationship between the Time to reach steady state (T_a) and the overshoot value?**
 - **Draw both open-loop and closed-loop position responses versus time.**

**Experiment
Eight**

Fuzzy Logic Control

**Control Laboratory
MX-0908453**

Mechatronics Eng. Dept.



Introduction

Traditional logic is based upon the idea that problems can be reduced to a series of statements which are either true or false. However, many everyday situations are not suited to this logical form. Many questions exist where the answer is neither 'yes' nor 'no', but somewhere an in-between answer is required. For example, on a pleasant summer's day, the statement 'the temperature is too high' is neither true nor false. The response to the question requires us to grade the response to indicate that the temperature is neither too hot nor too cold. Common sense tells us that there are grades of meaning or qualified responses to most problems. Philosophers and mathematicians have considered forms of logic for this situation by introducing concepts such as 'vagueness' and multi-valued logic. The topic of fuzzy logic is one way of dealing with things where there is vagueness, by allowing degrees of certainty to be associated with the answer to a question.

Fuzzy Control

The most useful application of fuzzy logic is in the control of events where precise regulation of a process variable is not a primary requirement. As such, the most suitable applications are where there are qualitative requirements for a satisfactory control action. Specifically, these qualitative requirements can be easily stated as fuzzy logic rules and then embedded in a fuzzy logic control algorithm. In this connection, fuzzy logic controllers are widely used to operate the automatic functions of washing machines, video recorders, compact disk players, air conditioning systems, cameras and so on.

It is also possible to use fuzzy logic in industrial feedback control problems that are conventionally solved using experienced human operators who have manual control over a complex process. The procedure followed is to put the operator's control procedure into a fuzzy rule set and hence develop a fuzzy control system. Specifically, the fuzzy logic designer notes the heuristic actions of a human operator when they control a process and writes down the corresponding fuzzy rule. By careful observations of a skilled operator, a complete set of fuzzy rules is obtained which hopefully will reproduce the best performance of the human operator. The result is an 'intelligent' control system which is obtained without reference to control systems theory. This is a simplified view of how a fuzzy controller is prepared, but the basic idea is that intuition and common sense ideas are used.

The intuitive nature of such control systems has a great appeal to many users.

Unfortunately, the set of rules for such a system may be very large indeed and must be carefully checked because human operators are often very subtle in their actions and it can be difficult to translate their nuances into fuzzy logical statements. Depending upon the complexity of the process to be controlled, the construction of the fuzzy rules can be time consuming and involve much fine tuning. The most effective industrial applications have been on processes which are inherently stable and the control actions are for keeping process variables within operational bounds, rather than accurate regulation or servo following

A further popular application is the control of simple loops of the kind usually controlled using three-term (PID) controllers. The use of fuzzy logic here is to emulate the PID action, often with some modifications to accommodate non-linear plant behavior. Figure 1 shows how a fuzzy logic system replaces the conventional controller in this form of application. Note that the fuzzy inference engine in the diagram will consist of a set of fuzzy rules.

Experiment 8: Fuzzy Logic Control

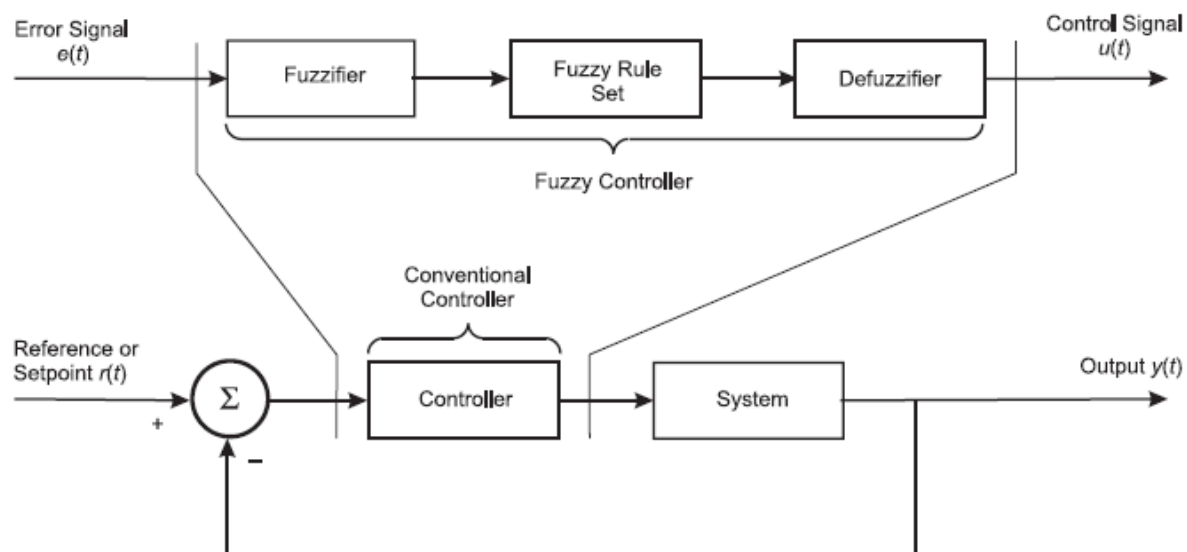


FIGURE 1: FUZZY CONTROLLER

Main Apparatus:

- CE124 Fuzzy Logic Trainer (figure 2).
- CE103 Thermal Control Process apparatus (figure 3).
- CE105 Coupled Tanks apparatus (figure 4).

Experiment 8: Fuzzy Logic Control

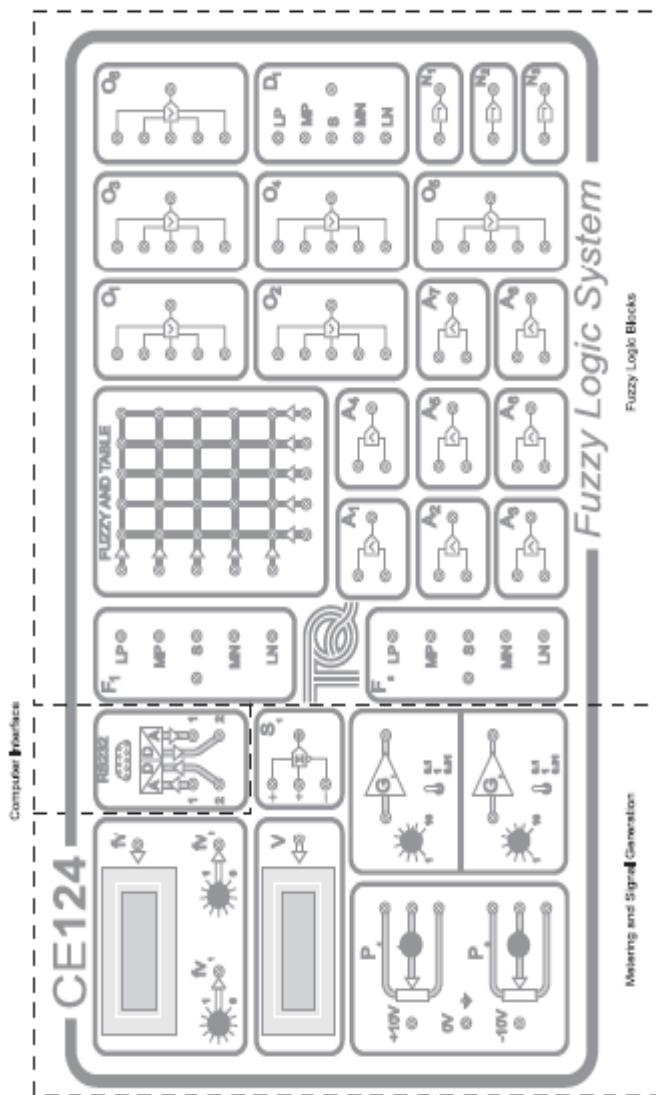


FIGURE 2: CE124 FUZZY LOGIC TRAINER



FIGURE 3: CE103 THERMAL CONTROL PROCESS APPARATUS



FIGURE 4: CE105 COUPLED TANKS APPARATUS

Part 1: Fundamentals of Fuzzy Logic

The object of this part is to investigate the basic principles of fuzzy logic including the following:

- How signals and voltages are converted or classified into fuzzy variables by fuzzifier blocks.
- How fuzzy variables are converted back into real signals by a defuzzifier block.
- The actions of the fuzzy logic operators: AND, OR and NOT.

A. Fuzzy Membership

- Connect the equipment as shown in figure 5.
- With a potentiometer output of -10 V, measure the classifier outputs using the fuzzy variable meter connected to the outputs LP (large positive), MP (medium positive), S (small), MN (medium negative) and LN (large negative).
- Increase the potentiometer output and repeat the above procedure for different values of potentiometer output.
- Record your results in table 1 and draw a block diagram for the experimental setup.

TABLE 1

Experiment 8: Fuzzy Logic Control

Input Voltage V	LP Degree of membership	MP Degree of membership	S Degree of membership	MN Degree of membership	LN Degree of membership

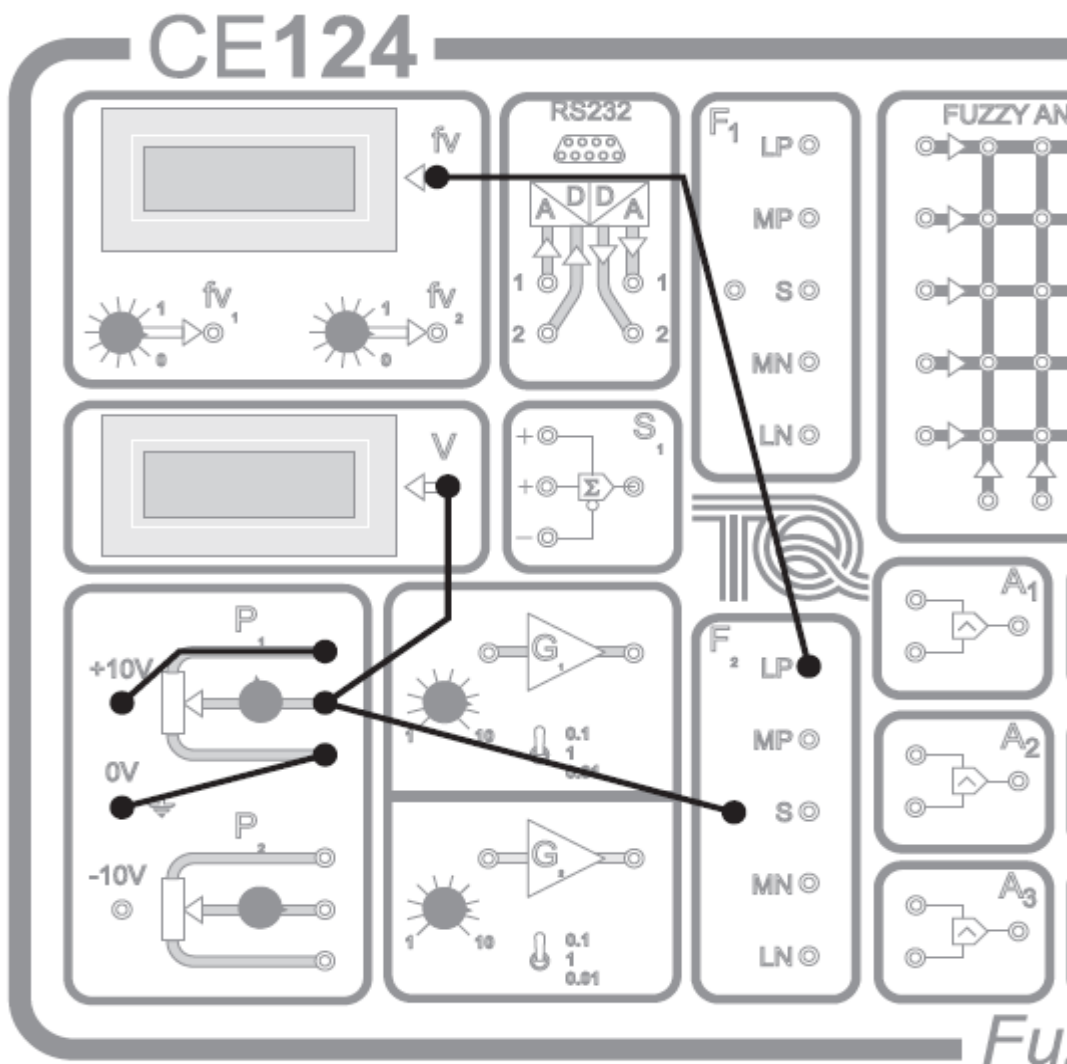


FIGURE 5: FUZZY MEMBERSHIP

B. Defuzzification

- Connect the equipment as shown in figure 6, including the dotted connection.
- Set the fuzzy variable potentiometer fv_1 to zero (fully anticlockwise) and the fuzzy variable potentiometer fv_2 to one (fully clockwise).
- Check that the fuzzy variable $fv_2=1$ and is connected to defuzzifier input MP (this is the dotted connection in figure 5). Increase the fuzzy variable fv_1 from 0 to 1 and record the reading.

Experiment 8: Fuzzy Logic Control

- Connect the fuzzy variable fv2 to defuzzifier input S by altering the dotted connection in figure 5 appropriately. Increase the fuzzy variable fv1 from 0 to 1 record the reading in table 2.
- Try to repeat the previous step to the rest of the defuzzifier inputs if you have free time!

TABLE 2

FV1	FV2	Output

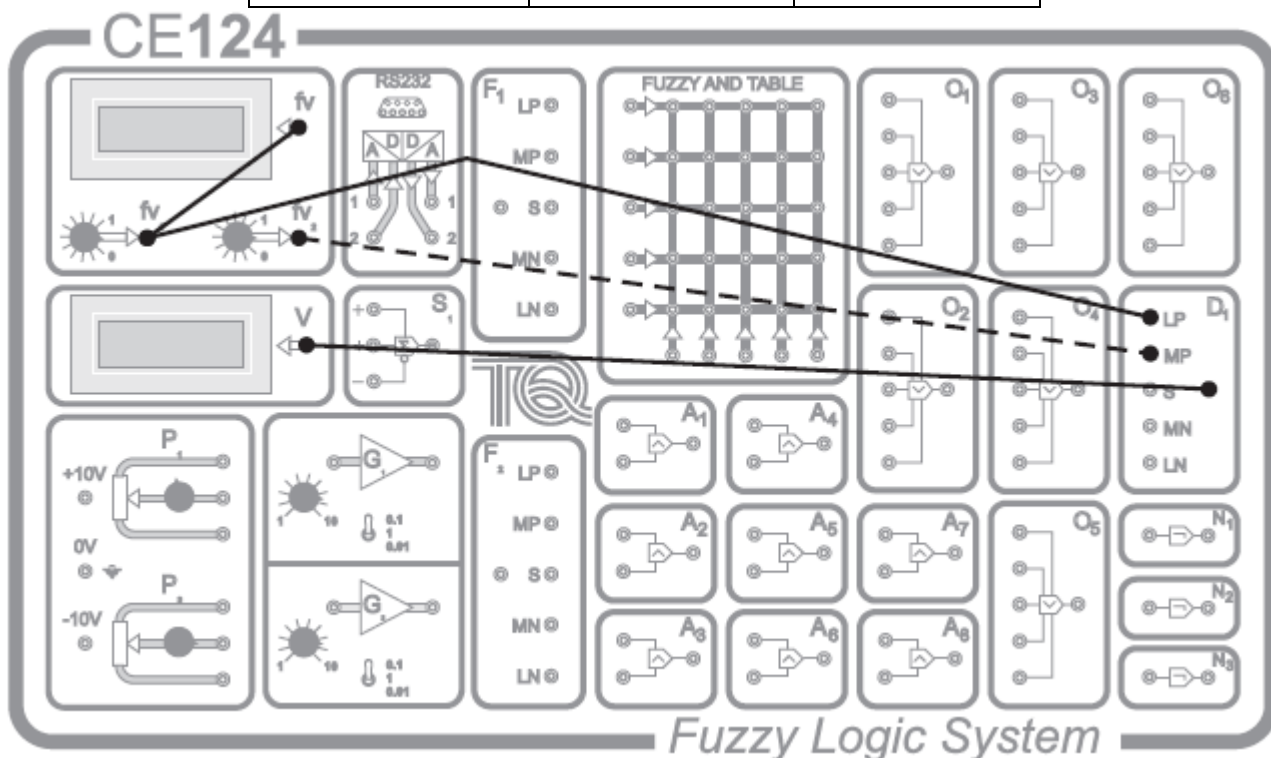


FIGURE 6: DEFUZZIFICATION

C. Fuzzy Logic Operators: AND, OR and NOT

- Connect the apparatus as shown in figure 7 using the solid connection only.
- Check that the fuzzy variable fv2=0.2. Increase the fuzzy variable fv1 from 0 to 1 in steps of 0.2. Note the reading of the fuzzy variable at the output of the fuzzy AND block at each step.
- Insert the output of the fuzzy AND block to a fuzzy NOT block (shown as shadow connections in figure 7). Note its effect on the fuzzy voltmeter value compared with the previous results.
- Repeat the previous procedure for the fuzzy OR block by altering the connection of fv1 and fv2 to the fuzzy OR block.
- Use your results to write relations that define the fuzzy AND, OR and NOT operations.

TABLE 3

Operation	FV1	FV2	Output	Output with not
AND				
OR				

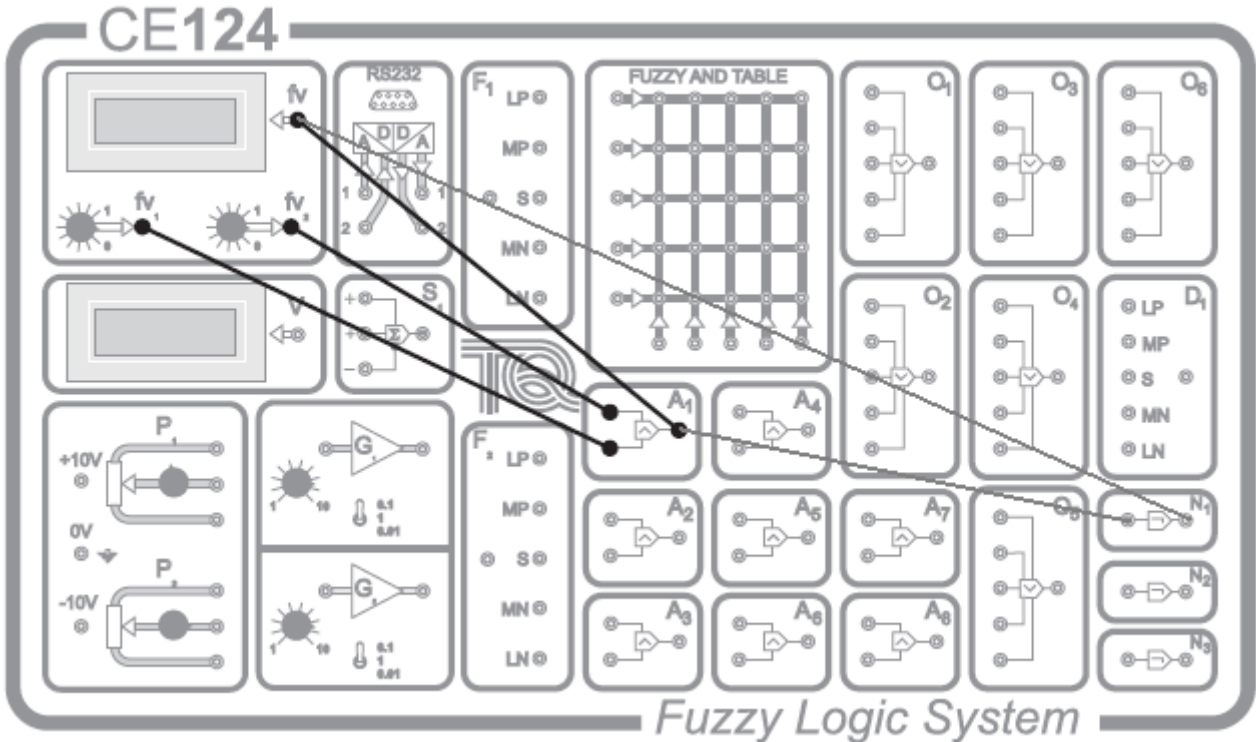


FIGURE 7: FUZZY LOGIC OPERATOR

Part 2: Proportional Control of the Thermal Control Process

The object of this exercise is to investigate fuzzy logic control applied to the thermal control process. The control is based on a fuzzy form of proportional (P) algorithm. The potentiometer P1 will be used to provide the reference (set-point) signal. The defuzzier output u is the control signal which is sent to the system input (the heater). The control signal is defuzzified from the fuzzy control law according to the classification:

- a) Large negative control=-10 V
- b) Medium negative control=-5 V
- c) Small control=0 V
- d) Medium positive control=5 V
- e) Large positive control=10 V

- Complete the following rule set so that it operate in a similar manner to a conventional proportional controller:

Rule 1: If {error LN} THEN {control _____}

Rule 2: If {error MN} THEN {control _____}

Rule 3: If {error S} THEN {control _____}

Experiment 8: Fuzzy Logic Control

Rule 4: If {error MP} THEN {control _____}

Rule 5: If {error LP} THEN {control _____}

The abbreviations used in this rule set are LN= large negative, LP=large positive, S=small, MN=medium negative, and MP= medium positive.

- Connect the apparatus as shown in figure 8 and complete the second half of connection to represent the fuzzy rule set written above, and then connect the output of the defuzzifier to the heater input on CE103.

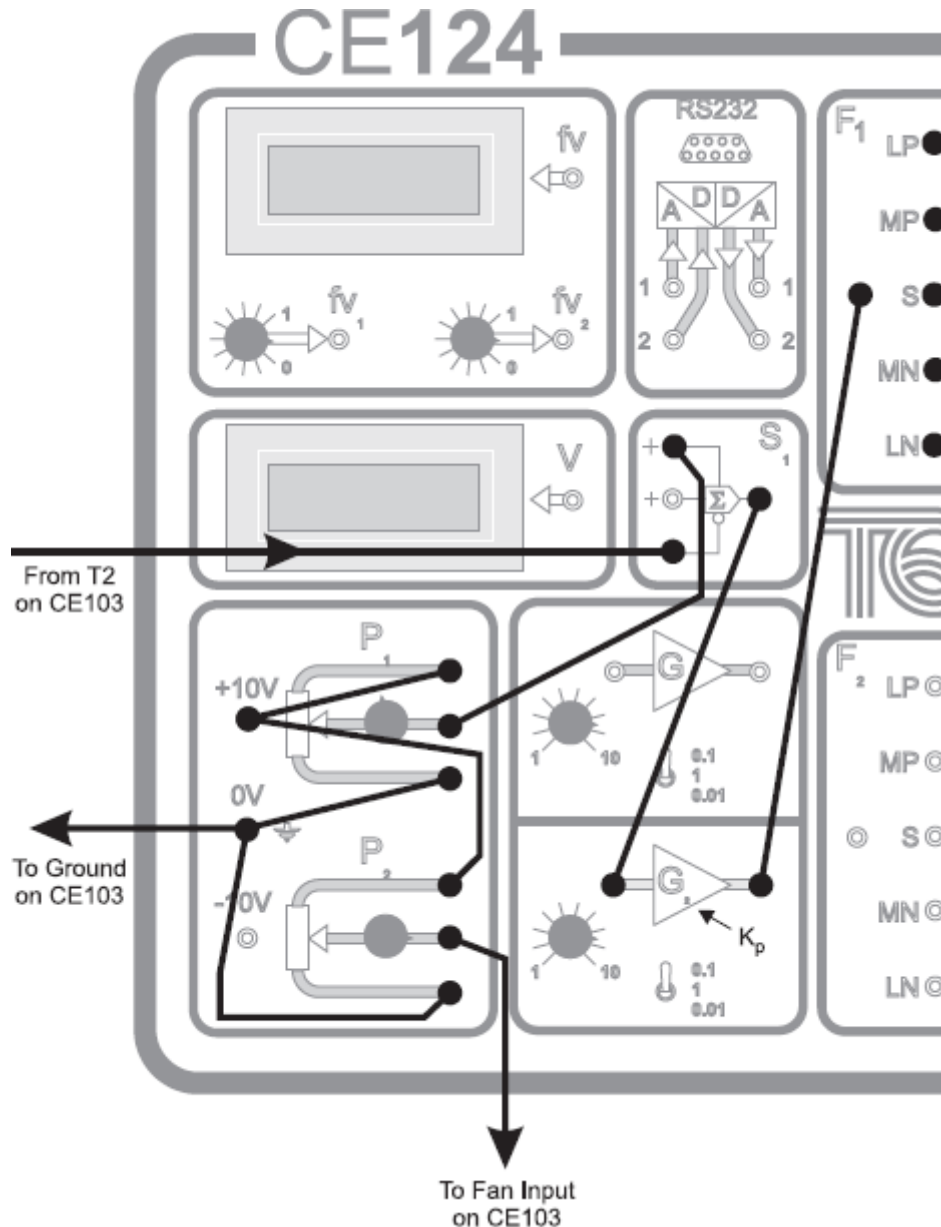


FIGURE 8: FUZZY CONTROL WIRING DIAGRAM 1

- Apply the following initial settings:
 - Set potentiometer P1 to 4 V and P2 to 3 V.
 - Pre-process gain K_p set to 10.

Experiment 8: Fuzzy Logic Control

- Output shutter of CE103 fully open.
- With P1 producing a set point (or reference) of 4 V, record the temperature output T2. After the temperature has settled to the desired value, increase the set point to 6 V and again record the output. After the temperature has settled to the desired value, decrease the set point to 4 V and again record the output. Observe the output of the fuzzy controller and compare it with what you expect from a conventional controller.
- Vary the gain K_p and monitor the effect upon the system response.
- Sketch a block diagram of the fuzzy control system which is used in this part. Compare the results from the fuzzy control system and what you would expect from a conventional proportional system.

Part 3: Proportional Control of the Coupled Tanks Apparatus

In this part of experiment a fuzzy logic controller is set up which applies a simple fuzzy proportional controller to the coupled tanks apparatus. The fuzzy rules are selected to insure that the controller output signal generates voltages which are inside the working range (0 V to 10 V) of the pump. The working range is 0 V to 10 V because:

- The pump cannot suck water out of the tanks hence the input voltage should not be less than 0 V.
- Since the pump maximum speed is achieved with a voltage of 10 V, the control signal should not be greater than 10 V.

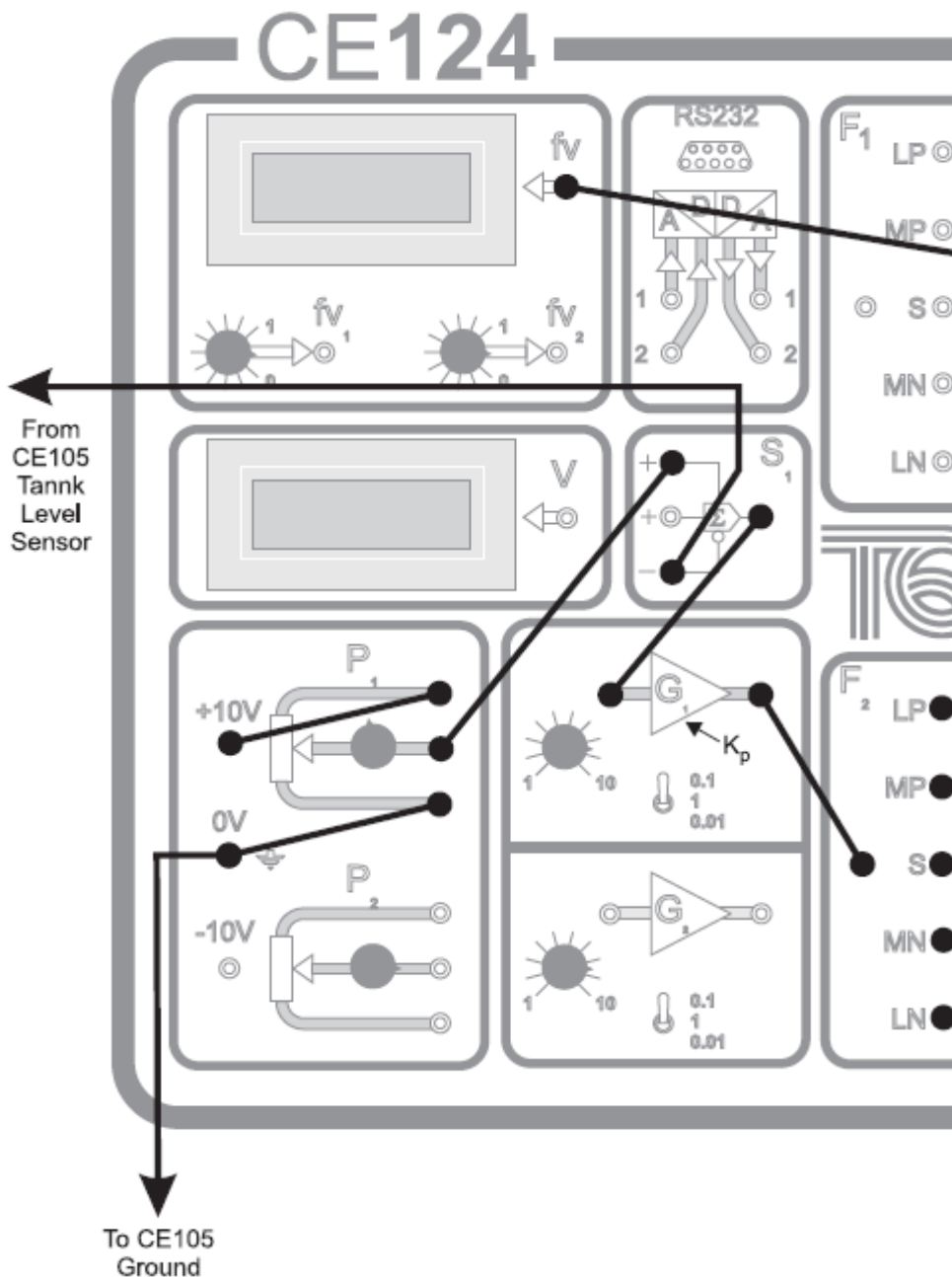


FIGURE 9: FUZZY CONTROL WIRING DIAGRAM 2

The potentiometer P1 will be used to produce the reference (set-point) signal. The defuzzifier output u is the control signal which is sent to the system input. The control signal is defuzzified from the fuzzy control law according to the classification mentioned in part 2.

- Connect the apparatus as shown in figure 9 and apply the following initial settings:
 - Set potentiometer P1 to 3 V.
 - Pre-processor gain K_p set to 10 which is included so that additional amplification of the error signal may be applied.
 - CE105 Coupled Tank Apparatus: valve A set to 5, valve B closed, valve C set to between 3 and 4.

Experiment 8: Fuzzy Logic Control

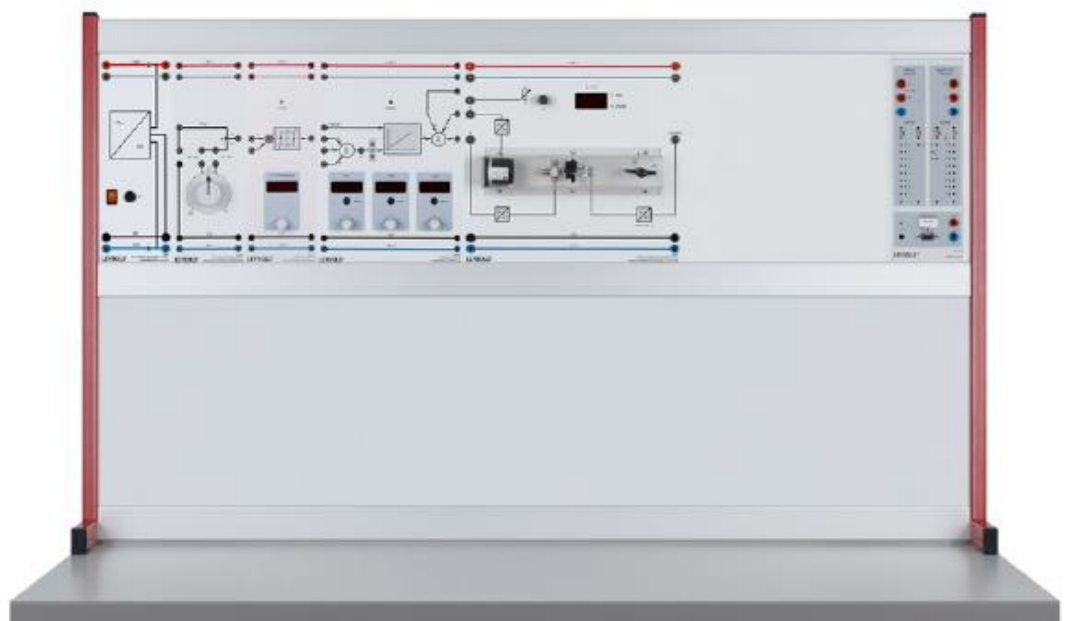
- Use the error and control signal to write a fuzzy logic rule set to provide a proportional positive control when error is positive taking in consideration that the minimum pump input is 0 V.
- Apply the fuzzy rule set written above on your connection and then connect the control signal to CE105 pump input.
- With P1 producing a set point (or reference) of 3 V, record the level in tank 1 using the voltmeter. After the level has settled to the desired value, increase the set point to 6 V and again record the output. After the level has settled to the desired value, decrease the set point to 3 V and again record the output. Observe the output of the fuzzy controller and compare it with what you would expect from a conventional controller.
- Vary the gain K_p and observe the effect upon the system response.

Experiment Nine

Control of Temperature System

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



Objectives

In this experiment, the students will learn the basic operation of a temperature controlled system and will also learn the static and transient behaviour of the temperature process.

Following are the objectives of the experiment:

- Study the effect of flap, ventilator-motor, and input power over the temperature output.
- Evaluation of the “step response” of the temperature process.
- Find the transfer function of the temperature process.
- Design a suitable PID controller for the system.

Introduction:

The objective of any automation is to be able to efficiently and reliably control the behaviour of the process. Temperature systems are widely used in real life as in industrial applications, domestic domain (home, office), medical and biological process engineering. The basic operation of a temperature-controlled system is analyzed in this experiment. Further, we should understand the system and determine the mathematical model of the process (Transfer function). This transfer function is used to design a PID controller that improves the performance of the system in closed loop.

Components and Equipments:

- | | |
|---------------------------------|-------|
| - Reference variable generator | 73402 |
| - Power amplifier | 73413 |
| - Power Supply +/- 15 V | 72686 |
| - Temperature controlled system | 73412 |
| - Two position controller | 73401 |
| - P controller | 73403 |
| - I controller | 73404 |
| - D controller | 73405 |
| - CASSY-interface with Compute | |
| - Set of bridging plug. | |
| - Cassy lab software | |

Experiment Procedure:

A. Static Performance

Static test

Experiment 9: Control of Temperature System

Static test is performed to check the linearity of the equipment under test. If we want to test the “input power” we need to keep the ventilator-motor and flap at constant values and vary the input power scale.

Experiment:

- Connect the “Temperature Control System Block Diagram” (Figure 1) to the Power Supply unit and with the “Reference Variable generator”.
- Set the output switch to 1V/10°C.
- Make sure that the temperature of the process settles at room temperature (Output voltage = the room temperature).

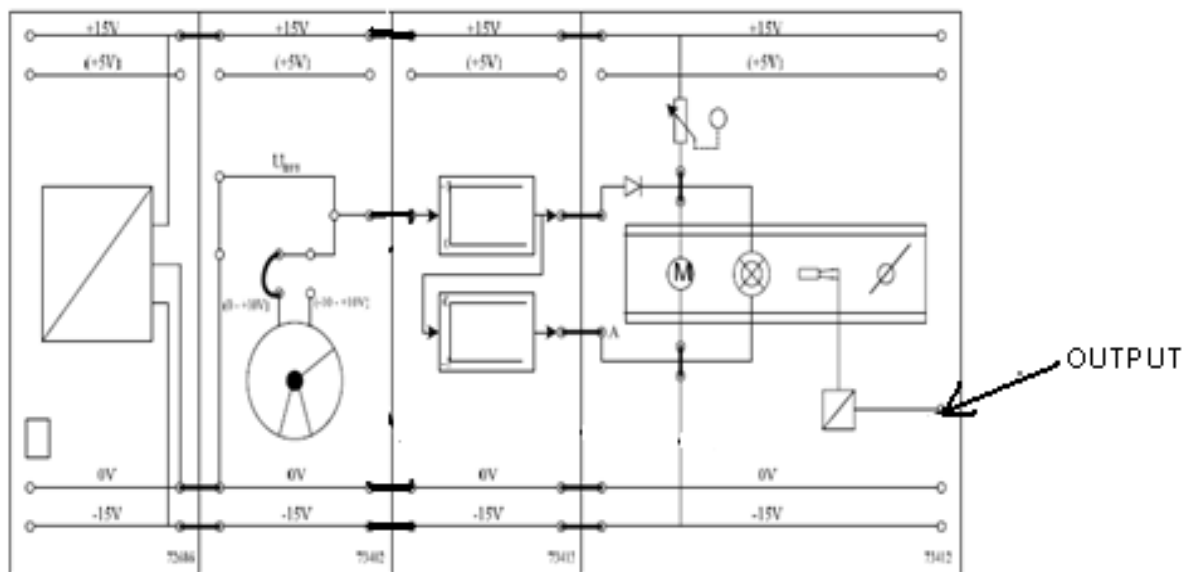


Figure 1 static test

- **Effect of “input power” over temperature**
 - Keep the flap at scale 2 and ventilator potentiometer at scale 3.
 - Record the temperature (using multimeter to reading the output voltage) of the system at inputs 0, 2, 4, 6, and 8 Volts.

Input Voltage (V)	0	2	4	6	8
Temperature (°C)					

2. Evaluation

- Plot the temperature versus the input power.

Label your scales very carefully.

B- System Identification.

The dynamic model of the temperature process can be described as first order linear transfer function:

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

Where K is the DC gain defined as the ratio of output voltage to input voltage.

$$K = \frac{y(\infty)}{u(\infty)}$$

$y(\infty)$ = final (steady state) value of the output

$u(\infty)$ = input to the system

τ : time constant when the output reaches 63% of the final value.

L : velocity lag (time delay) pure time delay (dead time), The time required for the system to start responding to the input change. as shown in Figure2

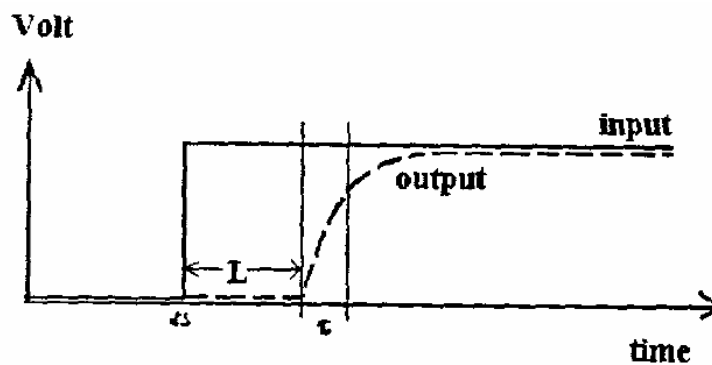


Figure 2 Representation of transient response

A linearized quantitative version of this model can be obtained with an open-loop experiment, by using the following procedure:

1. With the plant in open loop, take the plant manually to a normal operation point. Assume that the plant output settles at $y(t) = y_0$ for a constant input $u(t) = u_0$.
2. At an initial time $t = 0$, apply a step change to the plant input, from u_0 to u_∞ .
3. Record the plant output until it settles to the new operating point. Assume that you obtain the curve shown in figure 2. This curve is known the process reaction curve.
4. Compute the model parameters.

Experiment:

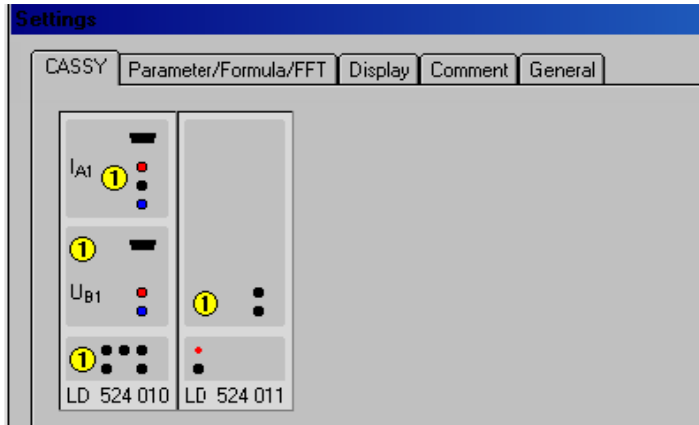
- 1- Connect the open loop Temperature control experiment as shown in the figure 3.
- 2- Connect the Temperature control system with the profi-cassy in order to plot the input and output signals

(Channel A: Output, channel B: input).

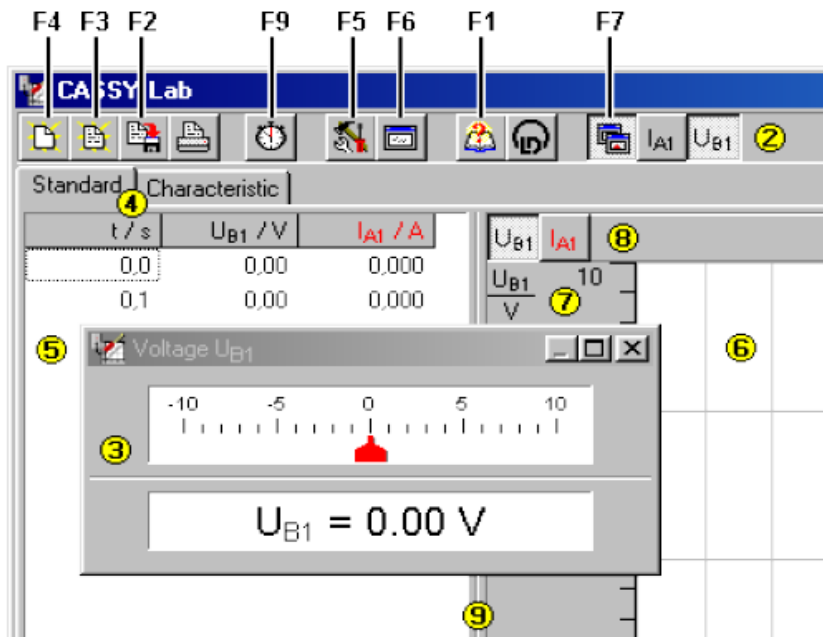
3- Open Cassy-lab software, Start => Programs => Cassy lab.

First measurements

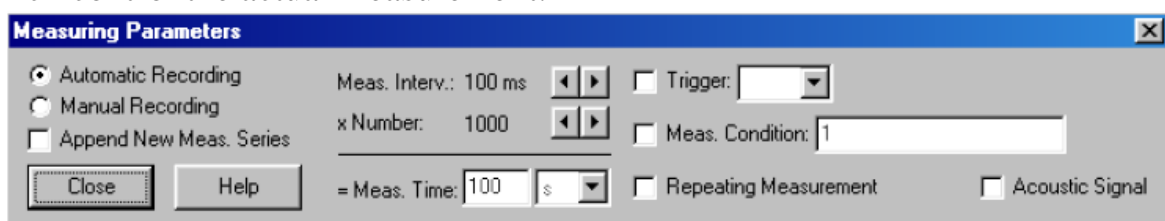
When the software detects one or more CASSY devices, the CASSY tab of the setup dialog (F5) displays the current configuration (including any attached sensor boxes). To conduct a measurement, **just click** on the corresponding input or output (1)




The channel initially appears automatically in the table (5) and in the diagram (6).



And will appear the box allows you to change the settings of measuring parameters which control the actual measurement.



Put the Meas. Time to 200 s.

When you want to start measurement just press to  **F9**, Starts and stops a new measurement. You can use the right mouse button to open the table display menu in the table and the evaluation menu in the diagram.

- 4- Using Cassy lab software, Record the transient behaviour for the following case: **(Step input):**
 - Cool down the system at room temperature (remove the input power from the temperature system open flap to 4 and increase the potentiometer to 10 to increase speed of motor until the output voltage equal the room temperature).
 - After cooling the system restore the input power, set the flap at scale 2 and ventilator potentiometer at scale 3.
 - Start with Reference voltage = 6 V until you reach a steady state.
 - Record the time required to achieve this state, name it T.

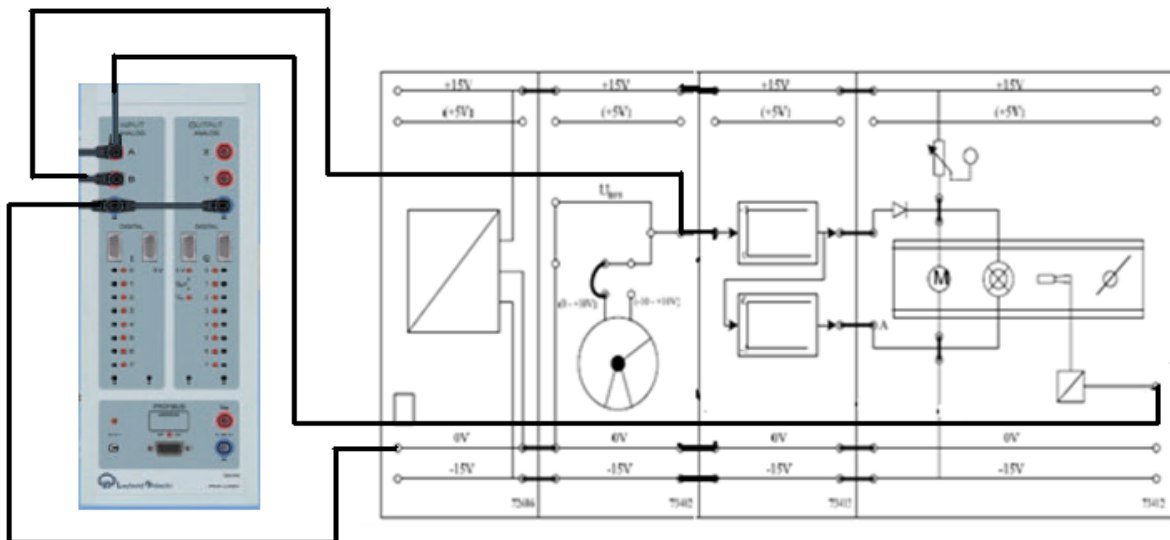


Figure 3

- 5- From the transient responses in above case, deduce the open loop transfer function of the temperature process (identify the parameters K , τ and L from the obtained response).
- 6- Comment on all of your results.

C. Open-Loop Two-Position Control

The main targets of the previous parts were to have an understanding of modelling concepts in terms of static and transient performance. The temperature process was used

in its open-loop form. This Part will introduce the idea of open and closed-loop control. Following are the objectives of this Part:

- Study the two-position (discontinuous) controller.
- Examine the effect of hysteresis on the temperature process.
- Maintaining output temperature close to a reference (set point) value.

There are many controllers that could be used to perform the task. They differ in simplicity and configuration. In this part, the two-position controller is introduced as the regulator. This controller will recognize only two states and will have only two actions: On or OFF.

Figure 4 shows a block diagram of the circuit under study:

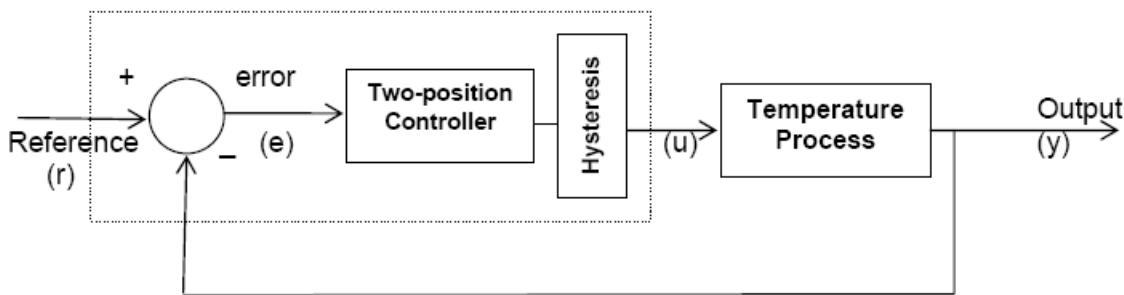


Figure 4: Block diagram of a closed loop two-position controller.

The controller that we will study can have hysteresis effects, by examining the block diagram, we can easily deduce that:

$$e(t) = r(t) - y(t)$$

If there is no switching hysteresis ($h = 0$), the two states of the controller are:

$$\begin{cases} e > 0 \rightarrow u: \text{ON} \\ e < 0 \rightarrow u: \text{OFF} \end{cases}$$

However, if there is switching hysteresis h , the two states:

$$\begin{cases} e > h \rightarrow u: \text{ON} \\ e < -h \rightarrow u: \text{OFF} \end{cases}$$

Experiment:

- 1- Cool down the system at room temperature (remove the input power from the temperature system open flap to 4 and increase the potentiometer to 10 to increase speed of motor until the output voltage equal the room temperature).
- 2- Set up the experimental arrangement as shown in the block diagram of Figure 5.
- 3- Keep the same settings as before (switch off).
- 4- Disconnect the feedback path from the output to the input of the controller.
- 5- Setting fan potentiometer to 3-scale division and the flap to 2-scale.
- 6- Set the reference to 6 V.

- 7- Set the switching hysteresis to 0 V and plot the reference and the output on the same graph.

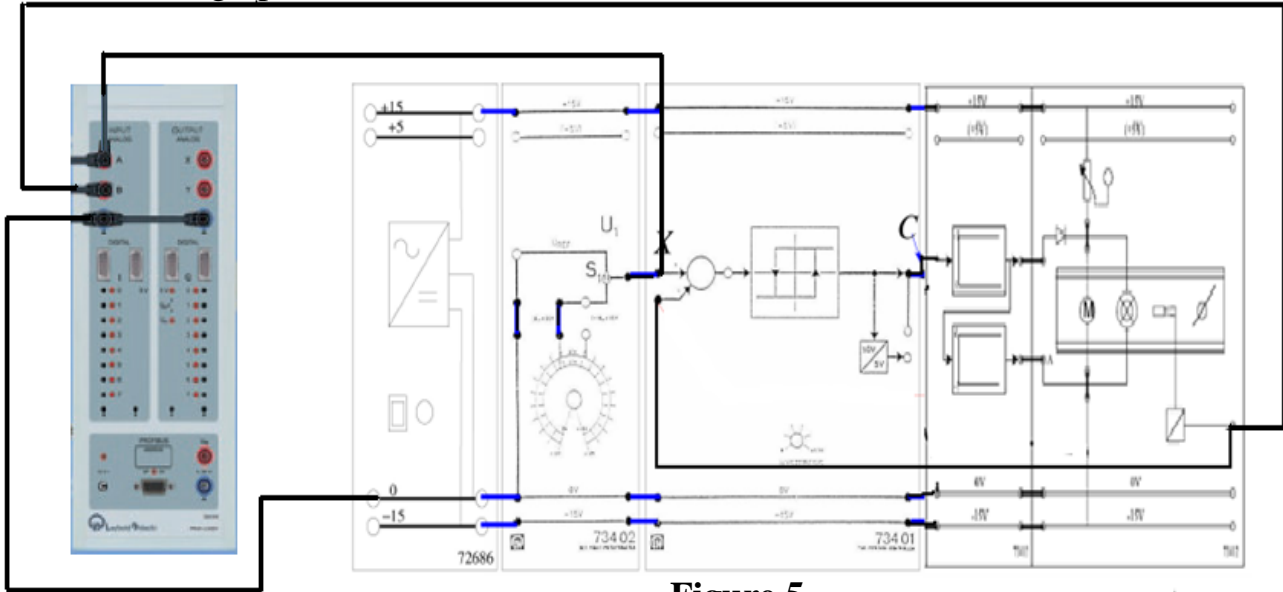


Figure 5

D. Closed-Loop Two-Position Control

Experiment:

- 1- Cool down the system at room temperature (remove the input power from the temperature system open flap to 4 and increase the potentiometer to 10 to increase speed of motor until the output voltage equal the room temperature).
- 2- Set up the experimental arrangement as shown in the block diagram of Figure 5.
- 3- Set the fan potentiometer to 3-scale division and the flap to 2-scale.
- 4- Set the reference to 6 V.
- 5- Plot the output and control signal for the following values of hysteresis:
 - 1) $h = 0$ V
 - 2) $h = \pm 0.5$ VObserve the behaviour of the lamp!, Deduce the frequency of the control signal.
- 6- Discuss the difference between the open-loop and the closed-loop two-position controller.
- 7- Discuss the effects of the hysteresis on the output and control signal.

E. PID-Controller:

So far, a simple control technique has been evaluated: the two-position control. This part will introduce another type of controller; which is called PID (Proportional, Integral and Derivative). This control approach is one of the oldest and most popular techniques used in the industry, because it is simple and effective.

Figure 6 shows a block diagram of this control.

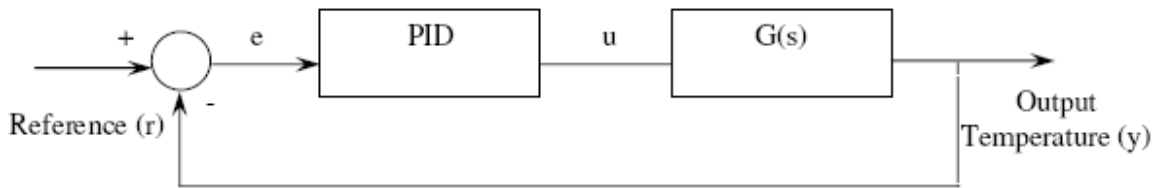


Figure 6

• **Cohen and Coon method.**

The model obtained (transfer function) can be used to derive various tuning methods for PID controllers. Cohen and Coon proposed one of these methods. The suggested parameters are shown in table 1.

Table 1

	K_p	T_r	T_d
P	$\frac{\tau}{K L} \left(1 + \frac{L}{3\tau} \right)$		
PI	$\frac{\tau}{K L} \left(0.9 + \frac{L}{12\tau} \right)$	$L \left(\frac{30\tau + 3L}{9\tau + 20L} \right)$	
PID	$\frac{\tau}{K L} \left(\frac{4}{3} + \frac{L}{4\tau} \right)$	$L \left(\frac{32\tau + 6L}{13\tau + 8L} \right)$	$\left(\frac{4\tau L}{11\tau + 2L} \right)$

The PID controller is given by:

$$u_{PID} = K_p e(t) + \frac{K_p}{T_r} \int e(s) ds + K_p T_d \frac{de(t)}{dt}$$

Experiment:

The connection for this experiment is given below in Figure 7:

1. Cool down the system at room temperature (remove the input power from the temperature system open flap to 4 and increase the potentiometer to 10 to increase speed of motor until the output voltage equal the room temperature).
2. Set the fan potentiometer to 3-scale division, the flap to 2-scale and power supply to 6 V.

Experiment 9: Control of Temperature System

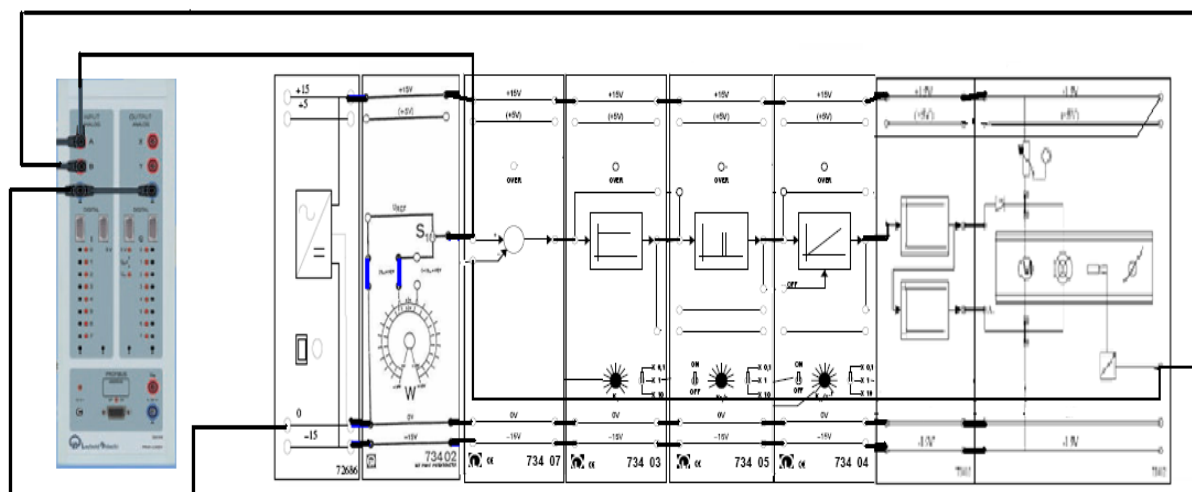


Figure 7

A. Effect of K_P , K_I , and K_D :

A proportional controller (K_p) will have the effect of reducing the rise time and will reduce, but never eliminate, the steady-state error.

An integral control (K_i) will have the effect of eliminating the steady-state error, but it may make the transient response worse.

A derivative control (K_d) will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response.

- Study the Effects of change each of controllers K_p , K_d , and K_i on a closed-loop system and summarized it in table shown next:

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
K_p		Increase	Small Change	
K_i	Decrease			Eliminate
K_D				

B. General tips for designing the PID controller:

When you are designing a PID controller for a given system, follow the steps shown below to obtain a desired response:

1. Obtain an open-loop response and determine what needs to be improved.
2. Add a proportional control to improve the rise time ($K_p = 10, 50$).
3. Add a derivative control to improve the overshoot.
4. Add an integral control to eliminate the steady state error.
5. Adjust each of the K_P , K_D and K_I until you obtain a desired overall response.

Experiment Ten

Twin Rotor MIMO System

Control Laboratory
MX-0908453

Mechatronics Eng. Dept.



1. Introduction

The TRMS workshop serves as a model of helicopter. However some significant simplifications are made. First is the fact that TRMS is attached to a tower and second of great importance that the helicopter position and velocity is controlled through the rotor velocity variation. In real helicopter the rotor velocity is more less constant and the propulsion is varied through the rotor blades angle modification.

Nevertheless, the most important dynamic characteristics present in a helicopter are captured in the TRMS mode. Like in a real helicopter there is significant cross coupling between two rotors. If we active the vertical position rotor the helicopter will also turn in the horizontal plane

With two inputs (the voltage supplied to the rotors) and Outputs (vertical and horizontal angles and angular velocities) the TRMS is an excellent MIMO plant for laboratory exercises.

2. Description

Figure 1 shows the TRMS mechanical unit consists of two rotors placed on beam together with a counterbalance. The whole unit is attached to the tower allowing for safe helicopter control experiments



Figure 1: TRMS Mechanical Unit

In order to design any control algorithm one must first understand the physical background behind the process and carry out identification experiments. A complete control system setup presented in figure 2

Experiment 10: Twin Rotor MIMO System

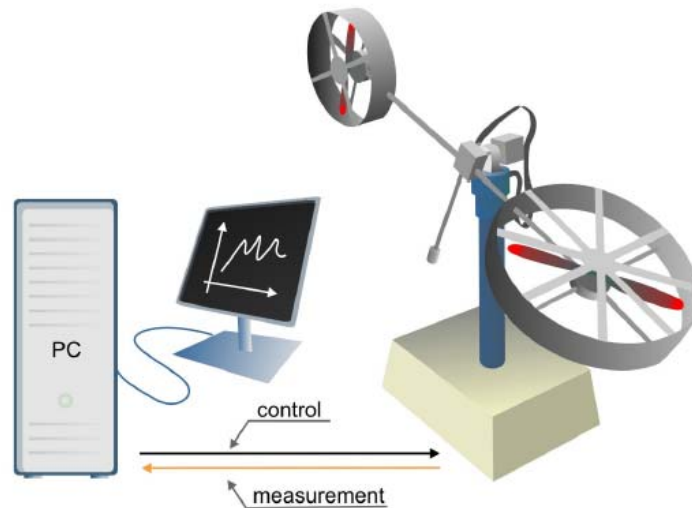


Figure 2: TRMS Control System

3. Mathematical Model of TRMS

The mechanical electrical model of the TRMS is presented in figure 3

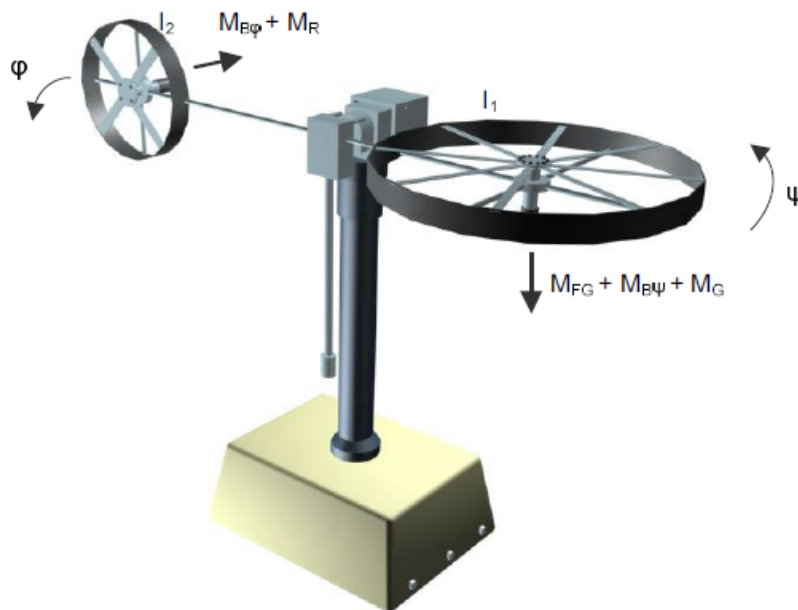


Figure 3: TRMS Model

TRMS models are nonlinear, that means at least one of the state (i - rotor current, ϕ – position) is an argument of a nonlinear function. In order to present such a model as a transfer function (a form of linear plant dynamics representation used in control engineering), it has to be linearized.

The following momentum equations can be derived for the vertical movements

Experiment 10: Twin Rotor MIMO System

$$I_1 \ddot{\psi} = M_1 - M_{FG} - M_{B\psi} - M_G \quad (1)$$

where

$$M_1 = a_1 \tau_1^2 + b_1 \tau_1 \quad \text{Nonlinear static characteristic} \quad (2)$$

$$M_{FG} = M_g \cdot \sin \psi \quad \text{Gravity momentum} \quad (3)$$

$$M_1 = B_{1\psi} \dot{\psi} + B_{2\psi} \text{sign}(\dot{\psi}) \quad \text{Friction force momentum} \quad (4)$$

$$M_G = K_{gy} M_1 \dot{\psi} \cos \psi \quad \text{Gyroscopic momentum} \quad (5)$$

The motor and the electrical control circuit is approximated by a first order transfer function thus in Laplace domain the motor momentum is described by:

$$\tau_1 = \frac{k_1}{T_{11}s + T_{10}} \cdot u_1 \quad (6)$$

Similar equations refer to the horizontal plane motion:

$$I_1 \ddot{\phi} = M_2 - M_{B\phi} - M_R \quad (7)$$

where

$$M_2 = a_1 \tau_1^2 + b_1 \tau_1 \quad \text{Nonlinear static characteristic} \quad (8)$$

$$M_2 = B_{1\phi} \dot{\phi} + B_{2\phi} \text{sign}(\dot{\phi}) \quad \text{Friction force momentum} \quad (9)$$

M_R is the cross reaction momentum approximated by

$$M_R = \frac{k_c(T_o s + 1)}{(T_p s + 1)} \cdot \tau_1 \quad (10)$$

Again the DC motor with the electrical circuit is given by:

$$\tau_2 = \frac{k_2}{T_{21}s + T_{20}} u_2 \quad (11)$$

4. System Identification

The TRMS is a MIMO plant – multiple input multiple output. Figure 4 presents a simplified schematic of the TRMS.

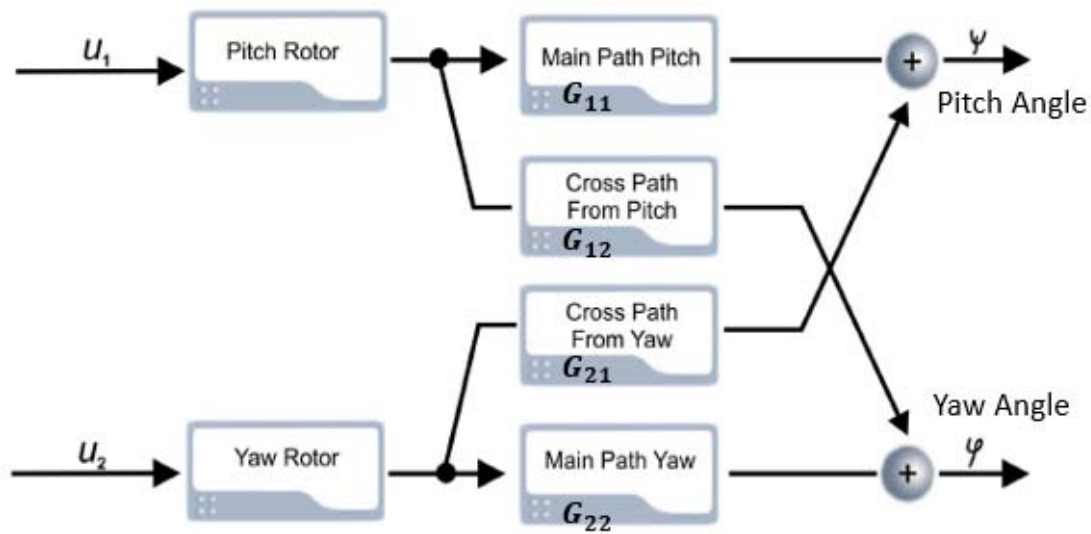


Figure 4: TRMS simplified system schematic

The TRMS is controlled with two inputs the u_1 and u_2 . The dynamics cross couplings are one of the key features of the TRMS. The position beams is measured with the means of incremental encoders, which provide a relative position signal. Thus every time the Real-Time TRMS simulation is run one must remember that setting proper initial conditions is important.

As mentioned in the previous description the MIMO is a nonlinear plant with significant cross couplings between the rotors (Figure 4). To keep the identification simple, the model can be treated as two linear rotor models with two linear couplings in between.

Thus four linear model have to be identified: two for the main dynamics path from u_1 to ψ (*pitch*) and u_2 to φ (*yaw*) and cross coupling dynamics paths from u_1 to φ (*yaw*) and u_2 to ψ (*pitch*).

Exercise 1: Voltage applied to input u_1 and $u_2 = 0$

In this part, the results are used to identified the model G_{11} that describe the relation between the control voltage u_1 and the angle ψ (*pitch*) and the model the model G_{12} that describe the relation between the control voltage u_1 and the angle φ (*yaw*)

The identified experiment is carried out using the model called (**Exercise 1.mdl**). This model excites the TRMS and records its response. The excitation signal is composed of several sinusoids. Two signals are collected in the form of vectors and available in the Workspace.

Exercise 2: Voltage applied to input u_2 and $u_1 = 0$

In this part, the results are used to identified the model G_{21} that describe the relation between the control voltage u_2 and the angle ψ (*pitch*) and the model the model G_{22} that describe the relation between the control voltage u_2 and the angle φ (*yaw*)

The identified experiment is carried out using the model called (**Exercise 2.mdl**). This model excites the TRMS and records its response. The excitation signal is composed of several sinusoids. Two signals are collected in the form of vectors and available in the Workspace.

Task

- 1) Carry out the identification experiment and collect the data from exercise 1. Identify the two model G_{11} and G_{12} using the MATLAB identification toolbox??
- 2) Carry out the identification experiment and collect the data from exercise 2. Identify the two model G_{21} and G_{22} using the MATLAB identification toolbox??
- 3) Build the Simulink Model like figure 4. Find the closed loop step response specification for two cases1: u_1 : step input, $u_2=0$ and case 2: $u_1=0$, u_2 : step input.

5. TRMS Control Design

There are numerous control algorithms. The PID controller algorithm is popular because of its simplicity. A general schematic of a simple control closed loop system is presented in figure 5.

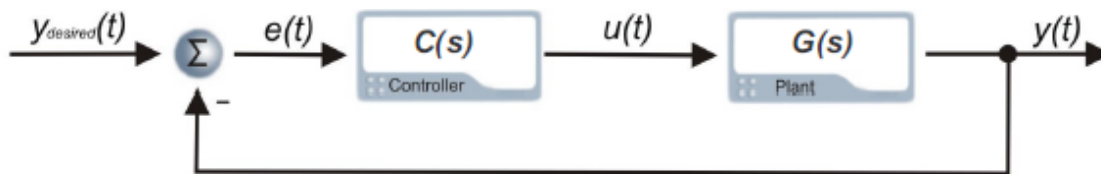


Figure 5: Simple control system - closed loop

Assuming that the plant is represented by its linear model its transfer function can be described as

$$G(s) = \frac{B(s)}{A(s)}$$

where (s) is Laplace operator.

The idea of control algorithm is to find a controller, which will fulfil our requirements (certain dynamic response, certain frequency damping, good response to the dynamic change of the desired value etc.). Every controller's input is $e(t)$ error signal, sometimes disturbance signals are also measured. Depending on the present and past values of the error signals, the controller performs such an action (changes the $u(t)$ control signal) that the $y(t)$ is as close to the $y_{desired}(t)$ value as possible at all the times.

Exercise 3: 1 DOF Pitch Rotor Control

In this exercise the pitch rotor will be controller To design a PID controller a model of the plant is needed. For this purpose, we can use a linearized plant model or use a model obtained in the identification experiment and design a PID controller based on that model.

Task 1: Simulation Result

- 1) Design using PID control tuner for the model obtained in the identification experiment. Make sure $u_2 = 0$ in this part.

Experiment 10: Twin Rotor MIMO System

- 2) Plot the simulation response of PID controller.
- 3) Write the parameters value of PID controller.

Task 2: Real Time Result

Open the Simulink file (**TRMS PID Pitch.mdl**) you will notice that this simulation is directed it an external module.

- 1) Plot the Response of pitch angle. Use the PID control that already designed in Task1. By change the parameters of the PID controller block (desired input is step).
- 2) Plot the Response of pitch angle. Set the following parameters **P=5, I=8, D= 10** of the PID controller with apply desired input is step.
- 3) Plot the Response of pitch angle. Set the following parameters **P=3, I=4, D= 15** of the PID controller with apply desired input is step.
- 4) Plot the Response of pitch angle. Set the following parameters **P=5, I=8, D= 10** of the PID controller with apply desired
- 5) input is sinusoid signal).
- 6) Compare between the response and write your comments

Exercise 4: 1 DOF yaw Rotor Control

In this exercise the yaw rotor will be controller to design a PID controller a model of the plant is needed. For this purpose, we can use a linearized plant model or use a model obtained in the identification experiment and design a PID controller based on that model.

Task 1: Simulation Result

- 1) Design using PID control tuner for the model obtained in the identification experiment. Make sure **u1 = 0** in this part.
- 2) Plot the simulation response of PID controller.
- 3) Write the parameters value of PID controller.

Task 2: Real Time Result

Open the Simulink file (**TRMS PID yaw.mdl**) you will notice that this simulation is directed it an external module.

Experiment 10: Twin Rotor MIMO System

- 1) Plot the Response of pitch angle. Use the PID control that already designed in Task1. By change the parameters of the PID controller block (desired input is step).
- 2) Plot the Response of pitch angle. Set the following parameters $P=2$, $I=0.5$, $D= 5$ of the PID controller with apply desired input is step.
- 3) Plot the Response of pitch angle. Set the following parameters $P=1$, $I=2$, $D= 5$ of the PID controller with apply desired input is step
- 4) Plot the Response of pitch angle. Set the following parameters $P=2$, $I=0.5$, $D= 5$ of the PID controller apply desired input is sinusoid signal.
- 5) Compare between the response and write your comments

Exercise 5: 2 DOF Position Control

In 1 DOF only one rotor has been used for control. The 2 DOF control includes 2 controllers working in parallel. For some MIMO plants such solution gives satisfactory results. If the cross coupling dynamics are significant, extra algorithms have to be incorporated. Either a decoupling scheme is introduced or multidimensional controllers should be designed.

The two controllers that have been designed in the previous sections are first tested in parallel.

Task 1: Simulation Result

- 1) Assume the desired value of two angle. And use the value of controller that was designed in exercise 3 and exercise 4
- 2) Plot the simulation response of PID controller.

Task 2: Real Time Result

Open the Simulink file (TRMS2DOF.mdl) you will notice that this simulation is directed it an external module.

- 1) Plot the Response of pitch angle. Use the PID control that already designed in Task1. By change the parameters of the PID controller block (desired input is step).

Experiment 10: Twin Rotor MIMO System

- 2) Set the following parameters **P=5, I=8, D= 10** of the PID controller for pitch angle, Set the following parameters **P=1, I=0.5, D= 5** of the PID controller with for yaw controller.
- 3) Plot the Response of pitch angle. And yaw angle for two desired input (step and sinusoidal)

Compare between the response and write your comments

References

- 1) <http://www.mathworks.com/help/index.html>
- 2) Farid Golnaraghi, Benjamin C.Kuo; Automatic Control Systems; Ninth Edition
- 3) Norman S.Nise; Control Systems Engineering; Sixth Edition
- 4) Richard C.Dorf, Robert H.Bishop; Modern Control Systems; Twelfth Edition.
- 5) Prof Zaer. Abo-Hammour Handouts.