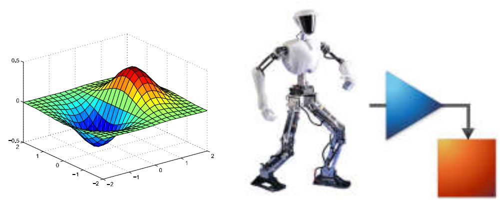
Control Engineering Sessional

Dr. Harunur Rashid
Md. Zahirul Islam
Md. Abu Jafar Rasel



Fall 2016

Gratitude

I am immensely grateful to the following person for the preparation and revision of this lecture.

- Mr. Md. Minal Nahin, Graduate Research Assistant , ME, IUPUI
- Mr. Anjan Goswami, Assistant Professor, MPE, AUST
- Dr. Md. Zahurul Haq, Professor , ME, BUET
- Dr. Sumon Saha, Assistant Professor , ME, BUET

Course Content & Syllabus

Tentative Course Plan [1]

Topic	Week[2]
Plotting and Linear Equations	1-2
Polynomials and Curve Fitting	3
Math Operations and Linear Equations in Simulink	4
Differential Equations and System Response	5
Transfer Functions, Zero-Poles and Stability	6
Laplace Transform and Residue	7
System Modelling and Controllers	8
PID Controller Tuning	9
Experiment	10
Experiment	11
Experiment	12
Final Examination	13

[1] with Mr. Abu Jafar Rasel

[2] Each lecture is approximately 120 min.

Marks Distribution

1. Quizes 50%
2. Classwork 40%
3. Assignment 10%

Marks Distribution

1. Quizes 50%
2. Classwork 40%
3. Assignment 10%
4. **Classwork will be held at the beginning of the class.**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Marks Distribution

1. Quizes 50%
2. Classwork 40%
3. Assignment 10%
4. **Classwork will be held at the beginning of the class.**
5. **There will be at least one sudden quiz.**

Marks Distribution

1. Quizes 50%
2. Classwork 40%
3. Assignment 10%
4. **Classwork will be held at the beginning of the class.**
5. **There will be at least one sudden quiz.**
6. **Absence from both quizes will result in a 'F' grade.**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Marks Distribution

1. Quizes 50%
2. Classwork 40%
3. Assignment 10%
4. **Classwork will be held at the beginning of the class.**
5. **There will be at least one sudden quiz.**
6. **Absence from both quizes will result in a 'F' grade.**
7. Calculators are not allowed for classwork and quiz.

Marks Distribution

1. Quizes 50%
2. Classwork 40%
3. Assignment 10%
4. **Classwork will be held at the beginning of the class.**
5. **There will be at least one sudden quiz.**
6. **Absence from both quizes will result in a 'F' grade.**
7. Calculators are not allowed for classwork and quiz.
8. Rough calculations on the question paper is not allowed either.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Marks Distribution

1. Quizes 50%
2. Classwork 40%
3. Assignment 10%
4. **Classwork will be held at the beginning of the class.**
5. **There will be at least one sudden quiz.**
6. **Absence from both quizes will result in a 'F' grade.**
7. Calculators are not allowed for classwork and quiz.
8. Rough calculations on the question paper is not allowed either.
9. For rough calculations you will need to use MATLAB command window.

Introduction

- **Control engineering** or **control systems engineering** is the engineering discipline that applies control theory to design systems with desired behaviors.
- The practice uses sensors to **measure** the output performance of the device being controlled and those measurements can be used to give **feedback** to the input **actuators** that can make corrections toward desired performance.
- When a device is designed to perform without the need of human inputs for correction it is called **automatic control** (such as cruise control for regulating a car's speed).
- Multi-disciplinary in nature, control systems engineering activities focus on implementation of control systems mainly derived by mathematical modeling of systems of a diverse range.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Introduction

- Control engineering or control systems engineering is the engineering discipline that applies control theory to design systems with desired behaviors.
- The practice uses sensors to measure the output performance of the device being controlled and those measurements can be used to give feedback to the input actuators that can make corrections toward desired performance.
- When a device is designed to perform without the need of human inputs for correction it is called automatic control (such as cruise control for regulating a car's speed).
- Multi-disciplinary in nature, control systems engineering activities focus on implementation of control systems mainly derived by mathematical modeling of systems of a diverse range.

ME 3204: Control Engineeering Sessional Spring 2016 5 / 99

Introduction

- **Control engineering** or **control systems engineering** is the engineering discipline that applies control theory to design systems with desired behaviors.
- The practice uses sensors to **measure** the output performance of the device being controlled and those measurements can be used to give **feedback** to the input **actuators** that can make corrections toward desired performance.
- When a device is designed to perform without the need of human inputs for correction it is called **automatic control** (such as cruise control for regulating a car's speed).
- Multi-disciplinary in nature, control systems engineering activities focus on implementation of control systems mainly derived by mathematical modeling of systems of a diverse range.

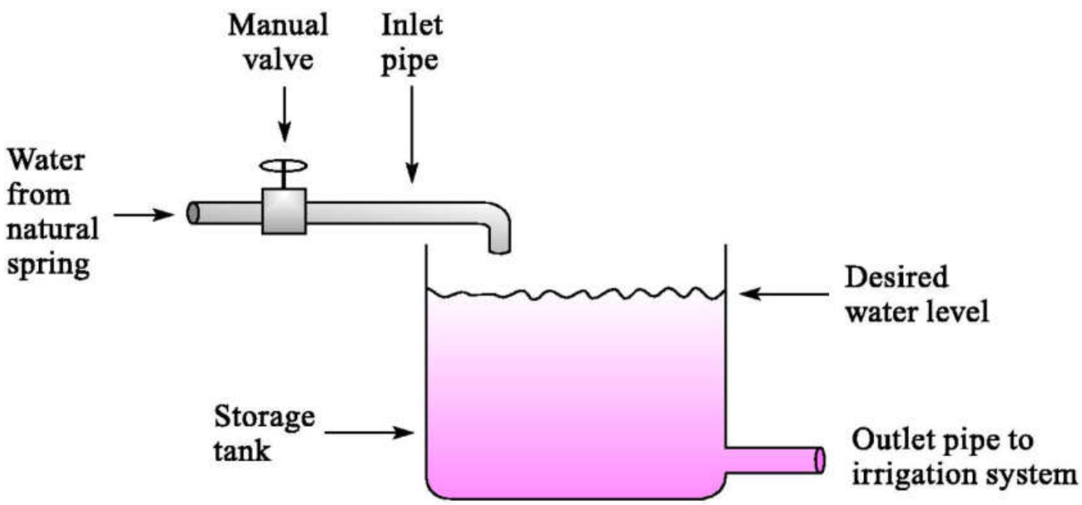
Control Systems *MATLAB* *SIMULINK* *Response* *Modeling* *Feedback-Control*

Introduction

- **Control engineering** or **control systems engineering** is the engineering discipline that applies control theory to design systems with desired behaviors.
- The practice uses sensors to **measure** the output performance of the device being controlled and those measurements can be used to give **feedback** to the input **actuators** that can make corrections toward desired performance.
- When a device is designed to perform without the need of human inputs for correction it is called **automatic control** (such as cruise control for regulating a car's speed).
- Multi-disciplinary in nature, control systems engineering activities focus on implementation of control systems mainly derived by mathematical modeling of systems of a diverse range.

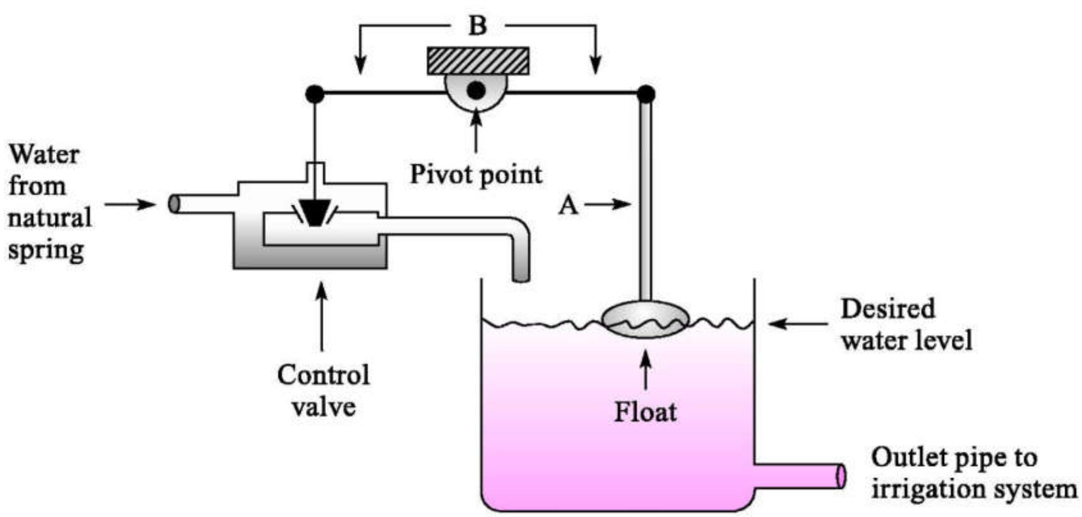
ME 3204: Control Engineeering Sessional *Spring 2016* 5 / 99

Liquid Level Control

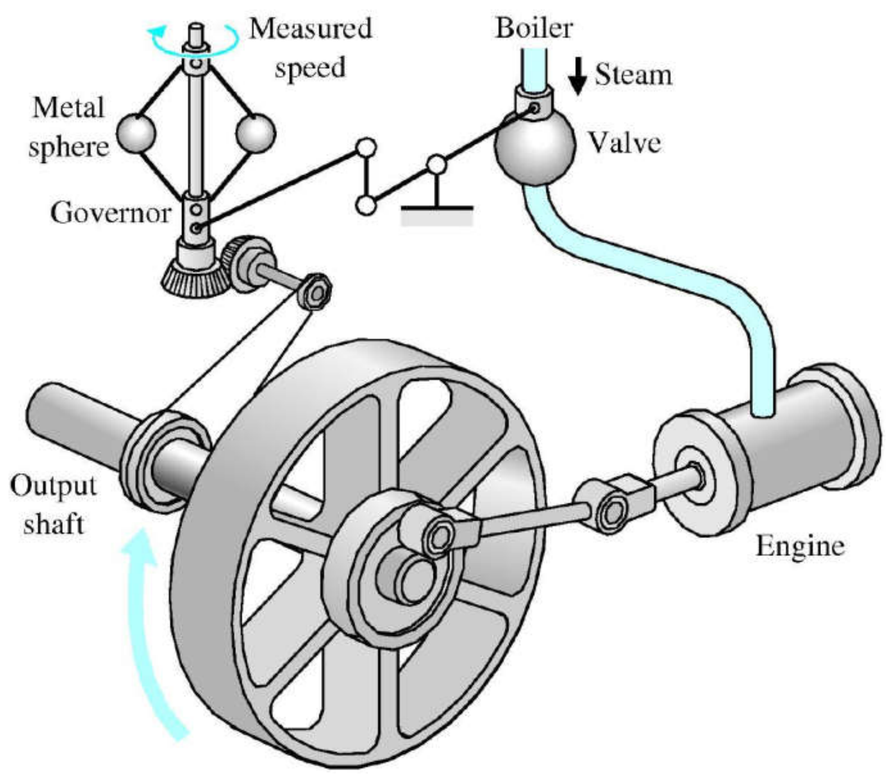


Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Automatic Control

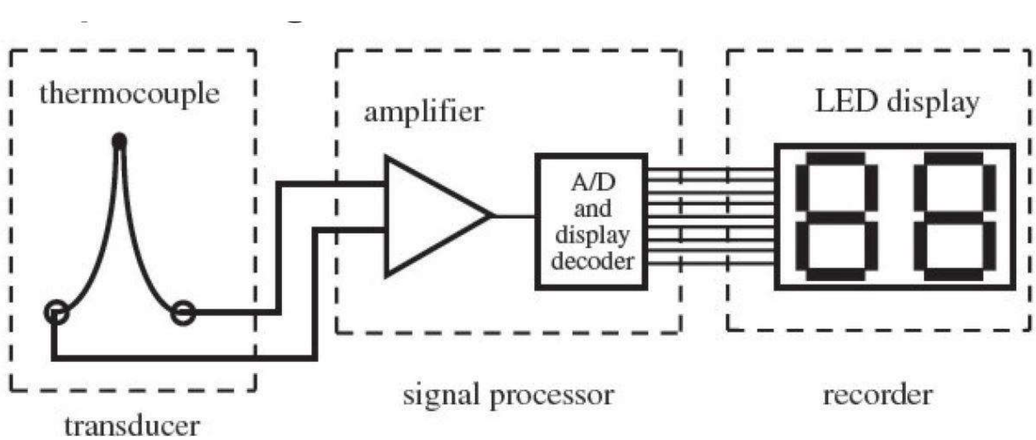


Watt's Flyball Governor



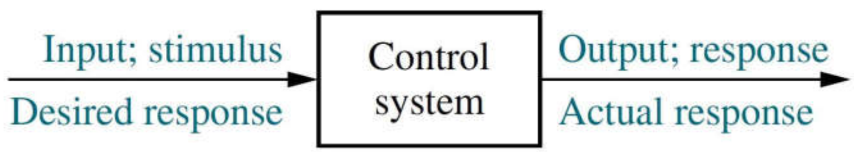
Elements of Measurement System

Consists of 3 basic elements



Control System

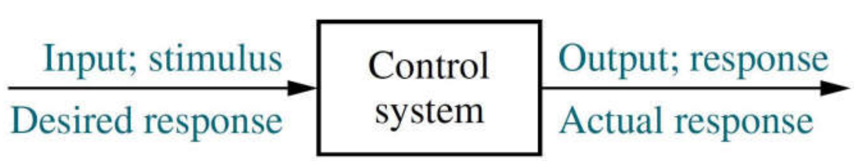
- The term **Control** means to regulate, to direct or to command.
- A **control system** is defined as a combination of devices and components connected or related so as to command, direct or regulate itself or another system.



Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Control System

- The term **Control** means to regulate, to direct or to command.
- A **control system** is defined as a combination of devices and components connected or related so as to command, direct or regulate itself or another system.



Classification of Control Systems

Toaster **Air Conditioner**

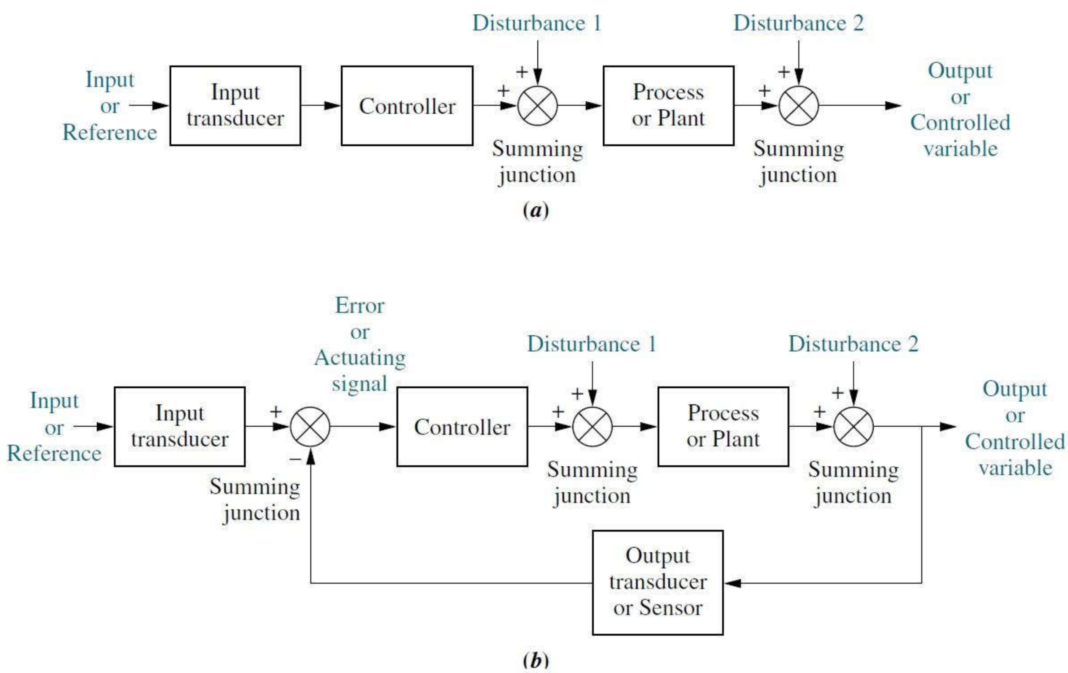
Classification of Control Systems

- **Open Loop System :** Does not correct for feedback/disturbance. Also called non-feedback system.
- **Close Loop System :** Corrects for feedback/disturbance. Also called feedback system.

Classification of Control Systems

- **Open Loop System :** Does not correct for feedback/disturbance. Also called non-feedback system.
- **Close Loop System :** Corrects for feedback/disturbance. Also called feedback system.

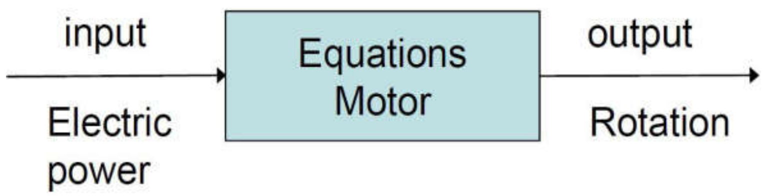
Classification of Control Systems



Block diagrams of control systems: **a.** open-loop system; **b.** closed-loop system

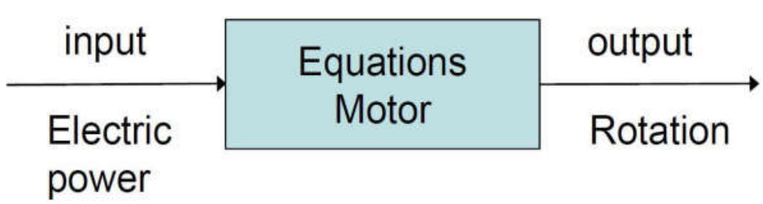
Definition and Modeling of Systems

- A **system** can be thought of as a box which has an input and an output.
- **Equations** are used to describe the relationship between the input and output of a system.
- **Response** of a system is a measure of its fidelity to its purpose.
- **Modeling** is the process of representing the behavior of a system by a collection of mathematical equations & logics.
- **Simulation** is the process of solving the model and it is performed on a computer.
- In this course, we're going to learn to simulate systems using **MATLAB-Simulink**



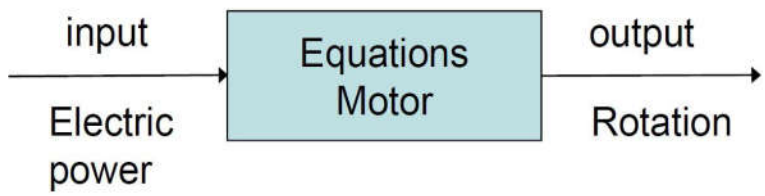
Definition and Modeling of Systems

- A **system** can be thought of as a box which has an input and an output.
- **Equations** are used to describe the relationship between the input and output of a system.
- **Response** of a system is a measure of its fidelity to its purpose.
- **Modeling** is the process of representing the behavior of a system by a collection of mathematical equations & logics.
- **Simulation** is the process of solving the model and it is performed on a computer.
- In this course, we're going to learn to simulate systems using **MATLAB-Simulink**



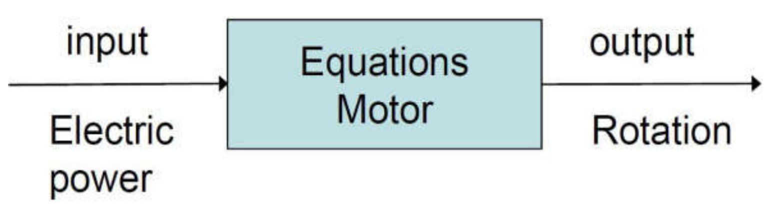
Definition and Modeling of Systems

- A **system** can be thought of as a box which has an input and an output.
- **Equations** are used to describe the relationship between the input and output of a system.
- **Response** of a system is a measure of its fidelity to its purpose.
- **Modeling** is the process of representing the behavior of a system by a collection of mathematical equations & logics.
- **Simulation** is the process of solving the model and it is performed on a computer.
- In this course, we're going to learn to simulate systems using **MATLAB-Simulink**



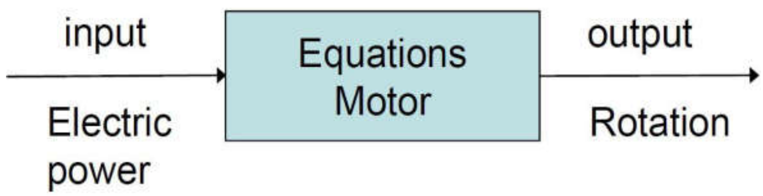
Definition and Modeling of Systems

- A **system** can be thought of as a box which has an input and an output.
- **Equations** are used to describe the relationship between the input and output of a system.
- **Response** of a system is a measure of its fidelity to its purpose.
- **Modeling** is the process of representing the behavior of a system by a collection of mathematical equations & logics.
- **Simulation** is the process of solving the model and it is performed on a computer.
- In this course, we're going to learn to simulate systems using **MATLAB-Simulink**



Definition and Modeling of Systems

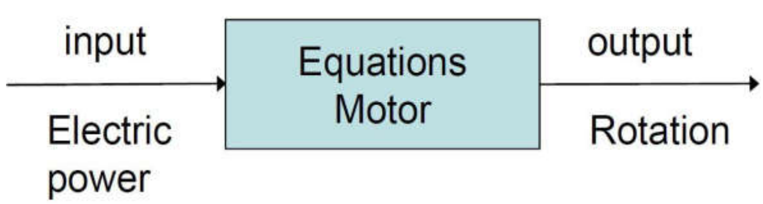
- A **system** can be thought of as a box which has an input and an output.
- **Equations** are used to describe the relationship between the input and output of a system.
- **Response** of a system is a measure of its fidelity to its purpose.
- **Modeling** is the process of representing the behavior of a system by a collection of mathematical equations & logics.
- **Simulation** is the process of solving the model and it is performed on a computer.
- In this course, we're going to learn to simulate systems using **MATLAB-Simulink**



Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Definition and Modeling of Systems

- A **system** can be thought of as a box which has an input and an output.
- **Equations** are used to describe the relationship between the input and output of a system.
- **Response** of a system is a measure of its fidelity to its purpose.
- **Modeling** is the process of representing the behavior of a system by a collection of mathematical equations & logics.
- **Simulation** is the process of solving the model and it is performed on a computer.
- In this course, we're going to learn to simulate systems using **MATLAB-Simulink**



Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

ME 3204: Control Engineeering Sessional Spring 2016 15 / 99

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

ME 3204: Control Engineeering Sessional Spring 2016 15 / 99

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

ME 3204: Control Engineeering Sessional Spring 2016 15 / 99

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands.*
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size.*
  4. Current Directory: *Shows file path to run a file.*
  5. Figure Window: *To display plots or graphics.*
  6. Edit Window: *To create or modify a new or existing file respectively.*

To restore default window go to **Menu: Desktop**/**Desktop Layout**/**Default**

ME 3204: Control Engineeering Sessional Spring 2016 15 / 99

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands*.
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size*.
  4. Current Directory: *Shows file path to run a file*.
  5. Figure Window: *To display plots or graphics*.
  6. Edit Window: *To create or modify a new or existing file respectively*.

To restore default window go to **Menu: Desktop/Desktop Layout/Default**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Introduction

- **MATLAB** developed by *The MathWorks Inc.*, stands for Matrix Laboratory.
- It is primarily used to perform scientific computation and visualization.
- **Run MATLAB**
- **MATLAB Environment** consists of
  1. Command Window: *To execute commands*.
  2. Command History: *To keep track of previously entered commands*
  3. Workspace: *Collection of all variables, their data type and size*.
  4. Current Directory: *Shows file path to run a file*.
  5. Figure Window: *To display plots or graphics*.
  6. Edit Window: *To create or modify a new or existing file respectively*.

To restore default window go to **Menu: Desktop/Desktop Layout/Default**

ME 3204: Control Engineeering Sessional Spring 2016 15 / 99

Basic Plotting (2D & 3D)

plot(x,y)

Plot the function $y = x - 3$ over the range $-1 \le x \le 1$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Basic Plotting (2D & 3D)

plot(x,y)

Plot the function $y = x - 3$ over the range $-1 \le x \le 1$

Solution:

- Create a vector of dependent variable x within the given range.
  $\rightarrow x = -1 : 0.1 : 1$

Basic Plotting (2D & 3D)

plot(x,y)

Plot the function $y = x - 3$ over the range $-1 \leq x \leq 1$

Solution:

- Create a vector of dependent variable x within the given range. $\rightarrow x = -1 : 0.1 : 1$
- Independent variable y is assigned within the given range. $\rightarrow y = x. - 3$

Basic Plotting (2D & 3D)

plot(x,y)

Plot the function $y = x - 3$ over the range $-1 \leq x \leq 1$

Solution:

- Create a vector of dependent variable x within the given range. $\rightarrow x = -1 : 0.1 : 1$
- Independent variable y is assigned within the given range. $\rightarrow y = x. - 3$
- Plot the graph in a figure window by $\rightarrow plot(x, y)$

Basic Plotting (2D & 3D)

plot(x,y)

Plot the function $y = x - 3$ over the range $-1 \leq x \leq 1$

Solution:

- Create a vector of dependent variable x within the given range. $\rightarrow x = -1 : 0.1 : 1$
- Independent variable y is assigned within the given range. $\rightarrow y = x. - 3$
- Plot the graph in a figure window by $\rightarrow plot(x, y)$
- Note the use of .(dot) :(colon) and ;(semicolon) in MATLAB

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Basic Plotting (2D & 3D)

plot(x,y)

Plot the function $y = x - 3$ over the range $-1 \leq x \leq 1$

Solution:

- Create a vector of dependent variable x within the given range. $\rightarrow x = -1 : 0.1 : 1$
- Independent variable y is assigned within the given range. $\rightarrow y = x. - 3$
- Plot the graph in a figure window by $\rightarrow plot(x, y)$
- Note the use of .(dot) :(colon) and ;(semicolon) in MATLAB
- The function linspace is used to assign 100 values within a given range.

ME 3204: Control Engineeering Sessional Spring 2016 16 / 99

Basic Plotting (2D & 3D)

plot(x,y)

Plot the function $y = x - 3$ over the range $-1 \leq x \leq 1$

Solution:

- Create a vector of dependent variable x within the given range. $\rightarrow x = -1 : 0.1 : 1$
- Independent variable y is assigned within the given range. $\rightarrow y = x. - 3$
- Plot the graph in a figure window by $\rightarrow plot(x, y)$
- Note the use of .(dot) :(colon) and ;(semicolon) in MATLAB
- The function linspace is used to assign 100 values within a given range.
- $\rightarrow x = linspace(-1, 1)$

Basic Plotting (2D & 3D)

Meshgrid & Surf

Plot the function $z = e^{-x^2-y^2}$ over the range $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$

Basic Plotting (2D & 3D)

Meshgrid & Surf

Plot the function $z = e^{-x^2-y^2}$ over the range $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$

Solution:

```
>> x= linspace(-2,2);
>> y= linspace(-2,2);
>> [X,Y] = meshgrid(x,y);
>> Z= exp(-X.^2 -Y.^2);
>> surf(X,Y,Z)
```


Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Basic Plotting (2D & 3D)

Meshgrid & Surf

Plot the function $z = e^{-x^2-y^2}$ over the range $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$

Solution:

```
>> x= linspace(-2,2);
>> y= linspace(-2,2);
>> [X,Y] = meshgrid(x,y);
>> Z= exp(-X.^2 -Y.^2);
>> surf(X,Y,Z)
```

Note the use of Plot tools

Basic Plotting (2D & 3D)

Meshgrid & Surf

Plot the function $z = e^{-x^2-y^2}$ over the range $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$

Solution:

```
>> x= linspace(-2,2);
>> y= linspace(-2,2);
>> [X,Y] = meshgrid(x,y);
>> Z= exp(-X.^2 -Y.^2);
>> surf(X,Y,Z)
```

Note the use of Plot tools

Try mesh(X,Y,Z) instead of surf(X,Y,Z)

Practice

Plot the function $c = a^2 - b^2$ over the range $-2 \leq a \leq 2$ and $-2 \leq b \leq 2$
Plot the function $y = sinx$ over the range $-2\pi \leq x \leq 2\pi$
Plot the function $y = x^3 - x^2 + 3$ over the range $-4 \leq x \leq 4$

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.

ME 3204: Control Engineeering Sessional Spring 2016 18 / 99

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.
5. **legend('a,b')** ⇒ Creates legend in figure window.

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.
5. **legend('a,b')** ⇒ Creates legend in figure window.
6. **shading interp** ⇒ Changes shading of plot(faceted, flat and interp).

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.
5. **legend('a,b')** ⇒ Creates legend in figure window.
6. **shading interp** ⇒ Changes shading of plot(faceted, flat and interp).
7. **hold on** ⇒ holds the current figure window for further plotting.

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.
5. **legend('a,b')** ⇒ Creates legend in figure window.
6. **shading interp** ⇒ Changes shading of plot(faceted, flat and interp).
7. **hold on** ⇒ holds the current figure window for further plotting.
8. **ezplot('x')** ⇒ plots the graph of the string input function

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.
5. **legend('a,b')** ⇒ Creates legend in figure window.
6. **shading interp** ⇒ Changes shading of plot(faceted, flat and interp).
7. **hold on** ⇒ holds the current figure window for further plotting.
8. **ezplot('x')** ⇒ plots the graph of the string input function
9. **area(x,y)** ⇒ same as **plot(x,y)** but the area under the curves are filled.

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.
5. **legend('a,b')** ⇒ Creates legend in figure window.
6. **shading interp** ⇒ Changes shading of plot(faceted, flat and interp).
7. **hold on** ⇒ holds the current figure window for further plotting.
8. **ezplot('x')** ⇒ plots the graph of the string input function
9. **area(x,y)** ⇒ same as **plot(x,y)** but the area under the curves are filled.
10. **bar(x,y)** ⇒ plots bar graph .

Some useful functions for plotting

1. **figure** ⇒ Creates an empty figure window.
2. **title('string')** ⇒ Creates title in figure window.
3. **xlabel('x-axis')** ⇒ Creates x-axis label in figure window.
4. **ylabel('y-axis')** ⇒ Creates y-axis label in figure window.
5. **legend('a,b')** ⇒ Creates legend in figure window.
6. **shading interp** ⇒ Changes shading of plot(faceted, flat and interp).
7. **hold on** ⇒ holds the current figure window for further plotting.
8. **ezplot('x')** ⇒ plots the graph of the string input function
9. **area(x,y)** ⇒ same as **plot(x,y)** but the area under the curves are filled.
10. **bar(x,y)** ⇒ plots bar graph .
11. **mesh(x,y,z)** ⇒ same as **surf(x,y,z)** except patches between lines are not filled.

Solving Linear Equations : linsolve function

Any linear system of equations must have 0, 1 or infinite number of solutions.

linsolve(A,B) takes coefficient matrix A and constant matrix B as input and solves the system.

Solve the following system.

$$\text{System of Linear equations} \begin{cases} x + 3y + z = 5 \\ 2x - y + 2z = 17 \\ 3x + 4y + 5z = 26 \end{cases}$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Solving Linear Equations : linsolve function

Any linear system of equations must have 0, 1 or infinite number of solutions.

linsolve(A,B) takes coefficient matrix A and constant matrix B as input and solves the system.

Solve the following system.

$$\text{System of Linear equations} \begin{cases} x + 3y + z = 5 \\ 2x - y + 2z = 17 \\ 3x + 4y + 5z = 26 \end{cases}$$

ME 3204: Control Engineeering Sessional Spring 2016 19 / 99

Solving Linear Equations : linsolve function

Any linear system of equations must have 0, 1 or infinite number of solutions.
linsolve(A,B) takes coefficient matrix A and constant matrix B as input and solves the system.

Solve the following system.

$$\text{System of Linear equations} \begin{cases} x + 3y + z = 5 \\ 2x - y + 2z = 17 \\ 3x + 4y + 5z = 26 \end{cases}$$

Solution visualization

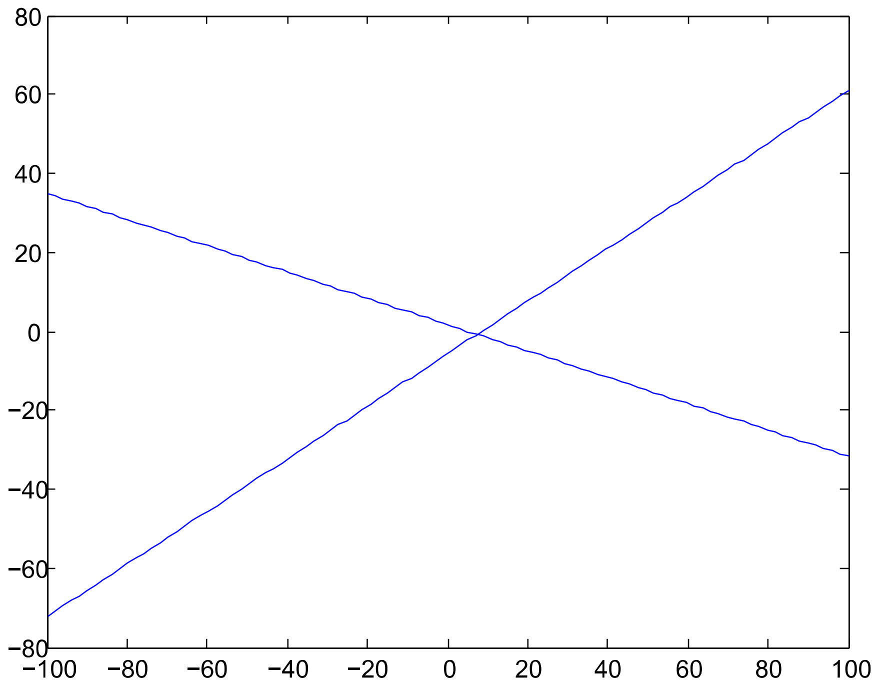
Draw the plot of the following equations in same graph.

$$\text{System of Linear equations with one solution} \begin{cases} x + 3y = 5 \\ 2x - y = 17 \end{cases}$$

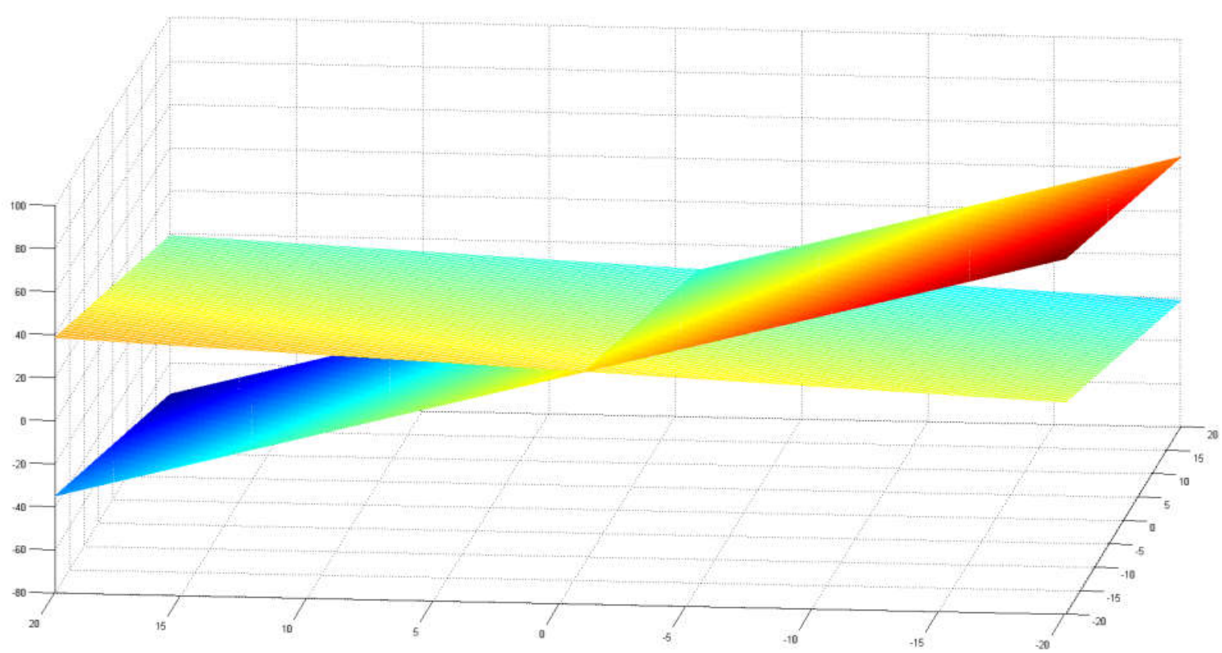
Draw the mesh plot of the following equations in same graph.

$$\text{System of Linear equations with infinite solutions} \begin{cases} x + 3y + z = 5 \\ 2x - y + 2z = 17 \end{cases}$$

Solution visualization (contd.)



Solution visualization (contd.)



Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n+1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Polynomials

- MATLAB interprets an nth order polynomial as a row vector of $n+1$
- Consider a polynomial $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- P = [1 3 -15 -2 9] $\leftarrow$
- or P = [1, 3, -15, -2, 9] $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command polyval(P,s)
- polyval(P,3) $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

ME 3204: Control Engineeering Sessional Spring 2016 23 / 99

Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n + 1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n + 1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

ME 3204: Control Engineeering Sessional Spring 2016 23 / 99

Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n + 1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n + 1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

ME 3204: Control Engineeering Sessional Spring 2016 23 / 99

Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n + 1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n + 1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

ME 3204: Control Engineeering Sessional Spring 2016 23 / 99

Polynomials

- **MATLAB** interprets an **nth** order polynomial as a row vector of $n+1$
- *Consider a polynomial* $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$
- Enter it in MATLAB as row vector P as follows
- **P = [1 3 -15 -2 9]** $\leftarrow$
- or **P = [1, 3, -15, -2, 9]** $\leftarrow$
- To evaluate the value of P at $s = 3$ or P(3) use the following command **polyval(P,s)**
- **polyval(P,3)** $\leftarrow$
- MATLAB will show you the value of P(3) to be 30.
- If there are missing terms, zeros must be entered at the appropriate places.

Practice

Evaluate the value of $y = 2s^2 + 3s + 4$ at $s = 1, -3$

Practice

Evaluate the value of $y = 2s^2 + 3s + 4$ at $s = 1, -3$

Answer: $y(1) = 9$ $y(-3) = 13$

Practice

Evaluate the value of $y = 2s^3 + 1$ at $s = -4$

Practice

Evaluate the value of
$y = 2s^3 + 1$ at $s = -4$

Answer: $y(-4) = -127$

Roots

- **MATLAB** can find the roots of a polynomial P by the command **roots(P)**

Roots

- **MATLAB** can find the roots of a polynomial P by the command **roots(P)**
- or **r = roots(P)**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Roots

- **MATLAB** can find the roots of a polynomial P by the command **roots(P)**
- or **r = roots(P)**
- The answer is represented in a column vector.

Roots

- **MATLAB** can find the roots of a polynomial P by the command **roots(P)**
- or **r = roots(P)**
- The answer is represented in a column vector.

Find the roots of $s^2 + 3s + 2$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Roots

- **MATLAB** can find the roots of a polynomial P by the command **roots(P)**
- or **r = roots(P)**
- The answer is represented in a column vector.

Find the roots of $s^2 + 3s + 2$

Answer: $-2, -1$

Find the Polynomial

- If you know the roots(r) of a polynomial P, **MATLAB** can find the polynomial by the command **poly(r)**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Find the Polynomial

- If you know the roots(r) of a polynomial P, **MATLAB** can find the polynomial by the command **poly(r)**

- Represent the roots by a column vector.

Find the Polynomial

- If you know the roots(r) of a polynomial P, **MATLAB** can find the polynomial by the command **poly(r)**
- Represent the roots by a column vector.
- **r = [-1; -2]** ←

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Find the Polynomial

- If you know the roots(r) of a polynomial P, **MATLAB** can find the polynomial by the command **poly(r)**
- Represent the roots by a column vector.
- **r = [-1; -2]** ←
- **p = poly(r)** ←

Find the Polynomial

- If you know the roots(r) of a polynomial P, **MATLAB** can find the polynomial by the command **poly(r)**
- Represent the roots by a column vector.
- **r = [-1; -2]** ←
- **p = poly(r)** ←
- The answer is represented in a row vector p = [1 3 2].

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Find the Polynomial

- If you know the roots(r) of a polynomial P, **MATLAB** can find the polynomial by the command **poly(r)**
- Represent the roots by a column vector.
- **r = [-1; -2]** ←
- **p = poly(r)** ←
- The answer is represented in a row vector p = [1 3 2].

Find the polynomial whose roots are -5.5745,2.5836,-0.7951,0.7860 .

Find the Polynomial

- If you know the roots(r) of a polynomial P, **MATLAB** can find the polynomial by the command **poly(r)**
- Represent the roots by a column vector.
- **r = [-1; -2]** $\leftarrow$
- **p = poly(r)** $\leftarrow$
- The answer is represented in a row vector p = [1 3 2].

Find the polynomial whose roots are -5.5745,2.5836,-0.7951,0.7860 .

Answer: $P(s) = s^4 + 3s^3 - 15s^2 - 2s + 9$

Conv(x,y) Function: Convolution

- **MATLAB** can multiply two polynomials by the command **conv(x,y)**

Conv(x,y) Function: Convolution

- **MATLAB** can multiply two polynomials by the command **conv(x,y)**
- Define the polynomials separately.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Conv(x,y) Function: Convolution

- **MATLAB** can multiply two polynomials by the command **conv(x,y)**
- Define the polynomials separately.
- The product is returned as a row matrix.

ME 3204: Control Engineeering Sessional Spring 2016 28 / 99

Conv(x,y) Function: Convolution

- **MATLAB** can multiply two polynomials by the command **conv(x,y)**
- Define the polynomials separately.
- The product is returned as a row matrix.
- In case there is more than two polynomials perform multiplication two at a time.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Conv(x,y) Function: Convolution

- **MATLAB** can multiply two polynomials by the command **conv(x,y)**
- Define the polynomials separately.
- The product is returned as a row matrix.
- In case there is more than two polynomials perform multiplication two at a time.

Multiply $(s + 2)$ and $s^2 + 4s + 8$

Conv(x,y) Function: Convolution

- **MATLAB** can multiply two polynomials by the command **conv(x,y)**
- Define the polynomials separately.
- The product is returned as a row matrix.
- In case there is more than two polynomials perform multiplication two at a time.

Multiply $(s+2)$ and s^2+4s+8

Answer: $s^3+6s^2+16s+16$

Practice

Evaluate the product of $(s+3)$, $(s+6)$ and $(s+2)$

Practice

Evaluate the product of $(s+3)$, $(s+6)$ and $(s+2)$

Answer: $s^3 + 11s^2 + 36s + 36$

deconv(z,y) Function

- **MATLAB** can divide a polynomial z(s) by another polynomial y(s) by the command **deconv(z,y)**

deconv(z,y) Function

- **MATLAB** can divide a polynomial z(s) by another polynomial y(s) by the command **deconv(z,y)**
- The syntax [x,r] = deconv(z,y) is preferred.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

deconv(z,y) Function

- **MATLAB** can divide a polynomial z(s) by another polynomial y(s) by the command **deconv(z,y)**
- The syntax [x,r] = deconv(z,y) is preferred.

- Otherwise MATLAB may show the quotient only and skip the remainder.

deconv(z,y) Function

- **MATLAB** can divide a polynomial z(s) by another polynomial y(s) by the command **deconv(z,y)**
- The syntax [x,r] = deconv(z,y) is preferred.
- Otherwise MATLAB may show the quotient only and skip the remainder.
- The division is shown in two row matrix. One for quotient and one for remainder.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

deconv(z,y) Function

- **MATLAB** can divide a polynomial z(s) by another polynomial y(s) by the command **deconv(z,y)**
- The syntax [x,r] = deconv(z,y) is preferred.
- Otherwise MATLAB may show the quotient only and skip the remainder.
- The division is shown in two row matrix. One for quotient and one for remainder.

ME 3204: Control Engineeering Sessional Spring 2016 30 / 99

Practice

Evaluate the division of $(s^2 - 1)$ by $(s + 1)$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

Evaluate the division of $(s^2 - 1)$ by $(s + 1)$

Answer: $(s - 1) + 0$

ME 3204: Control Engineeering Sessional Spring 2016 31 / 99

Curve fitting: polyfit function

From Data to Equation.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Curve fitting: polyfit function

From Data to Equation.

- **MATLAB** can form a polynomial c that fits the given data for vectors x and y with the command **polyfit(x,y,k)**

Curve fitting: polyfit function

From Data to Equation.

- **MATLAB** can form a polynomial c that fits the given data for vectors x and y with the command **polyfit(x,y,k)**
- The syntax c = polyfit(x,y,k) is preferred.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Curve fitting: polyfit function

From Data to Equation.

- **MATLAB** can form a polynomial c that fits the given data for vectors x and y with the command **polyfit(x,y,k)**
- The syntax c = polyfit(x,y,k) is preferred.
- x,y specifies the vectors of the points available for curve fitting.

Curve fitting: polyfit function

From Data to Equation.

- **MATLAB** can form a polynomial c that fits the given data for vectors x and y with the command **polyfit(x,y,k)**
- The syntax c = polyfit(x,y,k) is preferred.
- x,y specifies the vectors of the points available for curve fitting.
- k specifies the order of the desired polynomial.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Curve fitting: polyfit function

From Data to Equation.

- **MATLAB** can form a polynomial c that fits the given data for vectors x and y with the command **polyfit(x,y,k)**
- The syntax c = polyfit(x,y,k) is preferred.
- x,y specifies the vectors of the points available for curve fitting.
- k specifies the order of the desired polynomial.
- The result c will be a row vector containing the coefficients of the polynomial.

Practice

Find a second order polynomial to predict the discharge(Q) of a pump after 1.5 seconds from the given data.

t(s)	0	1	2	4
Q(L/s)	1	6	20	100

Practice

Find a second order polynomial to predict the discharge(Q) of a pump after 1.5 seconds from the given data.

t(s)	0	1	2	4
Q(L/s)	1	6	20	100

Answer:

1. t=[0 1 2 4];
2. Q=[1 6 20 100];
3. c=polyfit(t,Q,2)
4. MATLAB returns, c = 7.3409 -4.8409 1.6818
5. $\therefore Q = 7.3409t^2 - 4.8409t + 1.6818$ is the equation of discharge.
6. Use **polyval(c,1.5)** to get the answer 10.9375 L/s.

Advanced Polynomial Operations

- **MATLAB** can differentiate a polynomial y(s) analytically by the command **polyder(y)**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Advanced Polynomial Operations

- **MATLAB** can differentiate a polynomial y(s) analytically by the command **polyder(y)**
- The differentiation is shown in a row matrix.

ME 3204: Control Engineeering Sessional Spring 2016 34 / 99

Advanced Polynomial Operations

- **MATLAB** can differentiate a polynomial y(s) analytically by the command **polyder(y)**
- The differentiation is shown in a row matrix.
- **MATLAB** can integrate a polynomial y(s) and integration constant k analytically by the command **polyint(y,k)**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Advanced Polynomial Operations

- **MATLAB** can differentiate a polynomial y(s) analytically by the command **polyder(y)**
- The differentiation is shown in a row matrix.
- **MATLAB** can integrate a polynomial y(s) and integration constant k analytically by the command **polyint(y,k)**
- The integration is shown in a row matrix.

Advanced Polynomial Operations

- **MATLAB** can differentiate a polynomial y(s) analytically by the command **polyder(y)**
- The differentiation is shown in a row matrix.
- **MATLAB** can integrate a polynomial y(s) and integration constant k analytically by the command **polyint(y,k)**
- The integration is shown in a row matrix.

Practice this commands using the polynomial $s^4 + 4s^3 + 8s^2 + 16$

SIMULINK

- **Simulink** is a software package and is a graphical extension of **MATLAB**.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- **Start Simulink**
- **Type Simulink** $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

SIMULINK

- **Simulink** is a software package and is a graphical extension of **MATLAB**.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- **Start Simulink**
- **Type Simulink** $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

SIMULINK

- **Simulink** is a software package and is a graphical extension of **MATLAB**.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- **Start Simulink**
- **Type Simulink** $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

ME 3204: Control Engineeering Sessional Spring 2016 35 / 99

SIMULINK

- **Simulink** is a software package and is a graphical extension of **MATLAB**.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- **Start Simulink**
- **Type Simulink** $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

SIMULINK

- Simulink is a software package and is a graphical extension of MATLAB.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- Start Simulink
- Type Simulink $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

ME 3204: Control Engineeering Sessional Spring 2016 35 / 99

SIMULINK

- **Simulink** is a software package and is a graphical extension of **MATLAB**.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- **Start Simulink**
- **Type Simulink** $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

SIMULINK

- **Simulink** is a software package and is a graphical extension of **MATLAB**.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- **Start Simulink**
- **Type Simulink** $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

SIMULINK

- **Simulink** is a software package and is a graphical extension of **MATLAB**.
- It is very useful for modeling, simulating and analyzing dynamic systems due to its GUI.
- **Start Simulink**
- **Type Simulink** $\leftarrow$
- It can also be started by clicking its icon.
- Or use the start menu to initiate Simulink.
- Familiarize yourself with Simulink Library Browser

SIMULINK Library Browser

- Note that there are many blocks to conveniently model dynamic systems.
- Frequently used blocks are collected under **Commonly Used Blocks**.
- In this course we'll be using blocks from the following
  1. Continuous
  2. Discontinuous
  3. Math Operations
  4. Sinks
  5. Sources
  6. Signal Routing
  7. Ports and subsystems

SIMULINK Library Browser

- Note that there are many blocks to conveniently model dynamic systems.
- Frequently used blocks are collected under **Commonly Used Blocks.**
- In this course we'll be using blocks from the following

1. Continuous
2. Discontinuous
3. Math Operations
4. Sinks
5. Sources
6. Signal Routing
7. Ports and subsystems

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

SIMULINK Library Browser

- Note that there are many blocks to conveniently model dynamic systems.
- Frequently used blocks are collected under **Commonly Used Blocks.**
- In this course we'll be using blocks from the following

1. Continuous
2. Discontinuous
3. Math Operations
4. Sinks
5. Sources
6. Signal Routing
7. Ports and subsystems

ME 3204: Control Engineeering Sessional Spring 2016 36 / 99

SIMULINK Library Browser

- Note that there are many blocks to conveniently model dynamic systems.
- Frequently used blocks are collected under **Commonly Used Blocks**.
- In this course we'll be using blocks from the following
  1. Continuous
  2. Discontinuous
  3. Math Operations
  4. Sinks
  5. Sources
  6. Signal Routing
  7. Ports and subsystems

Basic Math Operations

Demonstration

Scope Vs. Display

Demonstration

1. Add two numbers and check the output in both Scope and Display
2. Subtract two numbers and check the output in both Scope and Display
3. Multiply two numbers and check the output in Scope and Display
4. Divide two numbers and check the output in Scope and Display

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$ using 'add' and 'gain' operator.

Basic Math Operations

Demonstration

Scope Vs. Display

Demonstration

1. Add two numbers and check the output in both Scope and Display
2. Subtract two numbers and check the output in both Scope and Display
3. Multiply two numbers and check the output in Scope and Display
4. Divide two numbers and check the output in Scope and Display

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$ using 'add' and 'gain' operator.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Basic Math Operations

Demonstration

Scope Vs. Display

Demonstration

1. Add two numbers and check the output in both Scope and Display
2. Subtract two numbers and check the output in both Scope and Display
3. Multiply two numbers and check the output in Scope and Display
4. Divide two numbers and check the output in Scope and Display

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$ using 'add' and 'gain' operator.

ME 3204: Control Engineeering Sessional Spring 2016 37 / 99

Basic Math Operations

Demonstration

Scope Vs. Display

Demonstration

1. Add two numbers and check the output in both Scope and Display
2. Subtract two numbers and check the output in both Scope and Display
3. Multiply two numbers and check the output in Scope and Display
4. Divide two numbers and check the output in Scope and Display

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$

Demonstration

View the response of $50 + 60sin(0.25t + \phi)$ using 'add' and 'gain' operator.

Mux

Show the three input in a single graph.

- $100 + sin(0.25t + 2)$
- $50 + sin(0.5t)$
- $75 + sin(0.75t + 1)$

System of Equations

- **Simulink** can solve both linear and nonlinear system of equations.

System of equations $\begin{cases} 2x + 3y = 13 \\ 5x - y = 7 \end{cases}$

System of Equations

Write the equations in the following form:

$$2x + 3y = 13$$

$$\Rightarrow x = \frac{13 - 3y}{2}$$

$$5x - y = 7$$

$$\Rightarrow y = 5x - 7$$

System of Equations

Write the equations in the following form:

$$2x + 3y = 13$$

$$\Rightarrow x = \frac{13 - 3y}{2}$$

$$5x - y = 7$$

$$\Rightarrow y = 5x - 7$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

System of Equations

Write the equations in the following form:

$$2x + 3y = 13$$

$$\Rightarrow x = \frac{13 - 3y}{2}$$

$$5x - y = 7$$

$$\Rightarrow y = 5x - 7$$

ME 3204: Control Engineeering Sessional Spring 2016 40 / 99

System of Equations

Write the equations in the following form:

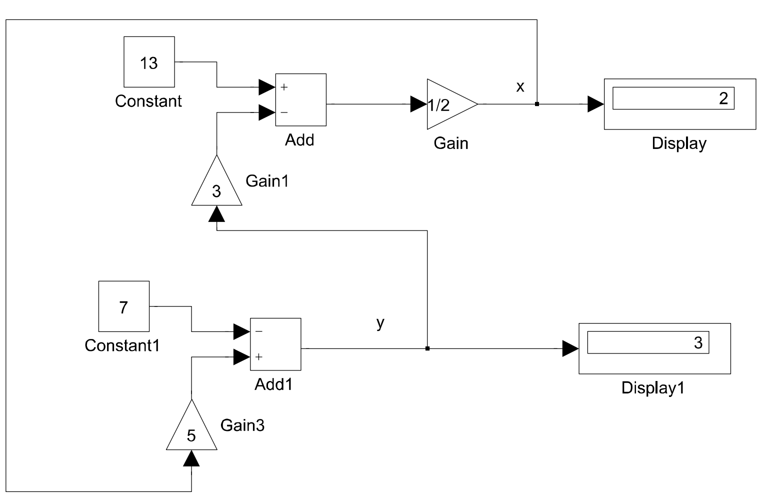
$$2x + 3y = 13$$

$$\Rightarrow x = \frac{13 - 3y}{2}$$

$$5x - y = 7$$

$$\Rightarrow y = 5x - 7$$

System of Equations



Classwork

Solve the following system.

$$\text{System 1}\begin{cases} x+3y+z=5 \\ 2x-y+2z=17 \\ 3x+4y+5z=26 \end{cases}$$

Classwork

Solve the following system.

$$\text{System 2}\begin{cases} a+b+c=6 \\ 2a+3b+4c=15 \\ a+2b-c=10 \end{cases}$$

Solution of ODE

Demonstration

If $\frac{dy}{dt} = 0.5t^2$ find y.
$y(0) = 0$

Demonstration

$\frac{dx}{dt} - 3t^2 = 2t + sin(0.25t)$; $x(0) = 0$

General System Modelling

A dynamic system can be represented in general by the differential equation:

General System Modelling

A dynamic system can be represented in general by the differential equation:

$$a_n \frac{d^n x}{dt^n} + a_{n-1} \frac{d^{n-1} x}{dt^{n-1}} + ... + \underbrace{a_2 \frac{d^2 x}{dt^2} + \underbrace{a_1 \frac{dx}{dt} + \overbrace{a_0 x = F(t)}^{\text{0th Order}}}_{\text{1st Order}}}_{\text{2nd Order}}$$

F(t) $\rightarrow$ Forcing Function
x(t) $\rightarrow$ Output or the response of the system
a's $\rightarrow$ Constants, System Parameters

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

General System Modelling

A dynamic system can be represented in general by the differential equation:

$$a_n \frac{d^n x}{dt^n} + a_{n-1} \frac{d^{n-1} x}{dt^{n-1}} + ... + \underbrace{a_2 \frac{d^2 x}{dt^2} + \underbrace{a_1 \frac{dx}{dt} + \overbrace{a_0 x = F(t)}^{\text{0th Order}}}_{\text{1st Order}}}_{\text{2nd Order}}$$

F(t) $\rightarrow$ Forcing Function
x(t) $\rightarrow$ Output or the response of the system
a's $\rightarrow$ Constants, System Parameters

Order of a system is designated by order of the differential equation.

Zeroth Order System

$$x = kF(t)$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Zeroth Order System

$$x = kF(t)$$

- $k = \frac{1}{a_0} \Leftrightarrow$
  Static sensitivity or gain. It represents scaling between the input and the output.

Zeroth Order System

$$x = kF(t)$$

- $k = \frac{1}{a_0} \Leftrightarrow$
  Static sensitivity or gain. It represents scaling between the input and the output.
- No equilibrium seeking force is present.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Zeroth Order System

$$x = kF(t)$$

- $k = \frac{1}{a_0} \Leftrightarrow$
  Static sensitivity or gain. It represents scaling between the input and the output.
- No equilibrium seeking force is present.
- Output follows the input without distortion or time lag.

Zeroth Order System

$$x = kF(t)$$

- $k = \frac{1}{a_0} \Leftrightarrow$
  Static sensitivity or gain. It represents scaling between the input and the output.
- No equilibrium seeking force is present.
- Output follows the input without distortion or time lag.
- System requires no additional dynamic considerations.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Zeroth Order System

$$x = kF(t)$$

- $k = \frac{1}{a_0} \Leftrightarrow$
  Static sensitivity or gain. It represents scaling between the input and the output.
- No equilibrium seeking force is present.
- Output follows the input without distortion or time lag.
- System requires no additional dynamic considerations.
- Represents ideal dynamic performance.

Zeroth Order System

$$x = kF(t)$$

- $k = \frac{1}{a_0} \Leftrightarrow$
  Static sensitivity or gain. It represents scaling between the input and the output.
- No equilibrium seeking force is present.
- Output follows the input without distortion or time lag.
- System requires no additional dynamic considerations.
- Represents ideal dynamic performance.
- Example: Potentiometer, ideal spring etc.

First Order System

$$a\frac{dq(t)}{dt} + bq(t) = u(t)$$

First Order System

$$a\frac{dq(t)}{dt} + bq(t) = u(t)$$

- a, b are constants or physical system parameters.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

First Order System

$$a\frac{dq(t)}{dt} + bq(t) = u(t)$$

- a, b are constants or physical system parameters.
- u(t) is the input or forcing function.

First Order System

$$a\frac{dq(t)}{dt} + bq(t) = u(t)$$

- a, b are constants or physical system parameters.
- u(t) is the input or forcing function.
- q(t) is the output or response of the system.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

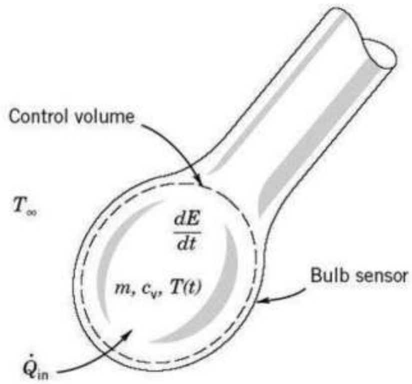
First Order System

$$a\frac{dq(t)}{dt} + bq(t) = u(t)$$

- a, b are constants or physical system parameters.
- u(t) is the input or forcing function.
- q(t) is the output or response of the system.
- May be written as

$$\tau\frac{dq(t)}{dt} + q(t) = \frac{u(t)}{b}$$

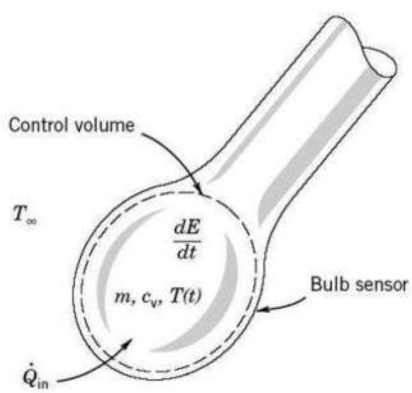
First Order System



Consider, a thermocouple initially at temperature T, exposed to ambient temperature T_α

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

First Order System

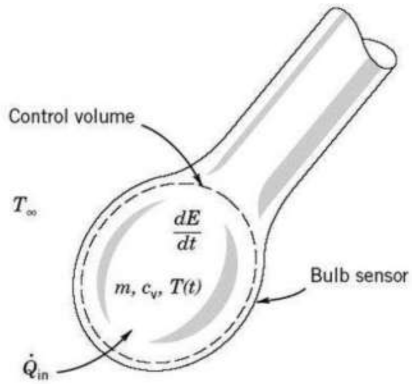


Consider, a thermocouple initially at temperature T, exposed to ambient temperature T_α

1. Convection Heat Transfer : $\dot{Q}_{in} = hA(T_\alpha - T)$

ME 3204: Control Engineeering Sessional Spring 2016 48 / 99

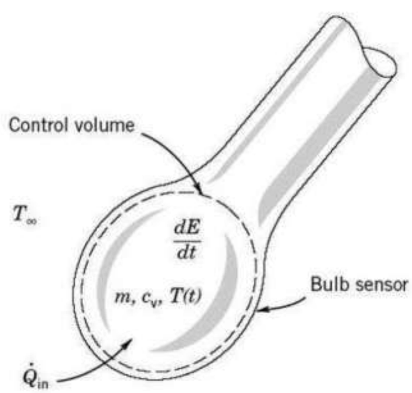
First Order System



Consider, a thermocouple initially at temperature T, exposed to ambient temperature T_α

1. Convection Heat Transfer : $\dot{Q}_{in} = hA(T_\alpha - T)$
2. Heat Transferred to Solid : $\dot{Q}_{out} = mC_v \frac{dT}{t}$

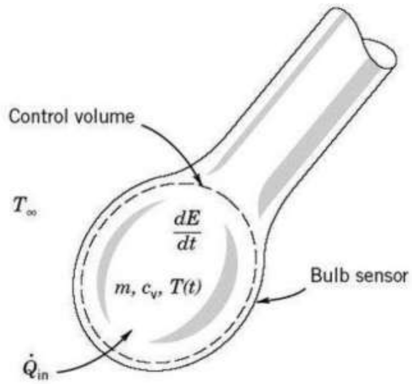
First Order System



Consider, a thermocouple initially at temperature T, exposed to ambient temperature T_α

1. Convection Heat Transfer : $\dot{Q}_{in} = hA(T_\alpha - T)$
2. Heat Transferred to Solid : $\dot{Q}_{out} = mC_v \frac{dT}{t}$
3. First Law of TD $\Rightarrow \dot{Q}_{in} = \dot{Q}_{out}$

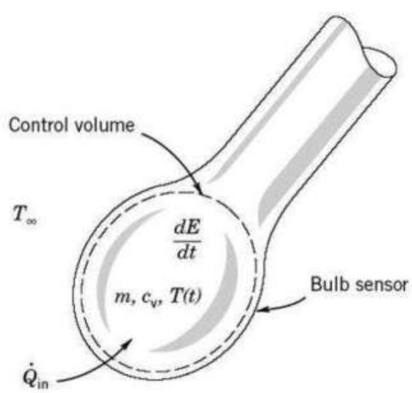
First Order System



Consider, a thermocouple initially at temperature T, exposed to ambient temperature T_α

1. Convection Heat Transfer : $\dot{Q}_{in} = hA(T_\alpha - T)$
2. Heat Transferred to Solid : $\dot{Q}_{out} = mC_v \frac{dT}{t}$
3. First Law of TD $\Rightarrow \dot{Q}_{in} = \dot{Q}_{out}$
4. $\Rightarrow hA(T_\alpha - T) = mC_v \frac{dT}{t}$

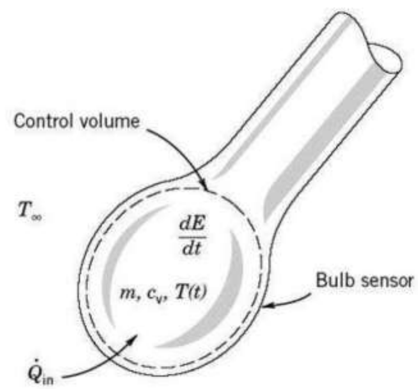
First Order System



Consider, a thermocouple initially at temperature T, exposed to ambient temperature T_α

1. Convection Heat Transfer : $\dot{Q}_{in} = hA(T_\alpha - T)$
2. Heat Transferred to Solid : $\dot{Q}_{out} = mC_v \frac{dT}{t}$
3. First Law of TD $\Rightarrow \dot{Q}_{in} = \dot{Q}_{out}$
4. $\Rightarrow hA(T_\alpha - T) = mC_v \frac{dT}{t}$
5. $\tau \frac{dT}{dt} + T = T_\alpha$

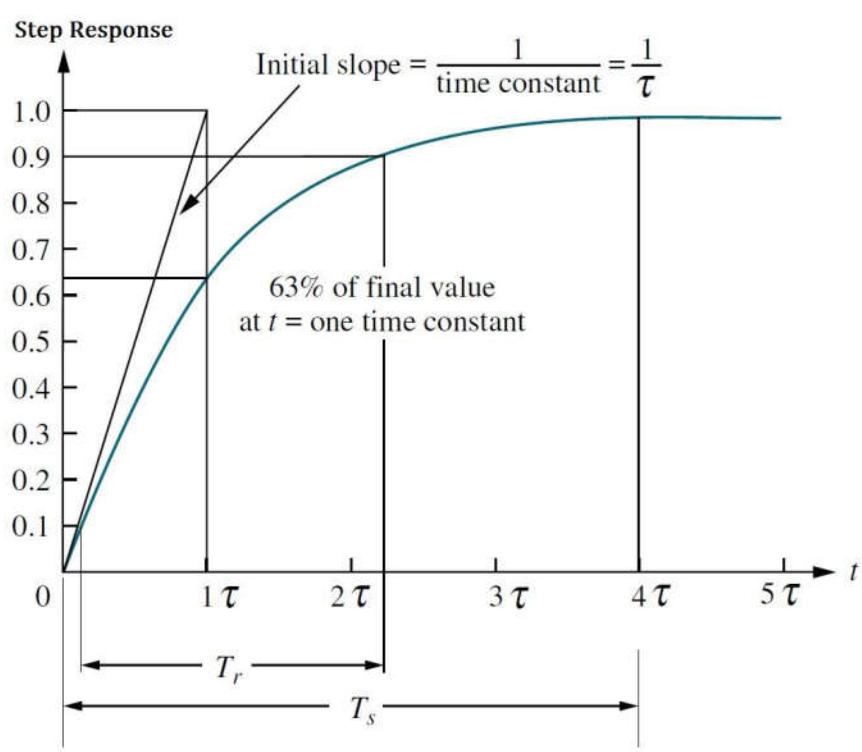
First Order System



Consider, a thermocouple initially at temperature T, exposed to ambient temperature T_α

1. *Convection Heat Transfer :* $\dot{Q}_{in} = hA(T_\alpha - T)$
2. *Heat Transferred to Solid :* $\dot{Q}_{out} = mC_v \frac{dT}{t}$
3. *First Law of TD* $\Rightarrow \dot{Q}_{in} = \dot{Q}_{out}$
4. $\Rightarrow hA(T_\alpha - T) = mC_v \frac{dT}{t}$
5. $\tau \frac{dT}{dt} + T = T_\alpha$
6. Where $\tau = \frac{mC_v}{hA}$

Response - 1st Order System: Step Input



First Order System

- **Time Constant,** τ - time required to complete 63.2% of the process.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

First Order System

- **Time Constant,** τ - time required to complete 63.2% of the process.
- **Rise Time,** T_r - time required to achieve response from 10% to 90% of final value.

First Order System

- **Time Constant,** τ - time required to complete 63.2% of the process.
- **Rise Time,** T_r - time required to achieve response from 10% to 90% of final value.
- For first order system, $T_r = 2.31\tau$ - $0.11\tau = 2.2\tau$.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

First Order System

- **Time Constant,** τ - time required to complete 63.2% of the process.
- **Rise Time,** T_r - time required to achieve response from 10% to 90% of final value.
- For first order system, $T_r = 2.31\tau$ - $0.11\tau = 2.2\tau$.
- **Settling Time,** T_s - the time for the response to reach, and stay within 2% of its final value.

ME 3204: Control Engineeering Sessional Spring 2016 50 / 99

First Order System

- **Time Constant,** τ - time required to complete 63.2% of the process.
- **Rise Time,** T_r - time required to achieve response from 10% to 90% of final value.
- For first order system, $T_r = 2.31\tau$ - $0.11\tau = 2.2\tau$.
- **Settling Time,** T_s - the time for the response to reach, and stay within 2% of its final value.
- For first order system, $T_s = 4\tau$.

First Order System

- **Time Constant,** τ - time required to complete 63.2% of the process.
- **Rise Time,** T_r - time required to achieve response from 10% to 90% of final value.
- For first order system, $T_r = 2.31\tau$ - $0.11\tau = 2.2\tau$.
- **Settling Time,** T_s - the time for the response to reach, and stay within 2% of its final value.
- For first order system, $T_s = 4\tau$.

- Process is assumed to be completed when $\tau \geq 5\tau$.

First Order System

- **Time Constant,** τ - time required to complete 63.2% of the process.
- **Rise Time,** T_r - time required to achieve response from 10% to 90% of final value.
- For first order system, $T_r = 2.31\tau$ - $0.11\tau = 2.2\tau$.
- **Settling Time,** T_s - the time for the response to reach, and stay within 2% of its final value.
- For first order system, $T_s = 4\tau$.
- Process is assumed to be completed when $\tau \geq 5\tau$.
- Faster response is associated with shorter τ.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

First Order System

- Time Constant, τ - time required to complete 63.2% of the process.
- Rise Time, T_r - time required to achieve response from 10% to 90% of final value.
- For first order system, $T_r = 2.31\tau$ - $0.11\tau = 2.2\tau$.
- Settling Time, T_s - the time for the response to reach, and stay within 2% of its final value.
- For first order system, $T_s = 4\tau$.
- Process is assumed to be completed when $\tau \geq 5\tau$.
- Faster response is associated with shorter τ.

ME 3204: Control Engineeering Sessional Spring 2016 50 / 99

Transfer Function

- **Transfer function** of a system, G(s), is defined as the ratio of the Laplace Transform (LT) of the output variable, X(s), to the LT of the input variable, F(s), with all the initial conditions are assumed to be zero.

$$G(s) = \frac{X(s)}{F(s)}$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function

- **Transfer function** of a system, G(s), is defined as the ratio of the Laplace Transform (LT) of the output variable, X(s), to the LT of the input variable, F(s), with all the initial conditions are assumed to be zero.

$$G(s) = \frac{X(s)}{F(s)}$$

- $s = \sigma + j\omega$

Transfer Function

- **Transfer function** of a system, G(s), is defined as the ratio of the Laplace Transform (LT) of the output variable, X(s), to the LT of the input variable, F(s), with all the initial conditions are assumed to be zero.

$$G(s) = \frac{X(s)}{F(s)}$$

- $s = \sigma + j\omega$
- Amplitude Ratio, $G_a = mod(G(j\omega))$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function of a 1st Order System

- For first order system, $G(s) = \frac{k}{s\tau+1}$.

Transfer Function of a 1st Order System

- For first order system, $G(s) = \frac{k}{s\tau+1}$.
- **Try the derivation**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function of a 1st Order System

- For first order system, $G(s) = \frac{k}{s\tau+1}$.
- **Try the derivation**

$$\tau \frac{dx(t)}{dt} + x(t) = kF(t)$$

ME 3204: Control Engineeering Sessional Spring 2016 52 / 99

Transfer Function of a 1st Order System

- For first order system, $G(s) = \frac{k}{s\tau+1}$.
- **Try the derivation**

$$\tau \frac{dx(t)}{dt} + x(t) = kF(t)$$

Apply Laplace Transform to both sides

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function of a 1st Order System

For first order system, $G(s) = \frac{k}{s\tau+1}$.
Try the derivation

$$\tau \frac{dx(t)}{dt} + x(t) = kF(t)$$

Apply Laplace Transform to both sides

$$\frac{d^n x(t)}{dt^n} \Rightarrow s^n X(s) \ \& \ F(t) \Rightarrow F(s)$$

Transfer Function of a 1st Order System

- For first order system, $G(s) = \frac{k}{s\tau+1}$.
- **Try the derivation**

$$\tau \frac{dx(t)}{dt} + x(t) = kF(t)$$

Apply Laplace Transform to both sides

$$\frac{d^n x(t)}{dt^n} \Rightarrow s^n X(s) \;\&\; F(t) \Rightarrow F(s)$$

$$\Rightarrow \tau s X(s) + X(s) = kF(s)$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function of a 1st Order System

- For first order system, $G(s) = \frac{k}{s\tau+1}$.
- **Try the derivation**

$$\tau \frac{dx(t)}{dt} + x(t) = kF(t)$$

Apply Laplace Transform to both sides

$$\frac{d^n x(t)}{dt^n} \Rightarrow s^n X(s) \;\&\; F(t) \Rightarrow F(s)$$

$$\Rightarrow \tau s X(s) + X(s) = kF(s)$$

$$\Rightarrow (\tau s + 1)X(s) = kF(s)$$

Transfer Function of a 1st Order System

- For first order system, $G(s) = \frac{k}{s\tau+1}$.
- **Try the derivation**

$$\tau\frac{dx(t)}{dt} + x(t) = kF(t)$$

Apply Laplace Transform to both sides

$$\frac{d^n x(t)}{dt^n} \Rightarrow s^n X(s) \ \& \ F(t) \Rightarrow F(s)$$

$$\Rightarrow \tau s X(s) + X(s) = kF(s)$$

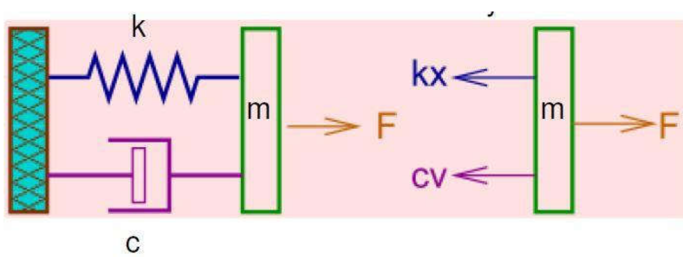
$$\Rightarrow (\tau s + 1)X(s) = kF(s)$$

$$\Rightarrow \frac{X(s)}{F(s)} = G(s) = \frac{k}{(\tau s + 1)}$$

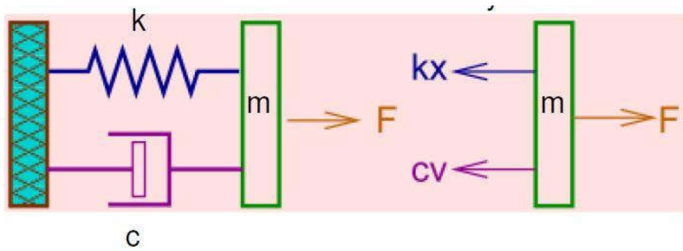
Second Order System

Write Newton's 2nd Law for the system.

$$\sum F = ma$$



Second Order System



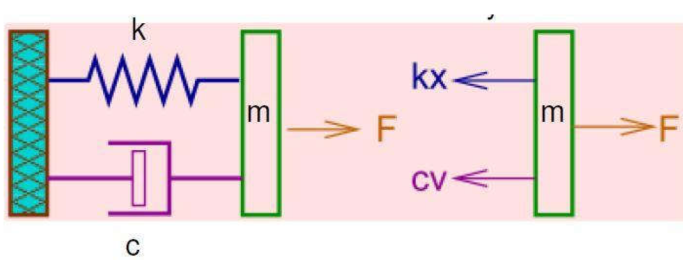
Write Newton's 2nd Law for the system.

$$\sum F = ma$$

$$\Rightarrow F - cv - kx = ma$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Second Order System



Write Newton's 2nd Law for the system.

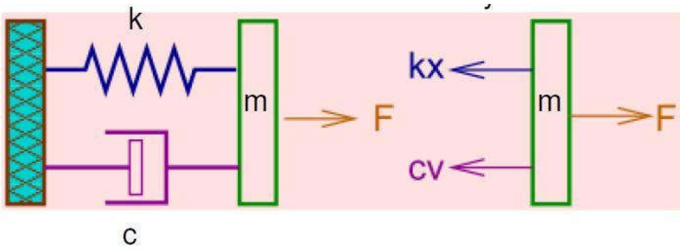
$$\sum F = ma$$

$$\Rightarrow F - cv - kx = ma$$

$$\Rightarrow ma + cv + kx = F$$

ME 3204: Control Engineeering Sessional Spring 2016 53 / 99

Second Order System



Write Newton's 2nd Law for the system.

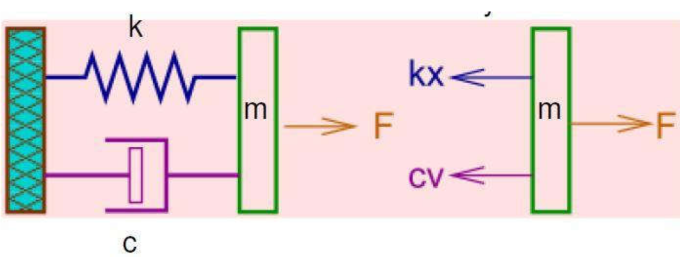
$$\sum F = ma$$

$$\Rightarrow F - cv - kx = ma$$

$$\Rightarrow ma + cv + kx = F$$

$$\Rightarrow m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = F$$

Second Order System



Write Newton's 2nd Law for the system.

$$\sum F = ma$$

$$\Rightarrow F - cv - kx = ma$$

$$\Rightarrow ma + cv + kx = F$$

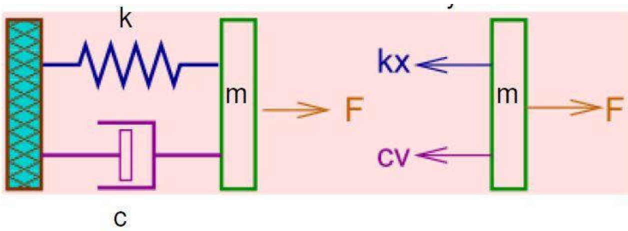
$$\Rightarrow m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = F$$

Substitute the following:

$\omega_n = \sqrt[k]{m}$ = Undamped Natural Frequency (rad/s)
$C_c = 2\sqrt{mk}$ = Critical Damping Coefficient
$\zeta = \frac{c}{C_c}$ = Damping Ratio

TF of Second Order System

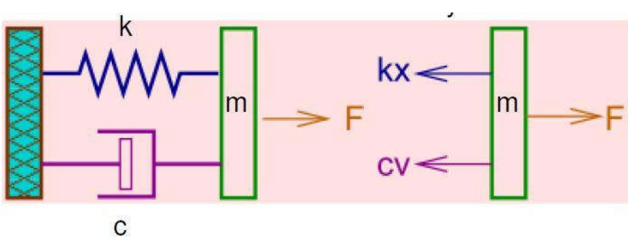
$$\Rightarrow m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = F(t)$$



TF of Second Order System

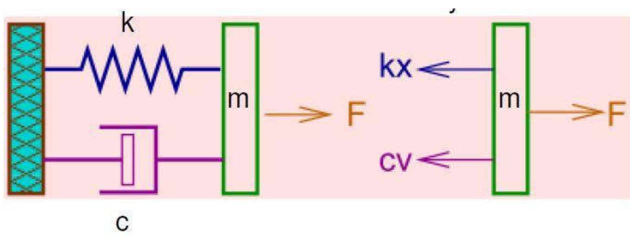
$$\Rightarrow m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = F(t)$$

$$\Rightarrow \frac{1}{\omega_n^2}\frac{d^2x}{dt} + 2\frac{\zeta}{\omega_n}\frac{dx}{dt} + x = \frac{F(t)}{k}$$



TF of Second Order System

$$\Rightarrow m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = F(t)$$



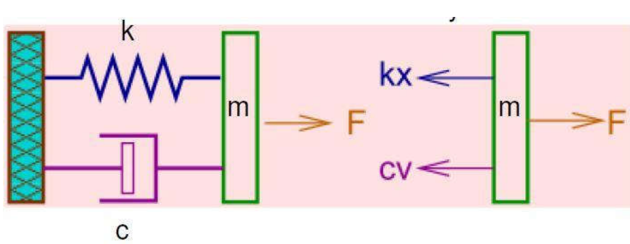
$$\Rightarrow \frac{1}{\omega_n^2}\frac{d^2x}{dt} + 2\frac{\zeta}{\omega_n}\frac{dx}{dt} + x = \frac{F(t)}{k}$$

Taking Laplace Transform of both sides.

$$\frac{d^nx(t)}{dt^n} \Rightarrow s^nX(s) \text{ \& } F(t) \Rightarrow F(s)$$

TF of Second Order System

$$\Rightarrow m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = F(t)$$



$$\Rightarrow \frac{1}{\omega_n^2}\frac{d^2x}{dt} + 2\frac{\zeta}{\omega_n}\frac{dx}{dt} + x = \frac{F(t)}{k}$$

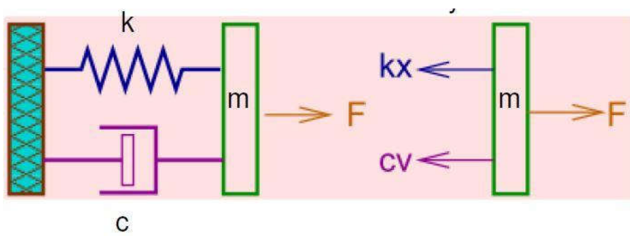
Taking Laplace Transform of both sides.

$$\frac{d^nx(t)}{dt^n} \Rightarrow s^nX(s) \text{ \& } F(t) \Rightarrow F(s)$$

$$\Rightarrow \frac{1}{\omega_n^2}s^2X(s) + 2\frac{\zeta}{\omega_n}sX(s) + X(s) = \frac{F(s)}{k}$$

TF of Second Order System

$$\Rightarrow m\frac{d^2x}{dt} + c\frac{dx}{dt} + kx = F(t)$$



$$\Rightarrow \frac{1}{\omega_n^2}\frac{d^2x}{dt} + 2\frac{\zeta}{\omega_n}\frac{dx}{dt} + x = \frac{F(t)}{k}$$

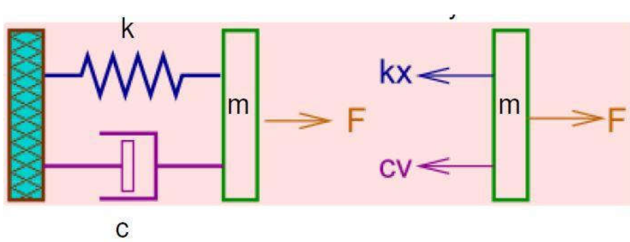
Taking Laplace Transform of both sides.

$$\frac{d^nx(t)}{dt^n} \Rightarrow s^nX(s) \ \& \ F(t) \Rightarrow F(s)$$

$$\Rightarrow \frac{1}{\omega_n^2}s^2X(s) + 2\frac{\zeta}{\omega_n}sX(s) + X(s) = \frac{F(s)}{k}$$

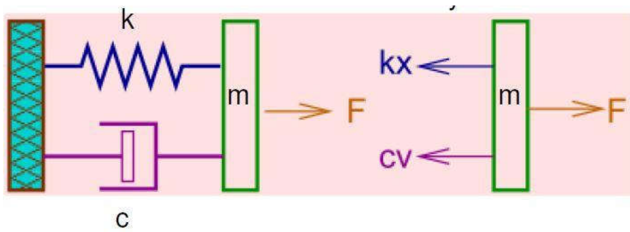
TF of Second Order System(contd.)

$$G(s) = \frac{X(s)}{F(s)} = \frac{\frac{1}{k}}{\frac{1}{\omega_n^2}s^2 + 2\frac{\zeta}{\omega_n}s + 1}$$



TF of Second Order System(contd.)

$$G(s) = \frac{X(s)}{F(s)} = \frac{\frac{1}{k}}{\frac{1}{\omega_n^2}s^2 + 2\frac{\zeta}{\omega_n}s + 1}$$

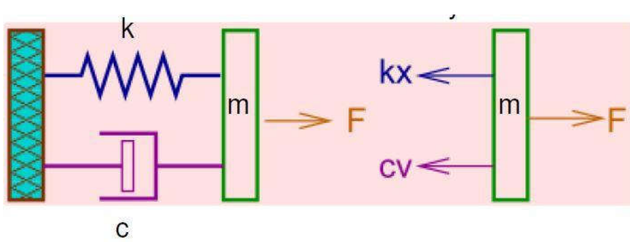


$$\Rightarrow G(s) = \frac{\frac{\omega_n^2}{k}}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

TF of Second Order System(contd.)

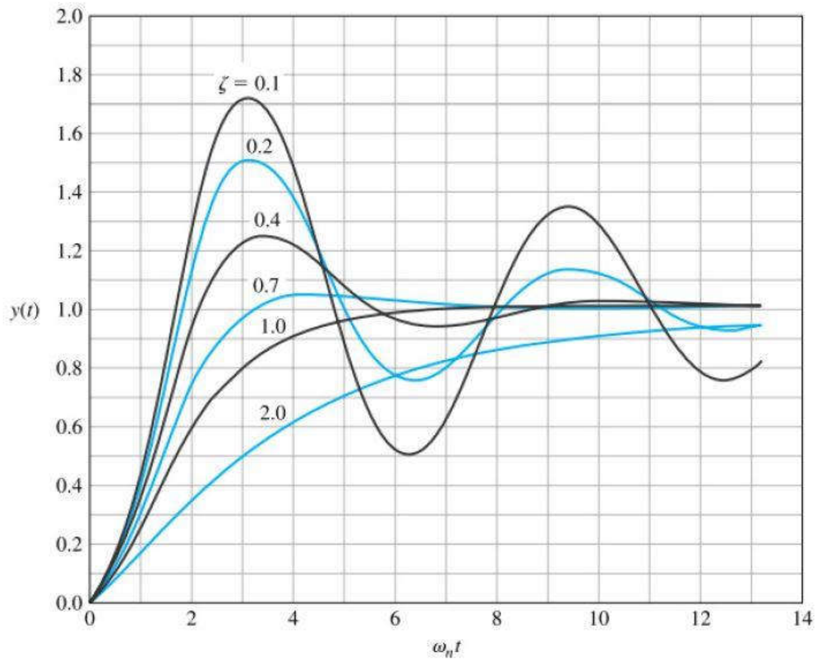
$$G(s) = \frac{X(s)}{F(s)} = \frac{\frac{1}{k}}{\frac{1}{\omega_n^2}s^2 + 2\frac{\zeta}{\omega_n}s + 1}$$



$$\Rightarrow G(s) = \frac{\frac{\omega_n^2}{k}}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

ME 3204: Control Engineeering Sessional Spring 2016 55 / 99

Response of a 2nd Order System: Step Input



Response of 2nd Order System: Step Input

- Steady state position is obtained after a long period of time.

Response of 2nd Order System: Step Input

- Steady state position is obtained after a long period of time.
- **Underdamped System ($\zeta < 1$):** response overshoots the steady-state value initially, & then eventually decays to the steady state value. The smaller ζ become , the larger is the overshoot.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Response of 2nd Order System: Step Input

- Steady state position is obtained after a long period of time.
- **Underdamped System ($\zeta < 1$):** response overshoots the steady-state value initially, & then eventually decays to the steady state value. The smaller ζ become , the larger is the overshoot.
- **Critically Damped System ($\zeta = 1$):** an exponential rise occurs to approach the steady state value without any overshoot.

Response of 2nd Order System: Step Input

- Steady state position is obtained after a long period of time.
- **Underdamped System ($\zeta < 1$):** response overshoots the steady-state value initially, & then eventually decays to the steady state value. The smaller ζ become , the larger is the overshoot.
- **Critically Damped System ($\zeta = 1$):** an exponential rise occurs to approach the steady state value without any overshoot.
- **Overdamped System ($\zeta > 1$):** response of the system approaches the steady state value without overshoot, but at a slower rate.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Response of 2nd Order System: Step Input

- Steady state position is obtained after a long period of time.
- **Underdamped System ($\zeta < 1$):** response overshoots the steady-state value initially, & then eventually decays to the steady state value. The smaller ζ become , the larger is the overshoot.
- **Critically Damped System ($\zeta = 1$):** an exponential rise occurs to approach the steady state value without any overshoot.
- **Overdamped System ($\zeta > 1$):** response of the system approaches the steady state value without overshoot, but at a slower rate.

ME 3204: Control Engineeering Sessional Spring 2016 57 / 99

Practice

1.Consider TF of a system , $G(s) = \frac{36}{s^2+4.2s+36}$ with gain k =1. Check the response in Simulink.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

1.Consider TF of a system , $G(s) = \frac{36}{s^2+4.2s+36}$ with gain k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 0.35$
Check the underdamped system response.

Practice

1.Consider TF of a system , $G(s) = \frac{36}{s^2+4.2s+36}$ with gain k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 0.35$
Check the underdamped system response.

2.Consider TF of a system , $G(s) = \frac{36}{s^2+42s+36}$ with k =1. Check the response in Simulink.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

1.Consider TF of a system , $G(s) = \frac{36}{s^2+4.2s+36}$ with gain k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 0.35$
Check the underdamped system response.

2.Consider TF of a system , $G(s) = \frac{36}{s^2+42s+36}$ with k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 3.5$
Check the overdamped system response.

Practice

1.Consider TF of a system , $G(s) = \frac{36}{s^2+4.2s+36}$ with gain k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 0.35$
Check the underdamped system response.

2.Consider TF of a system , $G(s) = \frac{36}{s^2+42s+36}$ with k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 3.5$
Check the overdamped system response.

3.Consider TF of a system , $G(s) = \frac{36}{s^2+12s+36}$ with k =1. Check the response in Simulink.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

1.Consider TF of a system , $G(s) = \frac{36}{s^2+4.2s+36}$ with gain k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 0.35$
Check the underdamped system response.

2.Consider TF of a system , $G(s) = \frac{36}{s^2+42s+36}$ with k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 3.5$
Check the overdamped system response.

3.Consider TF of a system , $G(s) = \frac{36}{s^2+12s+36}$ with k =1. Check the response in Simulink.

$\omega_n \Rightarrow 6$ & $\zeta \Rightarrow 1$
Check the critically damped system response.

Transfer Function in MATLAB

- **MATLAB** defines transfer function using the function **tf(n,d)**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function in MATLAB

- **MATLAB** defines transfer function using the function **tf(n,d)**
- A variable is used to store the transfer function.

ME 3204: Control Engineeering Sessional Spring 2016 59 / 99

Transfer Function in MATLAB

- **MATLAB** defines transfer function using the function **tf(n,d)**
- A variable is used to store the transfer function.
- ∴ Syntax is **variablename=tf(n,d)**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function in MATLAB

- **MATLAB** defines transfer function using the function **tf(n,d)**
- A variable is used to store the transfer function.
- ∴ Syntax is **variablename=tf(n,d)**
- n and d are polynomials representing the numerator and denominator of the transfer function.

Transfer Function in MATLAB

- **MATLAB** defines transfer function using the function **tf(n,d)**
- A variable is used to store the transfer function.
- ∴ Syntax is **variablename=tf(n,d)**
- n and d are polynomials representing the numerator and denominator of the transfer function.

Define a transfer function, $G(s) = \frac{5s^2+15s+10}{s^4+7s^3+20s^2+24s}$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function in MATLAB

- **MATLAB** defines transfer function using the function **tf(n,d)**
- A variable is used to store the transfer function.
- ∴ Syntax is **variablename=tf(n,d)**
- n and d are polynomials representing the numerator and denominator of the transfer function.

Define a transfer function, $G(s) = \frac{5s^2+15s+10}{s^4+7s^3+20s^2+24s}$

Answer:

1. n = [5 15 10]
2. d = [1 7 20 24 0]
3. sys = tf(n,d)

Transfer Function in MATLAB

- To perform the exact opposite, i.e., separating the numerator and denominator of a given transfer function use the following command.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function in MATLAB

- To perform the exact opposite, i.e., separating the numerator and denominator of a given transfer function use the following command.
- **[n,d] = tfdata(sys, 'v')** $\leftarrow$

Transfer Function in MATLAB

- To perform the exact opposite, i.e., separating the numerator and denominator of a given transfer function use the following command.
- **[n,d] = tfdata(sys, 'v')** $\leftarrow$
- The numerator and denominator will be returned in 'n' and 'd' variables.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function in MATLAB

- To perform the exact opposite, i.e., separating the numerator and denominator of a given transfer function use the following command.
- **[n,d] = tfdata(sys, 'v')** $\leftarrow$
- The numerator and denominator will be returned in 'n' and 'd' variables.
- 'v' is used in syntax to ensure that the polynomials are returned as row vectors and not cell arrays.

Transfer Function in MATLAB

- To perform the exact opposite, i.e., separating the numerator and denominator of a given transfer function use the following command.
- **[n,d] = tfdata(sys, 'v')** $\leftarrow$
- The numerator and denominator will be returned in 'n' and 'd' variables.
- 'v' is used in syntax to ensure that the polynomials are returned as row vectors and not cell arrays.

Practice separating the numerator and Denominator of $G(s) = \frac{5s^2+15s+10}{s^4+7s^3+20s^2+24s}$

Practice

Define a transfer function,
$$G(s) = \frac{s(s+1)(s+2)}{s(s+3)(s^2+4s+8)}$$

Practice

Define a transfer function,

$$G(s) = \frac{s(s+1)(s+2)}{s(s+3)(s^2+4s+8)}$$

Hint: Use poly and conv functions

Transfer Function in MATLAB

- **MATLAB** can also define transfer function using the following code.

Transfer Function in MATLAB

- **MATLAB** can also define transfer function using the following code.
- It is helpful when defining TF's with long numerator and denominator.

Define a transfer function, $G(s) = \frac{s+1}{s-4}$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Transfer Function in MATLAB

- **MATLAB** can also define transfer function using the following code.
- It is helpful when defining TF's with long numerator and denominator.

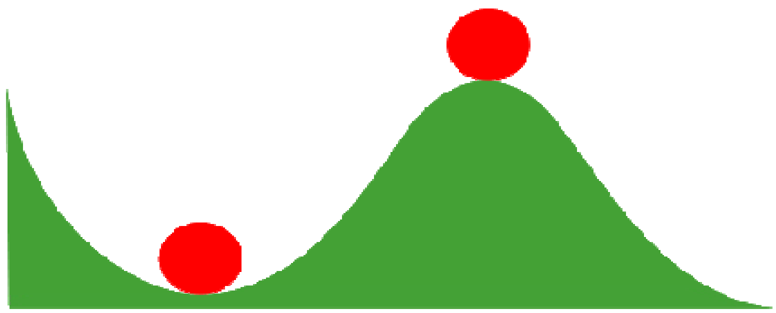
Define a transfer function, $G(s) = \frac{s+1}{s-4}$

Answer:

1. s = tf('s')
2. G = (s+1)/(s-4)

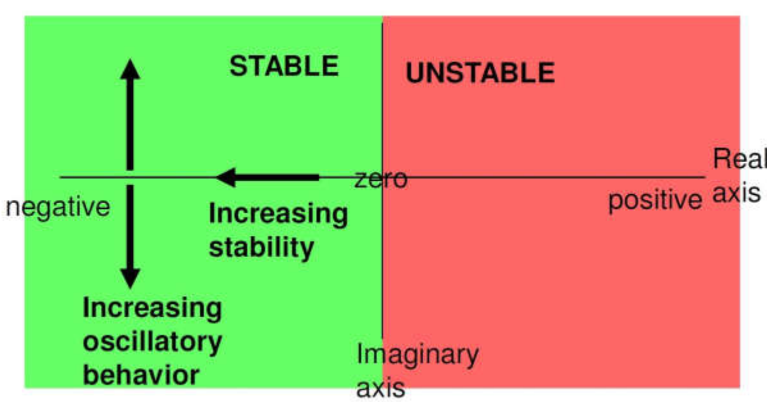
Stability Criteria

A necessary and sufficient condition for a feedback system to be stable is that all the poles of the system transfer function have negative real parts.



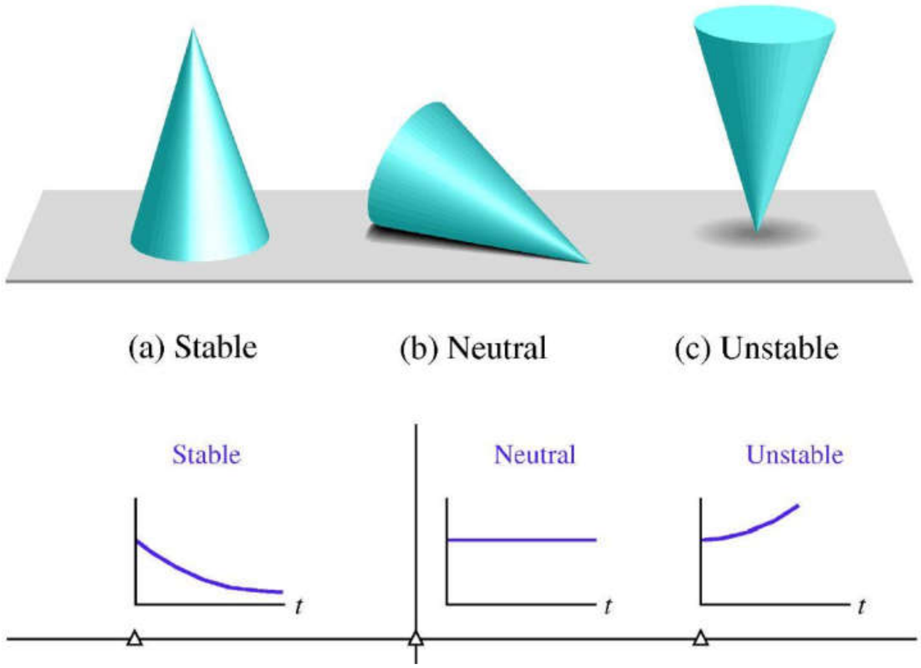
Stability Criteria

A necessary and sufficient condition for a feedback system to be stable is that all the poles of the system transfer function have negative real parts.



Stability

A stable system is a dynamic system with a bounded response to a bounded input. (BIBO)



Zeros and Poles

- Use **tf2zp** to obtain the zeros z, poles p, and gain k of the transfer function, which is defined as a ratio of two polynomials.

Zeros and Poles

- Use **tf2zp** to obtain the zeros z, poles p, and gain k of the transfer function, which is defined as a ratio of two polynomials.
- **[z,p,k] = tf2zp(n,d);** $\leftarrow$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Zeros and Poles

- Use **tf2zp** to obtain the zeros z, poles p, and gain k of the transfer function, which is defined as a ratio of two polynomials.
- **[z,p,k] = tf2zp(n,d);** $\leftarrow$
- To obtain the transfer function, as a ratio of two polynomials when zeros z, poles p and gain k are known use **zp2tf**.

Zeros and Poles

- Use **tf2zp** to obtain the zeros z, poles p, and gain k of the transfer function, which is defined as a ratio of two polynomials.
- **[z,p,k] = tf2zp(n,d);** ←
- To obtain the transfer function, as a ratio of two polynomials when zeros z, poles p and gain k are known use **zp2tf**.
- **[n,d] = zp2tf(z,p,k);**

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Zeros and Poles

- Use **tf2zp** to obtain the zeros z, poles p, and gain k of the transfer function, which is defined as a ratio of two polynomials.
- **[z,p,k] = tf2zp(n,d);** ←
- To obtain the transfer function, as a ratio of two polynomials when zeros z, poles p and gain k are known use **zp2tf**.
- **[n,d] = zp2tf(z,p,k);**
- The command **pzmap(n,d)** plots the **pole-zero map** of a given transfer function.

Practice

Zero, Pole, Pole-zero map, Stability

Find the location of zeros and poles and plot the pole-zero map of , $G(s) = \frac{2s^3+8s+6}{s^4+6s^3+12s^2+24s}$
Is the system stable?

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

Zero, Pole, Pole-zero map, Stability

Find the location of zeros and poles and plot the pole-zero map of , $G(s) = \frac{2s^3+8s+6}{s^4+6s^3+12s^2+24s}$
Is the system stable?

Answer: In column matrix form,
z= -3, -1, p= 0, -4.5198, $-0.7401 \pm 2.1822i$, k = 2. No.

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.
5. **step(sys)** ⇒ Shows step response.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.
5. **step(sys)** ⇒ Shows step response.
6. **stepinfo(sys)** ⇒ Rise time, settling time etc.

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.
5. **step(sys)** ⇒ Shows step response.
6. **stepinfo(sys)** ⇒ Rise time, settling time etc.
7. **isstable(sys)** ⇒ Returns 0 if system is unstable.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.
5. **step(sys)** ⇒ Shows step response.
6. **stepinfo(sys)** ⇒ Rise time, settling time etc.
7. **isstable(sys)** ⇒ Returns 0 if system is unstable.
8. **zpk** ⇒ Creates zero-pole gain model.

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.
5. **step(sys)** ⇒ Shows step response.
6. **stepinfo(sys)** ⇒ Rise time, settling time etc.
7. **isstable(sys)** ⇒ Returns 0 if system is unstable.
8. **zpk** ⇒ Creates zero-pole gain model.
9. **impulse(sys)** ⇒ Shows impulse response.

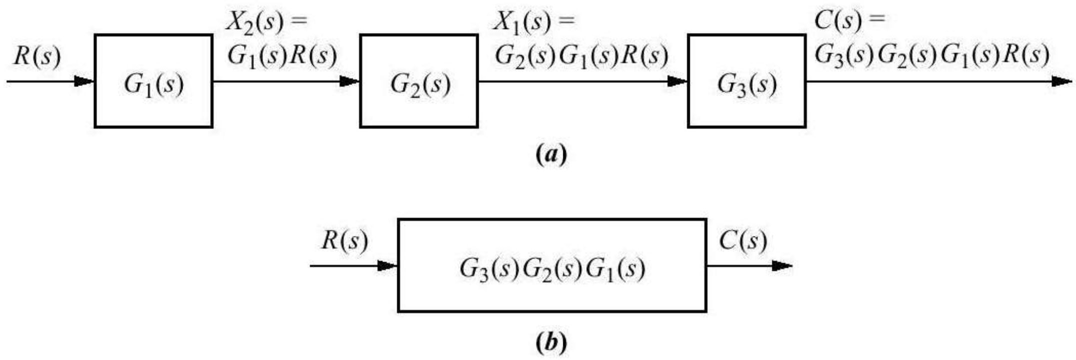
Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.
5. **step(sys)** ⇒ Shows step response.
6. **stepinfo(sys)** ⇒ Rise time, settling time etc.
7. **isstable(sys)** ⇒ Returns 0 if system is unstable.
8. **zpk** ⇒ Creates zero-pole gain model.
9. **impulse(sys)** ⇒ Shows impulse response.
10. **series(sys1,sys2)** ⇒ Equivalent series transfer function.

Some useful functions for control systems

1. **tf(n,d)** ⇒ Defines transfer functions.
2. **pzmap(sys)** ⇒ Draws the pole-zero map.
3. **bode(sys)** ⇒ Draws the bode plot.
4. **rlocus(sys)** ⇒ Draws the root-locus plot.
5. **step(sys)** ⇒ Shows step response.
6. **stepinfo(sys)** ⇒ Rise time, settling time etc.
7. **isstable(sys)** ⇒ Returns 0 if system is unstable.
8. **zpk** ⇒ Creates zero-pole gain model.
9. **impulse(sys)** ⇒ Shows impulse response.
10. **series(sys1,sys2)** ⇒ Equivalent series transfer function.
11. **parallel(sys1,sys2)** ⇒ Equivalent parallel transfer function.
12. **feedback(sys1,sys2)** ⇒ Equivalent feedback transfer function.

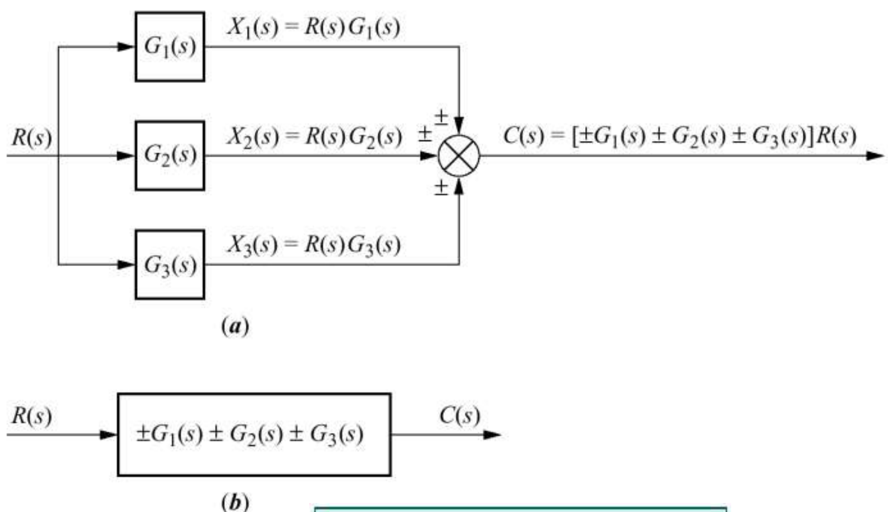
TF manipulation: Series



$$G_e(s) = G_3(s)G_2(s)G_1(s)$$

Homework: Take $G = \frac{1}{s+1}$ and $H = \frac{1}{s-1}$ to check **series(G,H)**.

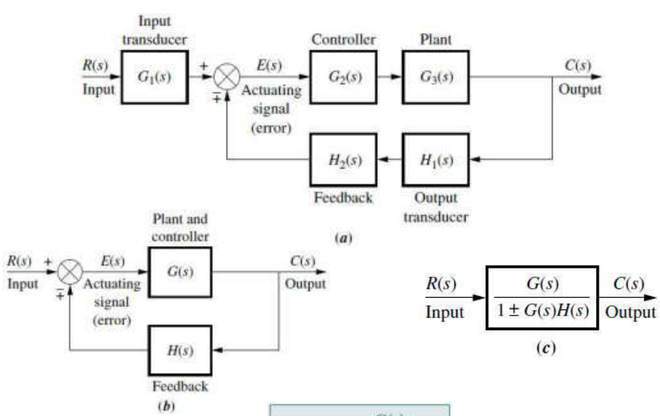
TF manipulation: Parallel



$G_e(s) = \pm G_1(s) \pm G_2(s) \pm G_3(s)$

Homework: Take $G = \frac{1}{s+1}$ and $H = \frac{1}{s-1}$ to check **parallel(G,H)**.

TF manipulation: Feedback



$G_e(s) = \frac{G(s)}{1 \pm G(s)H(s)}$

Figure: a) Feedback Control System.
b) Simplified Model.
c) Equivalent Transfer Function.

**Closed Loop System H=1

Homework: Consider,

$$G = \frac{1}{s+1}$$

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

- Negative feedback system: $G_{eqv} = \frac{G}{1+GH}$ use **feedback(G,H)** [a].
- Positive feedback system: $G_{eqv} = \frac{G}{1-GH}$ use **feedback(G,H,+1)**

[a] same as feedback(G,H,-1)

Laplace Transform ($\mathscr{L}$) in MATLAB

- Laplace transform helps to convert differential equations which describes the behavior of a dynamic systems into algebraic equations of a complex variable ’s’.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Laplace Transform ($\mathscr{L}$) in MATLAB

- Laplace transform helps to convert differential equations which describes the behavior of a dynamic systems into algebraic equations of a complex variable ’s’.
- Differentiation and integration are thus replaced by by algebraic operation in complex plane.

Laplace Transform ($\mathscr{L}$) in MATLAB

- Laplace transform helps to convert differential equations which describes the behavior of a dynamic systems into algebraic equations of a complex variable 's'.
- Differentiation and integration are thus replaced by by algebraic operation in complex plane.
- Both *transient* and *steady-state* component of the solution are obtained simultaneously.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Laplace Transform ($\mathscr{L}$) in MATLAB

- Laplace transform helps to convert differential equations which describes the behavior of a dynamic systems into algebraic equations of a complex variable 's'.
- Differentiation and integration are thus replaced by by algebraic operation in complex plane.
- Both *transient* and *steady-state* component of the solution are obtained simultaneously.
- From Definition :

$$\mathscr{L}[f(t)] = \int_0^\infty e^{-st}[f(t)]dt$$

Laplace Transform ($\mathscr{L}$) in MATLAB

- Laplace transform helps to convert differential equations which describes the behavior of a dynamic systems into algebraic equations of a complex variable 's'.
- Differentiation and integration are thus replaced by by algebraic operation in complex plane.
- Both *transient* and *steady-state* component of the solution are obtained simultaneously.
- From Definition :

$$\mathscr{L}[f(t)] = \int_0^\infty e^{-st}[f(t)]dt$$

Practice

- Determine the Laplace Transform of $f(t) = e^{-t}(1 - sin(t))$

Practice

- Determine the Laplace Transform of $f(t) = e^{-t}(1 - sin(t))$

Use the following command:

syms t ←

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

- Determine the Laplace Transform of $f(t) = e^{-t}(1 - sin(t))$

Use the following command:

syms t ←
ft = exp(-t) * (1 $- sin(t)$); ←

ME 3204: Control Engineeering Sessional Spring 2016 73 / 99

Practice

- Determine the Laplace Transform of $f(t) = e^{-t}(1 - sin(t))$

Use the following command:

syms t ←
ft = exp(-t) * (1 $-$ $sin(t)$); ←
fs = laplace(ft)←;

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

- Determine the Laplace Transform of $f(t) = e^{-t}(1 - sin(t))$

Use the following command:

syms t ←
ft = exp(-t) * (1 $-$ $sin(t)$); ←
fs = laplace(ft)←;
The result is shown as
fs = 1/(1+s) - 1/(2 + 2s + s^2)

ME 3204: Control Engineeering Sessional Spring 2016 73 / 99

Practice

- Determine the Laplace Transform of $f(t) = e^{-t}(1 - sin(t))$

Use the following command:

syms t ←
ft = exp(-t) * (1 $- sin(t)$); ←
fs = laplace(ft)←;
The result is shown as
fs = 1/(1+s) - 1/(2 + 2s + s^2)
i.e.,
$F(s) = \frac{1}{s+1} - \frac{1}{((s+1)^2+1)}$

Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).

Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).
- From Definition :

$$\mathscr{L}^{-1}[F(s)] = f(t)$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).
- From Definition :

$$\mathscr{L}^{-1}[F(s)] = f(t)$$

- Determine the Inverse Laplace Transform of $F(s) = \frac{1}{s+4}$

Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).
- From Definition :

$$\mathscr{L}^{-1}[F(s)] = f(t)$$

- Determine the Inverse Laplace Transform of $F(s) = \frac{1}{s+4}$

```
syms s t ←
```


Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).
- From Definition :

$$\mathscr{L}^{-1}[F(s)] = f(t)$$

- Determine the Inverse Laplace Transform of $F(s) = \frac{1}{s+4}$

```
syms s t ←
fs = 1/ (s+4) ←
```

Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).
- From Definition :

$$\mathscr{L}^{-1}[F(s)] = f(t)$$

- Determine the Inverse Laplace Transform of $F(s) = \frac{1}{s+4}$

```
syms s t ←
fs = 1/ (s+4) ←
ft = ilaplace(fs)←;
```


Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).
- From Definition :

$$\mathscr{L}^{-1}[F(s)] = f(t)$$

- Determine the Inverse Laplace Transform of $F(s) = \frac{1}{s+4}$

```
syms s t ←
fs = 1/ (s+4) ←
ft = ilaplace(fs)←;
The result is shown as
ft = exp(-4*t)
```

Inverse Laplace Transform ($\mathscr{L}^{-1}$) in MATLAB

- Inverse Laplace transform converts the Laplace Transform F(s) to the time function f(t).
- From Definition :

$$\mathscr{L}^{-1}[F(s)] = f(t)$$

- Determine the Inverse Laplace Transform of $F(s) = \frac{1}{s+4}$

syms s t ←
fs = 1/ (s+4) ←
ft = ilaplace(fs)←;
The result is shown as
ft = exp(-4*t)
i.e.,
$F(t) = e^{-4t}$

Residue Function

Determine the Partial Fraction expansion of $F(s) = \frac{s^3+9s+1}{s^3+s^2+2s+2}$

Residue Function

Determine the Partial Fraction expansion of $F(s) = \frac{s^3+9s+1}{s^3+s^2+2s+2}$

1. n = [1 0 9 1]; $\leftarrow$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Residue Function

Determine the Partial Fraction expansion of $F(s) = \frac{s^3+9s+1}{s^3+s^2+2s+2}$

1. n = [1 0 9 1]; $\leftarrow$
2. d = [1 1 2 2]; $\leftarrow$

Residue Function

Determine the Partial Fraction expansion of $F(s) = \frac{s^3+9s+1}{s^3+s^2+2s+2}$

1. n = [1 0 9 1]; ←
2. d = [1 1 2 2]; ←
3. [r, p, k] = residue(n,d) ←

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Residue Function

Determine the Partial Fraction expansion of $F(s) = \frac{s^3+9s+1}{s^3+s^2+2s+2}$

1. n = [1 0 9 1]; ←
2. d = [1 1 2 2]; ←
3. [r, p, k] = residue(n,d) ←

MATLAB outputs the following

r =
1.0000 - 1.7678i
1.0000 + 1.7678i
-3.0000
p =
0.0000 - 1.4142i
0.0000 + 1.4142i
-1
k =
1

Residue Function

MATLAB output:
r =
1.0000 - 1.7678i
1.0000 + 1.7678i
-3.0000
p =
0.0000 - 1.4142i
0.0000 + 1.4142i
-1
k =
1

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Residue Function

MATLAB output:
r =
1.0000 - 1.7678i
1.0000 + 1.7678i
-3.0000
p =
0.0000 - 1.4142i
0.0000 + 1.4142i
-1
k =
1

∴ the partial fraction expansion is

$$F(s) = \frac{s^3 + 9s + 1}{s^3 + s^2 + 2s + 2} = 1 + \frac{1.0000 - 1.7678i}{s - 1.4142i} + \frac{1.0000 + 1.7678i}{s + 1.4142i} - \frac{3}{s+1}$$

ME 3204: Control Engineeering Sessional Spring 2016 76 / 99

Practice

Determine the Partial Fraction expansion of $F(s) = \frac{s+1}{s^4+7s^3+16s^2+12s}$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Practice

Determine the Partial Fraction expansion of $F(s) = \frac{s+1}{s^4+7s^3+16s^2+12s}$

Answer: $F(s) = \frac{0.6667}{s+3} + \frac{-0.75}{s+2} + \frac{0.5}{(s+2)^2} + \frac{0.0833}{s+0}$

Reverse Residue

$$F(s) = 1 + \frac{1.0000 - 1.7678i}{s - 1.4142i} + \frac{1.0000 + 1.7678i}{s + 1.4142i} - \frac{3}{s+1}$$

Find the numerator and denominator of F(s).

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Reverse Residue

$$F(s) = 1 + \frac{1.0000 - 1.7678i}{s - 1.4142i} + \frac{1.0000 + 1.7678i}{s + 1.4142i} - \frac{3}{s+1}$$

Find the numerator and denominator of F(s).

1. define r,p and k first.
2. $[n, d]$ = residue(r,p,k) = $\leftarrow$

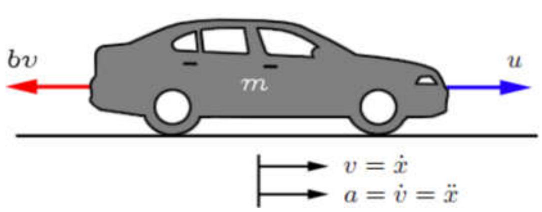
Reverse Residue

$$F(s) = 1 + \frac{1.0000 - 1.7678i}{s - 1.4142i} + \frac{1.0000 + 1.7678i}{s + 1.4142i} - \frac{3}{s+1}$$

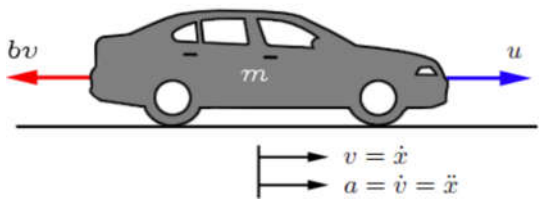
Find the numerator and denominator of F(s).

1. define r,p and k first.
2. $[n, d]$ = residue(r,p,k) = $\leftarrow$

Car Cruise Control

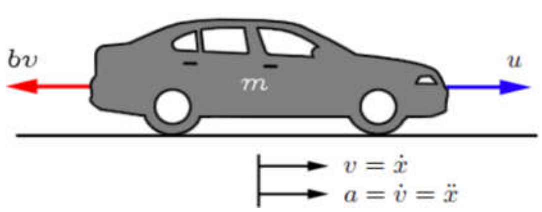


Car Cruise Control



Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

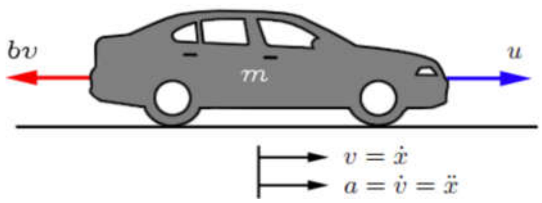
Car Cruise Control



Consider, a simple cruise control system with the assumption that rolling resistance and air drag are proportional to the car's speed (v).

Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

Car Cruise Control

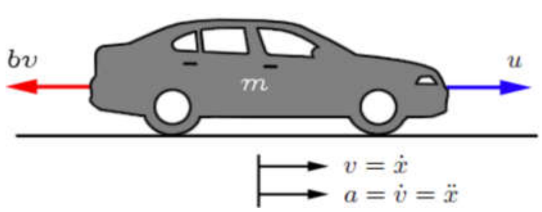


Consider, a simple cruise control system with the assumption that rolling resistance and air drag are proportional to the car's speed (v).

- *Newton's 2nd Law :* $\sum F = ma$

Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

Car Cruise Control

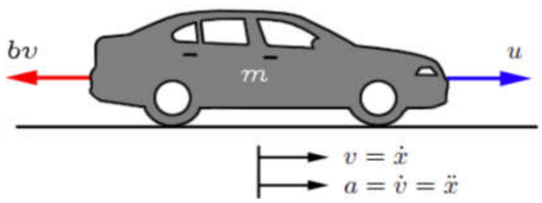


Consider, a simple cruise control system with the assumption that rolling resistance and air drag are proportional to the car's speed (v).

- *Newton's 2nd Law :* $\sum F = ma$
- *u is the force generated between the road/tire interface and can be controlled directly.*

Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

Car Cruise Control

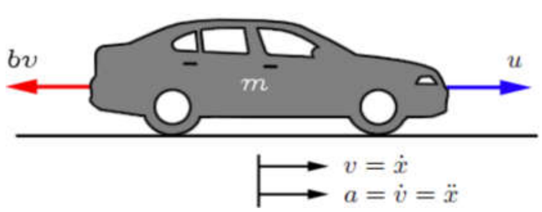


Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

Consider, a simple cruise control system with the assumption that rolling resistance and air drag are proportional to the car's speed (v).

- *Newton's 2nd Law :* $\sum F = ma$
- *u is the force generated between the road/tire interface and can be controlled directly.*
- $\Rightarrow ma = u - bv$

Car Cruise Control

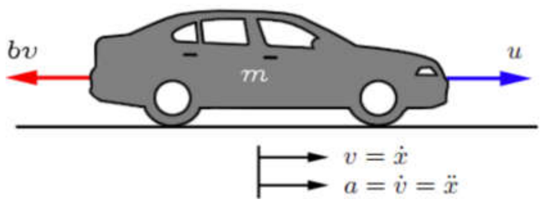


Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

Consider, a simple cruise control system with the assumption that rolling resistance and air drag are proportional to the car's speed (v).

- *Newton's 2nd Law :* $\sum F = ma$
- *u is the force generated between the road/tire interface and can be controlled directly.*
- $\Rightarrow ma = u - bv$
- $\Rightarrow m\frac{dv}{dt} = u - bv$

Car Cruise Control



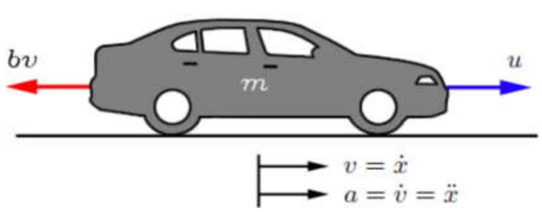
Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

Consider, a simple cruise control system with the assumption that rolling resistance and air drag are proportional to the car's speed (v).

- *Newton's 2nd Law :* $\sum F = ma$
- *u is the force generated between the road/tire interface and can be controlled directly.*
- $\Rightarrow ma = u - bv$
- $\Rightarrow m\frac{dv}{dt} = u - bv$
- $\Rightarrow \frac{dv}{dt} = \frac{1}{m} \times (u - bv)$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Car Cruise Control



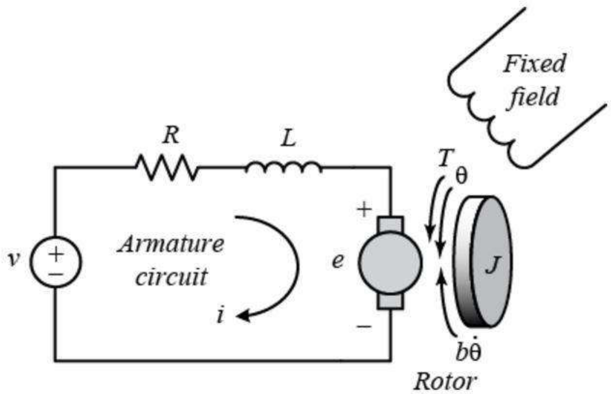
Physical Parameters
u = force generated between the road/tire interface = 500 N
b = rolling resistance proportionality constant = 50 Ns/m
m = mass of the car = 1000 kg

Consider, a simple cruise control system with the assumption that rolling resistance and air drag are proportional to the car's speed (v).

- *Newton's 2nd Law :* $\sum F = ma$
- *u is the force generated between the road/tire interface and can be controlled directly.*
- $\Rightarrow ma = u - bv$
- $\Rightarrow m\frac{dv}{dt} = u - bv$
- $\Rightarrow \frac{dv}{dt} = \frac{1}{m} \times (u - bv)$

$$\therefore \frac{dv}{dt} = \frac{1}{m} \times (u - bv)$$

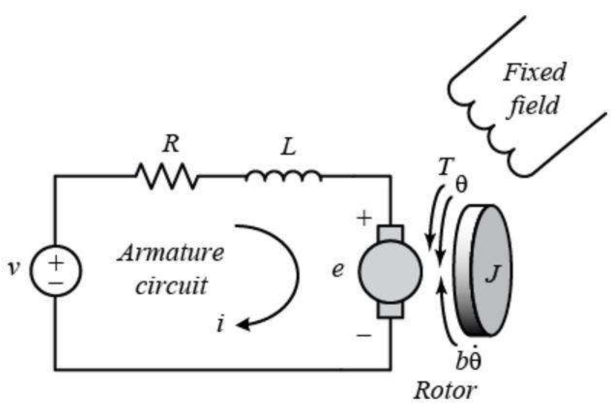
Response of a DC motor



- All physical systems can be modeled using differential equations.

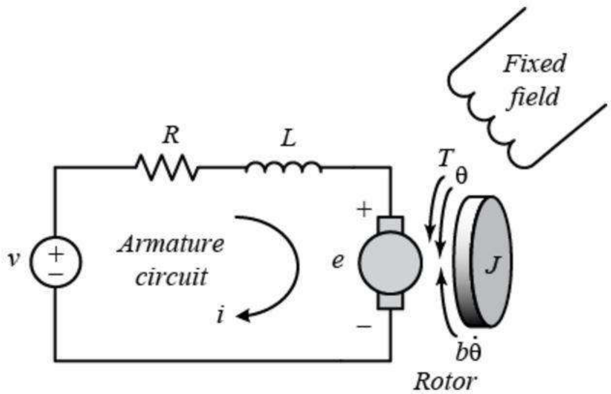
Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Response of a DC motor



- All physical systems can be modeled using differential equations.
- Response of such systems can be found by simultaneously solving the governing DEQ's.

Response of a DC motor

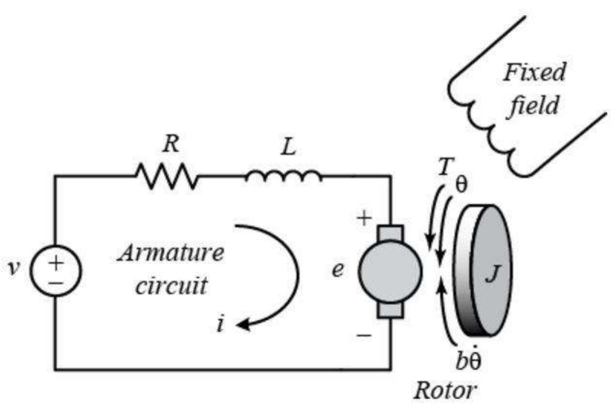


- All physical systems can be modeled using differential equations.
- Response of such systems can be found by simultaneously solving the governing DEQ's.

Physical Parameters

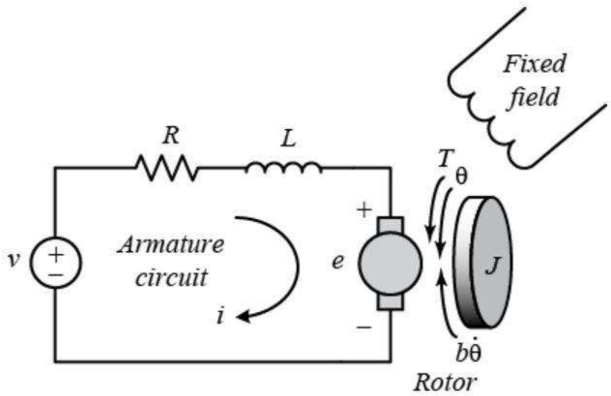
J = moment of inertia of the rotor (kgm^2)
b = motor viscous friction constant (Nms)
K_e = electromotive force constant (V/rad/sec)
K_t = motor torque constant (Nm/Amp)
R = electric resistance (Ohm)
L = electric inductance (H)

Response of a DC motor(contd.)



From Newton's 2nd Law of Motion, $\sum F = ma$

Response of a DC motor(contd.)

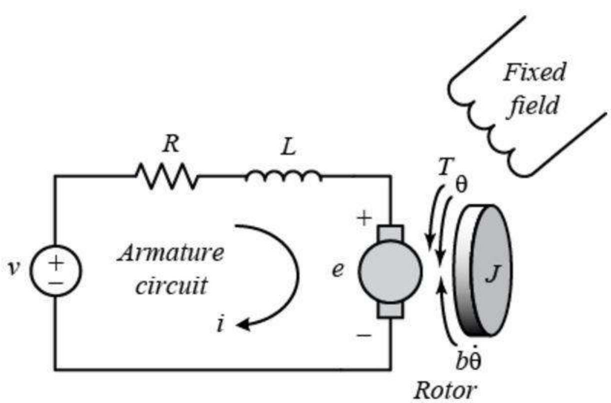


From Newton's 2nd Law of Motion, $\sum F = ma$

For rotational systems, $\Rightarrow \sum T = J\alpha$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Response of a DC motor(contd.)

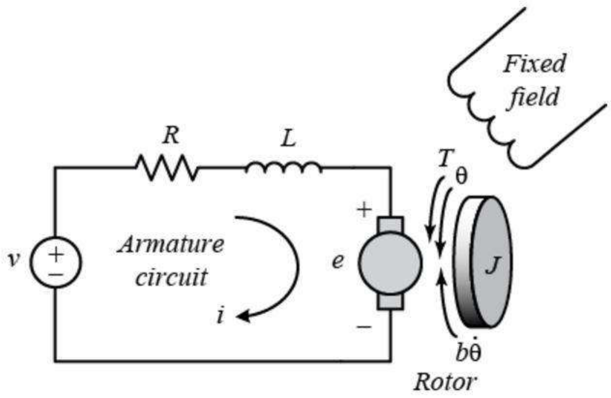


From Newton's 2nd Law of Motion, $\sum F = ma$

For rotational systems, $\Rightarrow \sum T = J\alpha$

$$\Rightarrow T - b\frac{d\theta}{dt} = J\frac{d^2\theta}{dt^2}$$

Response of a DC motor(contd.)



From Newton's 2nd Law of Motion, $\sum F = ma$

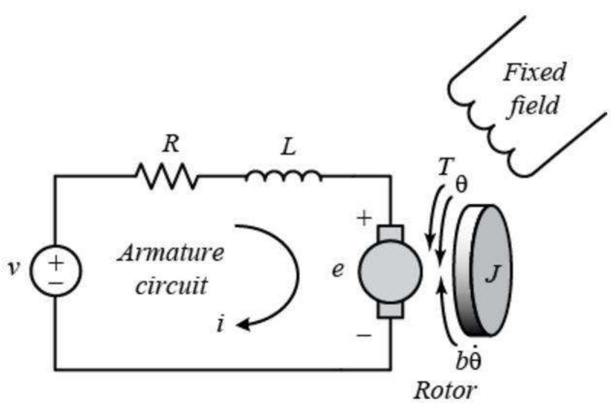
For rotational systems, $\Rightarrow \sum T = J\alpha$

$$\Rightarrow T - b\frac{d\theta}{dt} = J\frac{d^2\theta}{dt^2}$$

Motor Torque is proportional to Armature Current.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Response of a DC motor(contd.)



From Newton's 2nd Law of Motion, $\sum F = ma$

For rotational systems, $\Rightarrow \sum T = J\alpha$

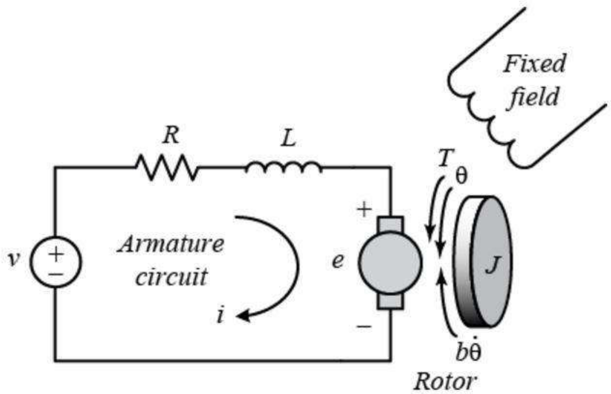
$$\Rightarrow T - b\frac{d\theta}{dt} = J\frac{d^2\theta}{dt^2}$$

Motor Torque is proportional to Armature Current.

$$\therefore T = K_t i$$

ME 3204: Control Engineeering Sessional Spring 2016 81 / 99

Response of a DC motor(contd.)



From Newton's 2nd Law of Motion, $\sum F = ma$

For rotational systems, $\Rightarrow \sum T = J\alpha$

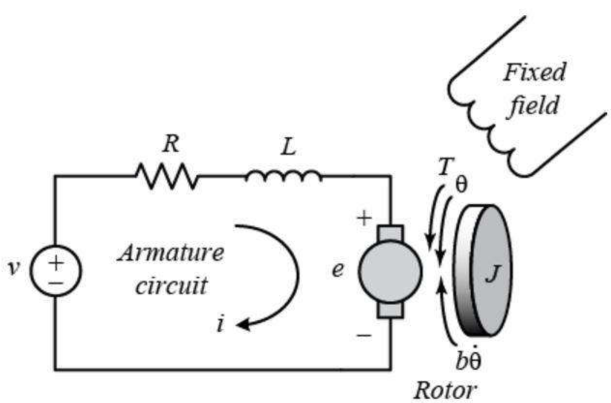
$$\Rightarrow T - b\frac{d\theta}{dt} = J\frac{d^2\theta}{dt^2}$$

Motor Torque is proportional to Armature Current.

$$\therefore T = K_t i$$

$$\Rightarrow K_t i - b\frac{d\theta}{dt} = J\frac{d^2\theta}{dt^2}$$

Response of a DC motor(contd.)



From Newton's 2nd Law of Motion, $\sum F = ma$

For rotational systems, $\Rightarrow \sum T = J\alpha$

$$\Rightarrow T - b\frac{d\theta}{dt} = J\frac{d^2\theta}{dt^2}$$

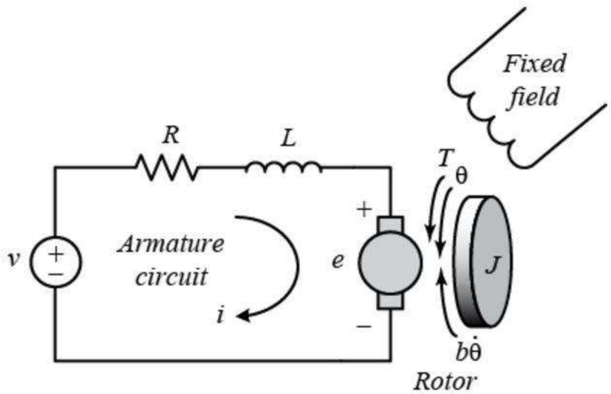
Motor Torque is proportional to Armature Current.

$$\therefore T = K_t i$$

$$\Rightarrow K_t i - b\frac{d\theta}{dt} = J\frac{d^2\theta}{dt^2}$$

$$\therefore \frac{d^2\theta}{dt^2} = \frac{1}{J}(K_t i - b\frac{d\theta}{dt}) \quad (1)$$

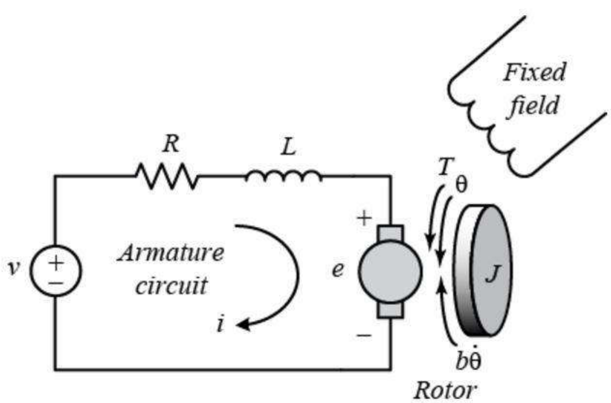
Response of a DC motor(contd.)



From Kirchoff's Voltage Law, $\sum V = 0$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Response of a DC motor(contd.)

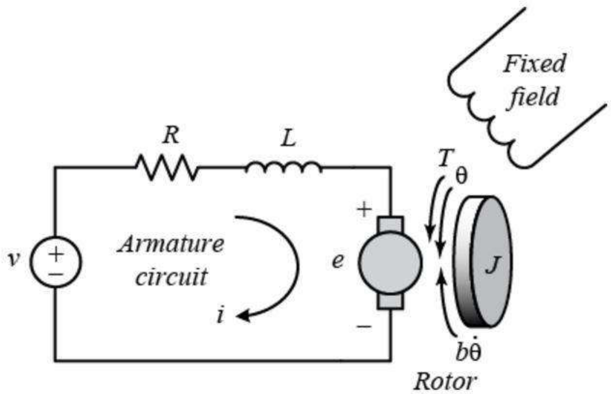


From Kirchoff's Voltage Law, $\sum V = 0$

$$\Rightarrow V = L\frac{di}{dt} + iR + e$$

ME 3204: Control Engineeering Sessional Spring 2016 82 / 99

Response of a DC motor(contd.)



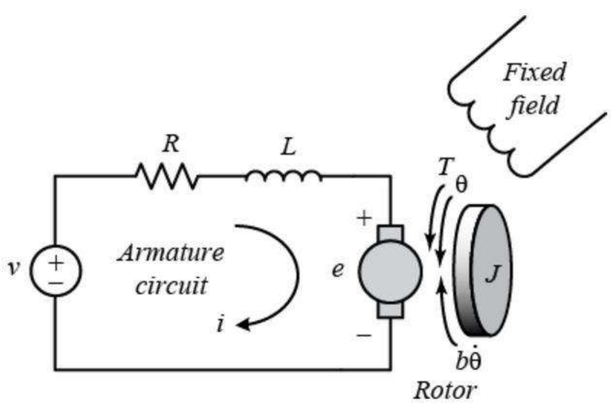
From Kirchoff's Voltage Law, $\sum V = 0$

$$\Rightarrow V = L\frac{di}{dt} + iR + e$$

Back emf(e),is proportional to the angular velocity of shaft.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Response of a DC motor(contd.)



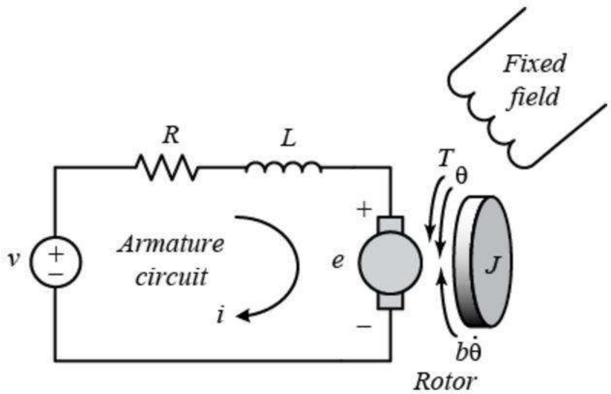
From Kirchoff's Voltage Law, $\sum V = 0$

$$\Rightarrow V = L\frac{di}{dt} + iR + e$$

Back emf(e),is proportional to the angular velocity of shaft.

$$\therefore e = K_e\frac{d\theta}{dt}$$

Response of a DC motor(contd.)



From Kirchoff's Voltage Law, $\sum V = 0$

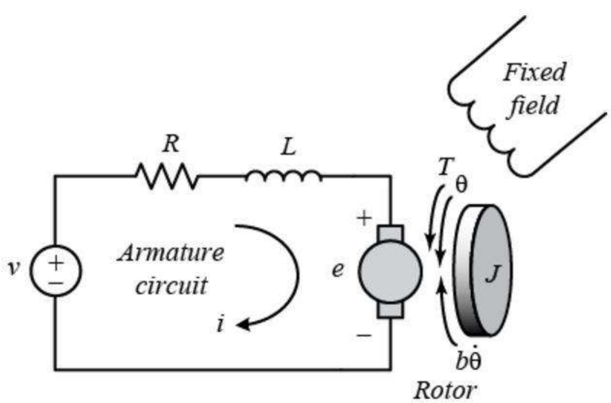
$$\Rightarrow V = L\frac{di}{dt} + iR + e$$

Back emf(e),is proportional to the angular velocity of shaft.

$$\therefore e = K_e\frac{d\theta}{dt}$$

$$\Rightarrow L\frac{di}{dt} = V - iR - K_e\frac{d\theta}{dt}$$

Response of a DC motor(contd.)



From Kirchoff's Voltage Law, $\sum V = 0$

$$\Rightarrow V = L\frac{di}{dt} + iR + e$$

Back emf(e),is proportional to the angular velocity of shaft.

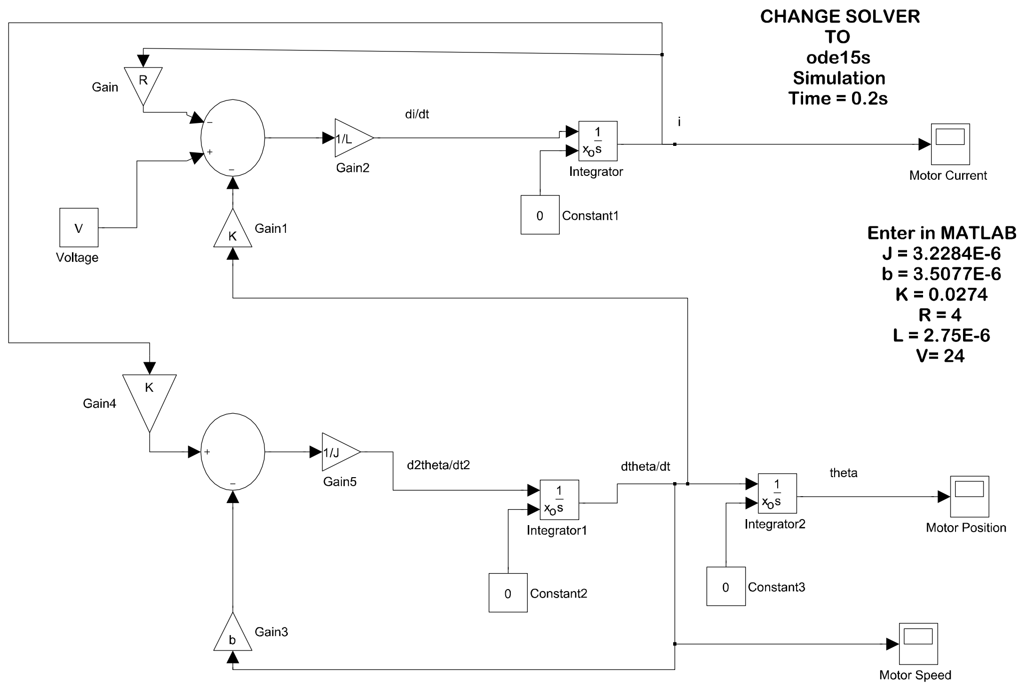
$$\therefore e = K_e\frac{d\theta}{dt}$$

$$\Rightarrow L\frac{di}{dt} = V - iR - K_e\frac{d\theta}{dt}$$

$$\therefore \frac{di}{dt} = \frac{1}{L}(V - iR - K_e\frac{d\theta}{dt}) \qquad (2)$$

Solve equation (1) and (2) simultaneously in Simulink for speed/position of motor rotor.

DC Motor Simulink Model



DC Motor Simulink Model

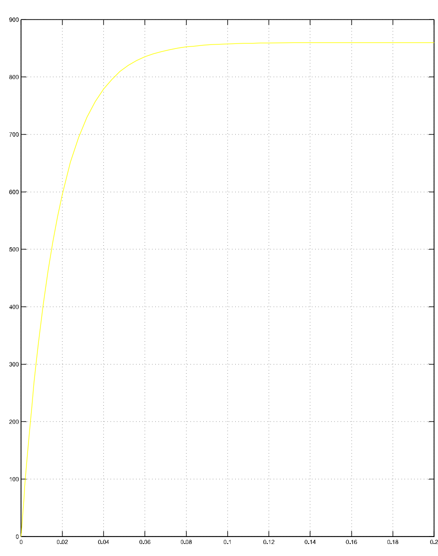


Figure: Motor Speed

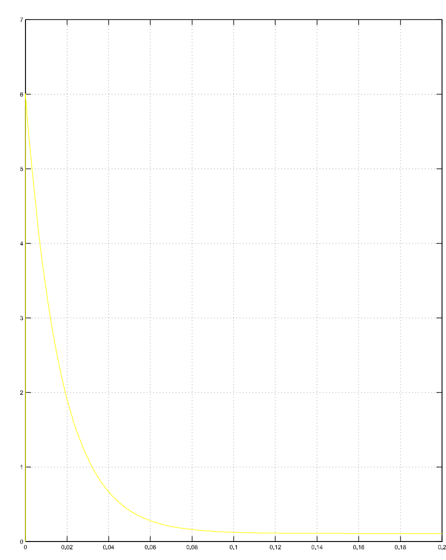


Figure: Motor Current

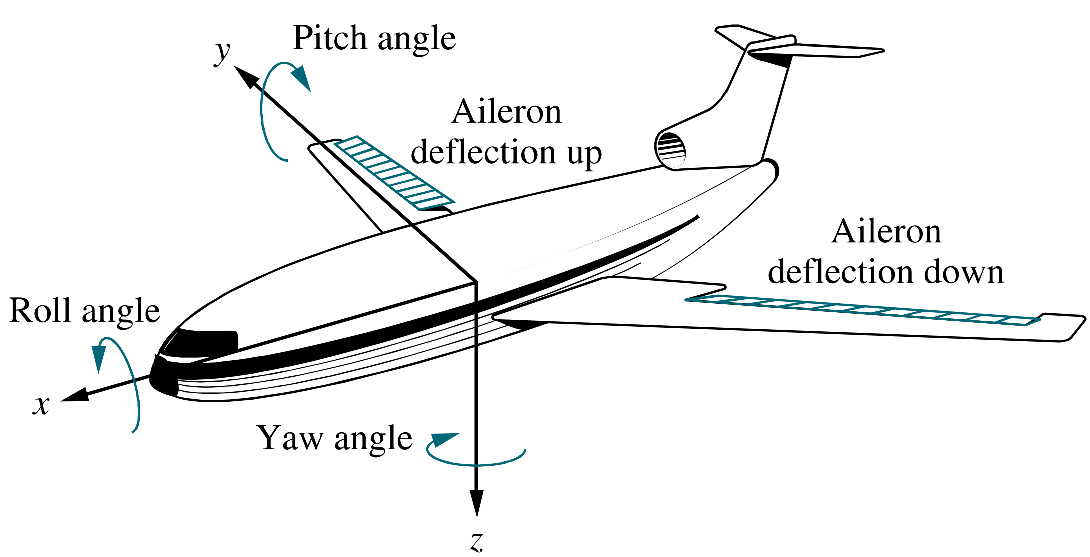
Estimating TF (Motor Speed)

Use the following data to estimate TF of motor (I=Voltage, O=Speed).

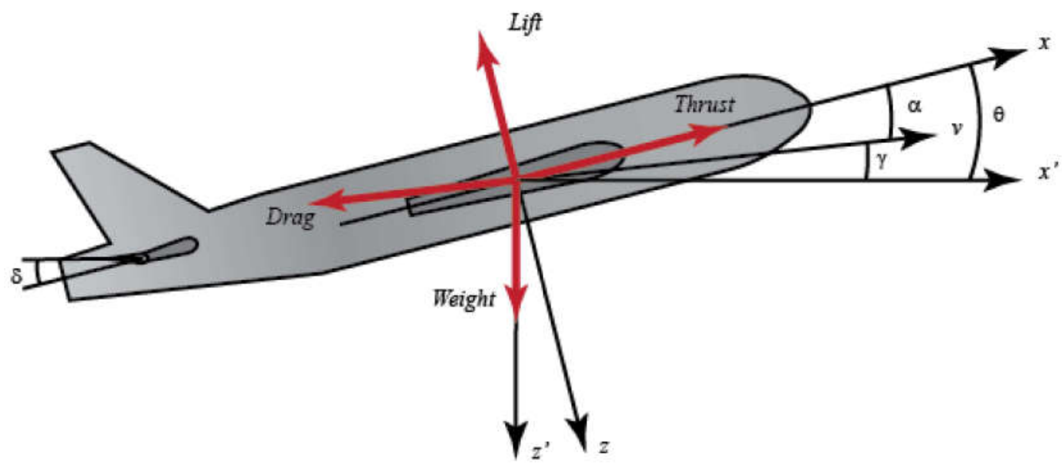
Parameter	Value
Moment of inertia of the rotor	0.01 kgm^2
Motor viscous friction constant	0.1 Nms
Electromotive force constant	0.01 $Vradsec$
Motor torque constant	0.01 $NmAmp$
Electric resistance	1 Ohm
Electric inductance	0.5 H

1. Declare Subsystem.
2. **Set Input and output linearization points. Go to Tools/Control Design/Linear Analysis Tool.**
3. Linearize Model. Export model to workspace from LTI viewer.
4. Use the command zpk(modelname).
5. $\frac{2}{(s+9.997)(s+2.003)}$ is the approximate transfer function.

Aircraft Pitch Control

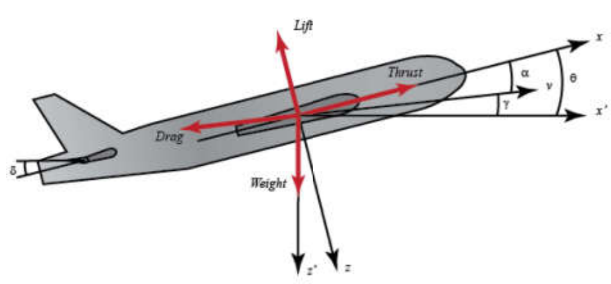


Aircraft Pitch Control



Transfer function, $G(s) = \frac{\Theta(s)}{\Delta(s)} = ??$

Aircraft Pitch Control



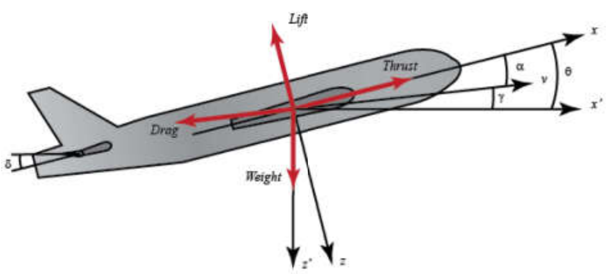
Physical Parameters

α = Angle of attack
q = Pitch Rate
θ = Pitch rate
δ = Elevator deflection

Physical Parameters

ρ = Density of air
S = Platform area of wing
$\bar{c}$ = Average chord length
m = Mass of aircraft
$\mu = \frac{\rho S \bar{c}}{4m}$
U = Equilibrium flight speed
C_T = Coefficient of thrust
C_D = Coefficient of drag
C_L = Coefficient of lift
C_W = Coefficient of weight
C_M = Coefficient of pitch moment
γ =Flight path angle
$\Omega = \frac{2U}{\bar{c}}$
$\sigma = \frac{1}{1+\mu C_L}$ = constant
i_{yy} = Normalized moment of inertia
$\eta = \mu\sigma C_M$ = constant

Aircraft Pitch Control



The equations governing the motion of an aircraft are a very complicated set of six nonlinear coupled differential equations.Therefore, in order to simplify, we will assume that the aircraft is in steady-cruise at constant altitude and velocity; thus, the thrust, drag, weight and lift forces balance each other in the x- and y-direction and that a change in pitch angle will not change the speed of the aircraft under any circumstance (unrealistic but simplifies the problem a bit!).

Under these assumptions, the following 3 governing equations are used by one of Boeing's commercial aircraft.

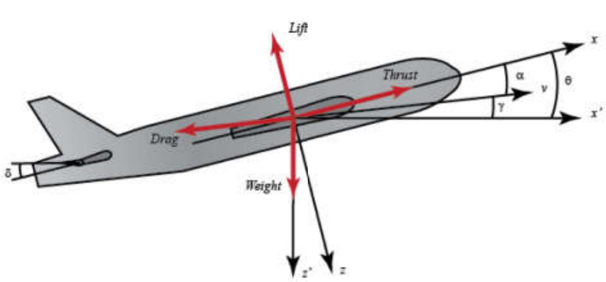
$$\frac{d\alpha}{dt} = -0.313\alpha + 56.7q + 0.232\delta$$

$$\frac{dq}{dt} = -0.0139\alpha - 4.426q + 0.0203\delta$$

$$\frac{d\theta}{dt} = 56.7q$$

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Aircraft Pitch Control



The equations governing the motion of an aircraft are a very complicated set of six nonlinear coupled differential equations.Therefore, in order to simplify, we will assume that the aircraft is in steady-cruise at constant altitude and velocity; thus, the thrust, drag, weight and lift forces balance each other in the x- and y-direction and that a change in pitch angle will not change the speed of the aircraft under any circumstance (unrealistic but simplifies the problem a bit!).

Under these assumptions, the following 3 governing equations are used by one of Boeing's commercial aircraft.

$$\frac{d\alpha}{dt} = -0.313\alpha + 56.7q + 0.232\delta$$

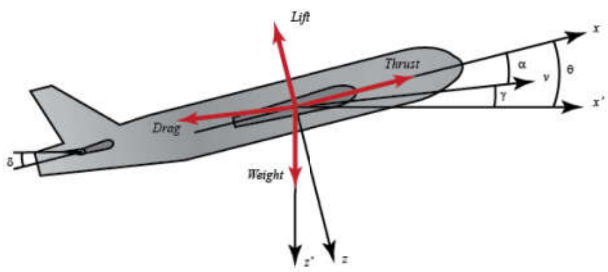
$$\frac{dq}{dt} = -0.0139\alpha - 4.426q + 0.0203\delta$$

$$\frac{d\theta}{dt} = 56.7q$$

Ans: $G(s) = \frac{1.151s+0.1774}{s^3+0.739s^2+0.921s}$

ME 3204: Control Engineeering Sessional Spring 2016 89 / 99

Aircraft Pitch Control



The equations governing the motion of an aircraft are a very complicated set of six nonlinear coupled differential equations.Therefore, in order to simplify, we will assume that the aircraft is in steady-cruise at constant altitude and velocity; thus, the thrust, drag, weight and lift forces balance each other in the x- and y-direction and that a change in pitch angle will not change the speed of the aircraft under any circumstance (unrealistic but simplifies the problem a bit!).

Under these assumptions, the following 3 governing equations are used by one of Boeing's commercial aircraft.

$$\frac{d\alpha}{dt} = -0.313\alpha + 56.7q + 0.232\delta$$

$$\frac{dq}{dt} = -0.0139\alpha - 4.426q + 0.0203\delta$$

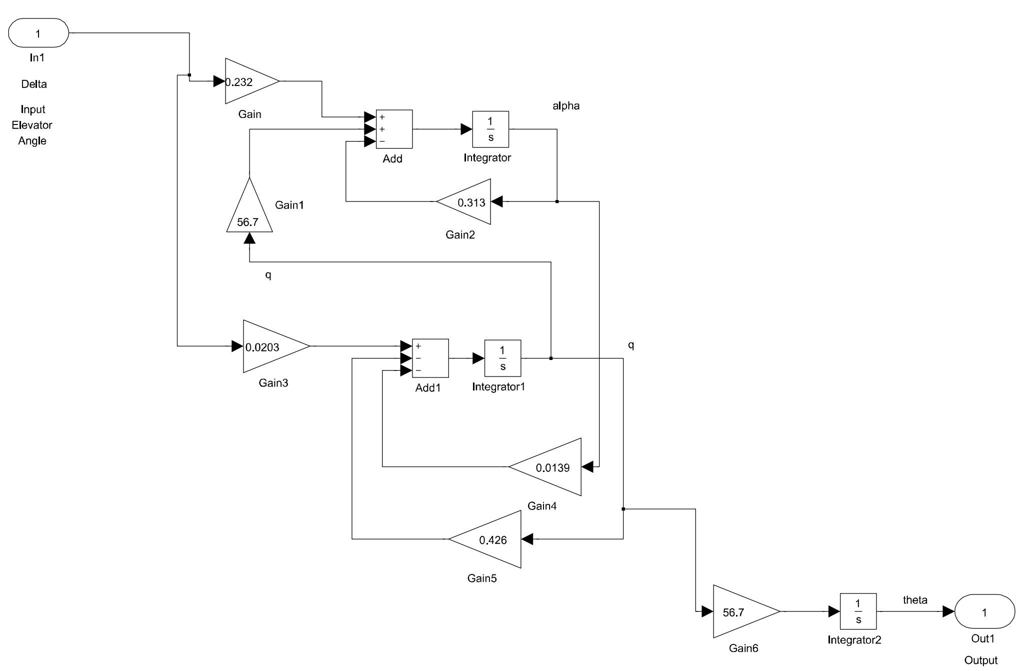
$$\frac{d\theta}{dt} = 56.7q$$

Ans: $G(s) = \frac{1.151s+0.1774}{s^3+0.739s^2+0.921s}$

Is the system stable?

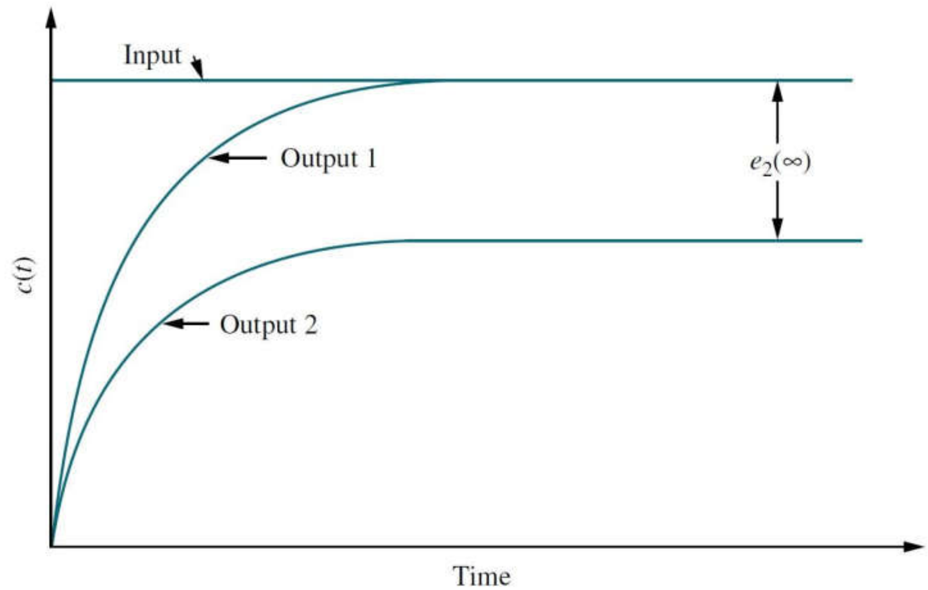
Draw step response when elevator deflection is 0.2 rad.

Aircraft Pitch Control

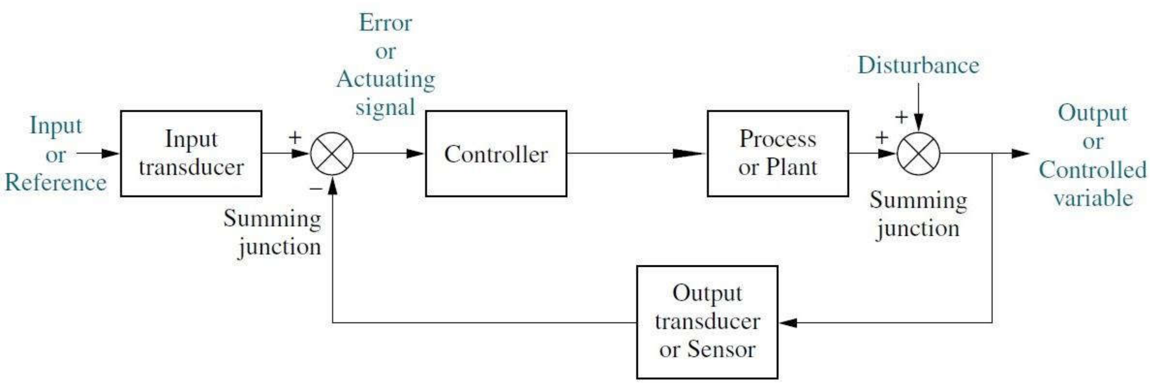


Steady State Error

The difference between the input and output for a prescribed test input as time, t approaches ∞.



Feedback Control



closed-loop system

Feedback Control

- In feedback control , the objective is to reduce error signal to zero.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Feedback Control

- In feedback control , the objective is to reduce error signal to zero.
- $e(t) = y_{sp}(t) - y_m(t)$

Feedback Control

- In feedback control , the objective is to reduce error signal to zero.
- $e(t) = y_{sp}(t) - y_m(t)$
- $e(t) =$ Error Signal

Feedback Control

- In feedback control , the objective is to reduce error signal to zero.
- $e(t) = y_{sp}(t) - y_m(t)$
- $e(t) =$ Error Signal
- $y_{sp}(t) =$ Set Point

Feedback Control

- In feedback control , the objective is to reduce error signal to zero.
- $e(t) = y_{sp}(t) - y_m(t)$
- $e(t) =$ Error Signal
- $y_{sp}(t) =$ Set Point
- $y_m(t) =$ Measured value of the controlled or process variable (equivalent sensor signal)

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Feedback Control

- In feedback control , the objective is to reduce error signal to zero.
- $e(t) = y_{sp}(t) - y_m(t)$
- $e(t) =$ Error Signal
- $y_{sp}(t) =$ Set Point
- $y_m(t) =$ Measured value of the controlled or process variable (equivalent sensor signal)
- Although the error signal equation implies that set point is time varying but in many applications it is kept constant over a long period of time.

Different types of feedback control

Proportional Control (P)

- For Proportional Control , the objective is to reduce error signal to zero.
- $p(t) = \bar{p} + K_p e(t)$
- $p(t)$ = Controller output
- $\bar{p}$ = Bias or steady state value
- K_p = Proportional Controller Gain

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

Different types of feedback control

Proportional Control (P)

- For Proportional Control , the objective is to reduce error signal to zero.
- $p(t) = \bar{p} + K_p e(t)$
- $p(t)$ = Controller output
- $\bar{p}$ = Bias or steady state value
- K_p = Proportional Controller Gain

Integral Control (I)

- For Integral Control , the rate of change of controller output is proportional to the error signal.
- $p(t) = \bar{p} + K_I \int e(t)$ i.e., $\frac{dp}{dt} = K_I e(t)$
- $p(t)$ = Controller output
- $\bar{p}$ = Bias or steady state value
- K_I = Integral Controller Gain

Different types of feedback control

Proportional Control (P)

- For Proportional Control , the objective is to reduce error signal to zero.
- $p(t) = \bar{p} + K_p e(t)$
- $p(t)$ = Controller output
- $\bar{p}$ = Bias or steady state value
- K_p = Proportional Controller Gain

Integral Control (I)

- For Integral Control , the rate of change of controller output is proportional to the error signal.
- $p(t) = \bar{p} + K_I \int e(t)$ i.e., $\frac{dp}{dt} = K_I e(t)$
- $p(t)$ = Controller output
- $\bar{p}$ = Bias or steady state value
- K_I = Integral Controller Gain

Different types of feedback control(contd.

Derivative Control (D)

- For Derivative Control , the controller output is proportional to the rate of change of error signal.
- $p(t) = \bar{p} + K_d \frac{de(t)}{dt}$
- $p(t)$ = Controller output
- $\bar{p}$ = Bias or steady state value
- K_d = Derivative Controller Gain

How PID affects a process/system

CL Response	Rise Time	Overshoot	Settling Time	S-S Error
K_p	Decrease	Increase	Small change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small change	Decrease	Decrease	No change

1. A proportional controller (K_p) reduces the rise time and will reduce but never eliminate the steady-state error.
2. An integral control (K_i) eliminates the steady-state error for a constant or step input, but it may make the transient response slower.
3. A derivative control (K_d) increases the stability of the system, reducing the overshoot, and improving the transient response.
4. Note that these correlations may not be exactly accurate, because K_p, K_i, and K_d are dependent on each other. In fact, changing one of these variables can change the effect of the other two. For this reason, the table should only be used as a reference when you are determining the values for K_i, K_p and K_d.

PID Controller Design: Step Input, Unit Feedback

```
%%Controller Design for Unit feedback
clc
clear all
disp('PID Controller Design for Unit feedback');
disp(' ');
Kp = input('Input Kp = ');
Ki = input('Input Ki = ');
Kd = input('Input Kd = ');

C=tf([Kd Kp Ki], [1 0]); % Defines the PID Controller

num = input('Input Plant TF numerator vector = ');
den = input('Input Plant TF denominator vector = ');

P = tf(num,den); % Defines the Process Transfer Function

step(P); % Draws step response of P
hold on; % Uses the same figure for next plot

T= feedback(C*P,1); % Unit feedback is fed to the controlled system

t=0:0.01:2;  %Sets simulation time

step(T,t); % Draws step response of closed loop feedback system T
```

PID Controller Tuning

Problem

Manually tune a PID controller for a plant having $G(s) = \frac{1}{s^2+10s+20}$. Your goal is to achieve -

- fast rise time
- minimum overshoot
- no steady-state error

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

PID Controller Tuning

Problem

Manually tune a PID controller for a plant having $G(s) = \frac{1}{s^2+10s+20}$. Your goal is to achieve -

- fast rise time
- minimum overshoot
- no steady-state error

Open Loop Response

- Put $K_P = 0$, $K_d = 0$, $K_I = 0$ in the code.
- The steady-state error is as much as 95%.
- Settling time is 1.5 sec.

PID Controller Tuning

Problem

Manually tune a PID controller for a plant having $G(s) = \frac{1}{s^2+10s+20}$. Your goal is to achieve -

- fast rise time
- minimum overshoot
- no steady-state error

Open Loop Response

- Put $K_P = 0$, $K_d = 0$, $K_I = 0$ in the code.
- The steady-state error is as much as 95%.
- Settling time is 1.5 sec.

P-Controller

- Put $K_P = 300$ in the code.
- Steady-state error is reduced.
- Rise time is reduced.
- Overshoot increased.
- Settling time slightly reduced.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

PID Controller Tuning

Problem

Manually tune a PID controller for a plant having $G(s) = \frac{1}{s^2+10s+20}$. Your goal is to achieve -

- fast rise time
- minimum overshoot
- no steady-state error

Open Loop Response

- Put $K_P = 0$, $K_d = 0$, $K_I = 0$ in the code.
- The steady-state error is as much as 95%.
- Settling time is 1.5 sec.

P-Controller

- Put $K_P = 300$ in the code.
- Steady-state error is reduced.
- Rise time is reduced.
- Overshoot increased.
- Settling time slightly reduced.

ME 3204: Control Engineeering Sessional Spring 2016 98 / 99

PID Controller Tuning(contd.)

PD-Controller

- Put $K_P = 300$ $K_d = 10$ in the code.
- Steady-state error is slightly reduced.
- Rise time is slightly reduced.
- Overshoot reduced.
- Settling time reduced.

Control Systems MATLAB SIMULINK Response Modeling Feedback-Control

PID Controller Tuning(contd.)

PD-Controller

- Put $K_P = 300$ $K_d = 10$ in the code.
- Steady-state error is slightly reduced.
- Rise time is slightly reduced.
- Overshoot reduced.
- Settling time reduced.

PI-Controller

- Put $K_P = 30$ $K_I = 70$ in the code.
- Proportional gain is reduced because integral controller alone reduces the rise time and increases the overshoot effect.
- If both of them have high values it will create a double effect.
- This controller eliminates the steady-state error.

PID Controller Tuning(contd.)

PD-Controller

- Put $K_P = 300$ $K_d = 10$ in the code.
- Steady-state error is slightly reduced.
- Rise time is slightly reduced.
- Overshoot reduced.
- Settling time reduced.

PI-Controller

- Put $K_P = 30$ $K_I = 70$ in the code.
- Proportional gain is reduced because integral controller alone reduces the rise time and increases the overshoot effect.
- If both of them have high values it will create a double effect.
- This controller eliminates the steady-state error.

PID-Controller

- Put $K_P = 350$ $K_I = 300$ $K_D = 5500$ in the code.
- Very fast rise time.
- No overshoot.
- No steady-state error