Week 7 – Modeling and Control of DC Motor

Advanced Mechatronics System Design – MANU2451

Dr Chow Yin LAI
Edited by Dr Milan Simic
School of Engineering
RMIT University, Victoria, Australia
Email: milan.simic@rmit.edu.au

RMIT UNIVERSITY

New Teaching Schedule

Week		Class Activity Before	Lecture	Class Activity During or After
1			Introduction to the Course / Introduction to LabVIEW	LabVIEW Programming
2			Introduction to LabVIEW / Data Acquisition	LabVIEW Programming
3			Gripper / Introduction to Solidworks / Safety	Gripper Design
4			Sensors I	myRIO Programming for Sensor Signal Reading / Gripper Design
5			Sensors II	myRIO Programming for Sensor Signal Reading
6			Actuators I	LabVIEW Tutorial
7		LabVIEW Assessment.	DC Motors I	Matlab Simulink Simulation
8		Design report submission	DC Motors II	Matlab Simulink Simulation / myRIO Programming for Control
9			Actuators II	Matlab Simulink Simulation Gripper CAD
10			Modeling and System Identification	Matlab Simulink Simulation / Gripper simulation testing
11			Artificial Intelligence I	Matlab Simulation / Finalize Gripper
12		Gripper Simulation / Submission of Report	Artificial Intelligent II	Revision

LabVIEW Test

- The Quiz is an assessment item graded with 20 points.
- It is available from 22nd of April 19:30 for following 24 hours.
- Once started, it will run for 60 min.
- You can attempt it just once.
- Correct answers will be shown after 25th.

Gripper Project

- Gripper only design, No arm,
- Now just sketch, simple drawings
- No need for the LabVIEW code, just algorithms

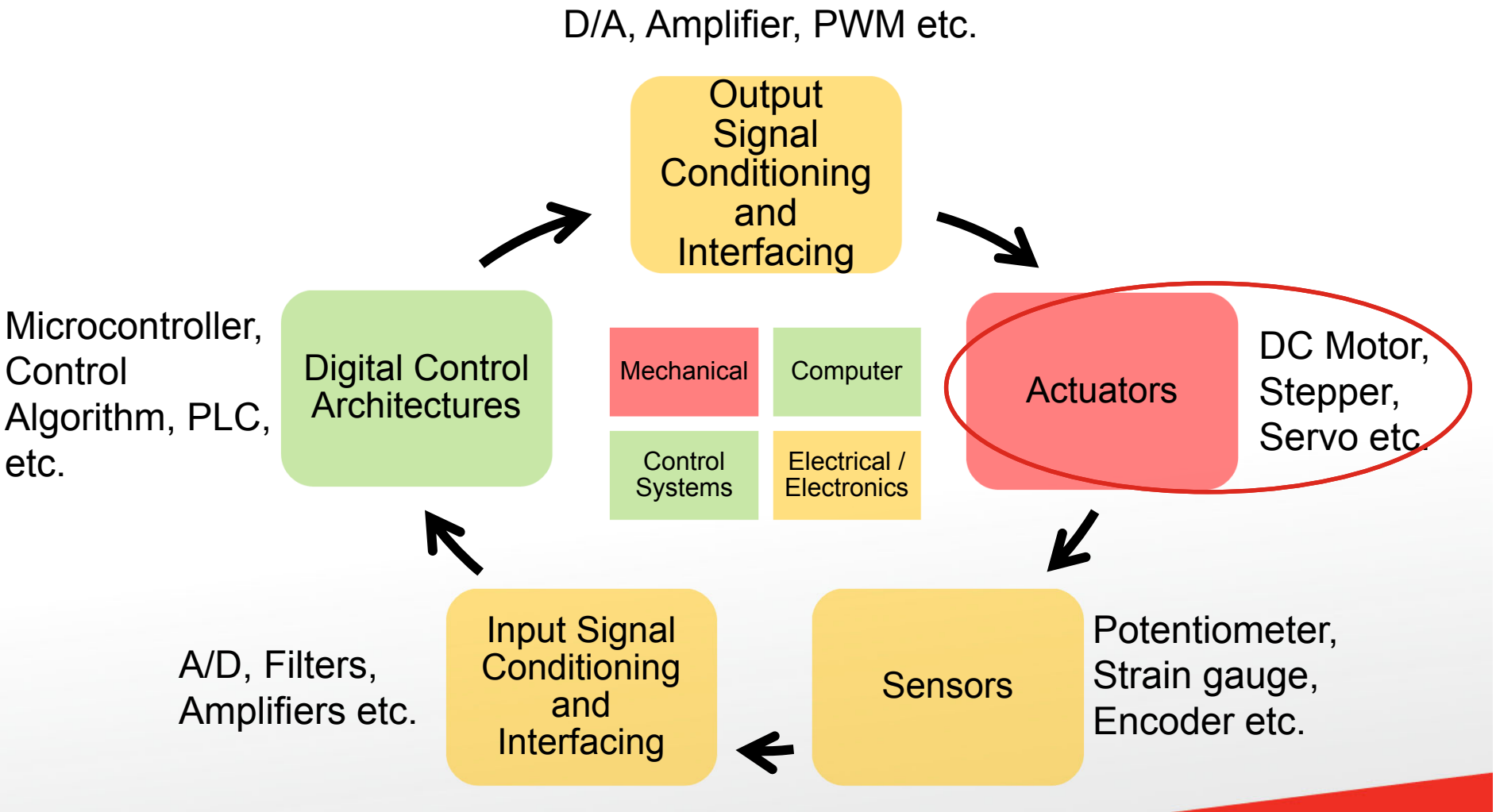
There are basically two separated tasks.

1. One is Gripper model design in any CAD software.
2. The other is LabVIEW Control program, i.e. simulation, could be called Gripper_Control.vi. Example is my Dual Temperature Control.vi
3. No need for the simulation that will link Gripper_Control.vi with CAD model. That would be too much.

- 4. If you can do that, it will be bonus.

RMIT UNIVERSITY

Mechatronics System Components



Mechatronics System Components

Sensors: Encoder at each joint

→ Input signal interfacing

↓

Robot controller:

- Generate desired motion trajectory
- Calculate current end-effector position based on angular position (kinematics)
- Calculate desired angular position for desired end-effector position and trajectory (inverse kinematics)
- Control algorithm
- Safety, collision detection etc.

↓

← Output signal interfacing

Actuators: Geared motor at each joint

Industrial Robots

https://commons.wikimedia.org/wiki/File:Float_Glass_Unloading.jpg

RMIT UNIVERSITY

Content

- Modeling of DC Motors
    - MATLAB / Simulink Simulation
- Speed Control of DC Motors
    - MATLAB / Simulink Simulation
- Position Control of DC Motors
    - MATLAB / Simulink Simulation
- Force Control of DC Motors
    - MATLAB / Simulink Simulation

Content

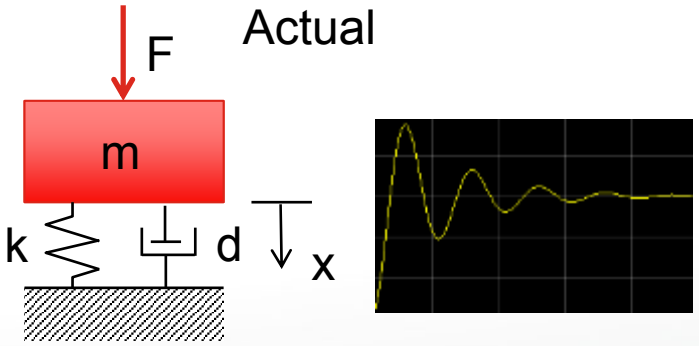
- Modeling of DC Motors
  - MATLAB / Simulink Simulation
- Speed Control of DC Motors
  - MATLAB / Simulink Simulation
- Position Control of DC Motors
  - MATLAB / Simulink Simulation
- Force Control of DC Motors
  - MATLAB / Simulink Simulation

Modeling

- Modeling in our context: Mathematical representation of the dynamical response / output of a (mechanical, electrical etc.) system for a certain input.
  - For e.g. if we change the voltage to a PM DC motor from 0V to 5V, how does the speed change?
  - Of course, if we have the actual setup, we can just run an experiment and measure the response.
  - However, this is sometimes not possible:
    - System too expensive or complicated to set up.
    - Experiment is dangerous (e.g. can lead to explosion or fire).
    - Experiment is not possible to be done on Earth (e.g. NASA scientists design a robot to walk on Pluto).
  - Therefore, if we develop a good model / representation of the real system, we can simulate and test the response before deploying the actual system.

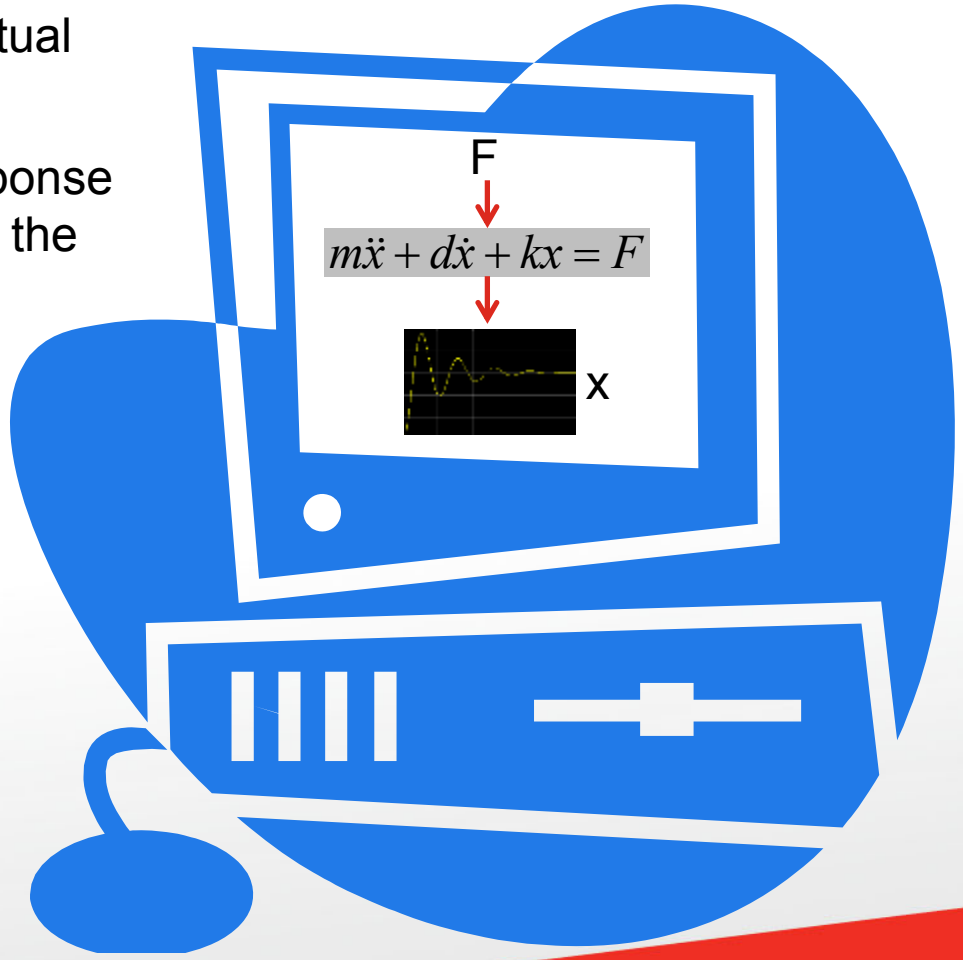
Modeling

- But what is a “good” model?
- It is a close representation of the actual system.
  - Calculated / Simulated output response to the given input is very similar to the real system



Actual

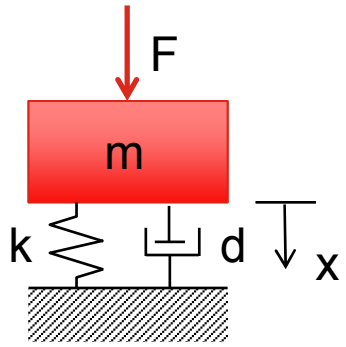
Model



- We can thus trust the calculation / simulation in our analysis or design.

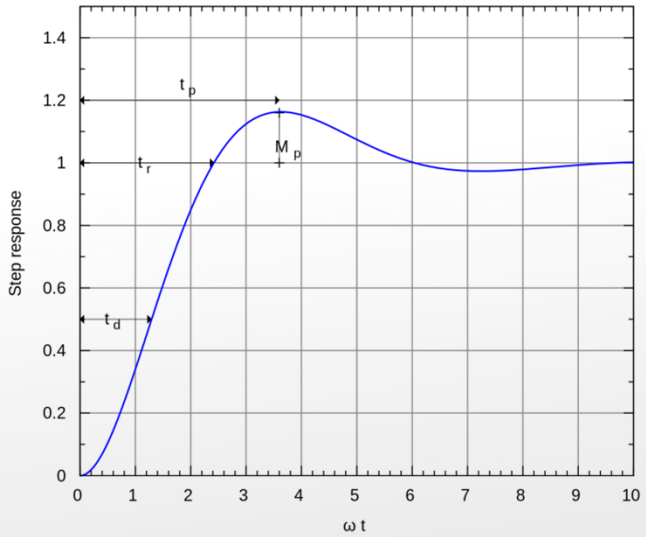
RMIT UNIVERSITY

Why modeling (1)



- To <u>understand</u> the characteristic of a system.
  - E.g. <u>given</u> a mass-spring-damper system (e.g. suspension of a car, or a damping table).
  - What is the response (x) when a constant F is applied to the system?
  - By having the mathematical model $m\ddot{x} + d\dot{x} + kx = F$, the response of the system can be calculated or simulated.

This can be obtained from physics

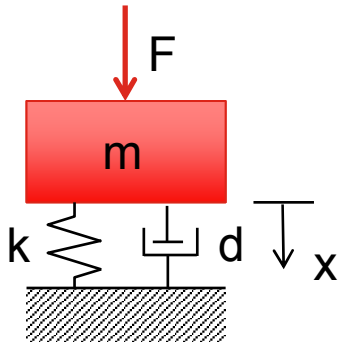


Response of 2nd Order System

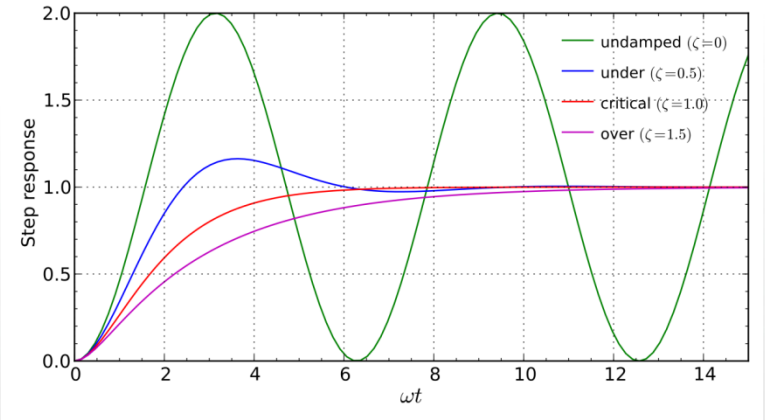
https://commons.wikimedia.org/wiki/File:Second_order_under-damped_response.svg

RMIT UNIVERSITY

Why modeling (2)



- To design a system for desired response.
  - E.g. You are asked to design a mass-spring-damper system (e.g. suspension of a car, or a damping table).
  - How to choose good values for k and d so that the response (x) is satisfactory, when a constant F is applied to the system?
  - By having the mathematical model $m\ddot{x} + d\dot{x} + kx = F$, the response of the system can be calculated or simulated, and thus we can choose proper values for k and d.

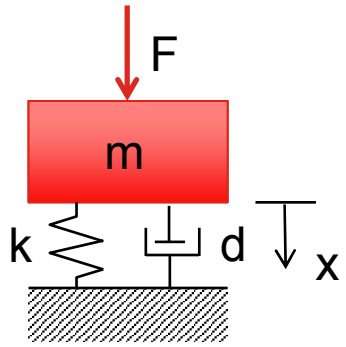


Different responses of 2nd Order System

https://commons.wikimedia.org/wiki/File:Second_order_transfer_function.svg

RMIT UNIVERSITY

Why modeling (3)



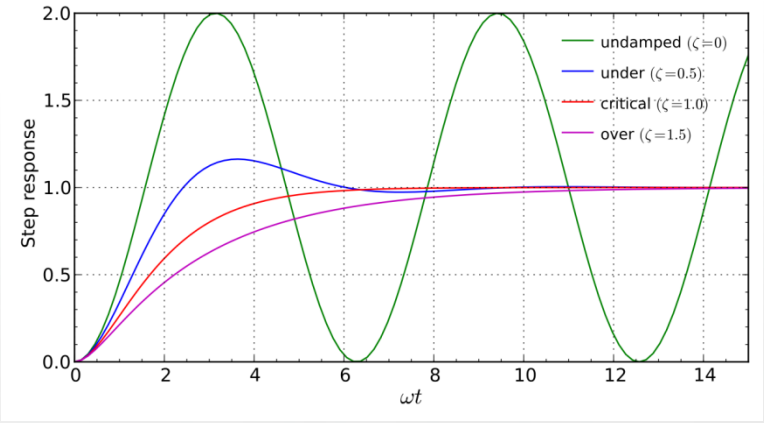
- To design a controller to obtain desired response.
  - E.g. given a mass-spring-damper system (e.g. suspension of a car, or a damping table). You can't change the k and d anymore.
  - However, imagine now you can alter the F, and you design it as:

$$F = -k_1 x - k_2 \dot{x} + F_r$$

  - Then the overall system including the controller becomes:

$$m\ddot{x} + (d + k_2)\dot{x} + (k + k_1)x = F_r$$

  - Thus you can design proper k_1 and k_2 to achieve desired performance.



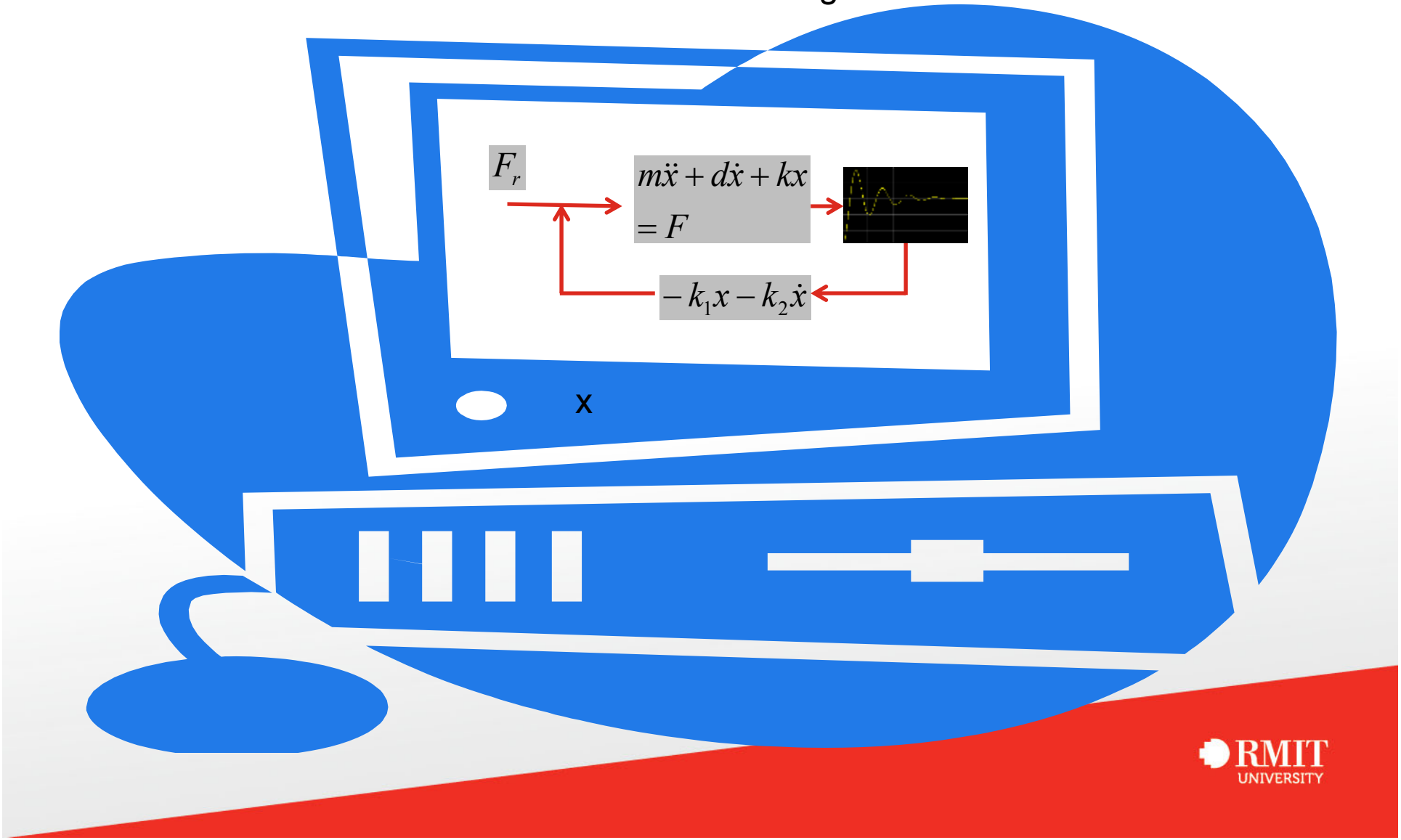
Different responses of 2nd Order System

https://commons.wikimedia.org/wiki/File:Second_order_transfer_function.svg

RMIT UNIVERSITY

Why modeling (3)

- Illustration of the use of model in control design:

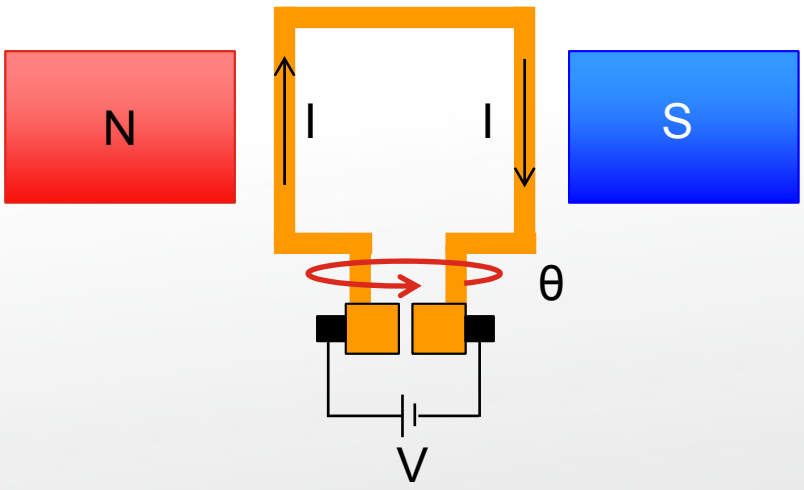


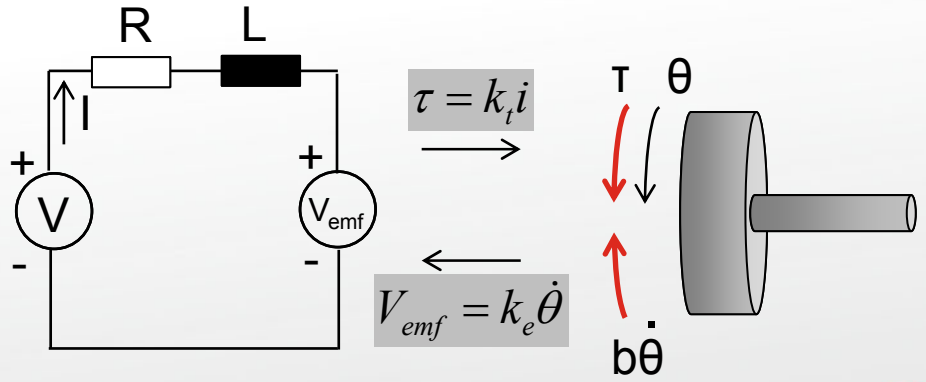
PM DC Motor Dynamic Equations

- Now, we will derive a model of PM DC motor, with voltage as the input, and either position or speed as the output.
  - To understand the response of the motor.
    - If we change the input voltage, how does the position or speed change?
  - For control design and simulation.
    - If we want the speed to maintain at e.g. 100rpm, how should the voltage be adjusted?

PM DC Motor Dynamic Equations

- From the illustration of a DC motor (left), we see that there are electrical quantities (V, I etc.) as well as a mechanical quantities (θ, w).
- The electrical part creates a torque for the mechanical part, but the mechanical part also induces a back *Electromotive Force (EMF)* to the electrical part.
- The graphical model is shown in the middle and on the right:





PM DC Motor Dynamic Equations

- Due to the interaction between stator field and armature current, the torque generated by a PM DC motor is proportional to the armature current:

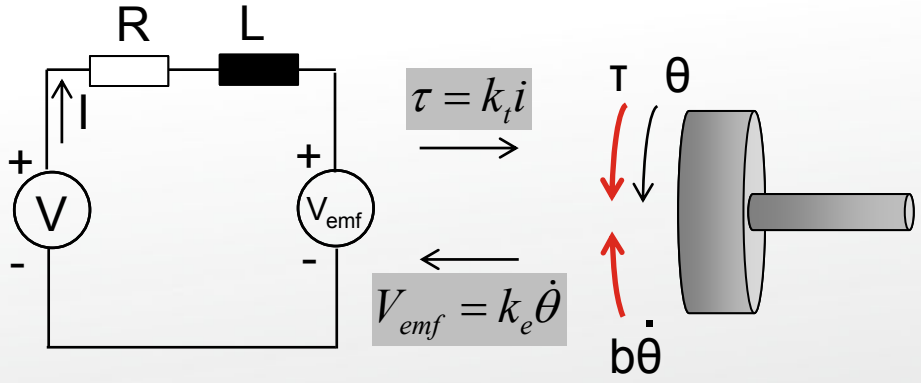
$$\tau = k_t i$$

  - Where k_t is the torque constant of the motor. Usually given in data sheet.
- We also knew that back EMF is proportional to rotor speed:

$$V_{emf} = k_e \omega$$

  - Where k_e is the electrical constant of the motor.
- The electric circuit equation is:

$$\begin{aligned} L\frac{di}{dt} + Ri &= V - V_{emf} \\ &= V - k_e \omega \\ &= V - k_e \dot{\theta} \end{aligned}$$



- The mechanical equation is: $J\ddot{\theta} + b\dot{\theta} = \tau = k_t i$
  - Where *b* is viscous friction.

RMIT UNIVERSITY

Dynamic Response of PM DC Motor

- The electrical and mechanical equations of the PM DC motor can give us the dynamic response (θ) when a voltage of V is supplied.

$$L\frac{di}{dt} + Ri = V - k_e\dot{\theta} \qquad J\ddot{\theta} + b\dot{\theta} = \tau = k_t i$$

- But it is not straightforward:
  - V affects I, I affects Tau, Tau affects Theta, and Theta_dot affects V again.

Dynamic Response of PM DC Motor

- The response (θ) to a voltage of V can be calculated easier using the Laplace Transformation.

f(t)	F(s)	f(t)	F(s)
$\delta(t)$	1	$\frac{1}{a}\left(at-1+e^{-at}\right)$	$\frac{a}{s^2(s+a)}$
1	$1/s$	$e^{-at}-e^{-bt}$	$\frac{b-a}{(s+a)(s+b)}$
t	$1/s^2$	$(1-at)e^{-at}$	$\frac{s}{(s+a)^2}$
t^2	$2!/s^3$	$1-e^{-at}(1+at)$	$\frac{a^2}{s(s+a)^2}$
t^3	$3!/s^4$	$be^{-bt}-ae^{-at}$	$\frac{(b-a)s}{(s+a)(s+b)}$
e^{-at}	$\frac{1}{s+a}$	$\sin(at)$	$\frac{a}{s^2+a^2}$
te^{-at}	$\frac{1}{(s+a)^2}$	$\cos(at)$	$\frac{s}{s^2+a^2}$
$\frac{1}{2!}t^2e^{-at}$	$\frac{1}{(s+a)^3}$	$e^{-at}\cos(bt)$	$\frac{s+a}{(s+a)^2+b^2}$
$\frac{1}{(m-1)!}t^{m-1}e^{-at}$	$\frac{1}{(s+a)^m}$	$e^{-at}\sin(bt)$	$\frac{b}{(s+a)^2+b^2}$
$1-e^{-at}$	$\frac{a}{s(s+a)}$	$1-e^{at}\left(\cos(bt)+\frac{a}{b}\sin(bt)\right)$	$\frac{a^2+b^2}{s\left((s+a)^2+b^2\right)}$

Differentiation

$$L\{\dot{f}\}=sF(s)-f(0)$$
$$L\{\ddot{f}\}=s^2F(s)-sf(0)-\dot{f}(0)$$
$$L\{f^{(m)}\}=s^mF(s)-s^{m-1}f(0)-s^{m-2}\dot{f}(0)-\cdots-f^{(m-1)}(0)$$

For control design, we usually ignore the initial conditions, thus:

$$L\{\dot{f}\}=sF$$
$$L\{\ddot{f}\}=s^2F$$

RMIT UNIVERSITY

Dynamic Response of PM DC Motor

- Using Laplace transform, and letting all initial conditions be zero, we have:

$$L\frac{di}{dt} + Ri = V - k_e\dot{\theta} \qquad J\ddot{\theta} + b\dot{\theta} = \tau = k_t i$$

$$\downarrow \qquad \downarrow$$

$$(Ls + R)I = V - k_e s\theta \qquad (Js^2 + bs)\theta = k_t I$$

- Eliminate I:

$$I = \frac{V}{Ls+R} - \frac{k_e s\theta}{Ls+R} \Rightarrow (Js^2 + bs)\theta = \frac{k_t V}{Ls+R} - \frac{k_t k_e s\theta}{Ls+R}$$

- After arranging, we would get:

$$\frac{\theta}{V} = \frac{k_t}{s(Js+b)(Ls+R) + k_t k_e s}$$

(♪ ♪)

"Transfer function"
Input to output relationship when initial conditions are zero.

RMIT UNIVERSITY

Dynamic Response of PM DC Motor

- The last equation can be written as:

$$\theta = \frac{k_t}{JLs^2 + (JR + bL)s + (bR + k_t k_e)} \cdot \frac{1}{s} \cdot V$$

- If V is a step input (i.e. 0V before t=0 and 1V after t=0), then the Laplace transform of V is $\frac{1}{s}$ (see Laplace table).

- Therefore $\theta(s)$ becomes:

$$\theta = \frac{k_t}{JLs^2 + (JR + bL)s + (bR + k_t k_e)} \cdot \frac{1}{s^2}$$

- Assume J = 0.01 kgm^2, b = 0.001Nms, k_t = k_e = 1, R = 10 Ohm, L = 1 H. Then:

$$\theta = \frac{100}{s^2 + 10.1s + 101} \cdot \frac{1}{s^2}$$

- Finally, we just need to do "inverse Laplace" for the expression above to get the response of $\theta(t)$.

RMIT UNIVERSITY

Dynamic Response of PM DC Motor

- By doing partial fraction, the previous equation can be written as:

$$\theta = \frac{100}{s^2+10.1s+101}\cdot\frac{1}{s^2} = \frac{0.09901s+0.009901}{s^2+10.1s+101} - \frac{0.09901}{s} + \frac{0.9901}{s^2}$$

- We modify it into a form where we can get inverse Laplace from Laplace transform tables:

$$\begin{aligned}
\theta &= \frac{0.09901s+0.009901}{(s+5.05)^2+75.4975} - \frac{0.09901}{s} + \frac{0.9901}{s^2} \\
&= \frac{0.09901(s+5.05)-0.4901}{(s+5.05)^2+8.6889^2} - \frac{0.09901}{s} + \frac{0.9901}{s^2} \\
&= 0.09901\frac{(s+5.05)}{(s+5.05)^2+8.6889^2} - 0.0564\frac{8.6889}{(s+5.05)^2+8.6889^2} - \frac{0.09901}{s} + \frac{0.9901}{s^2} \\
&= 0.09901\frac{(s+a)}{(s+a)^2+b^2} - 0.0564\frac{b}{(s+a)^2+b^2} - \frac{0.09901}{s} + \frac{0.9901}{s^2}
\end{aligned}$$

RMIT UNIVERSITY

Dynamic Response of PM DC Motor

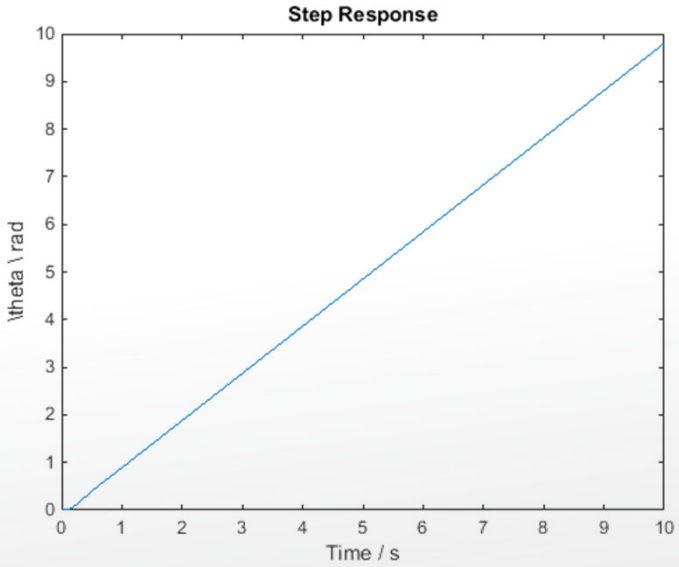
- By referring to Laplace transform table, the response $\theta(t)$ can be written as:

$$\theta = 0.09901e^{-5.05t}\cos(8.6889t) - 0.0564e^{-5.05t}\sin(8.6889t) - 0.09901 + 0.9901t$$

- The last equation shows that after a while, the last term dominates all the other terms, and θ grows linearly with t.

Matlab code:

```
t=0:0.01:10;
x=0.09901*exp(-5.05*t).*cos(8.6889*t)...
    -0.0564*exp(-5.05*t).*sin(8.6889*t)...
    -0.09901+0.9901*t;
plot(t,x)
```



- This is intuitive. If we give a constant voltage supply, the motor will rotate continuously and the position will increase.

RMIT UNIVERSITY

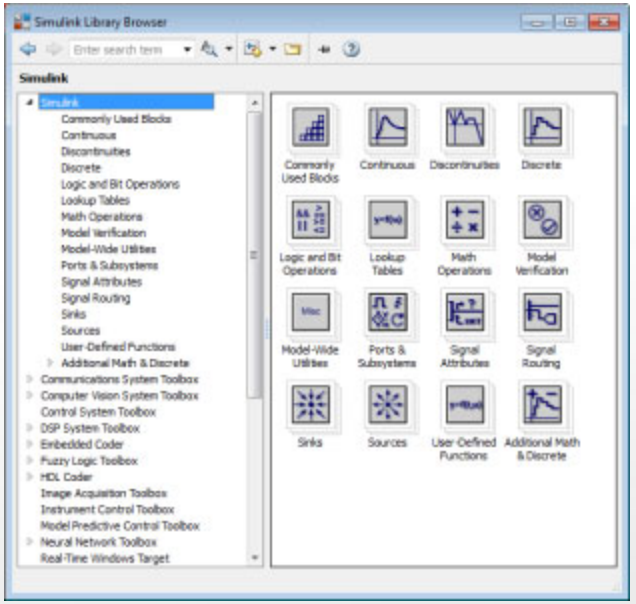
Content

- Modeling of DC Motors
  - MATLAB / Simulink Simulation
- Speed Control of DC Motors
  - MATLAB / Simulink Simulation
- Position Control of DC Motors
  - MATLAB / Simulink Simulation
- Force Control of DC Motors
  - MATLAB / Simulink Simulation

Simulation of PM DC Motor

- Performing our own inverse Laplace is a bit time-consuming. Luckily, we can use Matlab Simulink to simulate the response.

1) Start MATLAB
2) On the top panel, click “Simulink Library”.
3) The Simulink library will appear.

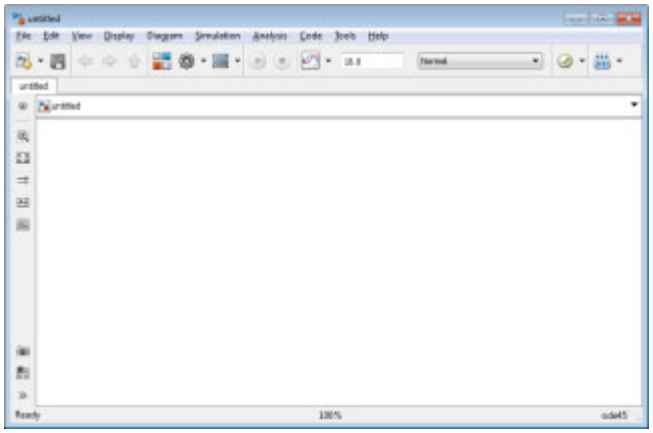


RMIT UNIVERSITY

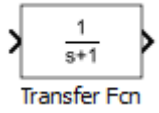
Simulation of PM DC Motor

4) Click on the “New Model” button 

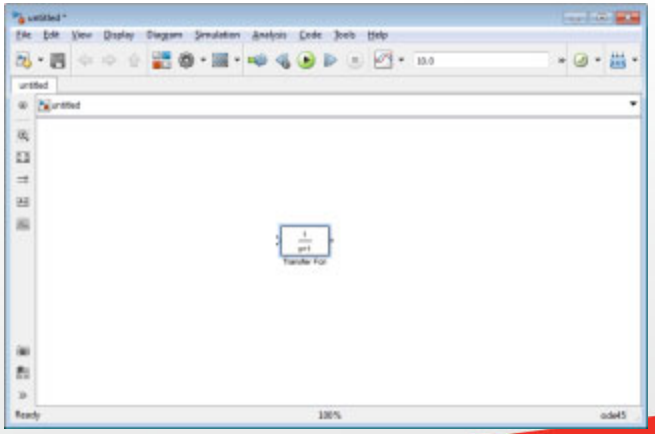
5) An empty window will open.



6) From the Simulink Library, look for “Transfer Fcn” block under “Continuous”.



7) Drag it and drop it into the model window.



RMIT UNIVERSITY

Simulation of PM DC Motor

8) Our transfer function was:

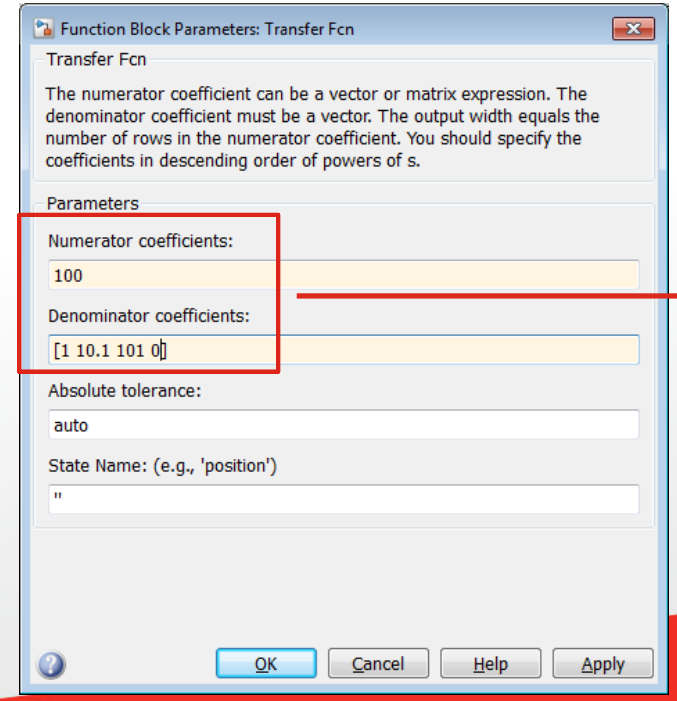
$$\frac{\theta}{V} = \frac{k_t}{s(Js+b)(Ls+R)+k_t k_e s}$$

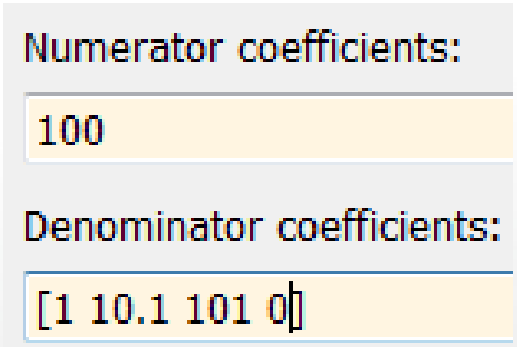
$$\frac{\theta}{V} = \frac{100}{s^2+10.1s+101} \cdot \frac{1}{s} = \frac{100}{s^3+10.1s^2+101s}$$

9) Double click on the transfer function block.

10) The function block parameters window will open.

11) Key in the coefficients as shown.





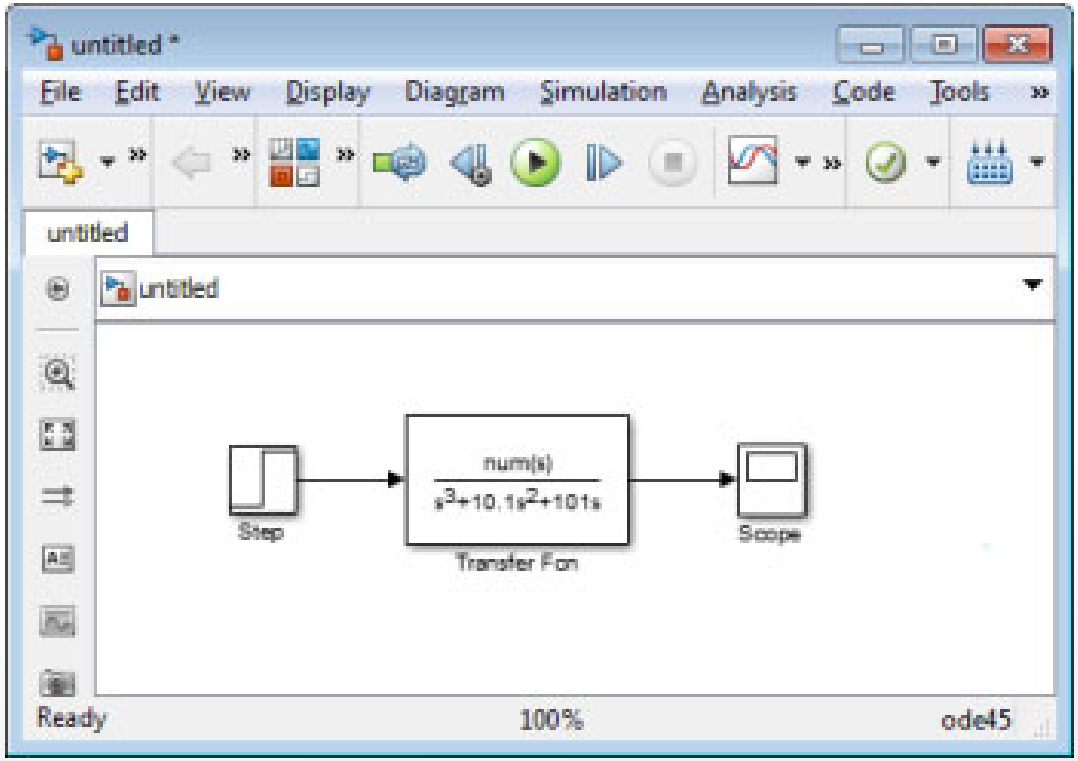
RMIT UNIVERSITY

Simulation of PM DC Motor

12) Drag and drop the “Step” block from “Sources”.

13) Drag and drop the “Scope” block from “Sinks”.

14) Build the model as shown:



15) Double click on the scope → look for “Parameters” button → go to “History” tab → Uncheck “Limit data points to Last 5000” → OK.

RMIT UNIVERSITY

Simulation of PM DC Motor

16) Back at the Model window, look for “Simulation” on the top panel.

17) Open “Model Configuration Parameters”.

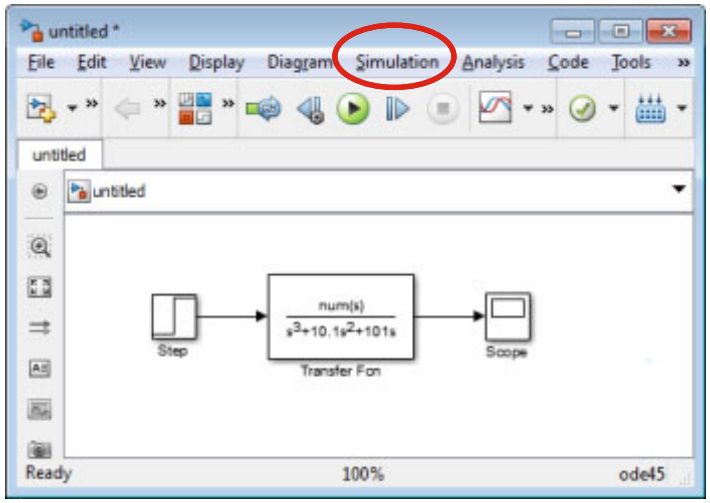
18) Under the solver options, set

    Type: fixed step.

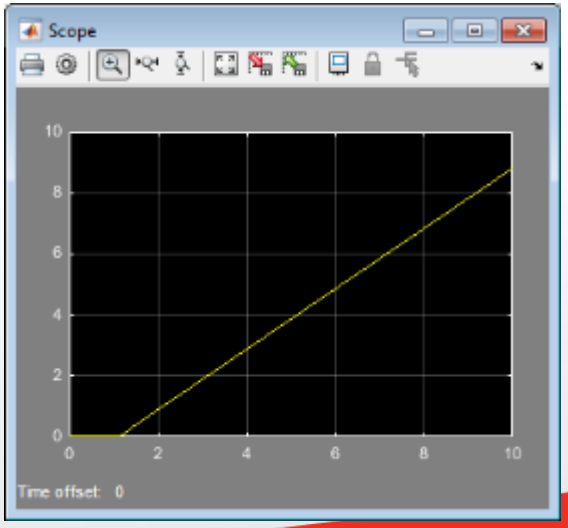
    Solver: ode4 (Runge-Kutta).

    Fixed step size: 1e-3

19) Click the run button on the model window.



20) Double click the scope and you will see the simulation results.



This is the angular position response of motor when given a constant voltage!

Note: The “Step” input starts at 1sec, thus the response starts only at 1sec.

RMIT UNIVERSITY

Speed Response

- The responses shown previously were for position / rotational angle.
- What if we are more interested at the speed of DC motor?
- Since $\omega = \dot{\theta}$
- The electrical and mechanical equations can be written as:

$$L\frac{di}{dt} + Ri = V - k_e\omega \qquad J\dot{\omega} + b\omega = \tau = k_t i$$

- Again, by using Laplace transform, we get:

$$(Ls + R)I = V - k_e\Omega \qquad (Js + b)\Omega = k_t I$$

- By eliminating I and rearranging, we will obtain:

$$\frac{\Omega}{V} = \frac{k_t}{(Js + b)(Ls + R) + k_t k_e}$$

- Let's try out Simulink together to get the speed response.

RMIT UNIVERSITY

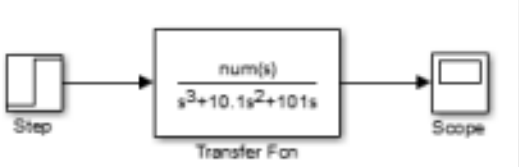
Summary

- In this section, we have derived the model of a PM DC Motor.
  - Electrical Circuit and Mechanical Equations.

$$L\frac{di}{dt} + Ri = V - k_e\omega \qquad J\dot{\omega} + b\omega = \tau = k_t i$$

  - Laplace Transform of the above, giving the transfer functions:
    - Voltage to Position response: $\frac{\theta}{V} = \frac{k_t}{s(Js+b)(Ls+R)+k_t k_e s}$
    - Voltage to Speed response: $\frac{\Omega}{V} = \frac{k_t}{(Js+b)(Ls+R)+k_t k_e}$

- We also learnt how to use simulink to simulate the response.



RMIT UNIVERSITY

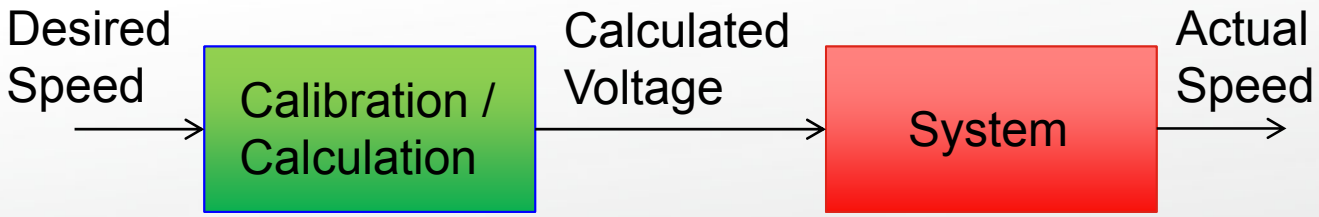
Content

- Modeling of DC Motors
  - MATLAB / Simulink Simulation
- Speed Control of DC Motors
  - MATLAB / Simulink Simulation
- Position Control of DC Motors
  - MATLAB / Simulink Simulation
- Force Control of DC Motors
  - MATLAB / Simulink Simulation

RMIT UNIVERSITY

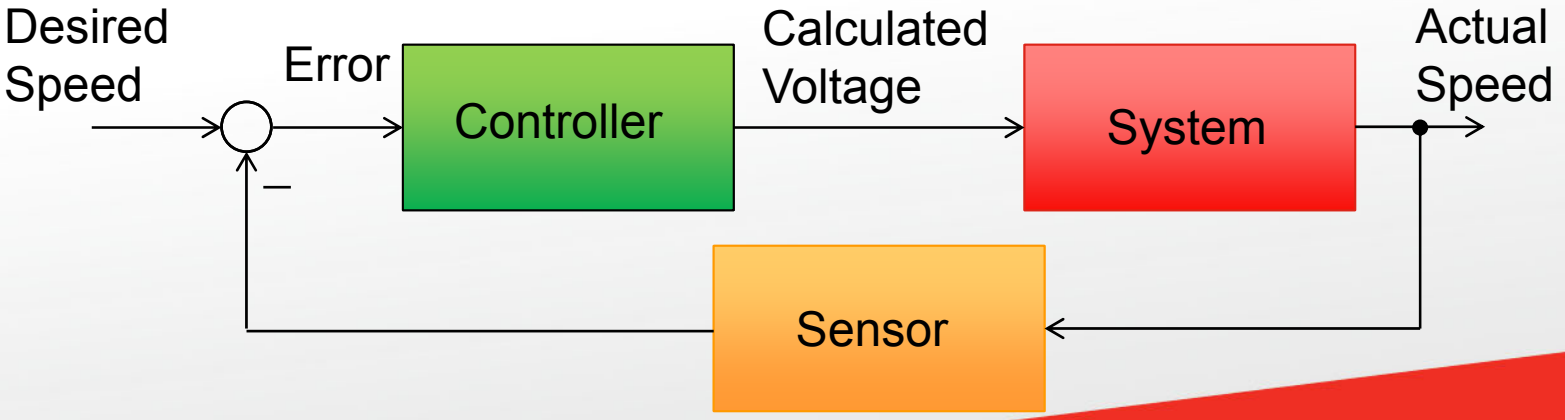
Feedforward Control

- From the step response for speed, we saw that if power supply = 1V, the speed will reach approximately 0.9901 rad/sec.
- Suppose we now want 2 rad/sec, we will do some calculations and decide to increase the power supply to 2/0.9901 = 2.02V.
- This is a "Feedforward" control! It may work in ideal circumstances but…
  - If there is inaccuracy / nonlinearity in the system, or if there is a disturbance torque acting on the motor, the motor will not reach 2 rad/sec.
  - Without speed sensor, we would not know!



Why Feedback Control?

- It is better to use “Feedback” control:
  - The actual speed is measured.
  - If the speed is less than (more than) the desired speed, the power supply will increase (decrease) to achieve the desired speed.
  - This is a more robust method!
    - Even if model is inaccurate, and
    - Even if there is external disturbances,
    - Sensor will tell us that the speed is incorrect and power supply will be adjusted.



RMIT UNIVERSITY

PID Controller (1)

- We will use the PID controller as follows:

$$e(t) = \omega_r(t) - \omega(t)$$

ω : actual speed

ω_r : desired speed

$$V = k_p e + k_I \int_{t_0}^{t} e(t) d\tau + k_D \dot{e}$$

- The first portion $k_p e$ is proportional to the system error.
  - E.g. desired speed is 100 whereas actual speed is 10. The error is 90. If k_p is 0.1, then $k_p e = 9\text{V}$. This huge voltage will push the motor to turn faster.
  - When the actual speed is 80, the error is 20. Then $k_p e$ = 2V. This voltage will continue to make the motor turn faster.
  - However, if the actual speed is 130, the error is -30. $k_p e$ becomes -3V and thus reduces the speed of the motor, towards the desired value (100).
  - Note: The P-part does not guarantee error to be zero!

PID Controller (2)

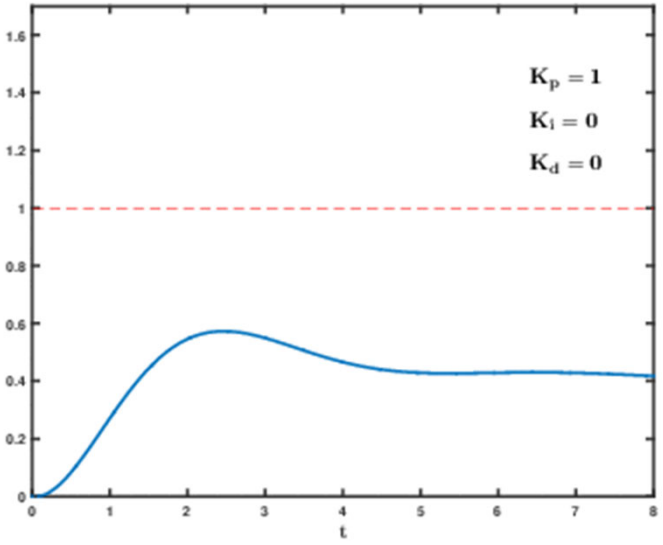
- The second portion is related to the integral of error. $k_I \int_{t_0}^{t} e(t) d\tau$
  - This integral part is important to achieve zero steady state error.
    - Because it will not stop changing until its input (i.e. Error) is zero.
  - Useful when there is a constant disturbance acting on the system.

PID Controller (3)

- The third portion is the differential control. $k_D\dot{e}$
  - The differential control provides "anticipatory" action.
  - E.g. The speed happens to increase from 90 to 100, but it still has some "momentum" to overshoot and become greater than 100.
  - The differential part will be non-zero & negative, and this helps to reduce the speed overshoot.

PID Controller Tuning

- It is possible to calculate the parameters of the PID controller mathematically, by specifying the desired performance and do some coefficient matching (pole placement method). However, we will not cover this in the lecture.
- Instead, we will just use manual tuning methods:



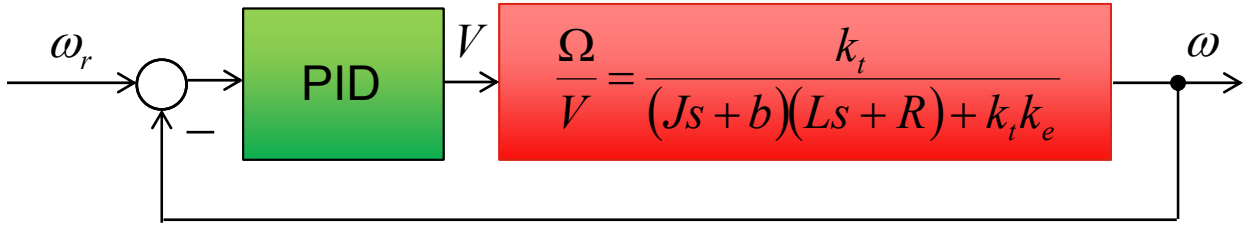
PID Tuning

https://commons.wikimedia.org/wiki/File:PID_Compensation_Animated.gif

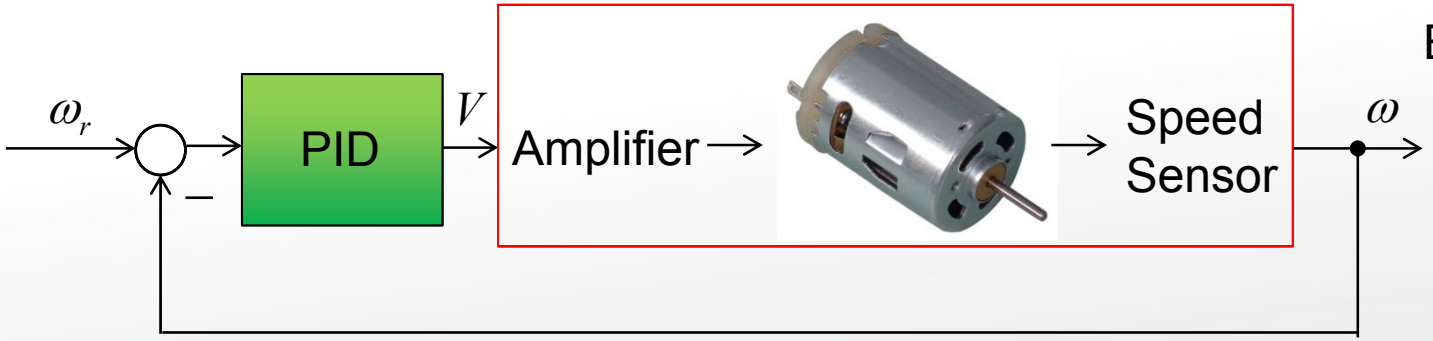
In-crease	Rise Time	Settling Time	Over-shoot	Steady State Error	Stability
K_P	Less		More	Less	Worse
K_I	Less	More	More	Zero	Worse
K_D		Less	Less		Improve

Speed Control (1)

- We will now put in the PID controller in the system to control the speed.



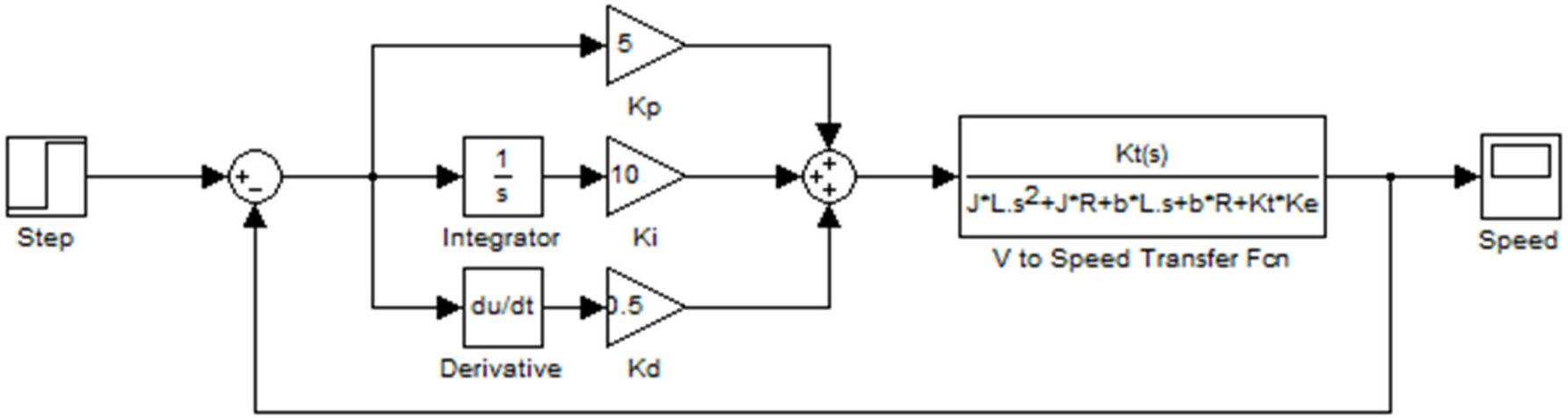
Simulation



Experiment

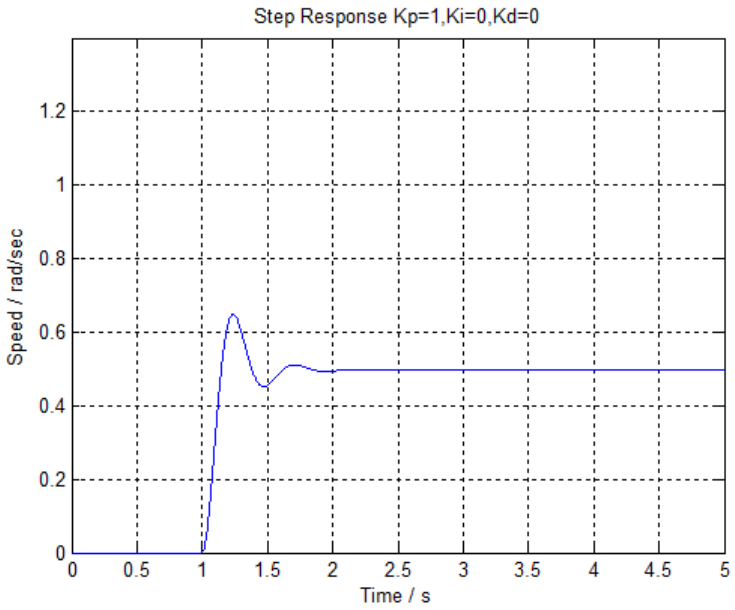
Speed Control (2)

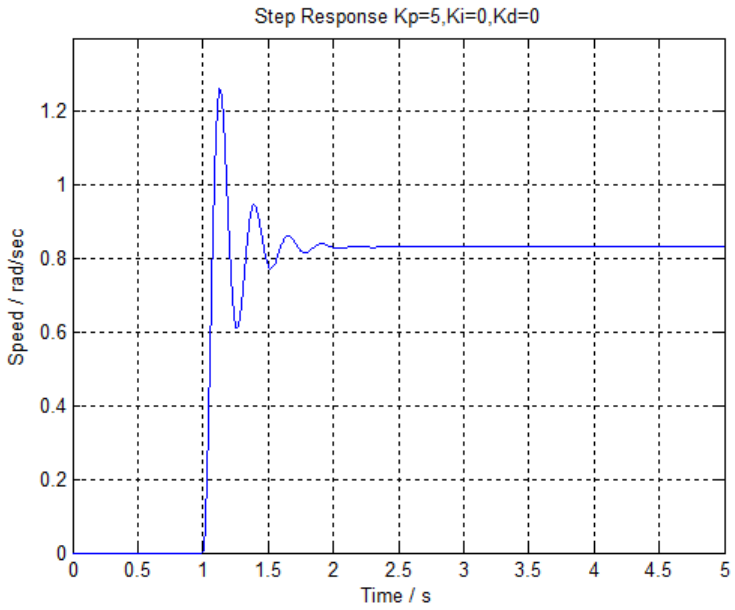
- Build the following system in Simulink



Speed Control- PID Tuning (1)

- Start with P first:

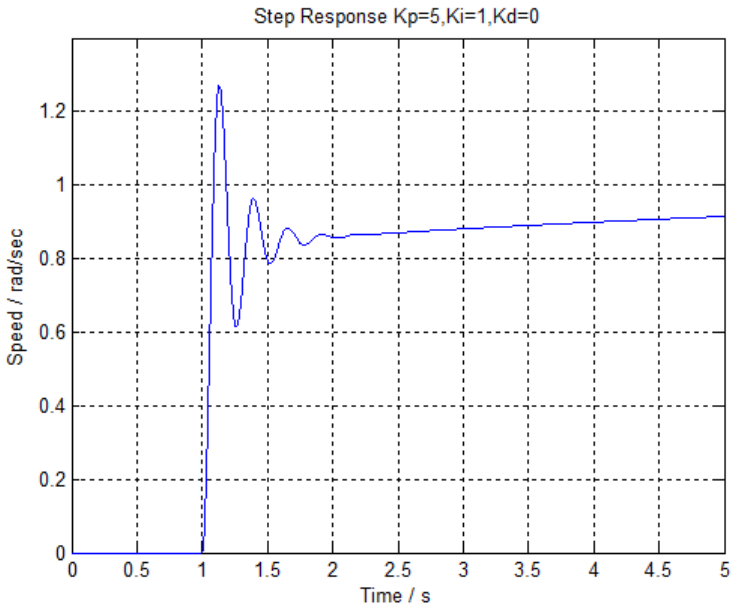


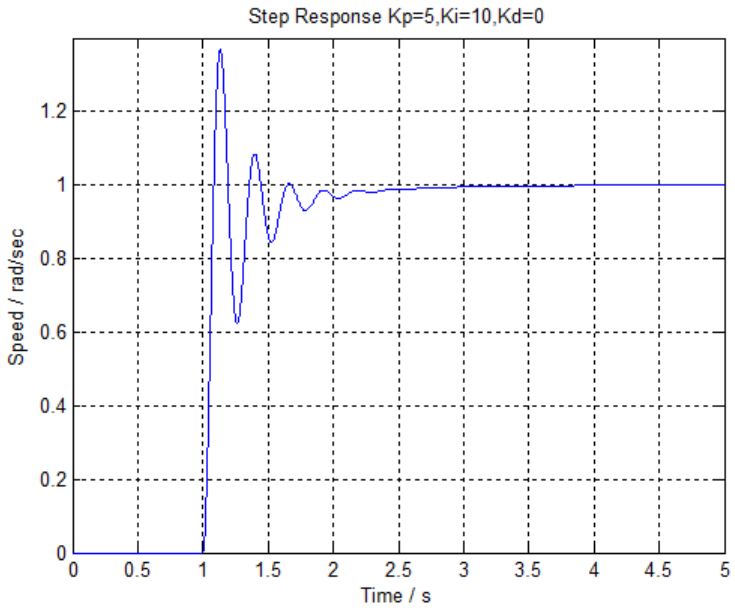


RMIT UNIVERSITY

Speed Control- PID Tuning (2)

- Tune I next:

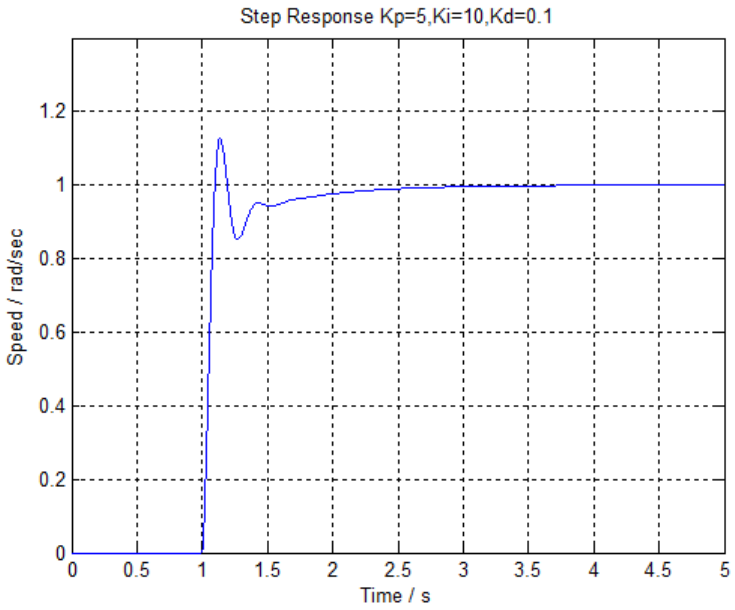


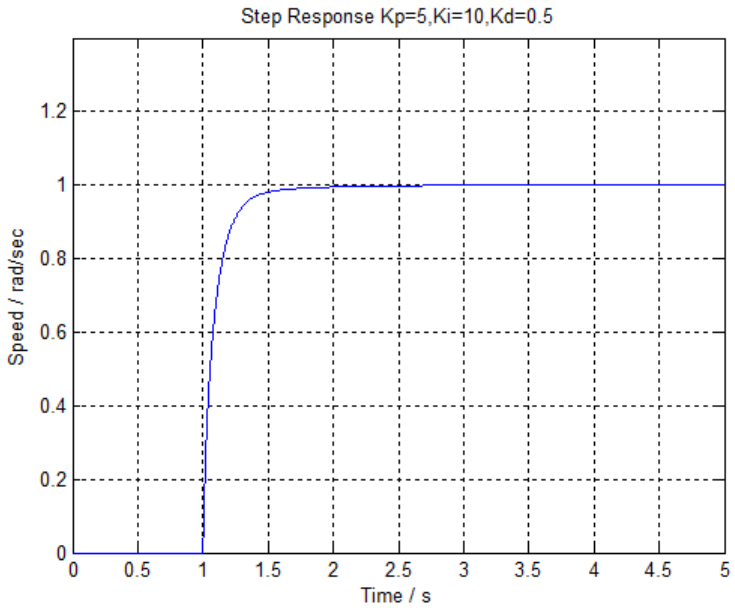


RMIT UNIVERSITY

Speed Control- PID Tuning (3)

- Finally, tune D:

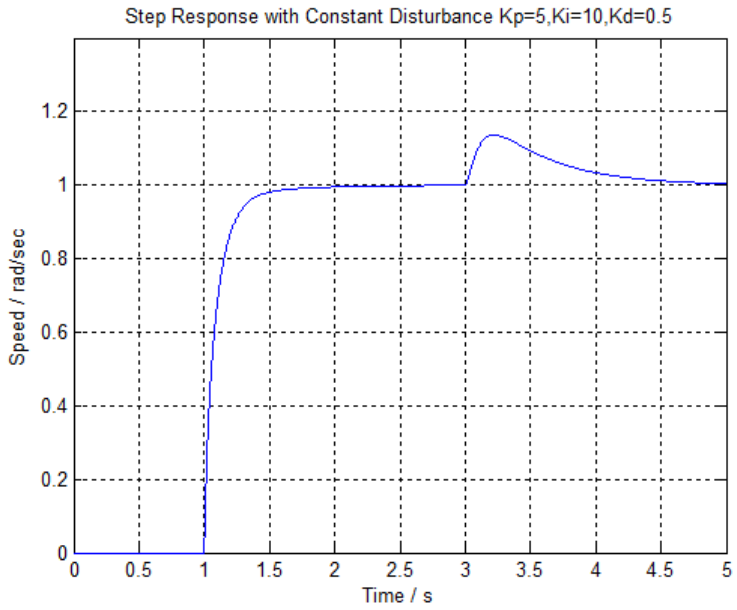




RMIT UNIVERSITY

Speed Control- Disturbance and PID

- If there is an constant input disturbance at t = 3s:

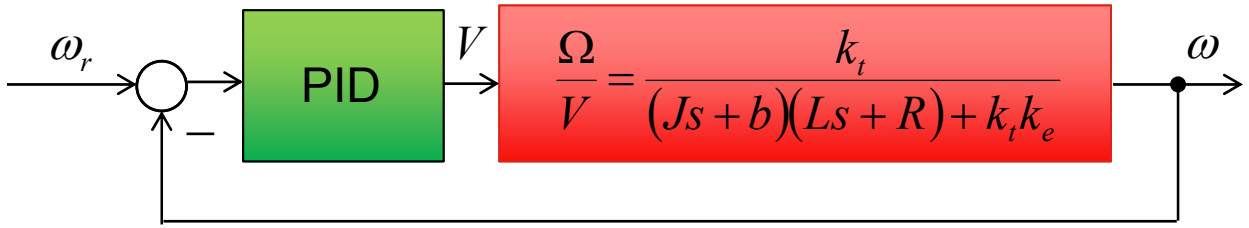


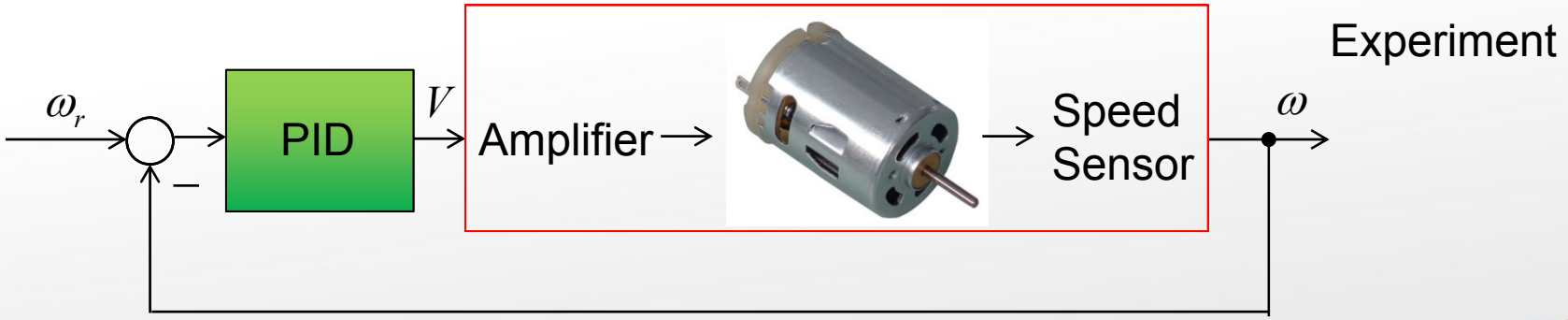
- The speed returns to desired speed. This is attributed to the I-control.

RMIT UNIVERSITY

Remark

- Remember that the model is a close representation of the actual system.
- If the control works well in simulation, it should work fine too when implemented on the actual system.





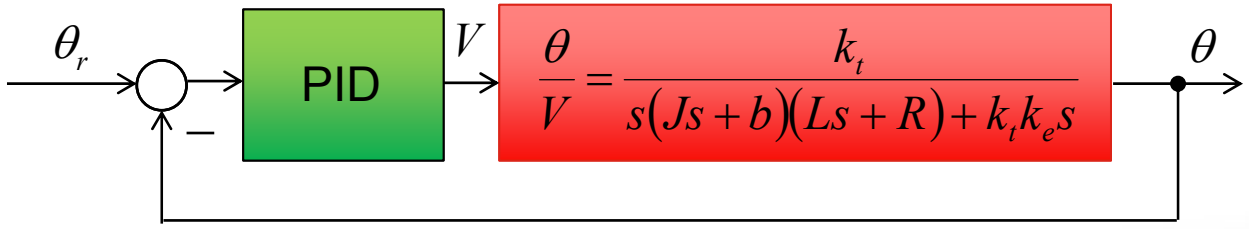
RMIT UNIVERSITY

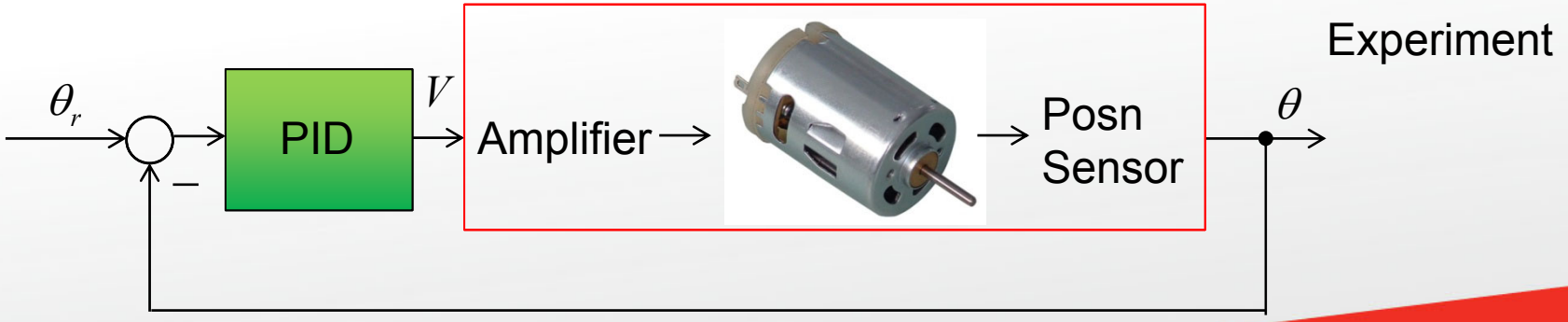
Content

- Modeling of DC Motors
  - MATLAB / Simulink Simulation
- Speed Control of DC Motors
  - MATLAB / Simulink Simulation
- Position Control of DC Motors
  - MATLAB / Simulink Simulation
- Force Control of DC Motors
  - MATLAB / Simulink Simulation

Position Control (1)

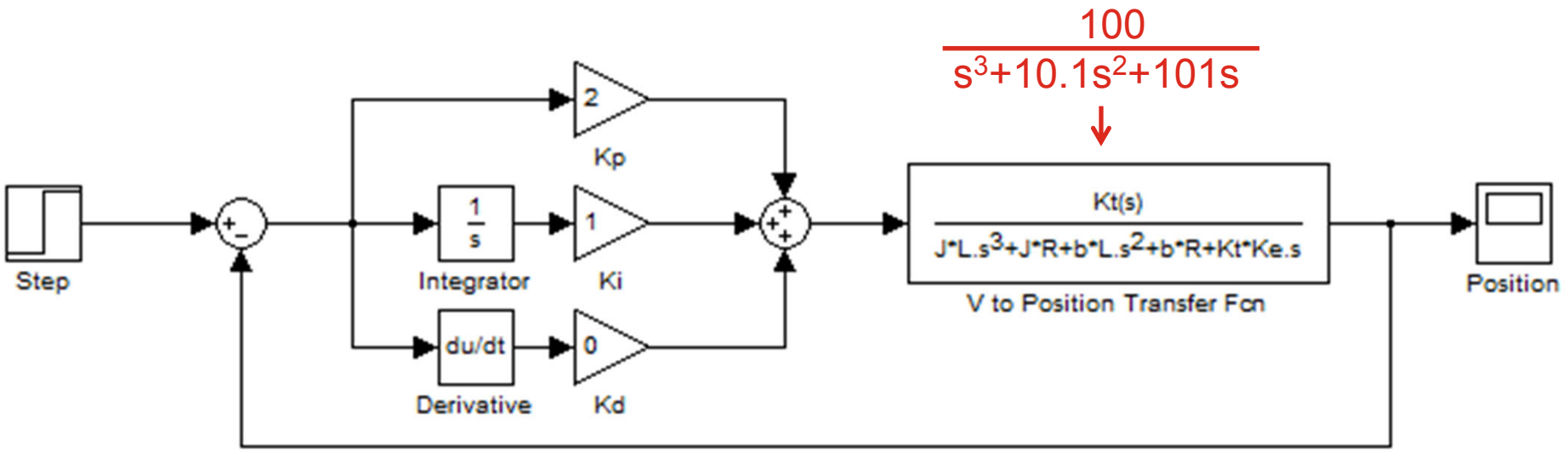
- Just now, we saw that when we supply a constant V to DC motor, the motor angle will keep increasing.
- What if we want to control the motor angle to a definitive value?
  - → Feedback control!





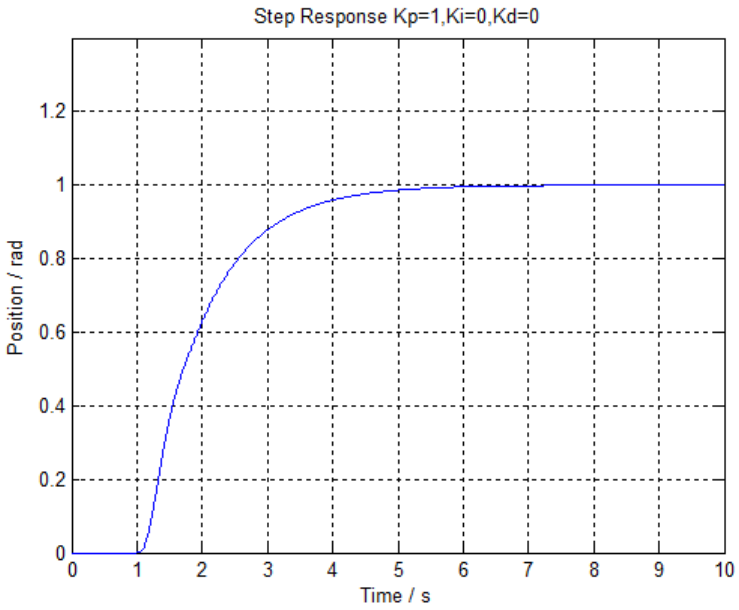
Position Control (2)

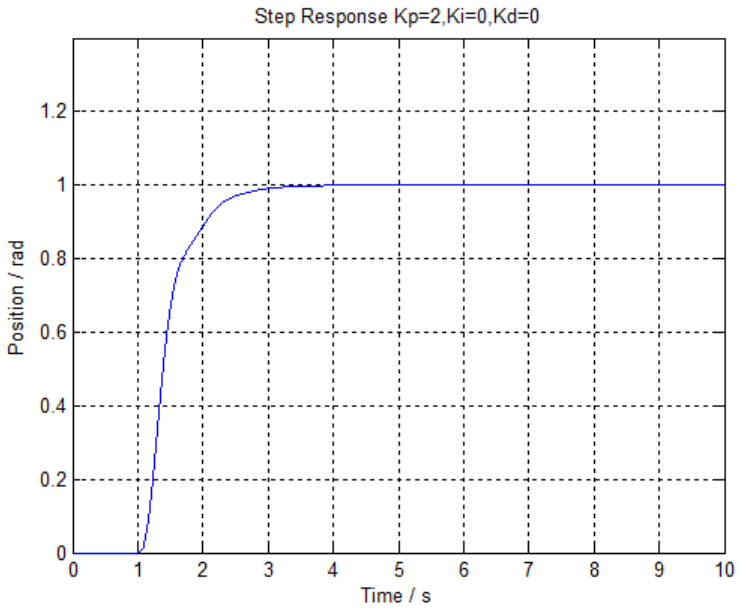
- Build the following system in Simulink



Position Control- PID Tuning (1)

- Start with P first:



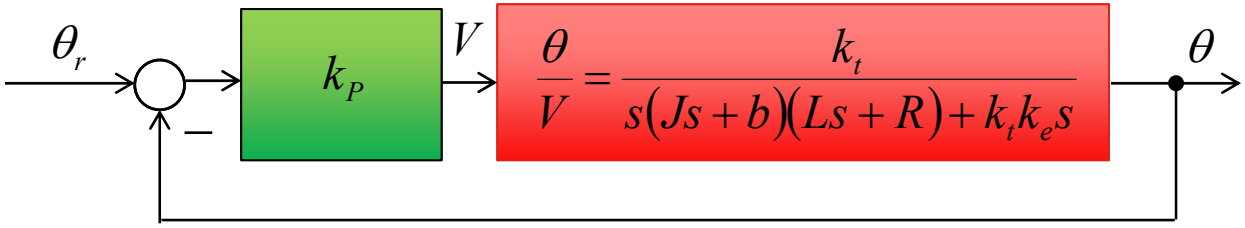


- The position reaches the desired position with only P-control.
- This is because the closed-loop gain is one.

RMIT UNIVERSITY

Aside: Closed-Loop Gain

- The closed loop transfer function, with only a P-control is:



- Closed loop transfer function in a loop like above is:

$$\frac{\theta}{\theta_r} = \frac{\text{forward}}{1+\text{forward}\cdot\text{backward}} = \frac{\dfrac{k_P k_t}{s(Js+b)(Ls+R)+k_t k_e s}}{1+\dfrac{k_P k_t}{s(Js+b)(Ls+R)+k_t k_e s}\cdot 1} = \frac{k_P k_t}{s(Js+b)(Ls+R)+k_t k_e s+k_P k_t}$$

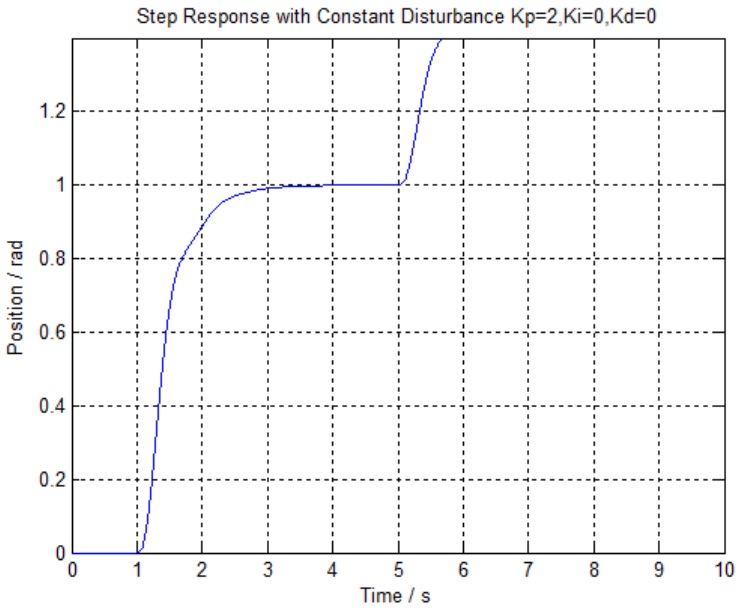
- To obtain the DC gain, we equate all the s as zero. Therefore:

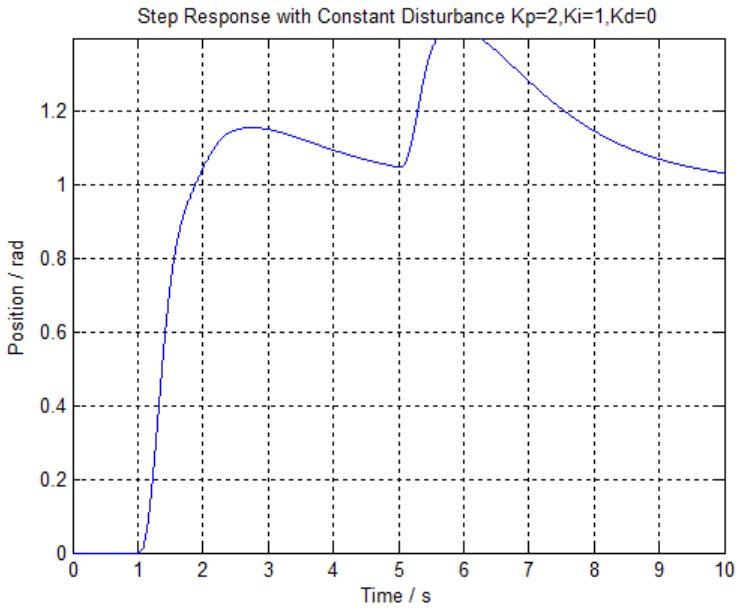
$$\left.\frac{\theta}{\theta_r}\right|_{DC} = \frac{k_P k_t}{k_P k_t} = 1$$

RMIT UNIVERSITY

Position Control- Disturbance and PID

- If we add a constant input disturbance, the position will move away from the desired position.





- We need I-control to make sure that the steady state error becomes zero.

RMIT UNIVERSITY

Content

- Modeling of DC Motors
  - MATLAB / Simulink Simulation
- Speed Control of DC Motors
  - MATLAB / Simulink Simulation
- Position Control of DC Motors
  - MATLAB / Simulink Simulation
- Force Control of DC Motors
  - MATLAB / Simulink Simulation

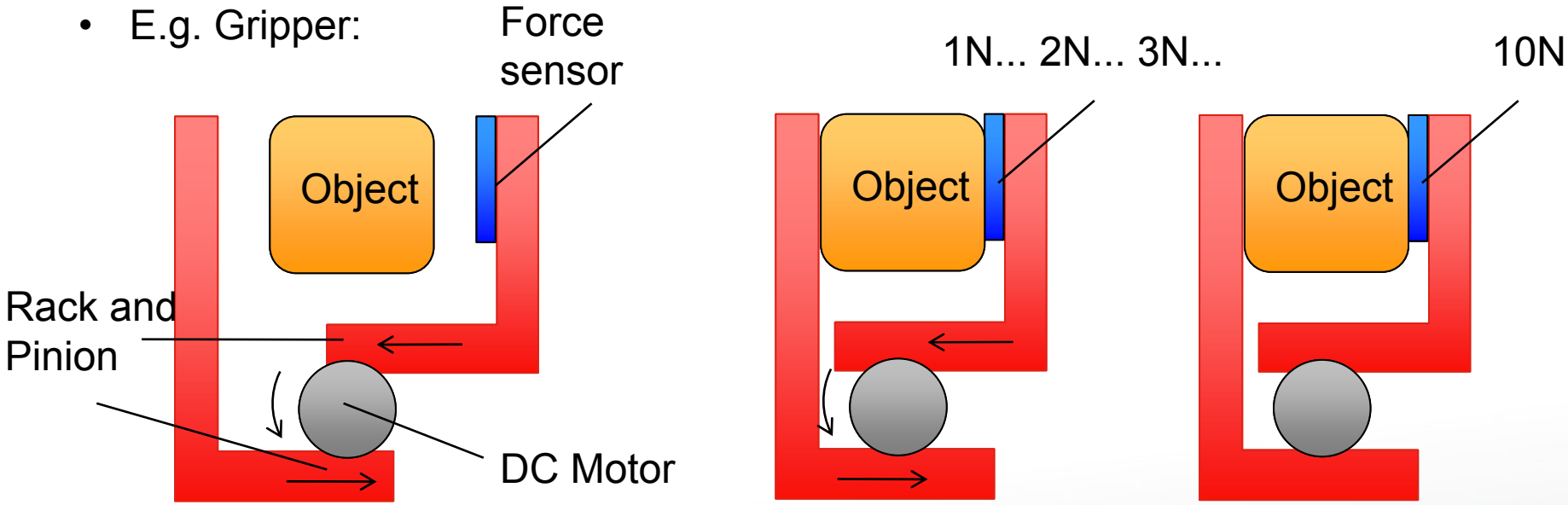
RMIT UNIVERSITY

Force Control (1)

- The DC motor has been shown to be useful for position control and speed control applications.
- Nowadays, force control applications have become very popular. E.g.
  - Robotic manipulator senses external force and reacts accordingly.
    - https://www.youtube.com/watch?v=vX_6tOrFmEg
  - Gripper grasp an object with definite force without breaking it.
    - https://www.youtube.com/watch?v=NgZqVjcI7P0
- How it works: Similar concept to position / speed control → Feeback!

Force Control (2)

- Uses force sensor for feedback.
- E.g. Gripper:



1. Desired force is 10N. At this moment there is no contact, thus actual force is 0N. Controller gives a positive V and motor turns. Gripper closes.

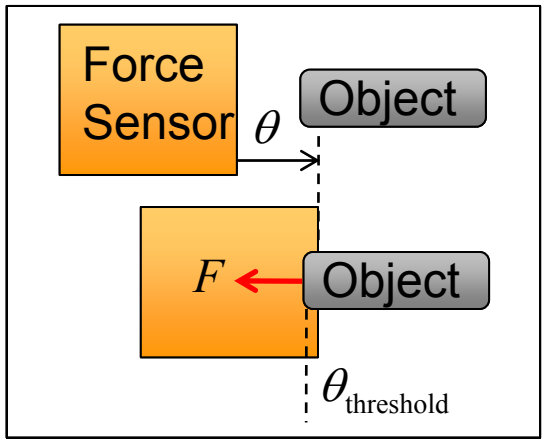
2. Gripper finally touches the object, but force only starts to build up, not 10N yet. Controller still outputs positive V.

3. Gripper now grips the object with 10N. Controller tells the motor to stop turning.

RMIT UNIVERSITY

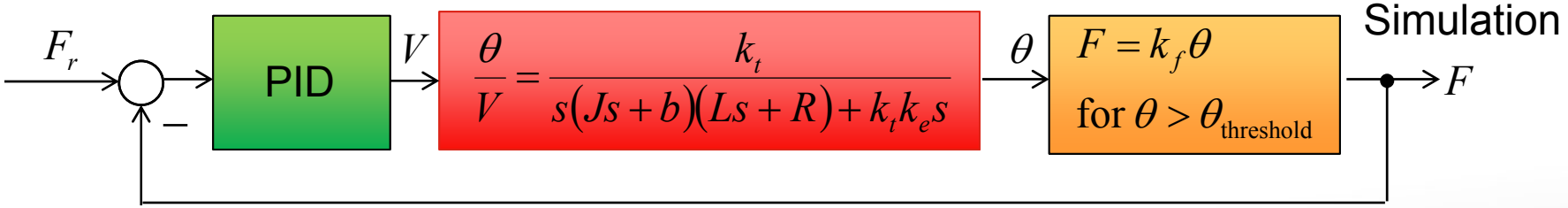
Force Control (3)

- This can be achieved again using the PID controller.

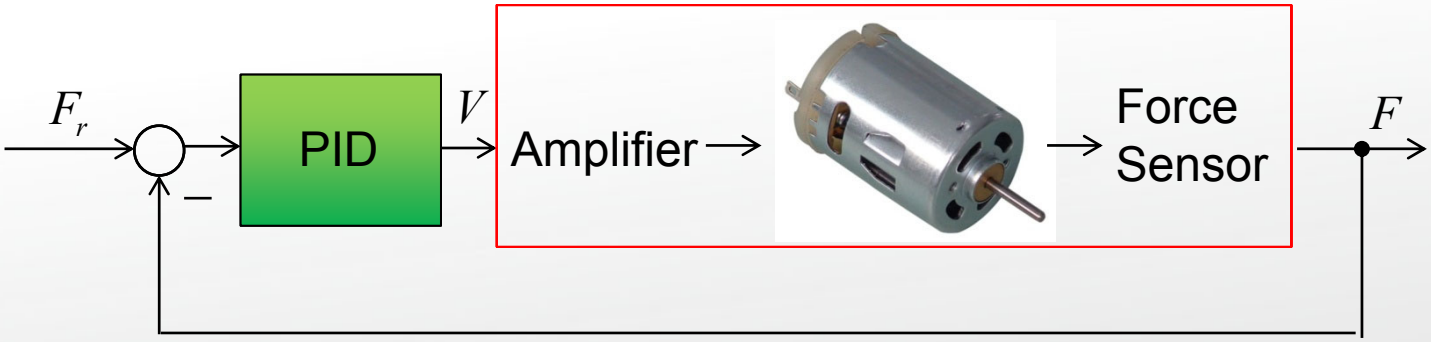


Force Sensor Model

Simulation



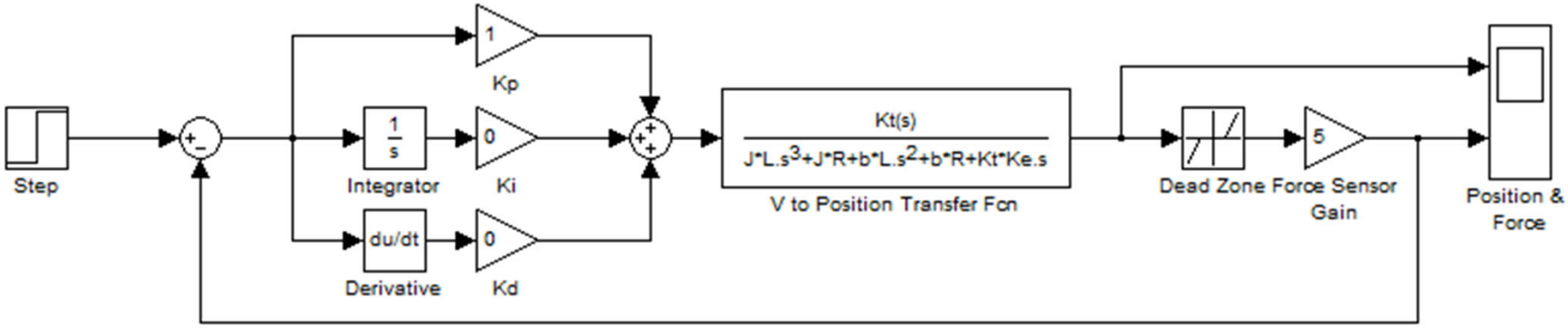
Experiment



RMIT UNIVERSITY

Force Control (4)

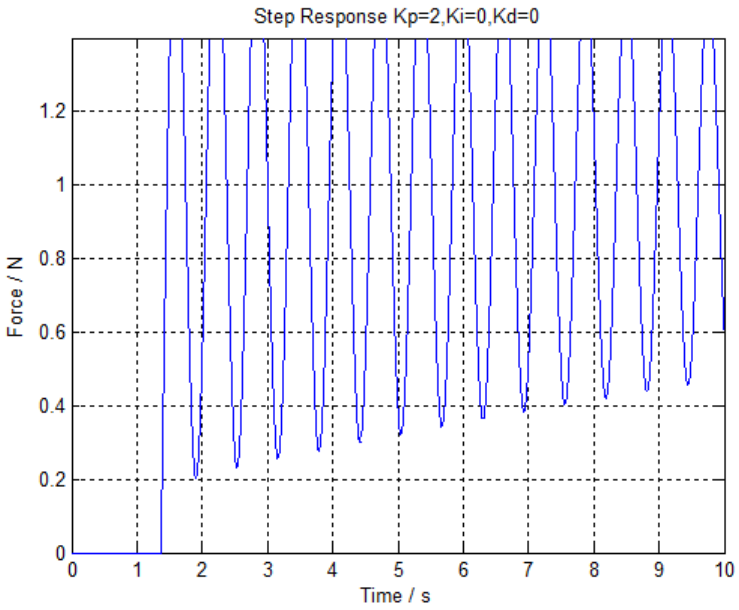
- Build the following system in Simulink

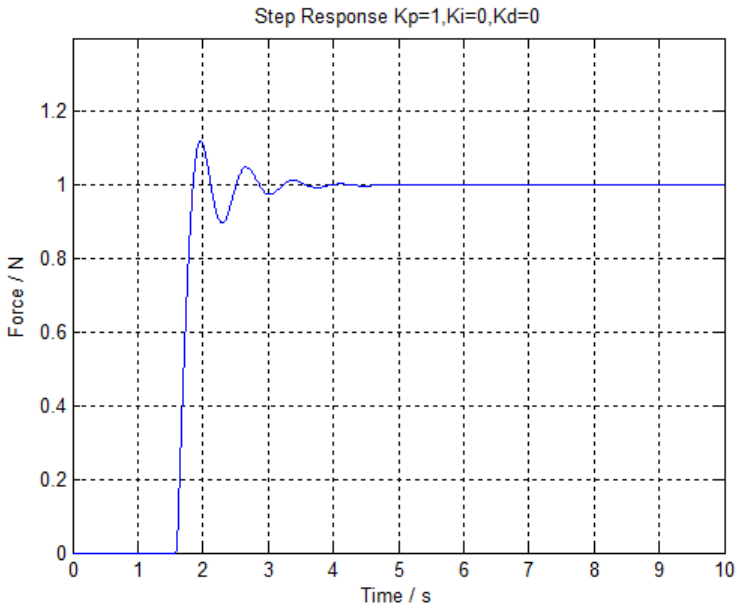


- The “Dead Zone” is to simulate the case where the motor is not touching the object initially.
- Set start of dead zone = -10000, end of dead zone = 0.5.
- Motor needs to rotate 0.5 rad before gripper touches the object.

Force Control- PID Tuning

- Start with P first:



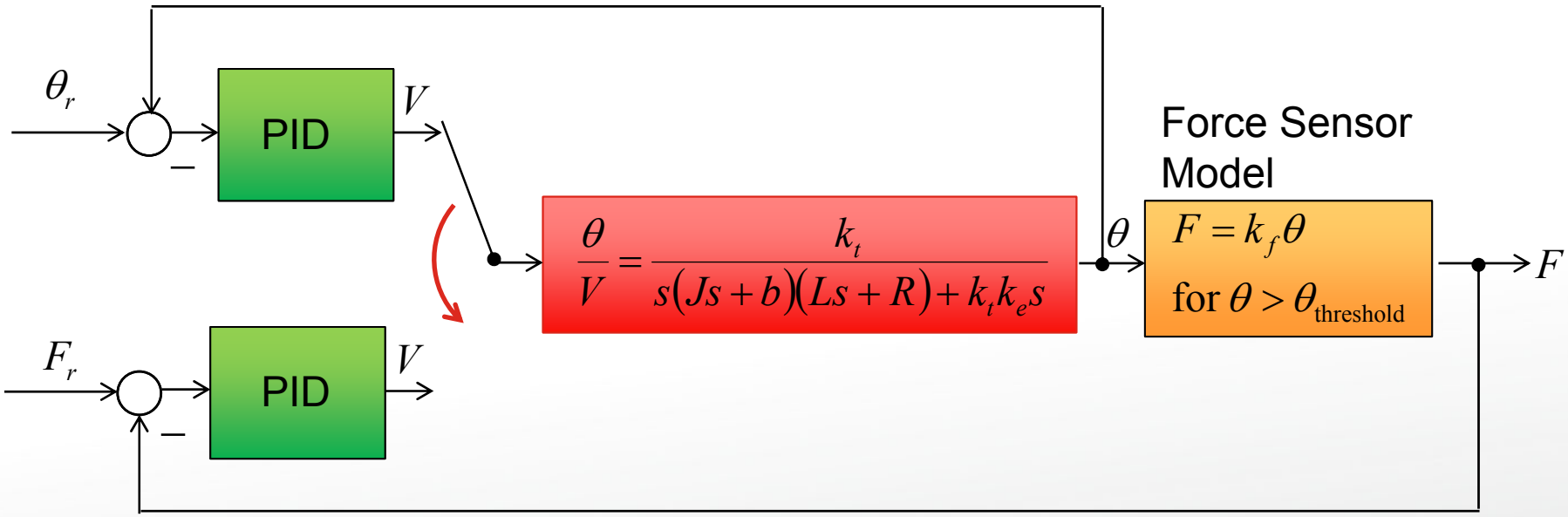


- Satisfactory performance after tuning.

RMIT UNIVERSITY

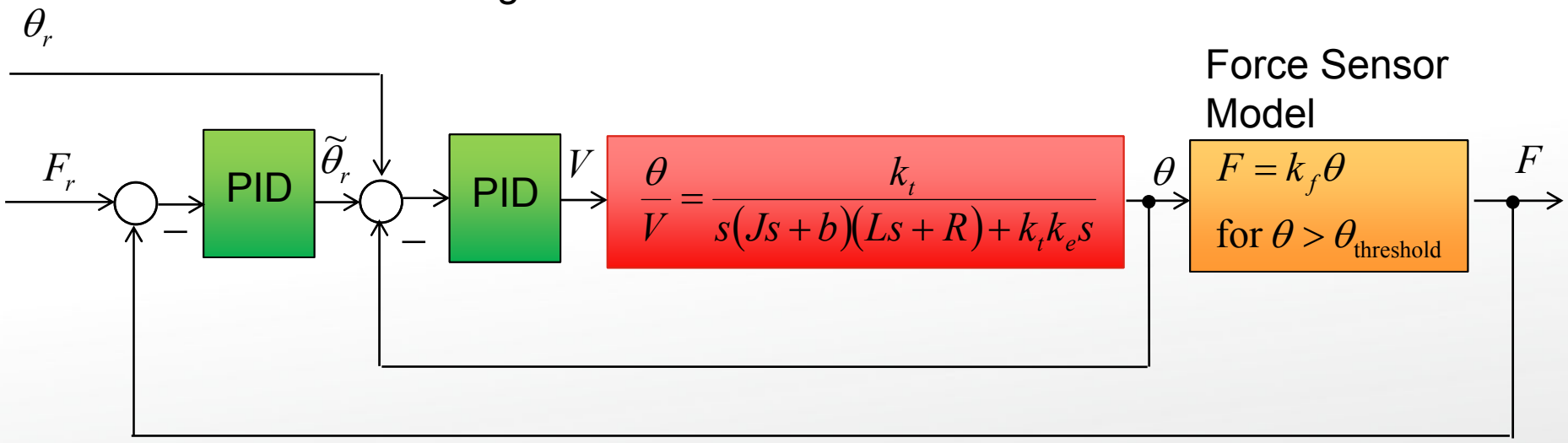
Hybrid Position / Force Control

- To switch between position and force control, as required by different tasks.



Parallel Position / Force Control

- Cascaded control system
  - When force controller is turned off, runs in position control mode.
  - When force controller is turned on, it changes the reference position so as to achieve the desired force: Forward if force is not strong enough, retreat if force is too strong.



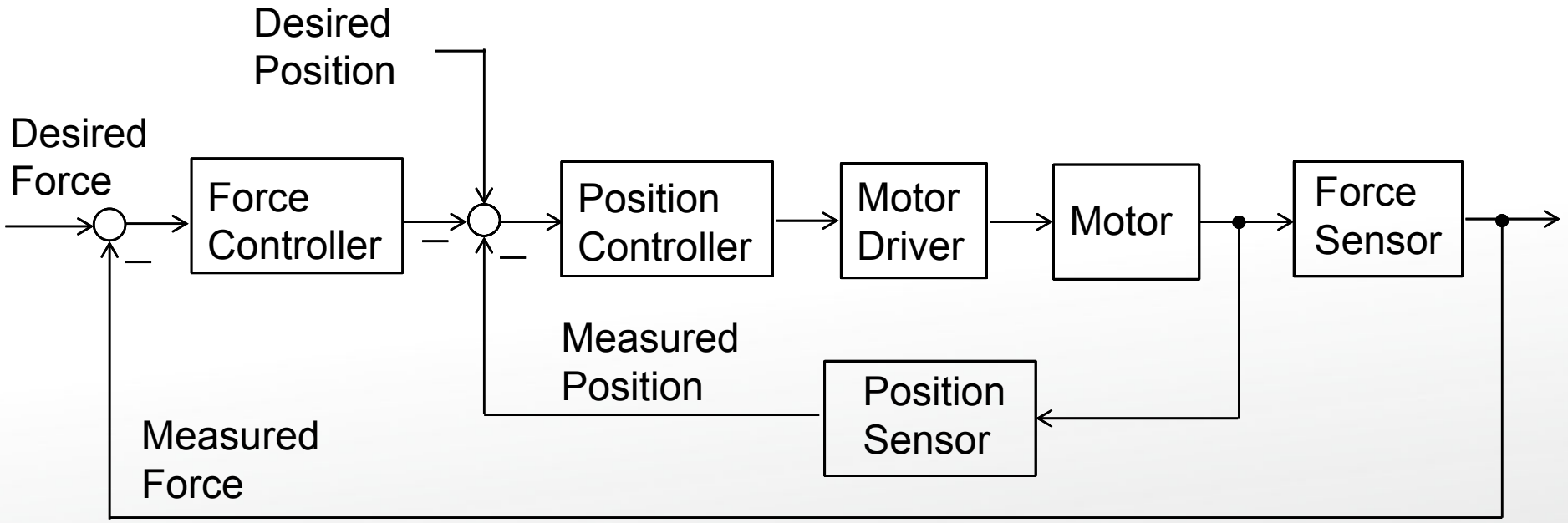
RMIT UNIVERSITY

Parallel Position / Force Control

How does this work?

Firstly, assume force controller is not active (all parameters zero).

You set the desired position (e.g. 100mm), then the position controller opens the gripper so that the gap is 100mm.

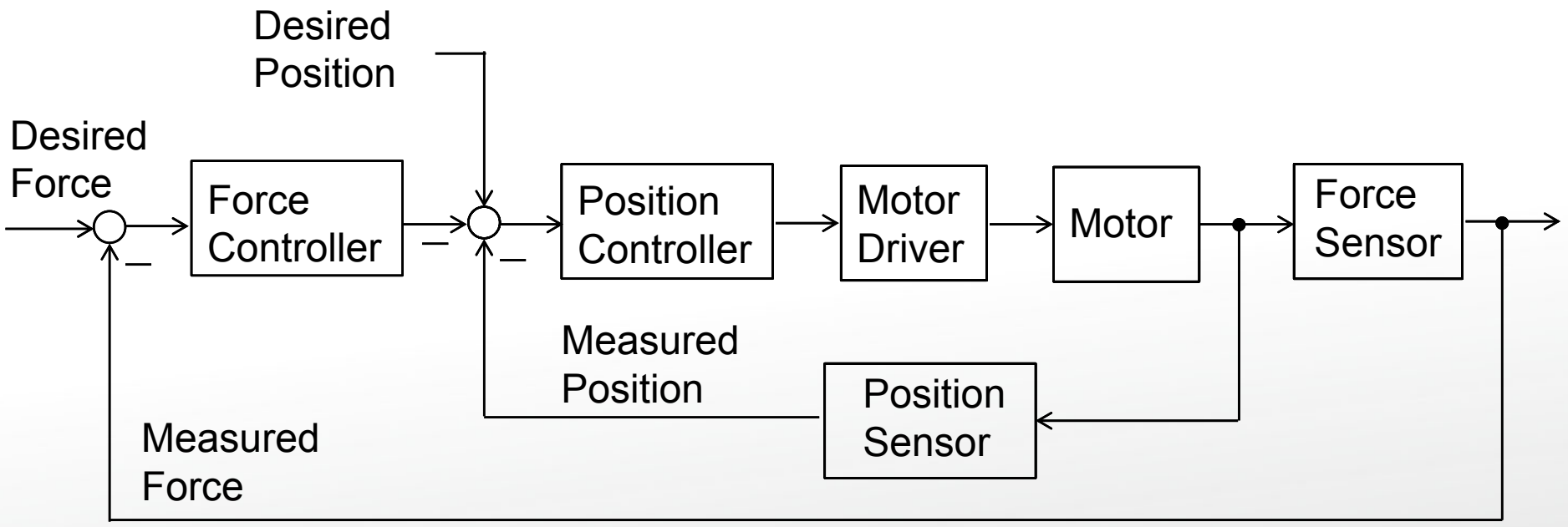


Parallel Position / Force Control

Next, assume you set the desired force to be 20N.

Activate the force controller (with nonzero parameters).

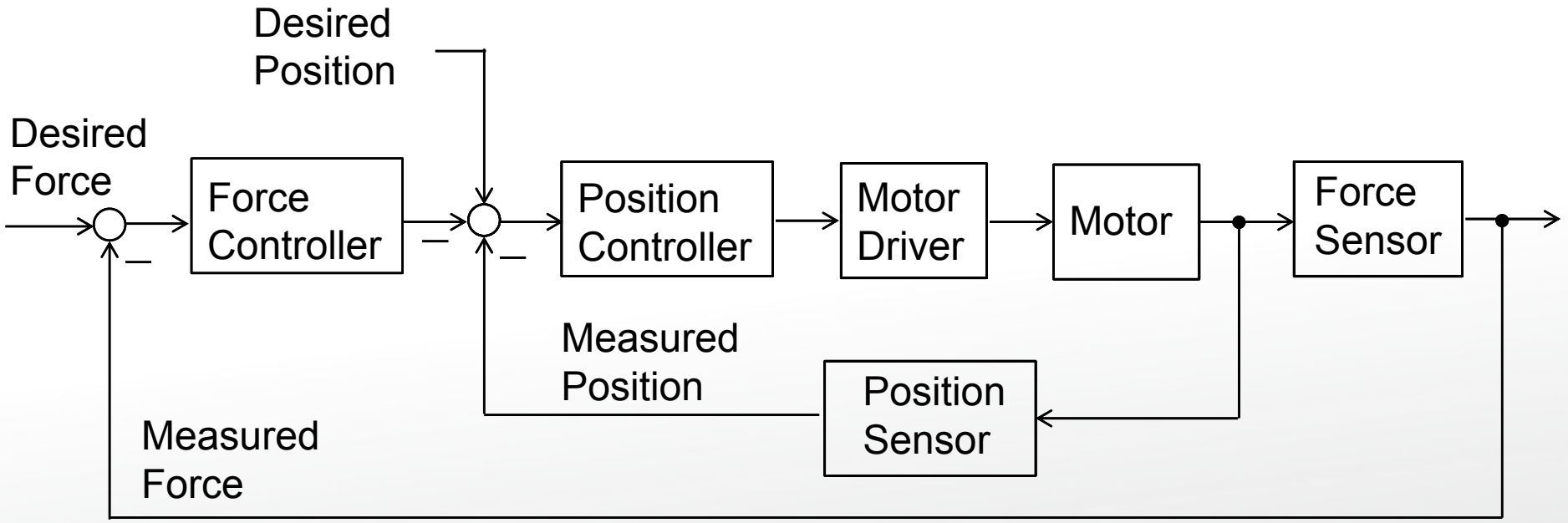
Initially, the gripper isn't touching anything thus the measured force is 0N.



Parallel Position / Force Control

The force error is therefore (20N – 0N) = 20N.

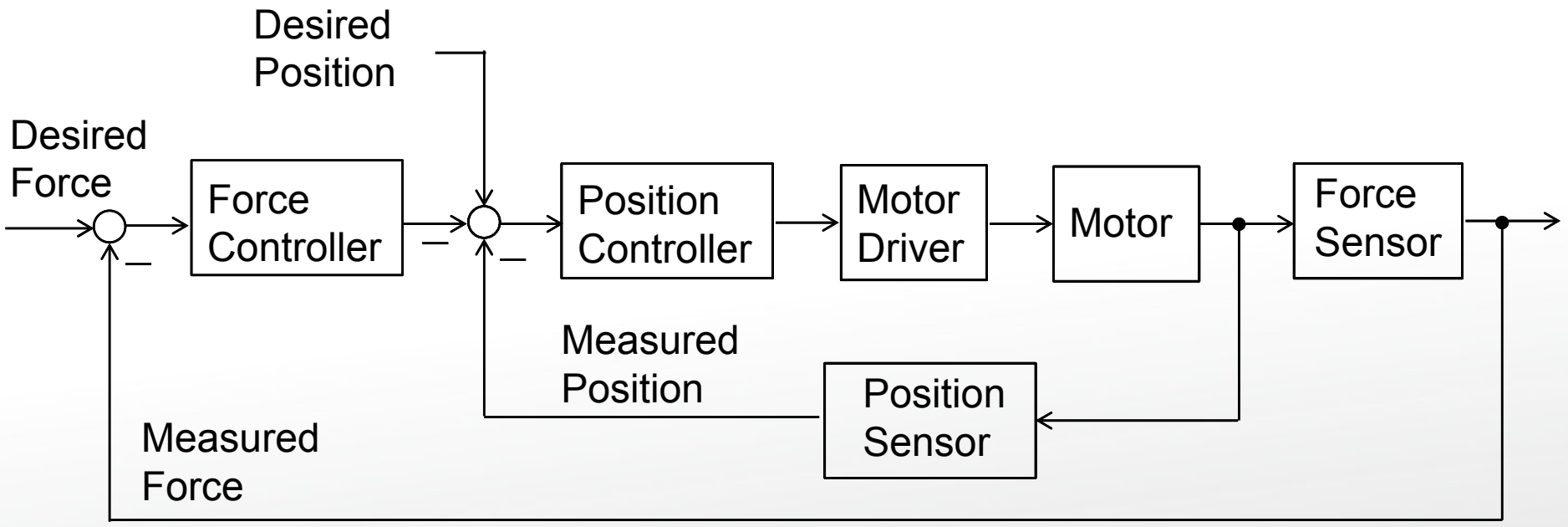
The force controller acts upon this error, and provides an output which is "subtracted from" the original desired position → the reference position becomes smaller.



Parallel Position / Force Control

Since the reference position becomes smaller, the position controller makes the gripper close its gap, until finally it grasps the object with the desired force of 20N.

Note: The force controller is usually just an integrator!



Thank you!

Any questions?

RMIT UNIVERSITY