Control Systems using Simulink

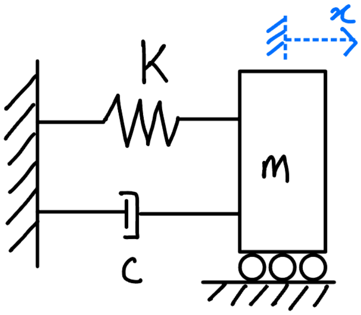
A control system allows the desired operation of a system to be achieved and maintains these desired operations in the event of a disturbance. The control system can be mechanical such as a float valve, biological (homeostasis) such as your body's temperature, electrical such as a computer fan, or manually such as someone visually observing a measuring instrument (such as a gauge) and adjusting (by turning a valve for example). For the engineering design of automatic control systems, the system (plant) we are trying to control must be modeled mathematically such that the control mechanism can be adequately chosen, tailored, implemented, and tested. Testing can be done physically by building components. This is a costly and time-consuming approach and is not suited for tuning the system. Virtual Prototyping involves building the system in the virtual space using software such as MATLAB/Simulink where the system to be controlled (plant) and the control mechanism are modeled. This module recaps system modelling and details how these systems of equations are implemented in Simulink. Control schemes are then implemented to maintain desired operation during disturbances.

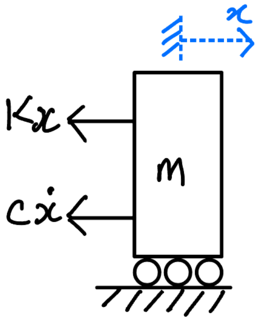
System Modeling Recap

Using theoretical laws and reasonable abstraction, systems can be mathematically represented by a system of equations. In the software space, these equations make up the virtual model with the solution to these equations being the behaviors of the model to inputs. In this section modelling of a spring-mass system and a DC motor are recapped followed by implementation in Simulink.

Spring Mass System

Let us consider a mass m (kg) attached to a spring with stiffness k (N/m) and a viscous damper with damping constant c (N/m/s):





Free body diagram

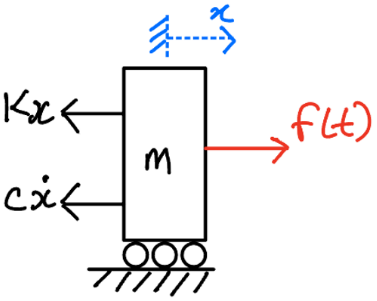
Newton's 2nd Law:

$$\sum F_x = ma$$

$$-Kx - c\dot{x} = m\ddot{x}$$

$$m\ddot{x} + c\dot{x} + Kx = 0$$

Now what if a force f (N) is applied to the mass as a *disturbance:*



$$-Kx - c\dot{x} + f(t) = m\ddot{x}$$

$$m\ddot{x} + c\dot{x} + Kx = f(t)$$

$$\ddot{x} = \frac{1}{m}\left[-c\dot{x} - Kx + f(t)\right]$$

In a **linear** control system design, we would express this equation as a *transfer function*. In this representation, the disturbance force will be treated as an *input*.

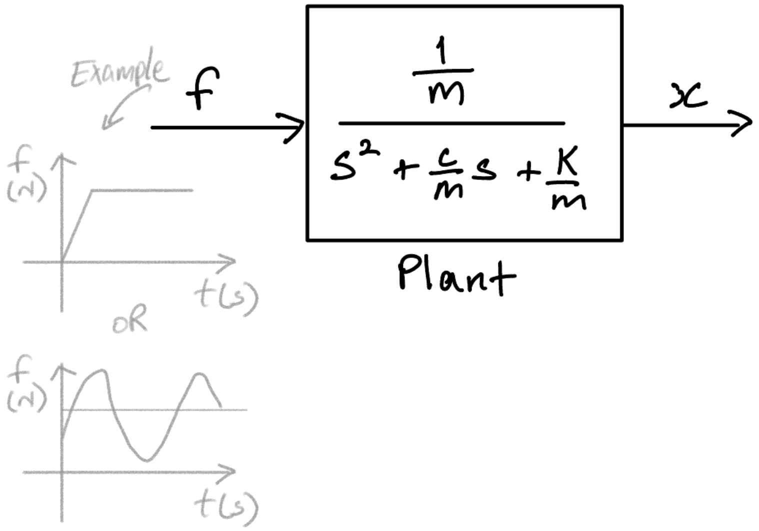
$$mD^2x + cDx + Kx = f(t)$$

$$x\,[mD^2 + cD + K) = f(t)$$

$$TF = \frac{\text{Output}}{\text{Input}} = \frac{x(t)}{f(t)}$$

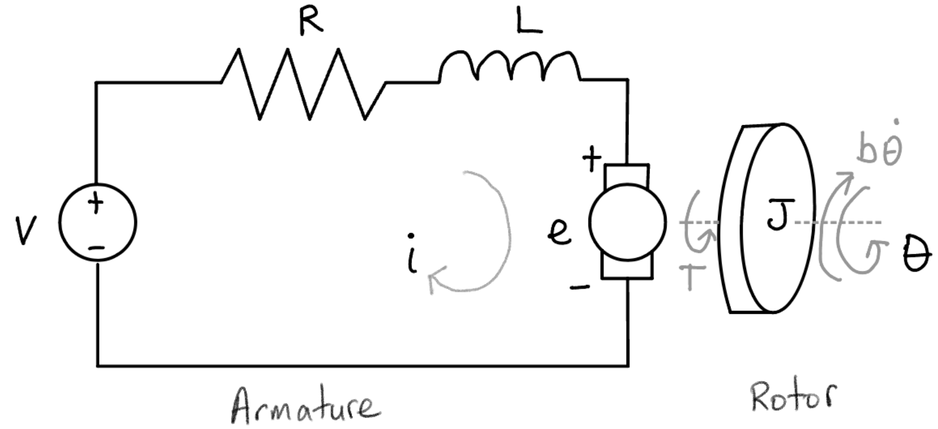
$$\frac{x(t)}{f(t)} = \frac{1}{MD^2 + cD + K}$$

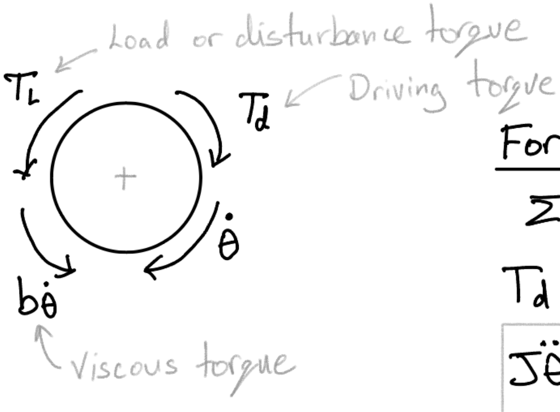
$$\boxed{\frac{X(s)}{F(s)} = \frac{\frac{1}{m}}{s^2 + \frac{c}{m}s + \frac{K}{m}}}$$



DC Motor

Let us model a DC motor assuming a coil resistance R (ohms), inductance L (H) and generates a back emf e (v) when rotating at a speed $\dot{\theta}$ (rad/s). The rotor has mass moment of inertia J (kg.m^2) with a viscous friction coefficient b (N.m/rad/s).



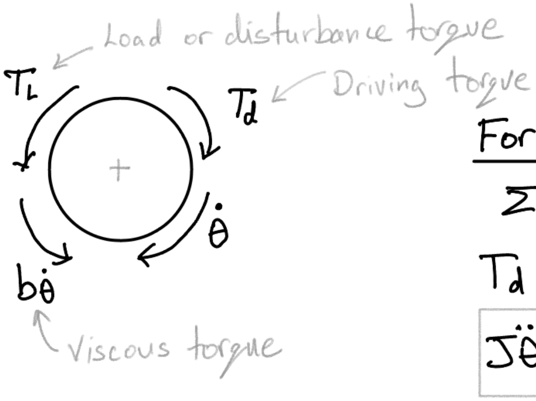


For the rotor:

$$\Sigma T = J\ddot\theta$$

$$T_d - b\dot\theta - T_L = J\ddot\theta$$

$$\boxed{J\ddot\theta = T_d - b\dot\theta - T_L}$$

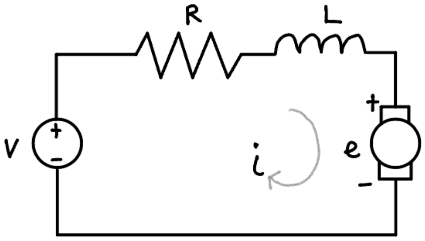


For the rotor:

$$\Sigma T = J\ddot{\theta}$$

$$T_d - b\dot{\theta} - T_L = J\ddot{\theta}$$

$$\boxed{J\ddot{\theta} = T_d - b\dot{\theta} - T_L}$$



For the armature:

Kirchoff's Voltage law:

$$\Sigma V_{loop} = 0$$

$$Ri + L\frac{di}{dt} + e - v = 0$$

$$\boxed{L\frac{di}{dt} = -Ri + v - e}$$

If we assume $T_d \propto i$ and $e \propto \dot{\theta}$, the equations become coupled:

$$J\ddot{\theta} = K_t i - b\dot{\theta} - T_L \quad \text{———} \quad ①$$

$$L\frac{di}{dt} = -Ri + V + K_e\dot{\theta} \quad \text{———} \quad ②$$

Letting $\dot{\theta} = \omega$:

$$J\dot{\omega} = K_t i - b\omega - T_L \quad \text{———} \quad ③$$

$$L\frac{di}{dt} = -Ri + V + K_e\omega \quad \text{———} \quad ④$$

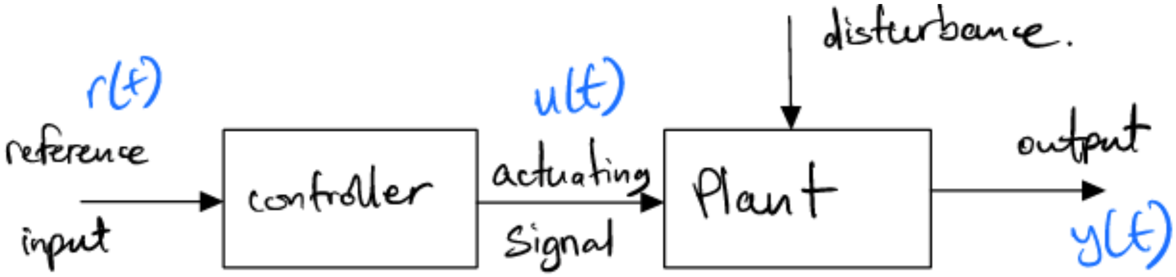
From which the TF: $\frac{\Omega(s)}{V(s)}$ can be obtained.

For more complicated systems, explicitly expressing the output can become cumbersome. Methods such as state-space modelling does however simplify this process. Simulink allows a system of equations to be graphically represented, thereby allowing for quicker equation modifications. Such modifications may include multiple inputs/disturbances and measuring various parameters not explicitly solved for when deriving the transfer function.

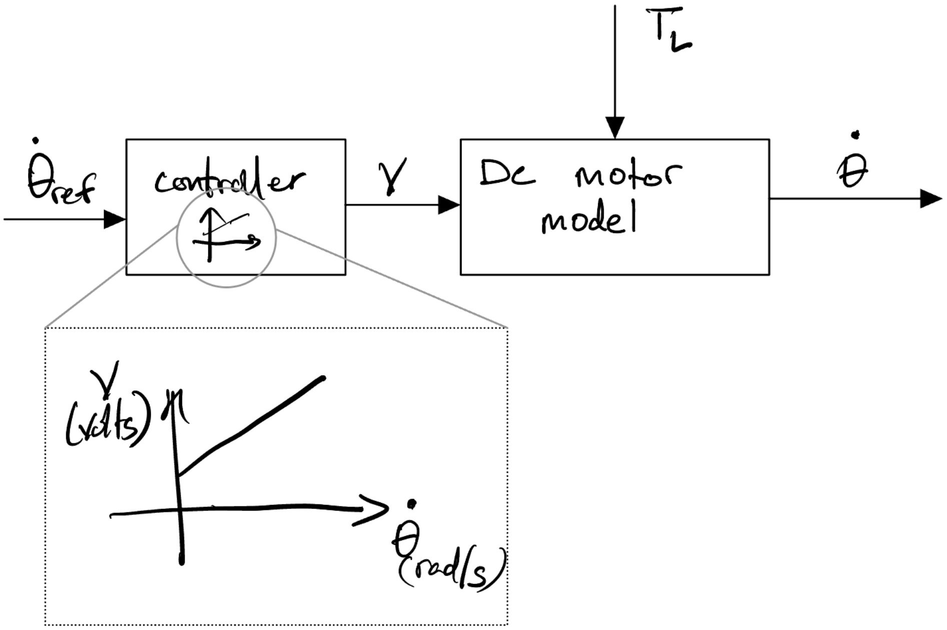
Control Systems

Open-Loop

In an open-loop system, the controller converts the reference signal, that is the parameter you wish to control, to the input of the plant. The plant output is not monitored and therefore the system cannot self-correct in the event of a disturbance.

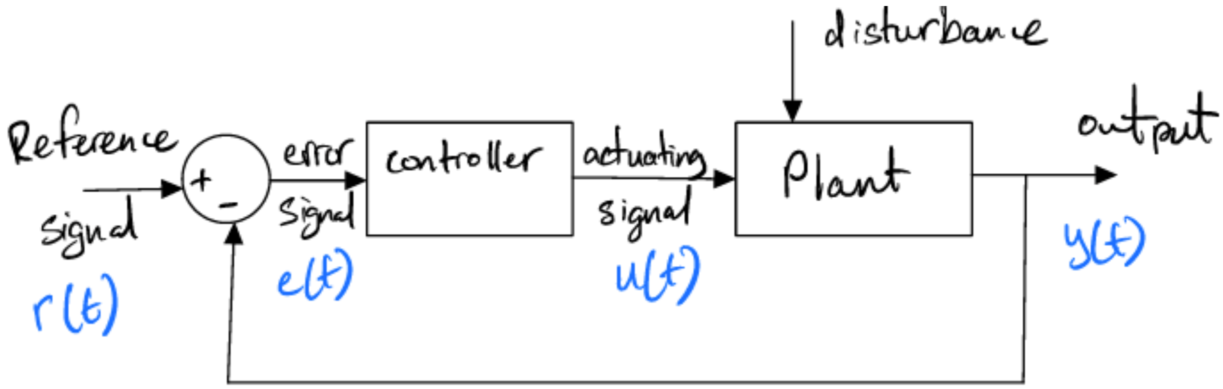


In the case of the DC motor:

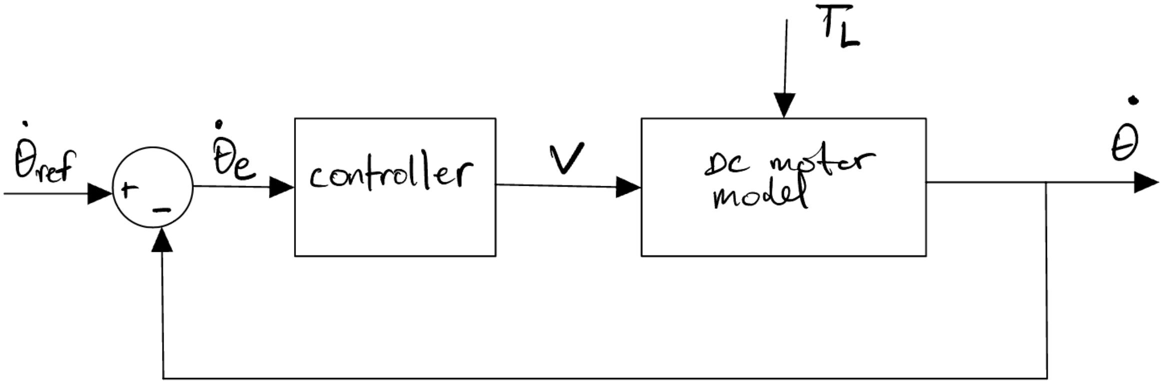


Closed-Loop

In a closed-loop system, the controller usually responds to the error signal generated by comparing the reference signal to the plant output. The output is continuously monitored and applies a correction to the input signal thereby restoring the output (if it needs to) to that of the reference.



In the case of the DC motor:



The controller’s purpose is to bring the error signal to 0.

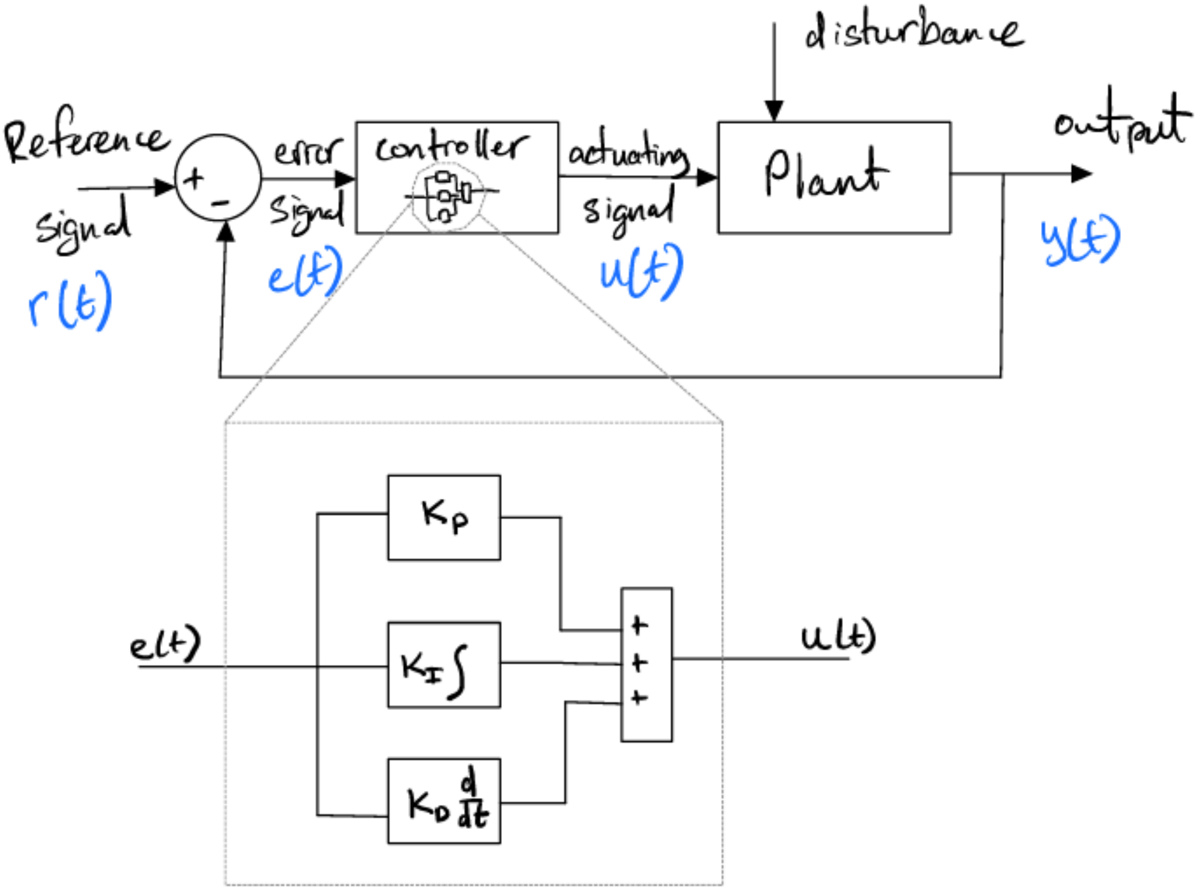
Proportional, Integral and Derivative (PID) Control Recap

PID control is the most common control scheme. PID control can be implemented on microprocessors or more traditionally, be built using analog electronics such as op-amps. The controller consists of 3 elements.

- Proportional - P (tuning parameter or gain K_P)
- Integral - I (tuning parameter or gain K_I)
- Derivative - (tuning parameter or gain K_D)

The effect of these elements on the error signal, e(t), is then summed to give the actuator signal u(t).

$$u(t) = K_p e(t) + K_I \int e(t)\,dt + K_D \frac{de(t)}{dt}$$

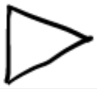


PID control is best suited for linear models. When it is to be used with a non-linear model, the model is linearized about an operating point. A well-tuned PID controller will perform optimally when there are only small deviations about this operating point such as a small change in the reference signal, a small disturbance, or both. Take for example an aircraft flying at level flight. The equations of motion are highly nonlinear but can be linearized about a pitch angle of 0 degrees. If there are small moments applied during flight (from wind gusts for example) then the PID controller will be able to send correction actuation signals to the control surfaces to restore the aircraft to its level flight. This is possible by designing the controller with a linearized model of the aircraft about the level flight angles and tuning the gains (K_P, K_I and K_D) to give the optimal response when perturbed about the point. If the disturbance is high enough to displace the aircraft significantly beyond the linearize point (in this case the level flight angles), then the tuned parameters will not be able to adequately restore the aircraft as desired. To mitigate this, adaptive controllers are used instead of PID controllers as the controller's tunable parameters or gains can adapt to a moving linearizing point. For example, a PID control scheme that runs optimizations in real-time that changes the gains as the aircraft if significantly perturbed.

Equations in Simulink

Please ensure you have completed and uploaded your Simulink OnRamp certificate on CANVAS before proceeding. This section serves as a Simulink recap before proceeding to control system design.

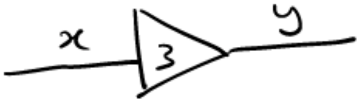
Some symbols frequently used

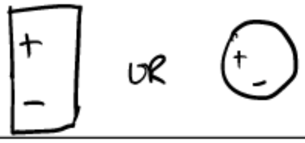


- gain or product block:

Example:

$y = 3x$:

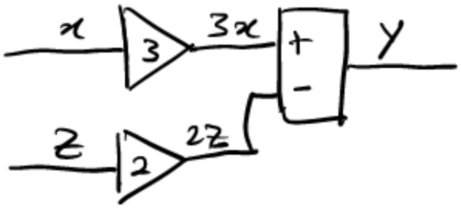


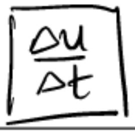


- summation block:

Example:

$y = 3x - 2z$:



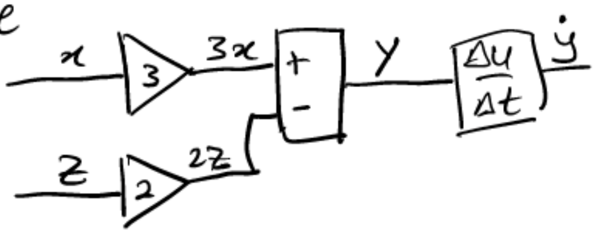


- Derivative block:

Example:

output is $\dot{y}$ where

$y = 3x - 2z$

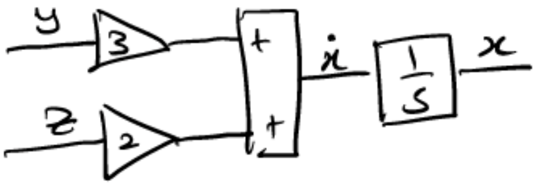


$\boxed{\frac{1}{s}}$ – Integral block.

example:

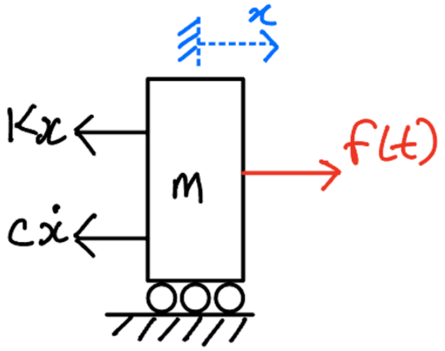
output is x where

$\dot{x} = 3y + 2z$



Inputting a differential equation in Simulink

Exercise 1 – Input the spring-mass-damper equation in Simulink

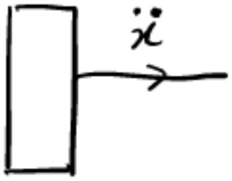


$$-Kx - c\dot{x} + f(t) = m\ddot{x}$$

$$m\ddot{x} + c\dot{x} + Kx = f(t)$$

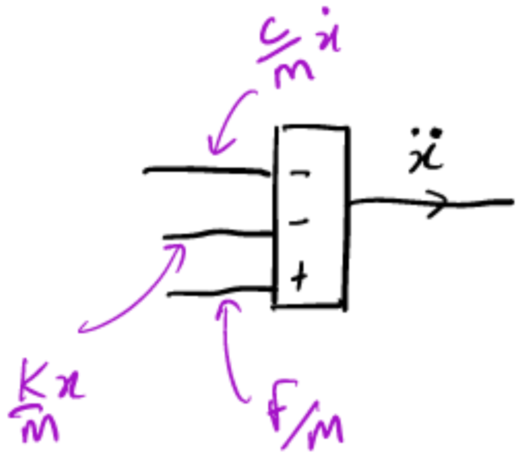
$$\ddot{x} = \frac{1}{m}\left[-c\dot{x} - Kx + f(t)\right]$$

The highest derivative order is usually equal to the sum of other terms. In this equation, $\ddot{x}$ is the highest order term. Our starting point will be at a summation block that outputs the $\ddot{x}$ signal:

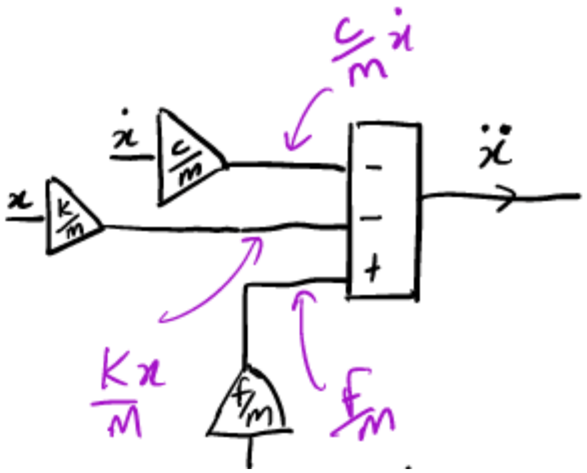


We then populate the inputs to the summation block with the rest of the equation terms:

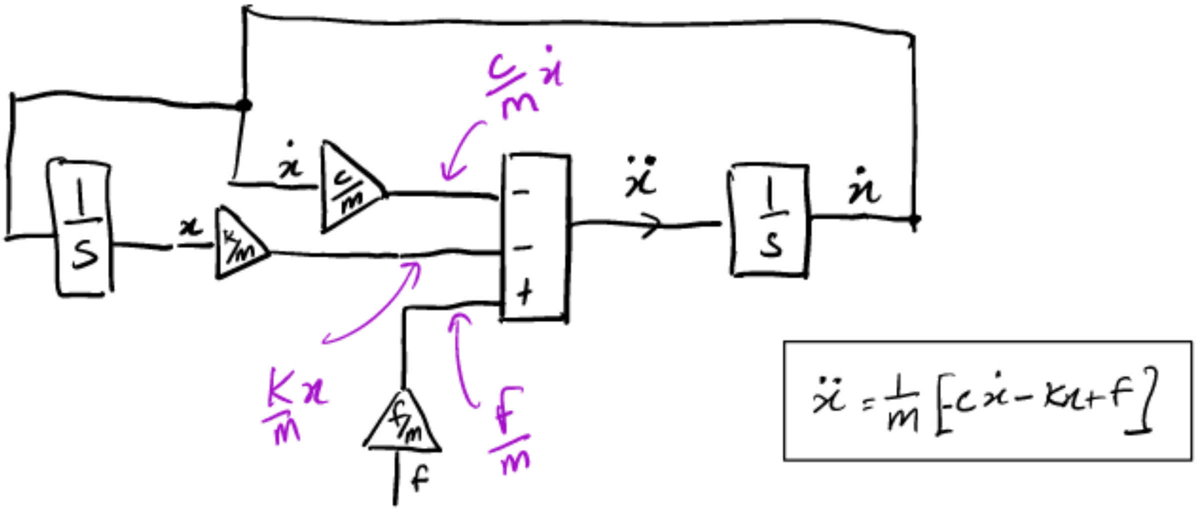
$$\boxed{\ddot{x} = \frac{1}{m}\left[-c\dot{x} - kx + f\right]}$$



Next, we insert out gain blocks:

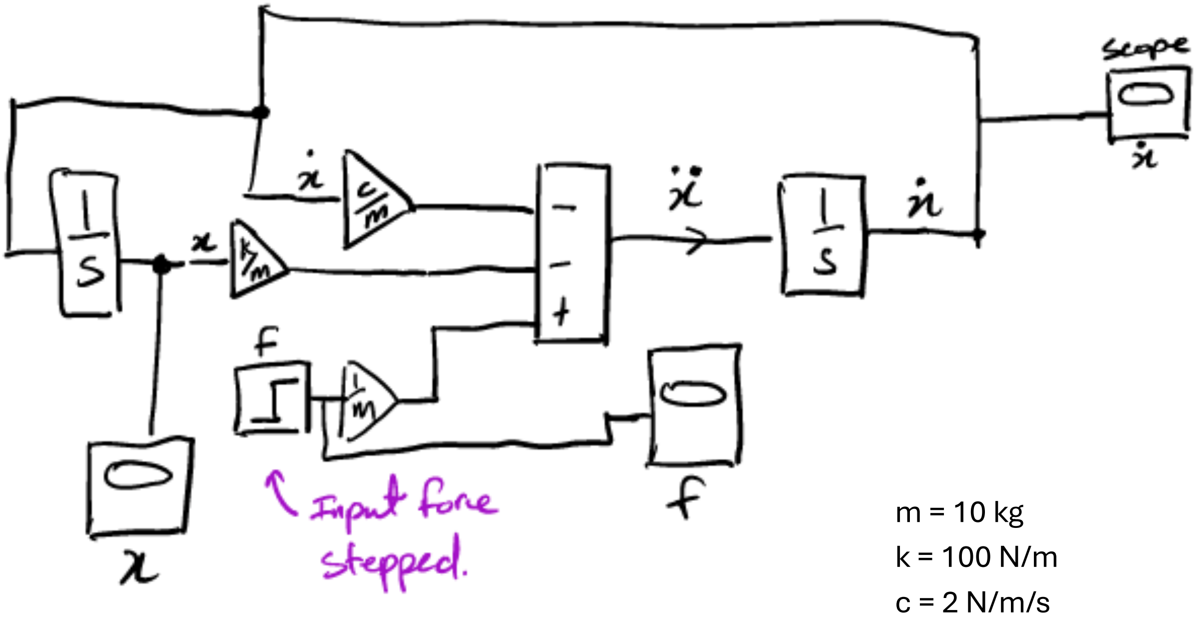


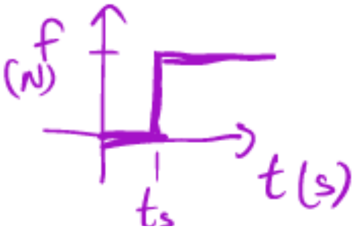
We then place the integral blocks on the $\ddot{x}$ signal to get $\dot{x}$ and x signals:



$$\ddot{x} = \frac{1}{m}\left[-c\dot{x} - kx + f\right]$$

To visualize the inputs and outputs, scopes can be placed at any point in the diagram to plot the signal versus time. For the input signal, many different options are available such as *step, constant, sinusoidal*, or even a *signal generator* that can generate a custom signal. For this example, we will give a step input signal.

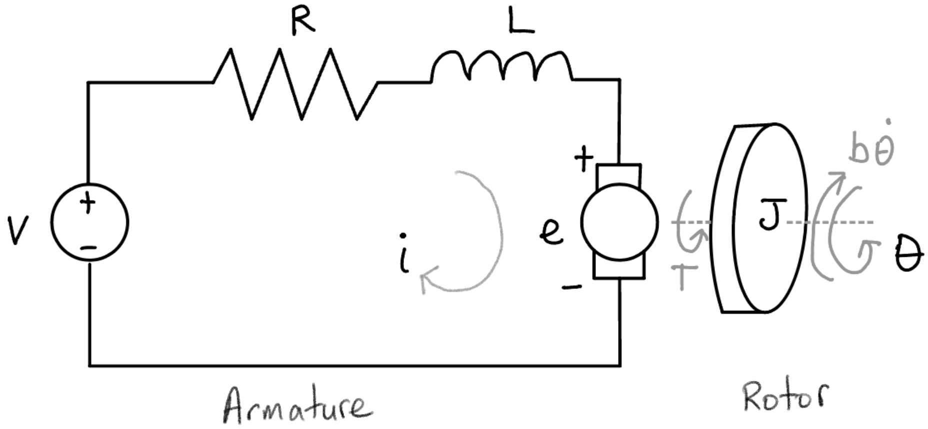




m = 10 kg
k = 100 N/m
c = 2 N/m/s
f =1 N

The numerical values of c, m, and k can be imported directly in the gain blocks. If these values will be required to be changed later, it is advisable to create a script file with these parameters. This will allow a central location to change these parameters which will be easier than searching through the diagram.

Exercise 2 – Input the coupled equations that govern a DC motor



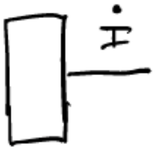
Letting $i \rightarrow I$ and rearranging equations 3 and 4:

$$\ddot{\theta} = \frac{K_t}{J} I - \frac{b}{J}\dot{\theta} - \frac{T_L}{J}$$

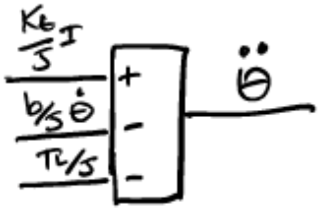
AND

$$\dot{I} = -\frac{R}{L} I + \frac{V}{L} - \frac{K_e}{L}\dot{\theta}$$

Since we have 2 equations, we will start with 2 summation blocks:

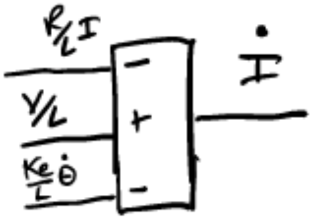


Next, we complete our summation signals:

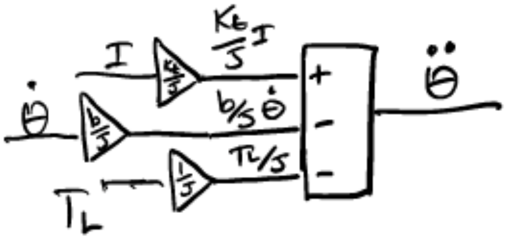


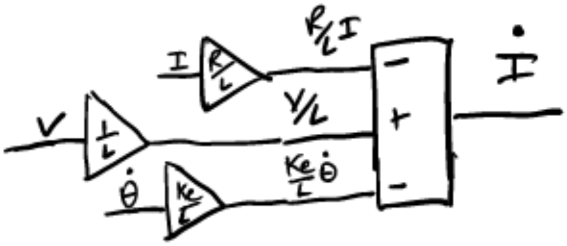
$$\ddot{\theta} = \frac{K_t}{J} I - \frac{b}{J}\dot{\theta} - \frac{T_L}{J}$$

$$\dot{I} = -\frac{R}{L} I + \frac{V}{L} - \frac{K_e}{L}\dot{\theta}$$

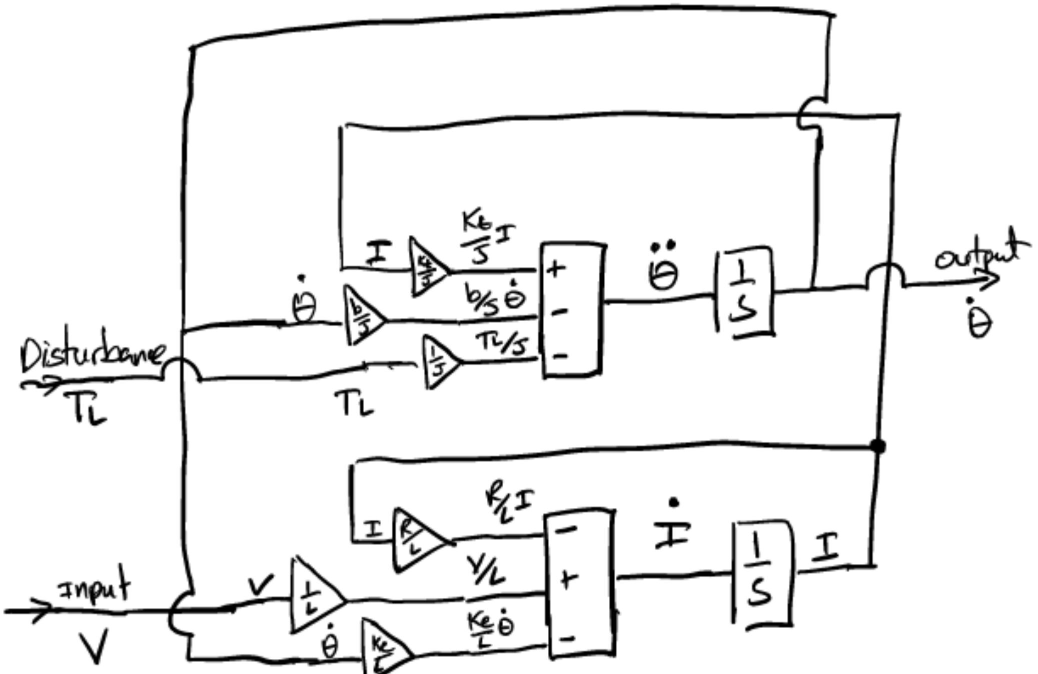


Include the gain blocks:





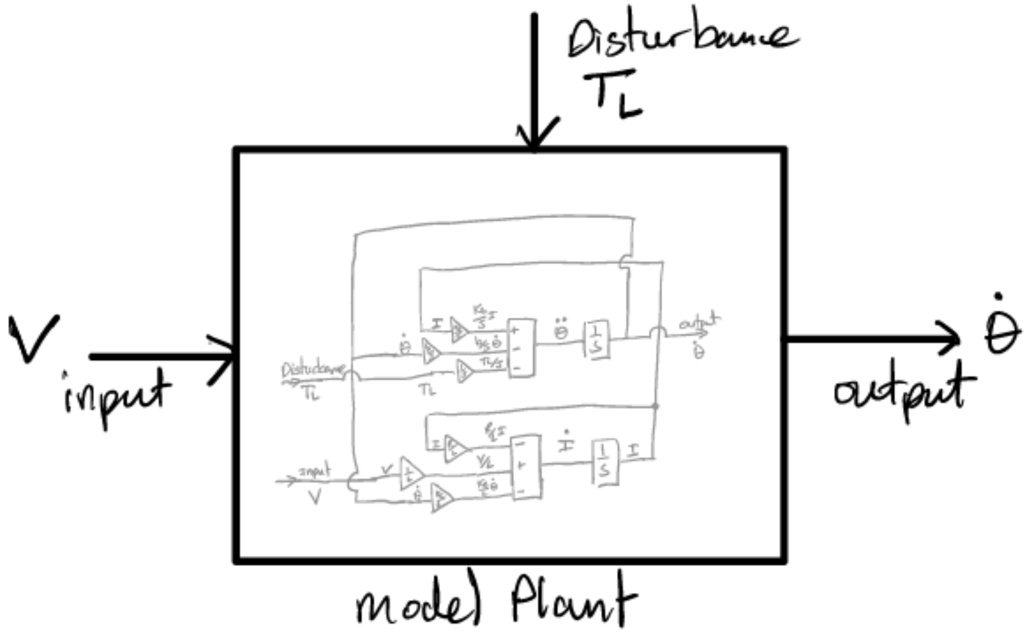
Next, we place our integral blocks to complete the model:



$$\ddot{\theta} = \frac{K_t}{J} I - \frac{b}{J}\dot{\theta} - \frac{T_L}{J}$$

$$\dot{I} = -\frac{R}{L} I + \frac{V}{L} - \frac{Ke}{L}\dot{\theta}$$

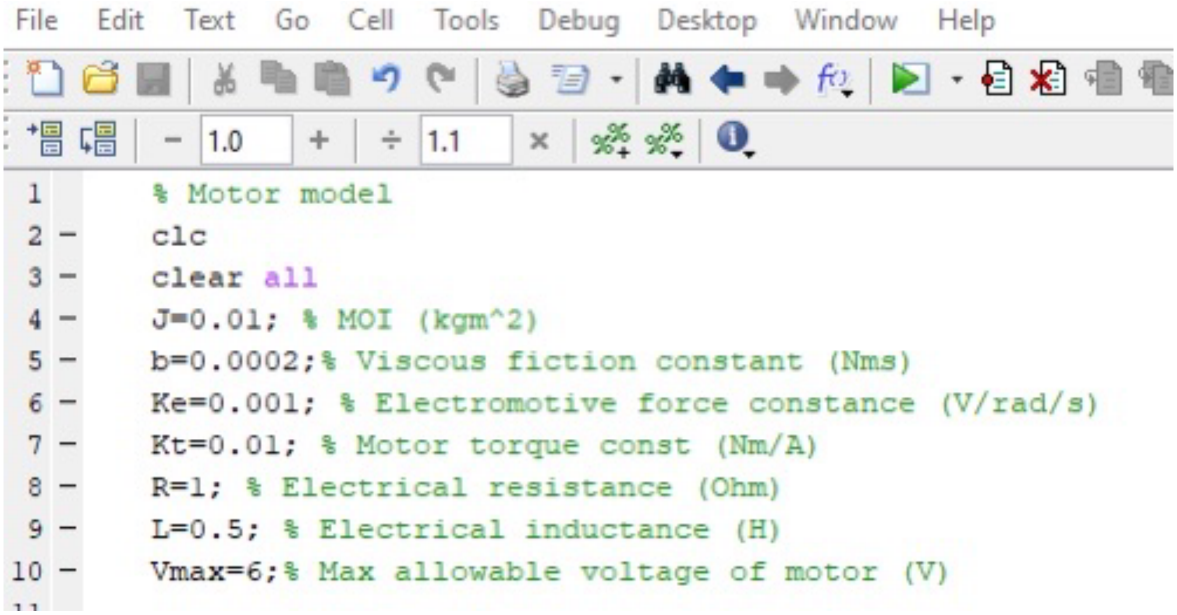
The entire system can be highlighted and placed in a subsystem for system organization:



Exercise 3 - PID Control of a DC Motor

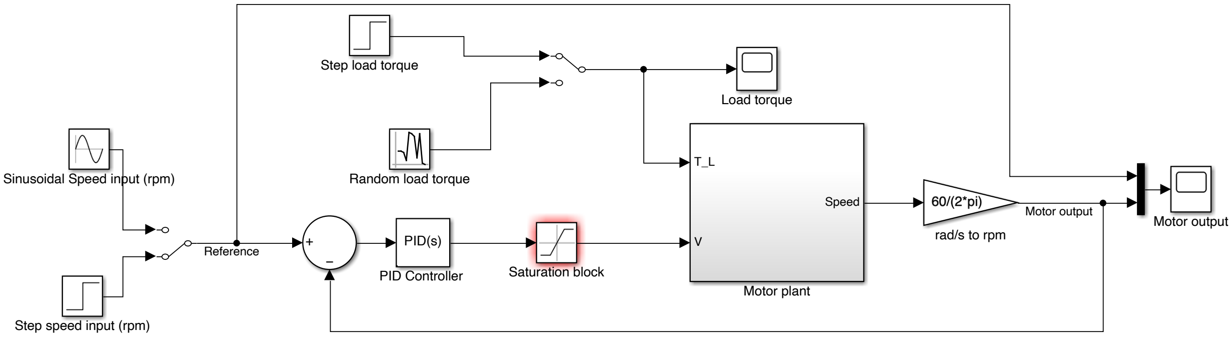
In this exercise, A PID controller will be implemented such that a reference motor speed signal is followed. A disturbance torque will be applied, and the reaction of the controller will be observed.

1. In a script file, enter the following motor parameters:

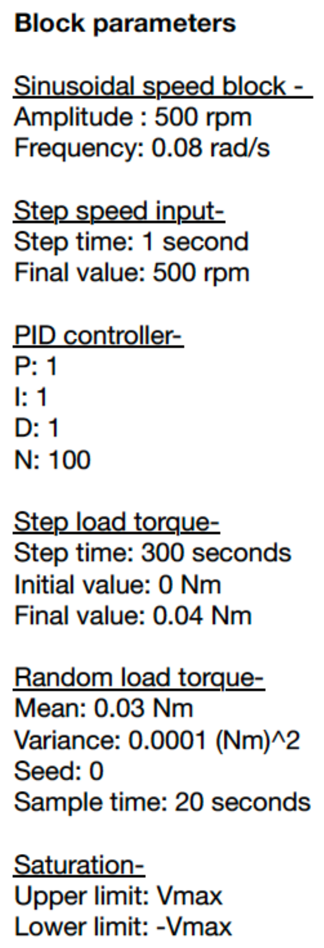


```
% Motor model
clc
clear all
J=0.01; % MOI (kgm^2)
b=0.0002;% Viscous fiction constant (Nms)
Ke=0.001; % Electromotive force constance (V/rad/s)
Kt=0.01; % Motor torque const (Nm/A)
R=1; % Electrical resistance (Ohm)
L=0.5; % Electrical inductance (H)
Vmax=6;% Max allowable voltage of motor (V)
```

2. Using your previously define motor model as the plant, complete the Simulink model as below:



3. Update the block parameters with the following values:



Block parameters

Sinusoidal speed block -
Amplitude : 500 rpm
Frequency: 0.08 rad/s

Step speed input-
Step time: 1 second
Final value: 500 rpm

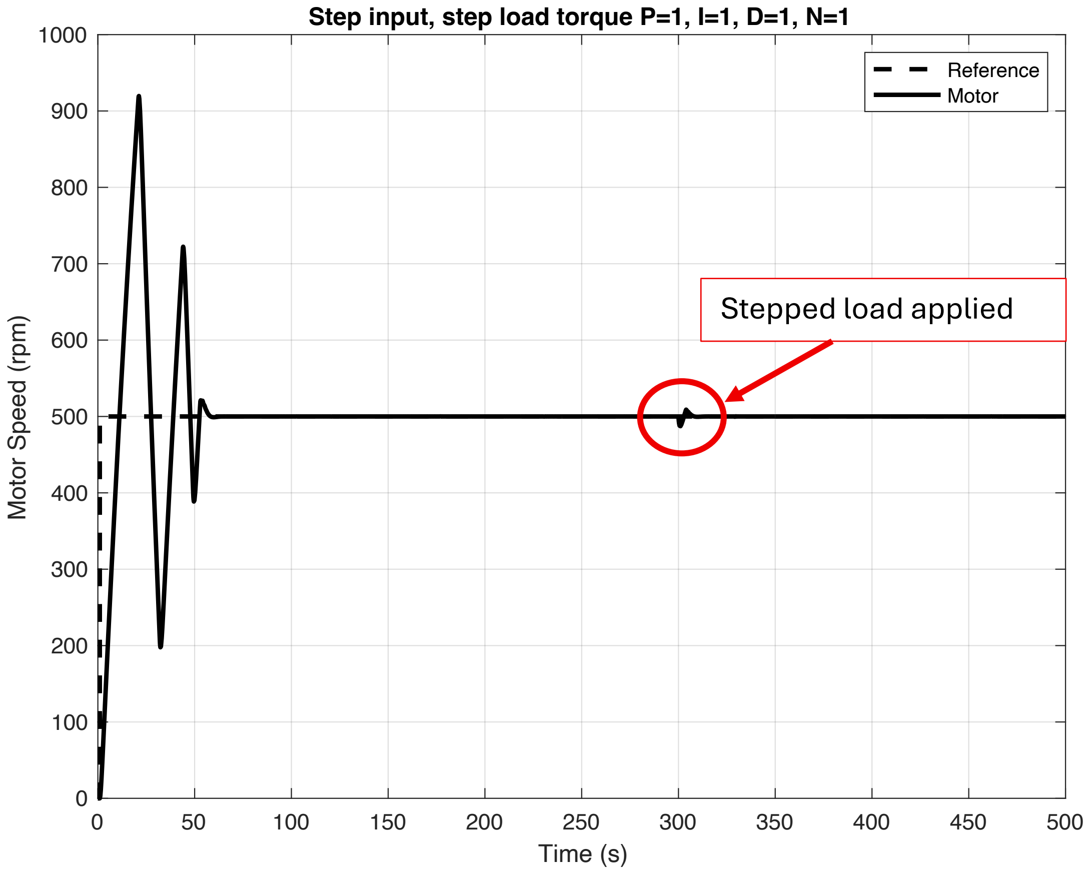
PID controller-
P: 1
I: 1
D: 1
N: 100

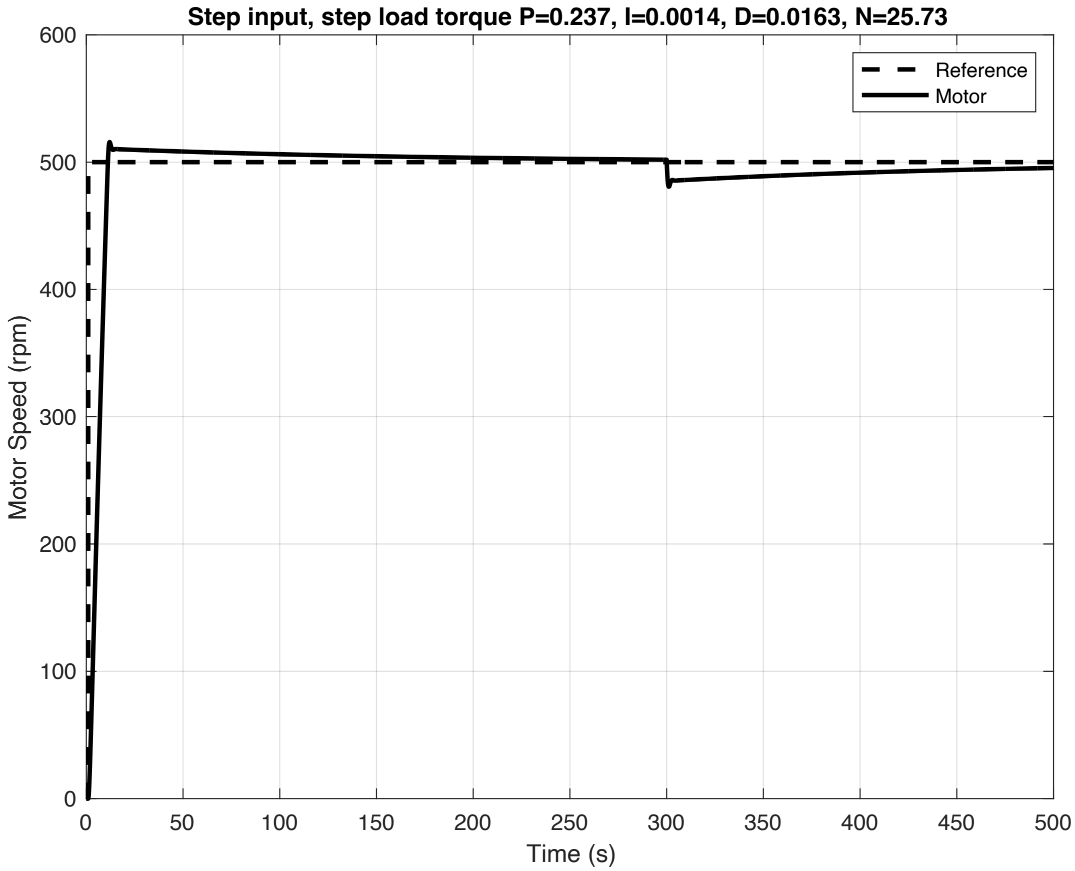
Step load torque-
Step time: 300 seconds
Initial value: 0 Nm
Final value: 0.04 Nm

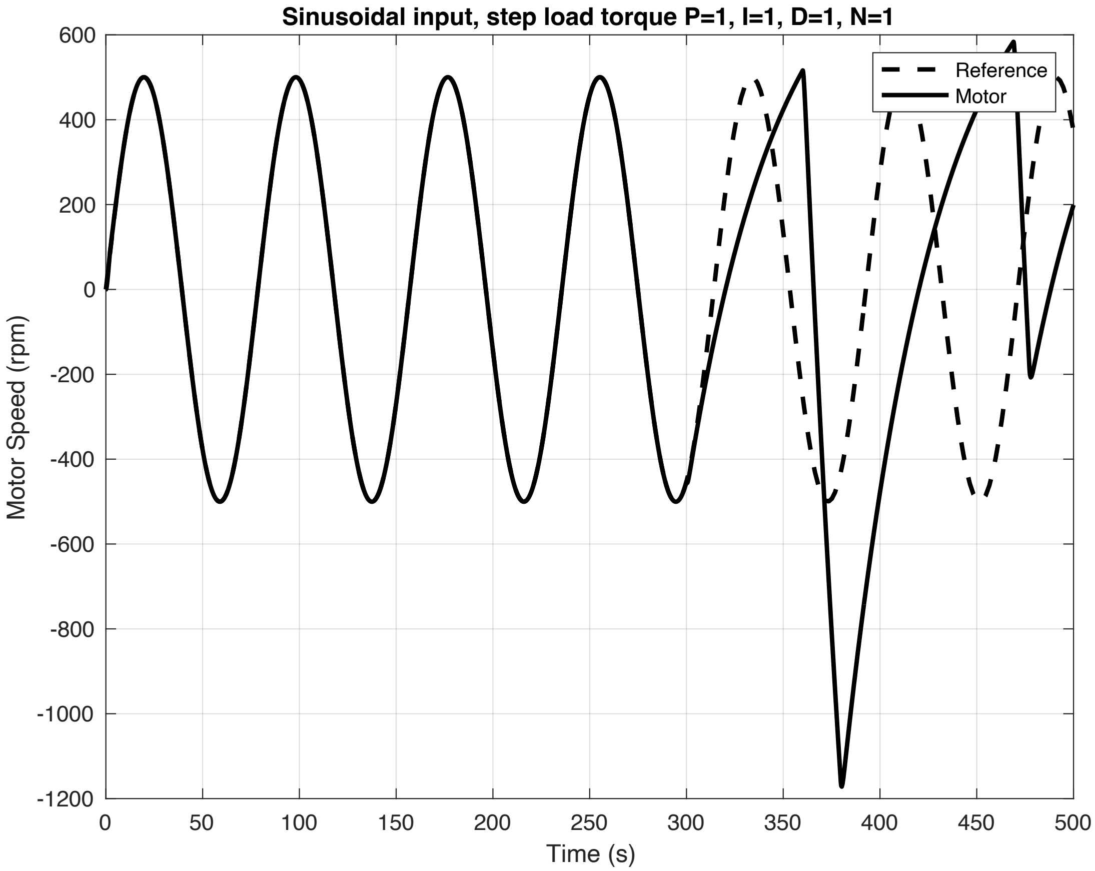
Random load torque-
Mean: 0.03 Nm
Variance: 0.0001 (Nm)^2
Seed: 0
Sample time: 20 seconds

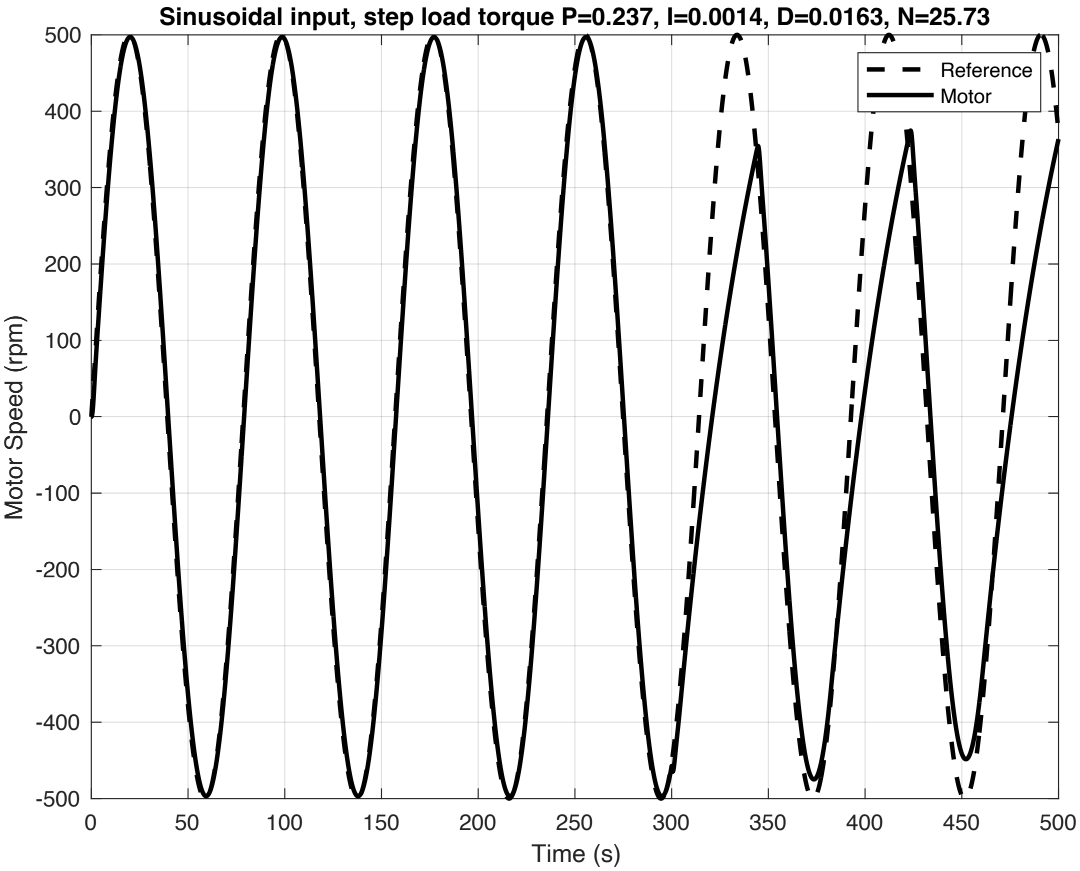
Saturation-
Upper limit: Vmax
Lower limit: -Vmax

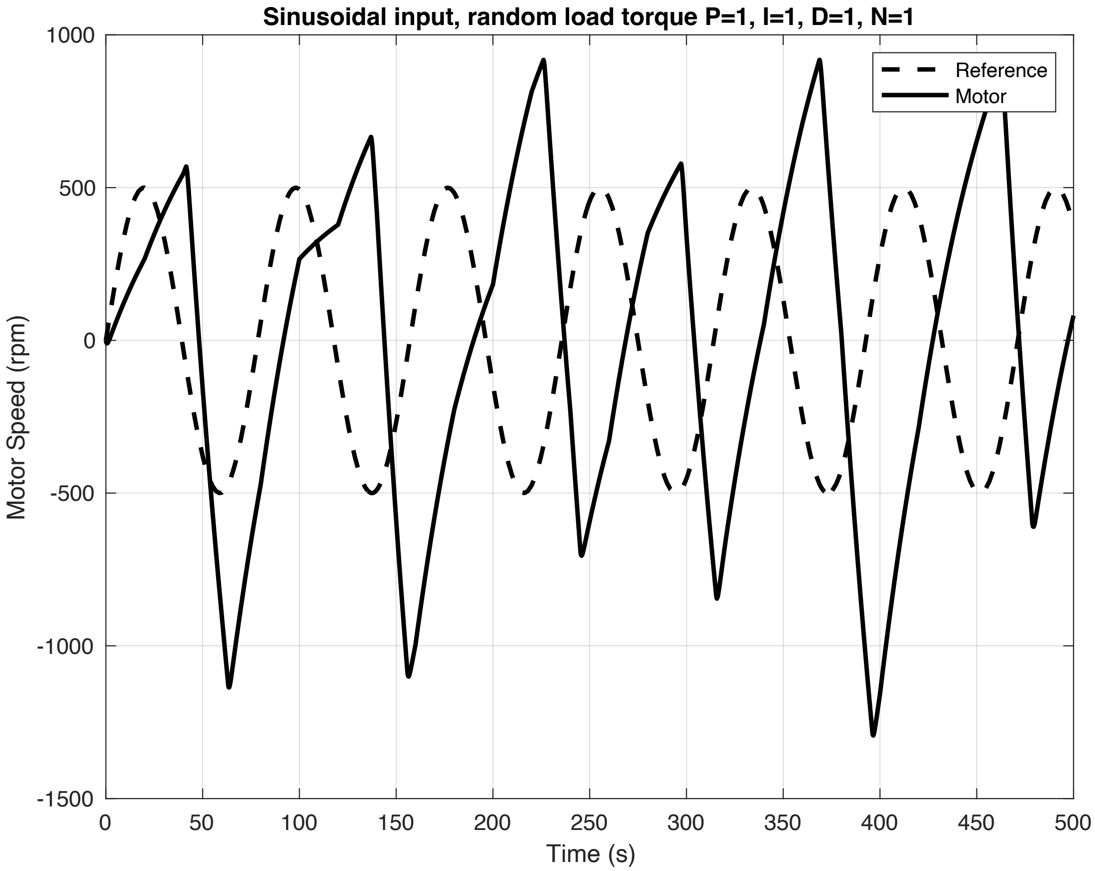
4. Switch between reference signals, load torques and adjust the PID gains and obtain the motor speed vs time plots below.

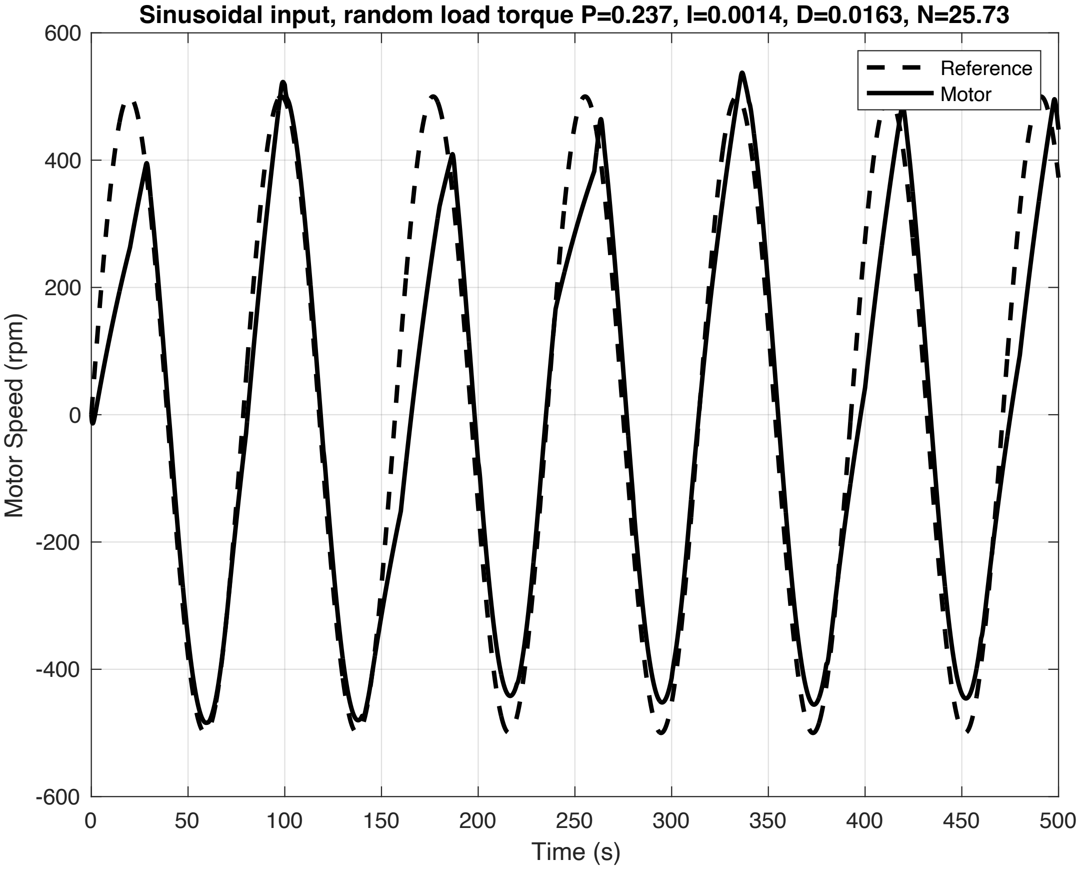

Step input, step load torque P=1, I=1, D=1, N=1
Reference
Motor
Stepped load applied
Motor Speed (rpm)
Time (s)
0
100
200
300
400
500
600
700
800
900
1000
0
50
100
150
200
250
300
350
400
450
500


Step input, step load torque P=0.237, I=0.0014, D=0.0163, N=25.73
Reference
Motor
Motor Speed (rpm)
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
450
500
Time (s)


Sinusoidal input, step load torque P=1, I=1, D=1, N=1
Reference
Motor
Motor Speed (rpm)
600
400
200
0
-200
-400
-600
-800
-1000
-1200
0
50
100
150
200
250
300
350
400
450
500
Time (s)


Sinusoidal input, step load torque P=0.237, I=0.0014, D=0.0163, N=25.73
Reference
Motor
Motor Speed (rpm)
500
400
300
200
100
0
-100
-200
-300
-400
-500
0
50
100
150
200
250
300
350
400
450
500
Time (s)


Sinusoidal input, random load torque P=1, I=1, D=1, N=1
Reference
Motor
Motor Speed (rpm)
1000
500
0
-500
-1000
-1500
0
50
100
150
200
250
300
350
400
450
500
Time (s)

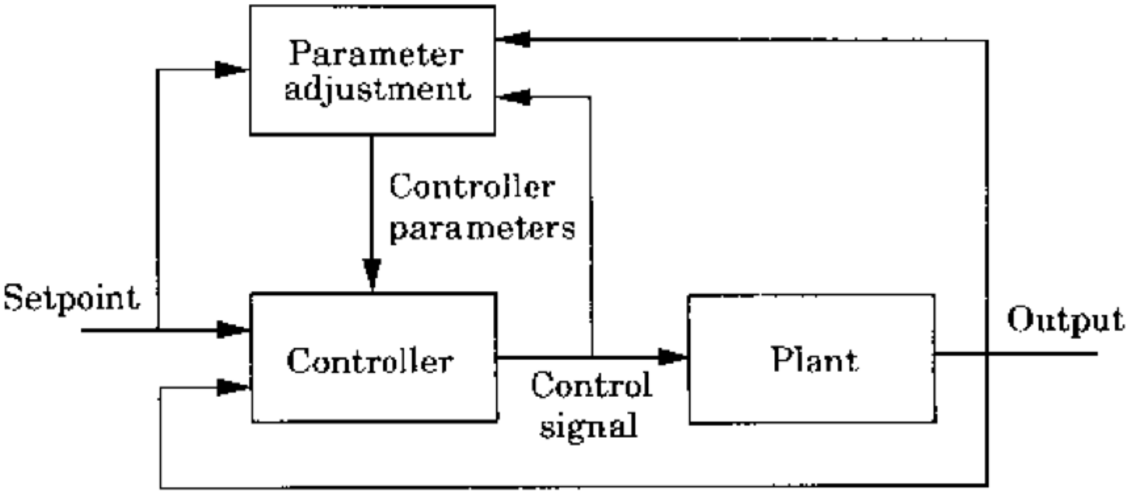


5. For the sinusoidal input, include a scope after the saturation block and **observe** the voltage going into the plant being capped at 6 volts. Remember the maximum allowable voltage to the motor is +/- 6 volts.

6. Reduce the rpm amplitude from 500 to 100 in the sinusoidal input block. After the model is run, look at the voltage after the saturation block. You will observe the voltage is not being cut off (less voltage is required to control at a lower rpm). Look at the motor response, you can observe that it tracks the reference better than when the reference rpm was 500.

Adaptive Control in Simulink

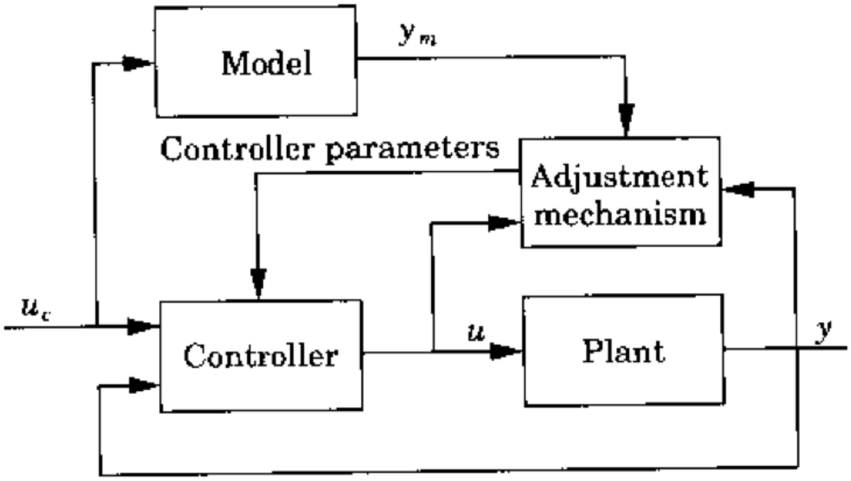
Adaptive means to change or conform to a situation or environment to give the best response to those changes. In control system design, adaptive control refers to the continuous, real-time changing of control parameters. Unlike PID controllers where the K_p, K_i, and K_d values are fixed during operation, adaptive controllers change their control parameters according to a defined control law. Adaptive control has its roots in the 1950s when the design of autopilot systems became difficult for high-performance aircraft undergoing significant dynamical changes during maneuvers. These maneuvers involved constant movements from one operating point to another thus making it difficult for traditional feedback control systems to keep up. In this section, we will discuss and implement one such adaptive control schemes, the Model-Reference Adaptive System (MRAS).



Block diagram of a general adaptive control system [1]

Model-Reference Adaptive Systems (MRAS)

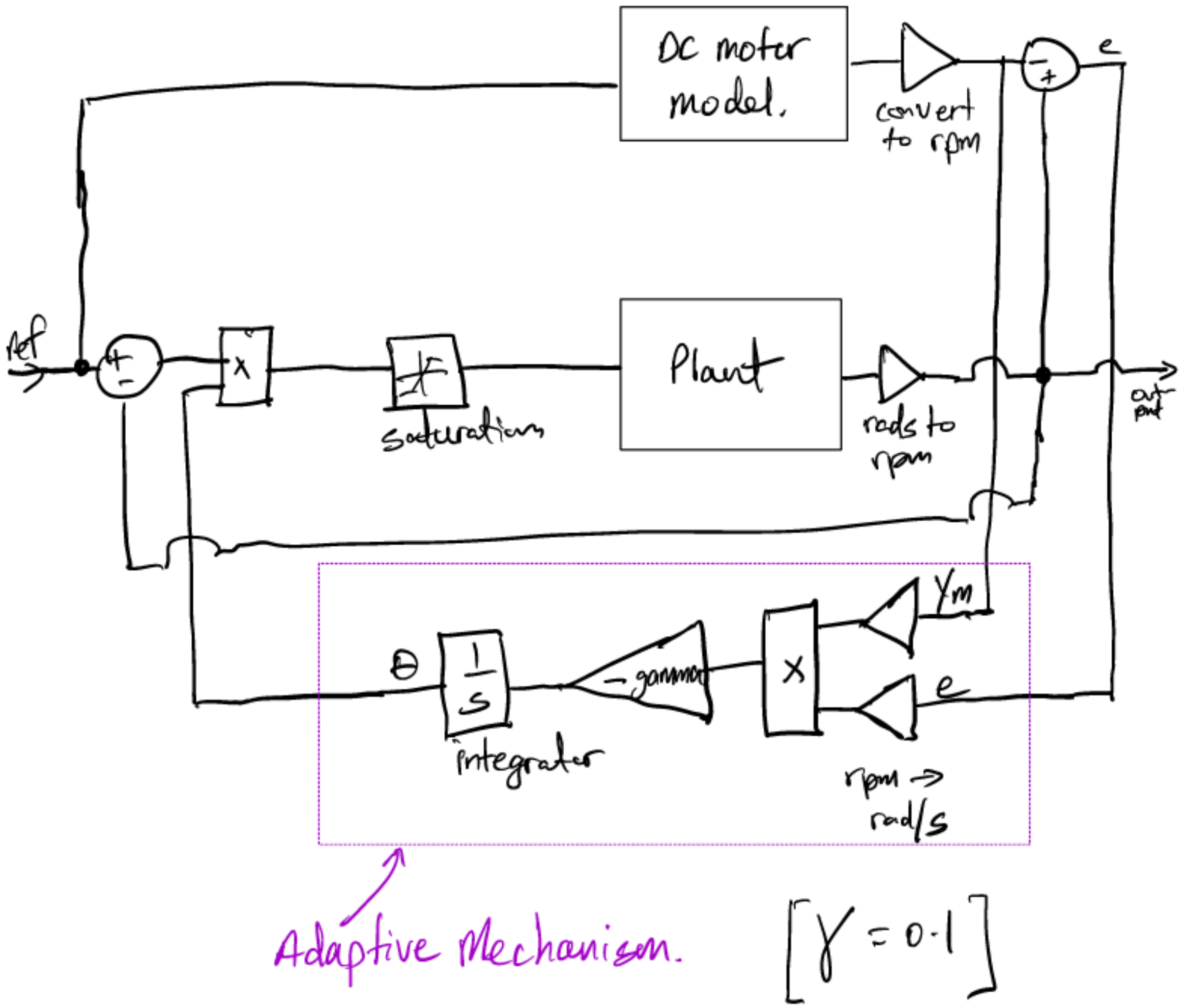
MRAS utilizes a reference model of the system to provide the controller with an indication of how plant should ideally respond. The control law aims to reduce the error between the reference model and the plant model. There are several versions of this control scheme and significant research is conducted in this area. Initially, MRAS was developed for flight control but has found uses in other fields.



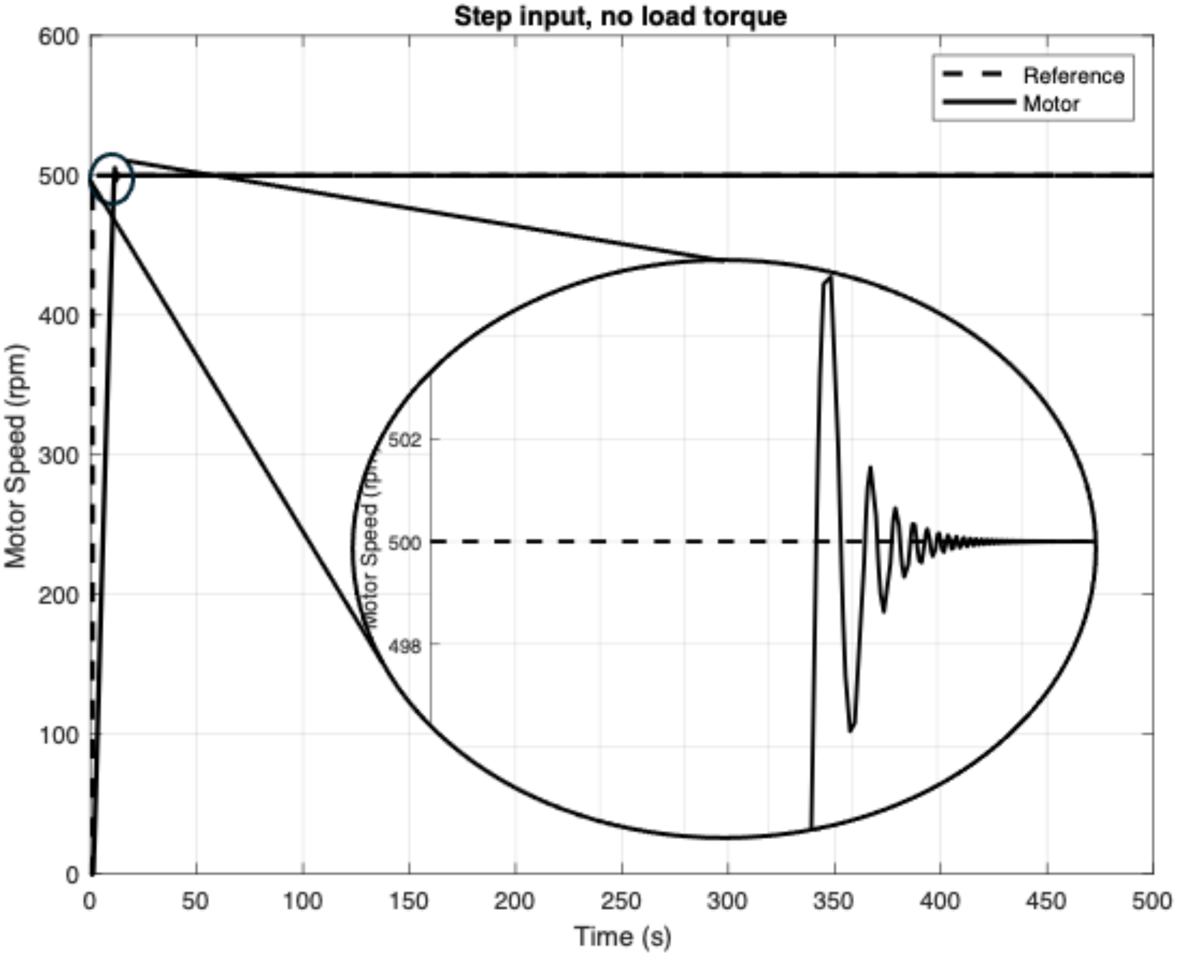
General MRAS control scheme [1]

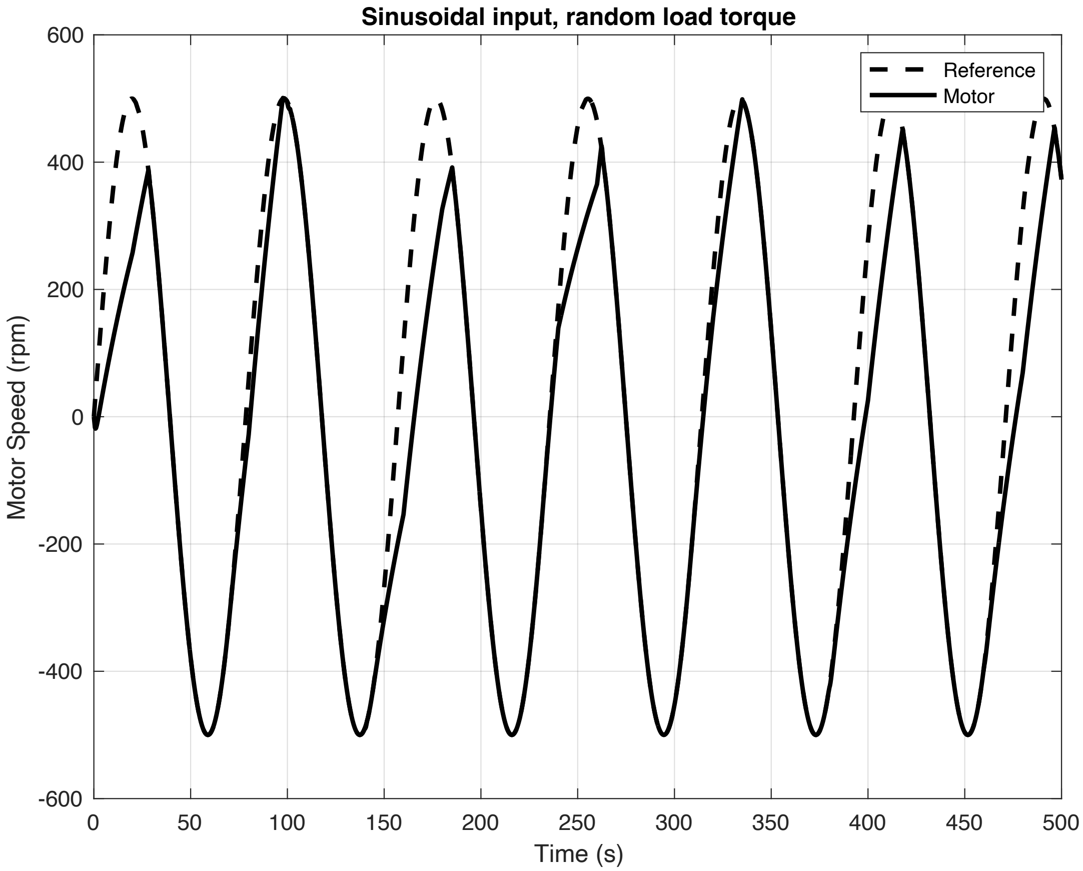
Exercise 4 - Implementing a MRAS in Simulink

On a new Simulink model window, input the adaptive control scheme developed by [2] for the control of a DC motor. You may reuse the Simulink model of the DC motor from the previous exercises as the ‘Plant’ and ‘DC motor model’.



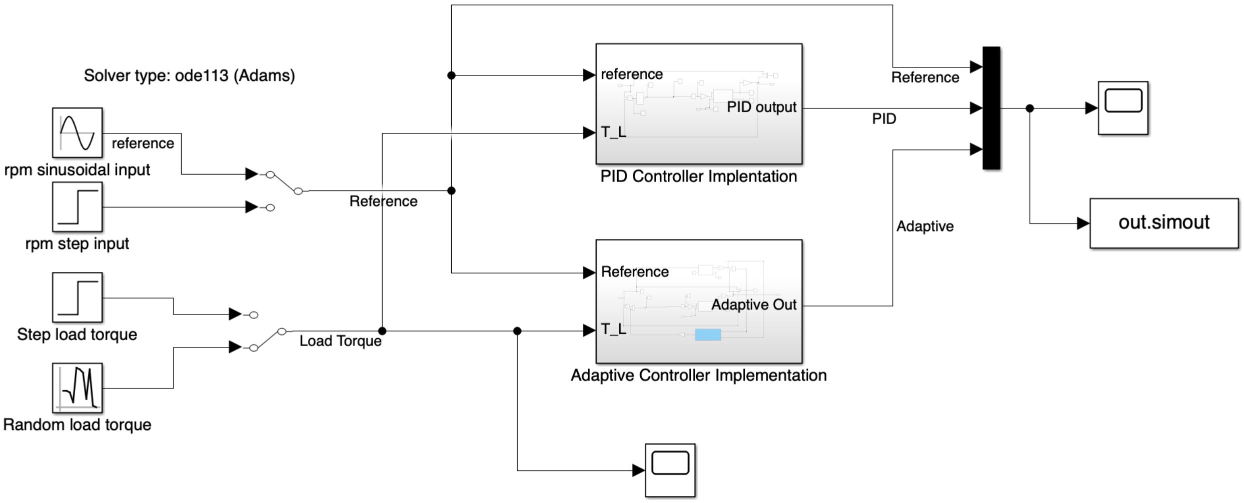
Obtain the plot of reference and motor speed given below. You may have to change your solver type to *ode113 (Adams)* in model settings. Try different values of gamma.


Step input, no load torque
Reference
Motor
Motor Speed (rpm)
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
450
500
Time (s)
Motor Speed (rpm)
502
500
498

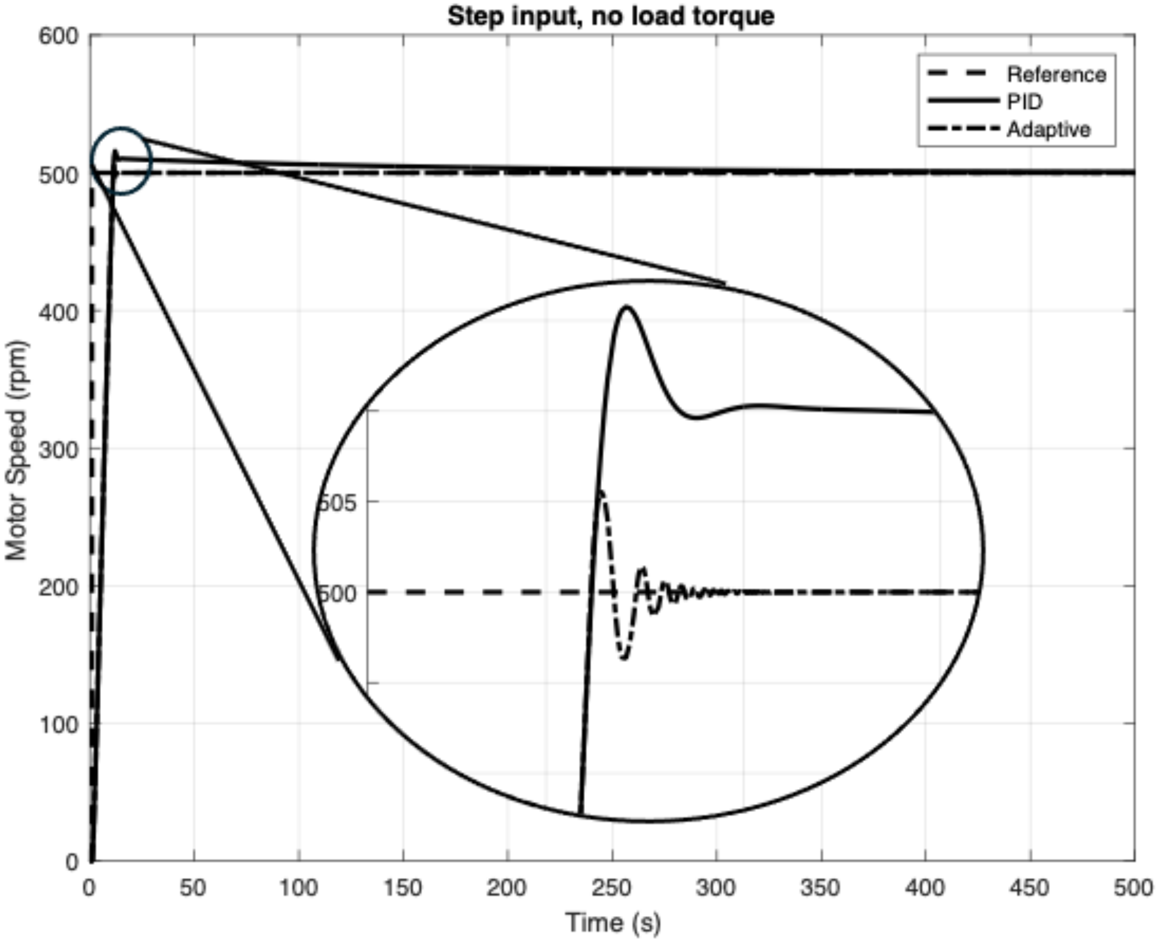


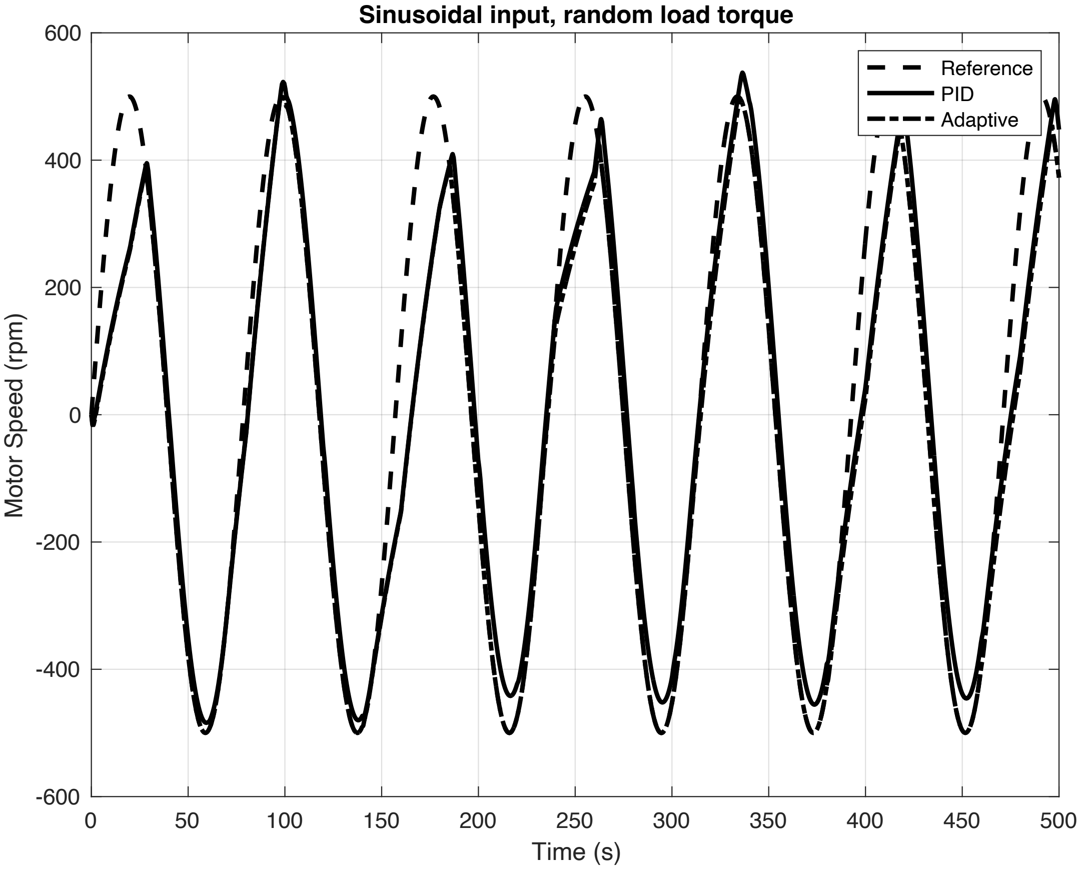
Exercise 5 – Comparing PID and Adaptive controllers for DC motor control

In a new Simulink file, create a model that compares the PID controller (P=0.237, I=0.0014, D=0.0163 and N=25.73) with the MRAS adaptive controller.



Compare the motor speeds (from both controllers) to the reference motor speed.





Conclusion and Additional Tasks

In this section, Simulink was used to model a DC motor. PID and adaptive controllers were experimented with, and the results were compared to each other for various inputs and disturbances. Some additional things to consider:

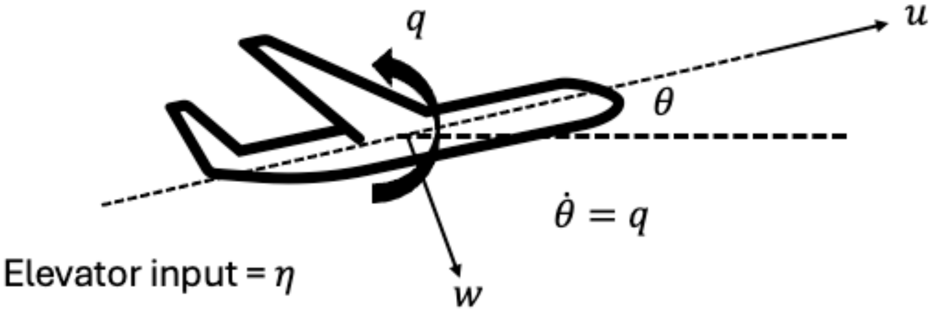
1. The actuation signal was in the form of an analogue voltage signal that was limited to 6 volts. Speed control for a DC motor is usually done with *Pulse Width Modulation (PWM)*. Modify the controller mechanism such that PWM signals are sent to the motor model/plant.
2. Implement a time delay in the feedback loop.
3. The speed sensor data may realistically be noisy. Have the feedback speed data be randomized about a constant value to simulate noise (‘Gaussian noise’ block) and apply a moving average window to filter the signal.

Generate comparison plots between all scenarios.

Assignment

Ling-Temco-Vought A-7A Corsair II longitudinal dynamics

The equations of motion for an aircraft in flight are non-linear and coupled. For small perturbations however, these equations can be decoupled and linearized. The development and linearization of these equations can be found in section 4.2 of [3]. The decoupled equations can be grouped into 2 sub-systems: longitudinal and lateral dynamics. In this assignment, we will simulate the longitudinal dynamics of a Ling-Temco-Vought A-7A Corsair II aircraft, subjected to an elevator input. The equations of motion in a state space format were obtained from [3] and corresponds to level flight at an altitude of 15,000 ft at Mach 0.3. The velocities in the aircraft frame are given as u and w with a pitch rate q.



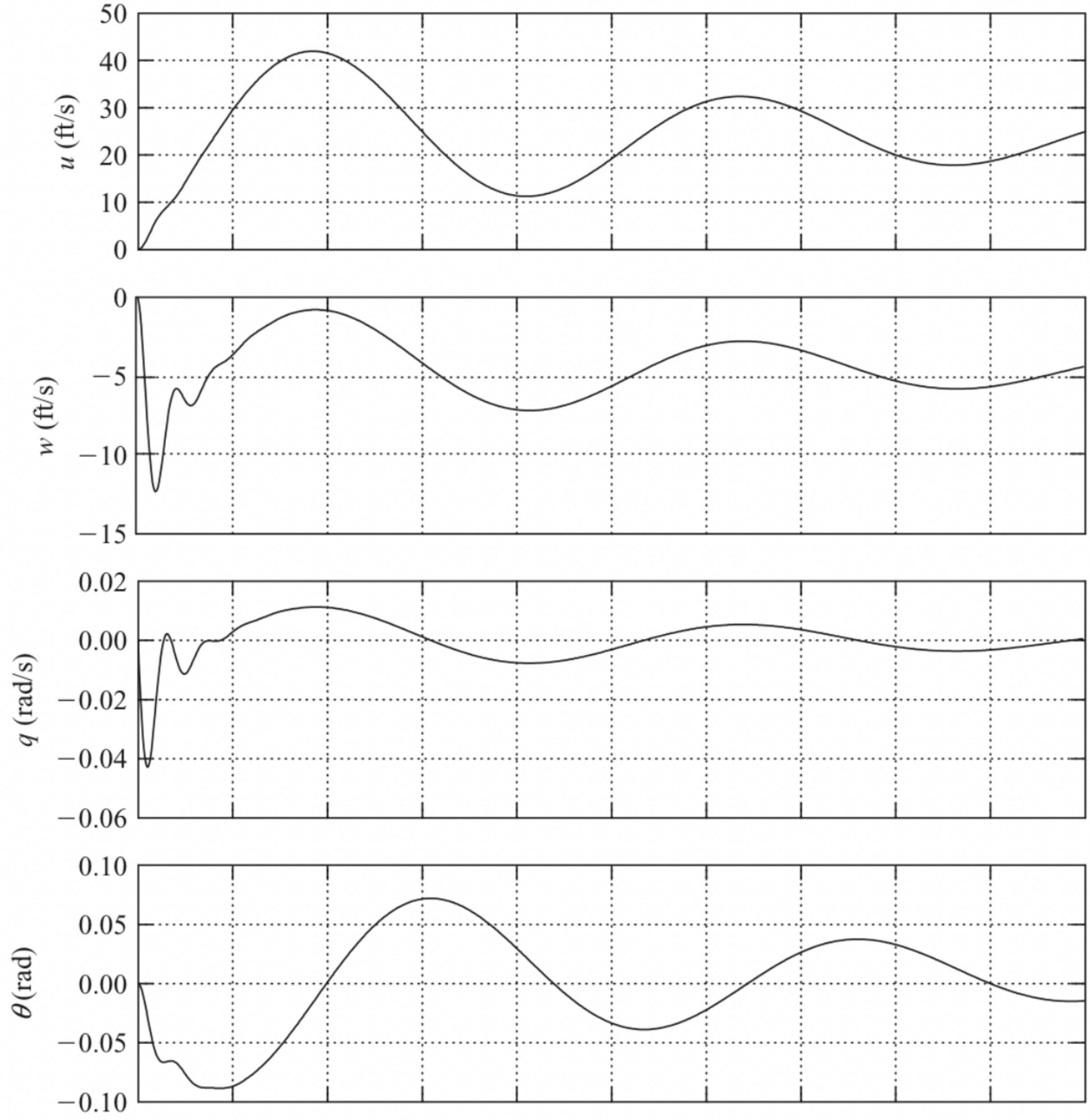
$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0.00501 & 0.00464 & -72.90000 & -31.34000 \\ -0.08570 & -0.54500 & 309.00000 & -7.40000 \\ 0.00185 & -0.00767 & -0.39500 & 0.00132 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} 5.63000 \\ -23.80000 \\ -4.51576 \\ 0 \end{bmatrix} \eta$$

Task 1 – Model Development and Validation

Create and validate a Simulink model of the aircraft system. You may expand the state-space model into individual equations.

- Perform a 1° step in the elevator angle input and obtain the validation plots provided below.
- Justify your modeling choices, including any assumptions made and the rationale for using specific Simulink blocks or subsystems.
- Discuss how well the simulation results match the expected theoretical response and identify possible sources of discrepancy.

- Reflect on how constructing the model deepened your understanding of the aircraft's longitudinal dynamics.

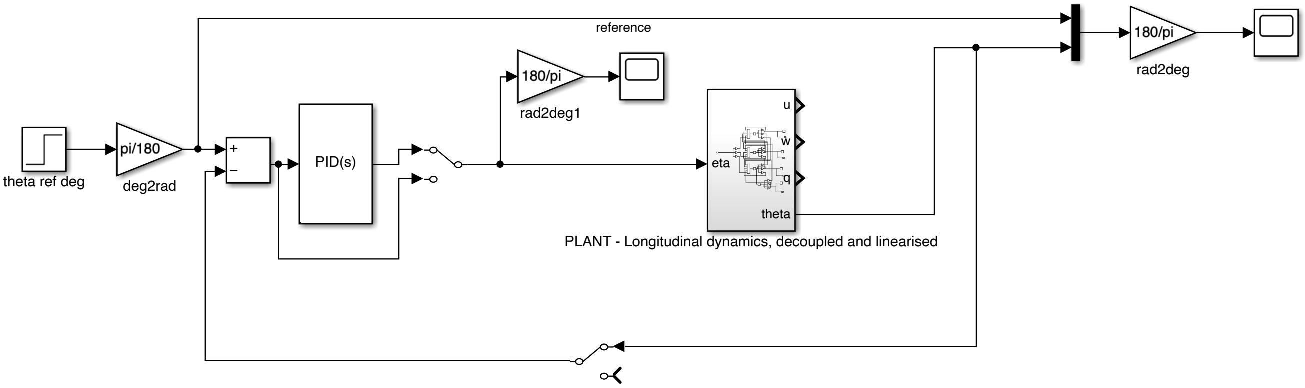


Task 2 – Controller Implementation and Evaluation

Use the validated aircraft model as the plant for both PID and MRAS adaptive control systems.

- Implement the PID control system using the following gains: P = –122.218, I = –333.05989, D = –10.5337, N = 133.1767.

- Implement a MRAS adaptive control scheme following the structure discussed in the module.
- Compare both controllers under a 1° step in the reference elevator angle input.
- Plot and analyze the reference vs. actual pitch angle responses on a single graph.
- Discuss controller performance using control metrics (rise time, overshoot, steady-state error, settling time).
- Critically evaluate which controller performs better and explain why, considering both theoretical and practical implications.



Implemented PID controller

Task 3 – Equation Manipulation and Disturbance Analysis

- Modify the model to include realistic disturbance terms representing physical phenomena (e.g., gusts, sensor noise, actuator delay).
- Clearly state and justify the disturbances introduced.
- Analyze how these affect system performance and controller behavior.
- Explain the physical significance of each disturbance and its connection to real-world aircraft dynamics.
- Reflect on how incorporating disturbances changes your understanding of control robustness and model realism.

Task 4 - Reflection and Synthesis

Prepare a technical report that documents:

- The model development process and key assumptions.
- Controller design, tuning strategy, and comparison results.

- Analysis of disturbances and their physical interpretation.
- A **reflection section** connecting what was learned through simulation to the theoretical concepts covered in control systems design and modeling.
- Include discussion on how these computational experiments inform real-world aircraft control system development.

Reports should be clear, well-structured, and supported with labeled plots, legends, and concise interpretations.

Submission Requirements

Submit a zipped folder containing:

- All Simulink and MATLAB script files.
- A PDF report following the structure outlined above.

References

[1] Åström, K. J., Wittenmark, B. (2008). Adaptive Control. United States: Dover Publications.

[2] Chirag (2025). Simple Adaptive Control Example (https://www.mathworks.com/matlabcentral/fileexchange/44416-simple-adaptive-control-example), MATLAB Central File Exchange. Retrieved September 30, 2025.

[3] Cook, M. V. (2012). Flight Dynamics Principles: A Linear Systems Approach to Aircraft Stability and Control. Germany: Elsevier Science.