Coventry University

Faculty of Engineering, Environment and Computing

7146CEM: Automotive Software Engineering

Course Work 1: Design and Development PID Controller design and its usage in Cruise Control and Motor Speed Plant Models

By

Student Name:	Adish V
Student ID:	11910156
Course:	Automotive Software Engineering

Dec 2021

Contents

Table of Figures

List of Tables

Abbreviations and Definitions:

PID	Proportional Integral Differential
Kp	Proportional Gain
Ki	Integral Gain
Kd	Differential Gain
Ts	Sampling Time
R	Resistance
L	Inductance
J	Inertia
K	Constant of Proportionality
b	Damping coefficient
m	mass
u	Force
v	velocity

Rise Time - Time taken for a signal to reach 90% from 10% of its final value.

Settling Time - Time taken for a signal to reach its saturation value.

Steady state error - Difference in output signal to input signal once Settling time is reached.

Overshoot - Deviation of output signal above the input signal.

ABSTRACT

A proportional-integral-derivative controller is a type of control loop mechanism that uses feedback to continuously control a process. This type of controller is commonly used in industrial control systems.

The first practical application of the concept was in the design and development of automatic steering systems. It was then widely used in the manufacturing industry.

The concept of PID is a responsive and accurate correction algorithm that works seamlessly with most control functions. For example, if a car's cruise control suddenly slows down, the controller's algorithm will automatically reverse the speed and increase it in a controlled manner.

PART A – SOFTWARE DEVELOPMENT LIFE CYCLE

INTRODUCTION

In this report, we will define some of the software development life cycle techniques used to develop PID controller using V-Model. Also, referencing the design and tested PID controller in two of the plant models, Cruise Control and Motor Speed Control and will also be tuning the gain for each of the plant models to get desired response meeting the given requirements. Further, for code generation MATLAB auto code generation feature will be used and all the design will be under version control for easy project management.
Software Development Life Cycle(section)

V-MODEL

V-Model model is the standard used across Automotive Industry. V-Model is the proven model for most of the applications and projects in Automotive Industry.
Refer Figure 1: V-Model

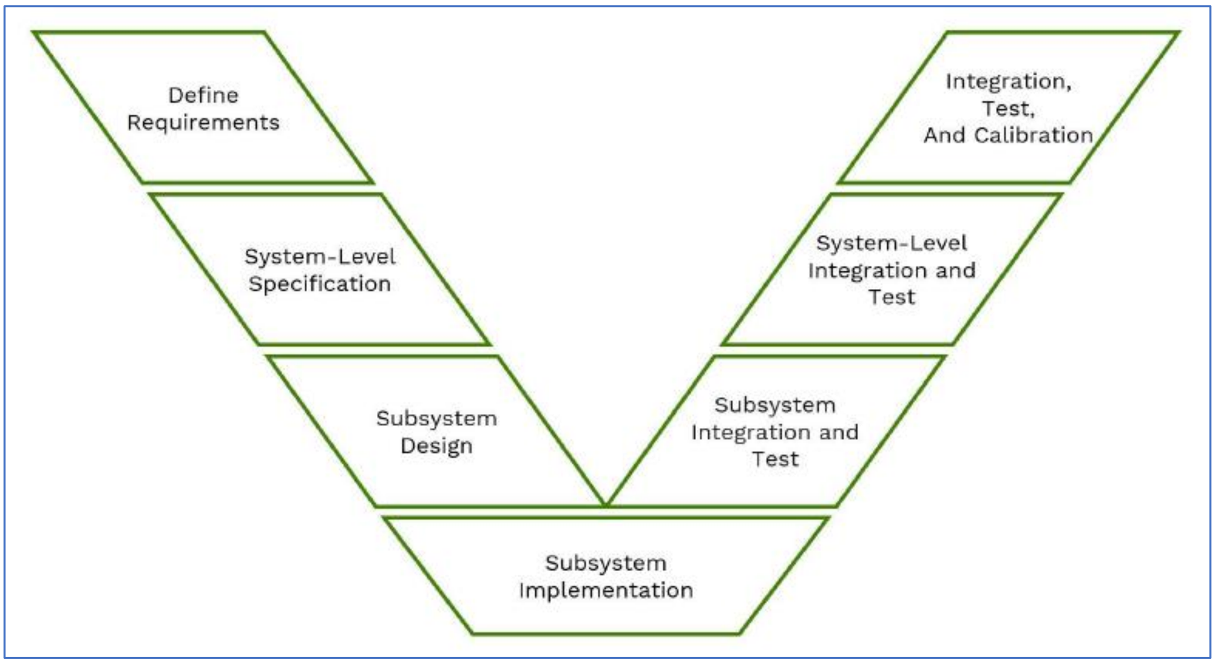

Define Requirements
System-Level Specification
Subsystem Design
Subsystem Implementation
Subsystem Integration and Test
System-Level Integration and Test
Integration, Test, And Calibration

Figure 1: V-Model

Since all the requirements for the design were clearly defined before, hence the use of V-Model for current use case.

PART B – PID CONTROLLER

BRIEF ON PID CONTROLLER

The proportional gain increases the control signal for the same error level. It tends to cause the system to react more quickly than it should, but also overshoot more.

The addition of a derivatives term to the controller allows the user to predict an error. This allows the controller to control the system's response if the error gets larger.

The addition of an integral term helps reduce steady-state error. It increases the control signal and helps in driving the error down. However, it can also make the system more sluggish.

The general effects of various controllers on a closed-loop system are presented in the table below.

FUNCTIONAL REQUIREMENTS OF PID CONTROLLER

In the first phase of V-Model below mentioned are the requirements for PID controller,

- Design PID controller using the equations mentioned below.

$y(k) = y_p(k) + y_i(k) + y_d(k)$

where,

$y_p(k) = K_p e(k)$

$y_i(k) = y_i(k-1) + K_i T_s e(k)$

$y_d(k) = \frac{K_d}{T_s}[e(k) - e(k-1)]$

T_s = 0.01

- Generate C/C++ Code for the PID controller.

TRANSFER FUNCTION OF PID CONTROLLER

$$u(t) = K_p e(k) + K_i \int e(t)dt + K_d \frac{d(e)}{dt}$$

$$P(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

where,

K_p= Proportional Gain

K_i = Integral Gain

K_d = Derivative Gain.

SUBSYSTEM DESIGN

Design phases include usage of MATLAB Simulink for designing the PID controller. Using the equations mentioned in the requirements section below is the Simulink model. Refer Figure 2.

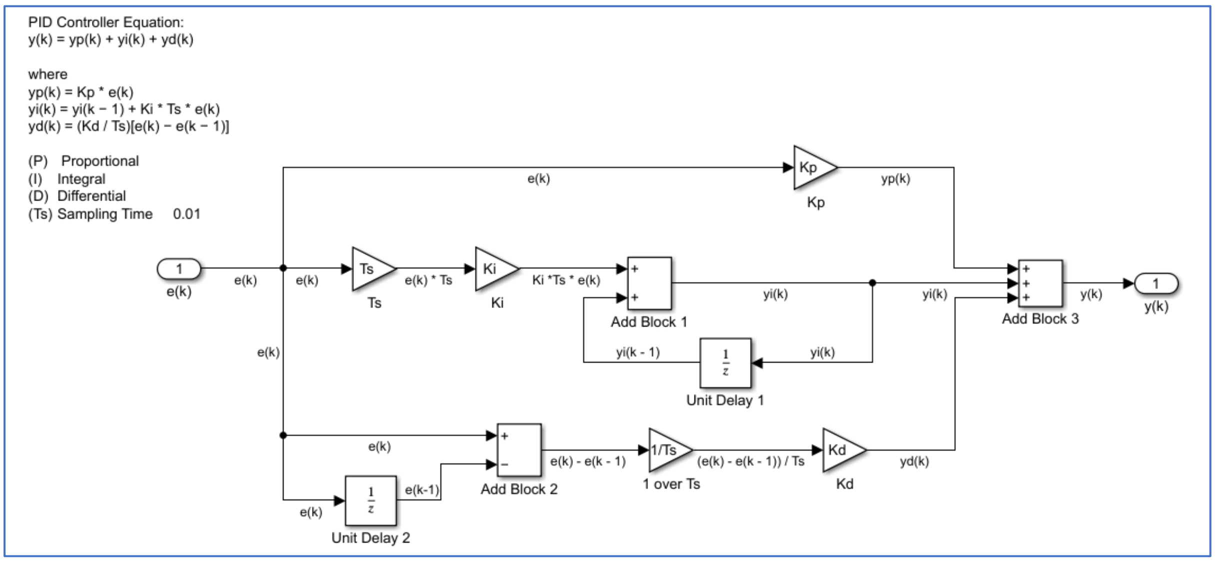


Figure 2: PID controller

TESTING

PID controller testing was performed to ensure the current behaviour. Testing is accomplished by using wrapping PID controller inside a Test-Harness, which is a feature available in MATLAB, and provide it with test signals to produce response and verifying the same. Refer Figure 3.

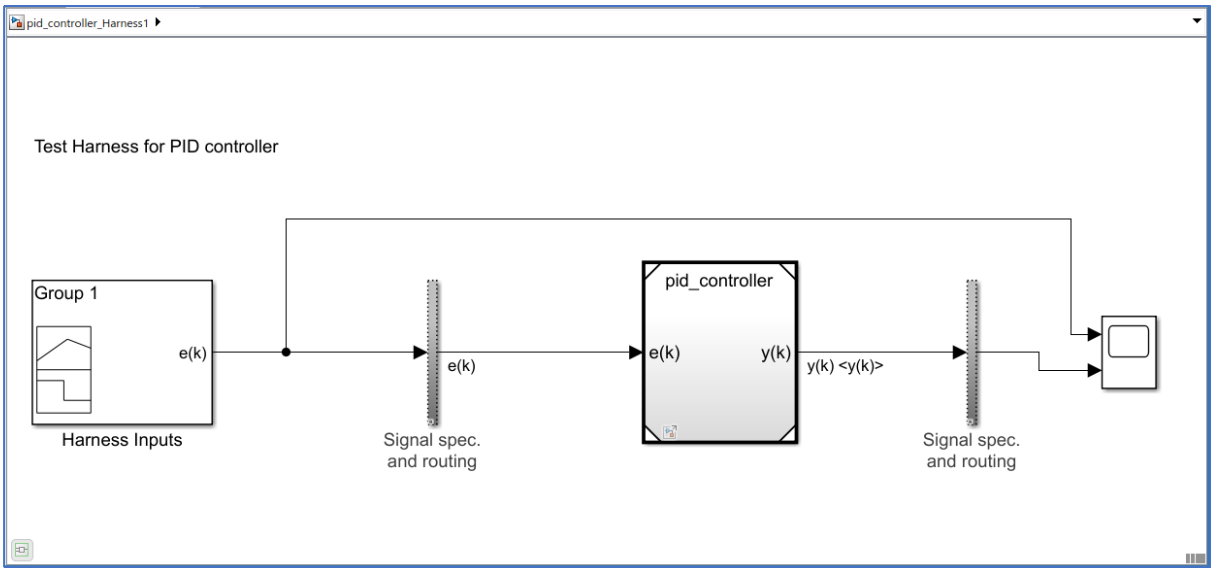


Figure 3: Test harness

For a given step input the PID controller outputs a ramp signal output. This ensures the proper functioning of the PID controller designed at this stage.

PID controller is designed can be reused in the plant models, things to ensure is that to tune the gain values for its respective plant model to achieve desired response.

One such methods to ensure proper settings for gain values and sampling time is the use of call-back in MATLAB.

PART C – CRUISE CONTROL

BRIEF ON CRUISE CONTROL

A cruise control system is an automatic vehicle control system that uses a set speed and a button to maintain a steady speed. The system uses a combination of electronic and mechanical components to control the vehicle's speed. The driver must manually set the cruise control to the desired speed.

FUNCTIONAL REQUIREMENTS OF CRUISE CONTROL

Following are the requirements for Cruise Control plant model,

- Design Cruise Control plant model
- Reuse the PID controller model designed in PART B
- Tune Kp, Ki, Kd gains for the PID controller of Cruise Control as follows:
    a. Rise time < 10s
    b. Overshoot < 10%
    c. Steady-state error < 1%

TRANSFER FUNCTION OF CRUISE CONTROL

$$m\dot{v} = u - bv$$

$$P(s) = \frac{1}{[ms + b]}$$

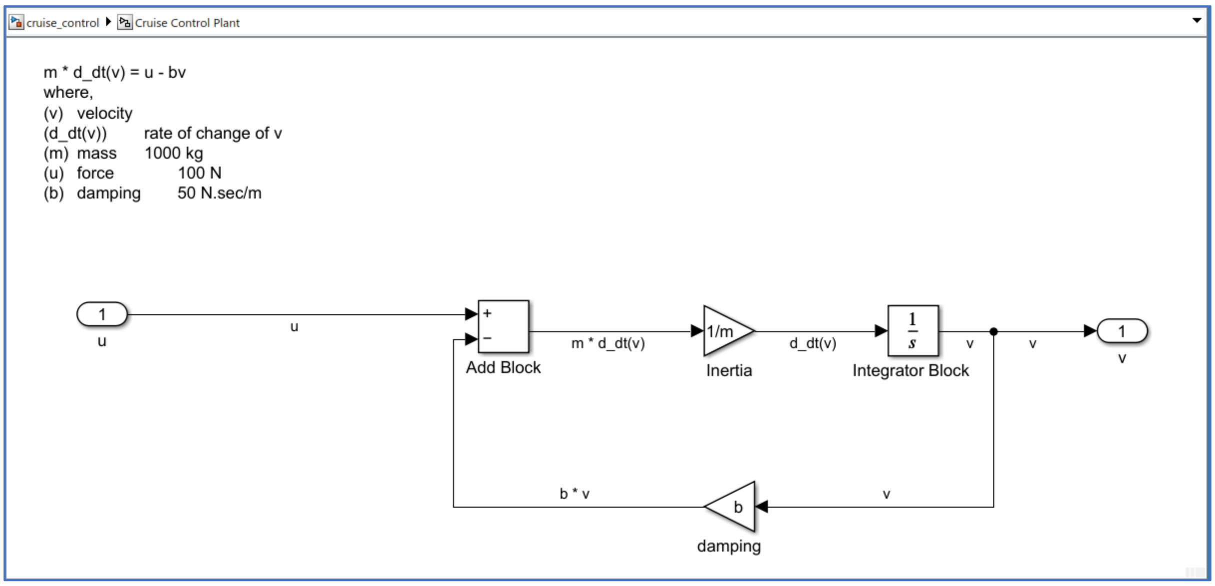
where,

$m = 1000$

$b = 50$

SUBSYTEM DESIGN OF CRUISE CONTROL

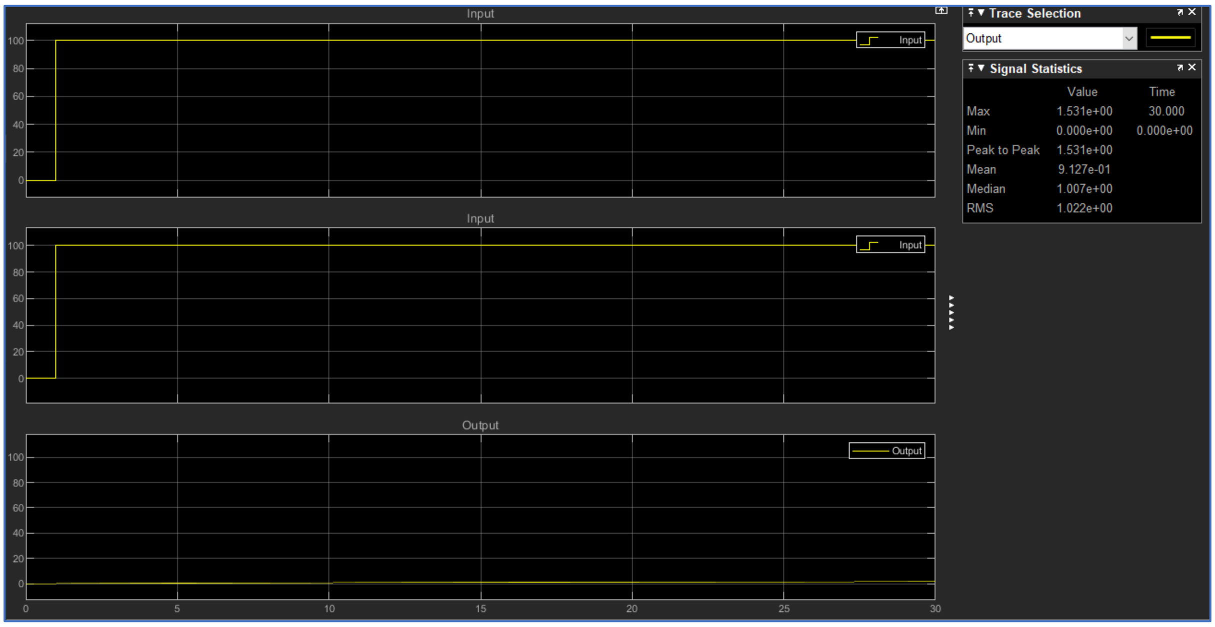
Using the equations defined in Section here above, Cruise Control plant model is designed in the Simulink. Refer Figure 4.



```
m * d_dt(v) = u - bv
where,
(v)   velocity
(d_dt(v))       rate of change of v
(m)  mass       1000 kg
(u)   force           100 N
(b)   damping       50 N.sec/m
```

Figure 4: Cruise Control Plant Model

OPEN LOOP RESPONSE OF CRUISE CONTROL



Trace Selection

Output

Signal Statistics

	Value	Time
Max	1.531e+00	30.000
Min	0.000e+00	0.000e+00
Peak to Peak	1.531e+00	
Mean	9.127e-01	
Median	1.007e+00	
RMS	1.022e+00	

Figure 5: Openloop response Cruise COntrol

Open loop response for Cruise Control shows the model designed is not able to reach the desired output for the given step input. Refer Figure

FEEDBACK LOOP MODEL OF CRUISE CONTROL

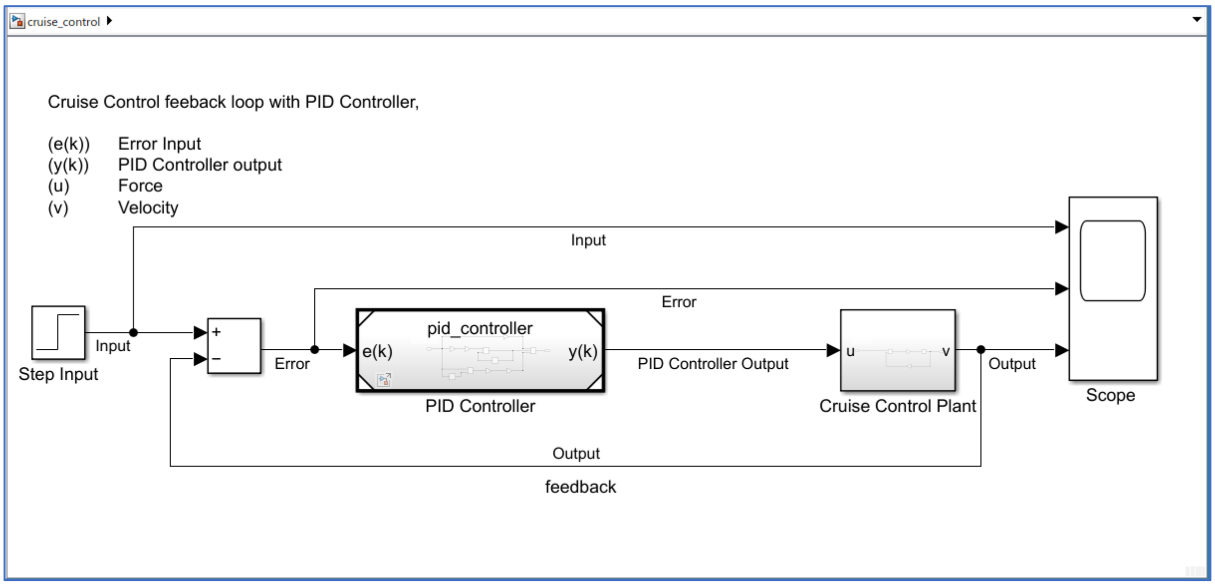
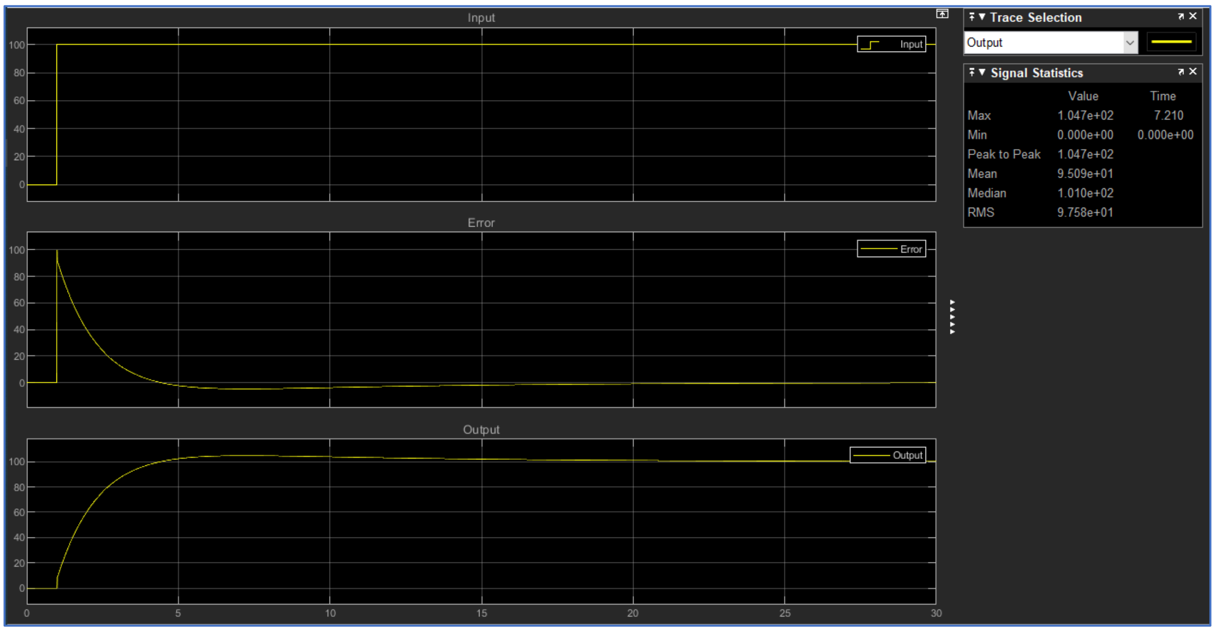


Figure 6: Feedback Cruise Control

FEEDBACK LOOP RESPONSE OF CRUISE CONTROL WITH PID CONTROLLER



Signal Statistics

	Value	Time
Max	1.047e+02	7.210
Min	0.000e+00	0.000e+00
Peak to Peak	1.047e+02	
Mean	9.509e+01	
Median	1.010e+02	
RMS	9.758e+01	

Figure 7: Feedback loop response Cruise Control

With proper tuning and feedback loop response, desired response with timing, overshoot and steady state errors are met as per given requirements.

Further details on tuning of PID controller Refer Section PID Tuning.

PART D – MOTOR SPEED CONTROL

BRIEF ON MOTOR SPEED CONTROL

A motor controller is a device that operates a brushless DC motor by coordinating the motor's performance. It uses a variety of manual and automatic means to start and stop the motor. It can also control the speed, limit the torque, and prevent overloading the motor.

FUNCTIONAL REQUIREMENTS OF MOTOR SPEED CONTROL

- Design Motor Speed Control plant model
- Reuse the PID controller model designed in PART B
- Tune Kp, Ki, Kd gains for the PID controller of Motor Speed Control as follows:
    a. Rise time < 5s
    b. Overshoot < 5%
    c. Steady-state error < 1%

TRANSFER FUNCTION OF MOTOR SPEED CONTROL

$$\frac{d^2(\theta)}{dt^2} = \frac{1}{J}\left(K_t i - b\,\frac{d(\theta)}{dt}\right)$$

$$\frac{d(i)}{dt} = \frac{1}{L}\left(-Ri + V - \ K_e\,\frac{d(\theta)}{dt}\right)$$

$$P(s) = \frac{K_d}{((Js+b)(Ls+R)+K^2)}$$

where,

$J = 0.01$

$b = 0.01$

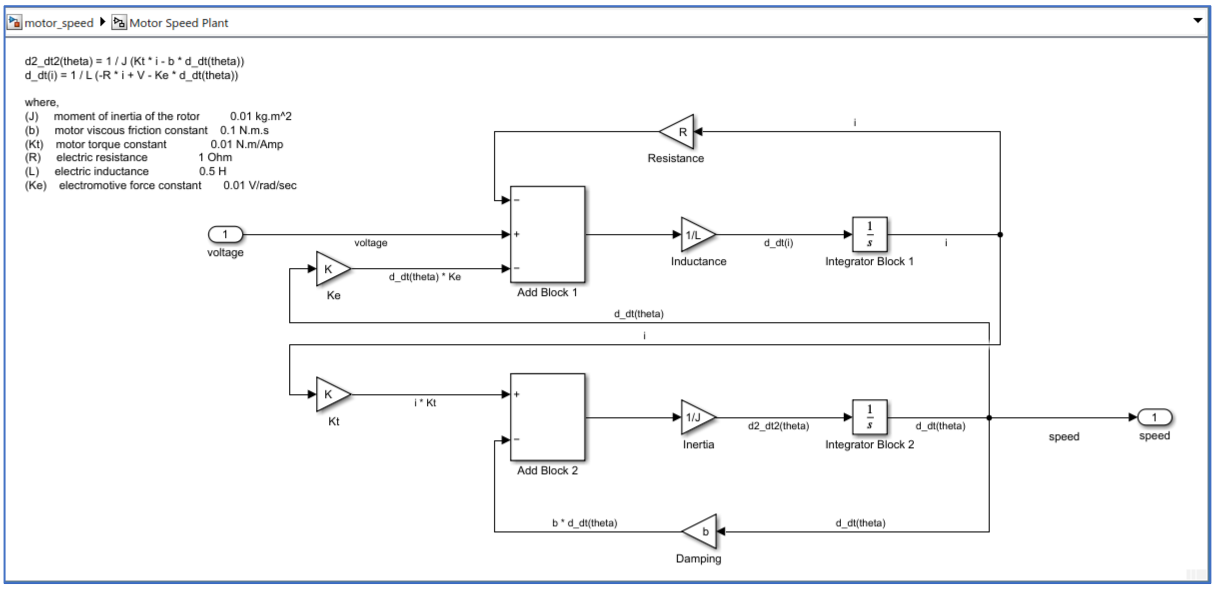
$K\ = 0.01$

$R = 1$

$L = 0.5$

SUBSYTEM DESIGN OF MOTOR SPEED CONTROL

Using the equations defined in Section here above, Motor Speed Control plant model is designed in the Simulink. Refer Figure 8.

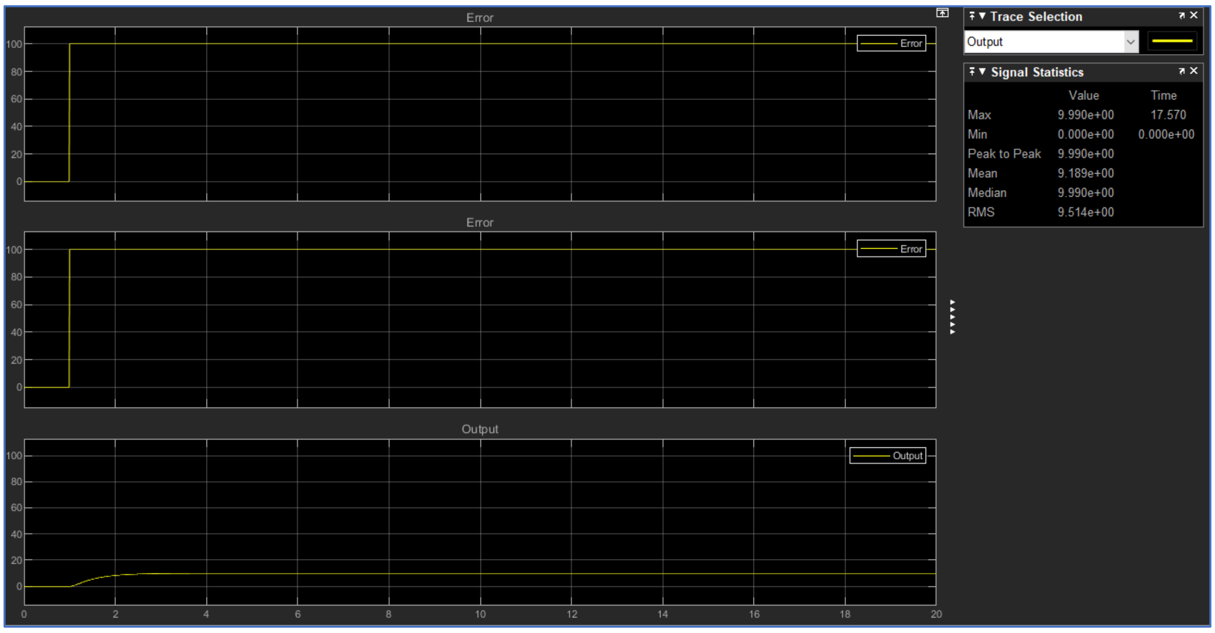


```
d2_dt2(theta) = 1 / J (Kt * i - b * d_dt(theta))
d_dt(i) = 1 / L (-R * i + V - Ke * d_dt(theta))

where,
(J)    moment of inertia of the rotor        0.01 kg.m^2
(b)    motor viscous friction constant    0.1 N.m.s
(Kt)   motor torque constant              0.01 N.m/Amp
(R)    electric resistance                1 Ohm
(L)    electric inductance                0.5 H
(Ke)   electromotive force constant       0.01 V/rad/sec
```

Figure 8: Motor Speed Control Plant Model

OPEN LOOP RESPONSE OF MOTOR SPEED CONTROL



Signal Statistics

	Value	Time
Max	9.990e+00	17.570
Min	0.000e+00	0.000e+00
Peak to Peak	9.990e+00	
Mean	9.189e+00	
Median	9.990e+00	
RMS	9.514e+00	

Figure 9: Openloop response Motor Speed Control

Open loop response for Motor Speed Control shows the model designed is not able to reach the desired output for the given step input. Refer Figure 9.

FEEDBACK LOOP MODEL OF MOTOR SPEED CONTROL

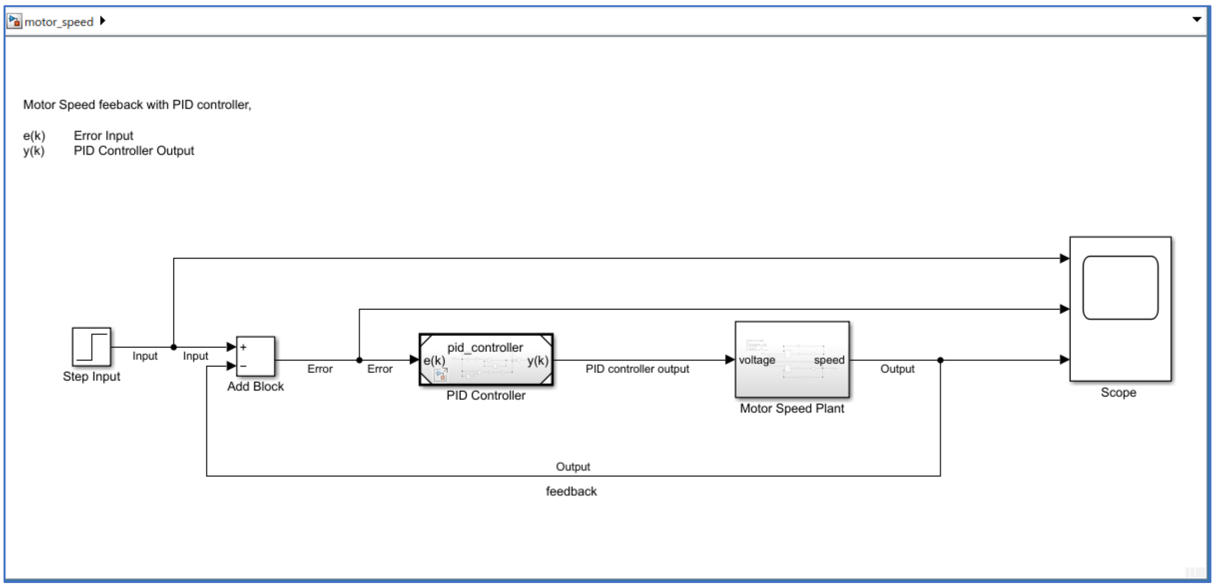
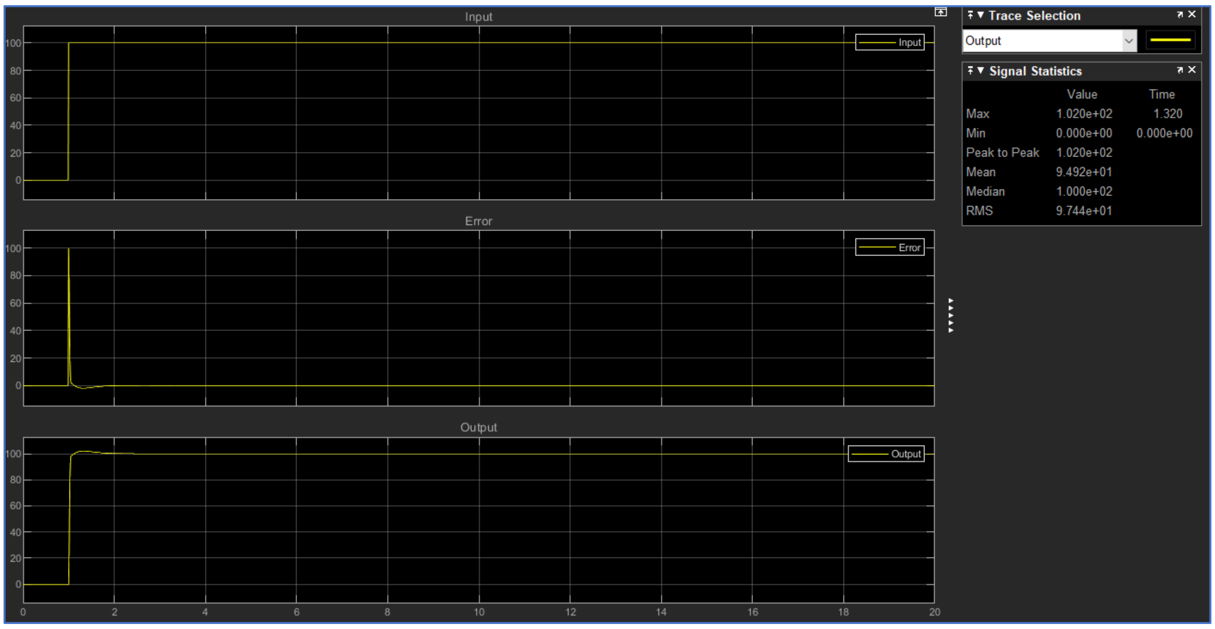


Figure 10: Feedback Motor Speed Control

FEEDBACK LOOP RESPONSE OF MOTOR SPEED CONTROL WITH PID CONTROLLER



	Value	Time
Max	1.020e+02	1.320
Min	0.000e+00	0.000e+00
Peak to Peak	1.020e+02	
Mean	9.492e+01	
Median	1.000e+02	
RMS	9.744e+01	

Figure 11: Feedback loop response

With proper tuning and feedback loop response, desired response with timing, overshoot and steady state errors are met as per given requirements.

Further details on tuning of PID controller Refer Section.

PID TUNING

Tuning of Cruise Control and Motor Speed Control involved input of step signal as the reference signal. Output was meased using the scope. The measurement of the output signal can be seen in the measurement window of the scope output.

Sampling time used for this simulation is T_s = 0.01.

K_p, K_i and K_d values were increased gradually for a change in output response.

K_p gain value was increased or decreased to obtain the fast response time for the simulated model.

K_i gain value was increased or decreased to decrease steady-state error.

K_d gain value was increased or decreased to reduce overshoot. K_d gain was kept within range to avoid damping or noise in the systems designed.

Table 1: PID Tuning reference

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
$\boldsymbol{K_p}$	Decrease	Increase	Small Change	Decrease
$\boldsymbol{K_i}$	Decrease	Increase	Increase	Decrease
$\boldsymbol{K_d}$	Small Change	Decrease	Decrease	No Change

PID TUNED VALUES FOR CRUISE CONTROL

$$K_p = 890$$

$$K_i = 110$$

$$K_d = 80$$

Achieved values was tested with step response, below the actual metrices for Cruise Control

- Rise time ≅ 3s
- Settling time ≅ 0.4s
- Overshoot ≅ 5%

PID TUNED VALUES FOR MOTOR SPEED CONTROL

$$K_p = 210$$

$$K_i = 630$$

$$K_d = 20$$

Achieved values was tested with step response, below the actual metrices for Cruise Control

- Rise time ≅ 0.05s
- Settling time ≅ 0.5s
- Overshoot ≅ 3%

CODE GENERATION

MATLAB built in feature Embedded Coder is used to auto generate the code for the PID Controller designed.

Following are the setting used for auto code generation,

- RAM efficiency
- MISRA C:2012 guidelines
- Execution efficiency
- Traceability
- Safety precaution

With is setting enabled the generated code is industry standard compliant and same was confirmed with static analyser tool.

STATIC ANALYSIS

Static program analysis is the process of analyzing a computer program that is not executed. This procedure is usually performed on a version of the program that has been compiled or is otherwise not running.

The term is often used to describe the analysis conducted by an automated tool or a human reviewer during a programming or program analysis.

The generated code was run under static analyser tool named clang-tidy in command window.

Below is the command used to perform the analysis,

```
clang-tidy <file-name> -checks=”*”
```

In this case the <file-name> is replaced by pid_controller.c

The tool execution was successful with no errors.

MATLAB has built-in feature to check for errors, once the code is generated a window showing all the analysis with warning and errors, can be fixed without leaving the development environment or switching of tools is eliminated.

VERSION CONTROL

Git is a software package that enables developers to keep track of changes in any set of files. Its goal is to provide a fast, secure, and reliable way to collaborate on code.
Linus Torvalds created Git in 2005 to develop the Linux kernel. Its initial development was done by other developers. Git is a distributed version control system that enables anyone to create a repository of their own.
To ensure proper flow of development cycle git version was made in place. The entire design of model and code generation was under version control.
Git gives us the advantage of reverting the changes at any stage.
There are two types of version control system,

- Local
- Global

The Global version control is used for collaboration and when people having many different features and sharing code among the team.
GitHub is used for remote storage. Even if the local changes are lost we could restore files from remote repository.
In this course work, branching strategy was implemented to easy track of all the changes made.
Each feature had a branch of its own. Further, once the feature is complete the branch was merged into the dev branch.
This keeps all the branches independent and in software development people working on different branches can work on their independent branch and commit and revert as many times required on completion the same can we sent to merge using the merge request or pull request.

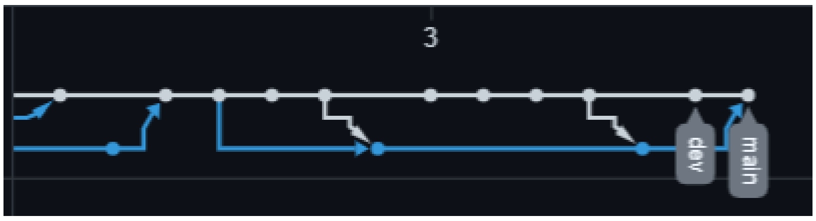


Figure 12: Git Branching

ADVANTAGES AND DISADVANTAGES

V-MODEL:

ADVANTAGES OF V-MODEL:

- Simple in its usage
- Industry proven standards
- All the requirements are defined in the initial stage
- Clear defining of design decisions
- Fall back if and stage fails

DISADVANTAGES OF V_MODEL:

- Not flexible in the process change
- No product visible till the end-of-life cycle
- Cannot be used for projects with varying requirements

MODEL BASED DESIGN(MBD)

ADVANTAGES OF MBD:

- Simulation of models for various systems
- Rapid prototyping of real systems
- Sandboxing
- Auto code generation
- Scalability of models designed
- Modularity of components
- Maintaining of designed models

DISADVANTAGES OF MBD:

- High computational power
- Requires mathematical background to work on or design

- Significant processing costs and delay

VERSION CONTROL

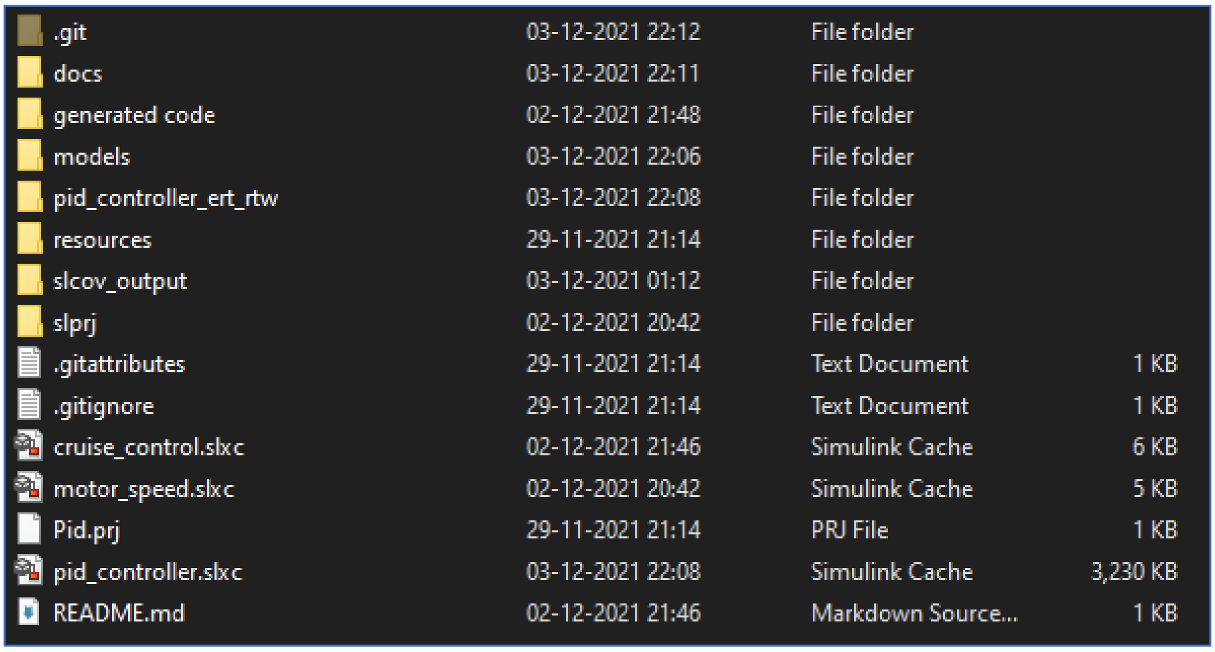
ADVANTAGES OF VERSION CONTROL

- Flexibility to use
- Tracking changes

DISADVANTAGES OF VERSION CONTROL

- Conflict resolution
- Managing branches and pipelines

FOLDER STRUCTURE



.git	03-12-2021 22:12	File folder	
docs	03-12-2021 22:11	File folder	
generated code	02-12-2021 21:48	File folder	
models	03-12-2021 22:06	File folder	
pid_controller_ert_rtw	03-12-2021 22:08	File folder	
resources	29-11-2021 21:14	File folder	
slcov_output	03-12-2021 01:12	File folder	
slprj	02-12-2021 20:42	File folder	
.gitattributes	29-11-2021 21:14	Text Document	1 KB
.gitignore	29-11-2021 21:14	Text Document	1 KB
cruise_control.slxc	02-12-2021 21:46	Simulink Cache	6 KB
motor_speed.slxc	02-12-2021 20:42	Simulink Cache	5 KB
Pid.prj	29-11-2021 21:14	PRJ File	1 KB
pid_controller.slxc	03-12-2021 22:08	Simulink Cache	3,230 KB
README.md	02-12-2021 21:46	Markdown Source...	1 KB

Figure 13: Folder Structure

GITHUB REPOSITORY LINK

GitHub repository link: https://github.com/adish-v/7146CEM.git

CONCLUSIONS

Model Based design has been chosen to implement the system as its easy to tune and test our designed models in a sandbox before testing it on a real hardware or system.

Once response is meeting the requirements the same model could be tested in real scenario.

Hence, the use of MBD is advantageous in this course work carried out.

Tuned values are in response to the changes of input in real scenarios.

Version Control helped in keep in track for the changes and kept for smooth functioning. This version control can also be linked to get pipelines to automate the git flow.

REFERENCES

- https://en.wikipedia.org/wiki/PID_controller
- https://en.wikipedia.org/wiki/Cruise_control
- https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=ControlPID
- https://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControl&section=SimulinkModeling
- https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SimulinkModeling
- https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines
- https://tlemp.com/download/rule/MISRA-CPP-2008-STANDARD.pdf
- https://en.wikipedia.org/wiki/Static_program_analysis
- https://en.wikipedia.org/wiki/Git