7146CEM: Automotive Software Engineering - Design and Development

Coursework

Vignesh Babu Aravamuthan

[adnanv@coventry.ac.uk](mailto:adnanv@coventry.ac.uk)

**Table of content**

[1 Introduction 4](#_Toc89280360)

[2 Software Development Life Cycle 4](#_Toc89280361)

[2.1 Requirement Gathering 4](#_Toc89280362)

[2.1.1 PID controller 4](#_Toc89280363)

[2.1.2 Cruise Control 4](#_Toc89280364)

[2.1.3 Motor Speed 5](#_Toc89280365)

[2.1.4 Code 5](#_Toc89280366)

[2.2 Design 5](#_Toc89280367)

[2.2.1 PID Controller 5](#_Toc89280368)

[2.2.2 PID Tuning Algorithm 7](#_Toc89280369)

[2.2.3 Cruise Control 7](#_Toc89280370)

[2.2.4 Motor Speed 8](#_Toc89280371)

[2.3 Development and Coding 9](#_Toc89280372)

[2.3.1 Cruise Control 9](#_Toc89280373)

[2.3.2 Motor Speed Control 12](#_Toc89280374)

[2.4 Testing 14](#_Toc89280375)

[2.5 Validation 14](#_Toc89280376)

[3 Advantages of the used SDLC model 15](#_Toc89280377)

[4 GitHub Workflow 15](#_Toc89280378)

[5 References 15](#_Toc89280379)

**Table of figures**

[Figure 1 P Controller Implementation 6](#_Toc89280387)

[Figure 2 I Controller Implementation 6](#_Toc89280388)

[Figure 3 D controller Implementation 6](#_Toc89280389)

[Figure 4 PID Controller Implementation 7](#_Toc89280390)

[Figure 5 Simulation Report 7](#_Toc89280391)

[Figure 6 Cruise Control Schematic 8](#_Toc89280392)

[Figure 7 Motor Speed Schematic 8](#_Toc89280393)

[Figure 8 Cruise Control Block Diagram 9](#_Toc89280394)

[Figure 9 Cruise Plant 10](#_Toc89280395)

[Figure 10 Cruise Control Model 10](#_Toc89280396)

[Figure 11 Cruise Control System Simulation Result 11](#_Toc89280397)

[Figure 12 Cruise Control Rise Time Plot 11](#_Toc89280398)

[Figure 13 Cruise Control Overshoot Plot 11](#_Toc89280399)

[Figure 14 Motor Speed Control Block Diagram 12](#_Toc89280400)

[Figure 15 Motor plant 12](#_Toc89280401)

[Figure 16 Motor Speed Control Model 13](#_Toc89280402)

[Figure 17 Motor Speed Control System Simulation Result 13](#_Toc89280403)

[Figure 18 Motor Speed Control Rise Time Plot 13](#_Toc89280404)

[Figure 19 Motor Speed Control Overshoot Plot 14](#_Toc89280405)

**Table of tables**

[Table 1 Technical Requirements 5](#_Toc89280406)

[Table 2 Technical Requirements 5](#_Toc89280407)

[Table 3 Cruise Control Model Validation 14](#_Toc89280408)

[Table 4 Motor Speed Model Validation 15](#_Toc89280409)

# Introduction

This document contains information and details regarding the workflow used to create the PID controller, tuning of PID, and generation of code for cruise control and motor speed control project. GitHub is used as a version control system for the project. It is integrated with MATLAB to easily facilitate the GitHub process.

Control systems are needed when the system deals with continuously varying parameters or external disturbances such as load, friction, wind, etc, which will affect the output of the system. To maintain the stable output of the system even in the presence of external disturbances a controller is needed to control those output variations. There are a lot of controlling algorithms available for the control system, we will use only a PID controller for cruise control and motor speed.

# Software Development Life Cycle

This section gives an overall view of the software development process used to develop the cruise control and motor speed model, starting with requirements, design, development, testing, and validation.

## Requirement Gathering

This is the first stage of the V-development cycle contains a detailed understanding of requirements and expectations for the final product.

### PID controller

#### Technical Requirements

* To design the PID controller using the following equations

Where,

* PID Controller block should contain discrete function blocks.

#### Non-Functional Requirement

* PID controller model should be designed and made it to referenced model.

### Cruise Control

#### Function Requirement

The system should have the following functional requirement

* Speed of the car should not fluctuate with respect to the external disturbances.

#### Technical Requirement

The system should have the following technical requirements

|  |  |
| --- | --- |
| **S.No** | **Requirements** |
| 1 | Rise time < 10s |
| 2 | Overshoot < 10% |
| 3 | Stead state error <1% |

Table 1 Technical Requirements

### Motor Speed

#### Technical Requirement

The system should have the following technical requirements

|  |  |
| --- | --- |
| **S.No** | **Requirements** |
| 1 | Rise time < 5s |
| 2 | Overshoot < 5% |
| 3 | Stead state error <1% |

Table 2 Technical Requirements

### Code

Developed code for the controller should have the following requirements:

* Code should be optimized for RAM efficiency.
* To develop the code in accordance with ISO26262.
* Code should follow MISRA C Guidelines.

## Design

### PID Controller

#### P Controller Design

P controller is design by implementing the following equation in Simulink. The following is model is made as separate subsystem to make it as a modular design.

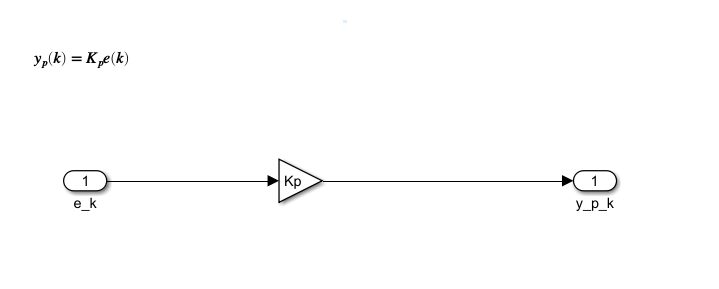


Figure 1 P Controller Implementation

#### I Controller Design

I controller is design by implementing the following equation in Simulink. The following is model is made as separate subsystem to make it as a modular design.

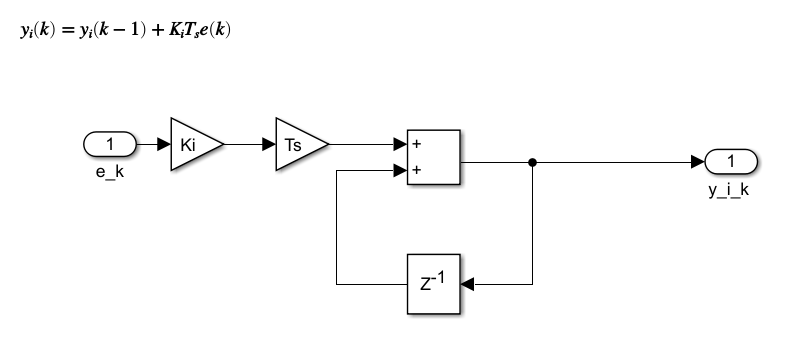


Figure 2 I Controller Implementation

#### D Controller Design

D controller is design by implementing the following equation in Simulink. The following is model is made as separate subsystem to make it as a modular design.

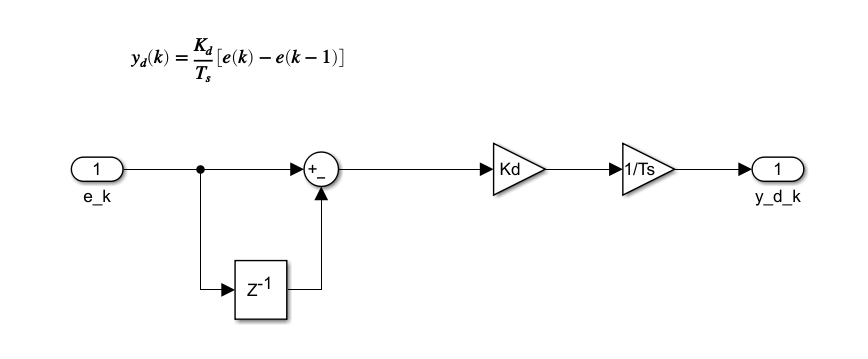


Figure 3 D controller Implementation

#### PID Controller Design

P, I, D controllers which are designed as a separate subsystem are integrated together by implementing following equation. Sampling time of the PID Controller is set to value Ts=0.01.

This model is the converted to an atomic subsystem to make it as a referenced subsystem. Referenced subsystem are useful in case of multiple definition of the same PID controller is needed. This referenced subsystem is then used for Cruise control model and Motor speed control.

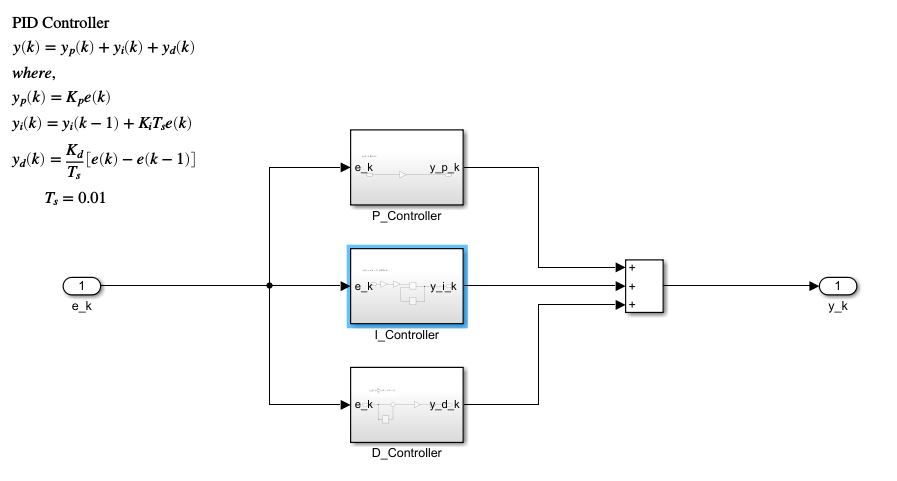


Figure 4 PID Controller Implementation

### PID Tuning Algorithm

Script ‘PID\_Turning\_Script.mlx’ is created to obtain Kp, Ki and Kd values to satisfy the requirements. Script uses trial and error method to find the values. Calculation for overshoot, rise time, steady state error for the output signal is to be implemented in the script to parallelly check whether the requirements are met with corresponding Kp, Ki and Kd values.

Script will also generate a short report (refer Figure 5) of output signal and its properties such as rise time, overshoot percentage, output value at simulation stop time, steady state error percentage for the corresponding Kp, Ki, Kd values.

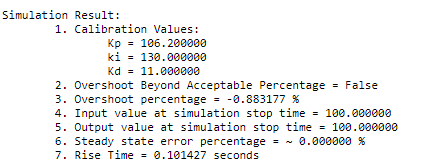


Figure 5 Simulation Report

### Cruise Control

#### Analysis

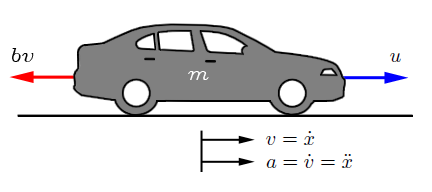


Figure 6 Cruise Control Schematic

The vehicle, of mass m, is acted on by a control force, u. The force u represents the force generated at the road/tire interface. For this simplified model we will assume that we can control this force directly and will neglect the dynamics of the powertrain, tires, etc., that go into generating the force. The resistive forces, bv, due to rolling resistance and wind drag, are assumed to vary linearly with the vehicle velocity, v, and act in the direction opposite the vehicle's motion.

#### System equations

With these assumptions we are left with a first-order mass-damper system. Summing forces in the x-direction and applying Newton's 2nd law, we arrive at the following system equation:

#### System parameters

Parameters of the system are assumed with following values:

* Mass of the Vehicle (m) = 1000
* Damping Coefficient (b) = 50

### Motor Speed

#### Analysis

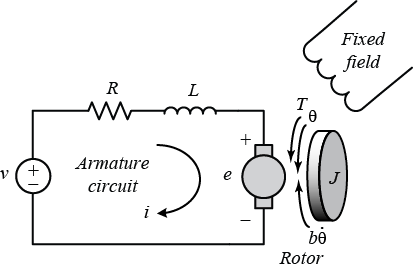


Figure 7 Motor Speed Schematic

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide translational motion. The electric equivalent circuit of the armature and the free-body diagram of the rotor are shown in the following figure.

The input of the system is the voltage source () applied to the motor's armature, while the output is the rotational speed of the shaft . The rotor and shaft are rigid. The friction torque is proportional to shaft angular velocity.

#### System Equations

The torque generated by a DC motor is proportional to the armature current and the strength of the magnetic field. In this example we will assume that the magnetic field is constant and, therefore, that the motor torque is proportional to only the armature current  by a constant factor  as shown in the equation below. This is referred to as an armature-controlled motor.

The back emf, , is proportional to the angular velocity of the shaft by a constant factor

The motor torque and back emf constants are equal, that is, ; therefore, we will use  to represent both the motor torque constant and the back emf constant. We can derive the following governing equations based on Newton's 2nd law and Kirchhoff's voltage law.

#### System Parameters

Parameters of the system are assumed with following values:

* Moment of inertia of the rotor () = 0.01
* Motor viscous friction constant () = 0.1
* Electromotive force constant () = 0.01
* Motor torque constant () = 0.01
* Electric resistance () = 1
* Electric inductance () = 0.5

## Development and Coding

Script for cruise control model is located inside ‘TASK 3/A\_CruiseControl/Scripts’ and for motor speed control it is located inside ‘TASK 3/B\_MotorSpeed/Scripts’.

### Cruise Control

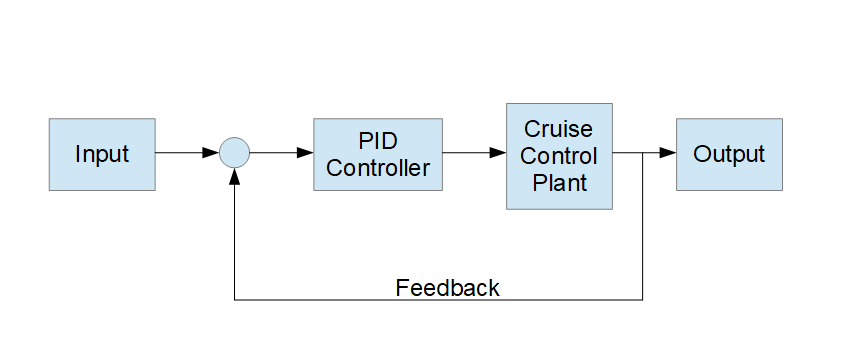


Figure 8 Cruise Control Block Diagram

Cruise control system is planned to implement as displayed in the Figure 8. Here, Input will be replaced by a ‘Step input’ block. For PID controller, referenced subsystem which is created in task 1 is used. Cruise control plant is modelled (refer Figure 9) using the mathematical equation defined in 2.2.3.2 above, which is derived from the analysis mentioned in 2.2.3.1 above. With this the model is developed in Simulink (refer Figure 10).

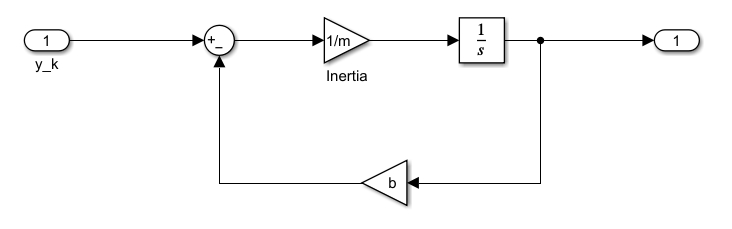


Figure 9 Cruise Plant

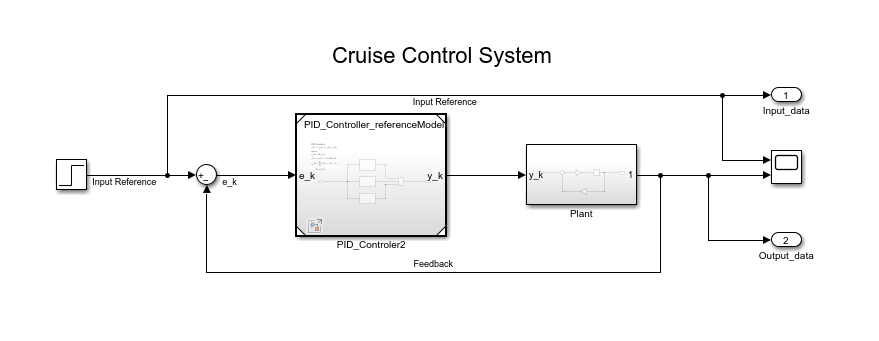


Figure 10 Cruise Control Model

Parameter ‘b’ and ‘m’ are need to complete the design, so values are assumed as per mentioned in 2.2.3.3 above, and saved the value in MATLAB base workspace. Kp, Ki, Kd values are tuned using the script which is developed. By trial and error method, Ki and Kd values are fixed and varied the Kp parameter once the output signal has reached the desired value then Kp values is set as constant and varied Ki alone, by increasing Ki value it is observed that steady state error was decreasing, when the steady state error percentage is less than the percentage mentioned in requirements of steady state error Ki values is kept constant, now to remove the overshoot created by increased Ki, Kd is increased to suppress the overshoot. Once the overshoot percentage is less than the percentage mentioned in requirements of overshoot, calibration is stopped. Now, PID controller is tuned to meet the requirements. Figure 11 , Figure 12, Figure 13 are the simulation result, rise time plot and overshoot plot respectively, is the generated report from the ‘‘PID\_Turning\_Script.mlx’’ script using the Tuned PID Controller.

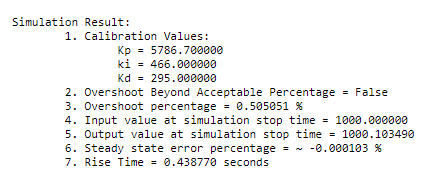


Figure 11 Cruise Control System Simulation Result

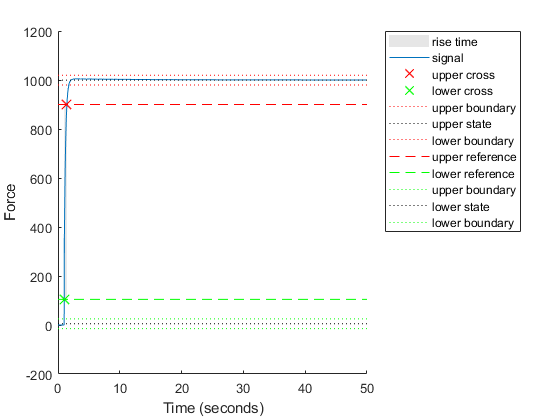


Figure 12 Cruise Control Rise Time Plot

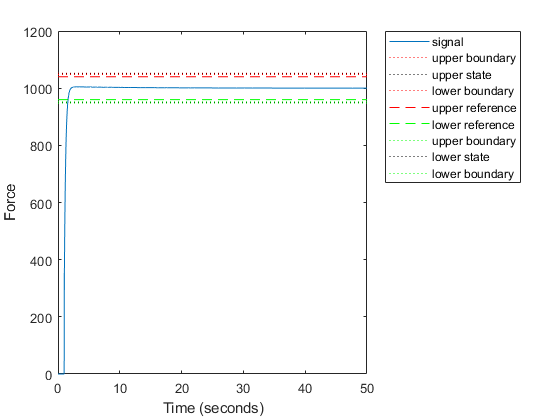


Figure 13 Cruise Control Overshoot Plot

### Motor Speed Control

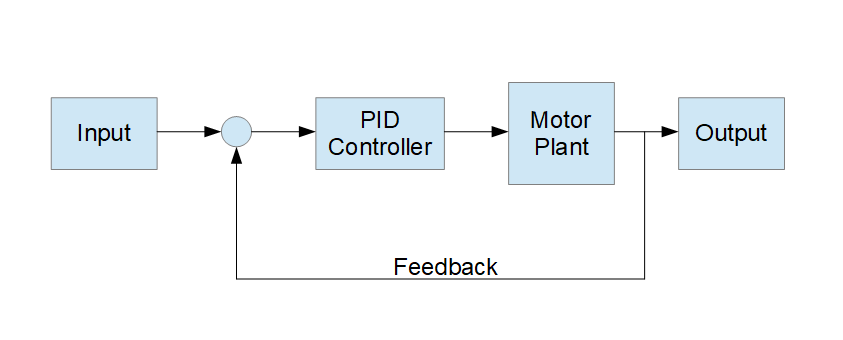


Figure 14 Motor Speed Control Block Diagram

Motor speed control system is planned to implement as displayed in the Figure 14. Here, Input will be replaced by a ‘Step input’ block. For PID controller, referenced subsystem which is created in task 1 is used. Motor speed control plant is modelled (refer Figure 15) using the mathematical equation defined in 2.2.4.2 above, which is derived from the analysis mentioned in 2.2.4.1. With this the model is developed in Simulink (refer Figure 16).

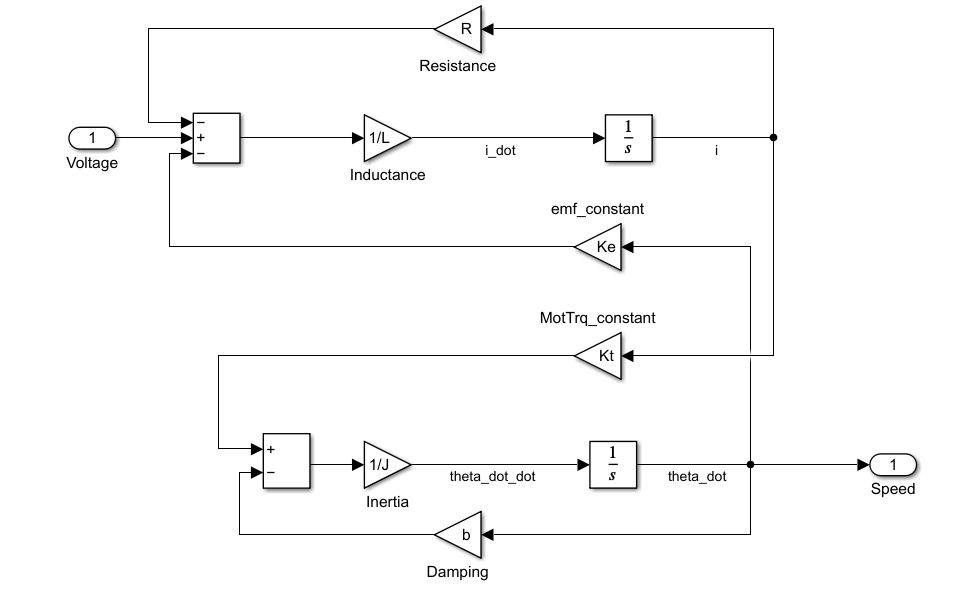


Figure 15 Motor plant

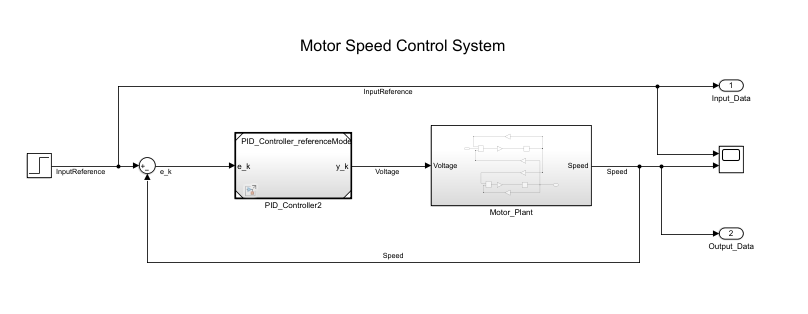


Figure 16 Motor Speed Control Model

Parameter ‘L’, ‘R’, ‘Ke’, ‘Kt’, ‘J’, and ‘b’ are need to complete the design, so values are assumed as per mentioned in 2.2.4.3, and saved the value in MATLAB base workspace. Kp, Ki, Kd values are tuned using the script which is developed. By trial and error method, Ki and Kd values are fixed and varied the Kp parameter once the output signal has reached the desired value then Kp values is set as constant and varied Ki alone, by increasing Ki value it is observed that steady state error was decreasing, when the steady state error percentage is less than the percentage mentioned in requirements of steady state error Ki values is kept constant, now to remove the overshoot created by increased Ki, Kd is increased to suppress the overshoot. Once the overshoot percentage is less than the percentage mentioned in requirements of overshoot, calibration is stopped. Now, PID controller is tuned to meet the requirements. Figure 17, Figure 18, Figure 19 are the simulation result, rise time plot and overshoot plot respectively, are the generated report from the ‘‘PID\_Turning\_Script.mlx’’ script using the Tuned PID Controller.

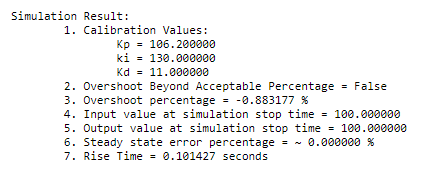


Figure 17 Motor Speed Control System Simulation Result

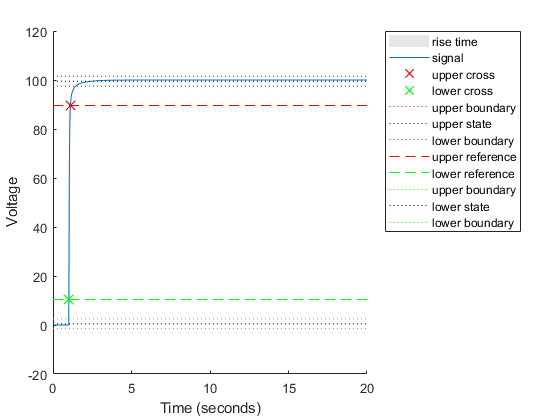


Figure 18 Motor Speed Control Rise Time Plot

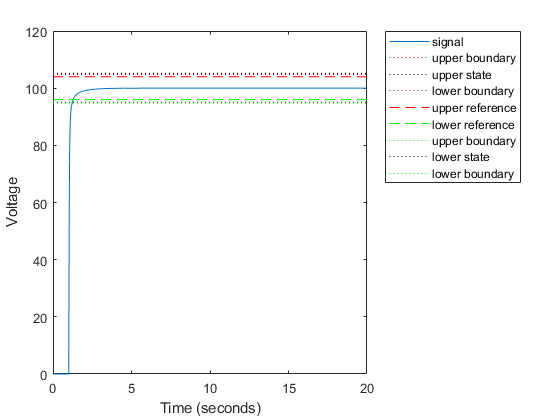


Figure 19 Motor Speed Control Overshoot Plot

### Code generation

Code is generated using embedded coder application in Simulink. According to requirement code should design for “RAM Efficiency”. So, all the configuration setting in the code generation is set to meet the requirement. Code generation advisor is used to check whether the generated code is for “RAM Efficiency” (refer Figure 20) .

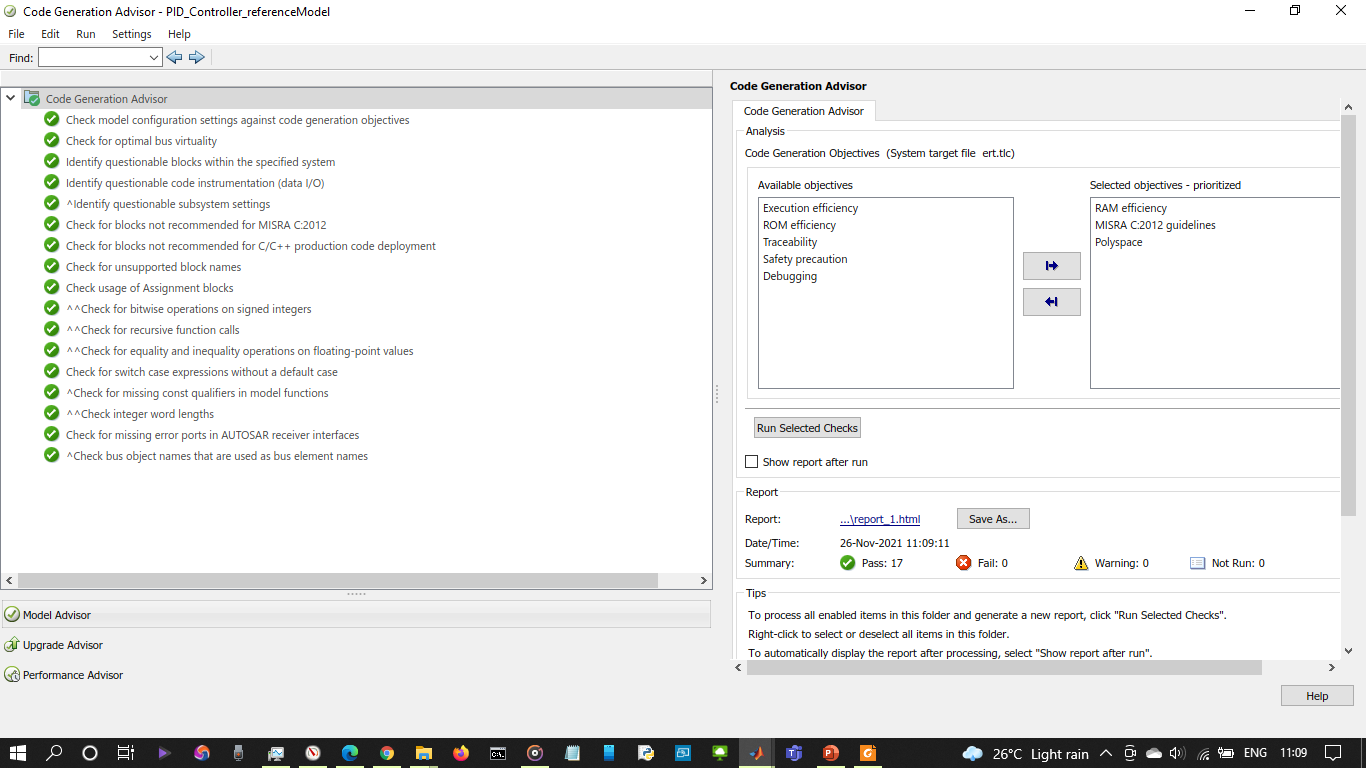


Figure 20 Code Generation Advisor

## Testing

### CPP Chec

CPP check is used to check whether the generated c files are following MISRA C guildlines

Uml diagram is missing

-Unit testing results, Polyspace statics analysis result. Cpp Check result

## Validation

### Validation of Cruise Control Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Requirements | Input | Requirement  Criteria | Result | Status |
| 1 | Rise time | 0-1000  (Step input) | Less than  10 seconds | 0.4 seconds | Accepted |
| 2 | Overshoot | 0-1000  (Step input) | Less than 10% | 0.5% | Accepted |
| 3 | Stead state error | 0-1000  (Step input) | Less than  1% | 0% | Accepted |

Table 3 Cruise Control Model Validation

### Validation of Motor Speed Model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Requirements | Input | Requirement  Criteria | Result | Status |
| 1 | Rise time < 5s | 0-100  (Step input) | Less than  5 seconds | 0.1 Seconds | Accepted |
| 2 | Overshoot < 5% | 0-100  (Step input) | Less than  5 % | 0 % | Accepted |
| 3 | Stead state error <1% | 0-100  (Step input) | Less than  1 % | 0 % | Accepted |

Table 4 Motor Speed Model Validation

# Advantages of the used SDLC model

* Mention the advantages of the SDLC model used with the development of the pid.

# GitHub Workflow

GitHub is integrated with MATLAB, and all the versioning process is done within MATLAB itself. The link for repository is as follows:

<https://github.com/vigneshbabu0717/coursework_7146CEM_vigneshbabu_11911348.git>

# Conclusion

Cruise control and Motor speed control system is developed with V-development cycle as SDLC. Requirements are gathered from the task given, and using the requirements design for both the system is created. After the design, actually development is done in Simulink.(add if testing is done)

# References

*Control Tutorials for MATLAB & Simulink*. (n.d.). Retrieved from https://ctms.engin.umich.edu/CTMS/index.php?example=CruiseControl&section=SystemModeling

*Control Tutorials for MATLAB & Simulink*. (n.d.). Retrieved from https://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling