**ME528: Control System Design**

201934594

Matthew Robert Norrie

# **Abstract**

Modern cruise control systems are a staple of 21st century driving, allowing drivers allocate speed control to the car leading to more consistent speeds, better fuel economy and a less tiring journey.

The aim of this project was to develop a computer representation of a car cruise control system using Simulink and MATLAB. The project was divided into two parts; first using a simplified version of the overall equation of motion for the car, then progressing on to simulation of the non-linear dynamics, summing all velocity variant forces and using these to determine engine output control torque.

A

**Contents**

[**1** **Abstract** i](#_Toc152684264)

[3 **Part 1** [20 Page Limit] 1](#_Toc152684265)

[*3.1* *Background* 1](#_Toc152684266)

[*3.2* *Methodology* 1](#_Toc152684267)

[3.2.1 Simulink Model 1](#_Toc152684268)

[3.2.2 Data Logging 1](#_Toc152684269)

[3.2.3 Controller Testing 2](#_Toc152684270)

[3.3 *Results & Analysis* 3](#_Toc152684271)

[3.3.1 Overall Performance Assessment 3](#_Toc152684272)

[3.3.2 Focussed Stability & Performance Assessment 9](#_Toc152684273)

[*3.4* *Conclusions* 10](#_Toc152684274)

[**4** **Part 2 [15 Page Limit]** 11](#_Toc152684275)

[*4.1* *Background* 11](#_Toc152684276)

[*4.2* *Methodology* 11](#_Toc152684277)

[4.2.1 Simulink Model 11](#_Toc152684278)

[4.2.2 Feedback Linearised Control 11](#_Toc152684279)

[4.3 11](#_Toc152684280)

[*4.4* *Results* 11](#_Toc152684281)

[4.4.1 Phase Space & Equilibrium 11](#_Toc152684282)

[*4.5* *Conclusions* 11](#_Toc152684283)

[**5** **Report Conclusions** 12](#_Toc152684284)

[**6** **Suggestions for Modification** 12](#_Toc152684285)

[*6.1* *Submission Overlap & Assessment Weighting* 12](#_Toc152684286)

[6.2 Choices for control system 12](#_Toc152684287)

[**7** **References** 13](#_Toc152684288)

# **Part 1** [20 Page Limit]

## *Background*

P, I, D, PI, PD, ID, PID graph to assess what aspects are needed.

Effects of each gain on controller behaviour.

What are root locus, bode plot and routh Hurwitz.

Calculating damping coefficient from gains.

## *Methodology*

### Simulink Model

For part 1, the following Simulink model shown below in Figure 1 was developed.

A diagram of a machine

Description automatically generated

Figure - Part 1 Simulink Model

Initially, the model used a step input to represent set velocity, however this was changed to a ramp input. The ramp input used a gradient that corresponded to a typical saloon car’s 0 – 60 mph time, in this case 8.7s which corresponded to 6.9 mph/s. Converting units to m/s gave a 0 – 26.82 m/s time which corresponded to 3.08 m/s2.

A saturation block was used to limit the input velocity of the ramp block. The lower limit of the saturation was set to 0 m/s, and the upper limit of saturation was set to 30 m/s. This corresponded in the simulation to accelerating from rest to 30 m/s in 9.74s. These settings were used for the following data logging steps.

### Data Logging

Initially, PID values were manually iterated and recorded to an excel file. Each iteration involved a change to a single gain or disturbance variable, in this case these included , , , and . Around 50 iterations were performed manually, with a comment of the results added to the excel sheet along with the accompanying gains.

This method was good for getting a general feel to how the variables affected the controller behaviour, and further showed that a PI controller was suitable for the problem. In the case of this report, however, it was not particularly effective for determining an effective combination of gains. The process was slow and tedious, and required a lot of manual data logging.

An output block was added to the Simulink file, which allowed the integration of a MATLAB script. By running the MATLAB script, Simulink would run and output the data as a structure. By importing the structure to MATLAB and logging the structure data to an array along, the controller behaviour was recorded for a given set of PID gains.

The use of a MATLAB script for running meant that variables used in Simulink were externally controlled. These variables were manually input for testing the script, and then initially again for iterating and testing controller behaviour. Doing this manually though was not necessary, as MATLAB allowed for the automatic iteration of gains and data logging.

Figure 1 was used for both stages of data logging,

By setting up 3 nested for loops, each of the three gains, , , and were iterated between a specified minimum and maximum value at a specified step size for each. For each simulation the controller behaviour and damping ratio was recorded using MATLAB’s inbuilt function – damp(sys).

This initial script was iterated through values for and , leaving for all iterations and produced a large array of gain combinations and performance statistics for each control system. To select suitable PI combinations from this large array, a second script was created which used the parameters of:

* Overshoot – Greater than 0%, Less than 4%.
* Damping Ratio – Greater than 0.75, Less than 1.

These conditions ensured desirable controller behaviour, and eliminated the majority of the undesirable PI combinations which left a selection small enough to manually analyse and choose from.

### Controller Testing

Three sets of gains were chosen for further study. For each set of gains, the Bode plot, Root Locus plot, and Routh-Hurwitz Array were generated.

## *Results & Analysis*

### Overall Performance Assessment

The following: Figure 2, Figure 5, and Figure 8 are shown throughout this section. These were chosen as 3 PI combinations from the refined array. Each combination was chosen for a specific and unique reason and studied further.

#### Kp = 0.26 , Ki = 0.02

A graph of a graph

Description automatically generated with medium confidence

Figure - Kp = 0.26 , Ki = 0.02: Velocity & Time

MATLAB determined this PI combination was numerically within the constraints set within the selection script, those being touched on above. In this case, the damping coefficient was determined to be , so it is barely within the range and is underdamped. This can be confirmed by analysing the root locus plot:

A graph of a line

Description automatically generated

Figure - Kp = 0.26 , Ki = 0.02: Root Locus

Shown in Figure 3, the poles on the imaginary axis represent that the system is underdamped. As well as this, all of the poles and zeros on the plot are to the negative side of the real axis, representing that the system will remain stable. The stability of the system can be further verified using its Bode plot and Routh Hurwitz Array, shown below:

A graph of a function

Description automatically generated

Figure - Kp = 0.26 , Ki = 0.02: Bode Plot

|  |  |  |
| --- | --- | --- |
| Routh – Hurwitz Array: | 1.00 | 0.02 |
|  | 0.28 | 0 |
|  | 0.02 | 0 |

There are a few overall flaws with this PI combination, notably the slow response time and lagging response to disturbances. Though MATLAB determined this combination to be within the margin of 4% overshoot, on visual inspection we can see it is much greater than that. This is because the PI combination was tested only for the 0o case and later iterated through multiple gradients for testing. An overshoot this large could be dangerous, especially when it stays so far overshot for more than 30 seconds.

This controller gain combination was determined as unsuitable for use.

#### Kp = 1.81 , Ki = 1.47

A graph with a line going up

Description automatically generated

Figure - Kp = 1.81 , Ki = 1.47: Velocity & Time

Figure 5 above shows a slightly more responsive controller with a much more consistent behaviour between gradients. MATLAB correctly determined this controller to fall withing the 4% overshoot requirement and unlike the previous controller, regardless of the gradient, the controller always overshoots by the same amount when it reaches the set speed. The controller is also underdamped, but by a slightly larger degree this time. MATLAB determined . Underdamped behaviour is confirmed on inspection of the root locus plot, shown below in Figure 6:

A screen shot of a graph

Description automatically generated

Figure - Kp = 1.81 , Ki = 1.47: Root Locus

Much alike Figure 3, Figure 6 has 2 roots on the imaginary axis which highlights underdamped behaviour, confirming the value produced from MATLAB. As well as this, for the 0o case, the plot is entirely on the negative side of the real axis and is therefore stable.

Stability can be corroborated again using the Bode diagram and Routh Hurwitz array, shown below in Figure 7 and below:

A screenshot of a graph

Description automatically generated

Figure - Kp = 1.81 , Ki = 1.47: Bode Plot

|  |  |  |
| --- | --- | --- |
| Routh – Hurwitz Array: | 1.00 | 1.47 |
|  | 1.83 | 0 |
|  | 1.47 | 0 |

This controller was a great candidate for Part 1, with consistent behaviour regardless of gradient and stability and underdamped behaviour at a gradient of zero it met the selection conditions that were specified in the MATLAB selection script. As well as having desirable damping and consistency, it had a relatively snappy response time to the initial increase ramp, and when adjusting to the end of the ramp.

This controller PI combination was determined as potentially suitable for use.

#### Kp = 2.38 , Ki = 1.44

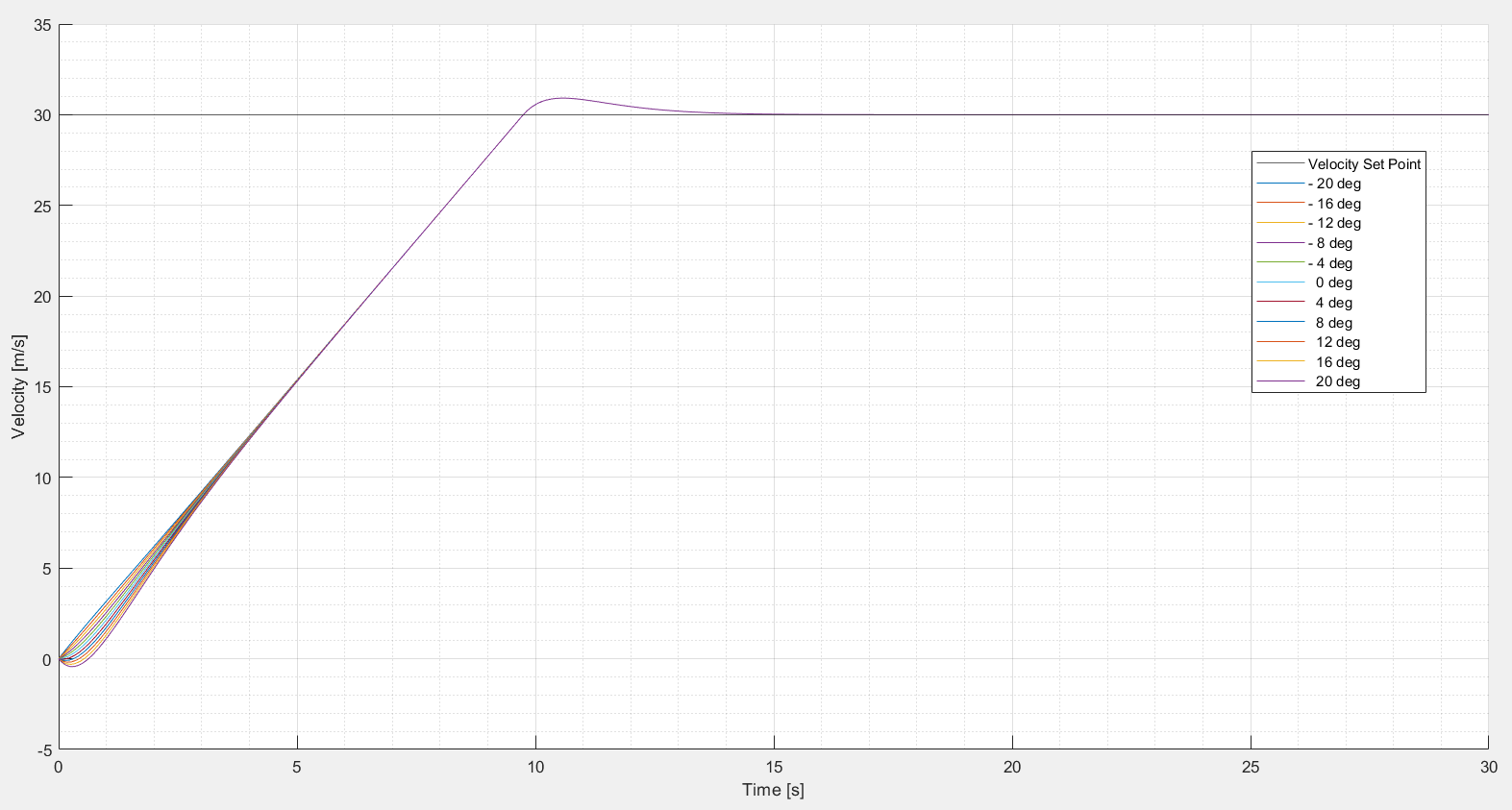


Figure - Kp = 2.38 , Ki = 1.44: Velocity & Time

A screen shot of a graph

Description automatically generated

Figure - Kp = 2.38 , Ki­ = 1.44: Root Locus

A graph of a graph

Description automatically generated

Figure - Kp = 2.38 , Ki = 1.44: Bode Plot

### Focussed Stability & Performance Assessment

It was decided that Controller be carried forward for further detailed study. This controller would now be tested under a number of different conditions to understand its performance and limitations.

The transfer function of the controller was input into MATLAB previously to determine the damping ratio of the system. This transfer function did not include the affects of gradient and therefore was not suitable for analysing gradient effects on stability. The transfer function was modified to include the effects of gradient:

This new transfer function was now suitable for further analysis.

#### Root Locus Differing Gradient

Gradient was again swept between -20o up to 20o, varying by 4o each step. The root locus plot of the study was recorded in MATLAB and is shown in Figure 11 below:

A graph of a diagram

Description automatically generated

Figure - Kp = 1.81 ,Ki = 1.47 Root Locus Gradient Sweep

The Root Locus plot displayed a large variance in behaviour over the range of angles.

At high up-hill angles, system behaviour turned from underdamped to overdamped as the complex component of the poles reduced to zero. It should be noted however that these overdamped cases remain stable as they are on the negative side of the real axis. This is an acceptable change in controller behaviour due to the conditions, however it is unlikely that a 20o uphill section of road will require a cruise control system at all, never mind set to 30 m/s.

Looking over to the other side of the gradient range now, as the downhill angle reaches 12o, the controller has become unstable due to its poles now lying on the positive side of the real axis. The behaviour has also remained underdamped.

Shown in Figure 12, these 3 data points are isolated, allowing them to be more easily seen:

A graph of a circle

Description automatically generated with medium confidence

Figure - Kp = 1.81 ,Ki = 1.47 Focussed Root Locus [-12, -16,-20] degrees

#### Bode Differing Gradient

A screenshot of a computer screen

Description automatically generated

Figure - Kp = 1.81 ,Ki = 1.47 Bode Diagram Gradient Sweep

RH with and without gradient.

## *Conclusions*

# **Part 2 [15 Page Limit]**

## *Background*

## *Methodology*

### Simulink Model

### Feedback Linearised Control

## 

## *Results*

### Phase Space & Equilibrium

Error Graphs! Error against time.

## *Conclusions*

# **Report Conclusions**

fes

# **Suggestions for Modification**

## *Submission Overlap & Assessment Weighting*

## Choices for control system

# **References**

**There are no sources in the current document.**