**ECE 730**

**Control of Adjustable Speed Drives**

Faculty of Engineering – McMaster University – Graduate Studies

Winter 2025

Professor: Babak Nahid-Mobarakeh

**Lab 1 Modeling of Permanent-Magnet Synchronous Machine**

**Report**

**Submitted by**: HaoJun(Henry) Zhao, zhaoh52, 400179164

**Submission Date**: March 2, 2025

Contents

[Introduction 3](#_Toc191822294)

[Methodology 4](#_Toc191822295)

[Simulation & Discussion 6](#_Toc191822296)

[Question 1 6](#_Toc191822297)

[Part II Question 1 & 2 7](#_Toc191822298)

[Part II Question 3 & 4 8](#_Toc191822299)

[Part II Question 5 11](#_Toc191822300)

[Part II Question 6 13](#_Toc191822301)

[Part III Question 1 19](#_Toc191822302)

[Appendix 21](#_Toc191822303)

[Question 1 21](#_Toc191822304)

# Introduction

The purpose of this lab is to model and simulate the vector control of a three-phase Permanent-Magnet Synchronous Machine (PMSM) using MATLAB Simulink. In Lab1, the basic modeling for PMSM is developed but there is direct or indirect torque control for the motor. The primary objective is to design and tune Proportional-Integral (PI) current controllers for the d-axis and q-axis currents to achieve specific performance criteria, such as zero steady-state error, no overshoot, and a closed-loop bandwidth. Furthermore, detailed analysis for is included in the later session. The simulation also includes the effects of parameter uncertainties and the effect on different time domains, for example continuous time domain and discrete time domain, which is essential for practical digital implementations. Through this lab, the simplified closed-loop feedback control for PMSM is generated. The modeling of this lab at a high level is shown in *Figure 1 High-Level Current Control Model of PMSM.*

A diagram of a computer program

AI-generated content may be incorrect.

Figure 1 High-Level Current Control Model of PMSM

# Methodology

To achieve the objectives outlined in this lab, a systematic approach was followed to model and simulate the PMSM in MATLAB-Simulink. To model PMSM properly, the overview of the model is shown in *Figure 2 Simplified PMSM Closed Loop Current Control.*

A diagram of a computer

AI-generated content may be incorrect.

Figure 2 Simplified PMSM Closed Loop Current Control

The main focus of this lab is the control part which is in the blue box. The PMSM model outputs the measured three phase current and motor position . These three phase currents are converted into frame using inverse Clarke transformation and inverse Park transformation. These two conversions simplify the three-phase currents into two rotations frames. These are the necessary procedures in the motor control field to simplify the complexity of the motor control. The Clarke and Park transformation formula is the following:

The converted and control reference current are the inputs of current controller. The current control loop consists of two PI controllers, one for the d-axis current and one for the q-axis current . The goal of these controllers is to ensure that the actual currents ​ and  track and with zero steady-state error and minimal overshoot. The outputs of the PI controllers are the reference voltages for the d-axis and q-axis which are transformed back to the three-phase voltage references using the Park and Clarke transformations. The mathematical representation for current control outputs is the following:

It is easy to see and are coupled with and . The coupling effect increase the difficulty of controller design therefore decoupling strategy must implement. In the controller side, coupling terms must be subtracted. The final and formulas are the following:

The decoupling control ensures that the d-axis and q-axis currents can be controlled independently, improving the dynamic performance of the system. In this model, the process is simplified because of assuming the output from PWM generator and VSI with equals to 1. The PI controllers are designed to achieve the desired performance criteria, such as zero steady-state error, no overshoot, and a specified closed-loop bandwidth (. The actual value for PI controller is decided by pole zero technique which are following:

In order to match the bandwidth requirement, can be decided as follows:

In practical applications, the current control is implemented on a digital control board, which requires the control system to be discretized. By converting the continuous-time control system to a discrete-time system, the control system must be triggered at certain frequency. The sampling frequency is set to 10 kHz. To minimize the coupling effect between the d-axis and q-axis currents due to the discrete-time control, an angle shift is implemented by give a gain to .

# Simulation & Discussion

In this section, the first section is for continuous current controller simulation and the second section is for discrete current controller simulation. To validate the correctness, at and The discussion includes behavior for . The subsection naming matches the parts in lab manual.

## Question 1

In order to achieve zero steady-state error, no overshoot and The equations discussed above are applied and the completed MATLAB code is attached to appdenix:

The results are the following:

## Part II Question 1 & 2

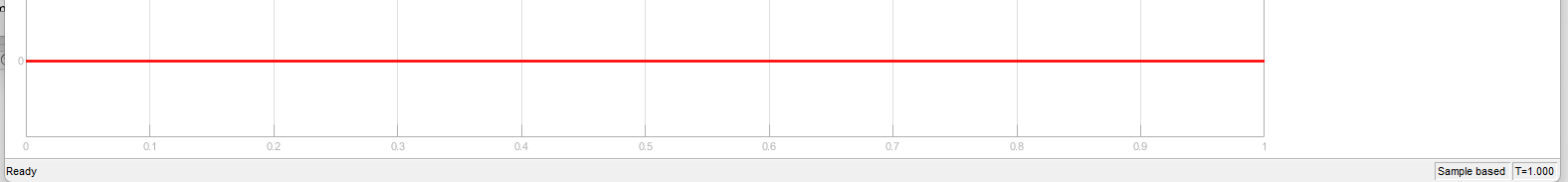
A screen shot of a computer

AI-generated content may be incorrect.

In order to implement PI controller for . The controller shall follow formulas:

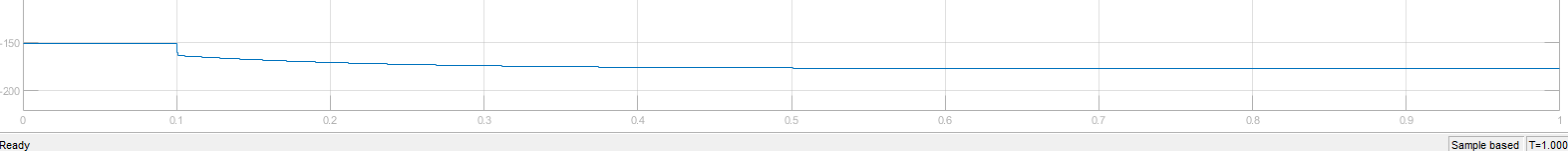
The red highlighted part is the decoupling part. In the code, the PI current control is in discrete time representation instead of continuous time domain which can be directly represented by integrator in Simulink library. The discrete time representation is more appropriate for motor control purpose. In control field, this type of control can be achieved by microcontroller. It is infeasible for a microcontroller to do a continuous integration so that discretization is the process must do. The discretization can be achieved by using discrete time numerical methods for example Euler Methods.

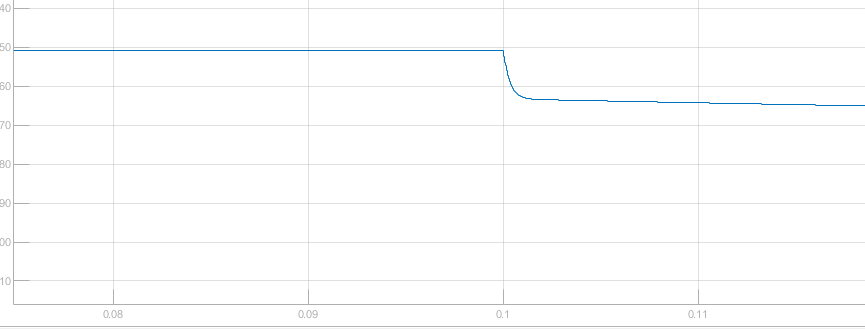
## Part II Question 3 & 4

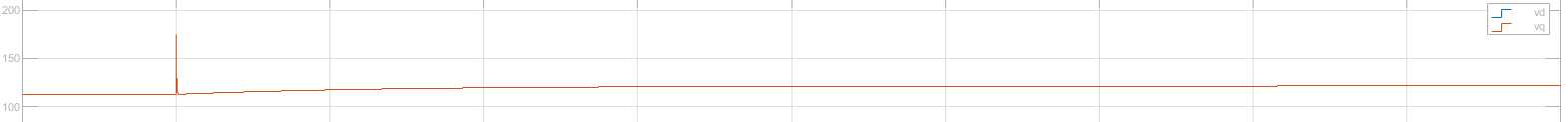
**

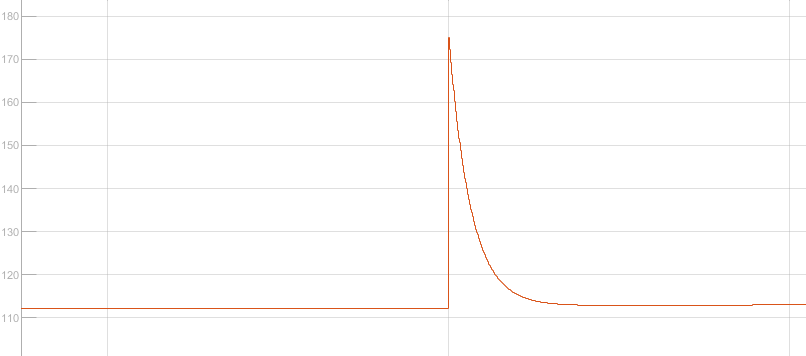
*A screenshot of a calendar

AI-generated content may be incorrect.*





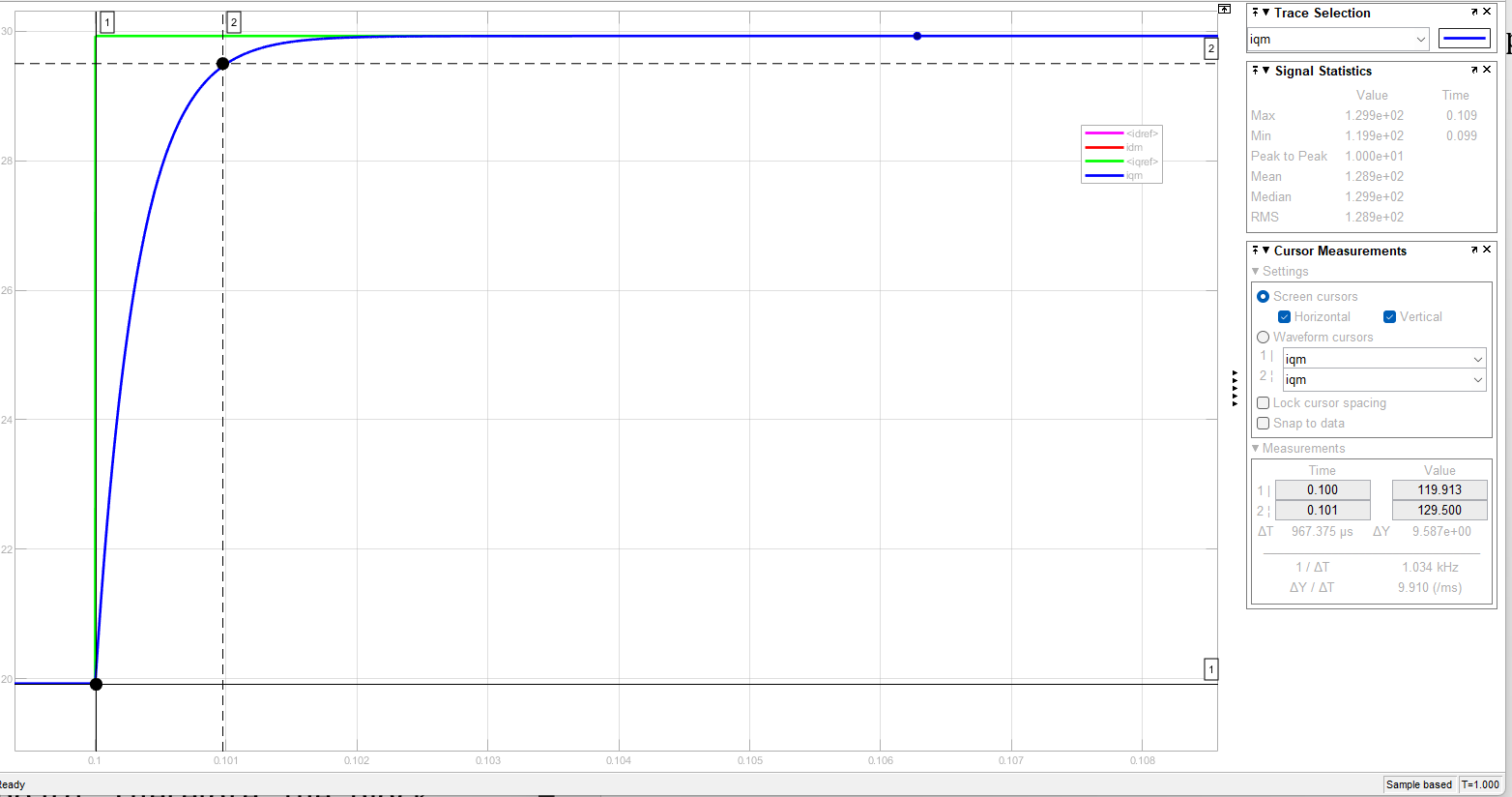




**A graph with a curve

AI-generated content may be incorrect.**

In the previous section,  *.*  The following pictures shows the response time of :

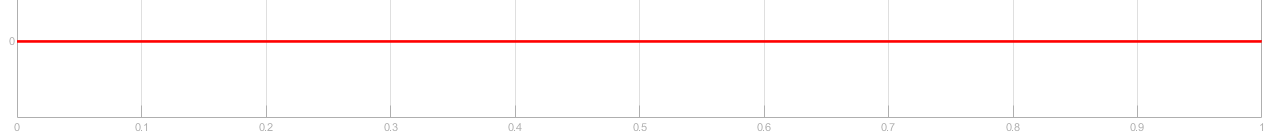


By using cursor to capture the time difference between 0% and 95% of the step change magnitude, the time difference is **967.375**  which equates to **0.967** . The error between theoretical value and simulation value is **1.36%** which include some measurement error. It can prove that the simulation match the implemented before. In conclusion, the simulation results successfully verify the intended rise time performance. This confirms that the control design methodology, including PI parameter tuning and bandwidth selection, has been effectively implemented.

For , the simulation result always keep 0 during the entire simulation period. The important reason is because there is no . Furthermore, the simulation results shows the decoupling term is correctly compensating for the effect of . In classic PMSM model, and can be represented as the following:

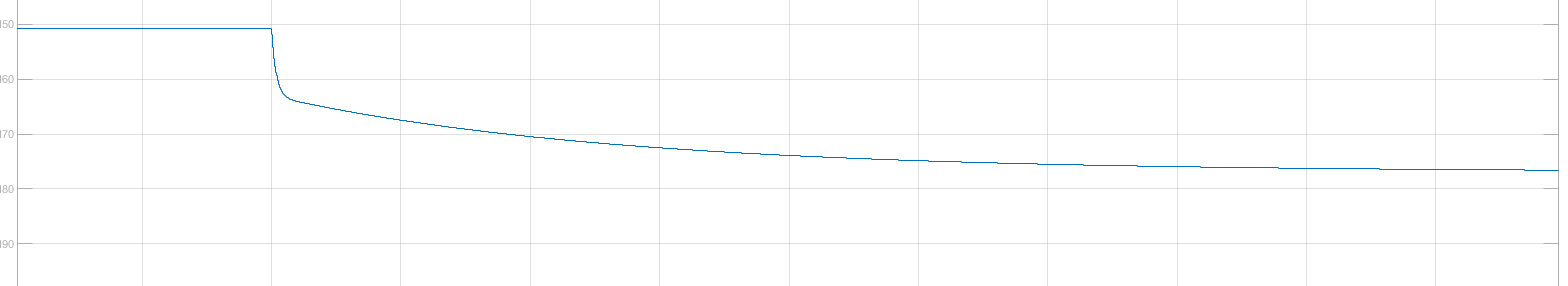
Before any step change for , is mainly controlled by therefore keep in a stable value. At change which cause sudden decrease because the whole system try to response the sudden change for . After the becomes stable, the behavior becomes stable again. Similar explanation can be applied to Before step change, is controlled by . After applied 10A current increase. starts to contribute to the system. That is the reason why there is a spike in the graph. Finally it decreases to a steady state due to the effect of controller to maintain the to a new value. For Ω, it is also expected that it keep increasing due to the current increase but it response time is much slower than electric parameters. That is due to the natural characteristics for mechanical parameters.

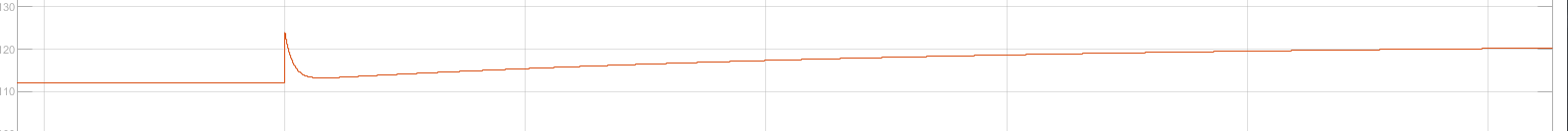
## Part II Question 5



A graph with a line drawn on it

AI-generated content may be incorrect.



**

A graph with a curve

AI-generated content may be incorrect.

In this question,  *.*  The following pictures shows the response time of :

A graph with lines and dots

AI-generated content may be incorrect.

By using cursor to capture the time difference between 0% and 95% of the step change magnitude, the time difference is **5.047** . The error between theoretical value and simulation value can be ignored because of some measurement error. It can prove that the simulation match the implemented before. In conclusion, the simulation results successfully verify the intended rise time performance. This confirms that the control design methodology, including PI parameter tuning and bandwidth selection, has been effectively implemented. The behavior of match the previous results and reason behind that are explained before.

## Part II Question 6

**(Step 4)**

A red line on a white background

AI-generated content may be incorrect.

A red line with a line drawn on it

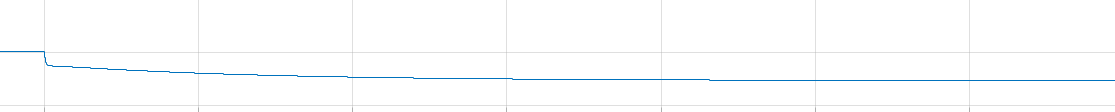
AI-generated content may be incorrect.

**(Step 4)**

A screenshot of a computer

AI-generated content may be incorrect.

**(Step 4)**

****

**(Step 4)**

A white grid with red lines

AI-generated content may be incorrect.

A graph with a curve

AI-generated content may be incorrect.

In previous section, . The following pictures shows the response time of :

A white grid with a line in the middle

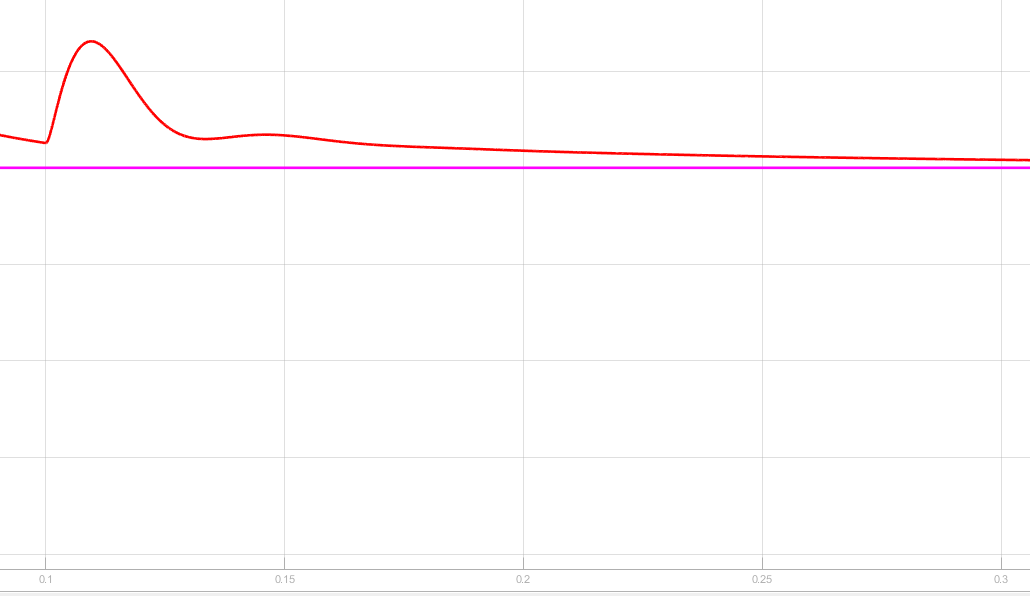
AI-generated content may be incorrect.

The result shows the response time is slower than the ideal situation that there is no error among parameters. Secondly, there is a overshoot in this simulation. Furthermore, value is not keep zero anymore, there is a sudden rise at and back to 0 gradually after system is stable. The reason can be explained through mathematics behind controller design. In the standard Pole-Zero Compensation method, the open-loop transfer function of the current controller is the following:

This set-up ensures that the closed loop transfer function is a first order system. As a result, there is no overshoot in the previous simulation and response time matches in the design parameters. Decoupling also work properly. In this section, parameters has errors:

As a result, . Zero introduced by the controller does not cancel the plant pole. This transforms the closed-loop system from a first-order to a second-order system. That is the reason why there an overshoot and slower response time. The behavior of is also expected because of the error the decoupling failed. That is the reason why when change, also changed. This is a classic coupling effect. The behavior of match the previous results and reason behind that are explained before. In conclusion, introducing the error to the parameters matches the behavior of PI controller.

**(Step 5)**



**(Step 5)**

A graph with a line graph

AI-generated content may be incorrect.

**(Step 5)**

A screenshot of a computer

AI-generated content may be incorrect.

**(Step 5)**

**A screenshot of a computer

AI-generated content may be incorrect.**

A graph with a line

AI-generated content may be incorrect.

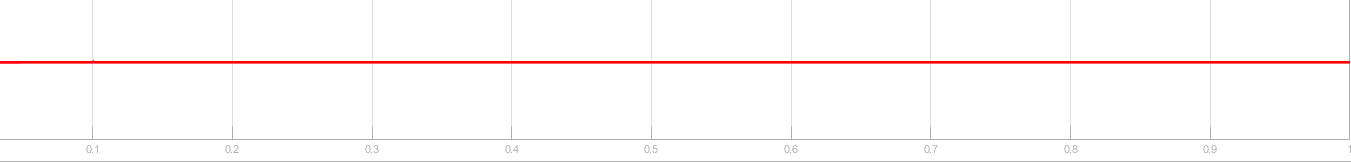
The graphs above are the system behavior for after introducing parameter errors. The overshoot for becomes worse and the following is the performance of the response time:

A graph with a line

AI-generated content may be incorrect.

From the response time point of view, the performance for this controller is not good. It is almost 9 times slower than the design values. These behaviours are expected because is increased. The direct consequence is that the bandwidth decreases. The controller gains also decreases therefore cause larger overshoot and slower response time. The decoupling effect becomes worse as well. The behavior of match the previous results and reason behind that are explained before.

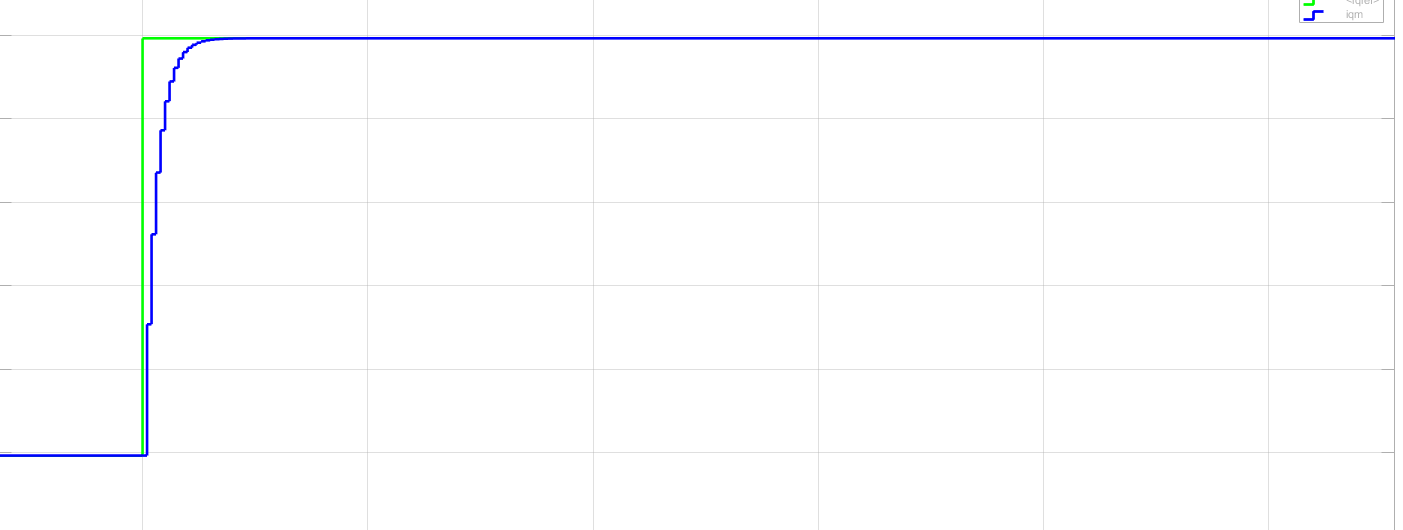
## Part III Question 1

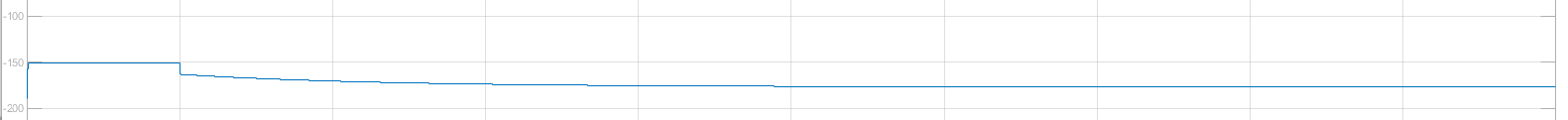


A graph with a line drawn on it

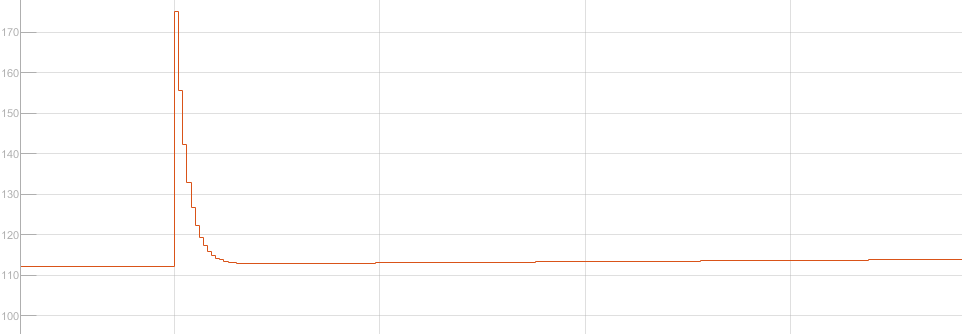
AI-generated content may be incorrect.

**(Step 5)**



****

****

****

**A graph with a curve

AI-generated content may be incorrect.**

Since the , the . The response time for discrete implementation is the following:

A screenshot of a computer

AI-generated content may be incorrect.

From the set-point tracking point of view, the controller performance is similar to the continuous controller performance. There is no overshoot in discrete one and the response time is almost the same of the continuous controller. This suggests that discretization effects have altered the system dynamics, leading to a response that differs from the continuous-time case. do have some coupling effect which is not appear in the continuous time domain controller. This is because the discrete implementation, the decoupling terms may not be applied with the same precision as in continuous-time control. This results in transient coupling. Different from the previous section, and have some value at the initial which means the controller has some initial values in the parameter set cause the calculation off at the first time step. Ω behavior is the same as continuous implementation which is expected.

# Appendix

## Question 1

A screenshot of a computer program

AI-generated content may be incorrect.