# EE 406
# Laboratory of Feedback Control Systems

## Experiment 7
## Position Control in Discrete Time

May 2022

# 1  Objectives

In this laboratory session, you will become familiar with the fundamentals of discrete time control system design. The challenge of this lab is to control the position of the IP02 linear motion servo plant with specified control performance by making use various discrete time controllers.

At the end of this lab, you are expected to

- comprehend the differences between various continuous to discrete transformation methods,

- analyze the effect of sampling period on the system response,

- to be able to design and implement a deadbeat controller.

# 2  Prerequisites

To successfully carry out this laboratory, the prerequisites are

- to be familiar with your IP02 main components (e.g., actuator, sensors), your power amplifier (e.g., UPM), and your data acquisition card (e.g., Q2 or Q8), as described in References [1], [2], [3], and [4],

- to have successfully completed the Experiment #2. In this experiment, we will make use of the controller that was previously designed during Experiment #2,

- to be familiar with the complete wiring of your IP02 servo plant, as per dictated in References [1],

- to be familiar with the fundamentals of discrete time controller design.

# 3  References

[1] Experiment #1: Control Hardware and Software Setup, Signal Interfaces
[2] IP02 User Manual.
[3] DAQ User Manual.
[4] Universal Power Module User Manual
[5] QuaRC Installation Guide.
[6] QuaRC User Manual (type doc quarc in Matlab to access).
[7] Experiment #2: Proportional Derivative Position Control
[8] G.F. Franklin and J. D. Powell, *Digital control of dynamic systems*. Reading, Mass.: Addison-Wesley Pub. Co., 1980.
[9] K. Ogata, *Discrete-time control systems, 2nd edition*. Englewood Cliffs, N.J.: Prentice-Hall, 1987.

# 4   Experimental Setup

## 4.1   Main Components

This laboratory is composed of the following hardware and software components:

- **Power Module:** Quanser UPM 1503 or UPM 2405

- **Data Acquisition Board:** Quanser Q2 or Q8

- **Linear Motion Servo Plant:** Quanser IP02

- **Real Time Control Software:** The QuaRC-Simulink configuration, as detailed in the References [5].

- **A Computer to Run Matlab-Simulink and the QuaRC software**

The setup is the same as the one in Experiment #2.

## 4.2   Wirings

To wire up the system, please follow the default wiring procedure for your IP02 as fully described in References [1]. When you are confident with your connections, you can power up the UPM. (If you are uncertain of any parts of the hardware, seek help from your lab assistant.)

# 5   Controller Design Specifications

We have already designed and implemented a continuous time controller based on Proportional-Velocity (PV) controller scheme which is a modified implementation of Proportional-Derivative (PD) control approach. The following requirements of IP02 closed-loop system were satisfied under the control of this controller.

i)   The percent overshoot was less than 10%,

$$PO \leq 10\%.$$

ii)   The time to first peak was 150 ms,

$$t_p = 0.15 \text{ sec}$$

ii)   The steady state error to unit step was zero,

$$e_{ss} = 0.$$

In the present laboratory, you will firstly build discrete time controllers with various sampling frequencies considering the PD controller whose parameters were previously acquired.

Subsequently, you will design and implement a deadbeat controller. You should revise your EE402 notes to remember deadbeat design procedure.

Student Name:
Student ID:

# 6 Preliminary Work

## 6.1 Transformations from s plane to z plane

Transfer functions in s domain can be transformed into z domain using different techniques. Here, we will be dealing with backward rectangular rule, forward rectangular rule, Tustin's method (bilinear transformation) and zero-pole matching.

The first three transformation rules are derived from the numerical integration methods. A transfer function in s domain describes a differential equation in continuous time and this differential equation can be solved by numerical integration methods. For example, suppose that we have the following transfer function

$$\frac{C(s)}{R(s)} = \frac{b}{s+a},\tag{1}$$

which defines the differential equation in (2).

$$\dot{c} + ac = br\tag{2}$$

This equation can be solved for $c$ by the following integration.

$$c(t) = \int_0^t [br(\tau) - ac(\tau)]d\tau\tag{3}$$

Assuming fixed-step size of $T$, the integration can be approximated using various numerical approaches.

$$c(kT) = \int_0^{k(T-1)} [br(\tau) - ac(\tau)]d\tau + \int_{k(T-1)}^{kT} [br(\tau) - ac(\tau)]d\tau\tag{4}$$

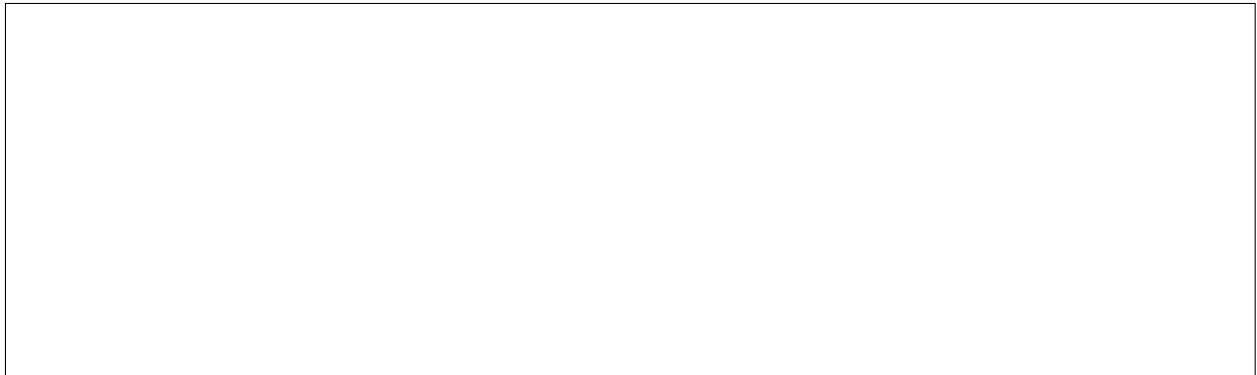$$= c\big((k-1)T\big) + \int_{k(T-1)}^{kT} [br(\tau) - ac(\tau)]d\tau\tag{5}$$

The second term on the right-hand side of (5) can be computed using one of the following methods

- Backward rectangular rule,

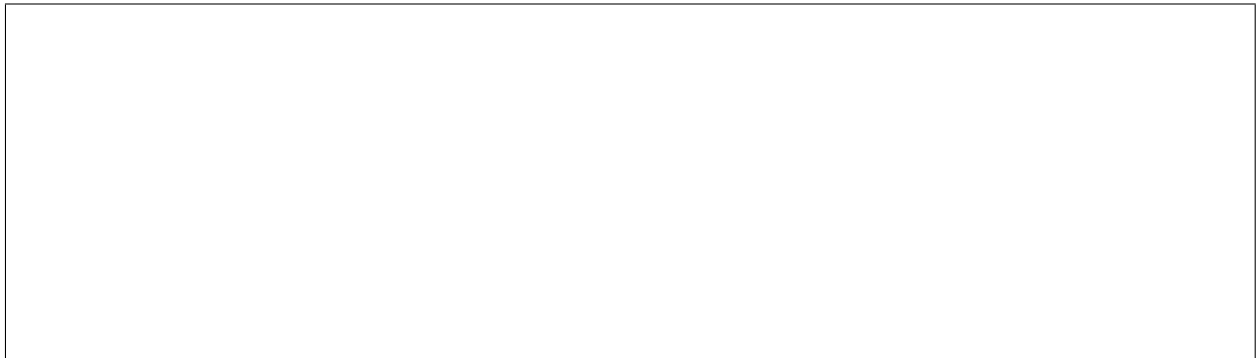- Forward rectangular rule,

- Trapezoidal rule.

| Table 1: Transformation methods | |
| --- | --- |
| Rule | Approximation |
| Backward rectangular rule | $\dfrac{z-1}{Tz}$ |
| Forward rectangular rule | $\dfrac{z-1}{T}$ |
| Trapezoidal rule | $\dfrac{z-1}{Tz}$ |

Using these methods, you may solve the numerical integration problem yourself, and then, you should be able to observe that these rules result in different transformations from s plane to z plane which are summarized in Table 1 as substitutions of approximate expressions for the variable *s*. Note that the trapezoidal rule is also known as Tustin's method or bilinear transformation. For detailed information, you can read Chapter 3, pages 53-66 of Franklin's book which is reference [8].

Using three methods explained above, transform the left-half of s plane to z plane. Plot your results indicating especially the mapping for *s = jw* axis.

Compare your results commenting on any possible advantages or disadvantages of these approaches. Which one would you prefer? Why?

Another method that we will cover in this experiment is pole-zero matching. In this approach, discrete time equivalent of a continuous time system is achieved by mapping all of the poles and zeros of the system using the following equality:

$$z = e^{Ts}, \qquad\qquad (6)$$

where $T$ is the sampling period.

Additionally, one may transform the system using hold equivalents. If a zero-order hold device is employed, then the equivalent discrete time system is computed using (7).

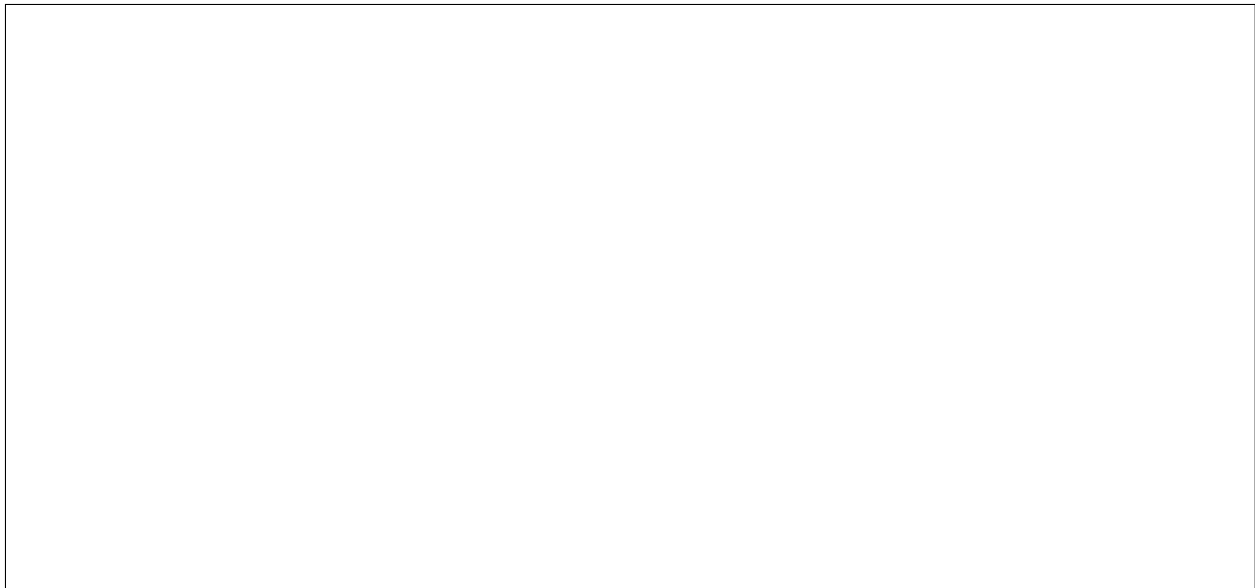$$H(z) = (1 - z^{-1})Z\left\{\frac{H(s)}{s}\right\}$$

In previous experiments, we derived the transfer function of the plant as in (8).

$$H(s) = \frac{2.45}{s(s+17.13)} \qquad\qquad (8)$$

Obtain the Bode plot of this transfer function by writing a MATLAB script. Then on the same figure, plot Bode plots of discrete equivalents of the system when transformation is achieved by zero order hold, Tustin's method, and pole-zero matching for sampling period of $T = 0.001$ sec.

Repeat the above step for $T = 0.01$ sec, $T = 0.1$ sec and $T = 1$ sec. Take the printouts of your results and attach them to end of your preliminary work.

Comment on your results.

7

## 6.2 Transforming the Continuous Time Controller using Different Sampling Periods

In Experiment #2, a PV controller that satisfies the design requirements was achieved, and the corresponding gains were attained as

$$K_p = 275.6, \quad K_v = 5.55. \tag{9}$$

In that experiment, a PV controller was preferred instead of a PD controller for computational purposes. Since they yield to the same characteristic equation for the closed loop transfer function, we will now implement a PD controller using the same gain values.

$$G_c(s) = K_p + K_d s, \tag{10}$$

where $K_p = 275.6, \quad K_d = 5.55.$

Transform this transfer function in (10) into z plane analytically using zero-order hold method for sampling frequencies $f = 500\text{Hz}$, $f = 100\text{Hz}$, $f = 50\text{Hz}$ and $f = 10\text{Hz}$.

### 6.2.1 Building Hybrid Structured Simulink Models

The Simulink model that is created to simulate the behavior of the discrete controllers would have a hybrid structure; since the plant would be represented by a *continuous transfer function*, while the controller should be realized by a *discrete transfer function*. Therefore, we need to make use of particular Simulink blocks such as *Zero Order Hold (ZOH)* and *Rate Transition* to enable reliable data transfer between continuous time and discrete time systems.

1. Examine closely the documentation for *ZOH* and *Rate Transition* blocks in Simulink.

2. Discuss the functionality of each block and indicate the discrepancies between formal purpose of usage which is stated in the documentation and your expectations from your theoretical background.
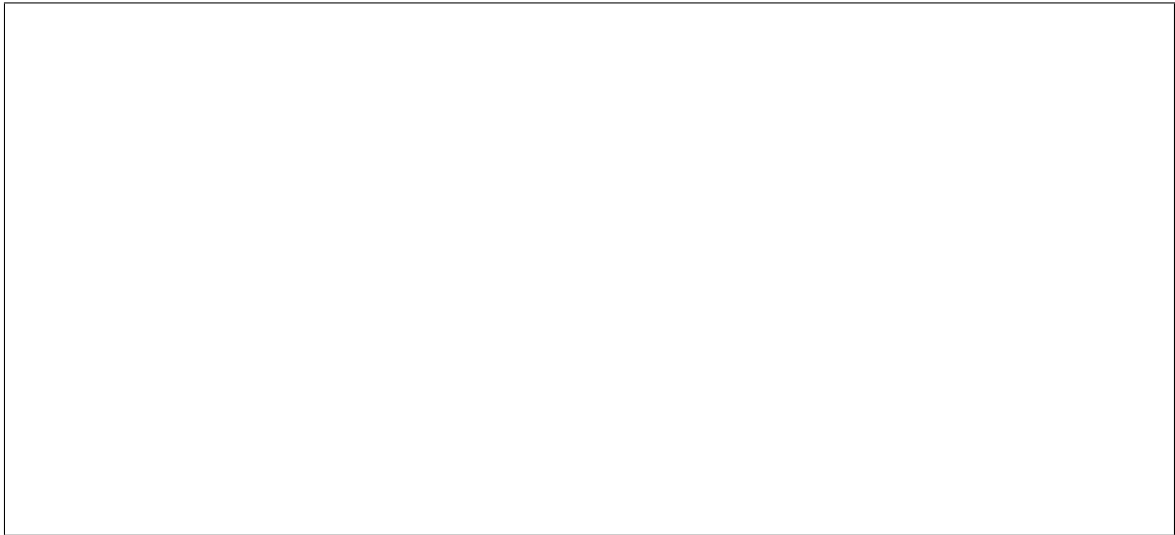
### 6.2.2 Simulating Discrete Time PD Controllers

At this stage, we will implement a Simulink model to simulate the control loop utilizing discrete time PD controller scheme and investigate corresponding performances of the controllers with different sampling frequencies. Your model should look like the one depicted in Figure 1. Throughout the implementation of the model, you should be careful about the following points.

- The solver of the model is to operate at 1 kHz. Therefore, the type of the solver is to be selected as *fixed-step-size*. Please check *Model Configuration Parameters* pane.

- The sample time for the discrete time blocks such as ZOH should be adjusted in accordance with the sampling frequency of the discrete time controller you investigate.

- The sample time of *Rate Transition* blocks is to be same with the fundamental sample time of the model.

- You can use a *Saturation* block to limit the output of the controller considering the limits of the UPM which are ±13V.

- You are free to use any suitable discrete time block to represent the discrete time controller, but usage of *Discrete Transfer Function* is suggested as illustrated in Figure 1.

- You are encouraged to use *Goto* and *From* blocks to render your model more coherent and organized.

- The position command should be a square wave with 100 mm peak-to-peak amplitude and a period of 4 seconds.

- The position tracking performances of the systems obtained by all controllers, including the continuous one, are to be fed to the same scope.

**Note**: You are expected to bring your models to the lab with a flash memory. Yourmodels should have the extension '.mdl' instead of '.slx'.
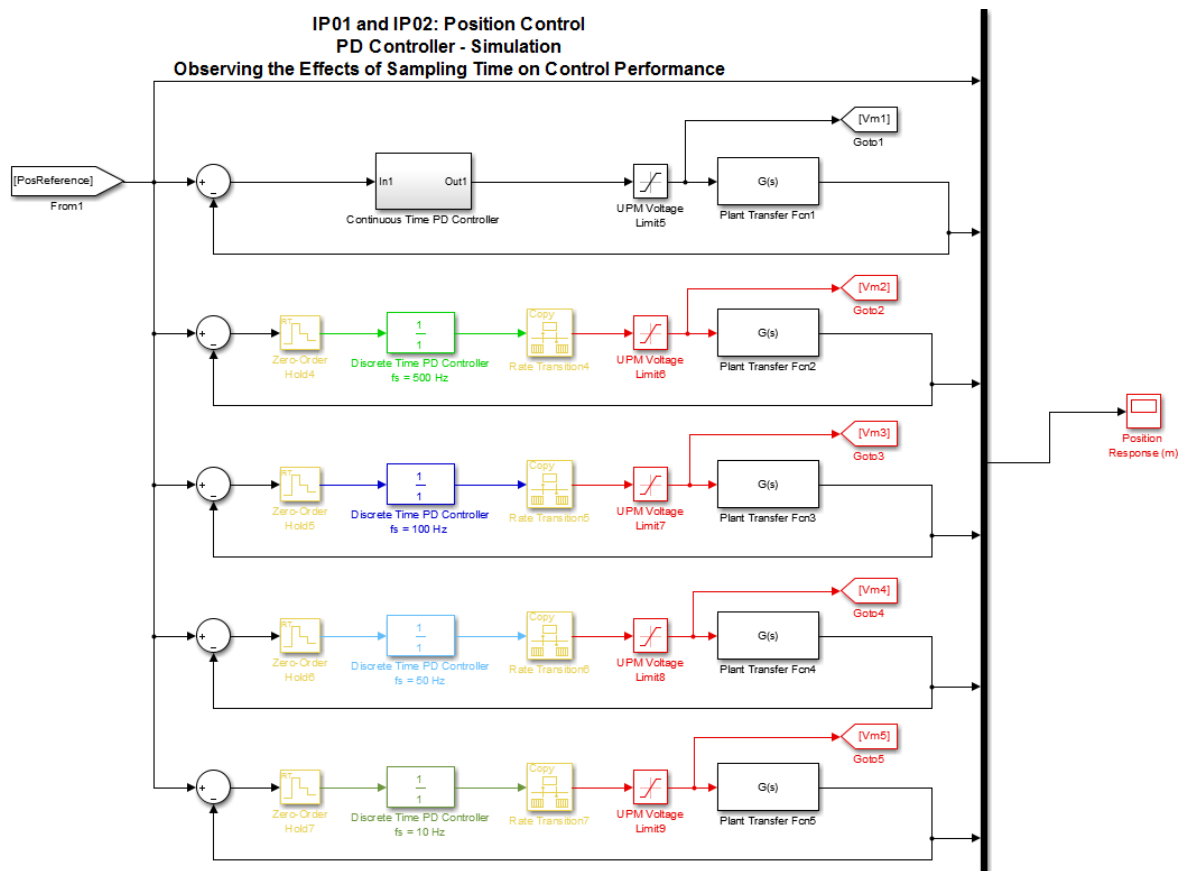


Figure 1: Simulink model to observe the effect of sampling time for discrete time control

Provide your simulation results by attaching them to your preliminary work. Comment on the simulation results comparing the effect of sampling frequency.

## 6.3   Deadbeat Controller Design

Deadbeat controller is an optimal discrete time controller that drives the output of the system to the desired constant value with zero steady state error at minimum number of steps. For an $N^{th}$ order system, it takes $N$ number of steps at most to settle. You are expected to go over *Section 4.7 - Analytical Design Method* in Discrete-Time Control Systems, Ogata. You will benefit a lot from the examples given in this section considering the rest of the Preliminary Work.

**Note:** In this part, represent your final results in the following form.

$$D(z) = K \frac{1 - z_0 z^{-1}}{1 - p_0 z^{-1}}$$

Design a deadbeat controller for a unit step input with 10 milliseconds of sampling time in order to control the position of the cart, i.e., $T_{controller}$ = 10 *msec*. Use ZOH equivalent of the transfer function of the plant.

Design another deadbeat controller for a unit step input with 250 milliseconds of sampling time, i.e., $T_{controller} = 250$ *msec*. Use ZOH equivalent of the transfer function of the plant.
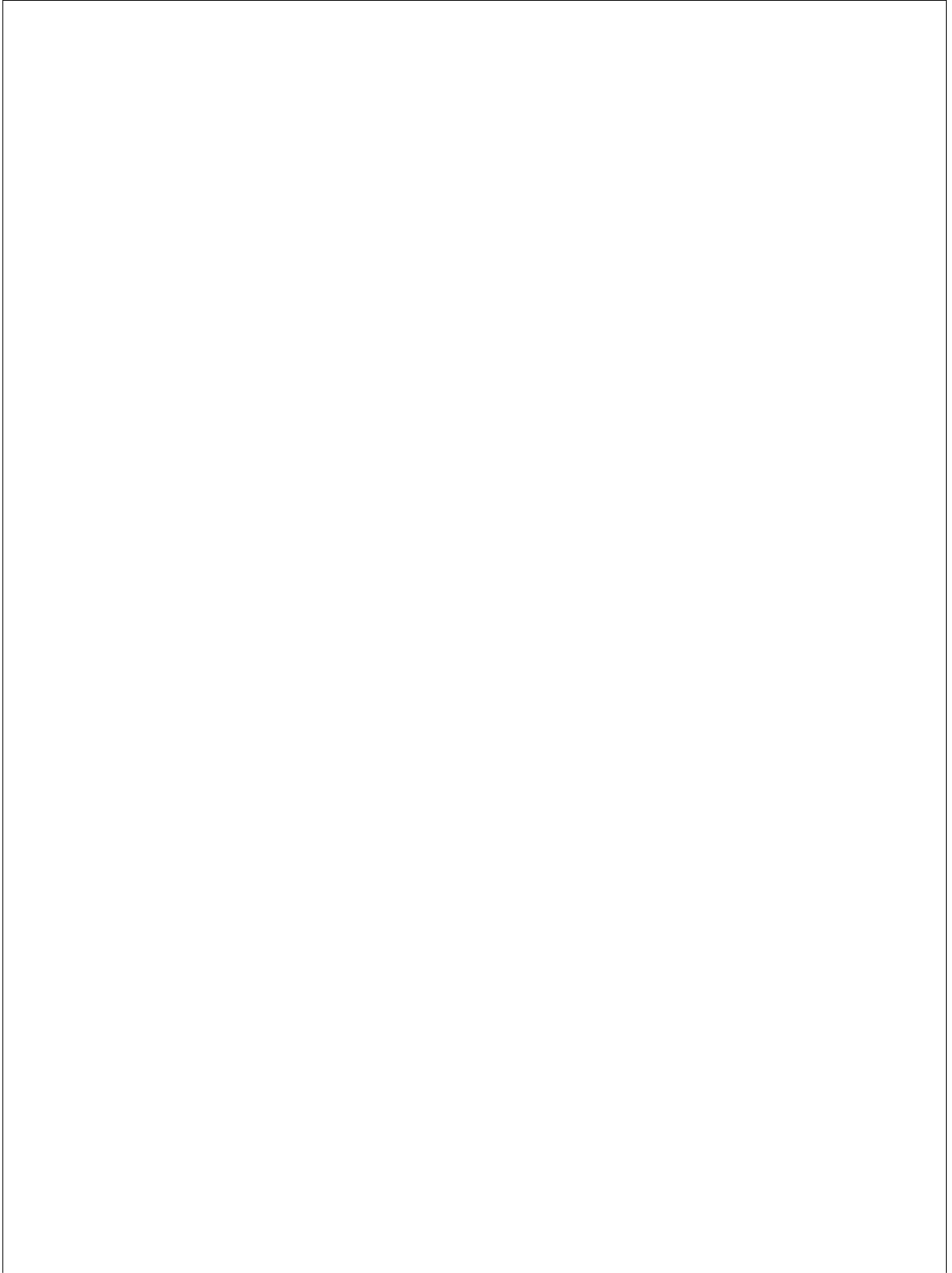
# 7  In-Lab Experimental Procedure

Before starting the experimental procedure, have your assistants check your discrete time equivalent of PD controller and the deadbeat controller you designed for $f_s$ = 100$Hz$ and $f_s$ = 4$Hz$.

Discrete PD controller, Deadbeat Controllers for $f_s$ = 100$Hz$ and $f_s$ = 4$Hz$:

Assistant's Signature

## 7.1  Experimental Setup

### 7.1.1  Check Wiring and Connections

Ensure that the complete system is wired as fully described in Reference [1]. If you are unsure of the wiring, please ask for assistance from the Teaching Assistant. When you are confident with your connections, you can power up the UPM. You are now ready to begin the lab.

### 7.1.2  IP02 Configuration

This experiment is designed for an IP02 cart without the extra weight on it. If the extra weight is mounted on top of the cart, please remove it.

## 7.2  Simulation of the Discrete PD Controllers

### 7.2.1  Objectives

- To simulate with a Simulink diagram your IP02 cart and to control the position of the cart by a discrete time PD controller.

- To observe the effect of sampling frequency of the controller on system response during simulation

## 7.2.2 Experimental Procedure

1. Start-up Matlab and open your Simulink model that you constructed in your preliminary work.

2. Before proceeding further, **have your assistant check your model**.

3. Set the input signal to square wave signal of amplitude 50 mm and period of 4 seconds.

4. Check that if the fixed-step-size (fundamental sample time) of the solver is set to $0.001$ sec.

5. Adjust the sample times of ZOH and the discrete time controller blocks in accordance with the sampling frequency you investigate.

6. Run the simulation. Bring up the scope displaying position response.

7. Sketch the position responses for all sampling frequencies using different line styles to distinguish them from each other.
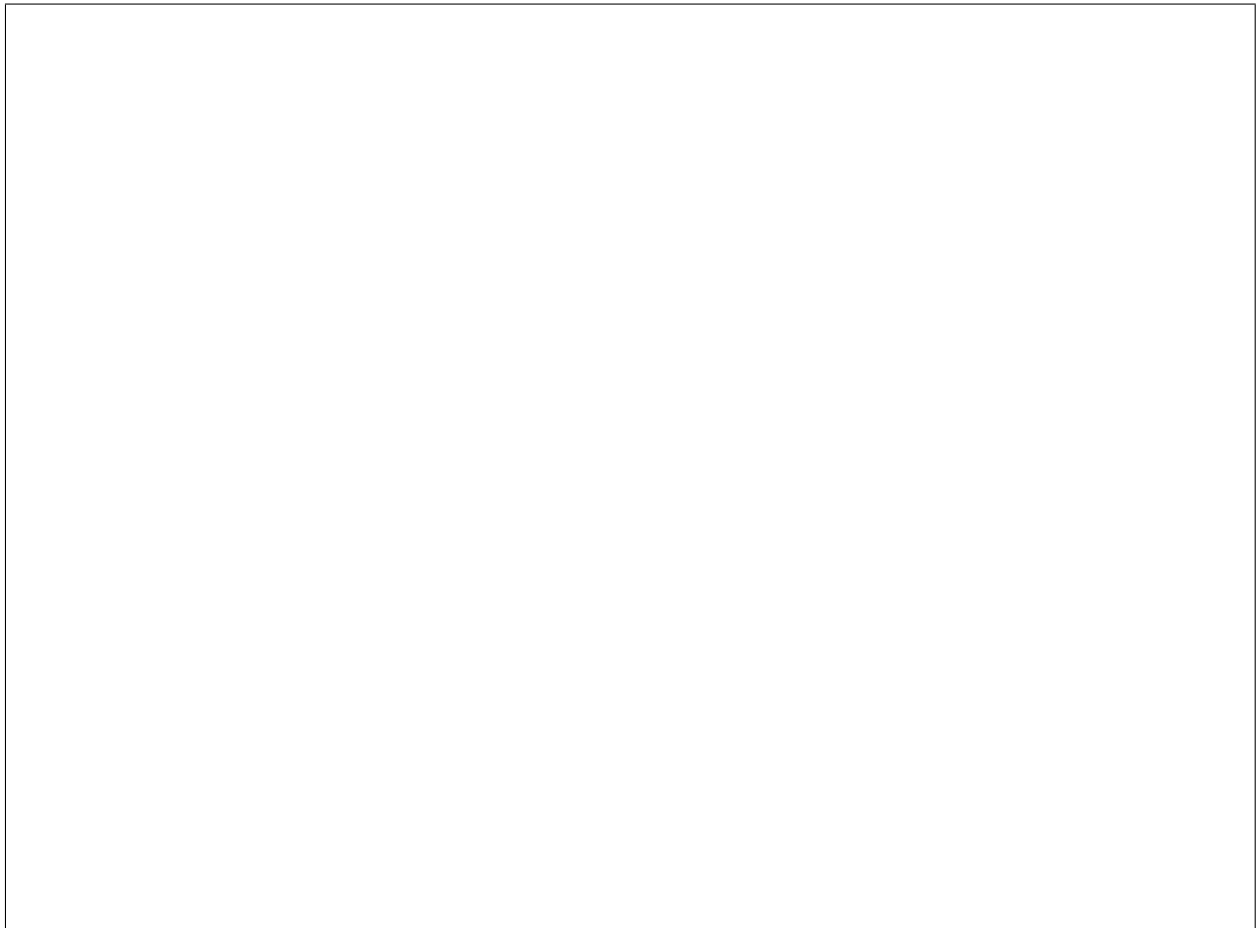
8. Compute the percent overshoot and the settling time for each sampling frequency.

| $f_{controller}$ | PO | $t_s$ |
|---|---|---|
| 500 Hz | | |
| 100 Hz | | |
| 50 Hz | | |
| 10 Hz | | |

9. Comment on your results. How does the performance of the system change when the sampling frequency of the controller is reduced? Which discrete time equivalent represents the continuous time controller better?

## 7.3 Real-Time Implementation of Discrete Time PD Controllers

### 7.3.1 Objectives

- To implement with QuaRC real-time environment, the previously designed discrete time PD controllers in order to command your IP02 servo plant.

- To run the simulation simultaneously, at every sampling period, in order to compare the actual and simulated responses.

### 7.3.2 Experimental Procedure

You can now implement your discrete controllers to observe sampling frequency effects on real-time. Please follow the steps described below:

1. Run the Matlab script called `setup_lab_ip01_2_Exp07.m` to initialize IP02 system parameters.

2. Open the model `q_position_pd_ip02.mdl`. This model is similar to the one used in Experiment #2. The difference is that instead of a PV controller in continuous domain, we now employ a discrete time PD controller with some additional blocks to regulate data transition between systems operating at different frequencies as illustrated in Figure 2.



**IP02 - PD Position Control:**
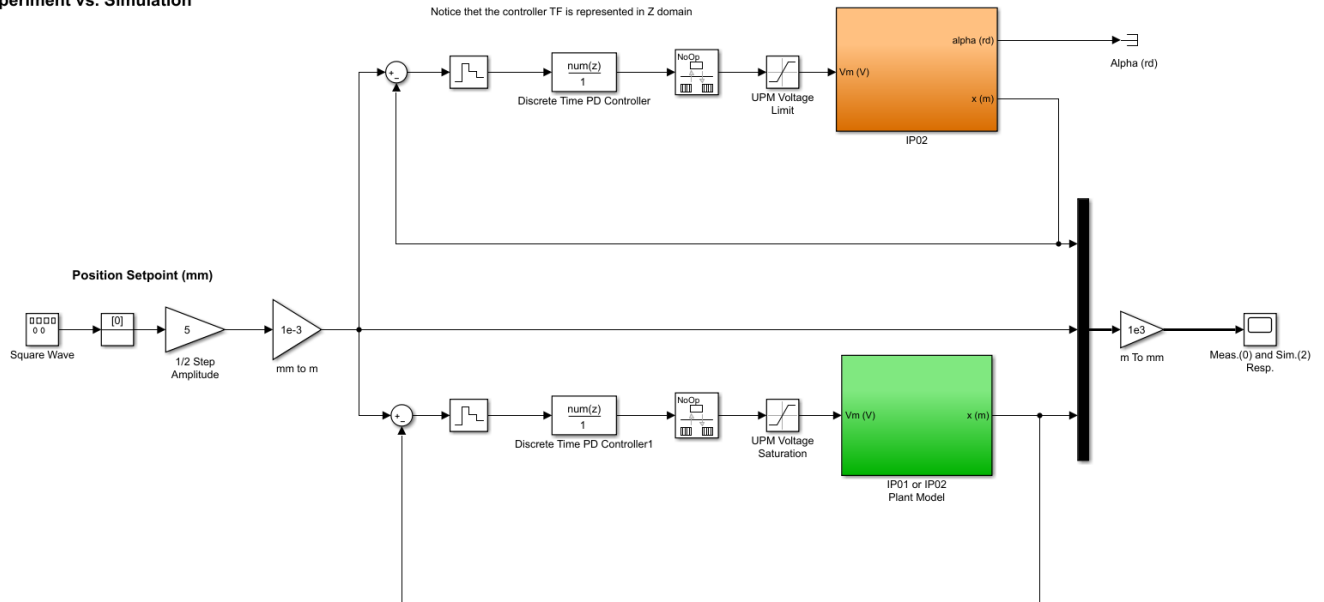**Experiment vs. Simulation**

Figure 2: Simulink diagram used for the real time implementation of the PD controller

3. Write the PD controller equivalent in discrete time that you found in your preliminary work into the *Discrete Filter* block for the sampling frequency of 500 Hz.

4. Check that the signal generating block is set to square wave with amplitude of 1 and period of 4 seconds while the following gain block is adjusted to 5 in order to have 10 mm peak-to-peak amplitude.

5. Adjust the sample times of *ZOH* and the *Discrete Filter* blocks according to the sampling frequency you investigate.

6. Configure DAQ: Double-click on the *HIL Initialize* block inside the *IP02 Actual Plant* subsystem and ensure it is configured for the DAQ device (*q2_usb or q8 usb*) that is installed in your system. See Reference [6] for more information on configuring the *HIL Initialize* block.

7. Build the real-time code corresponding to your diagram, by using the `QuaRC | Build` option from the Simulink menu bar.

17

8. Manually move your IP02 cart to the middle of the track (i.e., mid-stroke position) and make sure that it is free to move.

9. To start the control system, click on the `QuaRC | Start` from the Simulink model menu bar.

10. Open the scope `Cart Position`. You should now be able to monitor on-line, as the cart moves, the actual cart position as it tracks your pre-defined reference input signal and compare it with the simulation result produced by the IP02 model.

11. Plot the position response.
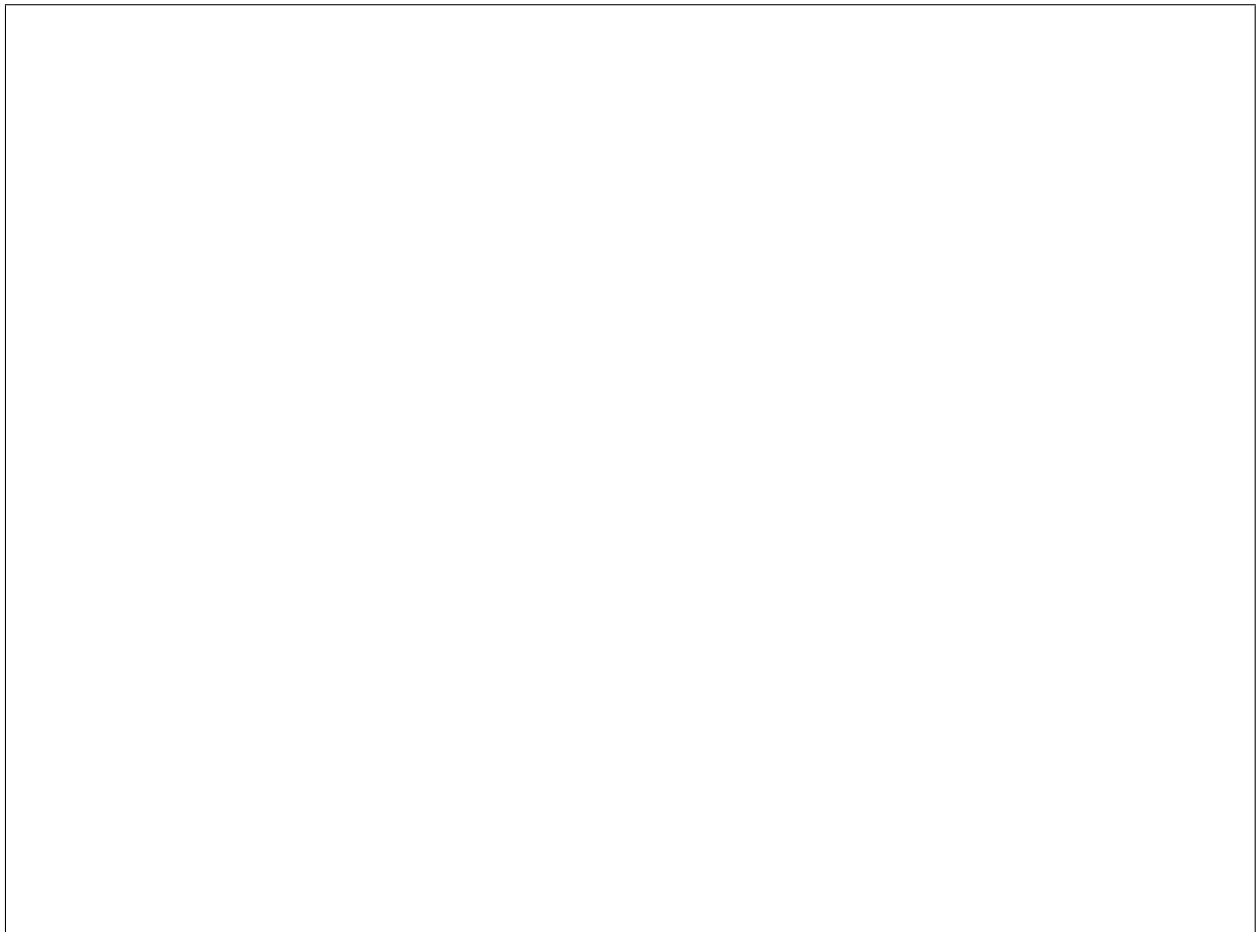
12. Repeat the steps above for

   i) $f_s$ = 100 *Hz*,

  ii) $f_s$ = 50 *Hz*,

 iii) $f_s$ = 10 *Hz*. (**Warning:** Be careful that the system may exhibit oscillations of great amplitude. Therefore, you may need to stop running the model immediately.)

Sketch of the cart position for $f_s$ = 100Hz:

Sketch of the cart position for $f_s$ = 50Hz:

Sketch of the cart position for $f_s$ = 10Hz:

13. Is there any discrepancy between the simulated and real-time results? Explain.

14. Is it possible to obtain a valid controller for any $f_s$ that satisfy the design requirements? Explain.

## 7.4  Simulation of the Deadbeat Controller

### 7.4.1  Objectives

- To simulate with a Simulink diagram your IP02 model and to close the position loop by implementing a deadbeat controller.

- To observe and compare the system responses corresponding to deadbeat controllers with different sampling frequencies.

- To investigate the effects of actuation constraints on the performance of the deadbeat controllers.

### 7.4.2   Experimental Procedure

1. Run the Matlab script called `setup_lab_ip01_2_Exp07.m` to initialize IP02 system parameters.

2. Open the model `s_position_deadbeat_ip02.mdl` which should look like the one represented in Figure 3.

3. Modify the *Gain* and *Discrete Filter* blocks to realize the deadbeat controller with sampling frequency of 100 Hz according to your design in the preliminary work.

4. Set the sample times of the *ZOH* and the *Discrete Filter* to 10 milliseconds.

5. Check that the signal generating block is set to square wave with 1 amplitude and period of 4 seconds. The gain block is adjusted to 50 in order to have 100 mm peak-to-peak amplitude.

6. Ensure that the simulation mode is set to *Normal*. Click on *Simulation |Start* from the Simulink menu bar and bring up the *Position Response (mm)* and the *Voltage Command (V)* scopes.
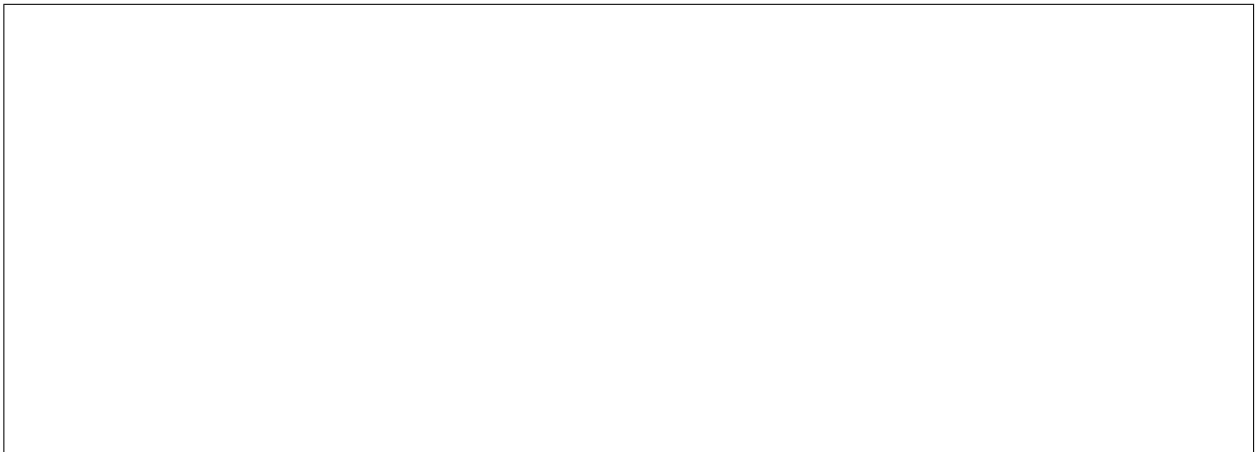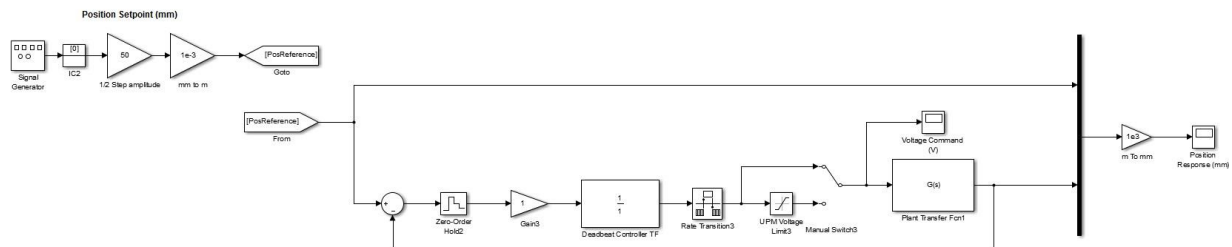


Figure 3: Simulink diagram used for the simulation of the deadbeat controller

7. Sketch the position response and the control effort.

8. Does the response follow the behavior you predicted? Explain in a comprehensive manner.

9. Is this performance physically achievable considering the limits of our experimental setup? Justify your answer.

10. At this step, click on the *Manual Switch* in the Simulink model so that the output of the controller goes through the *Saturation* block.

11. Repeat the steps 6 and 7 with this configuration.

12. How does the performance of the controller change? Can it be referred as a deadbeat performance? (Hint: Consider the settling time.)

13. Do you have a suggestion to design a deadbeat controller without getting affected by the actuation limits?

14. Modify the *Gain* block corresponding to the controller and *Discrete Filter* block to realize the deadbeat controller with sampling frequency of 4 Hz according to your design in the preliminary work.

15. Run the simulation one more time without changing the state of the *Manual Switch*.

16. Sketch the position response and the control effort.

17. Is the control effort saturated? Try to give your reasoning of why it in not saturated based upon a physical interpretation. Compare these results with the previous case (100 Hz with Saturation).
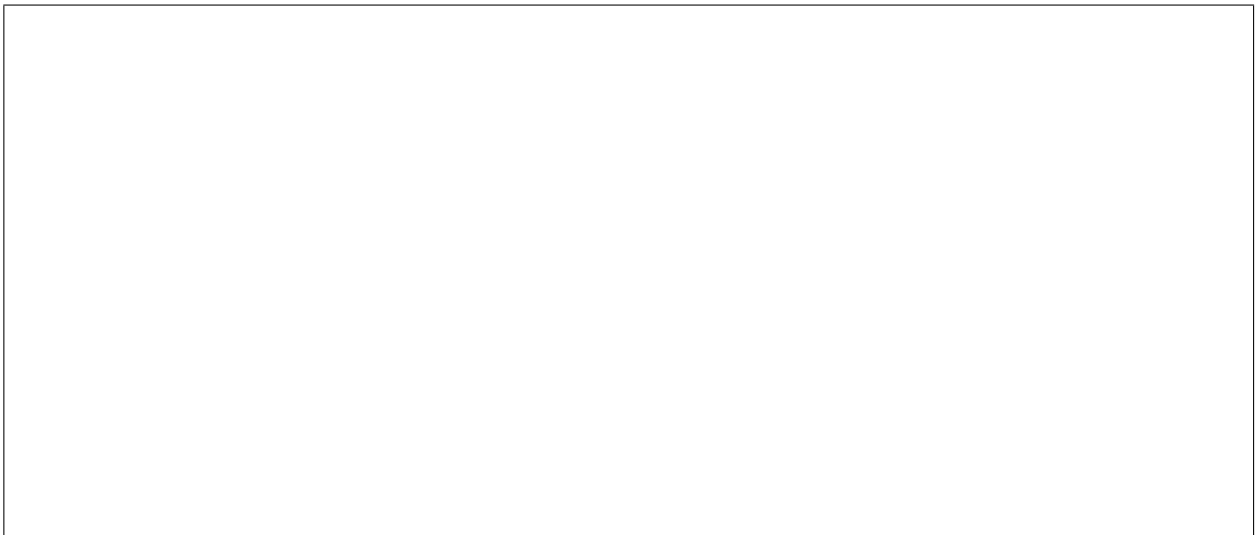
## 7.5  Real-Time Implementation of the Deadbeat Controller

### 7.5.1  Objectives

- To implement with QuaRC real-time environment, the previously designed deadbeat controller in order to command your IP02 servo plant.

- To run the simulation simultaneously, at every sampling period, in order to compare the actual and simulated responses.

### 7.5.2  Experimental Procedure

1. Run the Matlab script called `setup_lab_ip01_2_Exp07.m` to initialize IP02 system parameters.

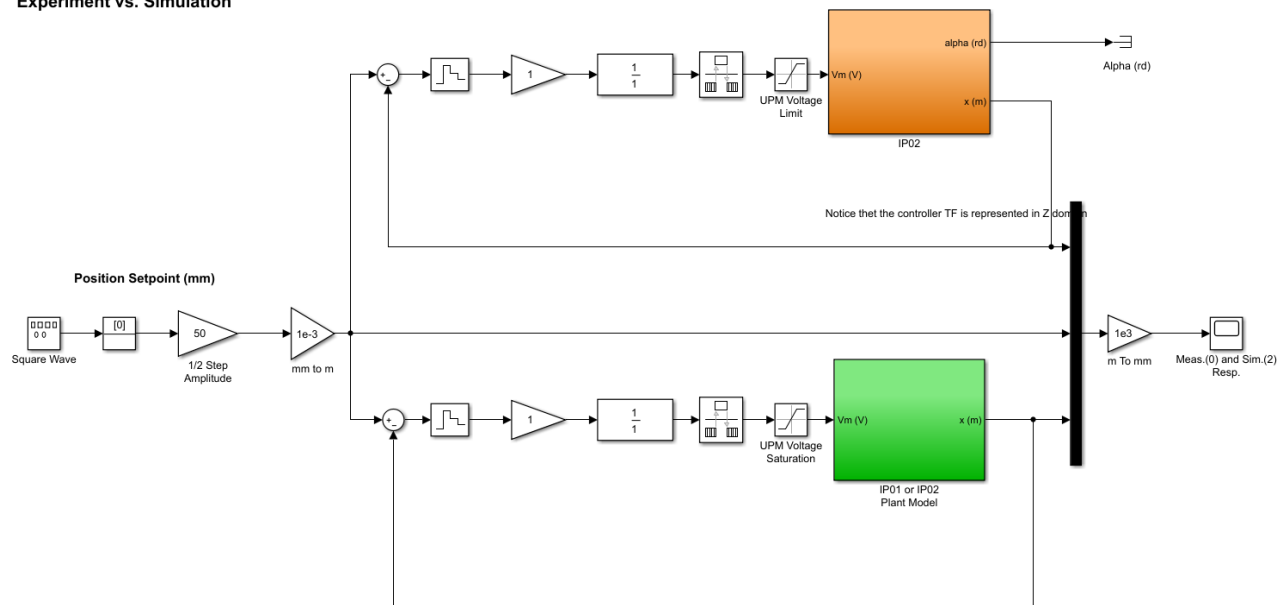2. Open the model `q_position_deadbeat_ip02.mdl` as in Figure 4.



Figure 4: Simulink diagram used for the real time implementation of the deadbeat controller

3. Modify the discrete controller block according to the deadbeat controller you designed in your preliminary work for the sampling frequency of 4 Hz.

4. Note that the model includes the *Saturation* block by default.

5. Check that the signal generating block is set to square wave with 1 amplitude and a period of 4 seconds. The gain block is adjusted to 50 in order to have 100 mm peak-to-peak amplitude.

6. Adjust the sample times of *ZOH* and the *Discrete Filter* blocks according to the sampling frequency you investigate which is 4 Hz.

7. Configure DAQ: Double-click on the *HIL Initialize* block inside the *IP02 Actual Plant* subsystem and ensure it is configured for the DAQ device (q2_usb or q8 usb) that is installed in your system. See Reference [6] for more information on configuring the *HIL Initialize* block.

8. Build the real-time code corresponding to your diagram, by using the `QuaRC | Build` option from the Simulink menu bar.

9. Manually move your IP02 cart to the middle of the track (i.e., mid-stroke position) and make sure that it is free to move.

10. To start the control system, click on the `QuaRC | Start` from the Simulink model menu bar.

11. Open the scopes `Cart Position` and `Control Effort`. You should now simultaneously observe the cart position and voltage input to the plant on-line comparingthem with the simulation results produced by the IP02 model.

12. Sketch the position response and the control effort.

13. Compare your results with the simulation results. Does the deadbeat controller characteristics are achieved? If not, explain why.

14. Did you expect the nonzero control effort after the settling time is reached? If not, why do we observe so? Explain.

# 8 Knowledge Test

1. What are the major considerations which influence the selection of sampling rate for a control scheme?

2. Give the overview of the path you take toward the design of a deadbeat controller.

3. What are the advantages and disadvantages of deadbeat control?