FH Joanneum

Graz

Model Based Design

Balanbot

Training Unit 05

Authors
David B. Heer
Jakob Soukup
Graz, January 16, 2019

Lecturer
Alfred Steinhuber

Contents

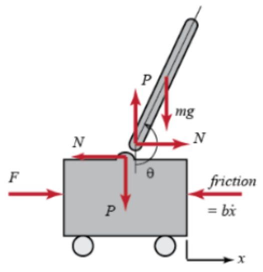
Part I
Laboratory Session 06

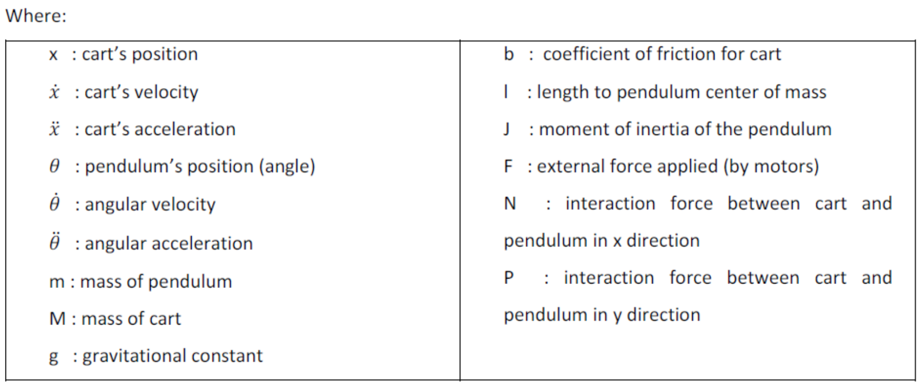
Introduction

In this laboratory unit the model of an inverse pendulum on a moving cart will be implemented snd simulated in simulink. In a first step the non linear model will be implemented and then discretized. After that the non linear model shall be linearized and discretized again. The differences between the two models are to be investigated. The two models shall be controlled with a PID controller. If the simulation works the model shall be deployed onto an actual moving robot to see if it holds up in real life.

1 Description of the Model

The model consists of a moving part with a hinged pendulum atop. The goal for the controller is to accelerate the cart in the right direction depending on the angle of the pendulum in order to keep it upright at all times.




Where:

x : cart's position	b : coefficient of friction for cart
$\dot{x}$: cart's velocity	l : length to pendulum center of mass
$\ddot{x}$: cart's acceleration	J : moment of inertia of the pendulum
θ : pendulum's position (angle)	F : external force applied (by motors)
$\dot{\theta}$: angular velocity	N : interaction force between cart and pendulum in x direction
$\ddot{\theta}$: angular acceleration	P : interaction force between cart and pendulum in y direction
m : mass of pendulum	
M : mass of cart	
g : gravitational constant	

Figure 1: *graphical description of the model*

The equations of the model are given by:

$$\ddot{x} = \frac{1}{M}\sum_{cart} F_x = \frac{1}{M}\left(F - N - b\dot{x}\right) \tag{1}$$

$$\ddot{\Theta} = \frac{1}{I}\sum_{pend} \tau = \frac{1}{I}\left(-Nlcos\Theta - Plsin\Theta\right) \tag{2}$$

$$N = m\left(\ddot{x} - l\dot{\Theta}^2 sin\Theta + l\ddot{\Theta}cos\Theta\right) \tag{3}$$

$$P = m\left(l\ddot{\Theta}^2 cos\Theta + l\ddot{\Theta}sin\Theta\right) \tag{4}$$

1.1 Implementation in simulink

The non linear model can be implemented using the equations above, this was already done in a previous lectore in the third semester. The resulting model can be seen in Figure 5.

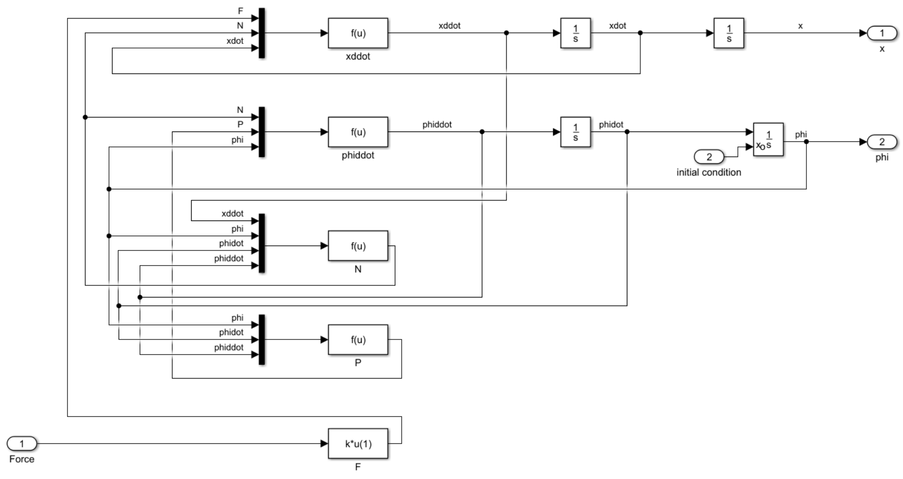
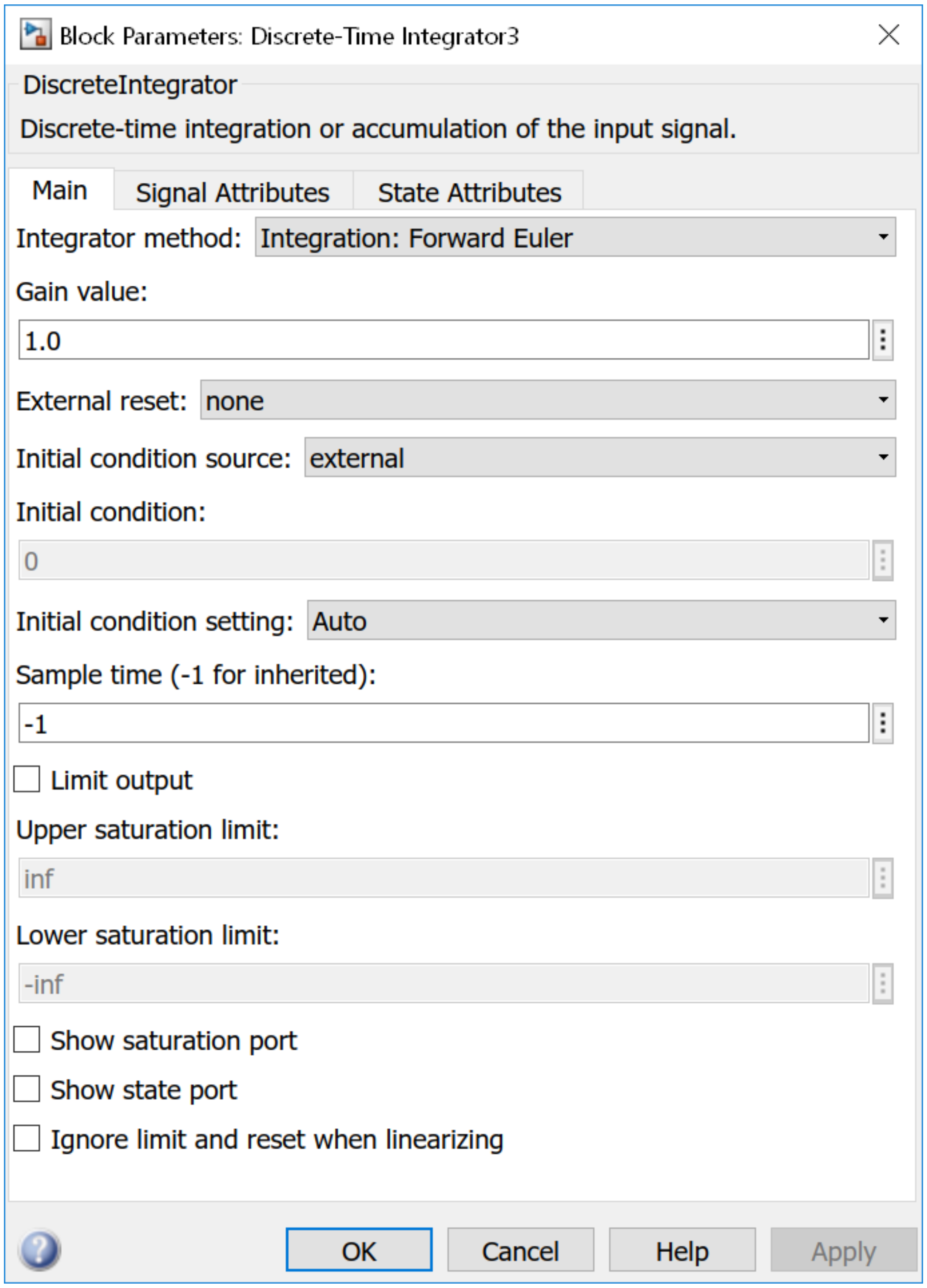


Figure 2: *Non linear continuous model in simulink*

2 Discretization from non-linear model

Since the model will later be used on an actual hardware, it is important to sicretize the system. This is done by simply replacing the continuous time integrators with discrete time integrators. The settings of the integrators are shown in Figure 3.



Block Parameters: Discrete-Time Integrator3

DiscreteIntegrator

Discrete-time integration or accumulation of the input signal.

Integrator method: Integration: Forward Euler

Gain value: 1.0

External reset: none

Initial condition source: external

Initial condition: 0

Initial condition setting: Auto

Sample time (-1 for inherited): -1

☐ Limit output

Upper saturation limit: inf

Lower saturation limit: -inf

☐ Show saturation port

☐ Show state port

☐ Ignore limit and reset when linearizing

Figure 3: *discrete time integrator settings*

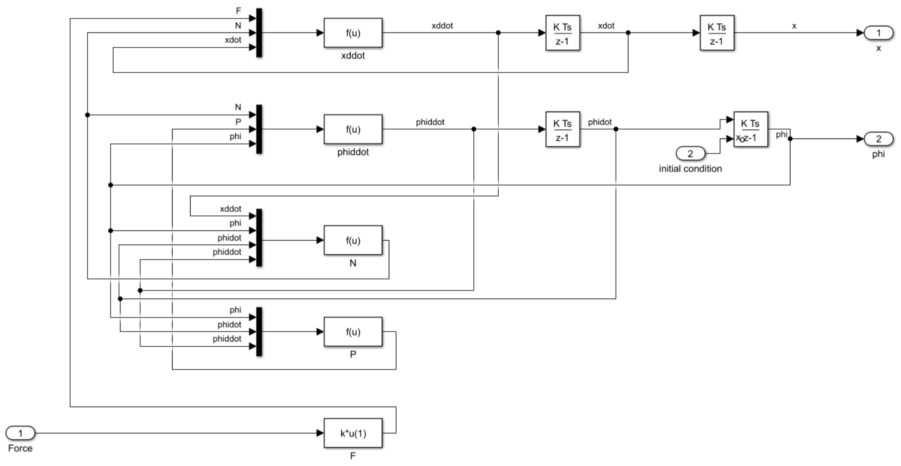


Figure 4: *Non linear discrete model in simulink*

2.1 Applying zero force to the system

As a first test the model was tested with a constant of zero at its input. It would be expected to do nothing but stay upright since there are no external forces applied to the pendulum in the horizontal axis.

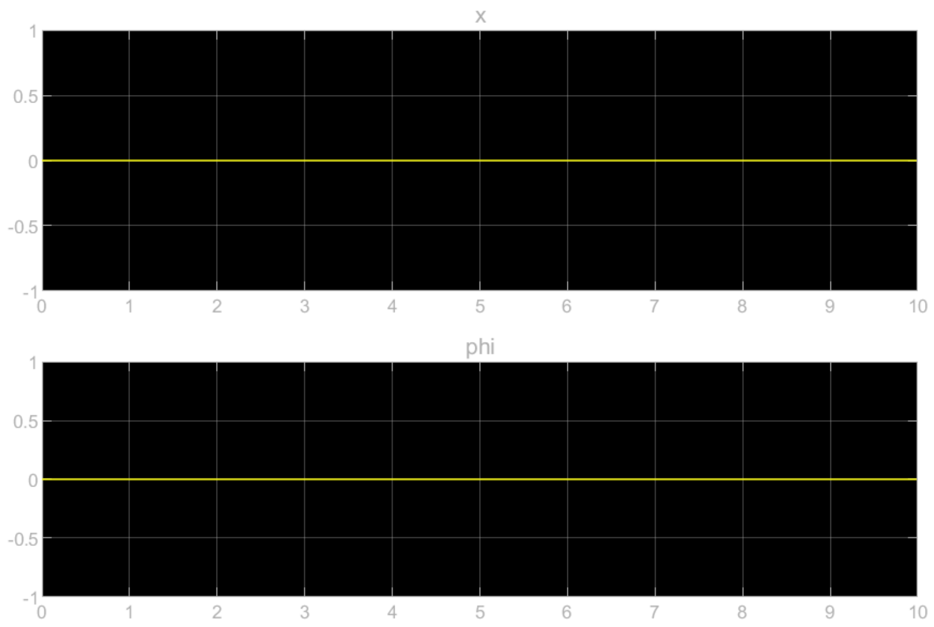


Figure 5: *Non linear discrete model simulation with zero force applied*

As a small test an initial step was applied to the system to see if it behaves correctly. The disturbance of the angle was accomplished by using the step function of simulink with an initial value of $10 \cdot \frac{\pi}{180}$, which is 10° in radians. As shown in Figure 6 the pendulum swings left and right and slowly loses height, so the model seems to be behaving correctly.

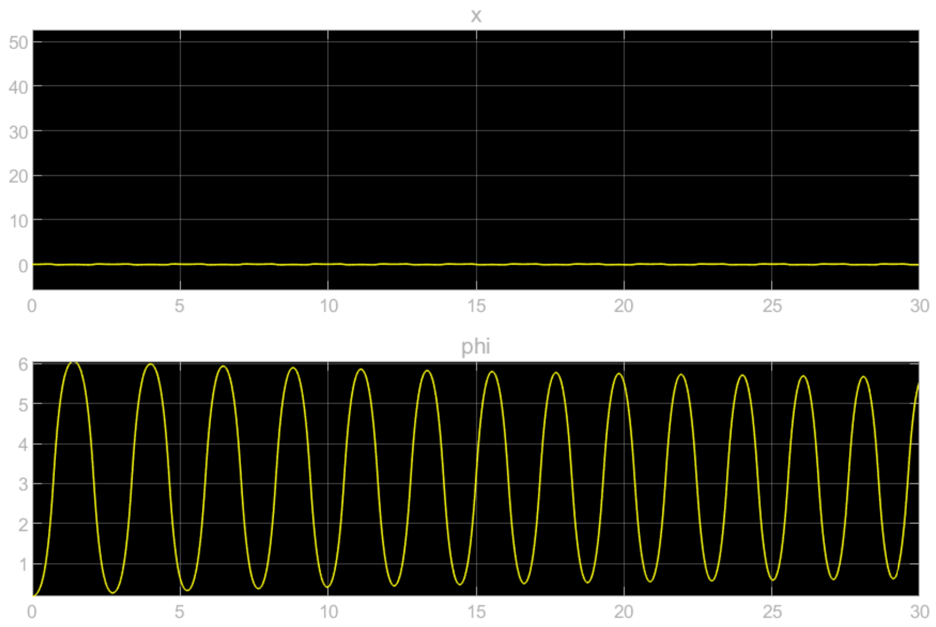


Figure 6: *Non linear discrete model simulation with an offset step in the angle*

3 Linearization

In order to further investigate the system for stability to make tuning the controller easier, it is mandatory to linearize the model. This is done by assuming that the slope of a sine wave is linear which of course is not the case but it is a valid approximation. The linearized equations are given by

$$X(s)\,s^2 = \frac{1}{M}\Big(F(s) - mX(s)\,s^2 + ml\Phi(s)\,s^2 - bX(s)\,s\Big) \tag{5}$$

$$\Phi(s)\,s^2 = \frac{1}{I}\,\Big(mlX(s)\,s^2 + mlg\Phi(s) - ml^2\Phi(s)\,s^2\Big) \tag{6}$$

From these equations the two transfer functions $G_1(s) = \frac{\Phi(s)}{F(s)}$ and $G_2(s) = \frac{X(s)}{F(s)}$ are to be found. This is simply a fact of rearranging the equations. $G_1(s)$ Solving equation 5 for $X(s)$:

$$X(s)\ s^2\ M = F(s) - mX(s)\ s^2 + ml\Phi(s)\ s^2 - bX(s)s \tag{7}$$
$$X\left(s\right)s^2M = F\left(s\right) + ml\Phi\left(s\right)s^2 + X\left(s\right)\left[-ms^2 - bs\right] \tag{8}$$
$$X\left(s\right)s^2M - X\left(s\right)\left[-ms^2 - bs\right] = F\left(s\right) + ml\Phi\left(s\right)s^2 \tag{9}$$
$$X\left(s\right)\left[s^2M + ms^2 + bs\right] = F\left(s\right) + ml\Phi\left(s\right)s^2 \tag{10}$$
$$X\left(s\right) = \frac{F\left(s\right) + ml\Phi\left(s\right)s^2}{s^2M + ms^2 + bs} \tag{11}$$

Inserting into equation 6 we get

$$\Phi\left(s\right)s^2 = \frac{1}{I}\left[mls^2 \cdot \frac{F\left(s\right) + ml\Phi\left(s\right)s^2}{s^2M + ms^2 + bs} + mlg\Phi\left(s\right) - ml^2\Phi\left(s\right)s^2\right] \tag{12}$$
$$\Phi\left(s\right)Is^2 = mls^2 \cdot \frac{F\left(s\right) + ml\Phi\left(s\right)s^2}{s^2M + ms^2 + bs} + mlg\Phi\left(s\right) - ml^2\Phi\left(s\right)s^2 \tag{13}$$
$$Is^2 = mls^2 \cdot \frac{\frac{F(s)}{\Phi(s)} + mls^2}{Ms^2 + ms^2 + bs} + mlg - ml^2s^2 \tag{14}$$
$$Is^2 = mls^2 \cdot \frac{\frac{F(s)}{\Phi(s)} + mls^2}{Ms^2 + ms^2 + bs} + mlg - ml^2s^2 \tag{15}$$
$$\frac{Is^2}{ml} = s^2 \cdot \frac{\frac{F(s)}{\Phi(s)} + mls^2}{Ms^2 + ms^2 + bs} + g - ls^2 \tag{16}$$
$$\frac{Is^2}{ml} = \frac{\frac{F(s)}{\Phi(s)} + mls^2}{M + m + \frac{b}{s}} + g - ls^2 \tag{17}$$
$$\frac{F\left(s\right)}{\Phi\left(s\right)} = \left[\frac{I_s^2}{ml} - g + ls^2\right]\left[M + m + \frac{b}{s}\right] - mls^2 \tag{18}$$
$$\frac{F\left(s\right)}{\Phi\left(s\right)} = \frac{MI}{ml}s^2 + \frac{I}{l}s^2 + \frac{Ib}{ml}s - gM - gm - \frac{gb}{s} + Mls^2 + mls^2 - mls^2 \tag{19}$$
$$\frac{F\left(s\right)}{\Phi\left(s\right)} = s^2\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s\left(\frac{Ib}{ml} + lb\right) - gM - gm - \frac{gb}{s} \tag{20}$$
$$\frac{\Phi\left(s\right)}{F\left(s\right)} = \frac{1}{s^2\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s\left(\frac{Ib}{ml} + lb\right) - gM - gm - \frac{gb}{s}} \tag{21}$$
$$\frac{\Phi\left(s\right)}{F\left(s\right)} = \frac{s}{s^3\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s^2\left(\frac{Ib}{ml} + lb\right) + s(-gM - gm) - gb} \tag{22}$$
$$\frac{\Phi\left(s\right)}{F\left(s\right)} = \frac{s}{s^3\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s^2\left(\frac{Ib}{ml} + lb\right) + s\left(-gM - gm\right) - gb} \tag{23}$$

$G_2(s)$ Solving equation 6 for $\Phi(s)$:

$$\Phi\left(s\right)s^2 = \frac{1}{I}\,\left(mlX\left(s\right)s^2 + mlg\Phi\left(s\right) - ml^2\Phi\left(s\right)s^2\right) \tag{24}$$

$$I\Phi\left(s\right)s^2 = mlX\left(s\right)s^2 + mlg\Phi\left(s\right) - ml^2\Phi\left(s\right)s^2 \tag{25}$$

$$\Phi\left(s\right) = \frac{mlX\left(s\right)s^2}{Is^2 + ml^2s^2 - mlg} \tag{26}$$

Inserting into the previously calculated transfer function:

$$\frac{\frac{mlX(s)s^2}{Is^2+ml^2s^2-mlg}}{F\left(s\right)} = \frac{s}{s^3\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s^2\left(\frac{Ib}{ml} + lb\right) + s(-gM - gm) - gb} \tag{27}$$

$$\frac{X\left(s\right)}{F\left(s\right)} = \frac{Is^2 + ml^2s^2 - mlg}{mls^2}\cdot\frac{s}{s^3\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s^2\left(\frac{Ib}{ml} + lb\right) + s(-gM - gm) - gb} \tag{28}$$

$$\frac{X\left(s\right)}{F\left(s\right)} = \frac{Is^2 + ml^2s^2 - mlg}{mls\cdot\left[s^3\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s^2\left(\frac{Ib}{ml} + lb\right) + s\left(-gM - gm\right) - gb\right]} \tag{29}$$

$$\frac{X\left(s\right)}{F\left(s\right)} = \frac{s^2\left(I + ml^2\right) - mlg}{s^4\left(MI + Im + Mml^2\right) + s^3\left(Ib + mbl^2\right) + s^2\left(-gml\left[M + m\right]\right) - s\left(gbml\right)} \tag{30}$$

$$\tag{31}$$

Our two transfer functions are therefore

$$G_1\left(s\right) = \frac{\Phi\left(s\right)}{F\left(s\right)} = \frac{s}{s^3\left(\frac{MI}{ml} + \frac{I}{l} + Ml\right) + s^2\left(\frac{Ib}{ml} + lb\right) + s(-gM - gm) - gb} \tag{32}$$

$$G_2\left(s\right) = \frac{X\left(s\right)}{F\left(s\right)} = \frac{s^2\left(I + ml^2\right) - mlg}{s^4\left(MI + Im + Mml^2\right) + s^3\left(Ib + mbl^2\right) + s^2\left(-gml\left[M + m\right]\right) - s\left(gbml\right)} \tag{33}$$

To validate the calculations, the results were compared to the ones yielded in the online documentation[1]. The poles and zeros matched and therefore it can be assumed that the calculations are correct.

4 Discretization linear model

4.1 Forward Euler

$$z = e^{sT} \approx 1 + sT \quad\rightarrow\quad s \approx \frac{z - 1}{T} \tag{34}$$

$$hallo \quad = \quad hallo \tag{35}$$

[1]http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

4.2 Backward Euler

$$z = e^{sT} \approx \frac{1}{1+sT} \quad \rightarrow \quad s \approx \frac{z-1}{Tz} \tag{36}$$
$$hallo = hallo \tag{37}$$

4.3 Trapezoidal or Tustin

$$z = e^{sT} \approx \frac{1+sT/2}{1-sT/2} \quad \rightarrow \quad s \approx \frac{2\,(z-1)}{T\,(z+1)} \tag{38}$$
$$hallo = hallo \tag{39}$$

4.4 Discretizing using Matlab

Matlab has a built in function *c2d*() that can discretize a continuous time transfer function. It only requires the transfer function and the sample time as an input. We using 0.0001 seconds as the sampling time. The Matlab code is shown below:

```
cart_n2 = (I+m*l^2)/q;
cart_n1 = 0;
cart_n0 = -g*m*l/q;
cart_d4 = 1;
cart_d3 = b*(I+m*l^2)/q;
cart_d2 = ((M + m)*m*g*l)/q;
cart_d1 = - b*m*g*l/q;
cart_d0 = 0;

pend_n1 = m*l/q;
pend_n0 = 0;
pend_d3 = 1;
pend_d2 = (b*(I + m*l^2))/q;
pend_d1 = -((M + m)*m*g*l)/q;
pend_d0 = -b*m*g*l/q;

P_cart = (cart_n2*s^2 + cart_n1*s + cart_n0)/(cart_d4*s^4 + cart_d3*s^3 + cart_d2*s
    ^2 + cart_d1*s + cart_d0)
P_pend = (pend_n1*s + pend_n0)/(pend_d3*s^3 + pend_d2*s^2 + pend_d1*s + pend_d0)


%% discretizing the transfer functions
d_P_cart = c2d(P_cart, 0.0001)
d_P_pend = c2d(P_pend, 0.0001)
```

This script puts out the discrete transfer function

$$\frac{2.273 \cdot 10^{-8} z^2 - 1.377 \cdot 10^{-13} z - 2.273 \cdot 10^{-8}}{z^3 - 3z^2 + 3z - 1} \tag{40}$$

5 System analysis

6 Control function

Part II
Laboratory Session 07

List of Figures