# Lab 3:“Hardware” and Proportional Control

---

**Name: Mingjun Wu, Matthew Domine**

**Date: 03/29/2021**

# Introduction

The goal of this lab:

1. derive and understand a model for the dynamics of the cart (without the pendulum).
2. use proportional control to generate a step response on the actual hardware

# Equipment (that is being simulated)

Cart system (no attachments), Quanser Q4/Q2-usb DAQ board w/ terminal board, amplifier and cables.

The terminal board in Figure 1 provides connectors for the inputs and outputs of the Q4/Q2-usb DAQ boards in the computers. Specifically, you will use a single analog output (this will provide the motor voltage) and two encoder inputs (for cart position and pendulum angle). Since the analog out ports of the Quanser Q4/Q2-usb board are not powerful enough to drive the DC motor on the cart, the signal needs to be amplified (in fact, the gain will be unity, but the amplifier can provide much higher currents than the DAQ boards). Figure 2 shows the amplifier that drives the DC motor on the cart. In all of the coming labs, you will only use the ports “From analog output (D/A)” and “To Motor”. The cables we use are special cables that have resistors between the connector ports built in so that the connection results in the correct op-amp circuit.

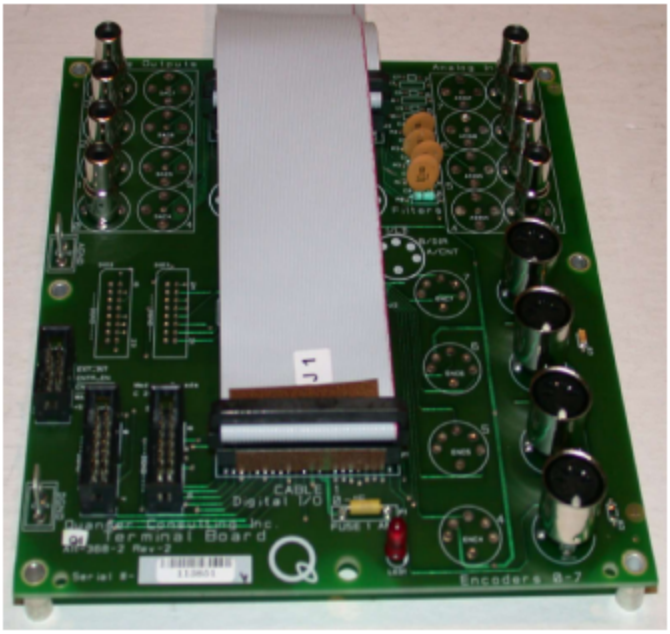



Figure 1: Terminal board of the Quanser Q4 DAQ card

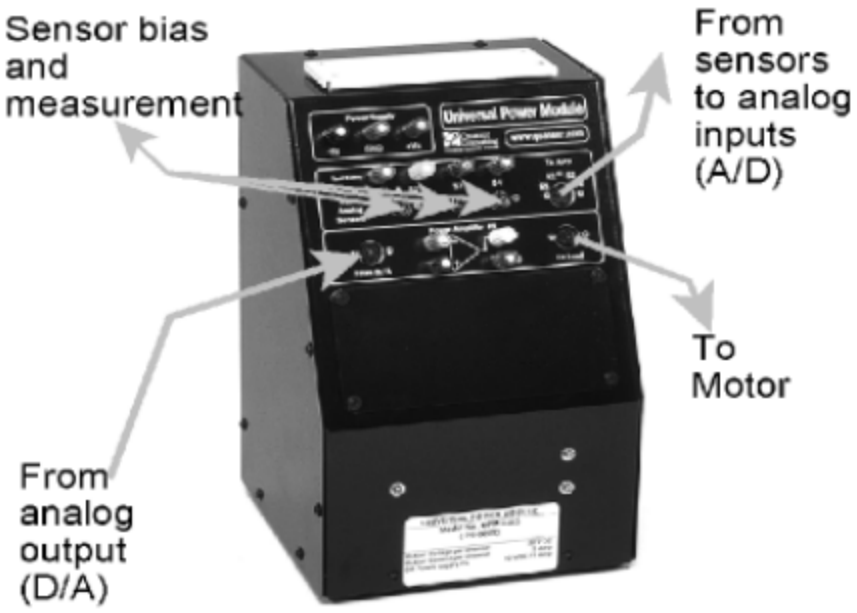



Figure 2: Amplifier for the cart's DC motor

# Theory

## Simulink Coder, QuaRC, and the Q4 DAQ board

MATLAB's Simulink Coder (formerly Real-Time Workshop) generates and executes C and C++ code from Simulink diagrams, Stateflow charts, and MATLAB functions. The generated source code can be used for real-time and nonreal-time applications, including simulation acceleration, rapid prototyping, and hardware-in-the-loop testing.

QuaRC is Quanser's rapid prototyping and production system for real-time control. QuaRC integrates seamlessly with Simulink to allow Simulink models to be run in real-time on Windows. It uses a host and target relationship that allows code generation and execution to occur on separate machines. However, we will be using QuaRC in "Single User Mode" or "Local Configuration", where we will be generating and executing code on the same computer, as shown in Figure 3.

The QuaRC Simulink Development Environment (SDE) is used to generate/build code to be later run on a real-time target from MATLAB/Simulink models. The QuaRC Windows Target feature is required to run the generated code from MATLAB/Simulink models on a real-time Windows target (local or remote). QuaRC Windows Target needs to be open to run any QuaRC-generated code.
While interesting, understanding the details of the implementation of the QuaRC software is not the focus of this lab. Your task is to design the controller based on either classic or state-space techniques. Then you will implement the controller in Simulink. This is then downloaded to the QuaRC target, which interfaces the plant through the Q4 DAQ board. This board supports 4 A/D converters, 4 D/A converters, 16 Digital I/Os, 2 Realtime clocks, and up to 4 Quadrature input decoders/counters. The Q4 board's functionality has also been abstracted from the user. The board has been set up to work with the cart and pendulum for all stations.

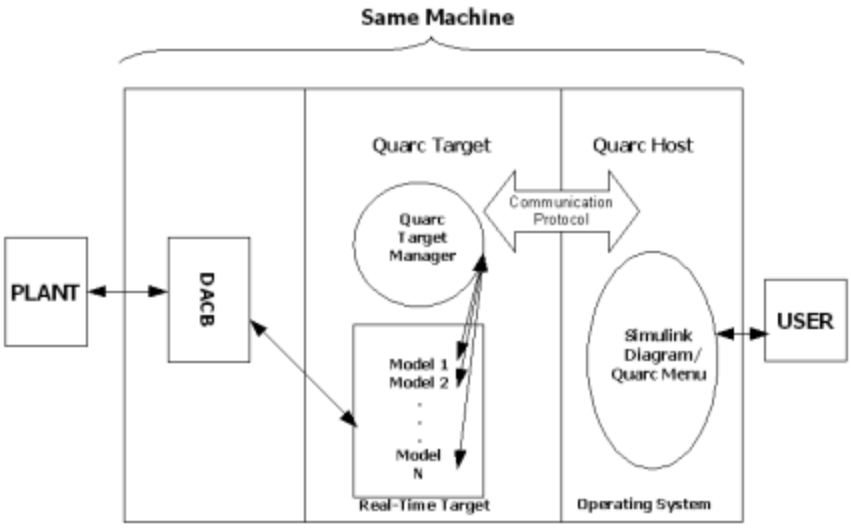



Figure 3: “Local Configuration” – QuaRC Host and Target on the same PC

## Dynamics of the Cart

Figure 4 shows a picture of the cart used in the lab setup for the pendulum experiments. The main components are the Cart motor pinion (5) with attached 6V DC motor and gearbox (not visible in the figure), the Cart position pinion (4) with attached encoder (8) and the pendulum axis (7) with attached encoder (9).

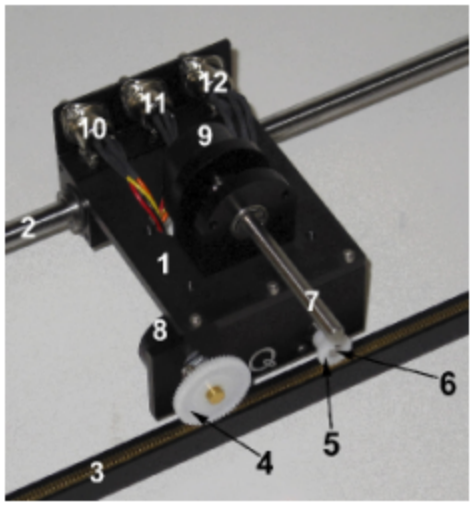



Figure 4: Quanser IP02 cart

Figure 5 shows the cart’s free body diagram. For simplicity, we will ignore the effects of friction. In the diagram, Fa is the input force exerted on the cart by the voltage applied to the

motor, mc is the mass of the cart. The encoder is used to keep track of the position of the cart on the track.

Using Figure 5 and basic Newtonian dynamics you can derive the equations governing the system

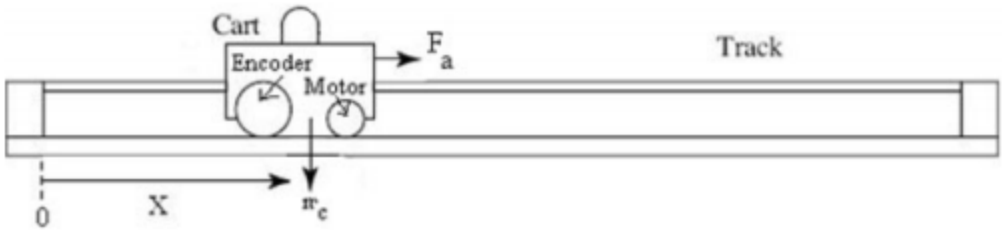



Figure 5: Free body diagram of the cart (ignoring friction)

## Motor Dynamics

The input to your system is actually a voltage to the cart's motor. Thus, you need to derive the dynamics of the system that converts the input voltage to the force exerted on the cart. These are the dynamics of the motor. Figure 6 shows a diagram of the electrical components of the motor.

Figure 6 shows a diagram of the electrical components of the motor.

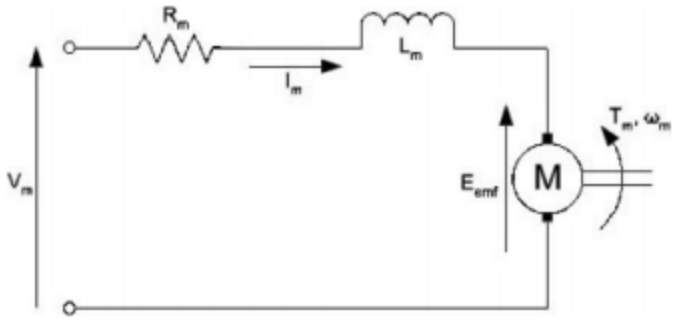



Figure 6: Classic armature circuit of a standard DC motor

For this derivation of the motor dynamics we assume the following:

- We disregard the motor inductance: $L_m \ll R_m$ , so we can use the approximation $L_m \approx 0$.
- Perfect efficiency of the motor and gearbox: $\eta_m = \eta_g = 1$.

The torque generated by the motor is proportional to the current flowing through the motor windings, but is lessened due to the moment of inertia:

$$T_m = K_t I_m - J_m \ddot{\theta} \tag{1}$$

Here $K_t$ is the motor torque constant, Im is the current flowing through the coil, $J_m$ is the moment of inertia of the motor and $\ddot{\theta}$ is the angular acceleration of the motor. The current flowing through the motor can be related to the motor voltage input by:

$$V = I_m R_m + E_{emf} = I_m R_m + K_m \dot{\theta} \tag{2}$$

where $\dot{\theta}$ is the angular velocity of the motor, $R_m$ is the resistance of the motor windings and Km is the back EMF constant (in $\frac{V}{rad/s}$ ).

The torque is related to the applied force via

$$K_g T_m = F_a \cdot r \tag{3}$$

where r is the radius of the motor gear and Kg is the gearbox gear ratio. The motor's angular velocity is related to the cart's linear velocity via

$$K_g \dot{x} = \dot{\theta} \cdot r \Rightarrow K_g \ddot{x} = \ddot{\theta} \cdot r \tag{4}$$

## Step Response of a Dynamical System

Figure 7 shows the typical step response of a SISO (single-input, single-output) dynamical system.

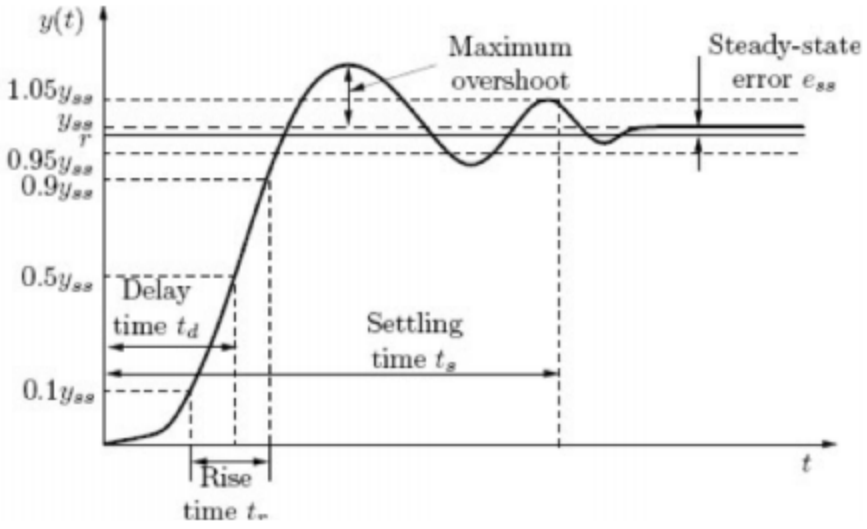



Figure 7: Typical step response of a control system

Recall the following quantities from class:

- **Steady-state value:** The steady-state value of the response y(t) is defined as $y_{ss} = \lim_{t\to\infty} y(t)$.
- **Steady-state error:** For a control system, we want the output, y(t), to follow a desired reference signal, r(t). Thus we can define the error as $e(t) = r(t) - y(t)$. Consequently, the steady-state error is given by $e_{ss} = \lim_{t\to\infty} e(t)$.
- **Maximum overshoot:** Let $y_{max}$ denote the maximum value of y(t). The maximum overshoot of the step response y(t) is defined by $maximum\ overshoot = y_{max} - y_{ss}$. It is often represented as a percentage of the steady-state value: $percent\ maximum\ overshoot = \frac{y_{max} - y_{ss}}{y_{ss}}$. The maximum overshoot is often used to measure the relative stability of a system. A system with a large overshoot is usually undesirable.
- **Delay time:** The *delay time* $t_d$ is defined as the time required for the step response to reach 50% of its steady-state value.
- **Rise time:** The rise time $t_r$ is defined as the time required for the step response to rise from 10% to 90% of its steady-state value.
- **Settling time:** The settling time $t_s$ is defined as the time required for the step response to stay within 5% of its steady-state value.

# Analysis & Results

## 1. Pre Lab

### 1.1. Equations Governing the Cart Dynamics

Derive the following equation of motion for the cart system shown in Figure 5:

$$(m_c r^2 R_m + R_m K_g^2 J_m)\ddot{x} + (K_t K_m K_g^2)\dot{x} = (r K_t K_g) V \quad (5)$$

Table 1 lists the parameters that appear in (5).

| Parameter | Unit | Description |
|---|---|---|
| $V$ | Volt | input voltage |
| $m_c$ | kg | mass of the car |
| $r$ | meter | radius of the motor gears |
| $R_m$ | $\Omega$ | resistance of the motor windings |
| $K_t$ | N·m/A | torque motor constant |
| $K_m$ | Vs/rad | back EMF constant |
| $K_g$ | - | gearbox ratio |
| $J_m$ | $\mathrm{kg\,m^2}$ | moment of inertia of the motor |

Table 1: Parameters of the cart system

In order to derive the equation (5), follow the steps below:

1. Using the free body diagram in Figure 5, apply Newton's second law to the cart.

1) $F_a = m_c a$

$$\boxed{F_a = m_c \ddot{x}} \qquad (*)$$

2. Combine the motor dynamics, equations (1) - (4), to obtain the relationship between the input voltage V and the applied force $F_a$. Substitute this relationship into your equation from Step 1. This is the final model of your plant.

$$T_m = K_t I_m - J_m \ddot{\theta} \quad (1)$$

$$V = I_m R_m + K_m \dot{\theta} \quad (2)$$

$$V - K_m \dot{\theta} = I_m R_m$$

$$I_m = \frac{1}{R_m}(V - K_m \dot{\theta}) \quad (2^*)$$

$$K_g T_m = F_a \cdot r \quad (3)$$

$$F_a = \frac{K_g T_m}{r} \quad (3^*)$$

$$K_g \dot{x} = \dot{\theta} \cdot r \Rightarrow K_g \ddot{x} = \ddot{\theta} \cdot r \quad (4)$$

$$\dot{\theta} = \frac{K_g}{r} \dot{x} \qquad \ddot{\theta} = \frac{K_g}{r} \ddot{x} \quad (4^*)$$

Substitute $(2^*)$ into $(1)$

$$T_m = \frac{K_t}{R_m}(V - K_m \dot{\theta}) - J_m \ddot{\theta}$$

Substitute $(4^*)$ into this EQN

$$T_m = \frac{K_t}{R_m}\left(V - \frac{K_m K_g}{r}\dot{x}\right) - J_m\left(\frac{K_g}{r}\ddot{x}\right)$$

Substitute $T_m$ into $(3^*)$

$$F_a = \frac{K_g}{r}\left[\frac{K_t}{R_m}\left(V - \frac{K_m K_g}{r}\dot{x}\right) - \frac{J_m K_g}{r}\ddot{x}\right]$$

Substitute $F_a$ into (*)

$$m_c \ddot{x} = F_a$$

$$m_c \ddot{x} = \frac{k_g}{r}\left[\frac{k_t}{R_m}\left(V - \frac{k_m k_g}{r}\dot{x}\right) - \frac{J_m k_g}{r}\ddot{x}\right]$$

$$m_c r \ddot{x} = \frac{k_g k_t}{R_m}\left(V - \frac{k_m k_g}{r}\dot{x}\right) - J_m \frac{k_g^2}{r}\ddot{x}$$

$$R_m m_c r^2 \ddot{x} = k_g k_t r V - k_t k_m k_g^2 \dot{x} - R_m J_m k_g^2 \ddot{x}$$

$$\Rightarrow R_m m_c r^2 \ddot{x} + R_m J_m k_g^2 \ddot{x} + k_t k_m k_g^2 \dot{x} = k_g k_t r V$$

$$\left(R_m m_c r^2 + R_m J_m k_g^2\right)\ddot{x} + k_t k_m k_g^2 \dot{x} = k_g k_t r V$$

Rearrange $\boxed{\left(m_c r^2 R_m + R_m k_g^2 J_m\right)\ddot{x} + \left(k_t k_m k_g^2\right)\dot{x} = (r k_t k_g)V}$

Compare $\left(m_c r^2 R_m + R_m K_g^2 J_m\right)\ddot{x} + \left(K_t K_m K_g^2\right)\dot{x} = (r K_t K_g)V \quad (5)$

3. Is this system linear? If not, linearize the system. If so, leave as is.

System is linear

## 1.2. Derive System Models

**Transfer Function:** Apply the Laplace transform to your linear system and solve for the transfer function $H(s) = \frac{X(S)}{V(S)}$.

$$\left(m_c r^2 R_m + R_m K_g^2 J_m\right)\ddot{x} + \left(K_t K_m K_g^2\right)\dot{x} = (r K_t K_g) V$$

Transfer function

$$\left[\left(m_c r^2 R_m + R_m K_g^2 J_m\right) s^2 + (K_t K_m K_g^2) s\right] X(s) = (r K_t K_g) V(s)$$

$$\Rightarrow \boxed{H(s) = \frac{X(s)}{V(s)} = \frac{r K_t K_g}{(m_c r^2 R_m + R_m K_g^2 J_m) s^2 + (K_t K_m K_g^2) s}}$$

**State Space Model:** Using cart position and velocity as states x1, x2, respectively, and the cart position as the system output y, derive a state space representation (i.e. matrices A, B, C and D) for your linear system.

$$\left(m_c r^2 R_m + R_m K_g^2 J_m\right)\ddot{x} + \left(K_t K_m K_g^2\right)\dot{x} = \left(r K_t K_g\right)V$$

State-Space Model

$$\dot{x} = Ax + Bu \qquad y = Cx + Du$$

$$x_1 = x \qquad x_2 = \dot{x} \qquad u = V \qquad y = x = x_1$$

$$\dot{x}_1 = \dot{x} = x_2$$

$$\dot{x}_2 = \ddot{x} = \frac{-\left(K_t K_m K_g^2\right)x_2 + \left(r K_t K_g\right)u}{m_c r^2 R_m + R_m K_g^2 J_m}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \hline y \end{bmatrix} = \left[\begin{array}{cc|c} 0 & 1 & 0 \\ 0 & \dfrac{-K_t K_m K_g^2}{m_c r^2 R_m + R_m K_g^2 J_m} & \dfrac{r K_t K_g}{m_c r^2 R_m + R_m K_g^2 J_m} \\ \hline 1 & 0 & 0 \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \hline u \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-K_t K_m K_g^2}{m_c r^2 R_m + R_m K_g^2 J_m} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{r K_t K_g}{m_c r^2 R_m + R_m K_g^2 J_m} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \end{bmatrix}$$

**SS to TF:** Using the following equation, derive a transfer function from your state space matrices and verify that it matches the transfer function you got directly from taking the Laplace transform of the equation of motion.

$$G(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D$$

let $d = -\frac{K_t K_m K_g^2}{m_c r^2 R_m + R_m K_g^2 J_m}$, $u_2 = \frac{r K_t K_g}{m_c r^2 R_m + R_m K_g^2 J_m}$

$$G(s) = \begin{bmatrix} 1 & 0 \end{bmatrix} \left( \begin{bmatrix} s & -1 \\ 0 & s-d \end{bmatrix} \right)^{-1} \begin{bmatrix} 0 \\ u_2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \end{bmatrix} \left( \frac{1}{s^2 - ds} \begin{bmatrix} s-d & 1 \\ 0 & s \end{bmatrix} \right) \begin{bmatrix} 0 \\ u_2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{s-d}{s^2-ds} & \frac{1}{s^2-ds} \\ 0 & \frac{s}{s^2-ds} \end{bmatrix} \begin{bmatrix} 0 \\ u_2 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{s-d}{s^2-ds} & \frac{1}{s^2-ds} \end{bmatrix} \begin{bmatrix} 0 \\ u_2 \end{bmatrix}$$

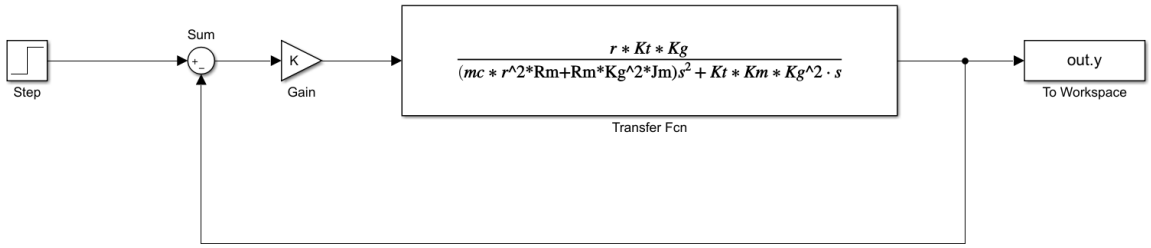
$$= \frac{1}{s^2 - ds} u_2$$

$$= \frac{1}{s^2 + \frac{K_t K_m K_g^2}{m_c r^2 R_m + R_m K_g^2 J_m} s} \left( \frac{r K_t K_g}{m_c r^2 R_m + R_m K_g^2 J_m} \right)$$

$$\boxed{G(s) = \frac{Y(s)}{U(s)} = \frac{r K_t K_g}{(m_c r^2 R_m + R_m K_g^2 J_m) s^2 + (K_t K_m K_g^2) s}}$$

## 1.3. Matlab Step Response

### Simulink Block Diagram



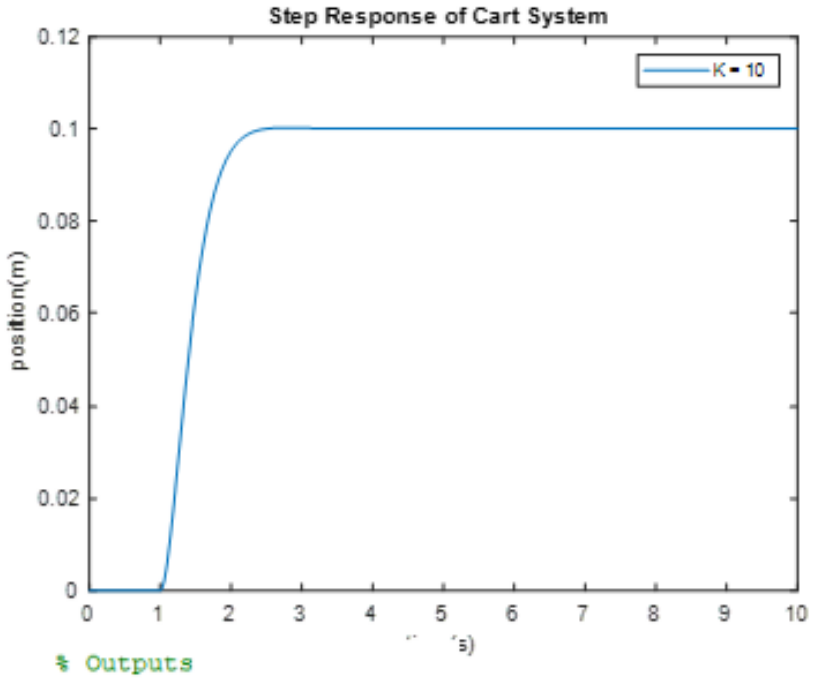

### Matlab Code

```
% Define Parameters
mc = 0.94; %[kg]
r = 6.36 * 10^(-3); %[m]
Rm = 2.6; %[Ohms]
Kt = 7.67*10^(-3); %[Nm/A]
Km = 7.67*10^(-3); %[Vs/rad]
Kg = 3.71;
Jm = 3.9 * 10^(-7); %[kgm^2]

% Set Gain Value
K = 10 %First Value Guess
% K = 15

% Simulate
out = sim('cartdynamics'); %Simulate
plot(out.tout,out.y) %Plot Output vs Time
xlabel('time(s)'),ylabel('position(m)') %Label Axes
title('Step Response of Cart System') %Title
legend(['K = ',num2str(K)]) %Legend

% Step Response Characteristics
yss = out.y(end); %Steady State
ymax = max(out.y); %Peak
overshoot = (ymax - yss)/yss*100 %Percent Overshoot
t1 = find(out.y >= 0.1*yss ,1);  %Index when step response is 10% of its steady state value
t2 = find(out.y >= 0.9*yss ,1); %Index when step response is 90% of its steady state value
tr = out.tout(t2) - out.tout(t1) %Rise Time
```

We will iterate through values of K until we reach a percent maximum overshoot < 3.0% and rise time $t_r$ <0.75s.

We start with K = 10 which appears to satisfy these conditions.

## Step Response & Outputs




```
% Outputs
%{
K =

    10

overshoot =

    0.1766

tr =

    0.7100
%}
```

```
>> S = stepinfo(out.y,out.tout)

S =

  struct with fields:

       RiseTime: 0.7154
   SettlingTime: 2.1626
    SettlingMin: 0.0902
    SettlingMax: 0.1002
      Overshoot: 0.1766
     Undershoot: 0
           Peak: 0.1002
       PeakTime: 2.7700
```

For when K = 10
$t_r$ = 0.71s
%OS = 0.1766 %

$t_r$ < 0.75s
%OS < 3%

# 2. Lab

## 2.1. Cart Dynamics

**Chosen System Representation: Transfer function**

$$\frac{r * Kt * Kg}{(mc * r\text{^}2*\text{Rm}+\text{Rm}*\text{Kg}\text{^}2*\text{Jm})s^2 + Kt * Km * Kg\text{^}2 \cdot s}$$

Transfer Fcn

*Figure 1: Transfer function of the plant model in Simulink model*

## 2.2. Using the "Actual" Hardware: Setting Up Simulation

- Unzip lab3.zip in the folder/directory you wish to complete the lab in. Make sure all the files used for this lab stay in the same folder/directory to prevent needing to add paths. The Simulink block provided needs to reference some of the provided files.
- The block, cartDynamics, inside the library, lab_3_pseudo_hardware.slx, represents the dynamics of the Quanser Cart. The actual cart shares the same input and output as the block. The input being voltage applied to the motor, and the output being the position of the cart measured by the encoder. Drag the block into your Simulink model and connect it like you did in the prelab with the plant model.

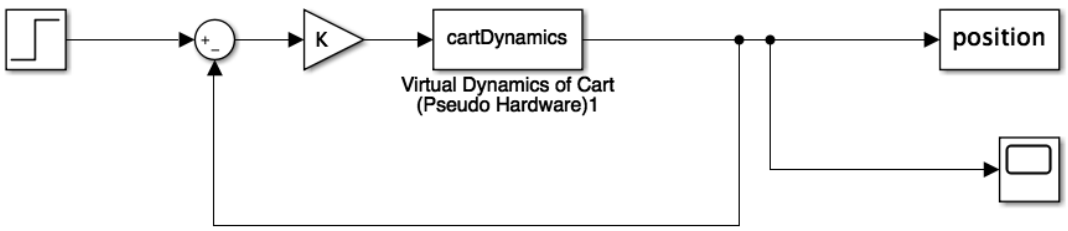



*Figure 2: Simulink model after replacing the plant model with cartDynamics*

- Encoder values for the position of the cart will be read in encoder counts, however, our input defined in the system equations were in meters. In feedback, the two values you compare must be in the same units, so we need a conversion factor. The Quanser manual gives the position encoder resolution to be 4096 counts/revolution. Given that the radius of the position pinion is $r_{pp}$ = 0.01482975 m, what is the "actual" encoder resolution in counts/m? Now add a gain block to the position signal to convert the signal's units to meters.

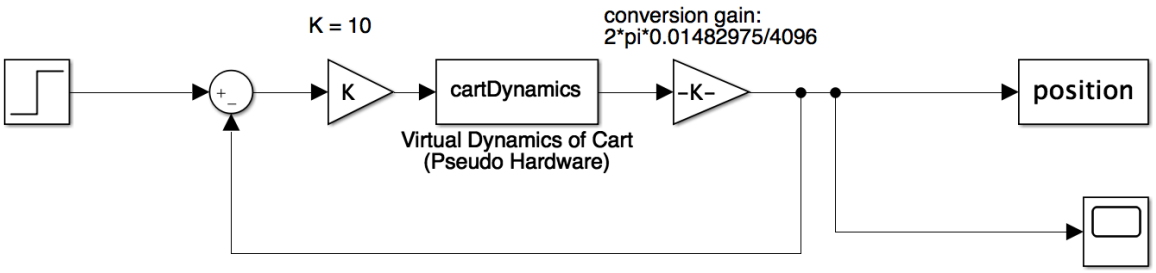



*Figure 3: Simulink model after* adding a gain block to the position signal to convert the signal's units to meters

The relationship between encoder counts and meter is determined:

$$\text{Encoder Conversion gain: } \frac{2\pi \times \textit{radius of the position pinion}}{\#\ \textit{counts/revolution}} = \frac{2\pi \times 0.01482975}{4096} = 2.275 \times 10^{-5}$$

- Attach 'to Workspace' blocks (in the 'Sinks' folder) to the reference and output signals. Assign an appropriate variable name, while keeping the other parameters their defaults:

- Limit data points to last: inf
- Decimation: 1
- Save format: Timeseries
- Log fixed-point data as a fi object: Checked
- Sample time: -1

This will create a struct in your Matlab workspace once the simulation completes successfully. If the variable name was set to tmp, you can find the signal data at out.tmp.Data with its corresponding time at out.tmp.Time.

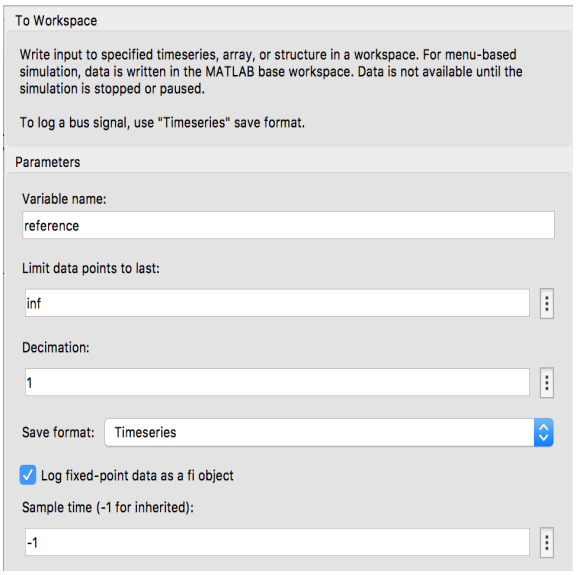



*Figure 4: Workspace setting for the reference signal*

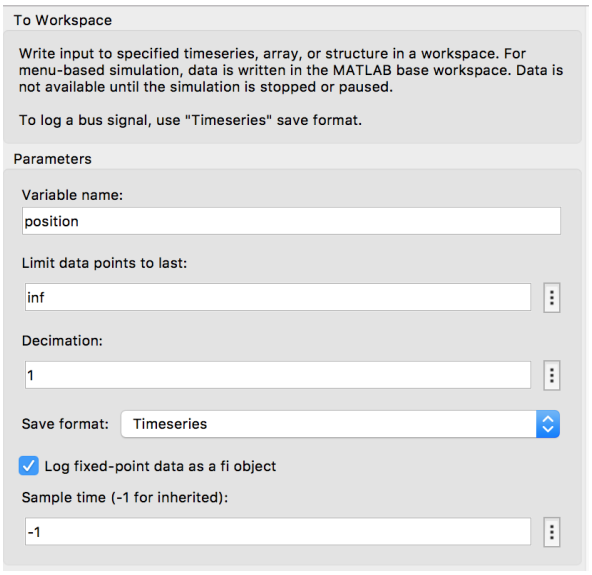



*Figure 5: Workspace setting for the output signal*

- After you run a simulation, in the next section, you will be able to play a short movie of the cart tracking at the set point. To visualize your cart,

call cartAnimation(time,position,reference) in the Command Window, where time is a vector of time, position is a vector of the cart position, and reference is a vector of the set point of the cart. The reference input is optional. In the movie that plays, the black line represents the rail, the blue rectangle is the cart, and the red box is the reference point. Make sure all the inputs are the same size, and that the time vectors for the reference signal and the position signal are the same.

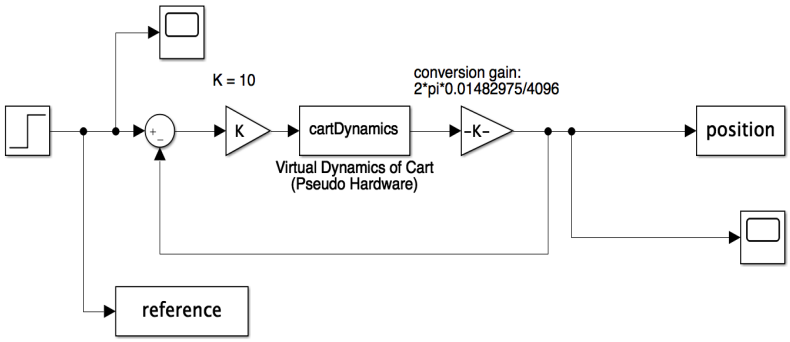



*Figure 6: The block diagram of the cart system in Simulink*

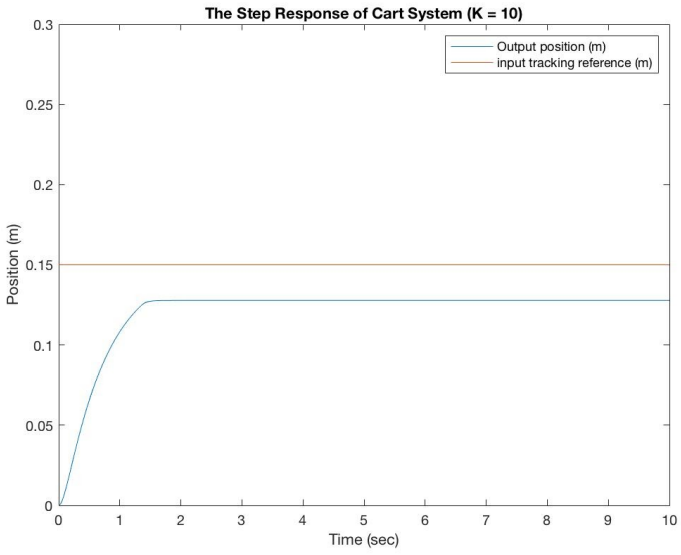



*Figure 7: The step response of the cart system at K = 10*

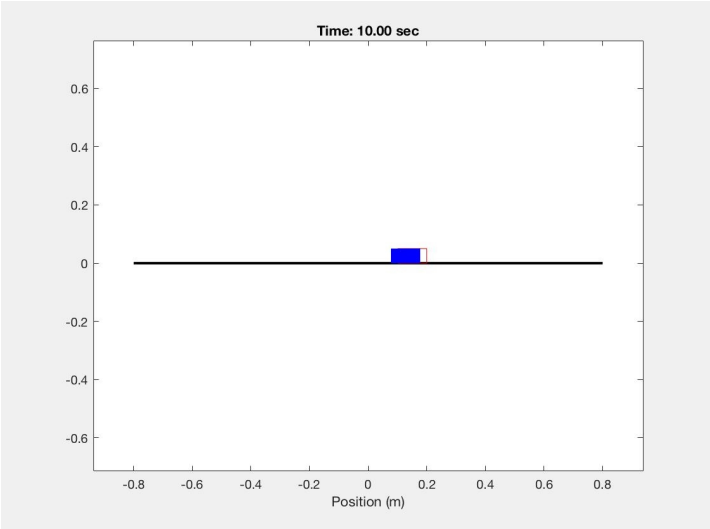



*Figure 8: Animation of the cart system at K = 10*

## 2.3. Using the "Actual" Hardware: Cart Step Response

Go back to your Simulink model of the cart system from the pre-lab. Now change the step function to be of height 0.15, corresponding to the cart moving 0.15 m. Again, try to find a value of K so that percent maximum overshoot < 3.0% and $t_r$ < 0.75s. Report this value in your lab report. How different is the value you found here from the value of K you found in the pre-lab for a step size of 1? Why is that?

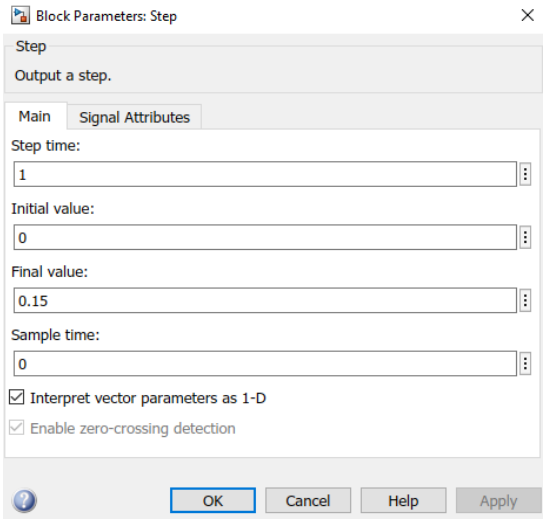



*Figure 9: Block Parameters setup of the step input*

**The K value found is K = 10 again.**

```
clear, clc, close all;
% Define Parameters
mc = 0.94; %[kg]
r = 6.36 * 10^(-3); %[m]
Rm = 2.6; %[Ohms]
Kt = 7.67*10^(-3); %[Nm/A]
Km = 7.67*10^(-3); %[Vs/rad]
Kg = 3.71;
Jm = 3.9 * 10^(-7); %[kgm^2]

% Set Gain Value
K = 10

% Simulate
out = sim('cartdynamics'); %Simulate
plot(out.tout,out.y) %Plot Output vs Time
xlabel('time(s)'),ylabel('position(m)') %Label Axes
title('Step Response of Cart System') %Title
legend(['K = ',num2str(K)]) %Legend

% Step Response Characteristics
yss = out.y(end); %Steady State
ymax = max(out.y); %Peak
overshoot = (ymax - yss)/yss*100 %Percent Overshoot
t1 = find(out.y >= 0.1*yss ,1);  %Index when step response is 10% of its steady state value
t2 = find(out.y >= 0.9*yss ,1); %Index when step response is 90% of its steady state value
tr = out.tout(t2) - out.tout(t1) %Rise Time

S = stepinfo(out.y,out.tout)
if (S.Overshoot < 3)&&(S.RiseTime < 1.5)    fprintf("K = %f works!\n",K);
end
```

*Figure 10: Matlab Code of plotting the step response using $K = 10$*

**Step Response & Outputs**

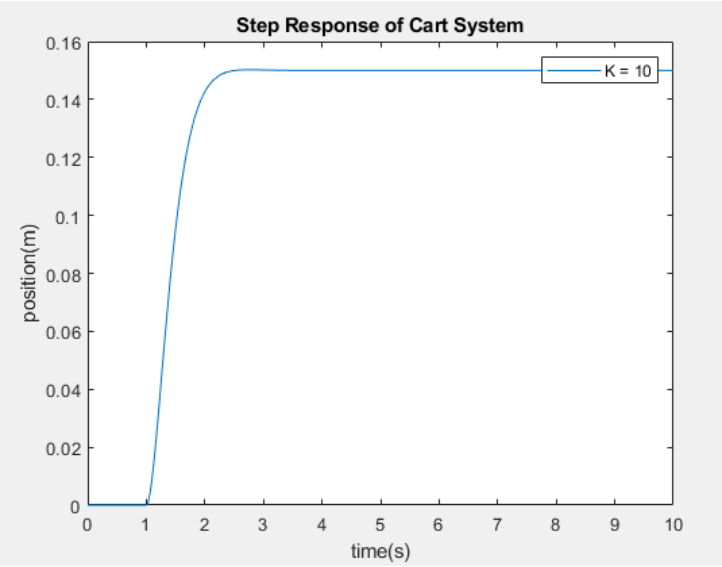



*Figure 11: the step response of cart system using $K = 10$*

```
K =

    10


overshoot =

    0.1766


tr =

    0.7100


S =

  struct with fields:

        RiseTime: 0.7154
    SettlingTime: 2.1626
     SettlingMin: 0.1353
     SettlingMax: 0.1503
       Overshoot: 0.1766
      Undershoot: 0
            Peak: 0.1503
        PeakTime: 2.7700

K = 10.000000 works!
```

*Figure 12: Matlab Code Outputs*

There is no difference between the K value found here and the value of K found in the pre-lab. This makes sense because the Simulink model is a second order LTI system, and amplifying the input would only result in amplifying the output. The step response characteristics do not depend on the input but the damping ratio $\zeta$ of the system. Mathematically speaking, the amplitude of the input plays no role in computing the rise time and the maximum overshoot.

$$Explicit\ formula\ for\ the\ rise\ time: \ t_r = 1.76\xi^3 - 0.417\xi^2 + 1.039\xi + 1$$

$$Maximum\ Overshoot: M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\xi^2}}}$$

- Once the simulated step response looks fine, you can move over to the simulated hardware. Plot the initial hardware response and compare with the plot from the Simulink model. How close was the actual to the

predicted? What might have caused any discrepancies? Hint: take a look at the control signal (i.e. the motor voltage).

## Matlab Code

```
clear, clc, close all
% Define Parameters
mc = 0.94; %[kg]
r = 6.36 * 10^(-3); %[m]
Rm = 2.6; %[Ohms]
Kt = 7.67*10^(-3); %[Nm/A]
Km = 7.67*10^(-3); %[Vs/rad]
Kg = 3.71;
Jm = 3.9 * 10^(-7); %[kgm^2]

% Set Gain Value
K = 10

% Simulate
out = sim('cartdynamicshard'); %Simulate
out2 = sim('cartdynamics');%Simulate
plot(out.tout,out.position.data) %Plot position vs Time
hold on
plot(out2.tout,out2.y.data)
hold on
plot(out.tout,out.reference.data) %Plot Reference vs Time
axis([0 10 0 0.2])
xlabel('time(s)'),ylabel('position(m)') %Label Axes
title(['Step Response of Cart System for K =',num2str(K)]) %Title
legend('Step Response of psuedoHardware','Step Response of Simulink Model','The Reference Target')%Legend

sysStepInfo = stepinfo(out.position.data,out.tout)

if (sysStepInfo.Overshoot < 3)&&(sysStepInfo.RiseTime < 0.75)    fprintf("K = %f works!\n",K);
end
```

*Figure 13: Matlab Code of plotting simulink model response, hardware response and reference response in one plot*

## Step Response

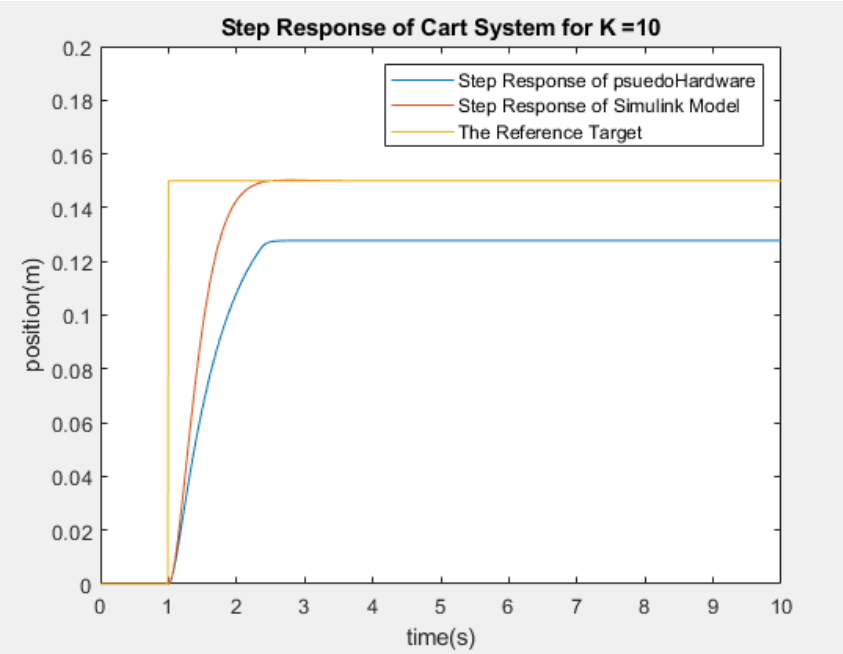



*Figure 14: Step responses of simulink model, hardware model and reference input*

$$Percent\ Error: \frac{simulink\ steady\ state\ value - hardware\ steady\ state\ value}{simulink\ steady\ state\ value} \times 100 = \frac{0.150 - 0.127756}{0.150} \times 100 = 14.8\%$$

The step responses of the Simulink and hardware models differ in steady-state value by 14.8% which is not considered close. The response's steady-state value of the physical hardware model is less than the response's steady-state value of the Simulink model. With a percent error of 14.8%, the discrepancies can be explained as a result of the hardware's physical limitation and the effects of nonlinearity such as saturation and dead zone.

While giving input but not receiving any output from the system in which range of input is called dead zone, which occurs when motors do not respond at very low input voltages due to frictional forces. Saturation could affect the transient performance of the response and even lead to undesirable inaccuracy. Saturation imposes the controller gain to the zero value. It happens because when the system enters saturation the slope of its characteristic curve becomes horizontal. Thus any change in input can not change the output of the system. In this lab, the input voltage is set to the saturation value of 6 Volts. Other effects which are not modeled into the system such as drag effect, physical hardware's tolerances, system errors, motor inductance, and capacitance, etc also might have caused the discrepancies. All these effects slow the hardware response from the response of the ideal Simulink model from the pre-lab.

- Now change your value of K until you achieve a percent maximum overshoot of < 3.0% and $t_r$ < 0.75s. Report your new K value and plot the hardware response. Why is the new K different? Show your modified hardware step response to the GSI before the end of the lab session.

## Simulink Model

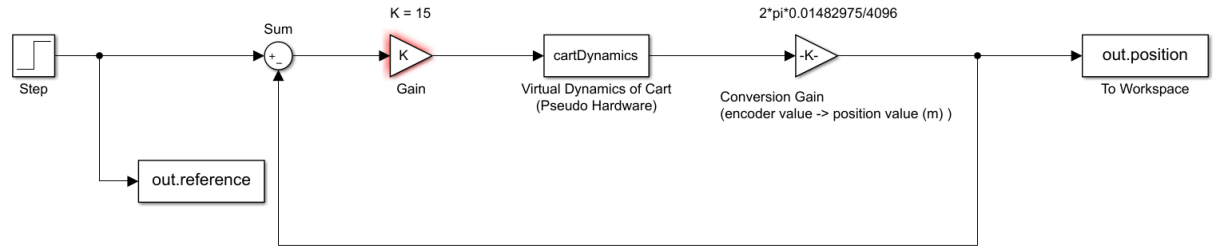



*Figure 15: Simulink Model of the hardware system*

## Matlab Code

```
clear, clc, close all
% Define Parameters
mc = 0.94; %[kg]
r = 6.36 * 10^(-3); %[m]
Rm = 2.6; %[Ohms]
Kt = 7.67*10^(-3); %[Nm/A]
Km = 7.67*10^(-3); %[Vs/rad]
Kg = 3.71;
Jm = 3.9 * 10^(-7); %[kgm^2]

% Set Gain Value
K = 15

% Simulate
out = sim('cartdynamicshard'); %Simulate
plot(out.tout,out.position.data) %Plot position vs Time
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 0 0.16])
title(['Step Response of Cart System for K =',num2str(K)]) %Title
legend('Step Response of psuedoHardware')

sysStepInfo = stepinfo(out.position.data,out.tout)
if (sysStepInfo.Overshoot < 3)&&(sysStepInfo.RiseTime < 0.75)    fprintf("K = %f works!\n",K);
end
```

*Figure 16: Matlab code for plotting the hardware model using K = 15*

## Output

```
sysStepInfo =

  struct with fields:

        RiseTime: 0.7404
    SettlingTime: 2.0701
     SettlingMin: 0.1228
     SettlingMax: 0.1363
       Overshoot: 0
      Undershoot: 0
            Peak: 0.1363
        PeakTime: 2.4600

K = 15.000000 works!
```

*Figure 17: Output of the Matlab Code*

## Step Response

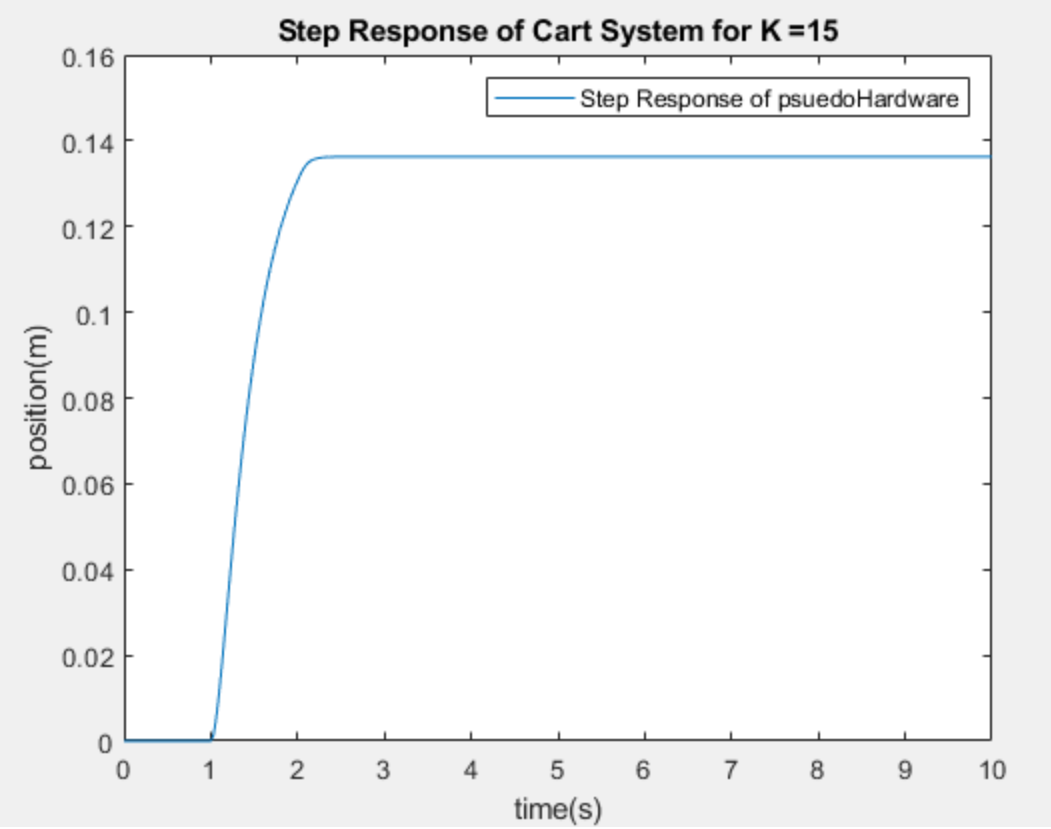



*Figure 18: Step response of Cart system at K = 15*

**New K value: 15**

To find this value, we started with the value of K = 10 and increased the value of K by increments of 1 until eventually the conditions of percent maximum overshoot of < 3.0% and tr < 0.75s were met which ended up being 15. Having different values of K implies that the cart dynamics model and the Simulink model are different. This is because our Simulink model is using a transfer function model to approximate the nonlinear physical model as a linear system with the assumptions of perfect conditions without the presence of any disturbance or effect from the environment. The cart dynamics model is most likely nonlinear, and hence it is a much more accurate and better representation of the physical hardware model.

- Finally, find a value of K that achieves a $t_r < 0.4$s and maximum overshoot < 10% using the hardware (do not use a K value over 60). Is this possible in your Simulink model from the prelab? Remember that the model of cartDynamics has a more accurate representation of the real system, so what else was modeled in the block that hasn't been modeled in the prelab? Discuss any discrepancies and make some hypotheses as to why they occur.

**Simulink Model**

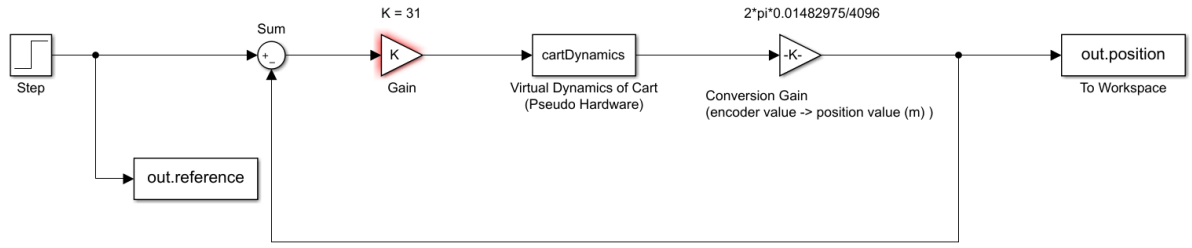



*Figure 19: Step response of Cart system at K = 15*

**Matlab Code**

```
clear, clc, close all
% Define Parameters
mc = 0.94; %[kg]
r = 6.36 * 10^(-3); %[m]
Rm = 2.6; %[Ohms]
Kt = 7.67*10^(-3); %[Nm/A]
Km = 7.67*10^(-3); %[Vs/rad]
Kg = 3.71;
Jm = 3.9 * 10^(-7); %[kgm^2]

% Set Gain Value
%K = 15
K = 31

% Simulate
out = sim('cartdynamicshard'); %Simulate
plot(out.tout,out.position.data) %Plot position vs Time
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 0 0.18])
title(['Step Response of Cart System for K =',num2str(K)]) %Title
legend('Step Response of psuedoHardware')

sysStepInfo = stepinfo(out.position.data,out.tout)
if (sysStepInfo.Overshoot < 10)&&(sysStepInfo.RiseTime < 0.4)    fprintf("K = %f works!\n",K);
end
```

*Figure 20: Matlab code for plotting the hardware model using K = 31*

```
K =

    31


sysStepInfo =

  struct with fields:

        RiseTime: 0.3990
    SettlingTime: 1.6282
     SettlingMin: 0.1343
     SettlingMax: 0.1482
       Overshoot: 0
      Undershoot: 0
            Peak: 0.1482
        PeakTime: 2.0300

K = 31.000000 works!
```

*Figure 21: Output of the matlab code*

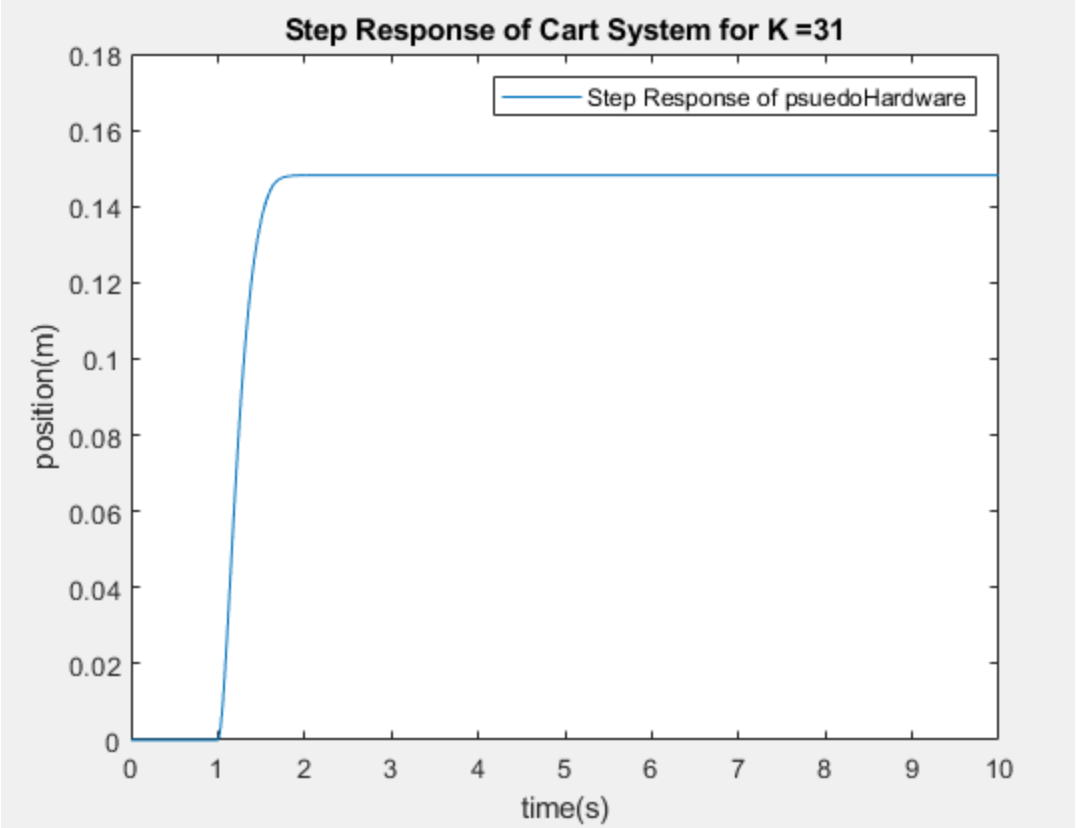



*Figure 22: Step response of the cart system at K = 31*

**New K value: 31**

To find this value we started with the value of K = 15 and doubled the value to K = 30. This value was close to reaching the rise time requirement. We then increased the value of K by 1 which met the conditions of percent maximum overshoot of < 10.0% and $t_r$ < 0.4s were met which was 31.The desired performance specification cannot be achieved with the ideal Simulink model from the pre-lab because percent overshoot would go up but reducing the rise time as K increases, and we would not able to find a value of K that meets both the requirements at the same time.

As mentioned in the previous part, there are other effects that are not modeled into the ideal simulink model such as drag effect, friction forces, physical hardware's tolerances, system errors, motor inductance and capacitance and many others might have caused the discrepancies. Nevertheless, the ideal Simulink model operates without the encoder conversion and saturation blocks and this could lead to a different result as well.