Final Report

CYBER PHYSICAL SYSTEMS

MODEL BASED CONTROL SYSTEM

FINAL REPORT (REVISION 2.1)

INTERNSHIP
ASSIGNMENT
[2015-2016]

[Submitted to]

Mark Stappers
Medewerker, Mechatronics
m.stappers@fontys.nl
R1 0.204
Fontys Hogeschool Engineering
[Submitted to]

Jeedella, Jeedella J.S.Y.
Medewerker, Electrical Engineering
j.jeedella@fontys.nl
R1 1.207
Fontys Hogeschool Engineering

Internship Student:

Bhonwal, Sparsh S.
(2223440) EE5

24-AUG-2016

Fontys Hogescholen
Department of Engineering

Final Report R2
Internship Assignment: Cyber Physical System (Model Based Control System)
Student: Bhonwal, Sparsh S. (2223440)
EE5 (3rd Year Internship) * 2015-2016

DOCUMENT HISTORY

Version	Date	Author	Changes
R0.1 (Draft)	08-SEP-2015	Bhonwal,Sparsh S. (2223440)	
R0.9 (Draft)	15-FEB*2016	Bhonwal,Sparsh S. (2223440)	
R1.0	20-FEB*2016	Bhonwal,Sparsh S. (2223440)	
R2	30-AUG-2016	Bhonwal,Sparsh S. (2223440)	
R2.1	30-AUG-2016	Bhonwal,Sparsh S. (2223440)	

DISTRIBUTION LIST

Version	R0.1 (Draft)	R0.9 (Draft)	R1	R2	R2.1
Jeedella, Jeedella J.S.Y.	✓	✓	✓	✓	✓
Mark Stappers	✓	✓	✓	✓	✓

X

Jeedella, Jeedella J.S.Y
University Tutor, Eletrical Engineering

X

Mark Stappers
Comany Supervisor, Mechatronics

X

Sparsh Bhonwal 2223440
Internship Student

CONTENTS

1. INTRODUCTION AND OBJECTIVES

1.1 GENERAL INTRODUCTION

This project was done as part of EE5 (5th semester, 3rd year internship) of Electrical and Electronic Engineering program of Fontys University of Applied Sciences, the internship project was done at Mechatronics department within Fontys Hogeschool Engineering. The project was done during first semester of Academic Year 2015-2016. The internship student Bhonwal, Sparsh S. (22234440) student at Electrical Engineering department worked under the supervision of Mark Stappers at Mechatronics department as his company supervisor. During this project Jeedella, Jeedella J.S.Y. was the university tutor from Electrical and Electronic Engineering department

1.2 ABSTRACT

The objective of this Internship project was to model a 4th order motion system with 2 degrees of freedom, then to design a PID controller for a position control of the system and finally to implement the controller over the EtherCAT network.

1.3 INTERNSHIP HOST (FONTYS MECHATRONICS AND ROBOTICS LAB)

Mechatronics and Robotics Lab (Lectoraat Mechatronics and Robotics in Dutch) is harbored in Fontys building R1, it is part of Mechatronics Department within Fontys Engineering. Mechatronics and Robotics Lab aims at doing improving the field on engineering. Mechatronics and Robotics lab does research for industries as well as for education. The research is done by expanding on the current knowledge the lab possesses as well as doing research with the collaboration with industry. The new knowledge gained and well as the new products developed with this research with directly integrated either into industry or in education. Here are some current projects undertaken by Mechatronics and Robotics Lab:

1. Rosbee
2. Composites in Mechatronics
3. Airhockey robot
4. Fontys FRC - team rembrandts
5. Low Cost Motion
6. IGUS robotic arm
7. Emotiv EPOC
8. IFM 3D-sensor
9. Stalker II
10. Outdoor Robot Platform
11. LabView programs
12. CIM FRF measurements
13. Objexlab
14. Mobile Robot Balance: ZMP

Reference: (Engineering, 2016)

2. FOURTH ORDER MODULE DESCRIPTION

2.1 INTRODUCTION

As described previously the aim on this internship project is to develop a position control system for a fourth order system. A fourth order module was provided for the use during this internship. This module is specifically designed to model/control a fourth order system; the module contains a mechanical fourth order system as well as the electronics components needed to produce a certain input to the system and well as read the output from the system.

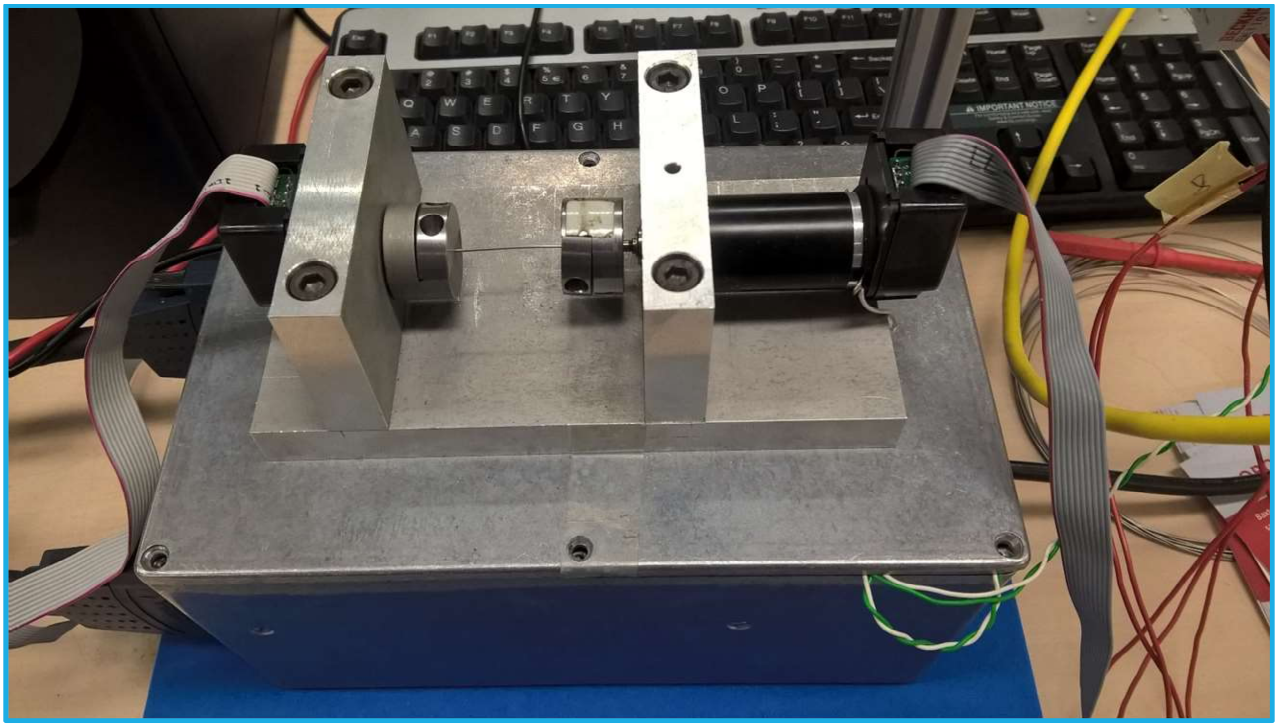


Figure 2-1 Picture the fourth order module used during this internship project

2.2 SYSTEM DESCRIPTION

2.2.1BLOCK DEFFINITION DIAGRAM STRUCTURE [FOURTH ORDER MODEL TOP LEVEL]

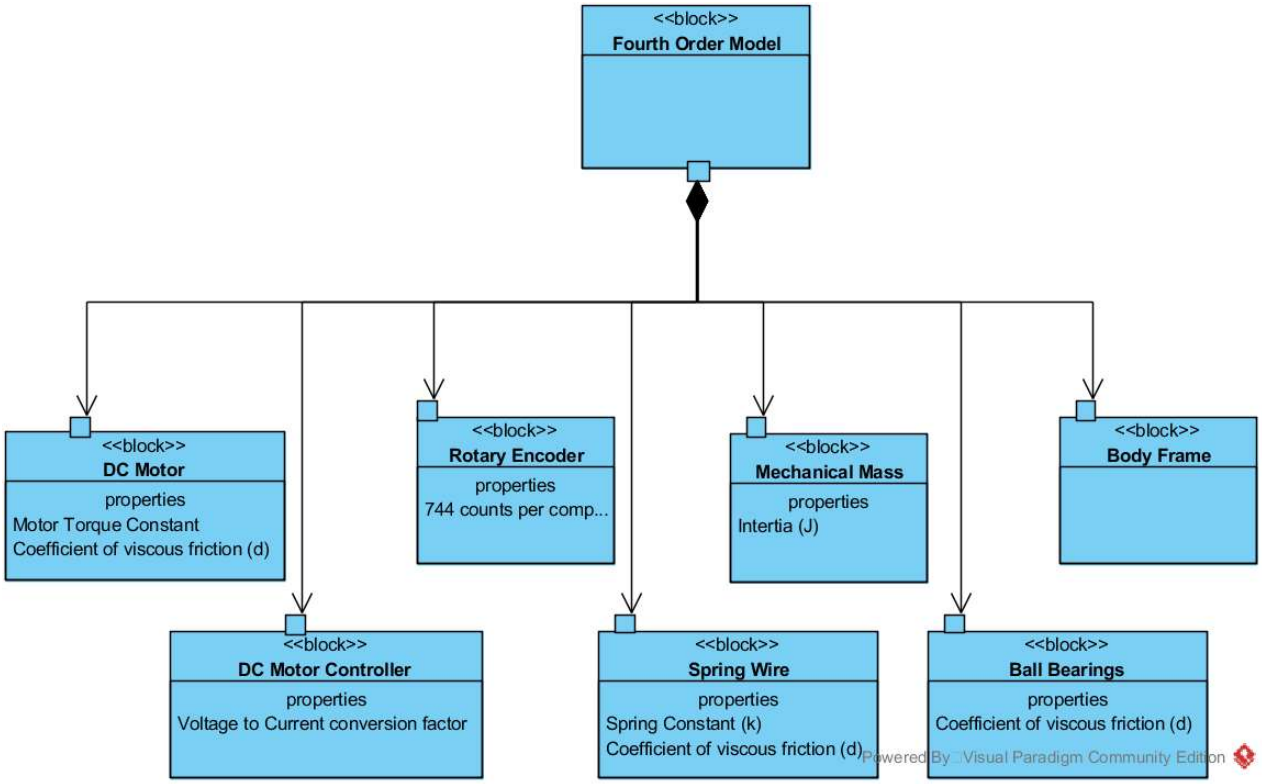


Figure 2a BDD Structure [Fourth Order Model Top level]

2.2.2INTERNAL BLOCK DIAGRAM FOURTH ORDER MODEL [PART CONNECTIONS]

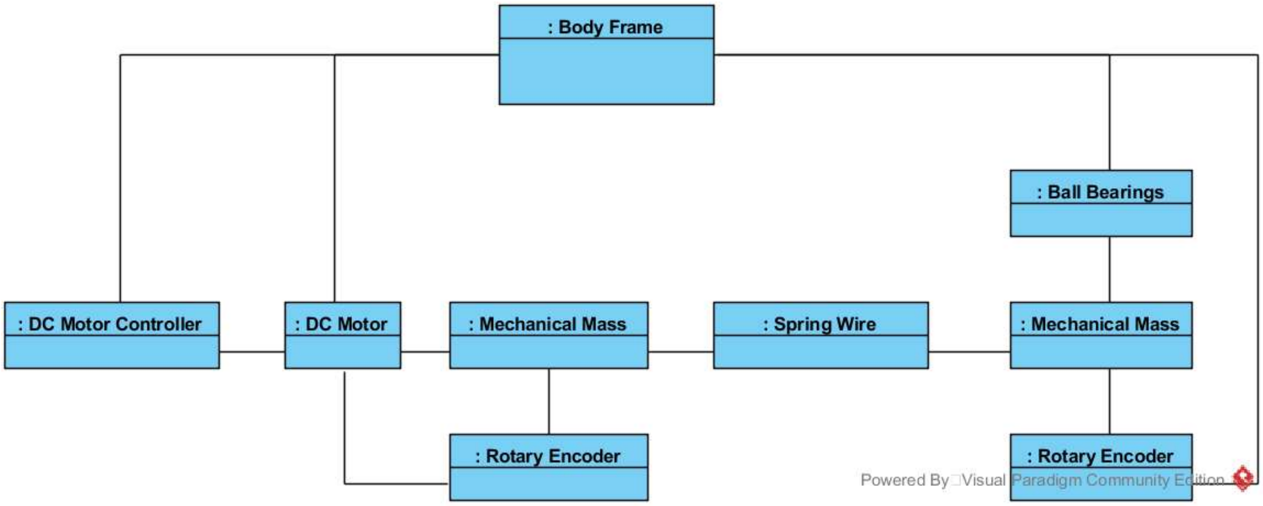


Figure 2b IBD Fourth Order Model [Part Connections]

2.2.3 INTERNAL BLOCK DIAGRAM FOURTH ORDER MODEL [ITEM FLOWS]

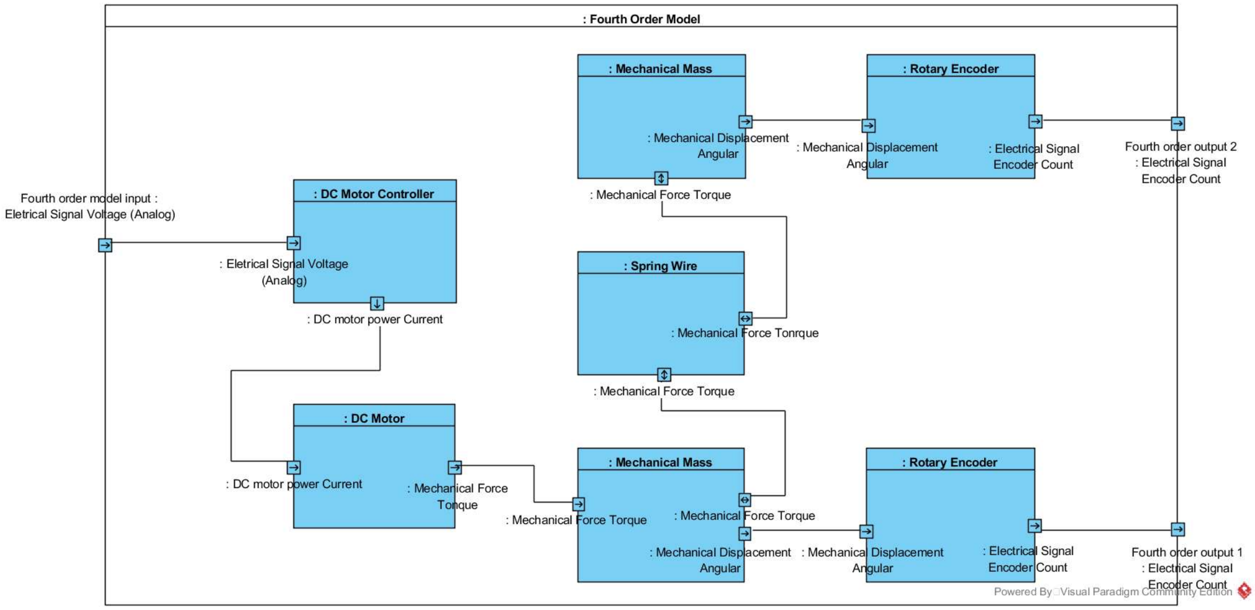

: Fourth Order Model
: Mechanical Mass
: Rotary Encoder
: Mechanical Displacement Angular
: Mechanical Displacement Angular
: Electrical Signal Encoder Count
Fourth order output 2 : Electrical Signal Encoder Count
: Mechanical Force Torque
Fourth order model input : Eletrical Signal Voltage (Analog)
: DC Motor Controller
: Eletrical Signal Voltage (Analog)
: DC motor power Current
: Spring Wire
: Mechanical Force Tonrque
: Mechanical Force Torque
: DC Motor
: DC motor power Current
: Mechanical Force Torque
: Mechanical Mass
: Mechanical Force Torque
: Mechanical Force Torque
: Mechanical Displacement Angular
: Rotary Encoder
: Mechanical Displacement Angular
: Electrical Signal Encoder Count
Fourth order output 1 : Electrical Signal Encoder Count
Powered By Visual Paradigm Community Edition

Figure 2c IBD Fourth Order Model [Item Flows]

3. SYSTEM IDENTIFICATION SETUP

3.1 INTRODUCTION

System Identification setup is used to identify a model for the fourth order model. For this we are going to a Simulink's, Simulink Real Time xPC setup. By this identification model can be created in Simulink, which provide proper input to excite the system (in this case a fourth order model) for identification and record the outputs from the system for identification purpose. This data is then transferred to the host PC for model identification, xPC is required in order to execute a model in real time, xPC target also provides dedicated hardware in order to interface with the system. In this NI 6703 card (Instruments, NI PCI-6703 Static Analog Voltage Output -- 16-Bit, 2016) is used as an analog output and input to the fourth order model and NI 6602 (Instruments, NI PCI-6602 8-Channel Counter/Timer with Digital I/O, 2016) counter card is used for reading the encoder signal from the fourth order model.

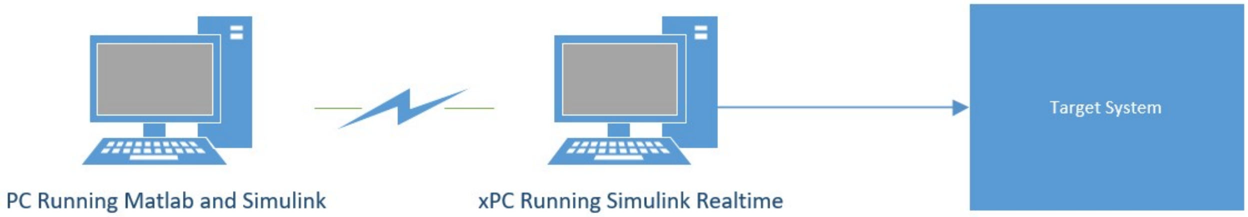


Figure 3a System Identification Setup

4. OBTAINING TRANSFER FUNCTION

4.1 INTRODUCTION

The last two chapters aimed are describing the hardware and software which is used to obtaining a model in form of a transform function of the fourth order model. Those hardware and software are used mainly for giving a certain input the fourth order model and measuring the output. This chapter aims on describing the techniques used to analyses those measurements in order to obtain a transfer function of the fourth order model.

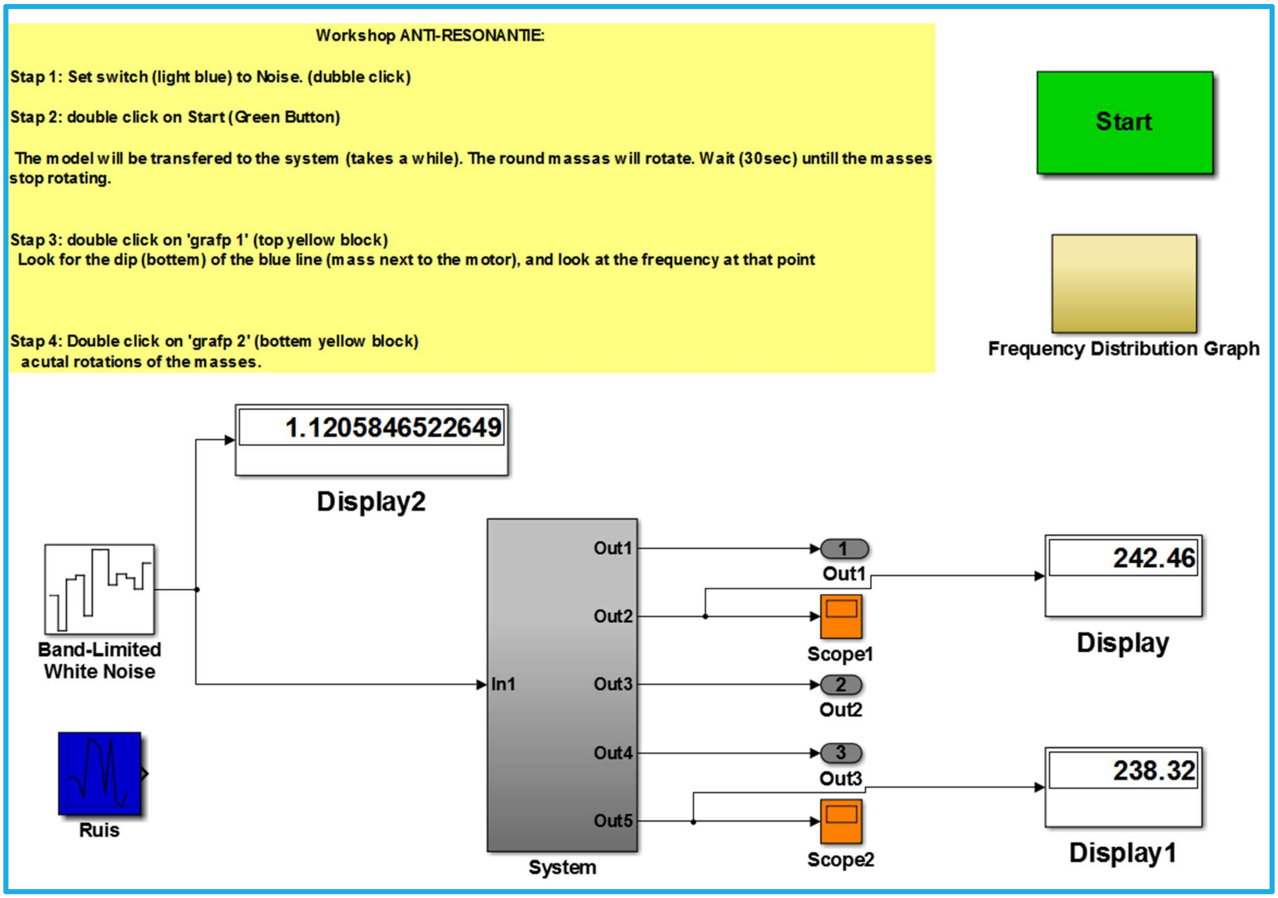
First we will look at the System Identification using black box model and frequency domain decomposition. Then we will go on and try to verify the Transfer Function against the real world system and come across the limitations of this type of system identification. Therefor we will on to the System identification using grey box model, but before we can do that we need to derive some equations for the fourth order system that we are trying to obtain the Transfer Function for. Finally, we will try to verify the Transfer function obtained by this method of system identification by comparing it with real world physical system.

4.2 OBTAINING EXPERIMENTAL DATA

4.2.2 OBTAINING FREQUENCY RESPONSE

Below Simulink model is provided in order to obtain the frequency response of the fourth order system as described in the Chapter 2, this model is compiled on the Simulink System and is then transferred on the Simulink Real Time Target for execution is real time as described in Chapter 3.

4.2.3 OVERALL SIMULINK MODEL



Workshop ANTI-RESONANTIE:

Stap 1: Set switch (light blue) to Noise. (dubble click)

Stap 2: double click on Start (Green Button)

The model will be transfered to the system (takes a while). The round massas will rotate. Wait (30sec) untill the masses stop rotating.

Stap 3: double click on 'grafp 1' (top yellow block)
Look for the dip (bottem) of the blue line (mass next to the motor), and look at the frequency at that point

Stap 4: Double click on 'grafp 2' (bottem yellow block)
acutal rotations of the masses.

Figure 4a Overall Simulink Model

4.2.4 System block

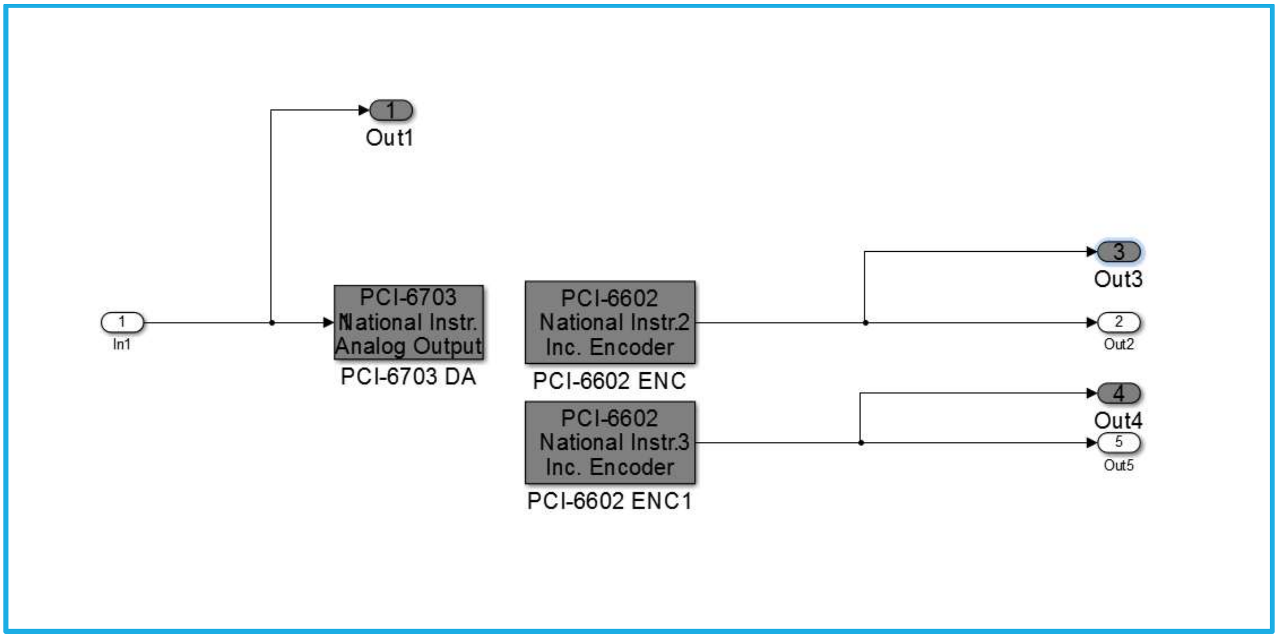


Figure 4b System Block

It is clear from the above Simulink Models (Figure 4-a and 4-b) that the noise is an input to the output Analog card on the Real Time target system which provides the analog input signal to the motor driver. The output of the fourth order model is registered using two rotatory encoders, as the model has two degrees of freedom therefore two encoders are used to register each motion, these encoders are the paired with encoder cards on the Real Time system. The details about this system has been provided in chapter 2.

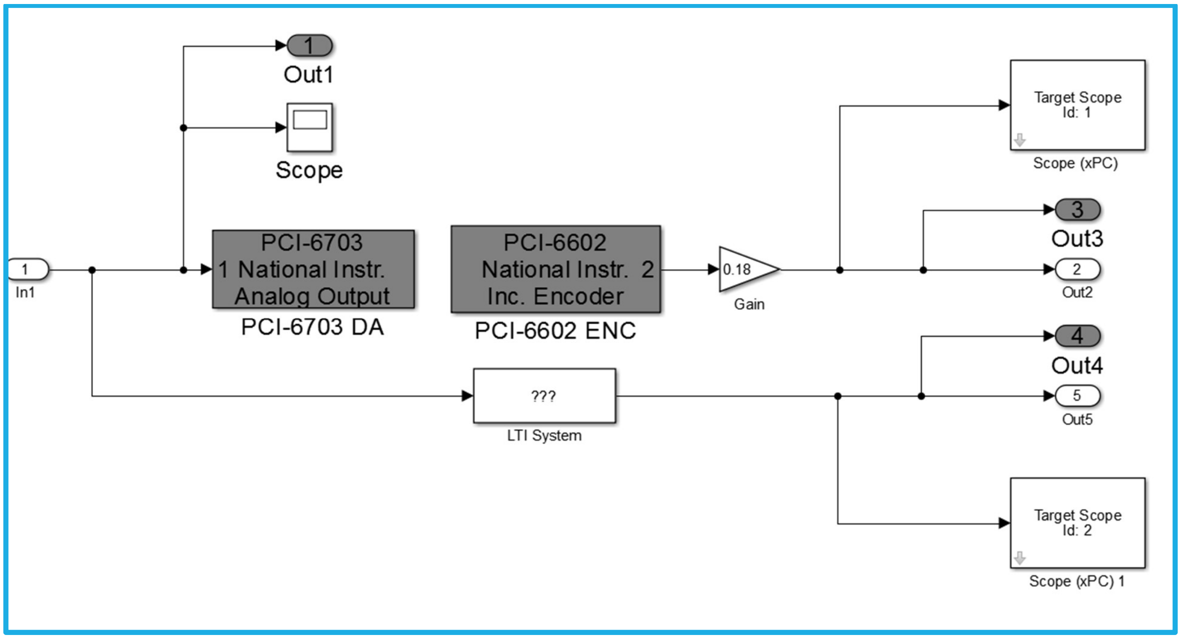
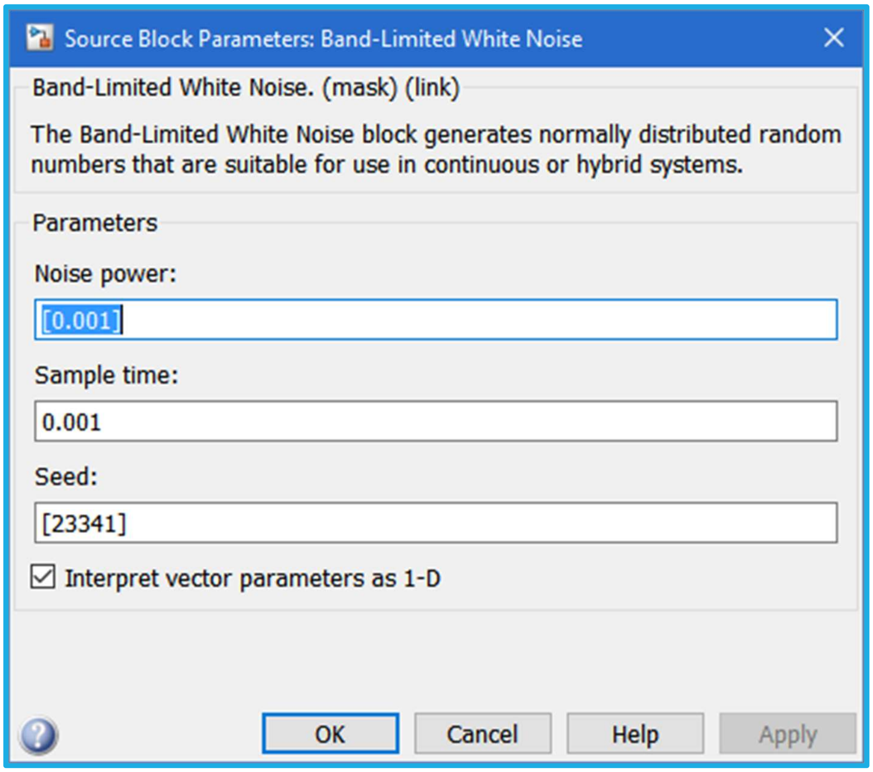


Figure 4c Modified System Blcok

As shown in Figure 4-b the output of the encoders is directly recorded without any manipulation, however it is necessary to multiply the output of the encoders by a factor of 0.18 is order to obtain the degree of rotation in degrees as, 2000 encoder counts correspond to 360 degree of rotation (Chapter 2 for further details), this is implemented as shown in Figure 4-c.

4.2.5 COFIGURATION BAND-LIMITED WHITE NOISE BLOCK



Band-Limited White Noise. (mask) (link)

The Band-Limited White Noise block generates normally distributed random numbers that are suitable for use in continuous or hybrid systems.

Parameters

Noise power: [0.001]

Sample time: 0.001

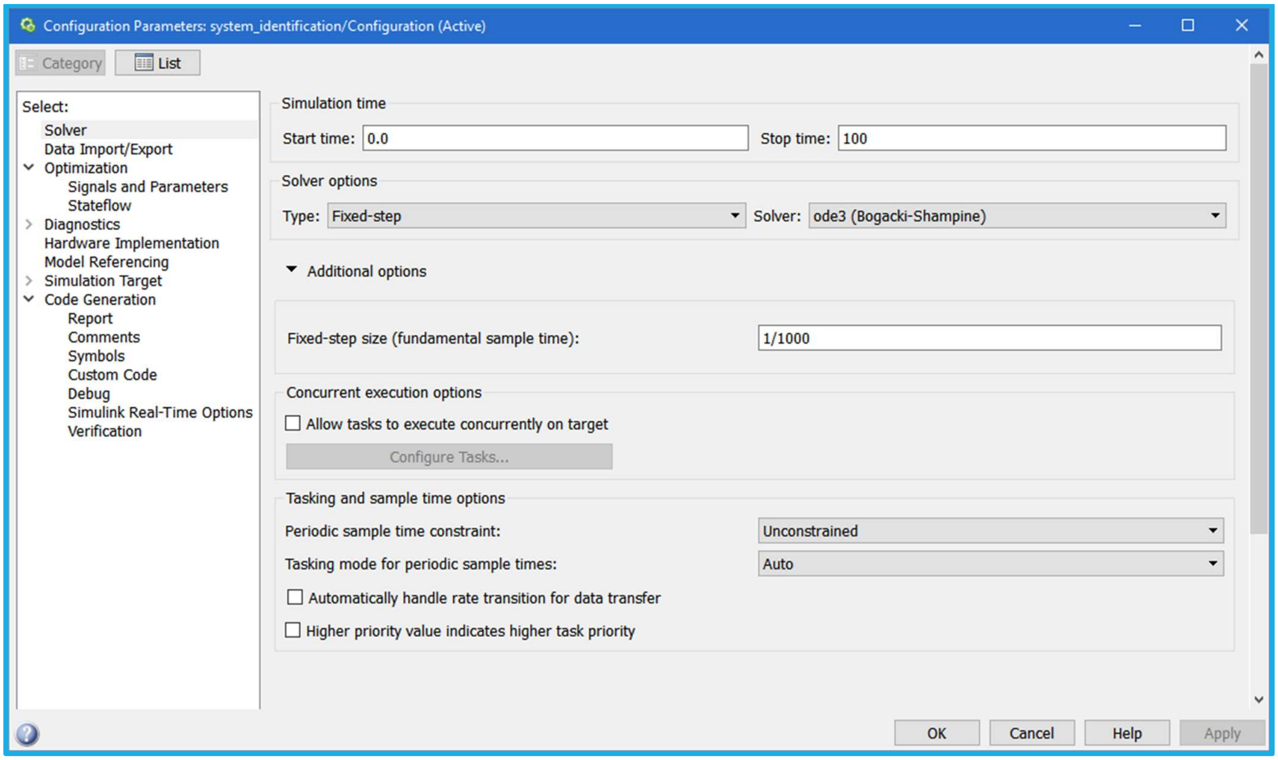
Seed: [23341]

☑ Interpret vector parameters as 1-D

Figure 4d Noise Block configuration

Figure 4-d shows the configuration of the Band-Limited white noise block, this the block which generate the noise which is them applied as input to the fourth order system. The noise power value is obtained experimentally, it should be high enough that peaks in the graph are shown clearly however not so high that is break the system, especially the flexible parts of system. Sample time value corresponds to the real time model execution time which is 1000 times per second, as we will see further.

4.2.6 REAL TIME MODEL COFIGURATION AND CODE GENERATION CONFIGURATION



Configuration Parameters: system_identification/Configuration (Active)

Simulation time

Start time: 0.0
Stop time: 100

Solver options

Type: Fixed-step
Solver: ode3 (Bogacki-Shampine)

Fixed-step size (fundamental sample time): 1/1000

Concurrent execution options

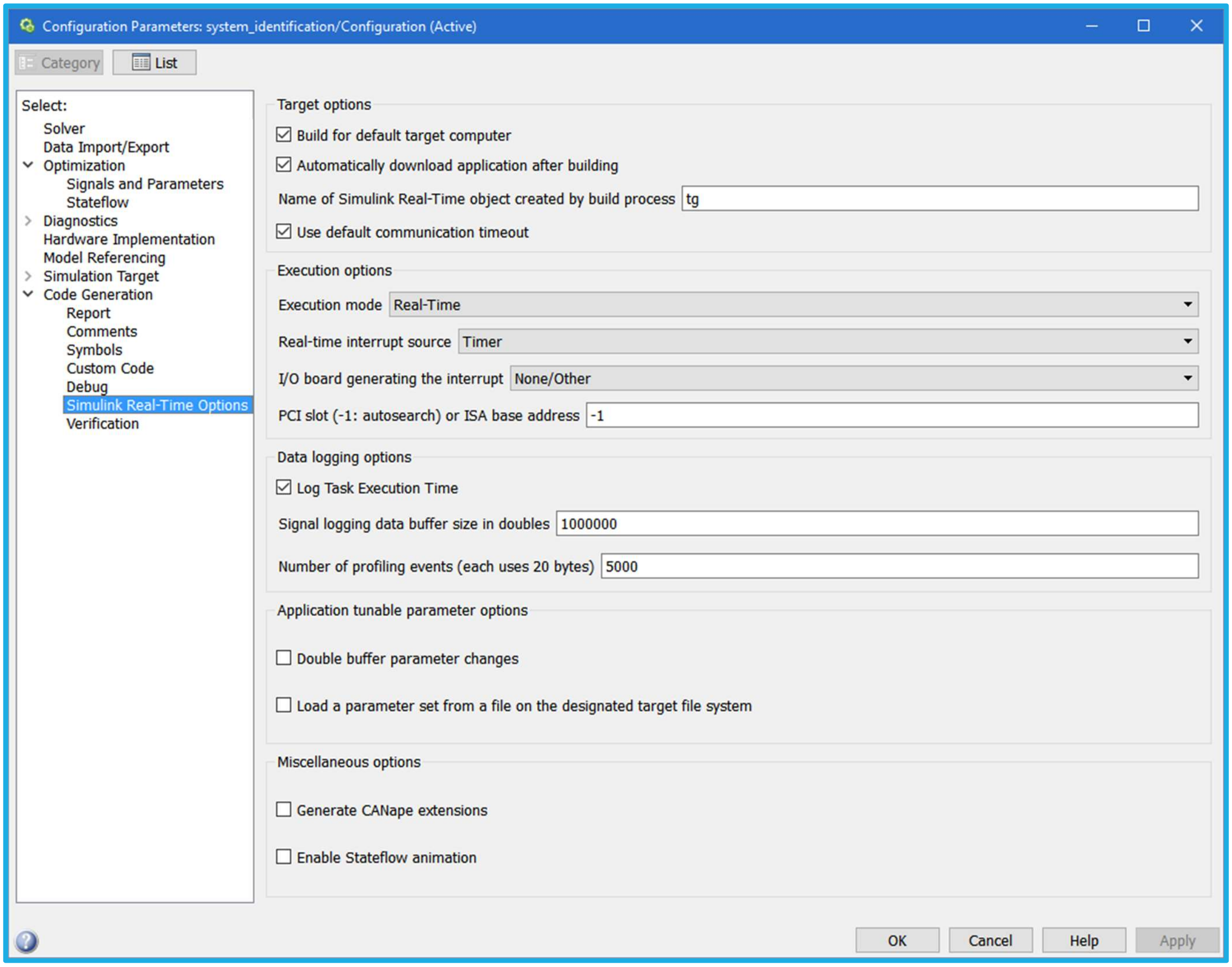
☐ Allow tasks to execute concurrently on target

Tasking and sample time options

Periodic sample time constraint: Unconstrained
Tasking mode for periodic sample times: Auto
☐ Automatically handle rate transition for data transfer
☐ Higher priority value indicates higher task priority

Figure 4e

As the Figure 4-e show the step size of the real time module is set to run at 1000 times every seconds, and with every step data from the encoders is logged in and out of module is updates, also the whole module is re-evaluated as well.



Target options

☑ Build for default target computer

☑ Automatically download application after building

Name of Simulink Real-Time object created by build process: tg

☑ Use default communication timeout

Execution options

Execution mode: Real-Time

Real-time interrupt source: Timer

I/O board generating the interrupt: None/Other

PCI slot (-1: autosearch) or ISA base address: -1

Data logging options

☑ Log Task Execution Time

Signal logging data buffer size in doubles: 1000000

Number of profiling events (each uses 20 bytes): 5000

Application tunable parameter options

☐ Double buffer parameter changes

☐ Load a parameter set from a file on the designated target file system

Miscellaneous options

☐ Generate CANape extensions

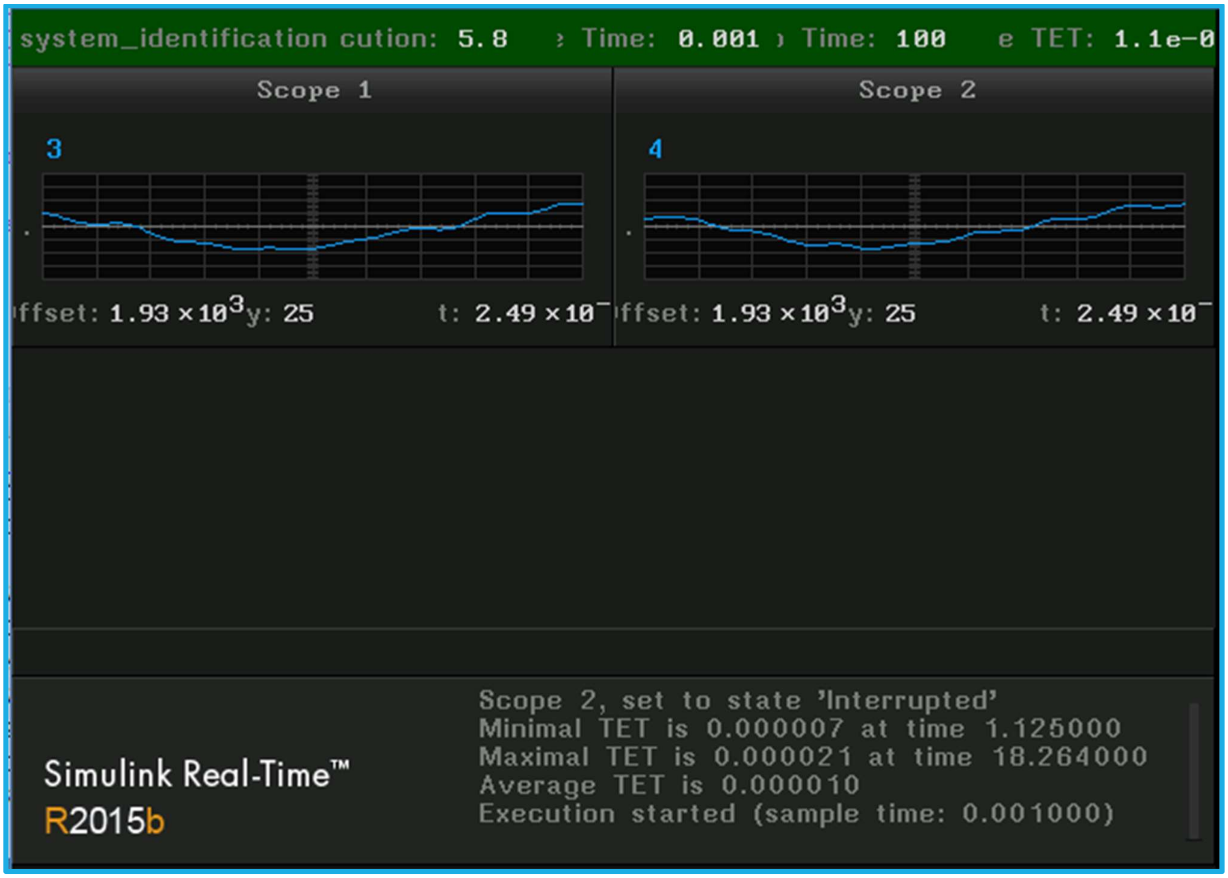
☐ Enable Stateflow animation

Figure 4-f

As figure 4-f shows the data buffer size (Data logging option -> Signal logging data buffer size in doubles) is chosen to be 100 time 1000, as discussed above the real time target records data from the encoders 1000 times every seconds, and total time for which the target run is 100 seconds. Therefore, every signal that is being logged would have buffer size of (100 times 1000), which in our case is two encoders and one analog output.

4.2.7 RUNNING THE SYSTEM

The module is then executed on the Simulink real time target, as described in Chapter 3



```
system_identification cution: 5.8  › Time: 0.001 › Time: 100   e TET: 1.1e-0
```

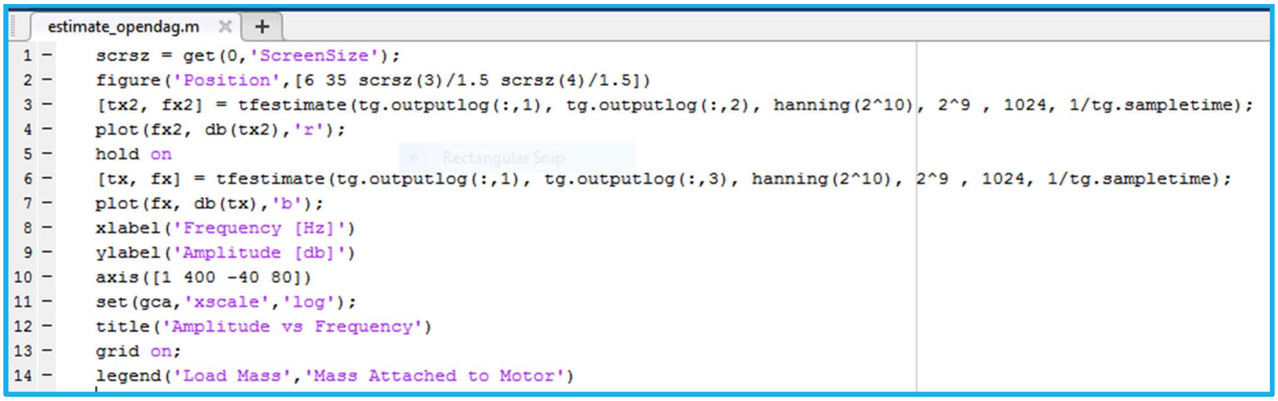
```
Scope 2, set to state 'Interrupted'
Minimal TET is 0.000007 at time 1.125000
Maximal TET is 0.000021 at time 18.264000
Average TET is 0.000010
Execution started (sample time: 0.001000)
```

Figure 4g

The Figure 4-g show the Simulink Target PC screen as the model is executed on it.

4.2.7 OBTAINING DATA AND PLOTTING FREUQENCY DOMAIN GRAPHS AFTER THE MODEL EXECUTION

Double clicking in the Frequency distribution graph block as show in Figure 4-a, executes the following Matlab script (Figure4-h):



```
scrsz = get(0,'ScreenSize');
figure('Position',[6 35 scrsz(3)/1.5 scrsz(4)/1.5])
[tx2, fx2] = tfestimate(tg.outputlog(:,1), tg.outputlog(:,2), hanning(2^10), 2^9 , 1024, 1/tg.sampletime);
plot(fx2, db(tx2),'r');
hold on
[tx, fx] = tfestimate(tg.outputlog(:,1), tg.outputlog(:,3), hanning(2^10), 2^9 , 1024, 1/tg.sampletime);
plot(fx, db(tx),'b');
xlabel('Frequency [Hz]')
ylabel('Amplitude [db]')
axis([1 400 -40 80])
set(gca,'xscale','log');
title('Amplitude vs Frequency')
grid on;
legend('Load Mass','Mass Attached to Motor')
```

Figure 4h

To describe in brief, the function of above MATLAB script if to feed the discrete time domain data (samples) acquired during the above steps, to MATLAB's tfestimate function, which then compares the input and output of the system to determine the response of the system in frequency domain, or the response as the function of frequency, the data is also filtered using the hanning window which filters out the high frequency noise (The MathWorks, Inc., 2016). The data obtained from the tfestimate function is then plotted on a graph to obtain the response on the system in frequency domain.

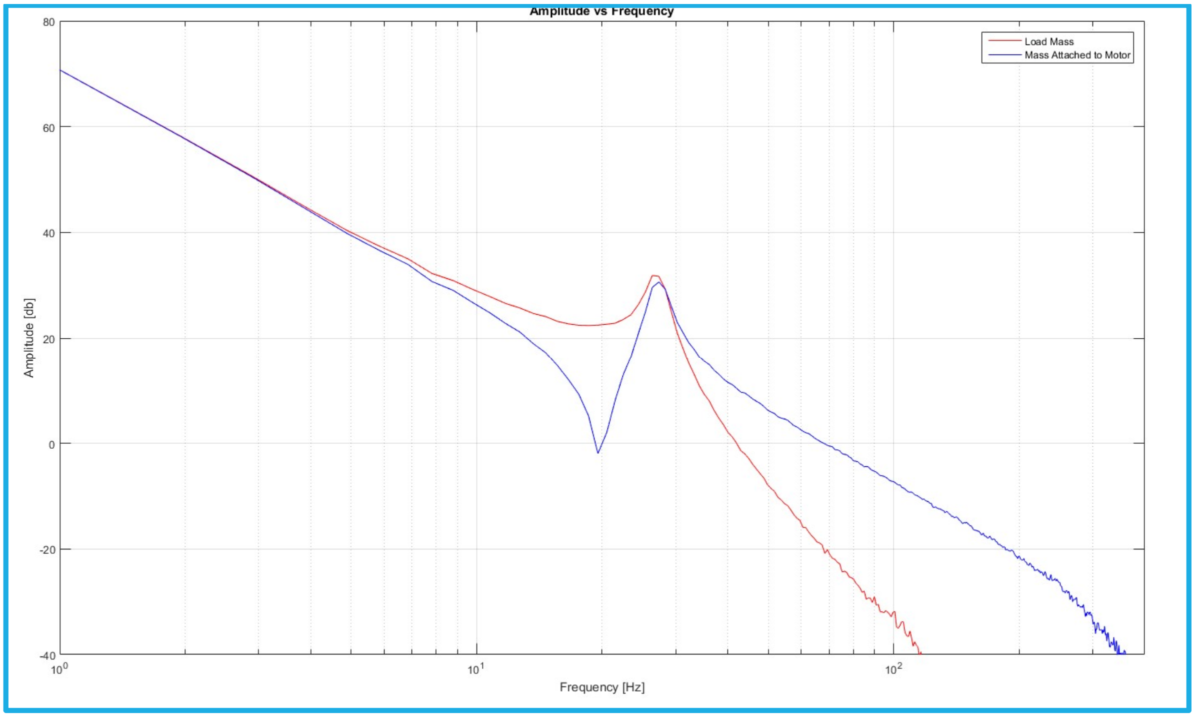


Figure 4-i

The running the above script (Figure 4-j) produces the following frequency response graph of the system (Figure 4-i).

5.3 PRELIMINARY RESEARCH

As we know from the briefing of this project that we would be working on a fourth order system. Therefore, we do a preliminary research on typical fourth order transfer function and their frequency domain graphs. A bulk of this research comes from (Coelingh, 2000), and other documents summarizing and refereeing to that source (Borger, 2010).

Based on the locations of the actuator and sensor, five different types of fourth order plant transfer functions can be characterized (Coelingh, 2000) (Borger, 2010):

1. Type AR
2. Type D
3. Type RA
4. Type R
5. Type N

For the purpose of this project only Type AR and Type R, types of a fourth order system are relevant as we will see later.

Type R, Fourth Order System transfer function: This plant transfer function is characterized by the fact that the anti-resonance frequency is situated at infinity in the s-lane. This zero pair is situated outside the scope of the model and therefore can be neglected. This results that the fourth order transfer function has no zero pair. The transfer function of a type R in the frequency domain notation is:

$$\boxed{P(j\omega) = \frac{x(j\omega)}{F(j\omega)} = \frac{1}{m(j\omega)^2} . \frac{\omega_r^2}{((j\omega)^2 + \omega_r^2)} \text{ , } \omega_{ar} \to \infty}$$

Equation 4-a Transfer Function of TYPE R, fourth order system

Bode Diagram of the transfer function of a Type R Fourth Order System (above equation) is:

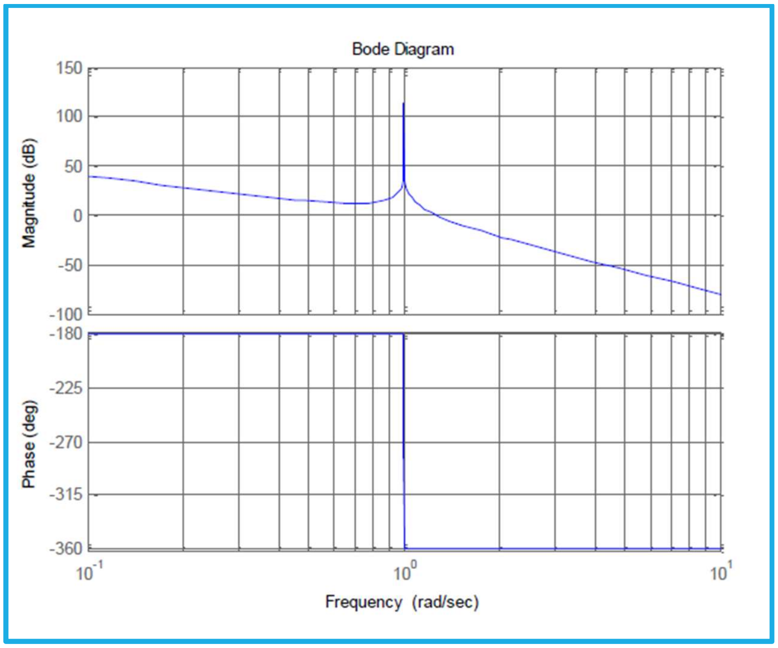


Figure 4-j

Type AR, Fourth Order System transfer function: This type of fourth order system's character is that anti-resonance frequency is lower than the resonance frequency. This results that the conjugated zero pair is located closer to the origin than the conjugated pole pair in the s-plane. The transfer function of a type AR in the frequency domain notation is:

$$P(j\omega)=\frac{x(j\omega)}{F(j\omega)}=\frac{1}{m(j\omega)^2}\cdot\frac{((j\omega)^2+\omega_{ar}^2)}{((j\omega)^2+\omega_r^2)}\cdot\frac{\omega_r^2}{\omega_{ar}^2}\text{, with }\omega_{ar}<\omega_r$$

Equation 4-b Transfer Function of TYPE R, fourth order system

Bode Diagram of the transfer function of a Type AR Fourth Order System (above equation) is:

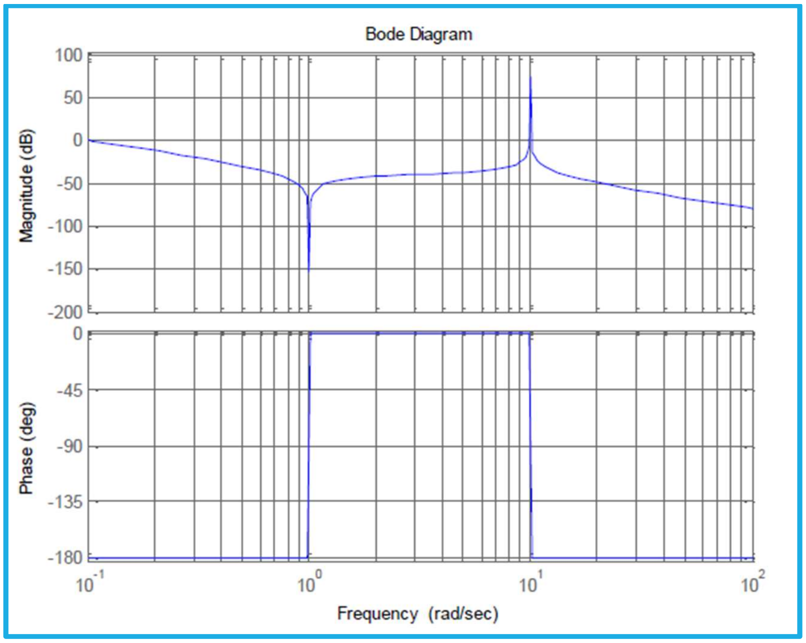


Figure 4-k

For now, let us assume that the movement of load mass in Figure 4-i represents a Type R fourth order transfer function, and mass attached to motor be represented by Type AR fourth order transfer function. We will verify this assumption when we will break down the frequency domain components of the frequency domain graph (Figure 4-i) to obtain the transfer function in the next step.

4.4 OBTAINING TRANSFER FUNCITON USING BLACK BOX MODEL AND FREQUENCY DOMAIN DECOMPOSITION

4.4.1 INTRODUCTION

This system identification technique takes care of absolutely nothing about the internal structure of the system, therefore considering system as a complete black box. An input containing a range of frequencies is applied to the system (in this a random noise), and the frequency response of the system is analyzed to obtain the transfer function of the system.

The method used below to described below to experimentally obtain a transfer function are based on the following sources (Borger, 2010) (Nise, 2011).

Theory: The method for obtaining the transfer function from frequency is based upon breaking a complex frequency response into simple frequency responses which are simple transfer functions which altogether makes up total transfer function which we are trying of obtain.

First we are going to find the transfer function for mass which has simpler function, this case it would be Load Mass from Figure 4-i. We also know that the transfer function we are going to obtain represents Position of mass (which recorded by the encoder) over the Voltage applied to the system.

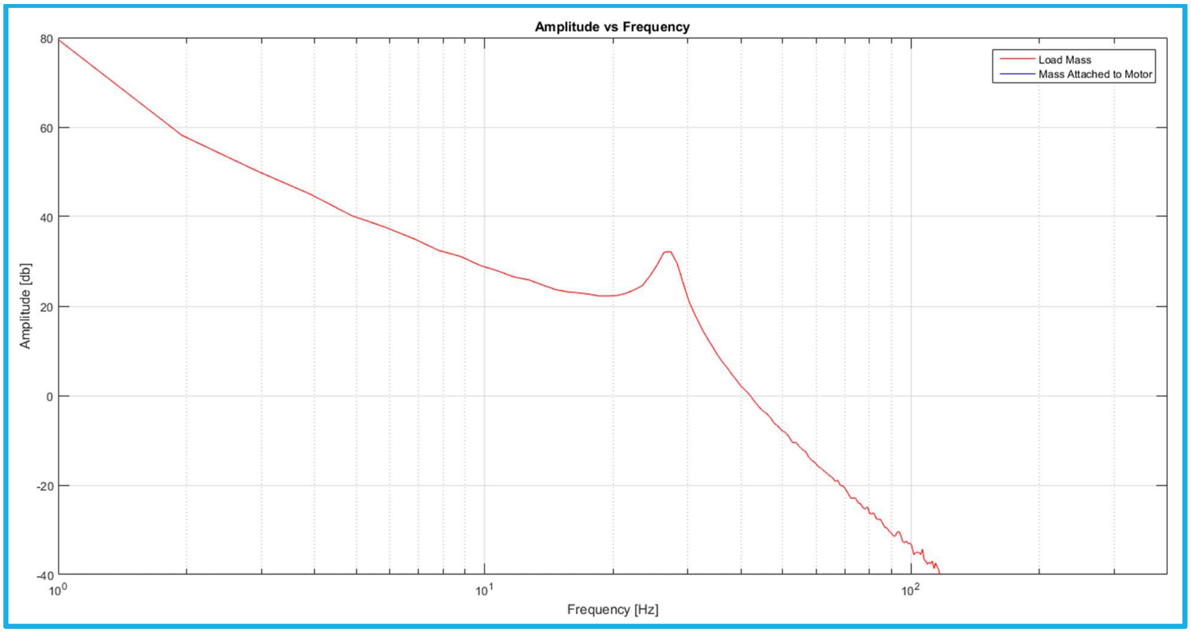


Figure 4l

Step 1: First Step is to analyze the slopes of the above frequency response, especially the slope at the starting and at the end of the experimental frequency response, the slope at the starting of the graph could be used to find the presence of an integrator function, by looking at the end of the frequency response we can know if the total transfer function has more number of pole or more number of zeroes and how many.

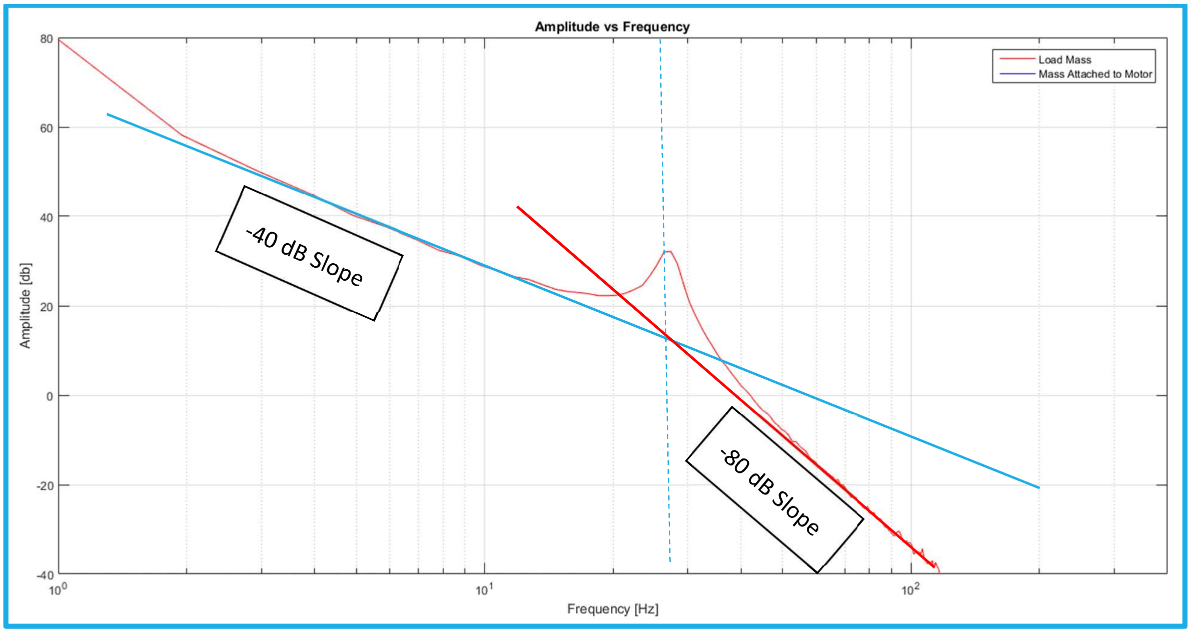


Figure 4m

As we can see from above figure (figure-4m) the frequency response starts with a slope of -40dB indicating a presence of a double integrator function ($\frac{1}{s^2}$), as the frequency approaches a frequency of about 27 Hz there is a resonant peak and slope changes to -80dB indicating a presence of a resonant second order function ($\frac{\omega_n}{s^2+2\zeta\omega_n s+\omega_n^2}$). Therefore, we can get a general structure for our transfer function with unknown parameters

$$T(s) = \frac{1}{s^2} \times \frac{\omega_n}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

STEP 2: Next we determine the gain of the Transfer Function by the manually trying to overlay an integrator function over the frequency response obtained from the experimental method.

STEP 3: Next we fill in the values of ω_n for the frequency of the resonant people from the graph for now we can leave the value of ζ to 1 which is the damping ration, by doing this we end up with a transfer function whose frequency response graph has same start and end slop as the experimental frequency response, as well as the resonant frequency matches.

STEP 4: Finally, we adjust the value of $\zeta$to obtain a transfer function who shape almost matches the experimental frequency response.

After following the above 4 steps we get following transfer function:

$$\frac{\theta(s)}{V(s)} = \frac{2.64 \times 10^9}{s^4 + 11 \times s^3 + 27950 \times s^2} \left[\frac{deg}{V}\right]$$

The figure below shows the frequency response of this transfer function vs. the frequency response from experimental data.

Now we have completely obtaining the transfer-functions only using the experimental data, using the black box model, nest we need to verify these models, we will do this by several methods in the next section.

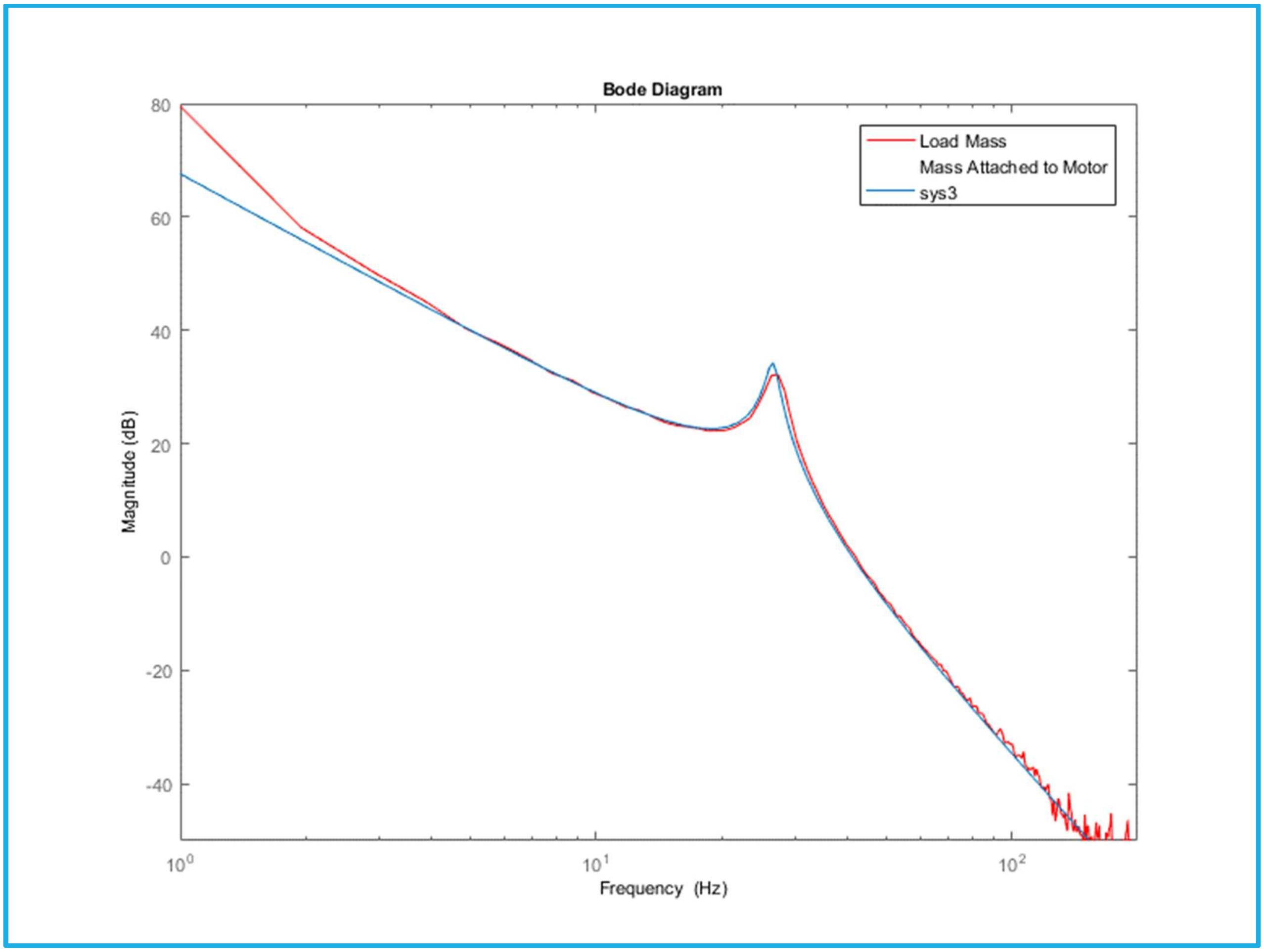


Figure 4n

In the above figure the blue line represents the bode plot of the transfer function obtained by following above steps and the red line is the frequency response obtained from the experimental data.

This confirms our assumption that we made in the last section that a TYPE R fourth order function (with added damping in the second order function) represents the characteristics of load mass.

Next we are going to try to find the transfer function for the mass attached to motor. Again it's the Position of mass over voltage applied to the system.

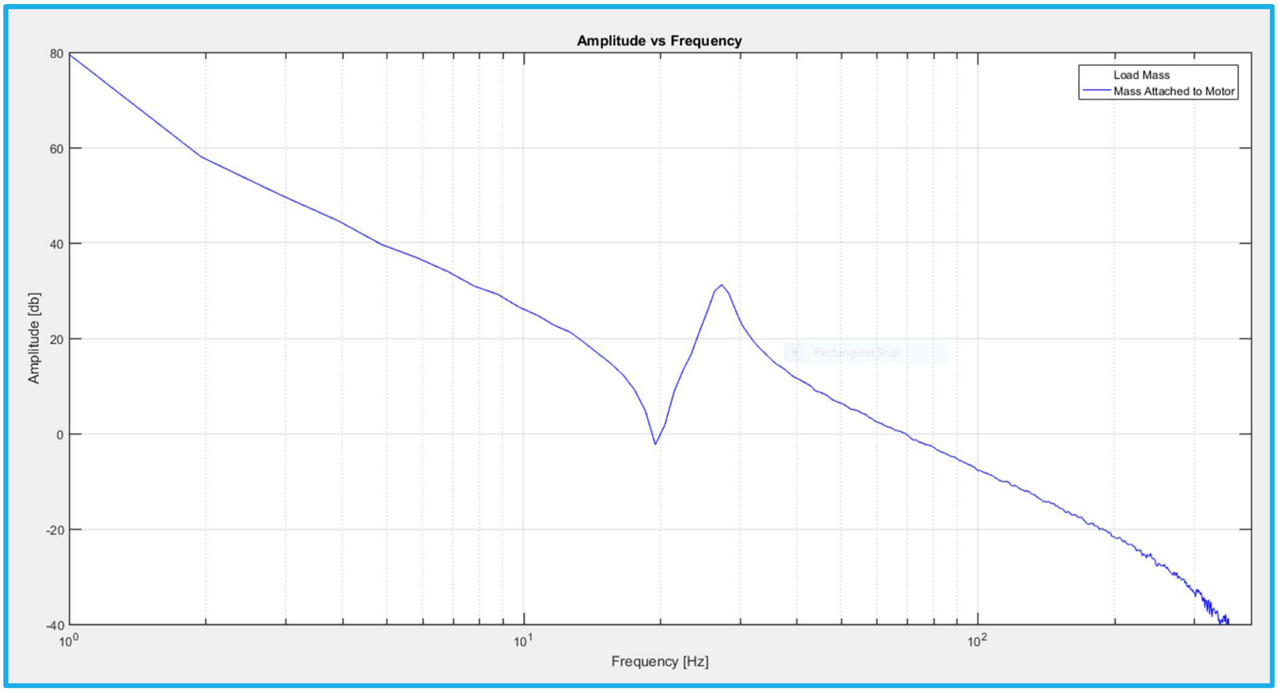


Figure 4o

STEP 1: Again we are going to analyze the starting and the ending slope of the experimental frequency response.

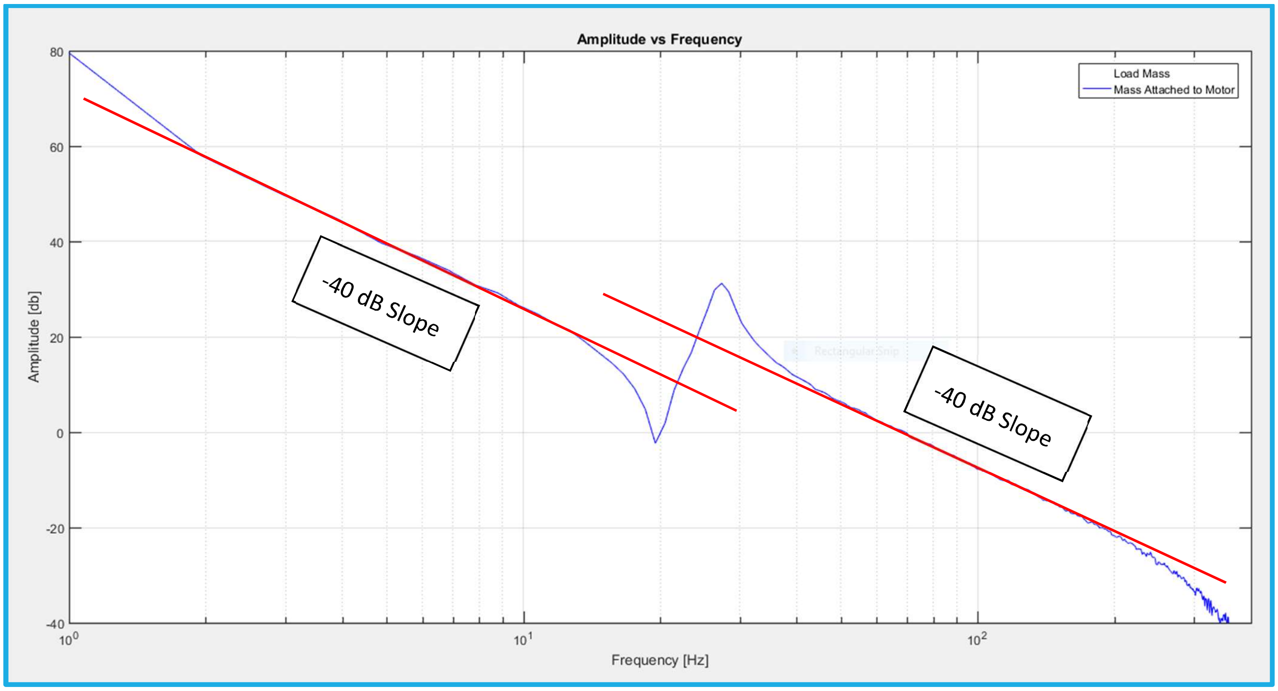


Figure 4p

As we can see from the abode figure (figure-4p) that the frequency response has the same starting and send slope of -40dB indicating equal number of poles and zeroes, with two additional poles from the double integrator function ($\frac{1}{s^2}$), in this response we see an additional resonant peak at about 20 Hz in addition to the one at 27 Hz only this one is inverted. This presence of additional inverted second order function explains why there is no change in the overall slope of the experimental frequency response. Therefore, we deduce that transfer function of this same as the one before, with the addition of an inverted second order function. Therefore, we can get a general structure for our transfer function with unknown parameters

$$T(s) = \frac{1}{s^2} \times \frac{\omega_r}{s^2 + 2\zeta\omega_r s + \omega_n^2} \times \frac{s^2 + 2\zeta\omega_{ar} s + \omega_n^2}{\omega_{ar}}$$

STEP 2: Next we determine the gain and other unknown parameters by overlaying and fine tuning as we did in the step for the last transfer function.

STEP 3: We add the anti-resonance frequency at about 20Hz and resonance at about 27Hz (same as the last transfer function).

After following these steps, we get the following transfer function:

$$\frac{\theta(s)}{V(s)} = \frac{2.64 \times 10^9 \times s^2 + 2.904 \times 10^{10} \times s + 3.976 \times 10^{13}}{15059s^4 + 165649 \times s^3 + 4.209 \times 10^8 \times s^2} \quad \left[\frac{deg}{V}\right]$$

The figure below shows the frequency response of this transfer function vs. the frequency response from experimental data.

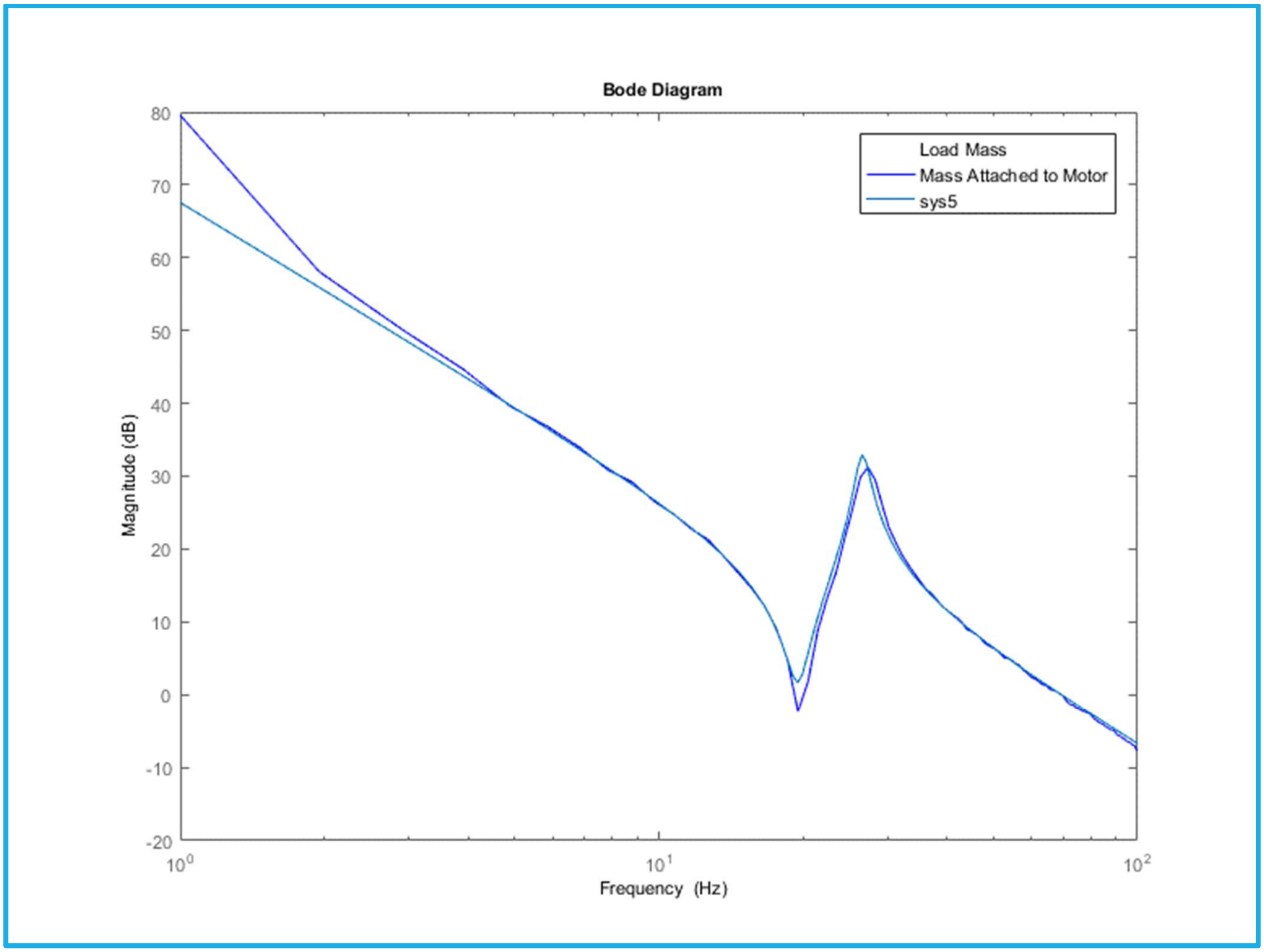


Figure 4-q

In the above figure (4-q) the light blue line represents the bode plot of the transfer function obtained by following above steps and dark blue line is the frequency response obtained from the experimental data.

This confirms out assumption and frequency response of Mass attached to motor can be represented by TYPE AR fourth order function (with added function).

4.5 VERIFYING THE MODELS OBTAINED BY BLACK BOX MODEL

From now on we are only going to focus on the transfer function of load mass, i.e. the mass which is attached not directly to the motor but through the spring, and is connected to encoder 2, because this is the mass that we intend on controlling though PID in the end.

We are going to use two methods for verifying the models (transfer functions) we have obtained so far, first by using Simulink and other by use System Identification Toolbox.

4.5.1 VERIFICATION ATTEMPT BY SIMULINK

Here we are going to use the same Simulink model that we used previously to obtain the experimental data, however with slight modification, we are going to replace the output of one of the encoder with the transfer function we previously estimated. So that we can compare the response of the transfer function and the actual hardware side by side.

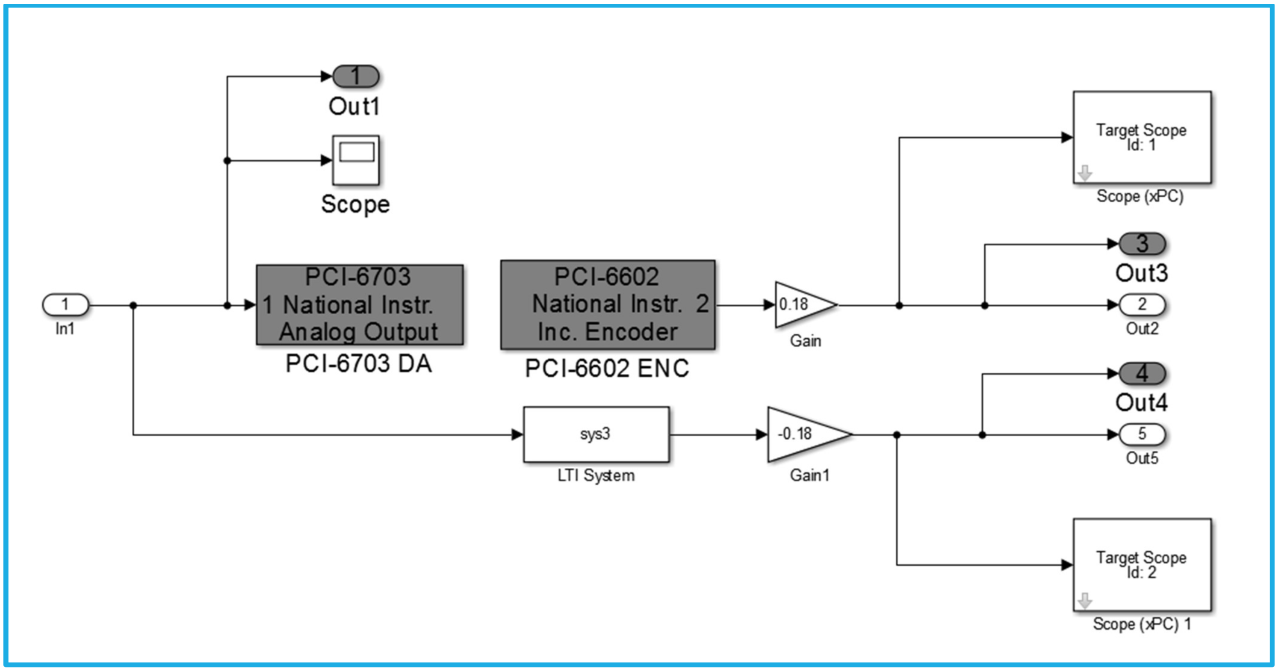


Figure 4-r

As we can see from the above figure (4-r) we have replaced of the output of the one of the encoder with a LTI System block, in this block we are going to enter the transfer function we intend of verifying, in this case

$$\text{sys3} = \frac{\theta(s)}{V(s)} = \frac{2.64\times10^{9}}{s^{4}+11\times s^{3}+27950\times s^{2}} \quad \left[\frac{deg}{V}\right]$$

the system input (random noise) is fed to both the actual hardware and the estimated model.

From the above model we get the following result:

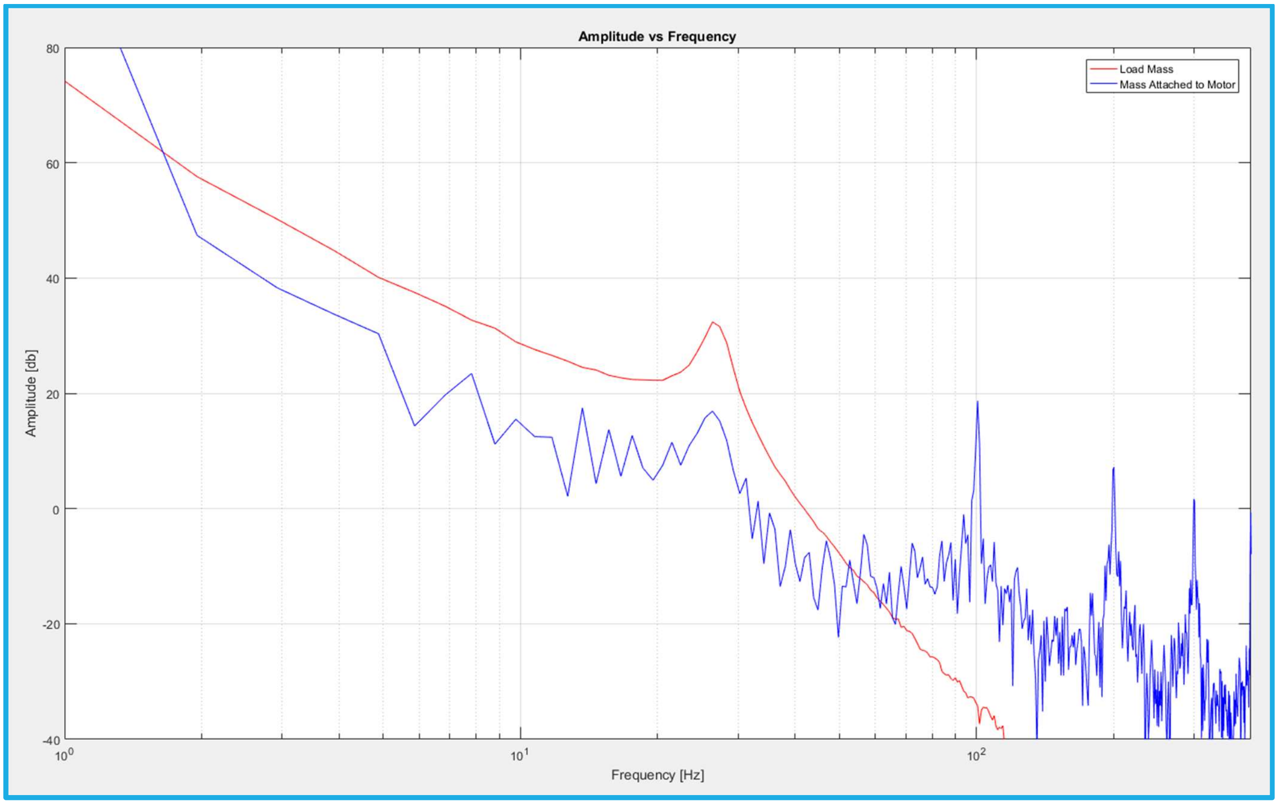
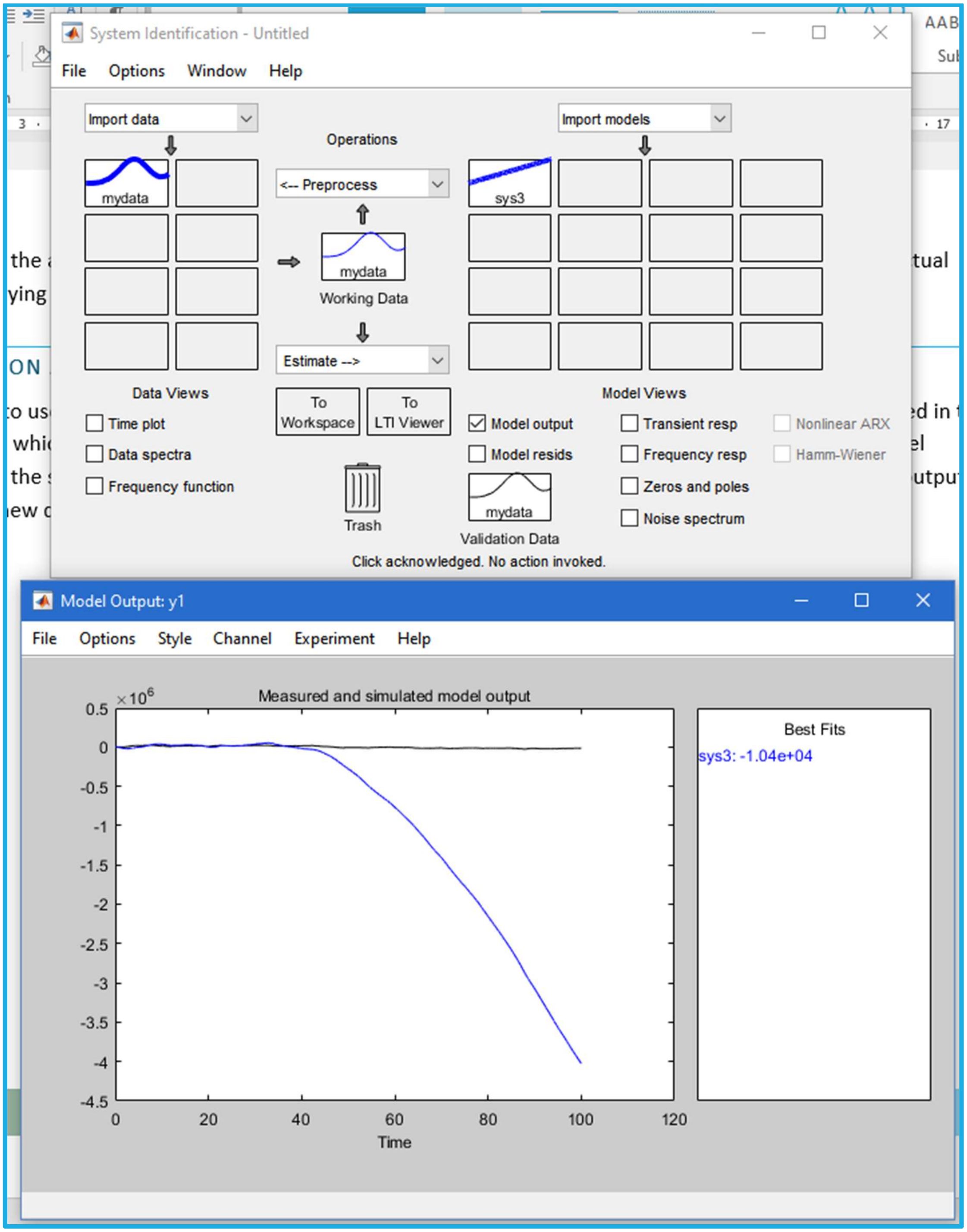


Figure 4-s

As we can we from the above figure (4-s) that the estimated transfer function (in blue) not barely represents the actual hardware we are trying to represent (in red).

4.5.2 VERIFICATION ATTEMPT BY SYSTEM IDENTIFICATION TOOLBOX

Next we are going to use System Identification Toolbox for verification. In the System Identification Toolbox, we feed in the Time Domain Data, which is the new data obtained by running the model again, and we feed in the estimated model (transfer function), the system identification toolbox shows how well the estimated model predicts this input and output correlation in this new data.



☑ Model output

Click acknowledged. No action invoked.

sys3: -1.04e+04

Figure 4t

As we can see from Figure (4-t), results from System Identification Toolbox is pretty bad as well, the correlation between the new real data and estimated is model is -1.04e-04% which means that there is almost no correlation and if any the difference between only increases as the time passes. Because the prediction made by estimated model keeps on deviating from real data.

5. IMPROVING TRANSFER FUNCITON USING GREY BOX MODELLING

5.1 INTRODUCTION

We have seen the results from black box modelling, the transfer function obtained by that method is highly in-accurate. Therefore, now we are going to obtain a theoretical model of the system, by laws of motion to different components of the systems and arrive to a model for overall system. Next we are going to compare this theoretical model with the model we obtained using black box modelling look for some major in-consistencies such as missing terms.

5.2 MODEL DECOMPOSITION

The First step in creating a theoretical model is to breakdown the system to be modelled into basic mechanical components. When we brake the 4th order model into basic mechanical components such as Mass, Spring and Damper we obtain the following model:

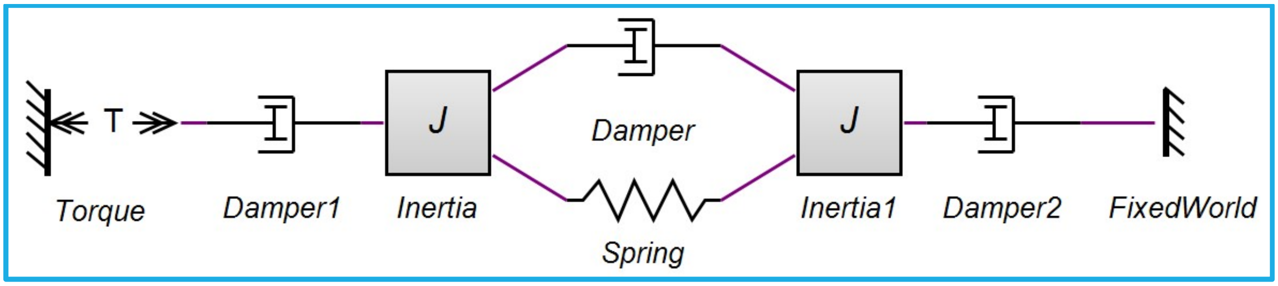


Figure 5a

5.3 EQUATION FORMATION

The basic mechanical components are then represented as a mathematical model.

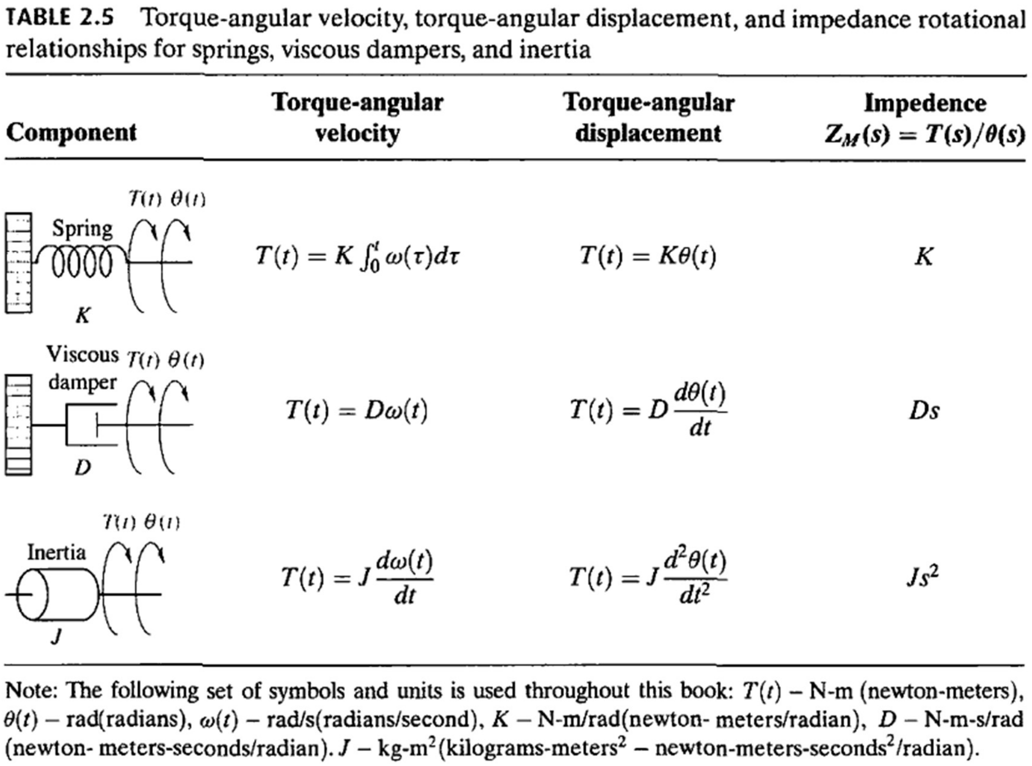


TABLE 2.5 Torque-angular velocity, torque-angular displacement, and impedance rotational relationships for springs, viscous dampers, and inertia

Component	Torque-angular velocity	Torque-angular displacement	Impedence $Z_M(s) = T(s)/\theta(s)$
Spring K	$T(t) = K\int_0^t \omega(\tau)d\tau$	$T(t) = K\theta(t)$	K
Viscous damper D	$T(t) = D\omega(t)$	$T(t) = D\frac{d\theta(t)}{dt}$	Ds
Inertia J	$T(t) = J\frac{d\omega(t)}{dt}$	$T(t) = J\frac{d^2\theta(t)}{dt^2}$	Js^2

Note: The following set of symbols and units is used throughout this book: $T(t)$ – N-m (newton-meters), $\theta(t)$ – rad(radians), $\omega(t)$ – rad/s(radians/second), K – N-m/rad(newton- meters/radian), D – N-m-s/rad (newton- meters-seconds/radian). J – kg-m^2(kilograms-meters^2 – newton-meters-seconds^2/radian).

5.4 THEORITICAL MODEL OBTAINED FOR LOAD MASS

$$\frac{\theta(s)}{T(s)} = \frac{(D_2)s+k}{J_1J_2s^4+(J_2D_2+J_2D_1+J_1D_2+J_1D_3)s^3+(J_1k+J_2k+D_1D_2+D_1D_3+D_2D_3)s^2+(D_1K+D_3k)s}\left[\frac{rad}{V}\right]$$

Equation obtained and solved as per Example 2.19 Control Systems Engineer, By Norman S Nise(Nise, 2011)

5.5 COMPARING WITH BLACK BOX MODEL

Comparing Equation by black box model the missing term can be clearly identified

$$\text{sys3} = \frac{\theta(s)}{V(s)} = \frac{2.64\times10^9}{s^4+11\times s^3+27950\times s^2}\left[\frac{deg}{V}\right]$$

$$\frac{\theta(s)}{T(s)} = \frac{(D_2)s+k}{J_1J_2s^4 + (J_2D_2 + J_2D_1 + J_1D_2 + J_1D_3)s^3 + (J_1k + J_2k + D_1D_2 + D_1D_3 + D_2D_3)s^2 + (D_1K + D_3k)s}\left[\frac{rad}{V}\right]$$

The term in red font is the missing term not present in the model obtained by black box modelling

5.6 COMPNESATING FOR THE MISSING TERM

By using trial and error method using system identification tool box the model obtained by black box model is compensated for the missing term. The Compensated model is:

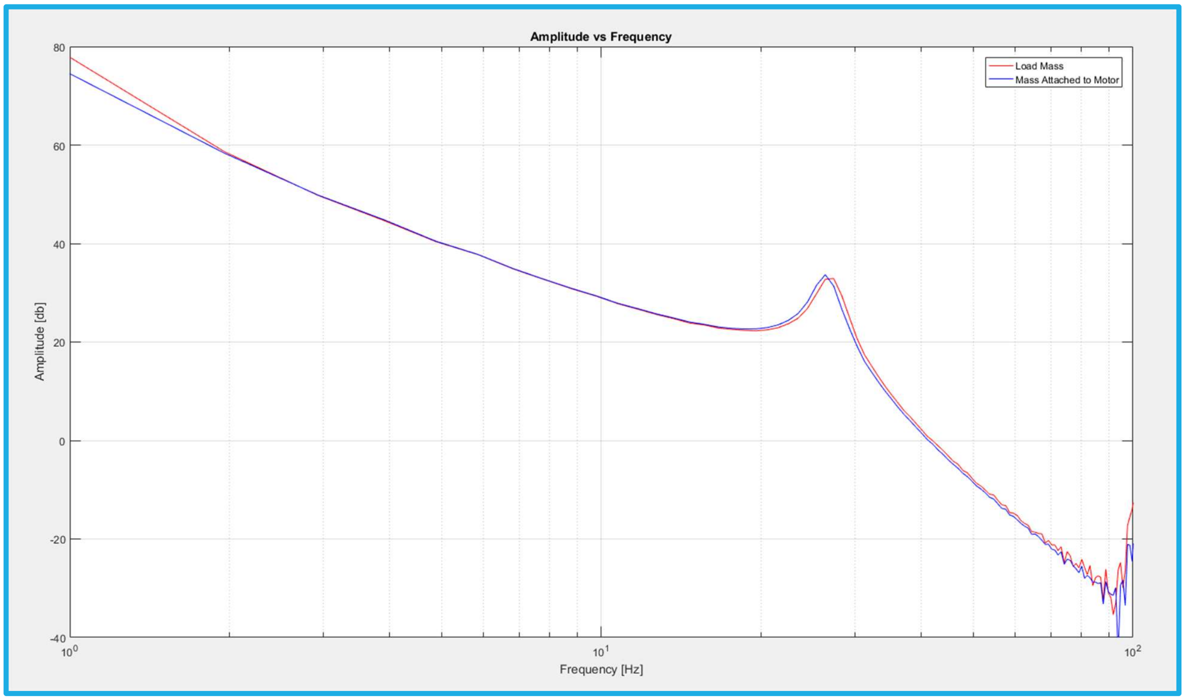
$$\text{sys3\_comp} = \frac{\theta(s)}{V(s)} = \frac{2.64\times1\;^{9}}{s^4+11\times s^3+27950\times s^2+7500}\ \left[\frac{deg}{V}\right]$$

The term in the blue font is added to the black box model to fit the theoretical model.

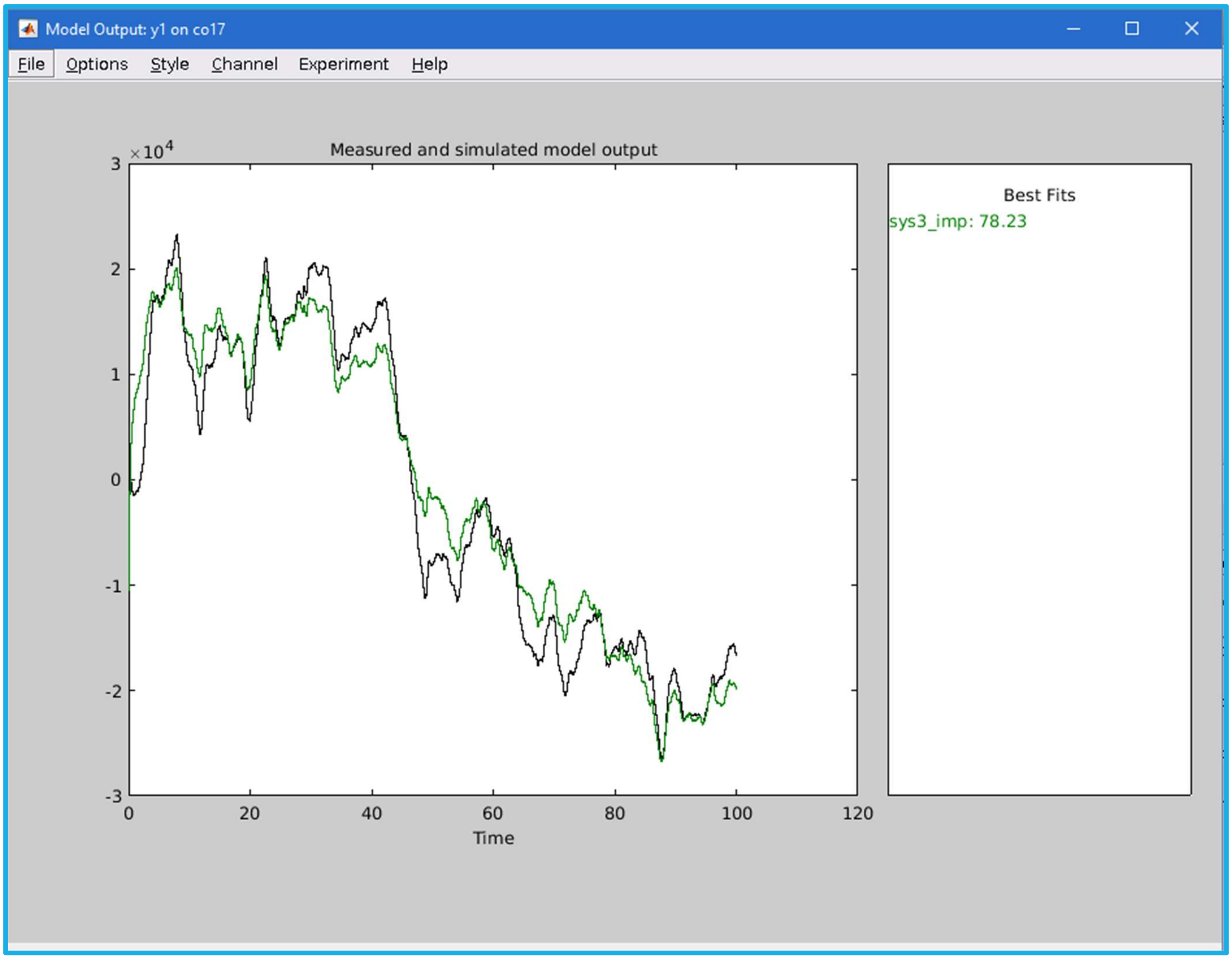
5.7 REVERIFICATION FOR THE MODEL

This new model is again verified using the same techniques as we did the last chapter.

5.7.1 VERIFICATION ATTEMPT BY SIMULINK



5.7.2 VERIFICATION ATTEMPT BY SYSTEM IDENTIFICATION TOOLBOX



5.8 GREY BOX MODELLLING CONCLUSION

In this chapter we saw we are able to improve a model obtained by black box modelling, making a theoretical model and comparing it with the black box model. By doing so we were able to identify major discrepancies (in this case a missing term) in the model obtained by black box modelling. We also saw that the compensated model was able to predict the behavior 4th order model much more accurately. Now this compensated model can be used to create a PID controller, as well will see in the next chapter

.

6. PID CONTROLLER

6.1 INTRODUCTION

In the last chapter we were able to obtain a reliable model for the fourth order model which we verified by several methods, in this chapter we are going to design a PID controller based on that model and verify that it functions correctly.

For this we are going to use the SISO tool, and use PID tuning wizard in it to generate a PID controller for us. Because of time limitations during this internship project I did not spend much time in this part of the project. So we are just going to generate a PID controller automatically from SISO tool based on the model we obtained earlier and verify that it results in a stable system in the actual hardware.

6.2 SIMULINK PID MODEL

First we are going to make a PID model and going to implement it on the same Real Time system (XPC target) that we used to make the model.

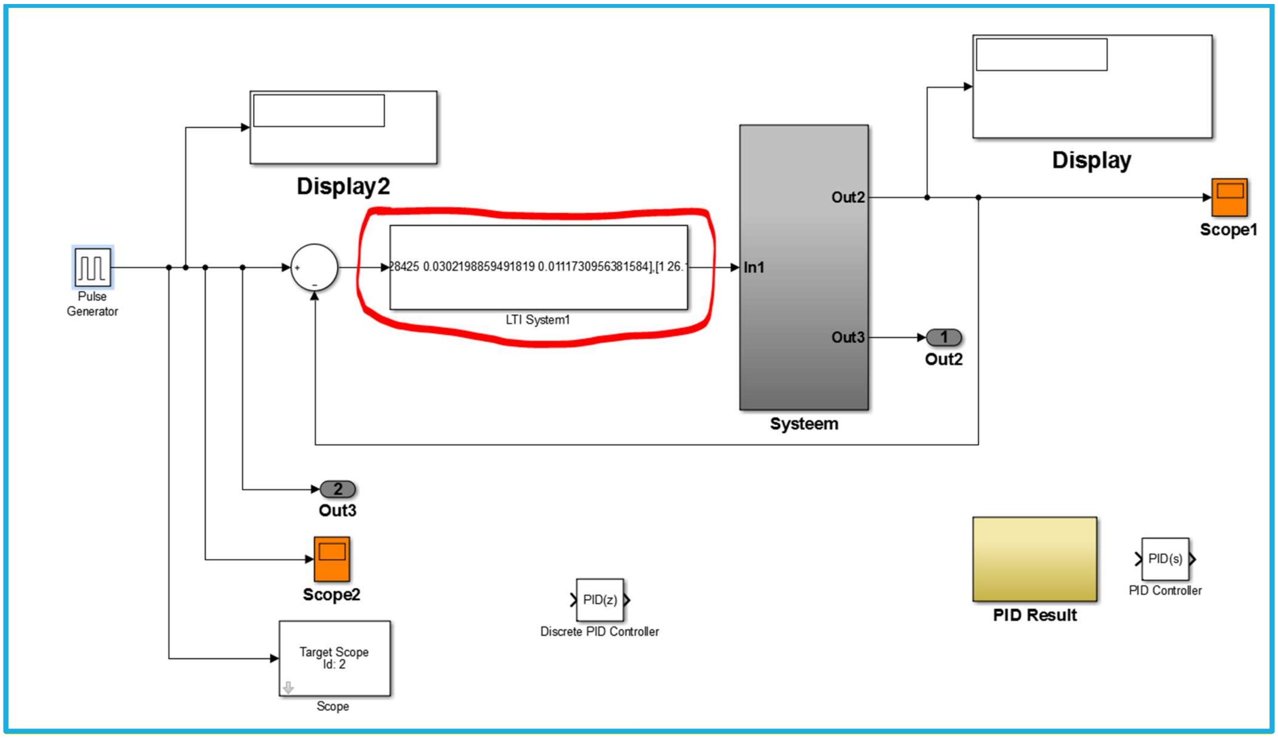


Figure 6a

The above figure is the Simulink model for the PID controller, the Pulse Generator generates the reference signal for the PID controller, it is configured to generate a pulse of 0 and 360 with an interval of 10 seconds. This also helps us simulate a step function to obtain a step response. In the above figure the red encircled block is the PID controller imported as a LTI system from SISO tool. The current value of position is subtracted from the reference and the difference is fed to PID block creating a closed loop.

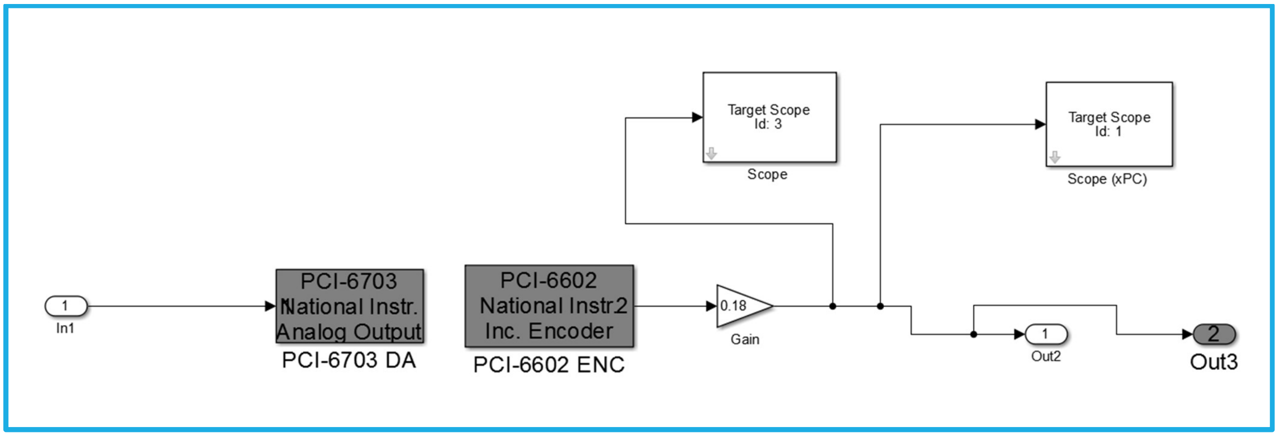
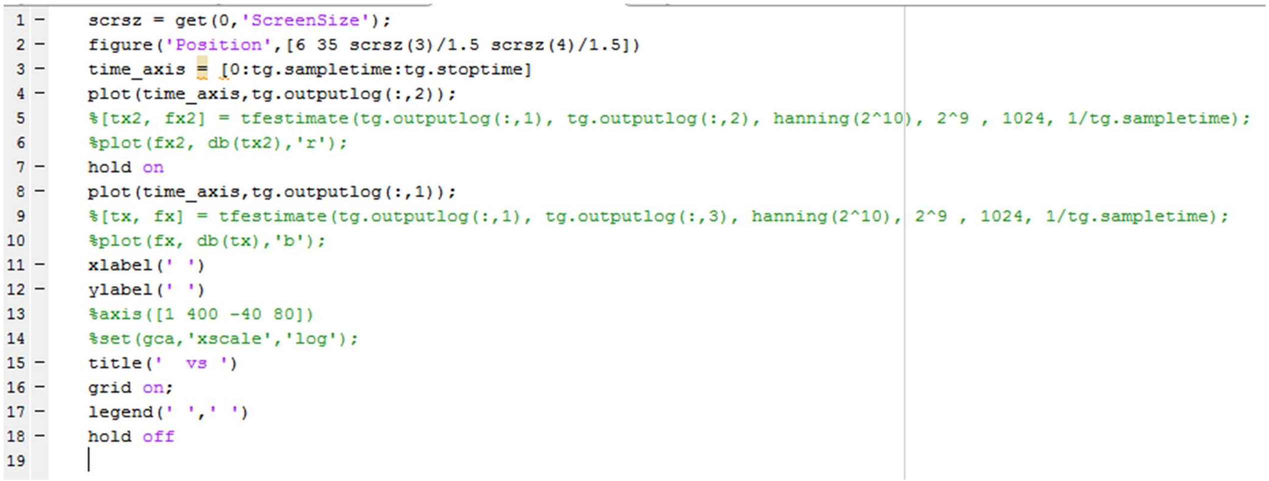


Figure 6b



```
scrsz = get(0,'ScreenSize');
figure('Position',[6 35 scrsz(3)/1.5 scrsz(4)/1.5])
time_axis = [0:tg.sampletime:tg.stoptime]
plot(time_axis,tg.outputlog(:,2));
%[tx2, fx2] = tfestimate(tg.outputlog(:,1), tg.outputlog(:,2), hanning(2^10), 2^9 , 1024, 1/tg.sampletime);
%plot(fx2, db(tx2),'r');
hold on
plot(time_axis,tg.outputlog(:,1));
%[tx, fx] = tfestimate(tg.outputlog(:,1), tg.outputlog(:,3), hanning(2^10), 2^9 , 1024, 1/tg.sampletime);
%plot(fx, db(tx),'b');
xlabel(' ')
ylabel(' ')
%axis([1 400 -40 80])
%set(gca,'xscale','log');
title('  vs ')
grid on;
legend(' ',' ')
hold off

```

Figure 6c

6.3 FIRST TESTS AND PROBLEMS

Well the first tests did result in a stable system, however they had a problem as shown in the figure below.

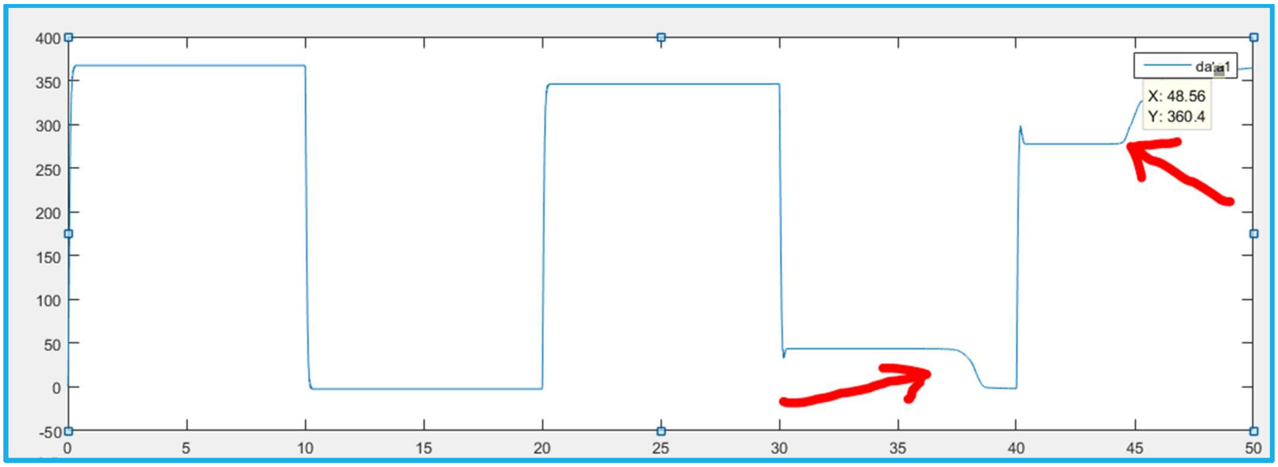


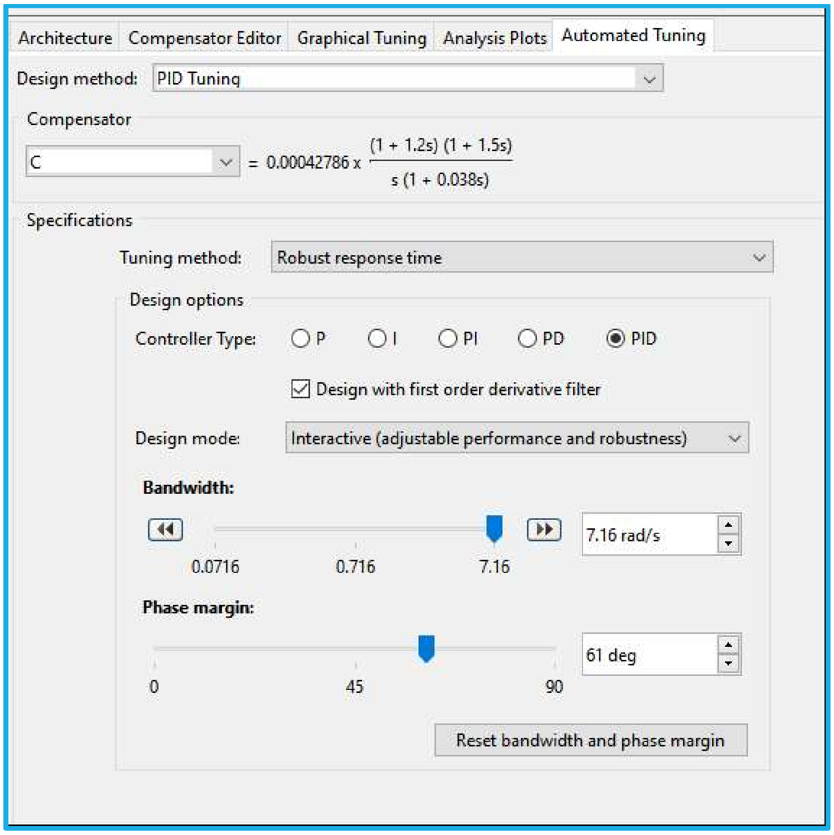
Figure 6d

As seen from the Figure above the system tends to undershoot and stay in that position for long period and the again move and slightly overshoot. This is indicated by the red arrows in the figure, this happens because the model of the system does not completely incorporate the coulomb damping, to be more specific static friction part of coulomb damping. Because static friction is non-linear is cannot be modelled using an LTI system.

6.4 BETTER PID CONTOLLER

In order to come over the problem mentioned in previous section we come up with a simple solution, we design a PID controller with very high response time, this increases the gain of the PID controller and helps overcome the static friction.

Finally, we design a PID controller using the following SISO tool settings:



Design method: PID Tuning

C = $0.00042786 \times \frac{(1 + 1.2s)(1 + 1.5s)}{s(1 + 0.038s)}$

Tuning method: Robust response time

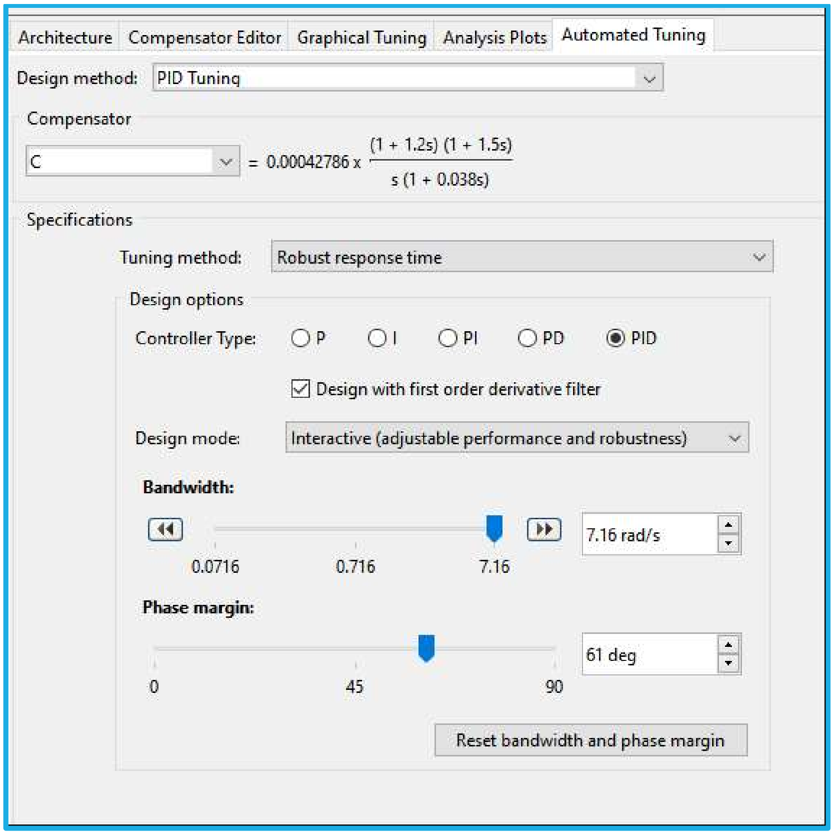
Controller Type: ○ P ○ I ○ PI ○ PD ● PID

☑ Design with first order derivative filter

Design mode: Interactive (adjustable performance and robustness)

Bandwidth: 7.16 rad/s

Phase margin: 61 deg



Design method: PID Tuning

$C = 0.00042786 \times \frac{(1 + 1.2s)(1 + 1.5s)}{s(1 + 0.038s)}$

Tuning method: Robust response time

Controller Type: ○ P ○ I ○ PI ○ PD ● PID

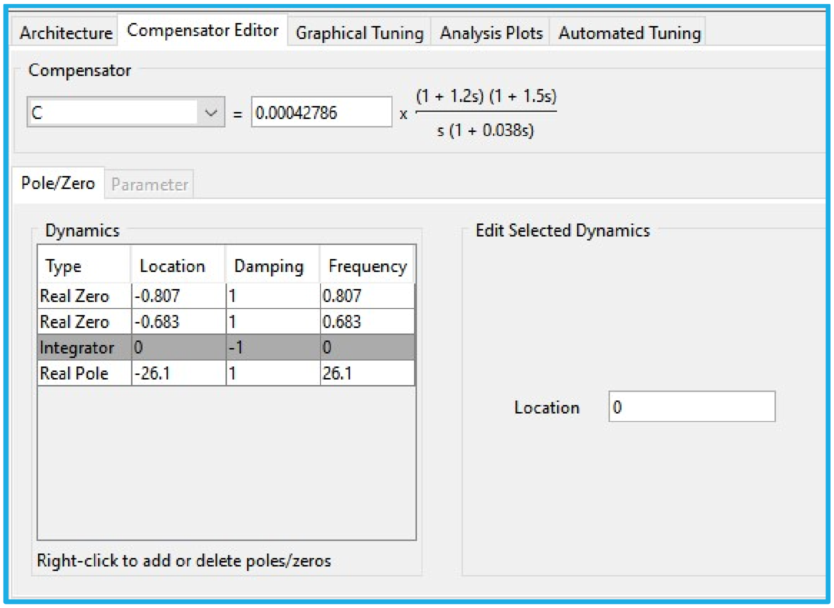
☑ Design with first order derivative filter

Design mode: Interactive (adjustable performance and robustness)

Bandwidth: 7.16 rad/s

Phase margin: 61 deg

Figure 6e



$C = 0.00042786 \times \frac{(1 + 1.2s)(1 + 1.5s)}{s(1 + 0.038s)}$

Type	Location	Damping	Frequency
Real Zero	-0.807	1	0.807
Real Zero	-0.683	1	0.683
Integrator	0	-1	0
Real Pole	-26.1	1	26.1

Right-click to add or delete poles/zeros

Location 0

Figure 6f

Finally, after implementing the controller with the above setting we get the following PID response.

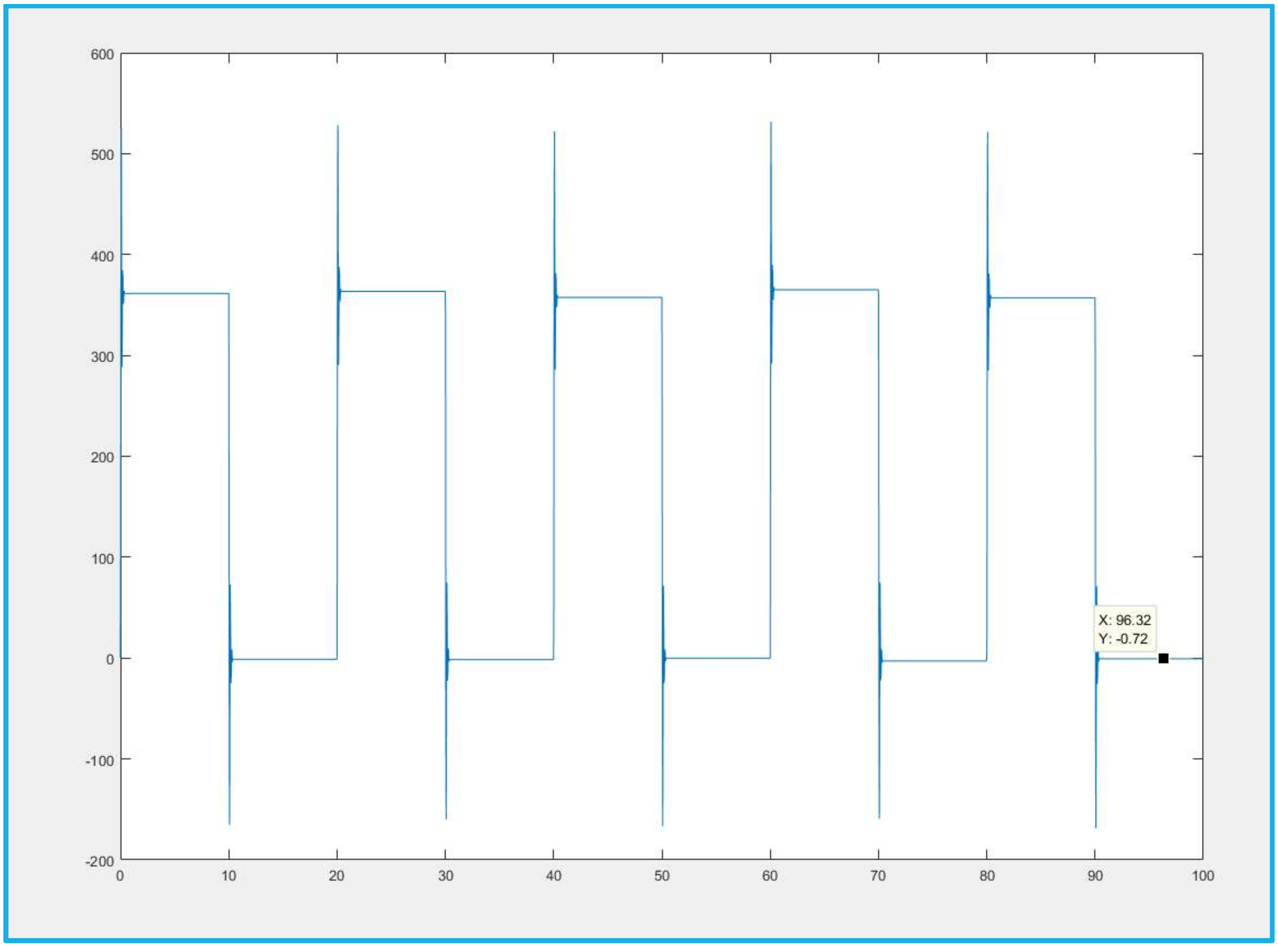
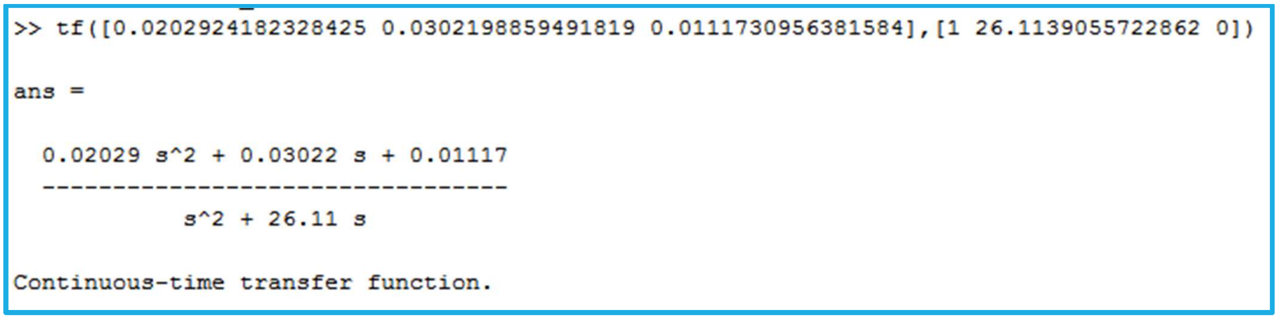


Figure 6g

As you can see from the above figure the system has a very high overshoot now (greater than 38%), however now the system stays most of the time is the desired position. We finalize this controller for now, in order to obtain better performance, we need to switch to a feed forward controller to take care of non-linear damping.

6.5 FINAL PID CONTROLLER

Finally, the PID controller which is designed in the above steps is expressed as a transfer function:



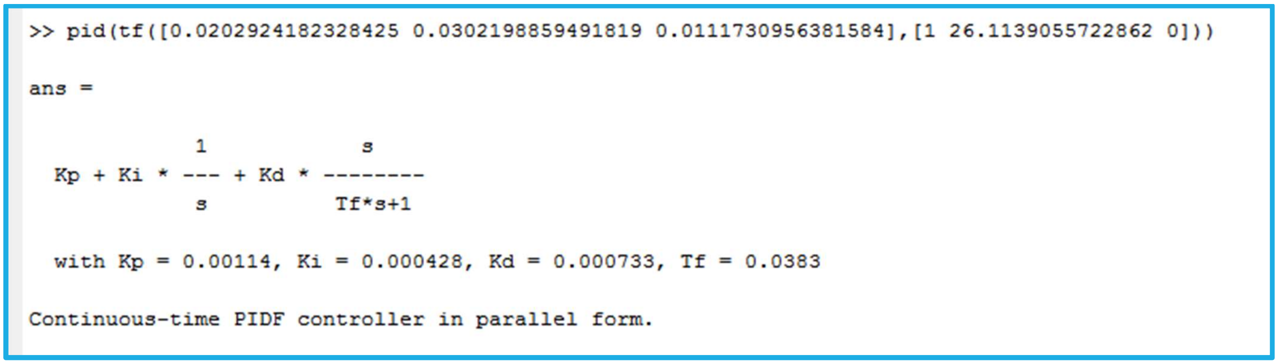
```
>> tf([0.0202924182328425 0.0302198859491819 0.0111730956381584],[1 26.1139055722862 0])

ans =

  0.02029 s^2 + 0.03022 s + 0.01117
  ---------------------------------
             s^2 + 26.11 s

Continuous-time transfer function.
```

That same transfer function is then expressed as a standard PID controller is values of Kp, Ki, Kd expressed separately.



```
>> pid(tf([0.0202924182328425 0.0302198859491819 0.0111730956381584],[1 26.1139055722862 0]))

ans =

             1             s
  Kp + Ki * --- + Kd * --------
             s          Tf*s+1

  with Kp = 0.00114, Ki = 0.000428, Kd = 0.000733, Tf = 0.0383

Continuous-time PIDF controller in parallel form.
```

6. CONCLUSION

6.1 FUTURE RECOMMENDATIONS

Based on the current project status I would like make these major future recommendations for the continuation of the project:

- The static friction must be properly modelled in for the fourth order model.
- The PID controller be changes to a feed forward as well as a feedback design in order to compensate for the static friction.
- Better PID control techniques for the fourth order modelled be researched and implemented
- The PID controller could be implemented on several different platform and could be verified against the results on this project.
- The PID could be implemented on the Distributed Control System network like EtherCAT.

6.2 ACCOMPLISHMENTS

- A clear method of System Identification a research and implemented.
- A fourth order model was modelled and verified to a reasonable level.
- A PID controller was designed and tested for the fourth order model

BIBLIOGRAPHY

Borger, D. (2010). *On the Stability porpertise of motion controllers.* University of Twente, Control Engineering. Enschede: University of Twente. Retrieved from https://www.ram.ewi.utwente.nl/aigaion/publications/show/966

Coelingh, H. J. (2000). *Design Supprt for Motion Control Systems.* University of Twente , Robotics and Mechatronics (CE). Enschede: UTpublications. Retrieved from http://purl.utwente.nl/publications/58112

Drew, B. (n.d.). *Control Systems Engineering - Lecture 6a - Frequency Response*. Retrieved from youtube.com: https://www.youtube.com/watch?v=PVV0f6KfjNg

Engineering, F. H. (2016, 08 29). *Kenniscentrum Mechatronica*. Retrieved from wikispaces: http://lectoraatmechatronica.wikispaces.com/

Instruments, N. (2016, 8 15). *NI PCI-6602 8-Channel Counter/Timer with Digital I/O*. Retrieved from National Instruments: http://sine.ni.com/nips/cds/view/p/lang/nl/nid/1123

Instruments, N. (2016, 8 15). *NI PCI-6703 Static Analog Voltage Output* -- *16-Bit*. Retrieved from National Instruments: http://sine.ni.com/nips/cds/view/p/lang/nl/nid/10704

Nise, N. S. (2011). Obtaining Transfer Functions Experimentally. In N. S. Nise, *Control Systems Engineering Sixth Edition.* John Wiley and Sons.

The MathWorks, Inc. (2016, March 27). *tfestimate*. Retrieved from mathworks.com: http://nl.mathworks.com/help/signal/ref/tfestimate.html

Words, M. (2016, 8 28). *Control System Designer*. Retrieved from mathworks: http://nl.mathworks.com/help/control/ref/controlsystemdesigner-app.html

APPENDIX A PLAN OF APPROACH

1. Background

1.1 Distributed control systems

DCS (Distributed Control Systems) is a more complex form of Industrial Automation, where control element are distributed throughout the system, instead of a one control at a central place. DCS has hierarchy of controllers, each level controlling looking after a particular function. These different controllers are connected together by communications networks for command and monitoring.

1.2 EtherCAT

Fieldbus is the name of family of industrial computer network protocols used for real-time distributed control typically implemented over existing non time critical computer communication network (e.g. Local Area network (LAN) implemented by Ethernet), one of such protocol is the EtherCAT which stands for Ethernet for Automation Technology, and it uses standard Ethernet protocol for automation purposes in an industry. Therefore this protocol can be used to connect different controllers in a Distributed control systems.

1.3 Problem Deffinition

The first part of the assignment aims at the study of working of Distributed Control Systems in which controllers are connected together with EtherCAT. It includes developing and measuring the transfer function for the given system, developing and measuring a PID control loop over the given system aimed at best performance possible; testing the controller over the EtherCAT network in two different configurations, one being PID controller over the EtherCAT network (remote PID controller), and second locally implementing a PID controller on a EtherCAT slave and only control signal being transmitted though the EtherCAT network; finally comparing and measuring these two EtherCAT configurations. Only and only if the previous phase finishes soon enough next problem would be to implement the PID controller and EtherCAT protocol on a general programmable device like a MCU and FPGA, and development of a system using those devices.

2. Objectives

2.1 Overview

The complete internship assignment is focused on properly finishing the first phase as mentioned below, and if time allow continue on to the next two phases.

2.2 Objectives

2.2.1 First Phase (PRIORITY “MUST”)

1. Describe and measure the transfer function of the given system.

2. Describe and measure and implements a PID controller for the given system.

3. Implement and test the above PID control system over the EtherCAT, remote PID control loop.

4. Implement and test the above PID control system locally over an EtherCAT slave with just the desired control signal being transmitted over the EtherCAT.

2.2.2 Second Phase (PRIORITY “NICE TO HAVE”)

Next step would be explore different possibilities in order to replace some of the EtherCAT modules with our own Modules, for example a micro-controller, there exploring how a control system can be implemented locally on a microcontroller using an external controller and also how a less intelligent device could be used locally with main control being implemented by a more intelligent central device.

2.2.3 Third Phase (PRIOTIY “IF TIME ALLOWS”)

Trying to implement the above system with the use of different microcontroller system such as PIC32 or AVR microcontrollers, the use of these microcontrollers with EtherCAT Networking device so that the controlling device could be replaced microcontroller, also maybe use of a microcontroller as a central intelligent device is possible.

2.3 Deliverables

2.3.1 First Phase (PRIORITY “MUST”)

Plan of Approach

Report containing description, simulation and test results and measurement of the transfer function of the given system

Report containing description, simulation and test results, measurements and performance of the PID control loop implemented on the above transfer function of the given system.

Report containing the simulation, test and performance measurement and comparison of the PID control loop system over the two different EtherCAT configurations. 2.2.1 Objectives of First Phase.

Demonstration showing the PID control loop implemented over EtherCAT in the two different configurations and mentioned above.

2.3.2 Second Phase (PRIORITY “NICE TO HAVE”)

Demonstration of the EtherCAT protocol or/and the control loop on a Programmable Microcontroller (Example PIC32) or on a Programmable logic like FPGA. (Depending upon the success so far)

Report about Demonstration and Research

2.3.3 Third Phase (PRIOTIY “IF TIME ALLOWS”)

Demonstrating of the Developed System

System Design Document

Final Report

3. Project Approach Sturcture

3.1 Overview

The complete project is divided into three Phases, each phase is dependent upon the phase before, and therefore each phase supports and is necessary for the next phase. Every Phase is completed using the Scrum method instead of the V-model because what exactly could be going to be achieved in that given phase is not well defined, and is dependent upon research done in that that phase. Every phase is going to have its set of backlogs (sets of demands) which are to be complete using sprints or iteration of one of different sprints. Backlogs lists is flexible which can be changes any time depending upon the success of the sprints, backlog lists can also be prioritized according to the needs, backlogs lists needs to be carefully maintained by weekly meeting with the company manager. Regardless the success of the sprints some requirements such are demonstration and research reports at the end of a phase and the final report has to be completed.

3.2 Project Activities

2.2.1 First Phase

Writing Plan of Approach.

Researching, exploring, learning about the existing EtherCAT protocol.

Developing, Simulating and Measuring the Transfer function for the given system.

Writing the Report regarding the Transfer function and test.

Developing, Simulating and Measuring the PID control system for the system described by the above transfer function.

Writing a Report Regarding the PID control system and test.

Implementing the above PID control system over the EtherCAT, remote PID control.

Implementing the above PID control system locally, only desired control signal over the EtherCAT.

Comparing and measuring the above two configuration against each other.

Writing the Research and Demonstration Report.

Preparing the Demonstration.

2.2.2 Second Phase

Researching the possibilities of implementing the EtherCAT protocol and control systems loops on custom programmable devices such as a Microcontroller or an FPGA.

Exploring the possibilities with EtherCAT useful for the further phases of the project.

Depending upon the success so far as per Scrum model further development with me done

Writing the Research and Demonstration Report

Preparing the Demonstration

2.2.3 Third Phase

Designing and implementing a system depending upon the progress of the last two phases

Writing System Design Document

3.3 Project Organization

To be done

3.4 Project Planning

https://my.fontys.nl/personal/294067/Documents/Internship/PoA/Internship%20Gantt%20Chart%20v1.mpp

4. Risk Management

To be done

5. SWOT Analysis

SWOT analysis on identifying the **INTERNAL STRENGTH** and **WEAKNESSES** as well as identifying the **EXTERNAL OPPORTUNITIES** and **THREATS**

Strengths Good Knowledge of embedded systems and Digital Design Fair Knowledge of Control Systems Hardware availability Ability to analyze problems Ability to quickly learn new techniques and software Readily available guidance from teachers Ability to concentrate on small problems for large amount of time Good Knowledge about Networking Time	**Opportunities** Possibility of implementing the control system on a micro-controller or FPGA Possibility of implementing EtherCAT networking protocol on a micro-controller or a FPGA Possibility of Eliminating the dependence on costly patented hardware and software from companies Possibility using and incorporating the EtherCAT network on the existing university network, therefore creating a more robust architecture.
Weaknesses Inability to write documents and document work Inability to focus on goals and primary objectives Inability to communicate about the problems Inability to realize the effect on small action on the overall project Inefficient and easily gets lost when doing multiple tasks at the same time	**Threats** Running into problems which couldn't be solved with the given hardware

In order to take advantage of the **SWOT** analyses it's important during the project to **BUILD UPON** your **STRENGTHS**, **ADDRESS** your **WEAKNESSES**, **CONSIDER** the **OPPORTUNITEIS** and **GUARD AGAINST** the **THREATS**.