EE 554

FINAL PROJECT

DC motor Control of Flexible Shaft

Andrew Murray

TABLE OF CONTENTS

1. INTRODUCTION

This paper describes my final controls project for EE554. The project is a DC motor control for a flexible shaft system. The purpose of this system is to create a controller in LabVIEW which has the ability to control the speed of the motor which is driving two masses connected with a flexible shaft. The principle of a control system is that you monitor the output of a system and feedback the error signal in order to adjust the input signal with the hopes of eliminating the error.

The type of control system used is Proportional-Integral (PI). The basic algorithm for PI control is:

$$\mathrm{u(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau$$

This equation consists of 2 parts, the proportional section and the integral section. . The proportional gain term, K_p, adjusts the output based on a proportion of the current error. The second gain term, K_i, is the integral gain. This term is multiplied by the accumulated instantaneous error of the system. This term speeds up the movement towards the desired point and eliminates the steady-state error. By adjusting the values of the gain terms you can tune the response of the system to perform optimally.

2. PHYSICAL SETUP

The physical setup of my setup can be seen below in figure 1. The design consists of a DC motor which drives a shaft using a gear train. This gear train is designed to transfer the rotational torque from the motor to the flexible shaft being controlled. The amount of torque transfer through the gear train can be determined using the gear ratio, N (2.1). By determining the product of the gear ratio and the torque from the motor you get the resulting torque after the gear train.

$$N = \frac{Out\ put\ gear\ \#\ teeth}{Input\ gear\ \#\ teeth} \tag{2.1}$$

The shaft from the gear train is connected to the first mass being monitored. This element is a steel gear with an outside diameter of 2 inches. The gear consists of 17 spokes which will be measured by the Hall Effect sensor mounted on the frame to create pulses which are used for velocity calculations. The shafts and gear train connections from the motor to this first mass are considered to be rigid. The first mass is then connected to an identical mass through a 4 inch flexible rubber shaft. This second mass also is monitored by a Hall Effect sensor to determine rotational velocity. This setup allows us to use a DC motor to drive a shaft which includes two masses connected by a flexible shaft. The rotation of the masses is able to be monitored and the RPM of each can be determined. The next section in the paper discusses the electrical setup for the experiment.

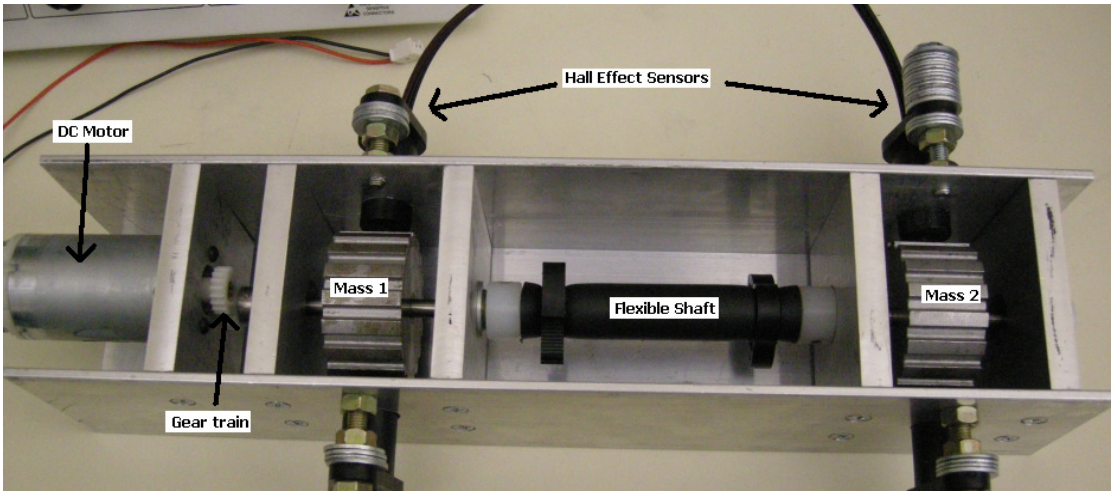

Figure 1: Setup of System

3. ELECTRICAL SETUP

In order to control the DC motor the Hall Effect sensors the system must be wired to the NI Elvis Prototyping Board. The NI Elvis is a prototyping board which is able to be interfaced with LabVIEW allowing it to be controlled with Virtual Instruments created in this platform. The wiring diagram for how this is done can be seen in Figure 2. As this Figure shows the DC motor is wired to the Supply + and – on the Elvis. This allows for the motor to be controlled using a variable power supply integrated into the NI Elvis system from LabVIEW.

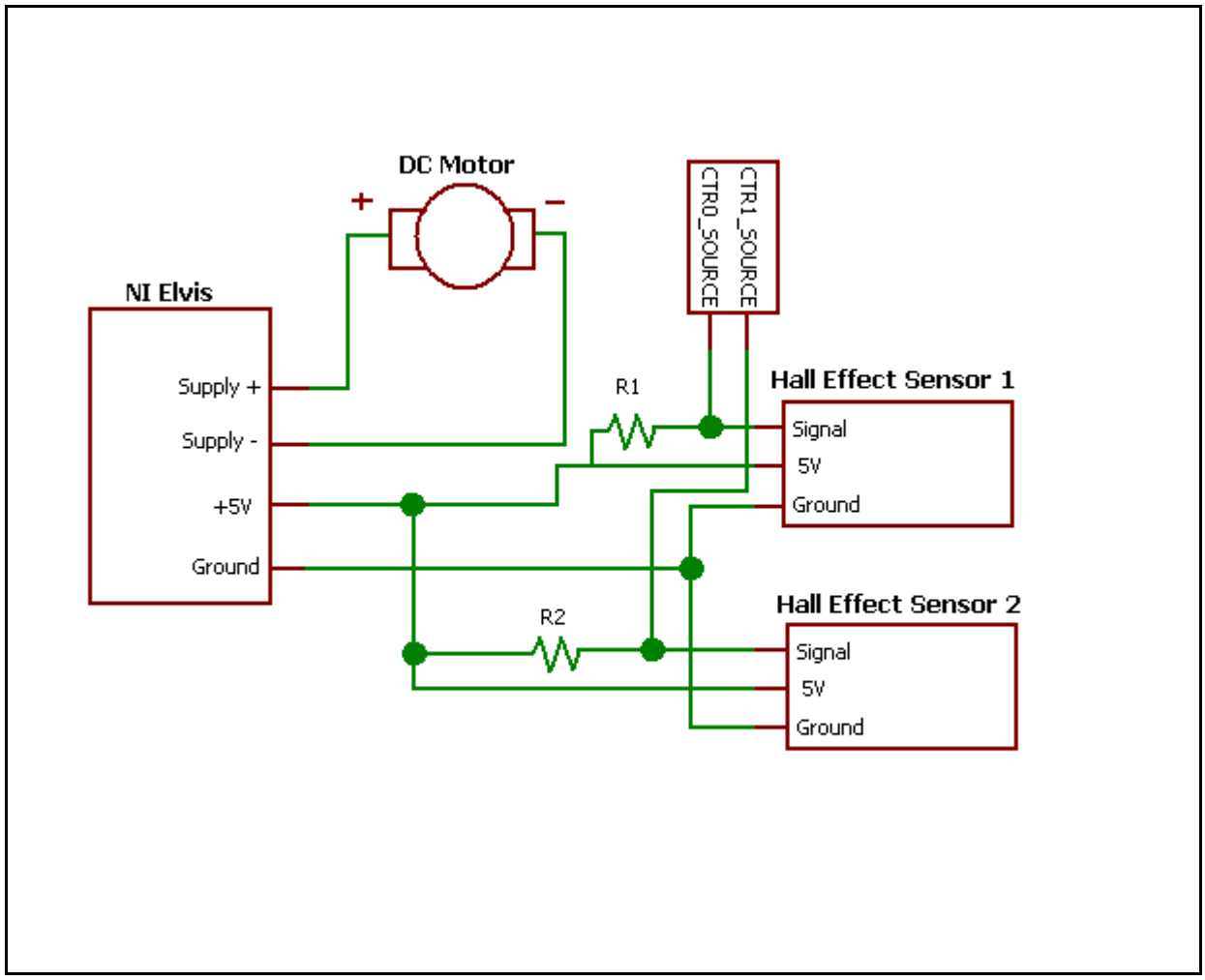

Figure 2: Wiring Diagram

The other electrical devices which need to be wired are the Hall Effect sensors. These sensors Have three external pins which need to be properly connected. These are the 5V power, ground and supply. Figure 2 shoes how these sensors were integrated into the circuit. This integration requires the use of pull up resistors to be implemented in order for the signal to properly work. The pulses emitted from the sensors are then connected to the 2 counters provided on the NI Elvis. These signals are used to determine the speed at which each mass is rotating in the system. This process is done in LabVIEW and is described in the next section.

4. LABVIEW

All of the programming for this project was done in NI LabVIEW. LabVIEW and the Elvis are made to easily interface with each other. This section will give a quick overview at the main systems of the program and explain briefly how each one works. The graphical user interface (GUI) is the portion of the LabVIEW program which allows the user to control the system. This can be seen in Figure 3. There are a few main sections in the GUI designed for this system. The upper left portion of the GUI consists of all the controller information. First is the text box which allows the user to determine the desired RPMs for the system. Second, the user is able to turn the controller on and off so that the system can be ran open or closed loop. There are also a few boxes which allow the user to control the gain values for the controller, this will be discussed more in the Control section of the paper. The user is also able to control which mass to monitor for control.

The second part of the LabVIEW GUI is the indicator section. This consists of two RPM gauges which allow the user to see the current speed at each mass. There are also two waveform graphs. The first plots the speed of each mass over time which allows the response of the system to be monitored. The second graph displays the voltage being supplied to the DC motor over time. There is also an option to save this data so that it can by taken and analyzed in another program such as MATLAB.

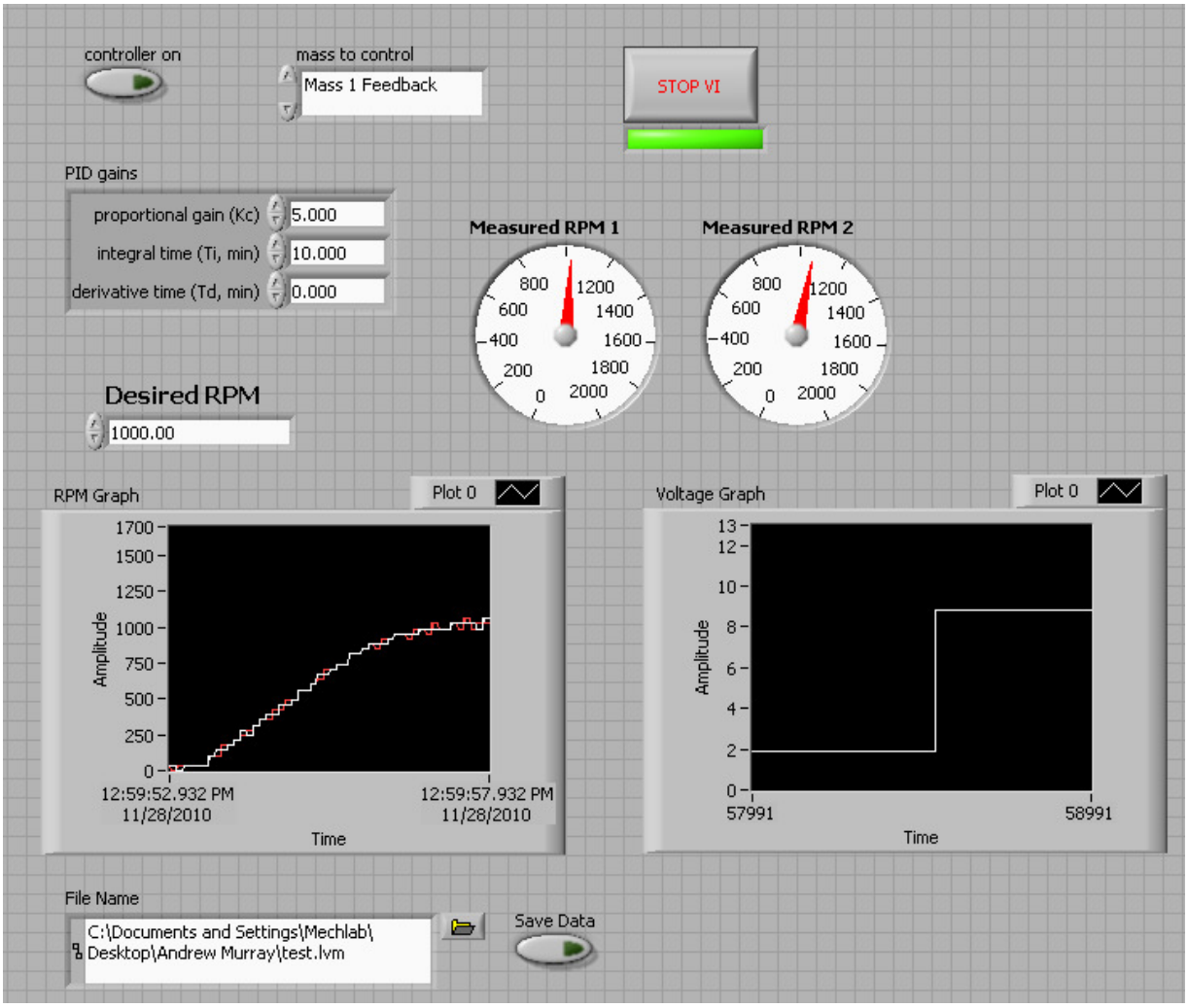

Figure 3: LabVIEW Graphical User Interface

The LabVIEW GUI is what the user interacts with in order to control the system as it is running. However, most of the LabVIEW programming is done in the Block Diagrams. The Block Diagram is where all the registers and subsystems are setup along with the actual programming of the system. The block diagram is setup to be running a case structure with 3 different substructures. Each one of these runs simultaneously and can be seen in Figures 4, 5 and 6.

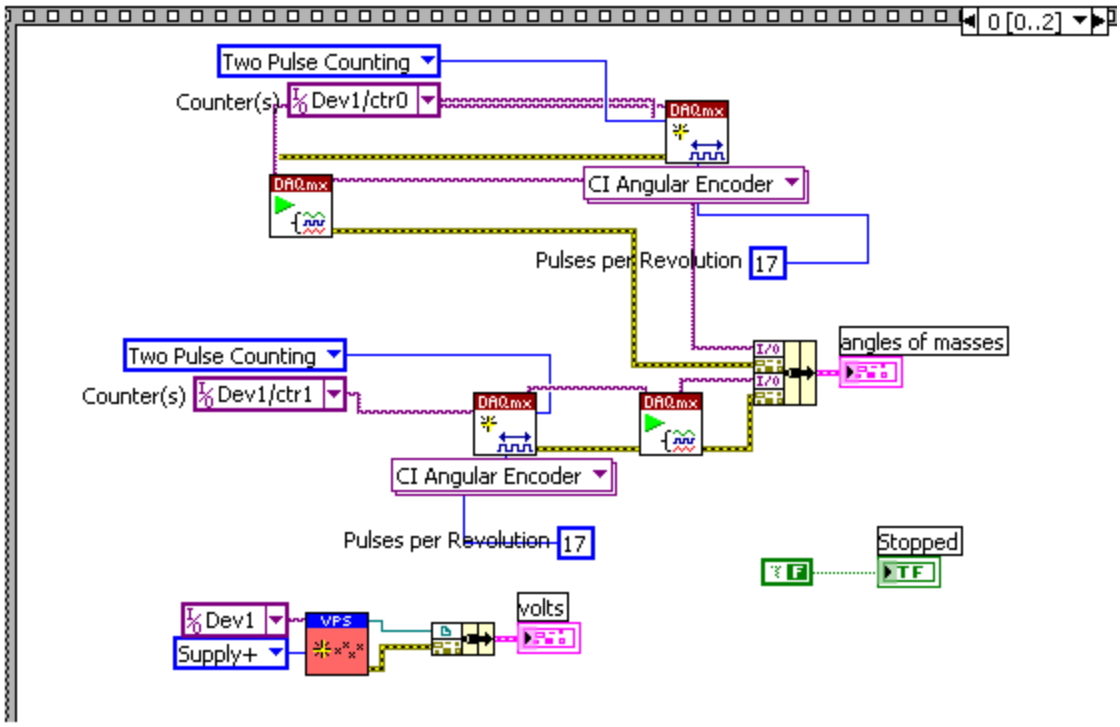

Figure 4: Block Diagram 1

Figure 4 shows the first structure which is responsible for setup of the input captures as well as the Variable Power Supply (VPS). The input capture counter is setup to count the pulses from each of the Hall Effect sensors. There is a sub vi in LabVIEW called the CI Angular Encoder. This is programmed to convert the amount of pulses into the angle of rotation based on the amount of pulses per revolution. These angles are then fed through to the second structure to be analyzed. The VPS is also configured so that is can be easily controlled. The second block diagram structure is shown in Figure 5.

This structure consists of most of the actual operations in the program. The top portion is a loop which calculates the actual RPMs of each mass based on the angles of rotation determined by the counter. The system is set up to sample the RPMs at a rate of 100 Hz. This is about the fastest sampling interval which can be used because it is limited by the speed of Windows. This sampling rate allows for rotational velocities as slow as 300 RPMs to be accurately measured. If

the motor is running any slower than this the pulses may not be captured within the sampling interval and accurate speed determination is unachievable.

The second portion of this structure is the actual control of the motor. The program is able to be ran open or closed loop. When the controller is off the program simply provides a constant voltage to the motor based on what the desired RPM is. There is no feedback. However, when the control is turned on the feedback is allowed. The monitored RPMs are compared to the desired RPMS and an error is determined. This error is fed into the PID control sub VI. This sub VI then uses the PID constants supplied to it from the GUI to determine the voltage which needs to be supplied to the motor to reach the correct desired speed. This is how the controller is implemented in the system.

The final structure in the block diagram is shown in Figure 6. The purpose of this structure is to check whether or not the data needs be be saved. If the user has toggled the save data button on the GUI then the collected data is saved at the path specified. This allows the data to be collected and analyzed in separate software.

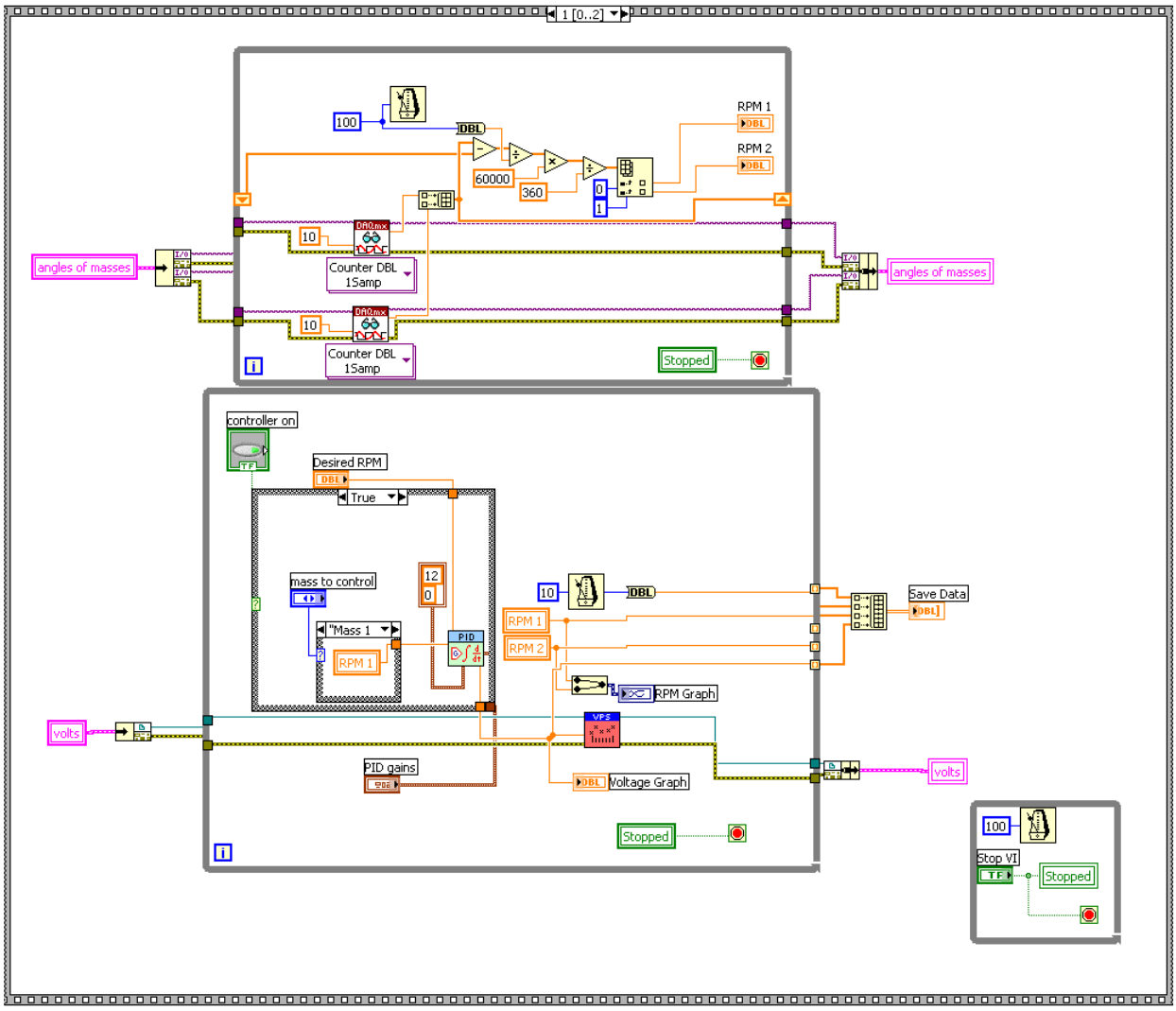

Figure 5: Block Diagram 2

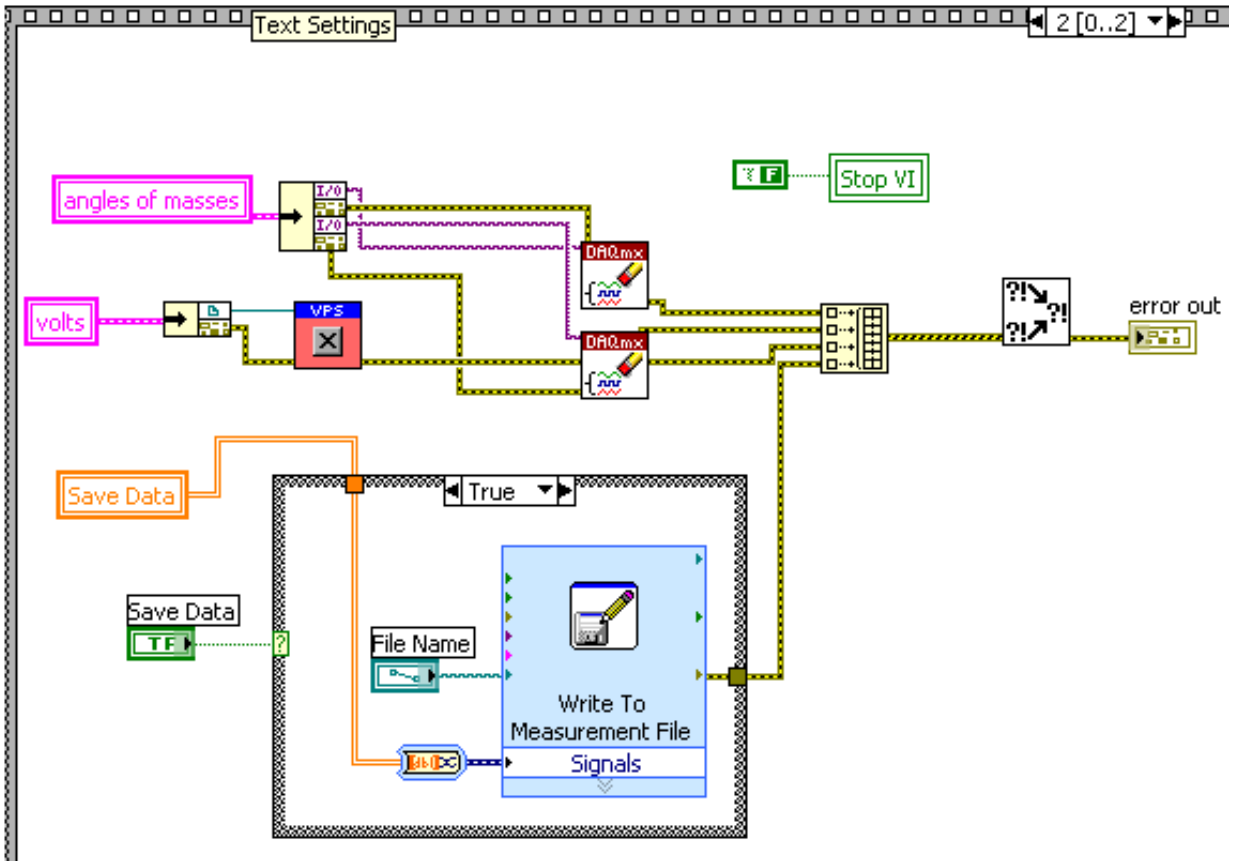

Figure 6: Block Diagram 3

5. MODELLING

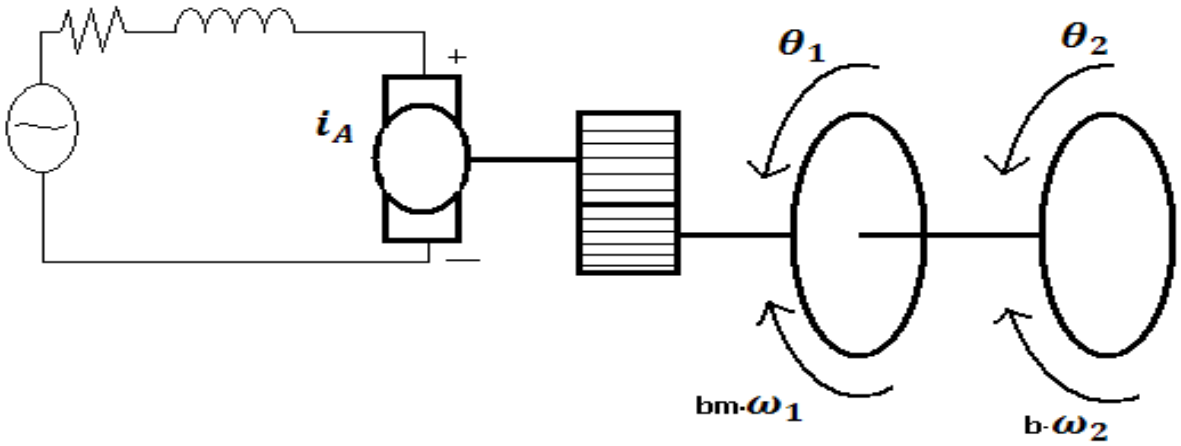

Figure 7: electrical circuit and free body diagram

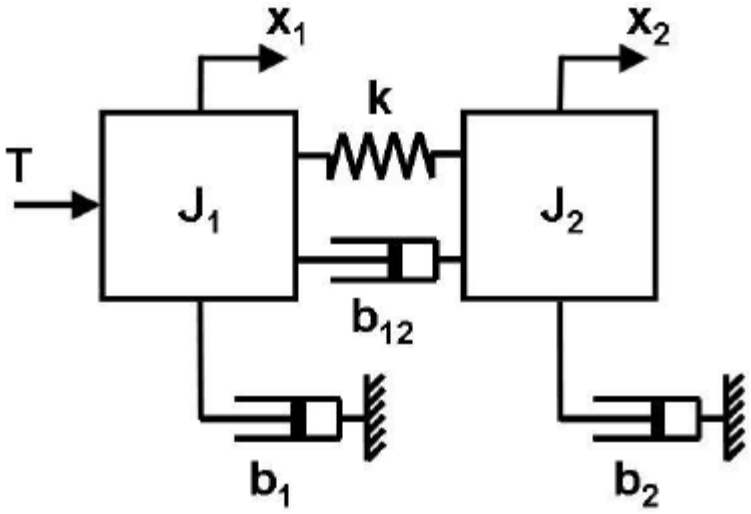

Figure 8:Schematic of the major mechanical components in the motion control plant

After the whole system was setup up it was then necessary to model the system. The free body diagram of the sytems as well as the schematics of the mechanical system is shown in Figures 7 and 8. The motor torque is related to the armature current, I, and the armature constant, k_m. This can be seen in equation 5.1. Also the back emf, e, is related to the rotational velocity, $\dot{\theta}$, and the motor constant, k_b(5.2). By using these free body diagrams as well as Kickoff's Voltage and Current Laws along with Newton's laws it is possible to come up with a system of equations to describe the system.

$$T = k_m * i \tag{5.1}$$

$$e = k_b\dot{\theta}$$

(5.2)

The state variables for the system are shown in Figure 8. These variables include position and velocity of both of the masses as well as the motor current. The equations which describe the system are shown by (5.3) , (5.4) and (5.5). The rotational moments of inertia, J, for the masses are calculated by (5.6) and (5.7). These values are then substituted back into the previous equations.

$$x_1 = \theta_1(t) \quad x_2 = \omega_1(t) \quad x_3 = \theta_2(t) \quad x_4 = \omega_2(t) \quad x_5 = i_A(t)$$

$$\dot{x_1}=(t) = \frac{d\theta_1(t)}{dt}=x_2 \quad \dot{x_2}(t) = \frac{d^2\theta_1(t)}{dt^2}$$

$$\dot{x_3}(t) = \frac{d\theta_2(t)}{dt} = x_4 \quad \dot{x_4}(t) = \frac{d^2\theta_2(t)}{dt^2}$$

$$\dot{x_5}(t) = \frac{di_A(t)}{dt}$$

Figure 9: state variables and derivatives

$$J_1\ddot{x_1} = -b_1\dot{x_1} - k(x_1 - x_3) - b_{12}(\dot{x_1} - \dot{x_3}) + T$$

(5.3)

$$J_2\ddot{x_3} = -b_2\dot{x_3} + k(x_1 - x_3) + b_{12}(\dot{x_1} - \dot{x_3})$$

(5.4)

$$\dot{x_5} = \frac{di_A(t)}{dt} = \frac{V - \frac{e}{N} - R}{L} = \frac{V - \frac{k_b}{N}\dot{\theta} - R}{L}$$

(5.5)

$$I = J_2 = \frac{m * R^2}{2} \tag{5.6}$$

$$J_1 = J_m + {}^{J}/_{N^2} \tag{5.7}$$

Once the equations which describe the system are determined it is necessary to put them into the state-space form which can be seen in Figure 10. The output was chosen to be the rotational velocities of the masses and the current. From the state-space equation it is possible to model the system in MATLAB. The MATLAB code is provided in the appendix of the report. The code calculates the step response of the open loop system. From there it creates a closed loop PI control. This is useful because the controller can be tuned in the model using the Ziegler-Nichols method. This method consists of first setting all the control gains to zero and slowly increasing the proportional gain. This is done until the step response begins to oscillate. Once this has been achieved the proportional gain is turned down a small amount and the integral gain is introduced until the steady-state error is eliminated. Since I am only using PI control this is all that is necessary to tune the system. Using this method the value for the proportional gain is .5 and the integral gain is .2. These will be the values implemented into LabVIEW for control of the system. The step response of the open loop and closed loop transfer functions are plotted by MATLAB as well. The open loop and closed loop step response are shown in Figures 11 and 12, respectively. These plots can be seen in the Results section so they can be easily compared to the actual system responses.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t)$$

$$y(t) = C(t)x(t) + D(t)u(t)$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ -k/J_1 & -(b_1 + b_{12})/J_1 & k/J_1 & b_{12}/J_1 & k_m/J_1 \\ 0 & 0 & 0 & 1 & 0 \\ k/J_2 & b_{12}/J_2 & -k/J_2 & -b_{12}/J_2 & 0 \\ 0 & 0 & 0 & -k_b * N/L & -R/L \end{bmatrix}$$

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

Figure 10: state-space form of equations

6. TESTING

The first test performed was in order to help characterize the motor and develop a relationship between rotational velocity and voltage. The LabVIEW program discussed earlier for the control of the motor was used for this section as well. However, the variable voltage supply was switched off which allowed the manual voltage control to be used. The voltage to RPM ratio was determined by manually setting the voltage to different values and recording the rotational speed determined by LabVIEW. After 15 data points were taken the values were plotted against each other in Matlab and the line of best fit was added, Figure 11.

The equation of this line was also determined. This equation allows the user to determine the voltage which should be supplied to the motor in order to achieve the specified RPM. Once this equation was determined it was implemented in the open loop control system so that the system would know the necessary voltage to reach the desired RPM.

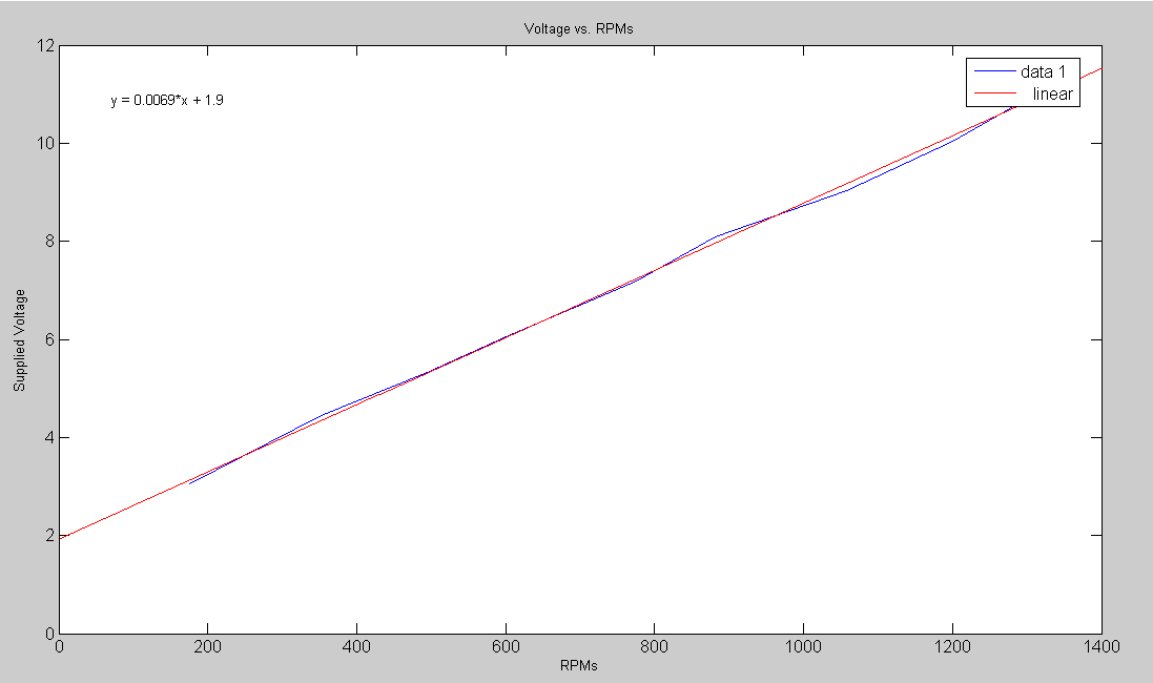

Figure 11: Plot of RPMs vs Voltage

The second test ran was the open loop step response which allowed the time constant to be determined. The LabVIEW program was ran in open loop mode and was subjected to a step voltage input. The response of the data can be seen in Figure 12. The time constant is determined by finding the location on the graph where the amplitude is 63.2% the final value. The time at which this takes place is the time constant. The time constant for this system as marked on the plot is 3.03 seconds.

After the open loop step response test was ran the closed loop response test was done. For this test the PI controller was turned on and the gains were set to those determined in modeling. The response is shown in Figure 13.

The final test was to see how well the controller responded to disturbances. In order to create a disturbance, the edge of an index card was placed in the path of the gears on the shaft controlled by the motor. This creates an added force from the outside world which slows done the system. The data from this test was collected and plotted in Figure ?. In this figure the supplied voltage is plotted against time alongside the RPM on the mass 1. This allows the voltage applied to be compared to the actual speed of the system for better visualization of the controller's performance.

7. RESULTS

The first result is the step response from the open loop model which is shown in Figure 12. The top figure shows the response of the first mass of the system and the bottom plot shows the response of mass 2. This figure can be compared to the step response of the actual open loop system shown in Figure 13. The response of the model seems to be a bit quicker at first than the actual response of the system. This is shown by the steeper slope observed in the model response. However, both the model and actual system take about 5 seconds to reach the final velocity.

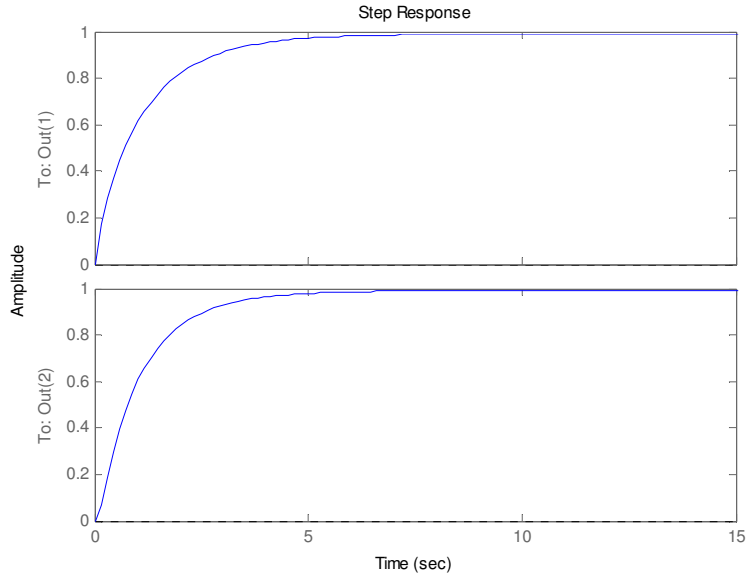

Figure 12: Model of Step Response

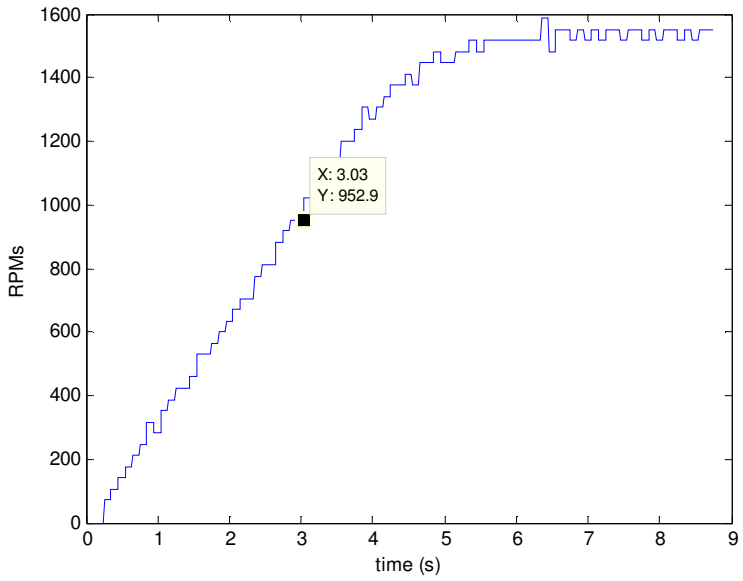

Figure 13: Step Response

The next results presented are the step responses for the closed loop system with PI control. The response of the model is shown in Figure 14 and the response of the system is shown in Figure 15. The two responses are fairly similar to each other. The settling time predicted by the model is

a longer than the actual settling time observed. The other difference between the plots is that the actual overshot of the system is a little more than predicted by the model.

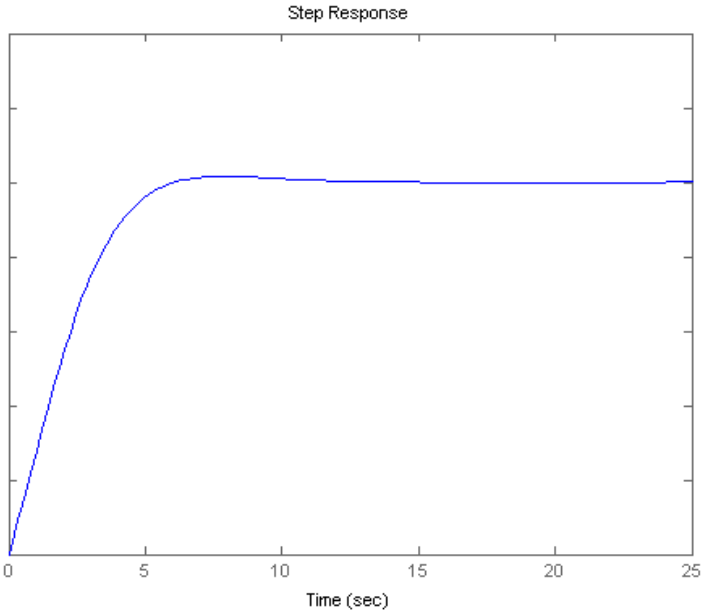

Figure 14: Modeled Step Response with PI control

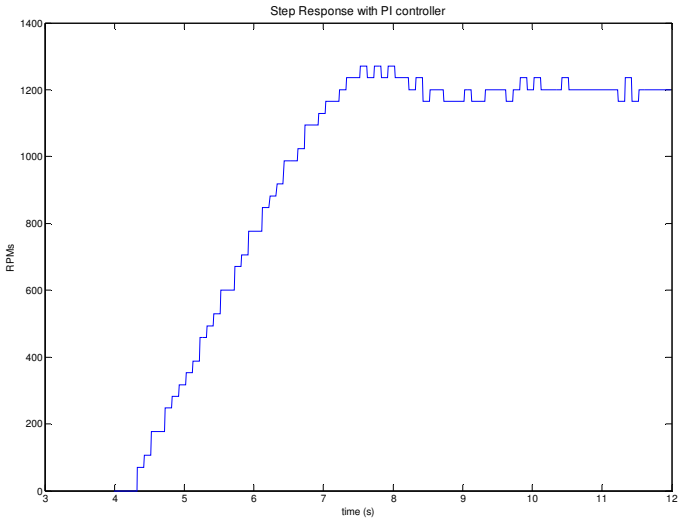

Figure 15: Step Response with Control

The final result of this paper is the response of the system to a disturbance. The data from this test is shown in Figure 16. There are several key features to notice in these plots. The disturbance was first introduced at around 3.5 seconds and is indicated by the sharp rise in the voltage graph. This rise shows the controller adjusting the supplied voltage to account for the disturbance. There are some modulations within the speed plot but it stays fairly constant as the disturbance force acts upon the system. When a disturbance is first introduced the speed slows down a bit before the controller begins to compensate. As the controller compensates it also introduces some overshoot into the response before it settles which is also observed in the graph. The disturbance is greatest from 13-15 seconds. The controller is almost completely saturating at the 12 volts limit at this time trying to keep the velocity constant. At 15 seconds the disturbance was removed completely. This causes the controller to abruptly decrease the voltage which is easily noticed in the plot and gives a little bit of undershoot.

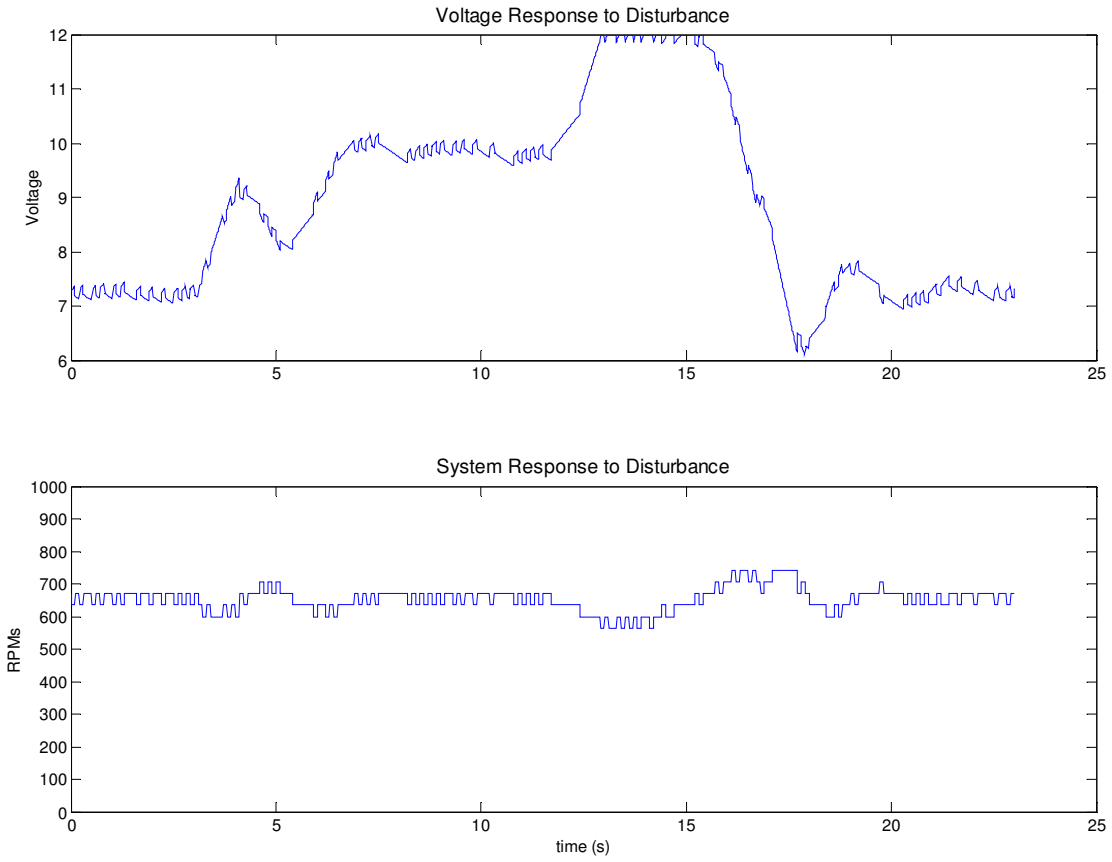

Figure 12: Response to Disturbance. Top graph shows voltage and bottom graph shows RPMs

8. CONCLUSIONS

This paper described the method for developing a control system for a DC motor driving masses connected by a flexible shaft. The first section described the physical setup of the system. Then the LabVIEW program which was written to control the setup was explained. Once the system setup was presented the modeling was discussed. The model was implemented in MATLAB and the open loop step response was plotted. Then a PI control was added to the system to and the gain values were tuned. This was done using the Ziegler-Nichols technique. These values were

implemented in LabVIEW and the actual system was ran and recorded. After this was done the model was compared to the actual data. This showed that the model fairly resembled the actual system although there were errors. The gain values determined in the model did an alright job controlling the system but the response could have been improved. One major drawback of the system was the fact LabVIEW was limited by the speed of Windows.

9. APPENDIX A: MATLAB CODE

```
clc; clear;%state-space
Jm=.059;
m=454;
r=.0254;
J=(m*(r^2))/2;
J2=(m*(r^2))/2;
N1=18;
N2=20;
Gr=N2/N1;
Km=8.64;
Kb=.905
bm=.001;
bf=1;
kf=.8;
Lm=0.01;
Rm=30;
J1=Jm+(J/(Gr^2));
A=[0 1 0 0 0; -kf/J1 -(bf+bm)/J1 kf/J1 bf/J1 Km/J1; 0 0 0 1 0; kf/J2 bf/J2 -kf/J2 -bf/J2 0; 0 0 0 -Kb*Gr/Lm -Rm/Lm];
B=[0 0 0 0 1/Lm];
B=B';
C=[0 1 0 0 0; 0 0 0 1 0; 0 0 0 0 1];
D=[0 0 0];
D=D';
statespace=ss(A,B,C,D);
 %step(statespace)
[num,den]=ss2tf(A,B,C,D);
num = [0 num(1,3) num(1,4) num(1,5) 0
          0 0 num(2,4)  num(2,5) 0
          num(3,2) num(3,3) num(3,4) num(3,5) 0];

 figure(1)
 step(num(1:2,:),den)
 num1=num(1,:);
Kp=.2;
Ki=.5;
Kd=0;
numc=[Kd Kp Ki];
denc=[1 0];
[numCL, denCL]=cloop(conv(num1,numc),conv(den,denc));
figure(2)
step(numCL, denCL)
```