LABORATORY EXPERIMENT III

IMPLEMENTATION OF A POSITION CONTROLLER

MTRN3020

Modelling and Control of Mechatronic Systems

I verify that the contents of this report are my own work.

Nicholas Murray

z3331840

7th October 2012

1. INTRODUCTION

This experiment is concerned with the design and implementation of both a speed controller and a position controller for a system involving a gearbox and motor that is connected to a vertically mounted arm that can be extended or shortened along an axis that is perpendicular to the axis of rotation of the motor. By changing the extension of the arm, the inertia of the revolute joint can be changed and hence controlled.

Students are given the opportunity to design a controller for both the speed and position aspects of the system which are then used on the rig to observe the results and obtain a set of data relating the speed and position of the arm to time and voltage input. We may then continue the experiment by constructing a simulation of the system and its controllers in Simulink and therefore compare our results from the real world to our simulated results.

1. AIM

The aim of this experiment is to firstly develop a speed and position controller to be used on a real world system to obtain a set of data from two distinct runs that relates the position of the arm to time for a changing voltage input.

Once we have successfully obtained our data readings, we can then use Simulink to construct a block diagram of the experiment. Using this, we can simulate our experiment and hence use this block diagram to generate plots for any desired load pattern, mainly the load pattern that was used for our experimental runs. By superimposing our experimental pots and the plot generated by our Simulink model, we can show the compatibility of our results, therefore validate our experiment.

The next part involves replacing the transfer function of our block diagram with the transfer function of the incorrect system allocated to us and then generating position outputs. Again, we will then superimpose the experimental data with the simulated data and observe our results.

As a conclusion, at this stage we can identify any discrepancies between the simulated and experimental plots, and investigate their causes.

1. THE EXPERIMENTAL PROCEDURE

Using the supplied guide along with our knowledge from lectures, we were required to use the direct analytical design method to calculate our speed and position controllers. Once we have our controller transfer function, we can use it to form our final difference equation. The coefficients that we obtain here are supplied to the user interface for the computer that is connected to the rig.

The lab experiment is split into two parts. Part A is designed to verify that our speed controller design was carried out correctly. We used an armature length in the test that was the same as what our controller had been designed for. If the actual response shows that the zero steady state error and a first order response that shows a time constant that was specified for us as one of the design parameters, the design is correct.

Part B is concerned with replacing the transfer function of our block diagram with the transfer function of the incorrect system allocated to us and then generating position outputs, after doing an experimental run with a different armature length which is equivalent to designing a controller without knowing the exact parameters of the system. Some clear deviation in responses will be shown here.

1. CONTROLLER DESIGN CALCULATION

We begin our calculations by obtaining a first order approximation to our no load open loop data, ie of the form:

We can use lsqcurvefit to fit a curve to our data and simultaneously obtain our *A* and . For my armature length, I obtained values A = 280000 and . These values can then be plugged into our transfer function that relates the applied voltage to the speed in counts/second:

Including an integrator to obtain a transfer function that relates voltage to speed in rad/s gives:

By combining all of the block in our block diagram for the system, excluding *Gc(z)* the transfer function for the plant is:

Using Matlab’s c2dm function to find us a discrete version of our continuous transfer function helps us find *Gp(z)*.

The next step is to form our *F(z)*. Using our unique time constant and the fact that we need zero steady state error, that is unity DC gain ie . Because our zero could cause ringing, it must be absorbed by the numerator of By knowing that a time constant of corresponds to in the *s*-domain, its location is therefore . So keeping in mind that

Now that we know we can plug it into the following expression to obtain :

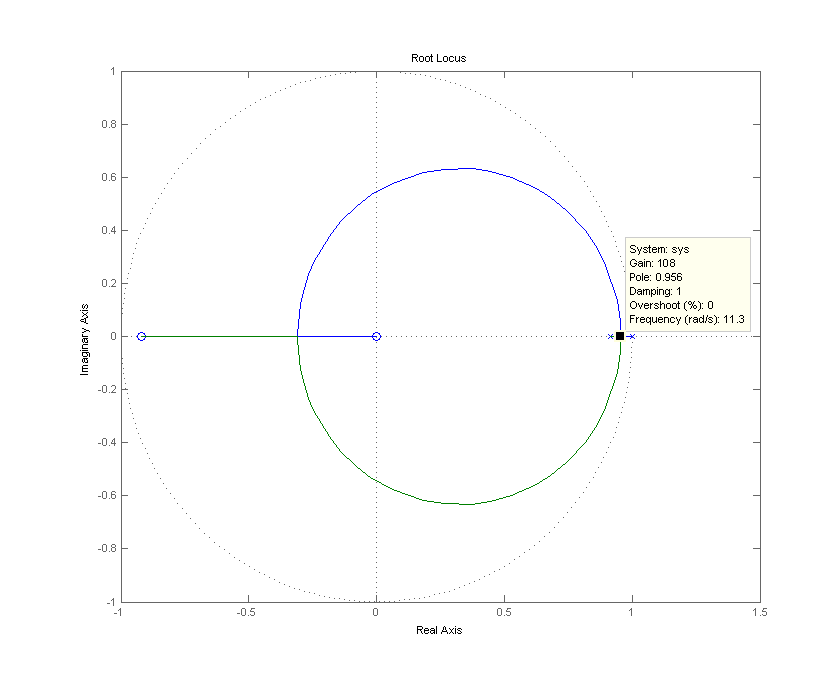
Using my sampling time of 4ms and my time constant of 45ms, our resulting speed controller is:

This gives us a difference equation of:

These are the values that we supply the system controller with.

A position controller must also be designed. This can be done by combining the entire velocity loop, integrator, the 2K gain, and the gear ratio to yield the forward path transfer function applicable to the position control loop:

We can obtain the gain by plotting the root locus of this transfer function in Matlab:

Figure 1: Root locus of G(z)

Observing this root locus gives us a gain of 108 and a break away point of 0.956.

1. SIMULINK BLOCK DIAGRAM

In order to create a simulation to compare to our experimental data, we use Matlab’s Simulink to easily construct a block diagram of our system. By referring to the appendix included in the guidelines, the following block diagram as seen in Figure 2 was made:

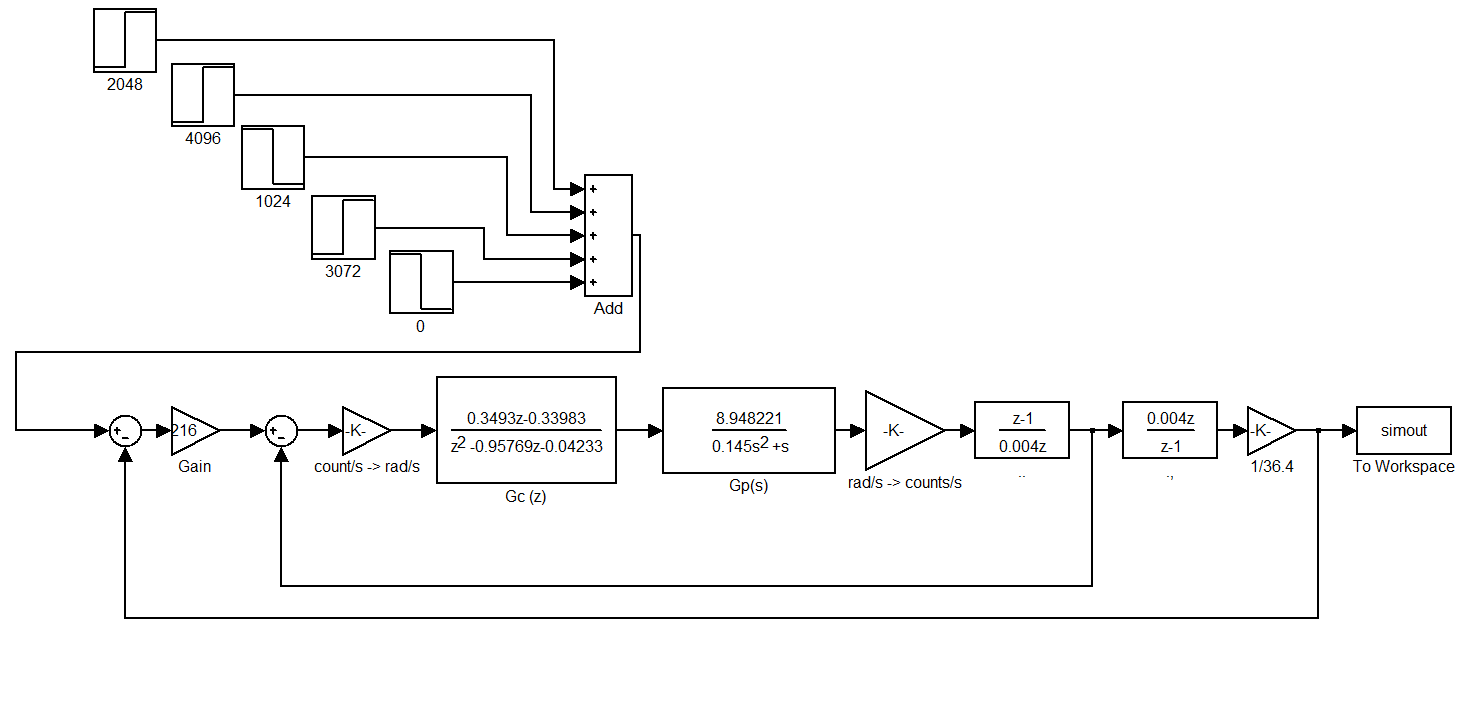


Figure 2: Simulink block diagram of the experiment.

1. PART A

Now that we have our block diagram constructed, we can input step functions to simulate our experiment. Just like the run in the lab, we will include steps to be activated at certain times to match the input for our experimental runs. We will then superimpose the simulated data with the experimental data to verify the design. Figure 2 shows the step function blocks as they appear in Simulink.

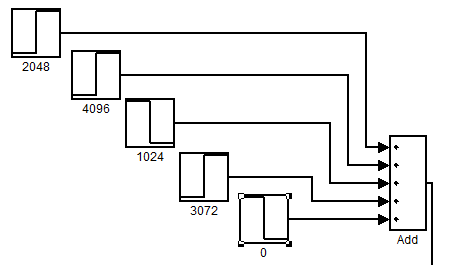


Figure 2: Simulink step function.

There are 5 blocks here that are activated 1600ms after each other. Obviously, this data is not of the same length as our experimental result’s data, so we use Matlab’s interp1 function to interpolate the simulated data to correspond to the length of the time vector of the experimental data. Now we can superimpose the plots of both sets of data to compare the block controller design as shown in Figure 3.

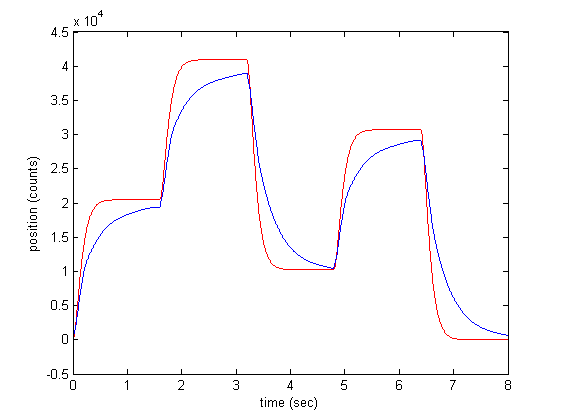


Figure 3: Superimposition of experimental and simulated theta vs time plots. Experimental is show in blue, whilst simulated is represented by red.

It is clear that there is large discrepancy between data. The rise time is very different, with the rise time of the experimental results not even reaching a steady state after 1600ms. This failure to reach the desired position in time results in a stacked error of steady state position as the run continues. As the experimental plot’s curves never reach their steady state, we cannot say if its final value for each phase would be equal to our final result.

It seems that from this simulation we can say that there was a problem in the experimental data acquisition. Even after several reruns of the experiment, coupled with multiple attempts of recalculating the controller, the experiment did not go as planned. As the very same controller values were later used in the Simulink block diagram to give much better results, it is unclear whether there is an actual error in the controller calculations, or there were problems with the experimental rig itself.

We could also attribute the slow rise time to being due to saturation in the control effort. Because of this, the motor cannot do any better, and hence the speed of response lags behind. The simulated acceleration may very well exceed the physical limitations of the rig. This would affect the rise time.

1. PART B

We can now use lsqcurvefit again to obtain values for our *A* and , this time for an armature length of 225mm. By using our given data from ROT225.m, we obtain values A = 280000 and . And so again, by knowing that

We can once again include an integrator to obtain a transfer function that relates voltage to speed in rad/s gives:

We can then go ahead and replace the transfer function of our block diagram with our new one for 225mm, as shown in Figure 4.

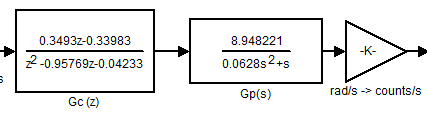


Figure 4: The new Gp(s) placed in the block diagram

We can now rerun the simulation with the new transfer function and retrieve a new set of data that we can superimpose onto the experimental data:

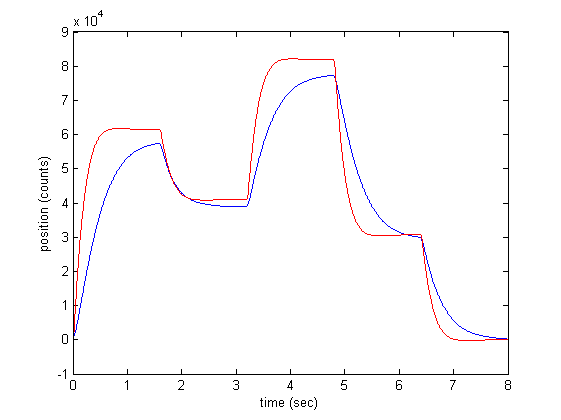


Figure 5: Superimposition of experimental and simulated theta vs time plots. Experimental is show in blue, whilst simulated is represented by red.

We can note that because of the lower inertia in this run from the smaller armature length, there is a small amount of overshoot in the simulated data’s curves.

However, once again, there is a similar set of discrepancies, this time more extreme. Whilst the simulation had acceptable controller properties, the experimental data has a very slow rise time, so much as that we cannot judge whether it would reach a desired steady state or not. This more extensive error result would be due to the different inertia from the new armature length, because the simulation does not account for this.

We can see that there were clearly problems in the experimental stage again, and the same can be said in this part that is unclear at this stage whether there is a problem in the controller, or there were physical errors in the rig. The reason could most likely be due to the rig as the controller values seemed to give a normal response.

We could also attribute the slow rise time to, again, being due to saturation in the control effort due to a maximum achievable voltage to accelerate the motor that would not be accounted for in the Simulink model. Because of this, the motor cannot do any better, and hence the speed of response lags behind. The simulated acceleration may very well exceed the physical limitations of the rig. This would affect the rise time.

1. CONLUSION

In conclusion, it can be stated that clearly, there was a problem in the experimental stage of this assignment. The runs involved very slow rise times which indicate some trouble with the speed controller aspect. As mentioned beforehand, the controller we calculated various times and verified but the success of another student’s controller calculations using the same methods, and the success of the simulated data that also used the same values as what was used in the experimental runs. Therefore it is difficult to say whether the large discrepancies were due to errors in the controller or problems with the physical rig.

By observing the graphs for the experimental data, if we add in a trend line for the curves we can estimate that the eventual steady state value would be quite close to the desired position, as seen in figure 6.

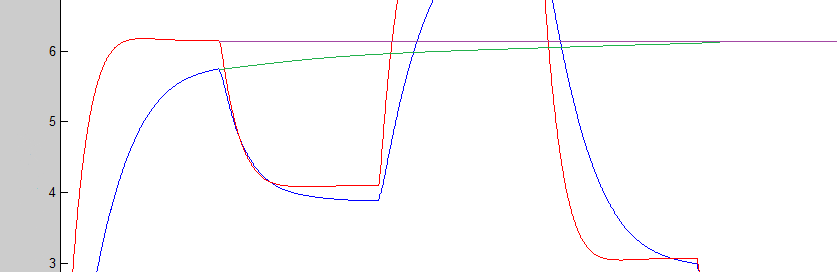


Figure 6: Superimposition of experimental and simulated theta vs time plots, with additional trend lines. The green line represents what the experimental data could eventually settle to, and the purple line represents the simulated and desired steady state position.

We can conclude from this that the steady state error, given a long enough time, should be quite small. However, this is overshadowed greatly by the rise time difference. 7 seconds is a very long time to reach a steady state for this experiment.

As well as the aforementioned saturations in the control effort due to a maximum achievable voltage, other errors may include the variability in plant parameters. These include the inertias both the motor and the armature, and the effects of air resistance. However, it would be expected that the impacts of such errors would be quite minimal.

The fact that the changing of the armature was done manually by hand would count as a potential source of human error that would affect the plant characteristics. There may also be a presence of instability in the test bench that was assumed to be completely stationary by the simulation.