LABORATORY EXPERIMENT II

IMPLEMENTATION OF A SPEED CONTROLLER

MTRN3020

Modelling and Control of Mechatronic Systems

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

Nicholas Murray

z3331840

17th September 2012

1. INTRODUCTION

This experiment is concerned with exercising our knowledge of controller design via direct analytical design, or the so-called Ragazzini method, using Simulink to construct a block diagram, as well as exploring the way controllers are implemented in to physical set ups. We were required to design our required controller prior to our lab session, where we would then validate our controller on a generator rig driven by an electric motor which was connected to a bank of 15 resistors that can individually be connected, resulting in different motor loads. We could then obtain a set of data for two different runs, including a disturbance rejection run. Once we have received out recorded data set, we may then simulate the experiment using Simulink, and show the compatibility of our simulation by applying it to a load pattern unique to every student.

1. AIM

The aim of this experiment is to firstly develop a controller using direct analytical design that will succeed in handling and rejecting disturbances in the form of a changing load on a motor generator, so as to maintain our desired speed of our motor despite the load changes. We then use our controller to obtain a set of data for two distinct runs (that will be elaborated on in the following section) which would be unique to each student.

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 is derived from our individual student number. 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.

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

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 control computer.

The lab experiment is split into two parts. Part A is designed to verify that our speed controller design was carried out correctly. 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. In order to generate a plot, the first experiment was run at 1000rpm, and then a step change of 1000rpm was applied which brings the speed to 2000rpm. This transition from 1000rpm to 2000rpm must show a first order response that corresponds to our time constant. After this, the zero steady state error can be detected at both speeds, 1000rpm and the 2000rpm. The veracity of our design will be verified by this graph.

Part B is concerned with handling disturbances to the load. Depending on the number of resistors connected to the generator, the load on the motor changes. For 8 sets of 200 consecutive data samples, the first and last sets have no resistors connected, whilst the other 6 take a varying number of resistors. This resistor pattern is determined using our student number. If the controller is designed correctly, when the speed drops or increases, the controller will make the system get back to the specified speed. The data gathered during this experiment can be used to make this plot.

1. CONTROLLER DESIGN CALCULATIONS

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 . 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 counts is:

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, my resulting controller is:

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 1 was made:

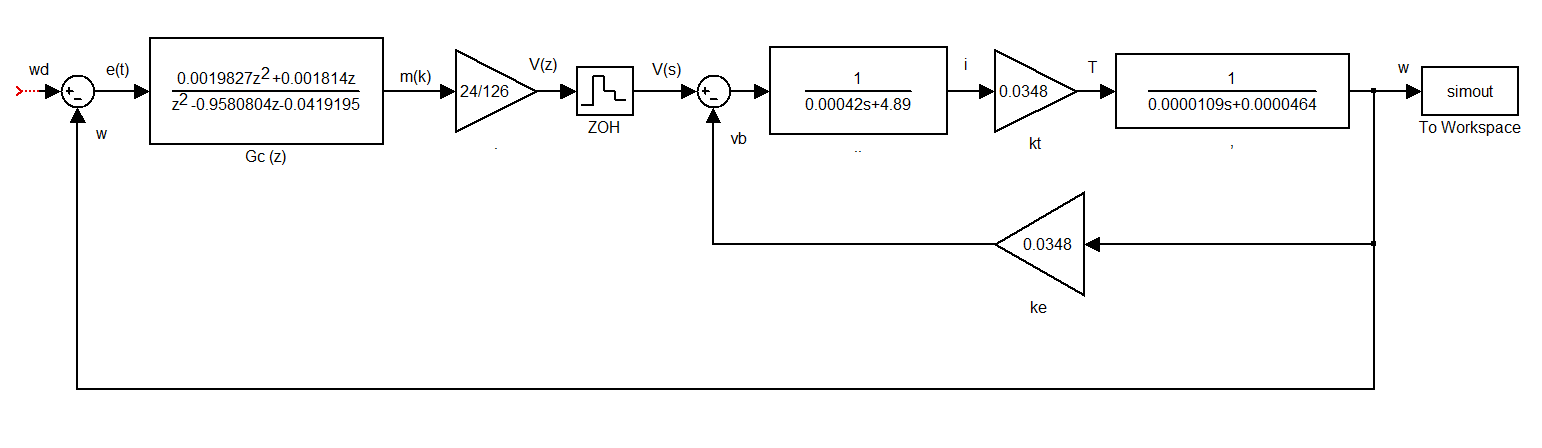


Figure 1: Simulink block diagram of the experiment.

1. PART A – 1000RPM TO 2000RPM

Now that we have our block diagram constructed, we can input step functions to simulate our experiment. Just like the second run in the lab, we will include an initial step to 1000rpm and then another 1000rpm step to bring the total rpm to 2000. 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.

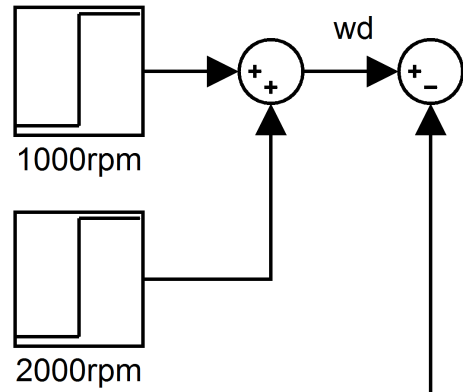


Figure 2: Simulink step function.

There are two step function blocks. The top most block is activated at t = 0 and the second one, whilst labeled as 2000rpm, only adds 1000rpm to the input, begins at the same time as the experimental results step up to 2000rpm. Once these two blocks are added, we run the simulation and the output block returns an array of data. 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.

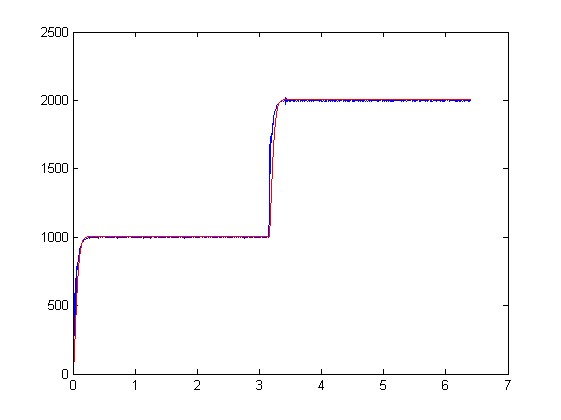


Figure 3: Superimposition of experimental and simulated theta vs time plots. Experimental is show in blue, whilst simulated is represented by red.

1. PART B – LOAD CHANGES

We can now adjust the same Simulink block diagram this time to represent the load changes discussed in section 3. To determine our load pattern we convert our 7 digit decimal student number into a 6 digit hexadecimal number. Each of the 6 digits in this number represents a corresponding number of resistors that are to be active at each 200 sample period. For my student number, 3331804, we get 32D6DC. We can see the effects of the resistors in our first experimental data file:

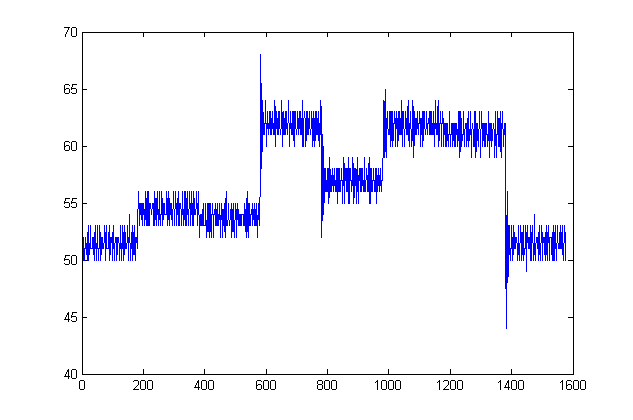


Figure 4: Resister loads in the pattern: 0-3-2-D-6-D-C-0

In order to simulate the load from the resistors, we use the “LoadSim” block that is made available to us. It will handle the complexities of the resistors themselves for us and enable us to only be concerned with feeding in our omega and the number of resistors currently active, and feed out the resulting load torque to be differenced with our motor torque to give us the torque available to accelerate inertia.

In order to manage the number of resistors that are to be active, we create a series of step functions that are all summated and fed into the LoadSim block. The way this work is when the number of resistors is to go up or down, another step function occurs that will add or subtract the required amount of resistors. The LoadSim and resistor blocks can be observed in Figure 5.

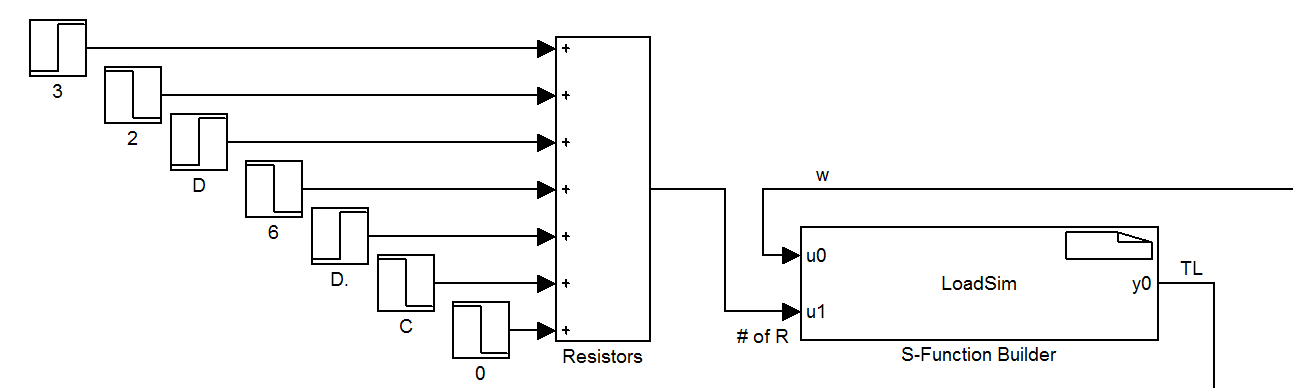


Figure 5: The summated resistor step files being fed into the LoadSim block.

Now that the load feature has been added we can go ahead and run the simulation at 2000rpm. The comparison of both plots, experimental and simulated is shown in Figure 6.

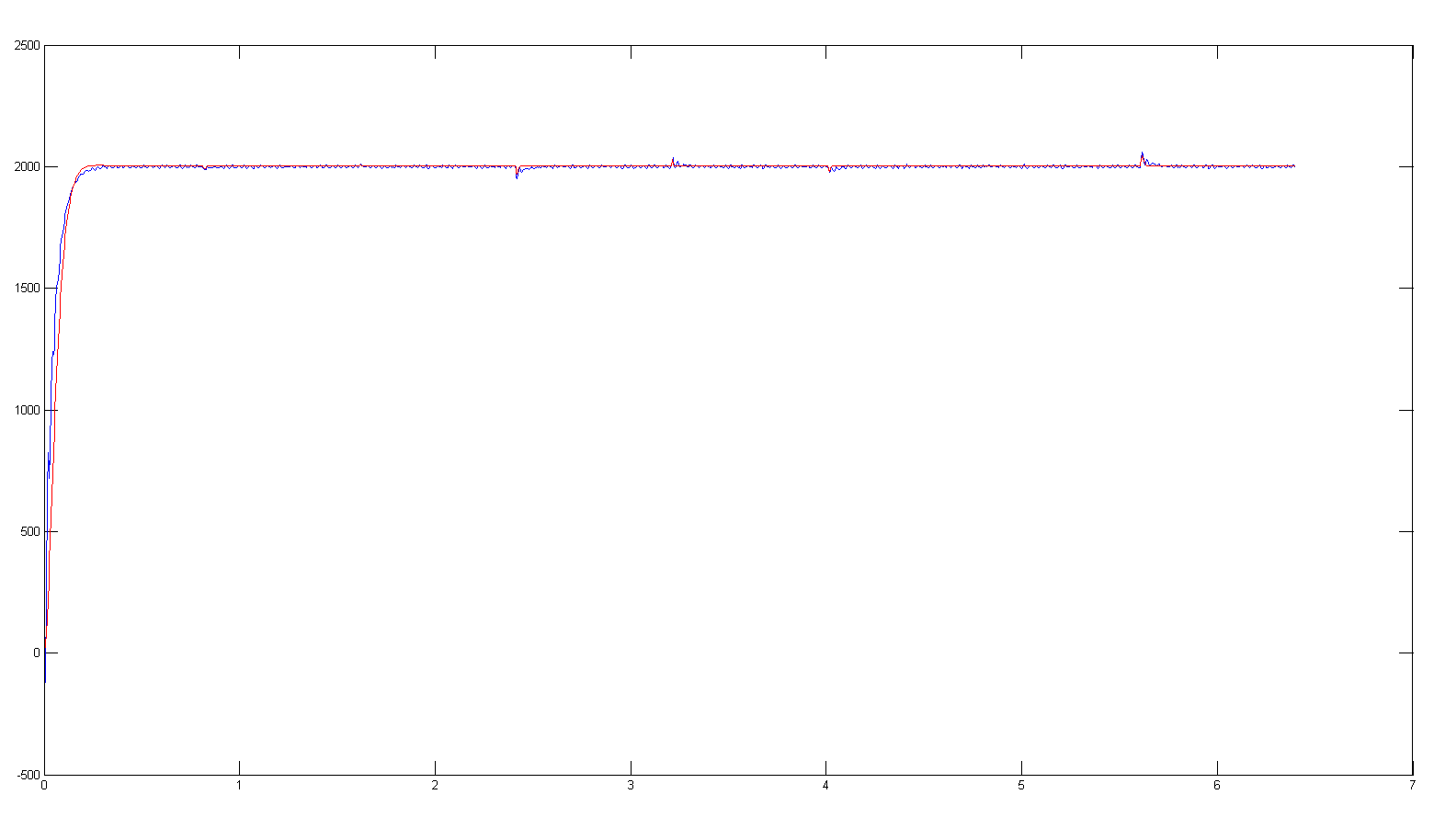


Figure 6: Experimental (blue) and simulated (red) load affected data superimposed.

A closer, more detailed view of two of the load shift points can be observed in Figure 7.

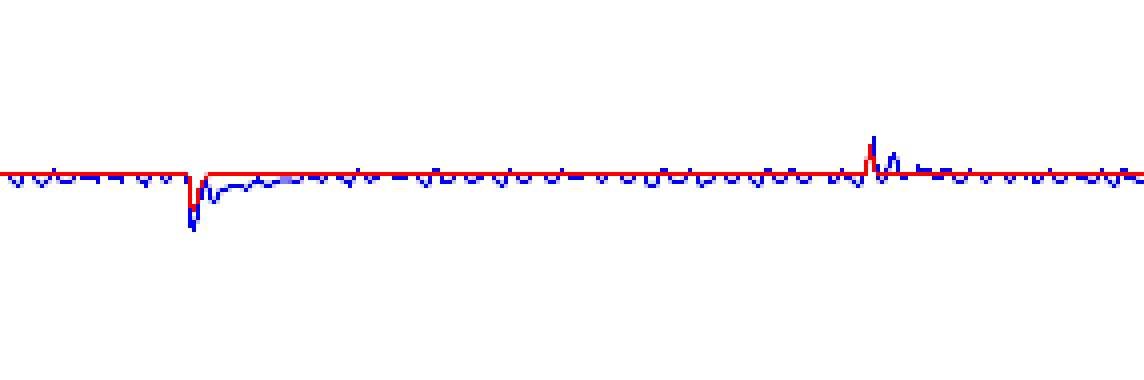


Figure 7: Zoomed in view of the resistor number changing from 2 to 13 and back down to 6.

It is clear that the simulation is compatible with the experimental data, as both plots respond to the changes in the loads accordingly.

1. CONCLUSION

In conclusion, it is apparent that the experiment functioned quite well compared to the simulation. By observing figures 3, 6 and 7, we can identify that there is not too much discrepancies, especially after both plots reach the steady state of 1000rpm or 2000rpm.

However, there is quite some discrepancy in the initial rise period of the controllers. By looking at Figure 6, we can see that the simulated plot rises quicker than the experimental plot. This is most likely due to the presence of friction in the system, and at such high rotational velocities, this is expected.

As well as this, there is very obvious ringing in the experimental plot at the steady state, as a zoomed in image of a section of one of the experimental plot shows in Figure 8.

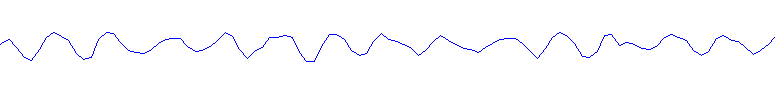


Figure 8: Ringing present in experimental plot.

This ringing is relatively small, peaking at +/- 0.5% deviation from the desired rotational velocity, which does not increase as time goes on. This ringing is most likely due to the lack of accuracy in the speed sensor.

Additionally, it is clear that not only has the simulated model manages to outperform the experimental rig controller as seen by the lower deviation from the steady state line, and the shorter recovery times in Figure 7. This could be due to friction in the system again, or even inaccuracies with the resistor system. The very non-uniform effects of the resistors can be observed in Figure 4.