Jed Storey
Mechatronics Project 4
8/31/2014

Motor System Identification

1. Simulink model structure

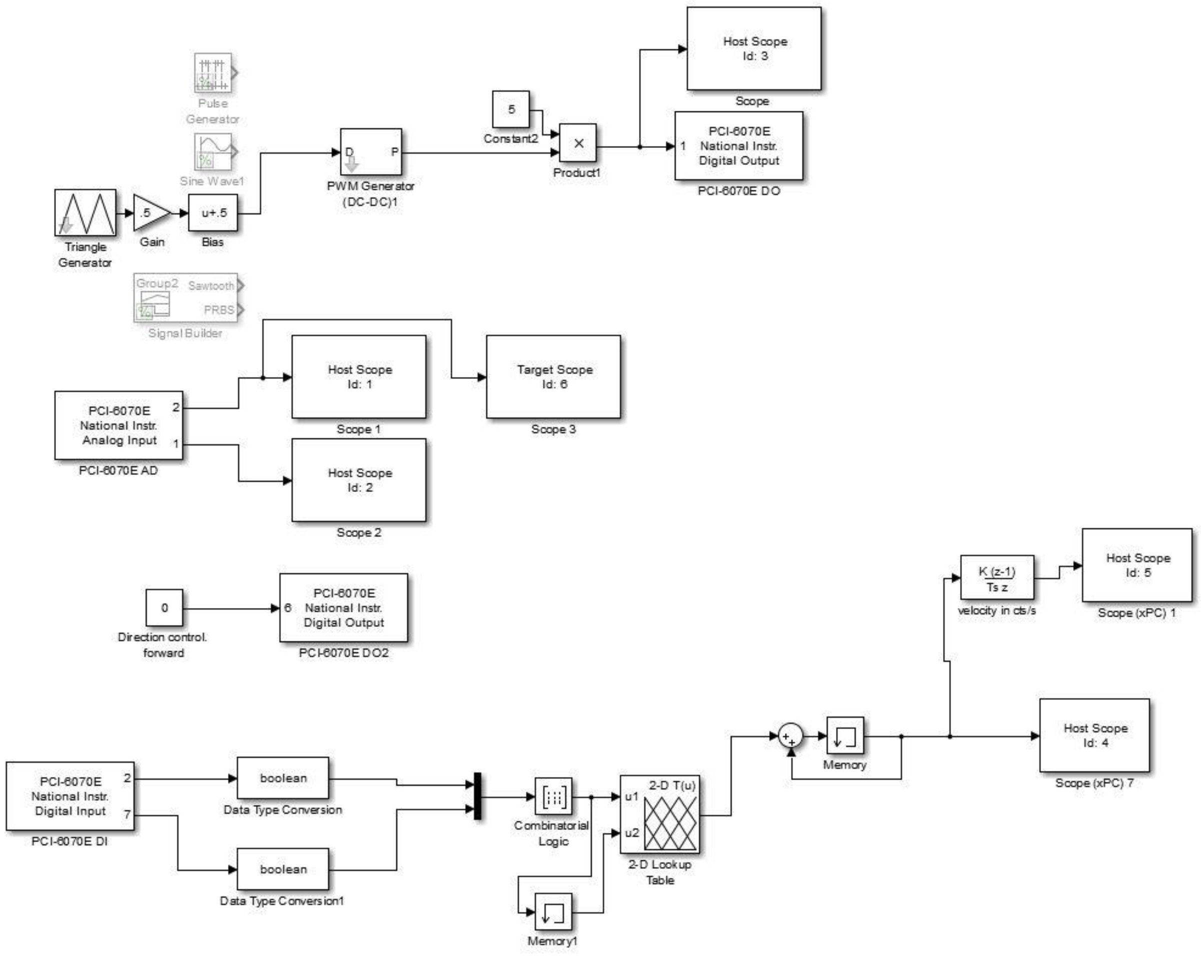

Figure 1. Simulink Model

Starting from the top of Figure 1. Five waveform patterns are selectable: Pulse, sine, triangle sawtooth, and PRBS. Frequency, amplitude, etc. are adjustable inside their blocks. These go through a PWM generator, which works by comparing a sawtooth waveform of the selectable carrier frequency to the input waveform to generate a PWM waveform output. The PWM waveform was recorded by a scope for error checking purposes and output of a DO port. This DO port went to IN1 pin on the motor driver.

Next is the analog reading of motor current and voltage. An external current sensor was wired with a differential input to the DAQ. A differential voltage measurement was taken at the motor leads and input into the DAQ. These signals were saved by scopes.

Direction control was handled by writing a 0 to a digital out port connected to IN2 on the motor driver. For reverse, PWM was sent to IN2 and a 0 was written to IN1.

Both encoder channels were read directly into the DAQ. A state model/look up table was used to convert from encoder pulses to up or down counts. While this project was only done in one direction, this method enables bidirectional position measurement. The position signal was then differentiated to obtain velocity. Both position and velocity were recorded with scopes.

Global sample time was 5E-5s (20kHz) and each run was 30s long. The computer hardware was 116’s dedicated XPC target, which has a NI 6070e DAQ card installed. I made a new kernel for it on a CD because the old kernel was on a floppy drive. My laptop, running Matlab 2014a, was hooked directly to it via Ethernet.

Below is a picture of the physical setup.

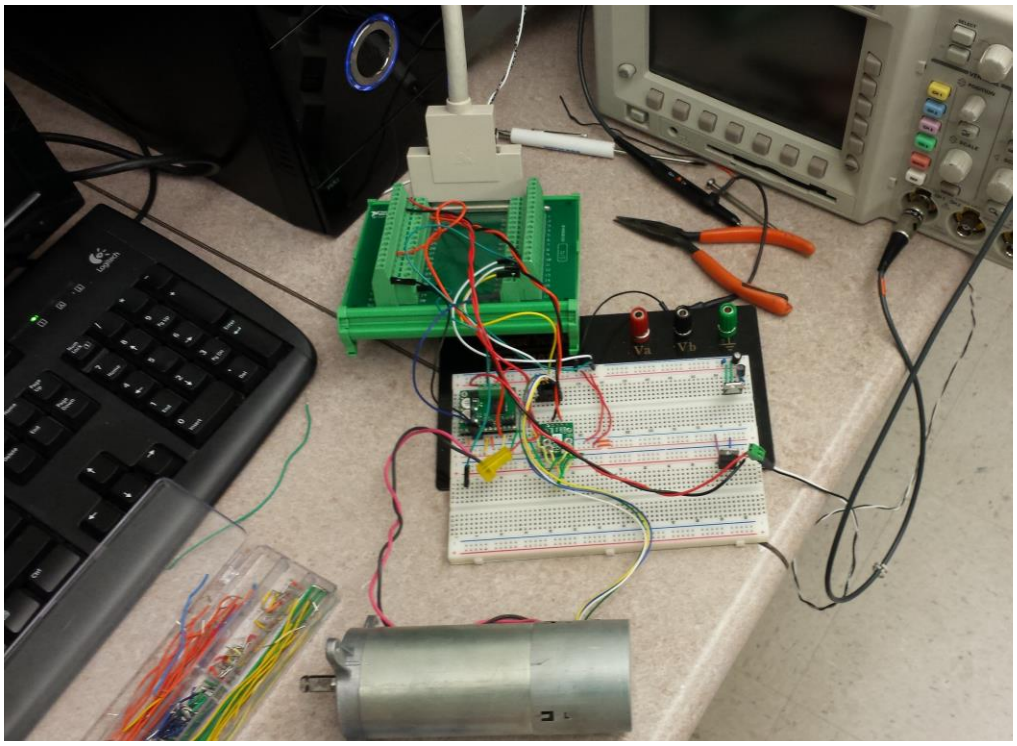

Figure 2. Physical setup

The current sensor and motor driver are on the breadboard. The motor is lower center. The DAQ accessory breakout board is the green thing center. The XPC target is in upper left. An oscilloscope (upper right) was used for debugging.

2. MATLAB Model Structure

Five MATLAB scripts were written for this project: two for the first model and two for the second model. The two pairs are very similar, so only the structure of the first model (three states) will be discussed here.

a. Data saving script

This is a short script that reads the xpc host scope data, places it in vectors, and saves it as .mat files.

b. Post Processing Script

This script reads in the .mat files output from the data saving script. Filtering is done on the data that needs it. The filtering is accomplished with idfilt and trial-and-error for idfilt parameters. A “noncausal” filter was specified to prevent phase offset. The current is then offset and scaled appropriately; the offset and scale were obtained with a handheld ammeter during testing. Angular position and velocity units are converted into rad and rad/s respectively.

Iddata objects are then created. The PRBS data is split in 2/3-1/3: 2/3 for model building and 1/3 for model validation. The whole square and triangle wave data is used in model verification. Iddata characteristics, e.g. names, are then set.

c. Sys_Id Script

This script first executes the post processing script. It then creates the A,B,C,D,K, and x0 matrices that represent the model. For Model 1, the various parameters were estimated from knowledge of brushed motors and engineering judgment, as opposed to guesses. Examples: J was estimated from the dimensions and mass of the rotor being modeled as a solid cylinder. K_b and k_tau were both estimated from the data published on the datasheet for the motor. As a result, no iteration in these values was required. For Model 2, I used the link provided on the problem statement. This link showed the use of two parameters, k and tau. Since I could not find the document MATLAB references that defines these in terms of real motor parameters, I had to guess at their values.

Next, and idss object is created and the free elements of the matrices are set. A PEM search is then run to determine the various free element values based on the model building PRBS data set. Then time-domain comparisons of the model with the PRBS model building data, PRBS model validation data, square wave verification data, and triangle wave verification data are created.

Next, the frequency response characteristics are analyzed. The bode plot of the transfer function of the model is first made. Then the frequency response of the time-domain data (PRBS model validation data) is created using the spafdr function. The bode plot of that data is overlaid onto the bode plot of the model.

3. Model Parameters

The model parameters considered and discovered (via PEM) are listed below.

a. Model 1

A = [(-R/L) 0 (-kb/L);0 0 1;(ktau/(J)) 0 (-c/(J))] = [-548.6 0 -7.3;0 0 1;2773 0 -13]

B = [1/L;0;0] = [119.3;0;0]

, where R is the resistance [Ohms], L is the inductance [H], kb is the backemf constant [V/(rad/s)], ktau is the torque constant [N-m/A], c is the friction constant [N-m/(rad/s)], and J is the inertia [kg m^2].

b. Model 2

A = [0 1;0 (-1/tau)] = [0 1;0 -48.9]

B = [0;(k/tau)] = [0;591]

, where k is the “motor static gain constant” [rad/V-s] and tau is the “motor time constant” [s].

4. Time-domain Comparison Plots (Model 1)

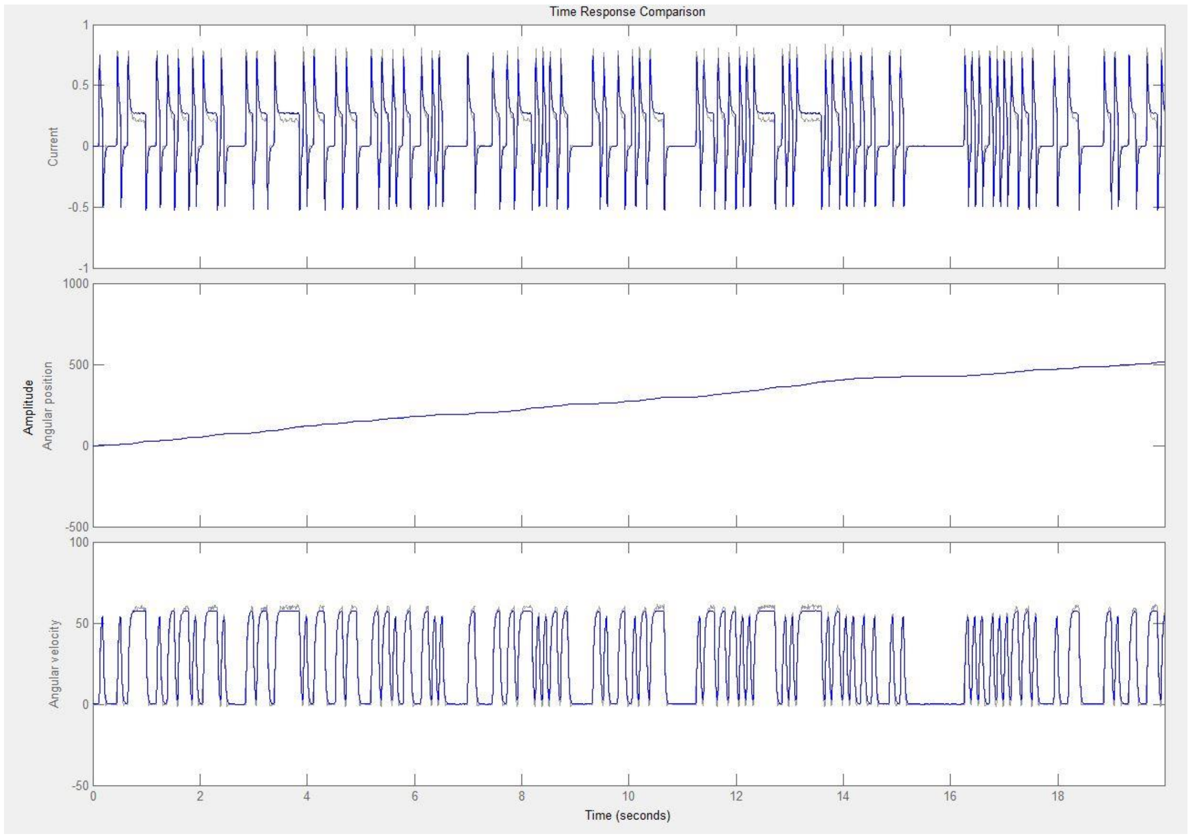

Figure 3: PRBS model building data (grey) vs. model (blue)

Current fit: 78.72%

Position fit: 99.7%

Velocity fit: 93.1%

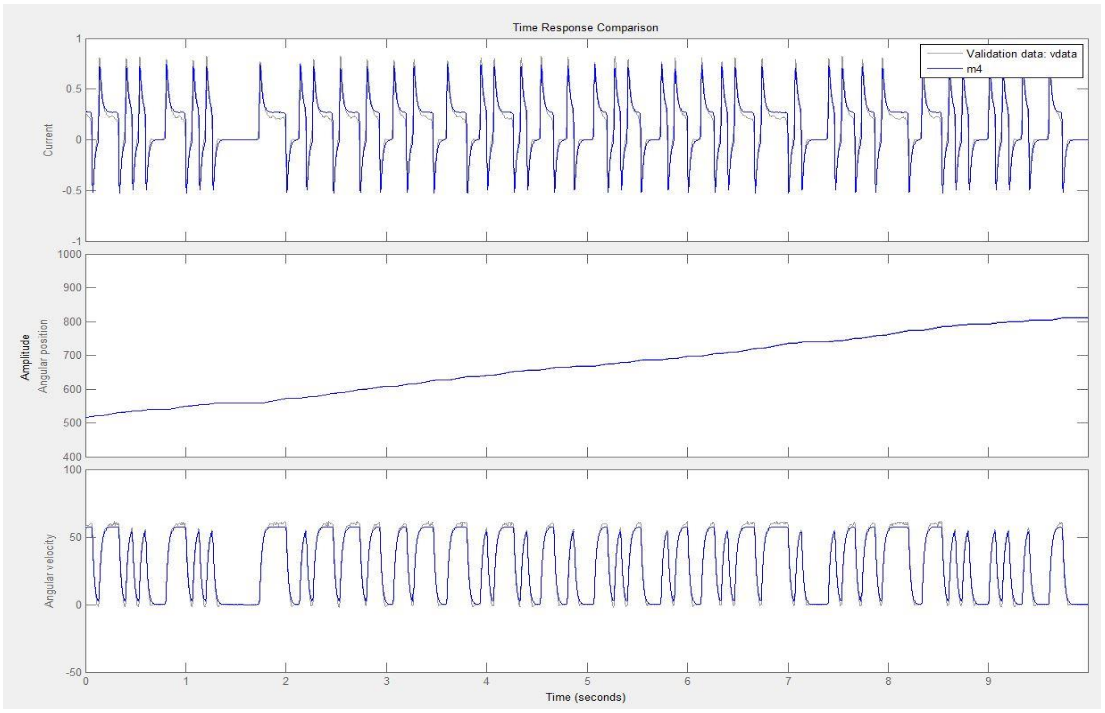

Figure 4: PRBS model validation data (grey) vs. model (blue)

Current fit: 78.15%
Position fit: 98.55%
Velocity fit: 92.06%

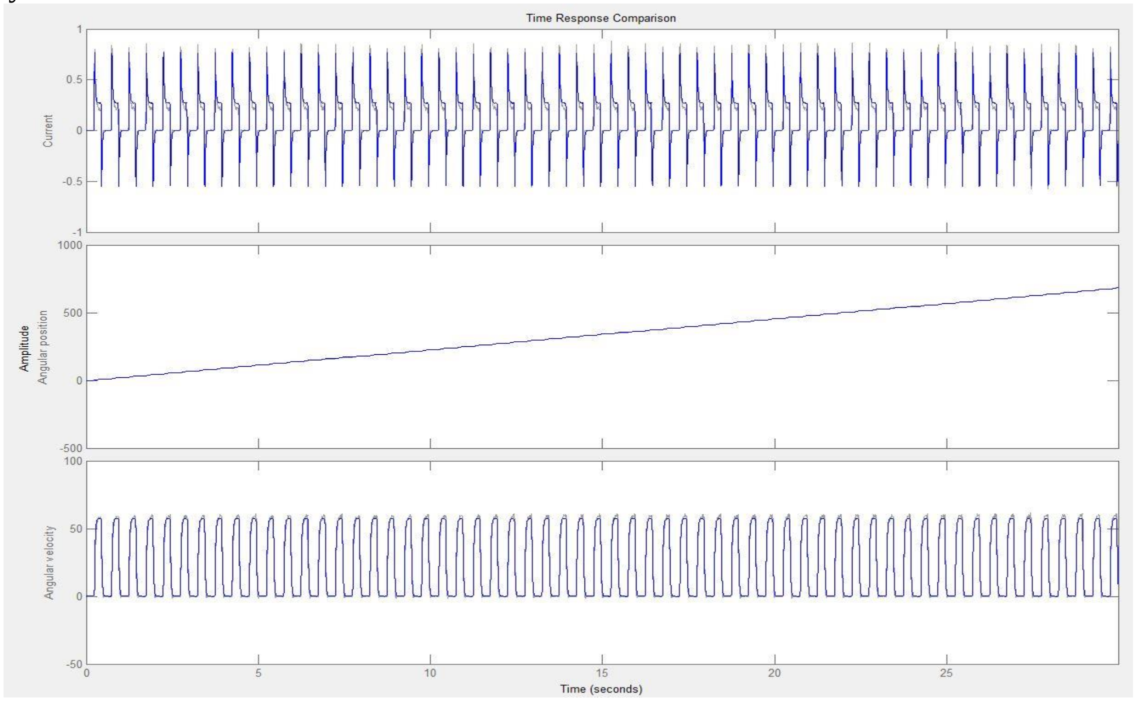

Figure 5: 2 Hz Square wave verification data (grey) vs. model (blue)

Current fit: 81.01%
Position fit: 99.62%

Velocity fit: 94.43%

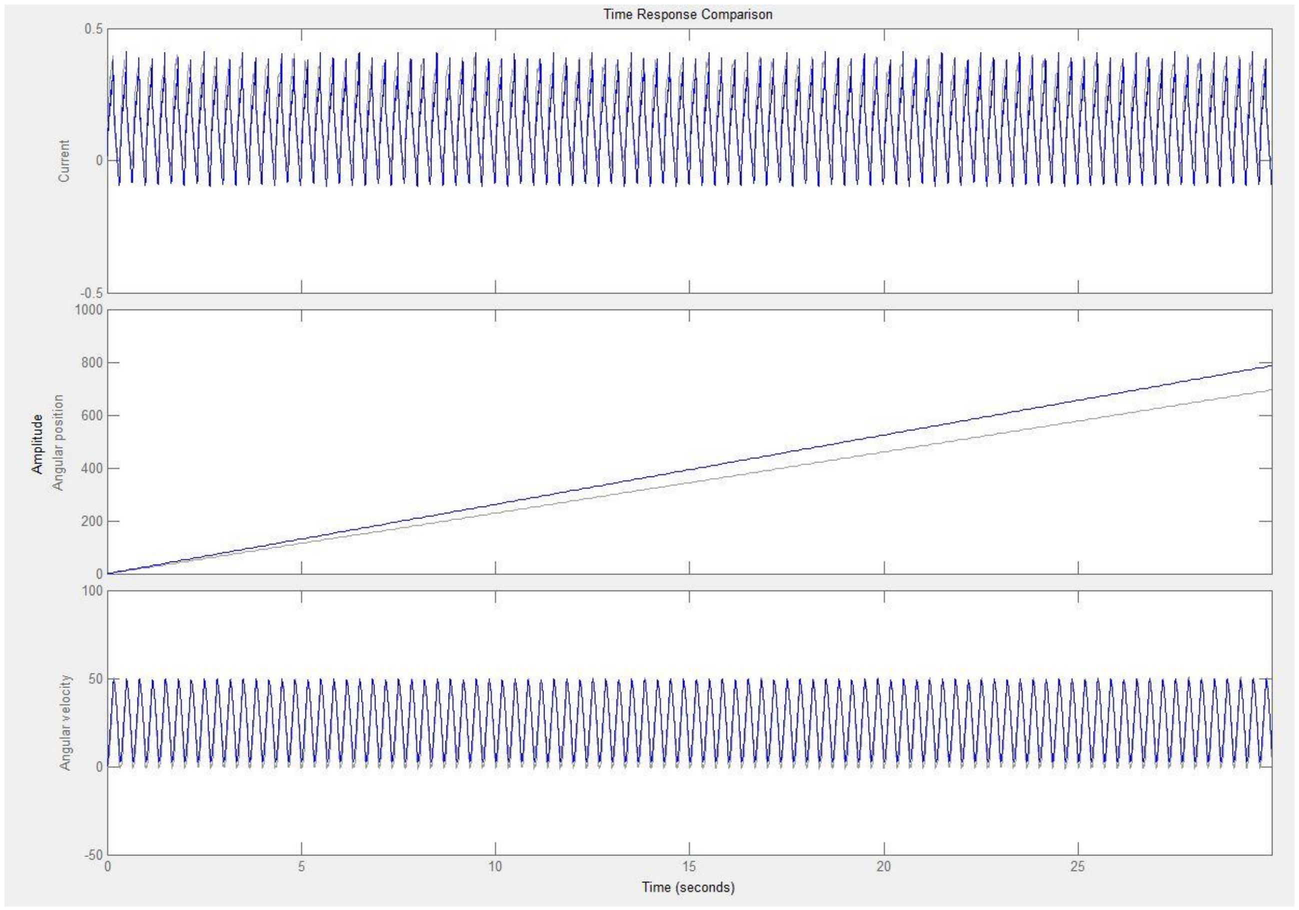

Figure 6: 3Hz Triangle wave verification data (grey) vs. model (blue)

Current fit: 52.08%
Position fit: 72.57%
Velocity fit: 76%

5. Frequency-domain Comparison Plot (Model 1)

The following plot is a bode diagram show the response from voltage to current, position, and velocity. It is organized as current magnitude, phase, position magnitude, phase, velocity magnitude, phase.

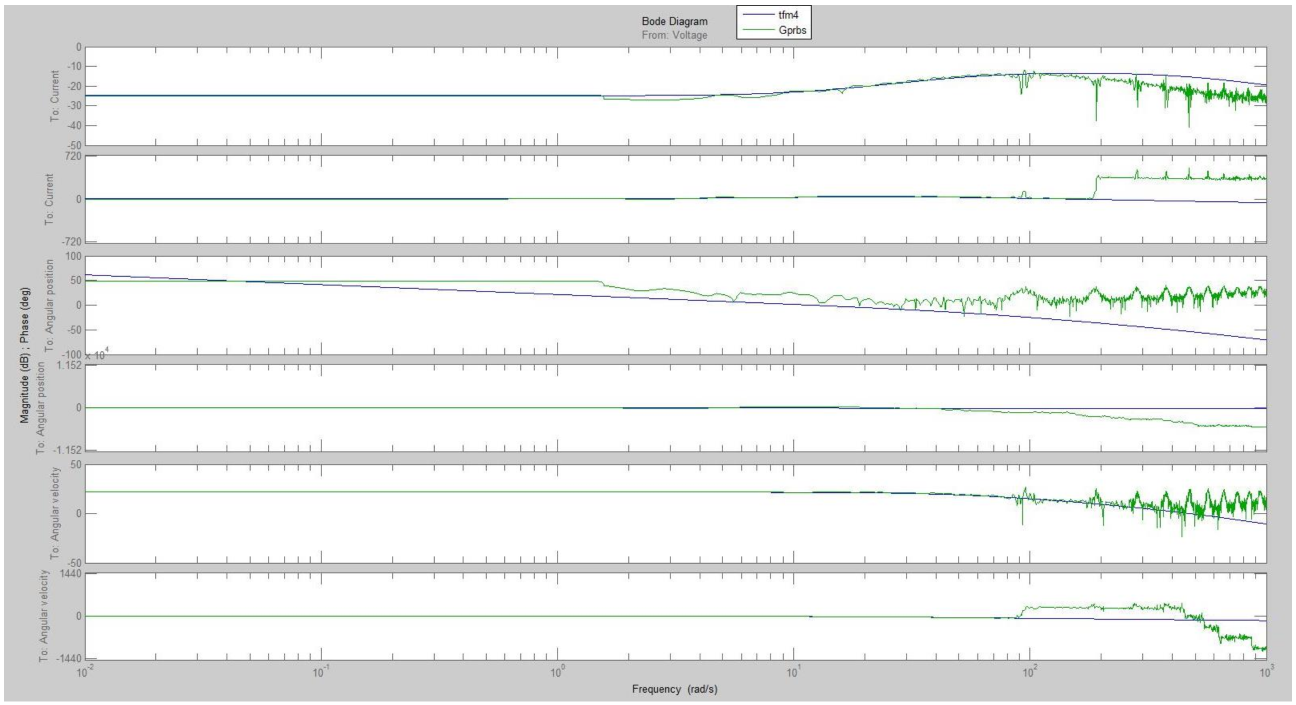

Figure 7: Bode plot. Model (blue) vs. PRBS validation data (green).

The frequency response compares well until about 60 rad/s (~10Hz).

6. Time-domain Comparison Plots (Model 2)

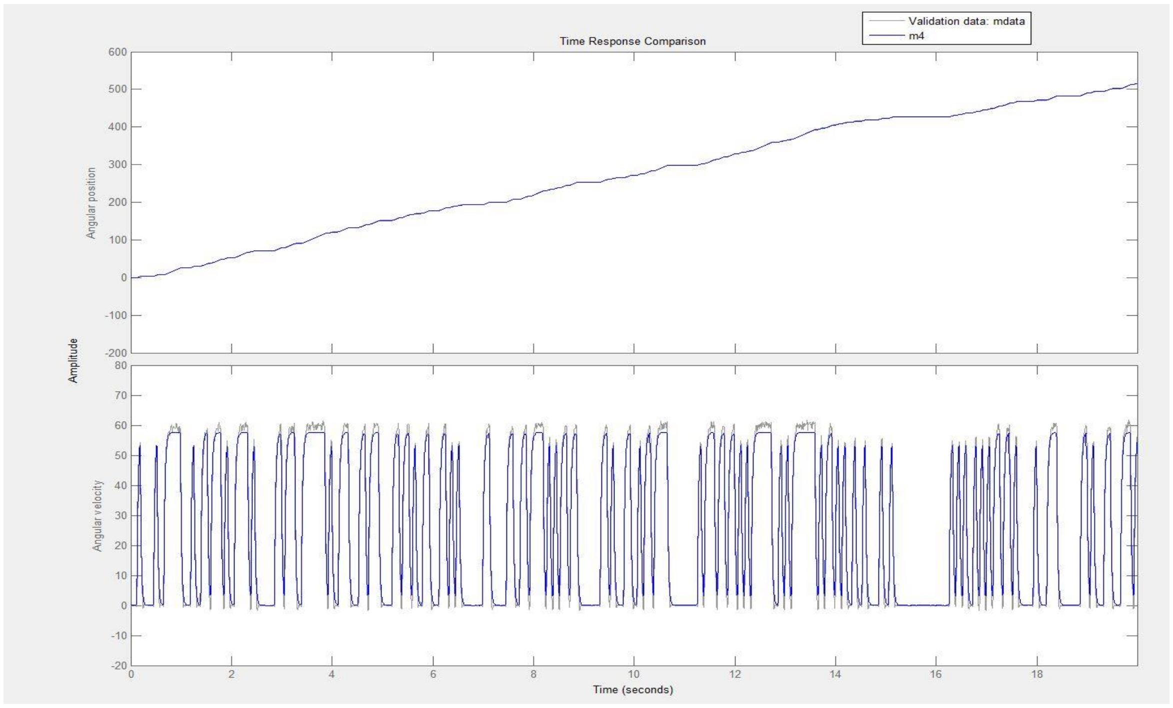

Figure 8: PRBS model building data (grey) vs. model (blue)

Position fit: 99.7%
Velocity fit: 91.28%

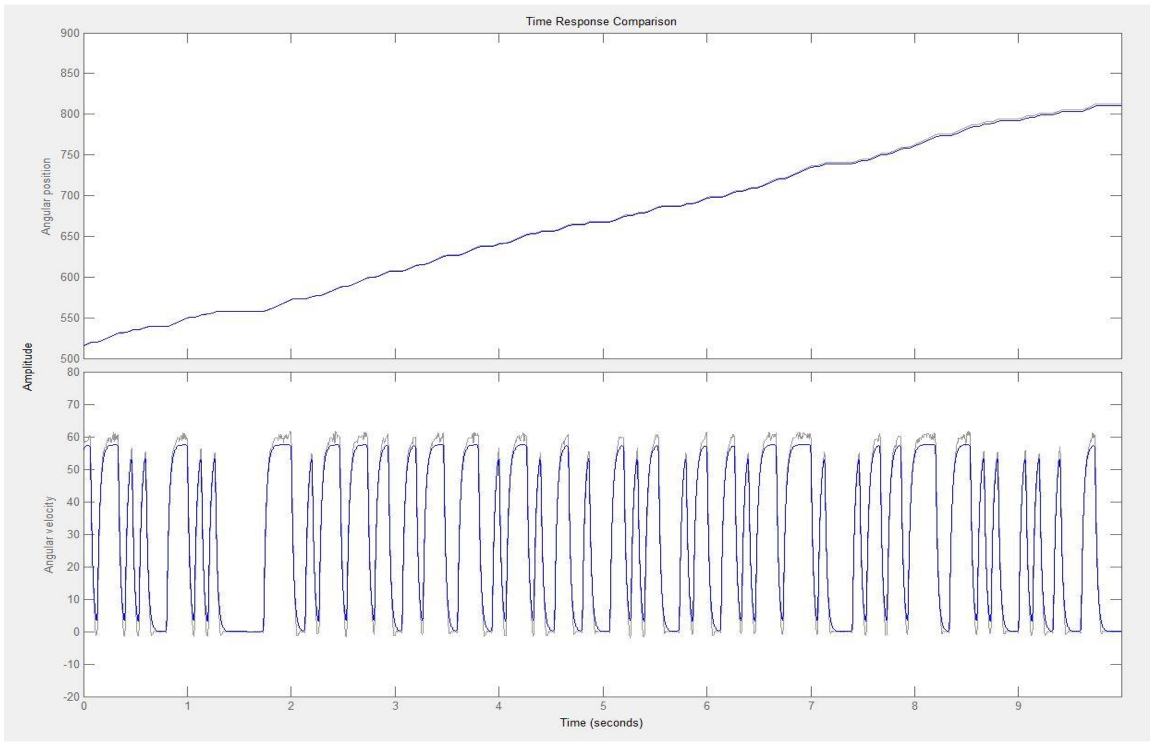

Figure 9: PRBS model validation data (grey) vs. model (blue)

Position fit: 98.54%
Velocity fit: 89.98%

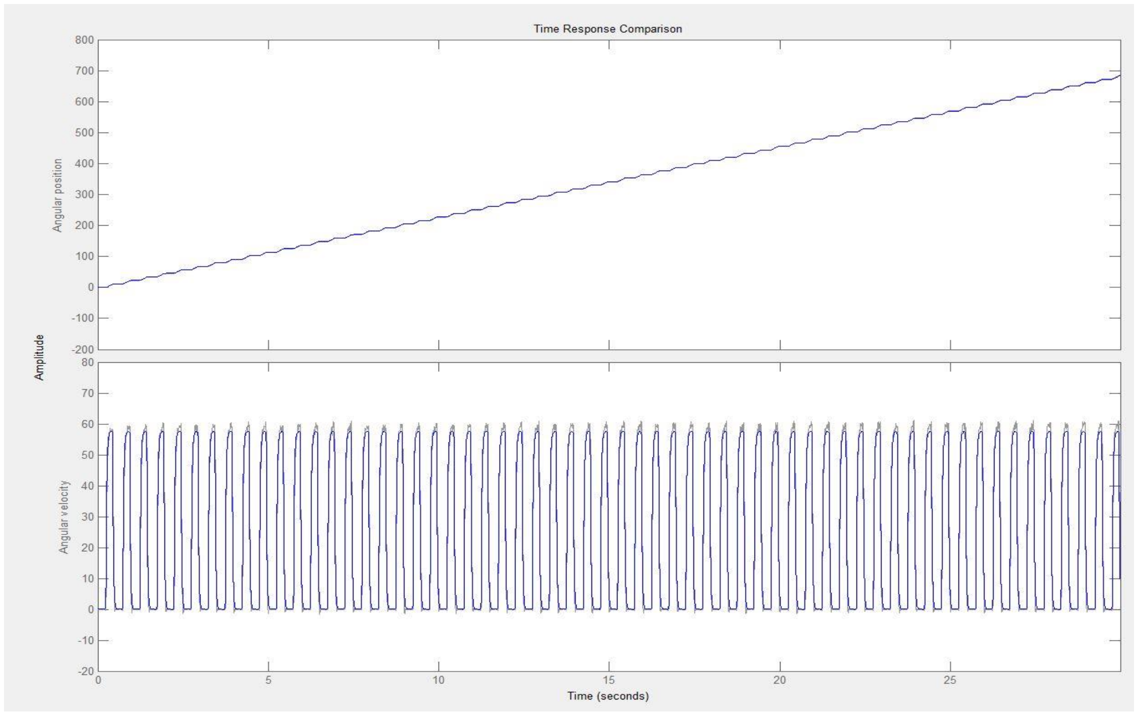

Figure 10: 2Hz square wave verification data (grey) vs. model (blue)

Position fit: 99.62%
Velocity fit: 93.43%

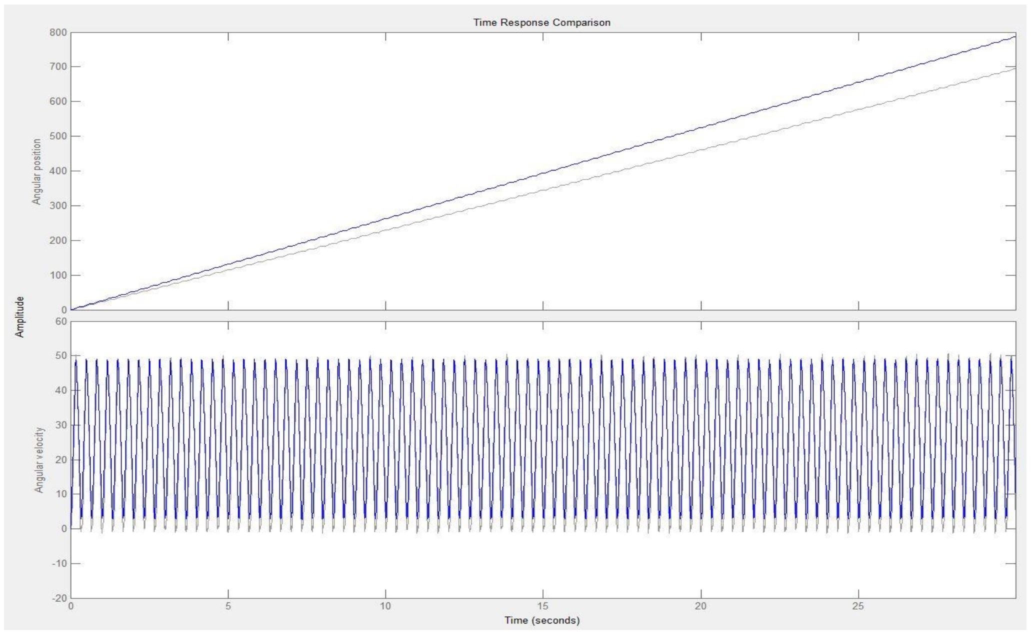

Figure 11: 3Hz triangle wave verification data (grey) vs. model (blue)

Position fit: 72.57%
Velocity fit: 75.45%

7. Frequency-domain Comparison Plot (Model 2)

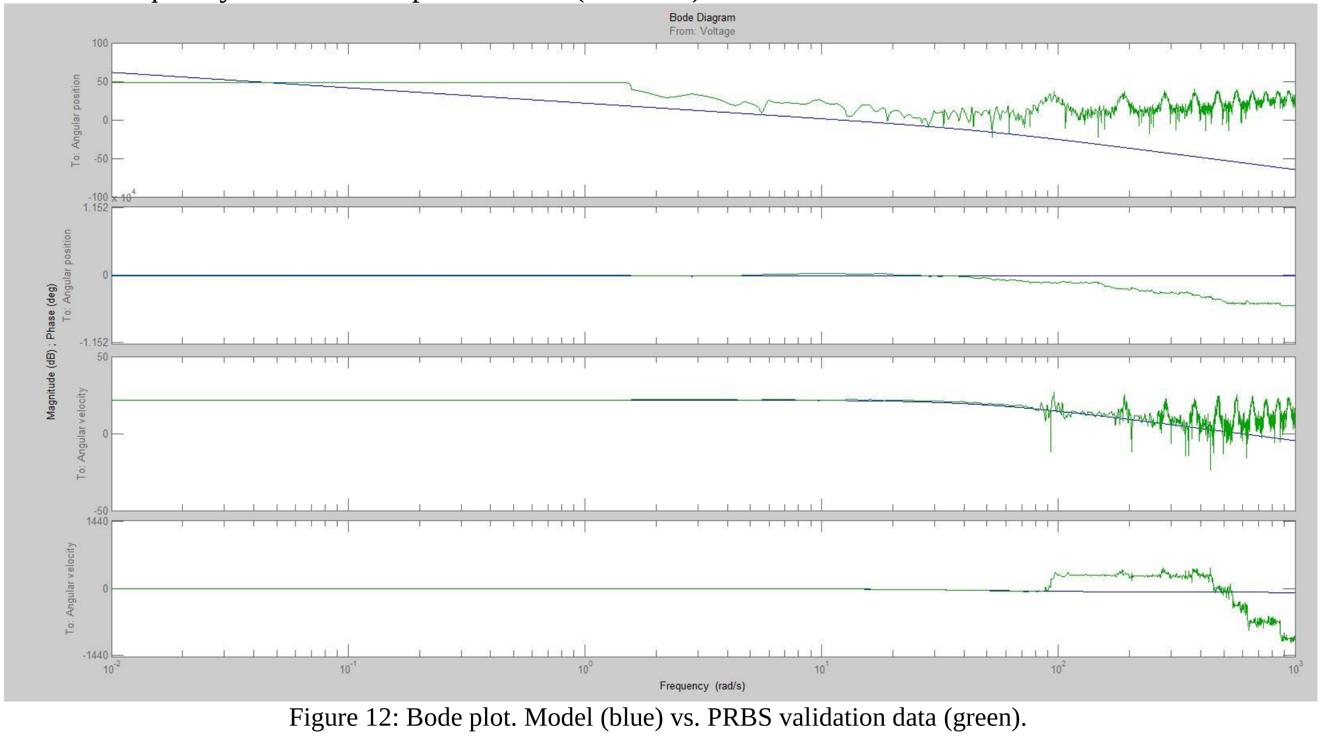

Figure 12: Bode plot. Model (blue) vs. PRBS validation data (green).

The frequency response compares well until about 60 rad/s (~10Hz).

8. Assessment of Model Error

Both models did very well. In model 1, the current prediction wasn't as close as the position or velocity predictions. The general shape is correct, but the percentage match is significantly lower. I believe this stems from noisy current sensor data that had to be heavily filtered, which tends to smear data. Some of the noise (for current and voltage measurements) comes from the fact that the motor is being driven by PWM and not a linear voltage source. A significant source of noise was probably the wiring (see Figure 2). While the voltage and current measurement wires *were* twisted, the general messiness of wiring probably introduced noise.

For both models, the triangle prediction wasn't as good as the pulse (prbs and square) prediction. The reason the position steadily deviates is because the model doesn't predict a zero velocity at the end of every triangle pulse (the test data shows a return to 0 velocity for every pulse). The predicted velocity not quite reaching the peaks can be clearly seen in the PRBS and square wave data, which is probably related to the triangle velocity problem. I am unsure of the source of this error. One possibility is the heavily filtered voltage input data (it had to be filtered down from the PWM waveform driving the motor). Another possible source of error is the way velocity is calculated. Instantaneous (single time step) velocities are calculated in the Simulink model, resulting in high frequency velocity pulses. These are then heavily filtered/averaged to extract the velocity waveform. A better method for calculating velocity probably needs to be implemented. One option to do this might be to use one of the on board counters and dividing by the time incremented associated with the number of counts, but this would only work in one direction.

The frequency response for all parameters both models agrees well out to about 10Hz. Even though the position had the best agreement in the time domain, it has the worst agreement in the frequency domain. The only reason for this that I can think of is that the model was built using noise, which may not be representing the full range of frequencies evenly up to 10Hz.

9. Concluding remarks

All requirements of the project were met. A good model of the motor was made using Simulink, XPC target, and System ID. Improvements to wiring layout and velocity calculation would probably improve the model substantially.

Total time spent on project 4: ~50 hours.
Total time spent setting up hardware and helping people: ~10 hours

Appendix A. Motor Driver Confusion

I spent some time trouble shooting the motor driver. I wasn’t seeing the expected negative currents from freewheeling. (You sent the following out to the class after I sent it to you.)

Referencing the motor driver data sheet page 16:

There are two ways to drive the kit's motor driver chip with PWM. One way is that you can set In1, In2, ground D1, and drive D2 with PWM. In this way, In1 and In2 are direction controls. The other way is ground D1, pull D2 high, pull either In1 or In2 low, and drive either In2 or In1 (respectively) with PWM. You would think that these would result in the same motor behavior...but they don't in one key way.

Option 1: When the D2 PWM pin is low, this triggers a disable state, which forces the Out1 and Out2 into high impedance mode, effectively opening the circuit that the motor is in. This stops any current that might be generated by the motor due to spin down. Our model does not account for this...we aren't modeling the motor driver. (Further note: you will still see a voltage during spin down if you use a high impedance volt meter (like the DAQ), but the current will be essentially 0 because it's acting like an open circuit. All of the left over rotational energy is being dissipated in friction and eddy currents in the motor. While we are accounting for viscous drag, we aren't modelling other sources of friction and energy sinks that suddenly become dominant due to the open circuit. Thus, the model fails to accurately model current).

Option 2: Since D1 and D2 are set, no disable state is ever triggered. Thus, when the PWM pin (either IN1 or IN2) is in its low state, the "free-wheeling" low state (figure 12 d) happens, which allows the motor to generate negative current in its coils due to free-wheeling. This is what our model accounts for.

For the second part of the project (where you don't use current), either option should theoretically work.

You should probably use an external current sensor; the onboard current sensor (FB) cannot measure the freewheeling current.