Interferometer Control System Code Simulation

Juan F. Castillo, The University of Texas at El Paso
Joseph Betzweiser, Staff Scientist
Second Progress Report, LIGO SURF
August 2, 2013

1 UPDATE

The objective of this project is to successfully implement a real-time simulation of the control code to the cavity and suspension models of the interferometer. The examination of the background knowledge in order to comprehend the system at a fundamental level was the basis of the previous objectives. Armed with this information, I have focused on two aspects of the project, the Matlab scripts that synthesize the filter definitions and the Matlab Simlink model, which is parsed by a script to generate the simulation code. These two facets, while sounding one-dimensional, involve various complex procedures. Accordingly, the filter definitions are the mainframe that accurately maintains the simulation model.

1.1 FILTER DEFINITION

I have written several Matlab scripts for the filter definitions. These scripts create individual filters that simulate actual noise taken from frequency response data in a Matlab array. The array was provided to us as data that had been gathered over one night. The filters are representations of this data and are realized by zero-pole gain (zpk) models. I had previously written the script to iterate a given number of times to obtain a more accurate representation. I changed the method of the script so that it can converge to an adjustable tolerance. This procedural change produces a more accurate model of the data set while ensuring the program runs more efficiently.

The script has had to be revised several times in order to produce a better output. Other than the changes mentioned above, the program was having difficulties accurately reproducing signals with a very small frequency magnitudes. By adjusting the “weight” option in the vectfit4 function, the program was better able to simulate the response behavior. The LIGO lab uses a function called vectfit4 that takes a vector (transfer function data set, amplitude and phase) and fits it to a state-space model with a common zero and pole set.

After these adjustment were made, another section of the program had to be added in order to build 72 filters out of the 6x12 array. Previously, the script was given a cell and outputted the results of one filter. The new script was altered to iterate through each row and column of the data array systematically. By cycling through the entire array, all 72 filters were coded at once. After the filters are in zpk format, they are processed into a function called autoquack and other related sub-functions. These functions prepare and write the zpk

models to the simulation model. The filters are embedded to the system through the use of the Foton (filter online tool) program.

Foton takes a zero pole gain model and attaches the filter to a predefined Simulink filter block. The tool reads and writes coefficient files for the online system to use in the specified block. These parameters are copied into a previously created text file representing the Simulink model.

1.2 Simulation Model

The simulation model is created using the program Matlab Simulink. Simulink is used as a graphical data entry tool that defines the controls algorithms. These algorithms are used to operate the real-time code of the system. Simply put, Simulink is used as a drawing tool whose output is parsed by script to generate C code, which is the actual code used to run the simulation. The first system that I have been working on represents the Horizontal Access Module for the Internal Seismic Isolation System (HAM ISI). We utilized a schematic previously used to operate the system and added my simulation block component which we named HAM2_SIM. The figure below shows the component HAM2_SIM block and how it relates to the schematic.

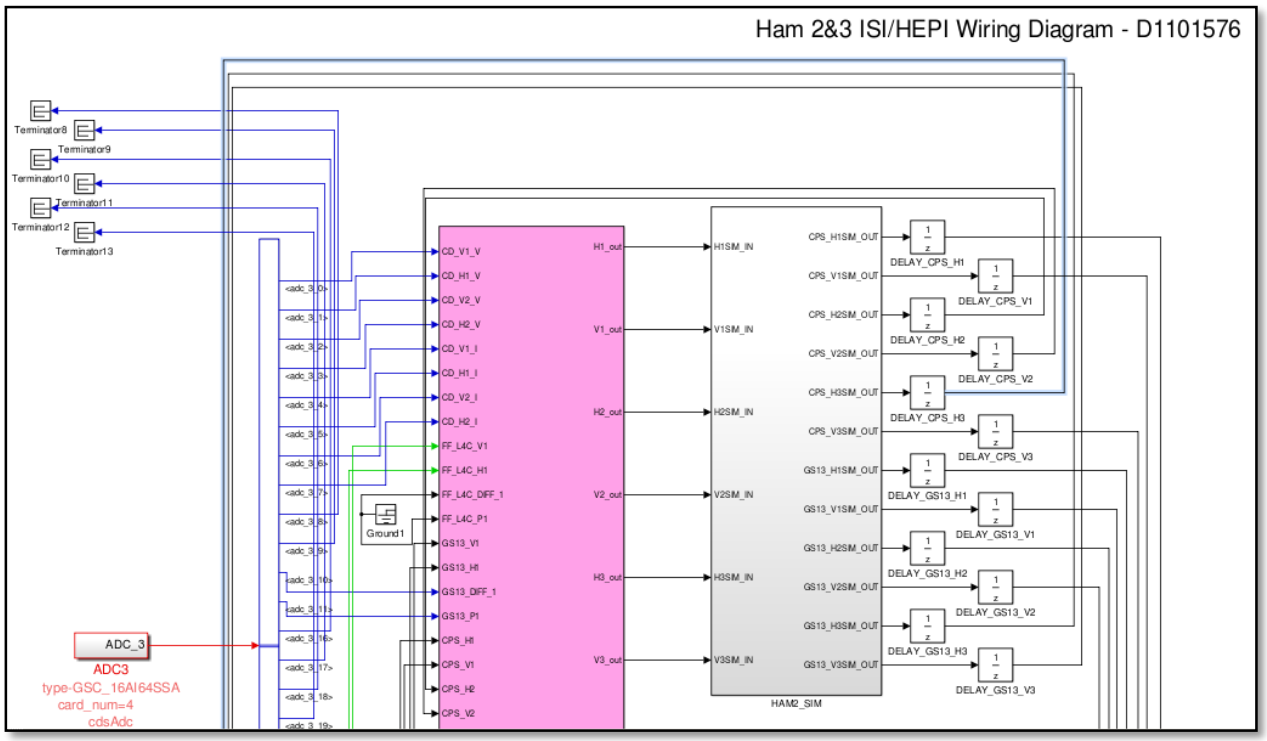

Figure 1: Simulink Model for the HAM ISI Simulation

The HAM2_SIM block component contains various stages of filtering that will be used to simulate the noise of the 6 actuators and the 12 sensors. This component will use as inputs the 6 actuators outputs from the HAM2 component and output our simulated noise to 12 sensors signals. These signals will then be inputted back into the HAM2 component. This process closes our system into a real-time operational simulation. The figure below illustrates what is contained in the HAM2_SIM component block thus far.

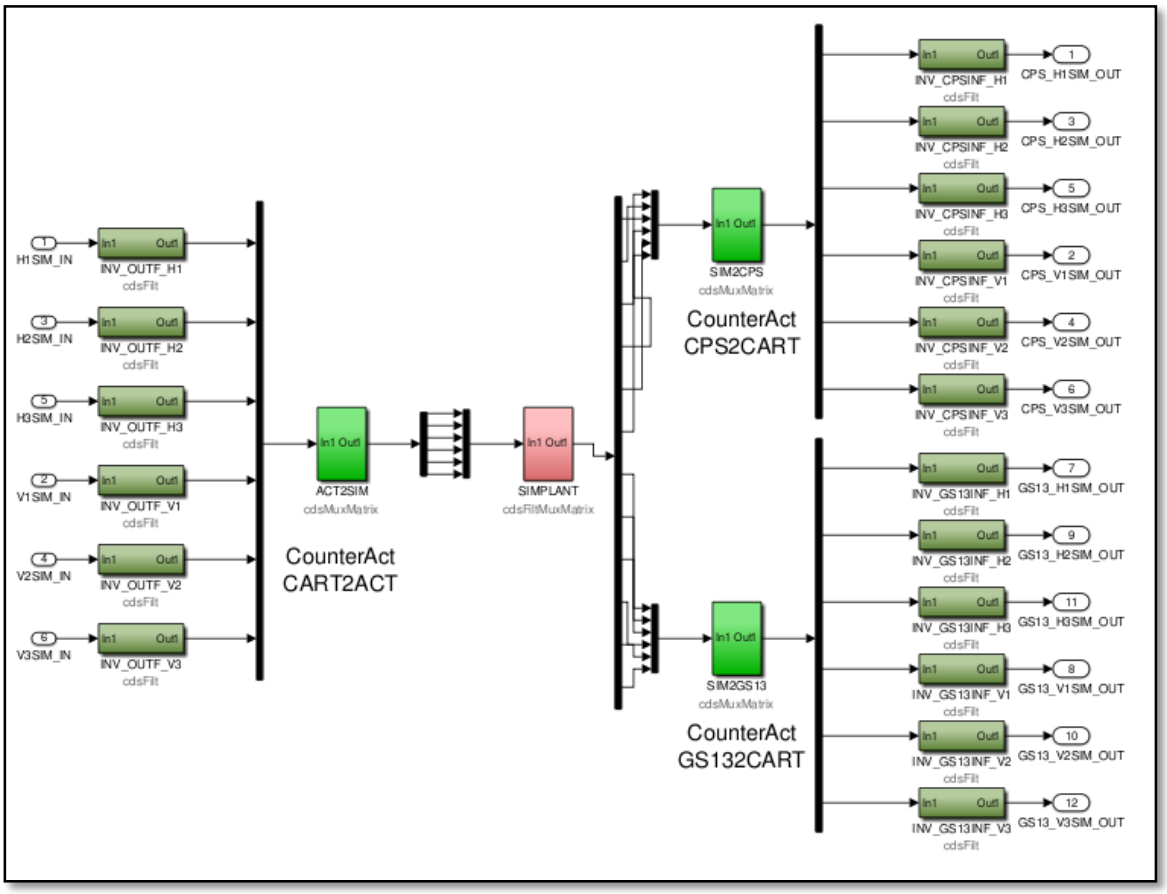

Figure 2: HAM2_SIM System Components

The first stage of filters, ACT2SIM, invert the matrix and convert back to the Cartesian basis of the internal seismic isolation (ISI) system from its actuator basis. This lets us apply the filters which were created from transfer functions measured in the Cartesian basis. After the data has been adjusted, it is inputted into the SIMPLANT component block. This is where the filter noise simulations from my zpk models act upon the inputted data. The 12 sensor

data outputs are separated by its respective sensor, the CPS and the GS13. Similarly, we invert the GPS2CART and CPS2CART matrices and apply them to do a similar process on the sensor side, namely go from the Cartesian basis to the sensor basis. The filtered data is outputted by the HAM2_SIM component and ready to be inputted back into the HAM2 component.

2 PROGRESS

2.1 THE PROCESS

The process outlined below is an extreme simplification of the steps taken to achieve the overall simulation. Hopefully this will further clarify the progress of the work accomplished so far.

1. The frequency response data is extracted from the .frd Matlab file.
2. The data is then translated into a form in which vectfit4 estimates the zpk models.
3. Through the autoquack function and related sub-functions, the data is translated into a Foton readable form.
4. The specified system block's text file is embedded with the filter models translation.
5. The actual data coming out of the main system is converted by our series of filters into a real-time simulator.

The project initially required an intense amount of background research and knowledge. The project has now progressed to applying much of that knowledge. I have been able to tweak the Matlab scripts to improve the model's accuracy. The Simulink diagram for the simulation has been easy to manage but requires patience and diligence to create.

Figure 3 shows the representation of one filter, cell (2,4). The graph was created using a Matlab bode plot. It represent the magnitude and phase of the filter in the frequency domain. Figure 4 shows the same filter that was created with my script that Foton reads. It is also the magnitude and phase in the frequency domain of the same cell. The two graphs match perfectly. Keep in mind that Matlab graphed the phase plot with a range of +180 and -180 degrees, which explains the fluctuations in the graph. These two figures effectively prove that the Matlab script created executes and writes the zpk model filters correctly into the Simulink model.

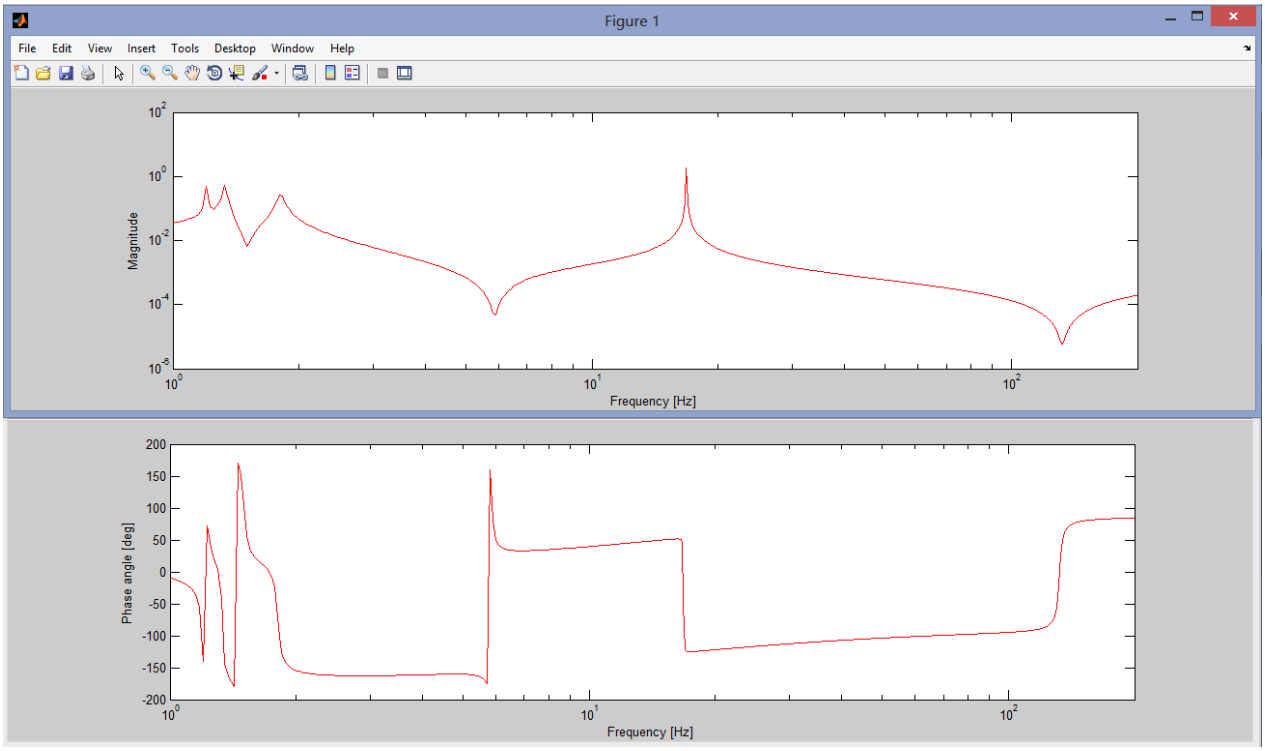

Figure 3: Filter Definition in Matlab

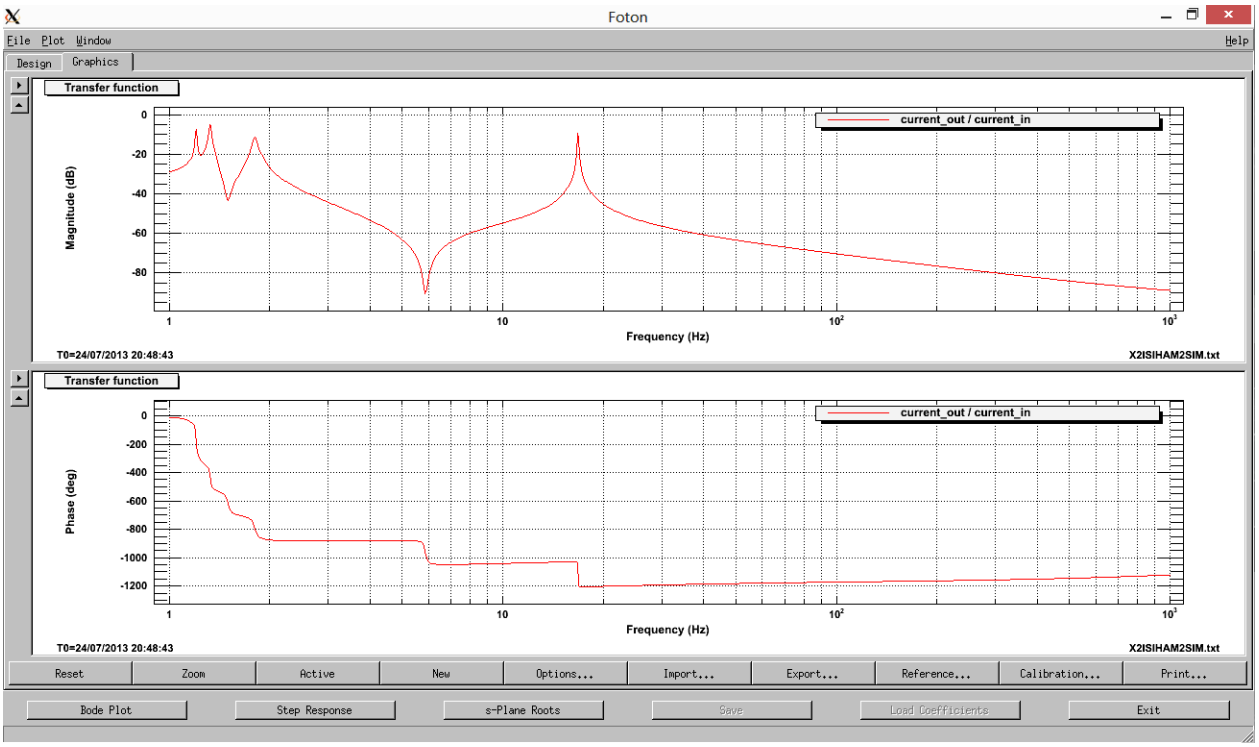

Figure 4: Filter Definition in Foton

3 CHALLENGES

The challenges this last month included many that were procedural in nature. I am still getting used to navigating through the LIGO directory system. The system is vast and it can be difficult finding what I am looking for. Fortunately what it lacks in simplicity it makes up for in organization. Particular files are located in sets that pertain to similar tasks. I have also improved my skill set using the command line. There are still many commands and system tricks that I have to learn. Admittedly, the system and certain procedures have special methods that I am learning as I progress in the project. I am still improving my efficiency in using these methods and look forward to further progress.

4 GOALS

We have several goals to accomplish for the remainder of the project. First of all, we must add another simulation plant to my current schematic. This block will add ground noise to the system. We will create a random noise generator that will produce ground noise filters, which will simulate random ground movements. We should then be able to simulate a "push" in one direction on the system. We want to see the controls damp the simulated ground noise in the same way the controls damp the real ground noise. We want the full control loop to operate under the same conditions to that of the actual system. After the HAM ISI is completed, the project can progress to create simulations for other parts of the system. We have laid the groundwork for future simulations to be incorporated in an overall effective simulator to test the control code and hardware.

5 REFERENCES

[1] Bork, R., and M. Aronsson, *AdvLigo CDS Real-time Code Generator (RCG) Application Developers Guide*. LIGO. https://dcc.ligo.org/public/0001/T080135/003/T080135-v3.pdf.

[2] Deschrijver, D., Mrozowski, M., Dhaene, T. & De Zutter, D., "Macromodeling of Multiport Systems Using a Fast Implementation of the Vector Fitting Method", IEEE Microwave and Wireless Components Letters, vol. 18, no. 6, pp. 383-385, June 2008.

[3] Gustavsen, B. & Semlyen, A., "Rational approximation of frequency domain responses by Vector Fitting", IEEE Trans. Power Delivery, vol. 14, no. 3, pp. 1052-1061, July 1999.

[4] Gustavsen, B., "Improving the pole relocating properties of vector fitting", IEEE Trans. Power Delivery, vol. 21, no. 3, pp. 1587-1592, July 2006.

[5] Lantz, B., Schofield, R., O'Reilly, B., Clark, D. E., & DeBra, D., "Requirements for a Ground Rotation Sensor to Improve Advanced LIGO", Bulletin of the Seismological Society of America, vol. 99, no. 2B, pp. 980-989, May 2009

[6] Saulson, Peter R., *Fundamentals of Interferometric Gravitational Wave Detectors*. Singapore: World Scientific, 1994.

[7] Zhdanova, A., *Real-time Simulation of a Suspended Cavity with the Advanced LIGO Digital Controls System*. LIGO T1200462-x0-v2 (2012).