Chapter 1

System ID with 2 Flexible Base Modes

Performing system identification on Bode data of SAMII operating around a nominal position was a significant part of what I did this semester. Not only did curve fitting take up a significant amount of my time, but it provided me with models that I performed root locus analysis on (root locus analysis was another significant portion of my research this semester).

At the start of the semester, I thought that only the first two modes of the flexible base were significant contributors to SAMII's base vibration. Toward the end of the semester I designed a controller that placed the poles for the first two modes of the base at higher damped locations. This controller made the third mode unstable which lead me to include the third mode in my curve fitting. System ID that includes the first three modes is document thoroughly in Chapter 9. Since that documentation is very thorough, I include only an overview of the curve fitting done with only two modes.

Section 1.1 describes the models used for system id in this chapter.

Section 1.2 shows the results of curve fitting where I was manually tuning coefficients through a trial and error process (this seemed like an easy way to get an approximate model at the time).

Section 1.3 shows the results of my first optimization curve fitting. The error function that I am seeking to minimize in this section does not include any phase error (the 3 mode curve fitting of Chapter 9 does). The error is the sum of the squared magnitude error in dB for both the actuator and the flexible base.

Section 1.4 compares the results from the manual curve fitting to the optimization results and overlays the Bode plots.

1.1 Transfer Function Models

The curve fitting done in this chapter uses the following model for the actuator:

$$\frac{\theta}{v} = \frac{K_1\omega_p^2\left(s^2 + 2\zeta_z\omega_z s + \omega_z^2\right)\tau}{s\omega_z^2\left(s^2 + 2\zeta_p\omega_p s + \omega_p^2\right)(s+\tau)} \tag{1.1}$$

and this model for the flexible base:

$$\frac{x}{\theta} = \frac{B_1\omega_1^2 s^4}{s^2 + 2\zeta_1\omega_1 s + \omega_1^2} + \frac{B_2\omega_z^2 s^4}{s^2 + 2\zeta_z\omega_z s + \omega_z^2} \tag{1.2}$$

1.2 Manual Curve Fitting

The coefficients from my manual curve fitting were

$$\bar{x} = \begin{bmatrix} 20 \\ 51.5221 \\ 0.11 \\ 62.8319 \\ 0.06 \\ 188.496 \\ 10.9956 \\ 0.05 \\ -2.10526e-005 \\ 3.33333e-007 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ B_1 \\ B_2 \end{bmatrix} \tag{1.3}$$

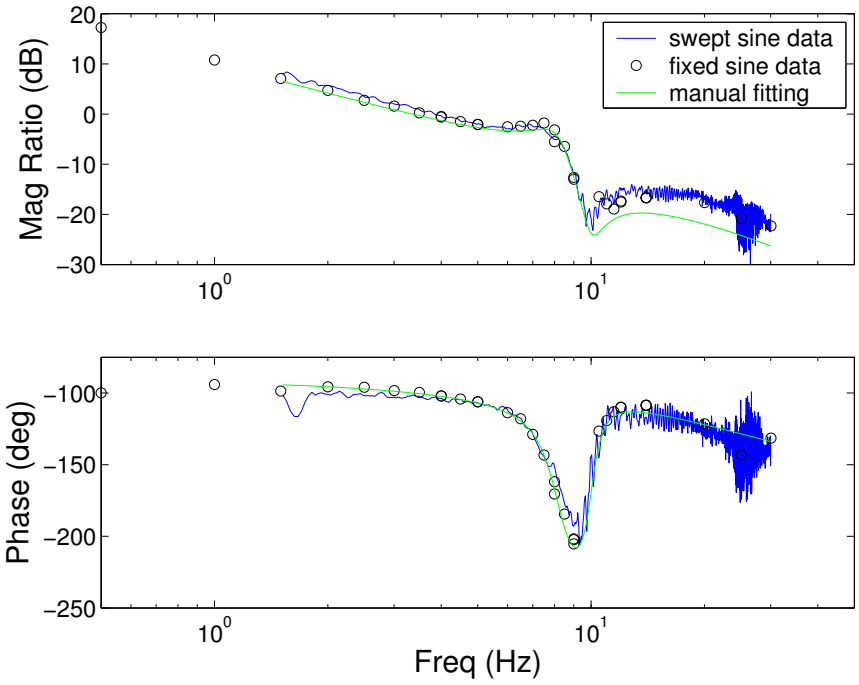

Figure 1.1: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position.

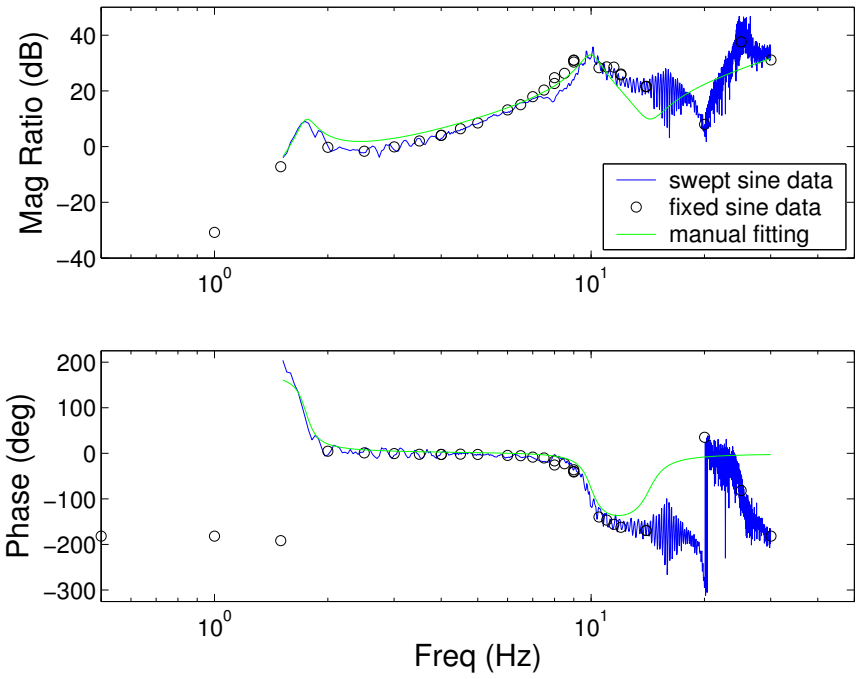

Figure 1.2: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration.

1.3 Optimization

The results shown in equation 1.4 and Figures 1.3 and 1.4 are for an optimization done over a frequency range of 1-12Hz and with an error function that minimizes the sum of the squared magnitude error in dB for both the actuator and the flexible base. This optimization uses a sixth order model with an s^4 term in the numerator of the transfer function $\ddot{x}/\theta$. This fit does not consider phase error at all. This fit includes only the first two modes of the flexible base.

The output coefficients from the optimization were

$$\bar{x} = \begin{bmatrix} 22.0616 \\ 52.3786 \\ 0.119069 \\ 61.5192 \\ 0.0767268 \\ 188.51 \\ 10.5585 \\ 0.0413066 \\ -1.71e-005 \\ 3.786e-007 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ B_1 \\ B_2 \end{bmatrix} \tag{1.4}$$

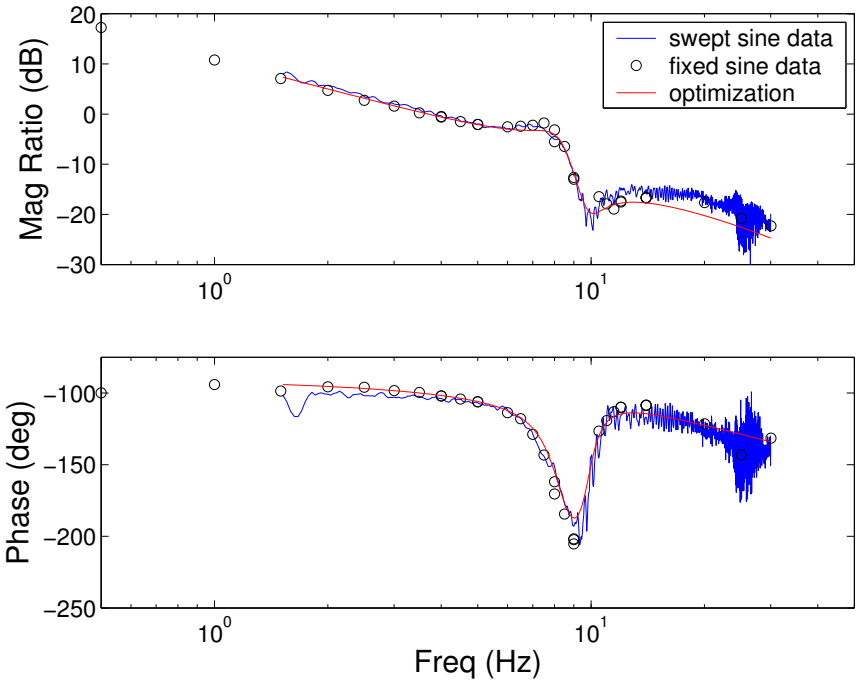

Figure 1.3: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position.

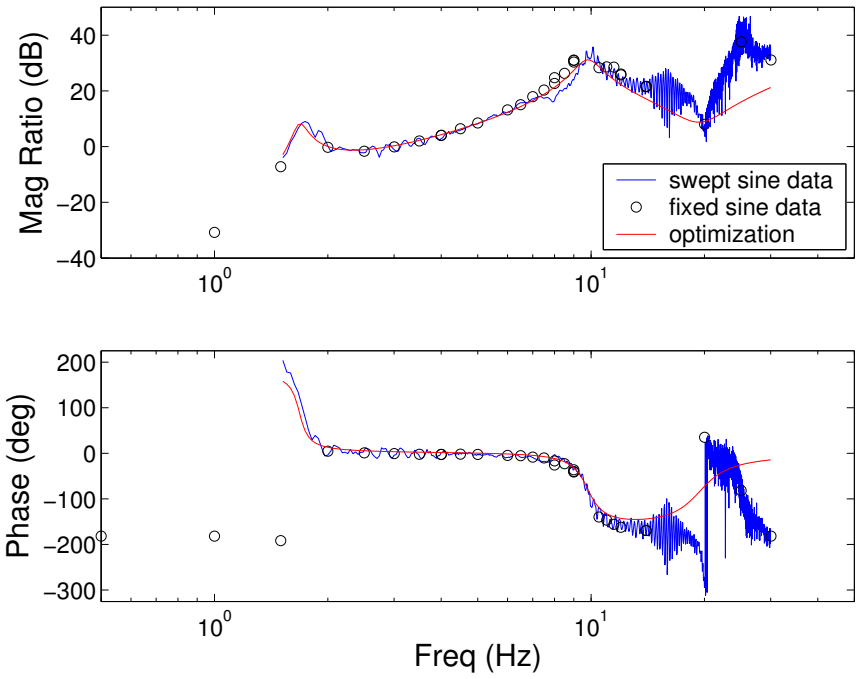

Figure 1.4: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration.

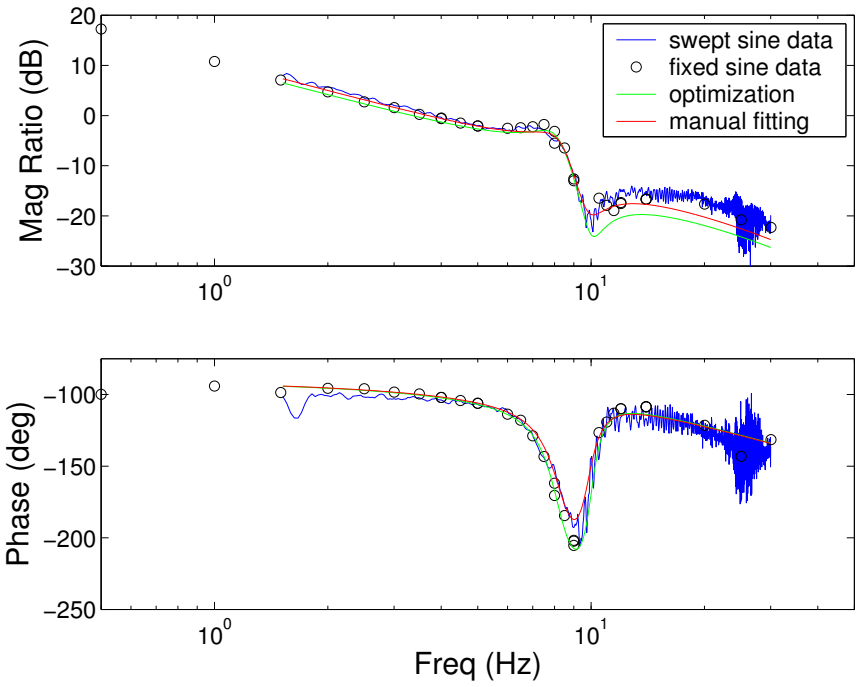

Figure 1.5: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position.

1.4 Overlaying Manual Curve Fits with Optimization Results

Figures 1.5 and 1.6 overlay the results from my manual curve fitting with results from my first optimization attempt.

Comparing the results of my manual curve fitting effort to the optimization result gives

$$\bar{x}_m = \begin{bmatrix} 20 \\ 51.5221 \\ 0.11 \\ 62.8319 \\ 0.06 \\ 188.496 \\ 10.9956 \\ 0.05 \\ -2.10526e-005 \\ 3.33333e-007 \end{bmatrix} \quad \text{and} \quad \bar{x}_o = \begin{bmatrix} 22.0616 \\ 52.3786 \\ 0.119069 \\ 61.5192 \\ 0.0767268 \\ 188.51 \\ 10.5585 \\ 0.0413066 \\ -1.71e-005 \\ 3.786e-007 \end{bmatrix} \tag{1.5}$$

where $\bar{x}_m$ refers to the manual curve fit and $\bar{x}_o$ refers to the optimization result.

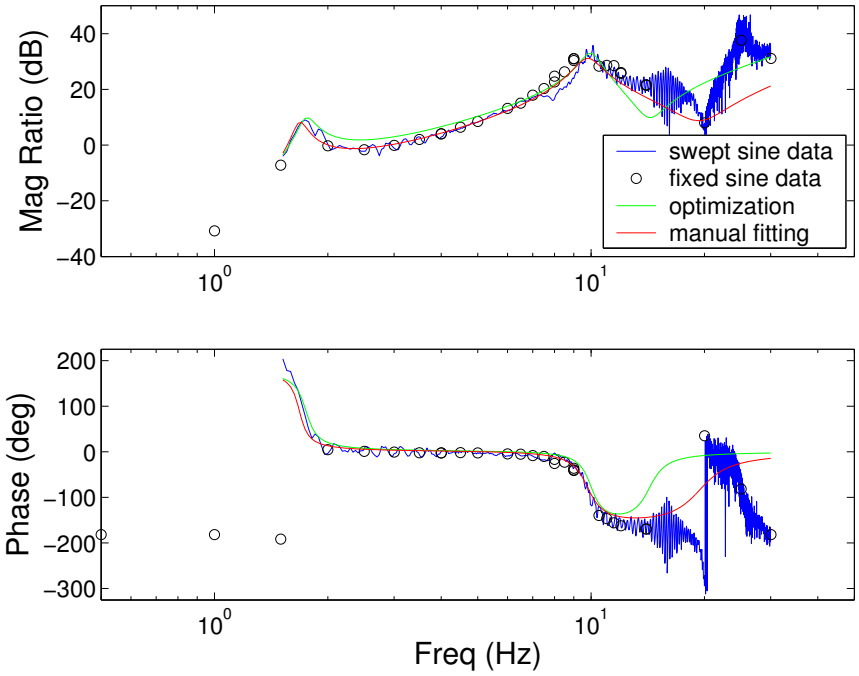

Figure 1.6: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration.

A.1 Matlab Files Used

A.1.1 Main Matlab File

The main Matlab file is `plottwobode.m` This file fills in a gap in my notebook for Fall 03 by formally documenting and comparing the results of my initial curve fitting at the beginning of the semester where I was tuning the parameters by hand and by trial and error to the results of my initial optimization work. The optimization was done over a frequency range of 1-12Hz and the error function is the sum of the squared magnitude error in dB for both the actuator and the flexible base. This optimization uses a sixth order model with an s^4 term in the numerator of the transfer function $\ddot{x}/\theta$. This fit does not consider phase error at all. This fit includes only the first two modes of the flexible base.

This file calls the file `load_data.m` and `genbodeplots_s4m2.m` which depends on the function `samiimodelss4m2.m`.

A.1.2 Additional Matlab Files

`genbodeplots_s4m2.m`

This file generates the bode plots for manual curve fitting results and/or my first optimization work (depending on the setting of `casenum`). This file is called by `plottwobode.m`.

`load_data.m`

This file loads experimental swept and fixed sine data. The swept sine data is loaded into global variables used by the curve fitting cost function. This file is called by `plottwobode.m`.

`samiimodels4m2.m`

This function is called by `genbodeplots_s4m2.m`. This function takes a vector of coeffiecients as an input and outputs the bode magnitudes and phases for both the actuator and flexible base models.

```
    coeffsin=varargin{1};

    varargout{1}=act_fit_mag;
varargout{2}=base_fit_mag;
varargout{3}=act_fit_ph;
varargout{4}=base_fit_ph;
```

A.1.3 Verbatim Matlab Files

`comparing_s4m2_fits_combined_editted_verb_apndx.pdf`

Chapter 2

Root Locus Analysis: s^4 Model for Flexible Base TF

Root locus analysis of the linearized system models that came out of my system identification efforts made up a significant chunk of the work done this semester. This analysis provided a way to simulate modifications to the system that would have taken significant time and effort to implement. This analysis shows that these modifications would not have solved the problems we were having. So, this analysis saved me a lot of time that would have been wasted. The results of this work have been accepted as a paper for the 2004 IEEE Aerospace Conference in Big Sky, Montana.

This analysis was all done twice after a necessary change in the model was discovered.

If the system is modeled as a 2DOF system with position as the output and force as the input and if force is proportional to angular acceleration ($\ddot{\theta}$), then a transfer function for the system would be

$$\frac{x}{\ddot{\theta}} = \frac{B_1\phi_1(L)}{s^2 + 2\zeta_1\omega_1 s + w_1^2} + \frac{B_2\phi_2(L)}{s^2 + 2\zeta_2\omega_2 s + \omega_2^2} \tag{2.1}$$

I initially fit this model to the input/output data for the system and it did not look quite right. The phase was off by 180° from what I expected. I said to myself that I was actually measuring acceleration rather than position so that the model should be

$$\frac{\ddot{x}}{\ddot{\theta}} = \frac{s^2B_1\phi_1(L)}{s^2 + 2\zeta_1\omega_1 s + w_1^2} + \frac{s^2B_2\phi_2(L)}{s^2 + 2\zeta_2\omega_2 s + \omega_2^2} \tag{2.2}$$

This model was able to fit the data fairly well and I proceeded to perform the root locus analysis included in Chapter 3. I refer to this model as the s^2 model throughout this work.

Later on it dawned on me that I have another s^2 term missing from my model. Not only am I measuring acceleration rather than position, but I am measuring angular position (θ) rather than angular acceleration ($\ddot{\theta}$). So the model I finally decided upon is

$$\frac{\ddot{x}}{\theta} = \frac{s^4B_1\phi_1(L)}{s^2 + 2\zeta_1\omega_1 s + w_1^2} + \frac{s^4B_2\phi_2(L)}{s^2 + 2\zeta_2\omega_2 s + \omega_2^2} \tag{2.3}$$

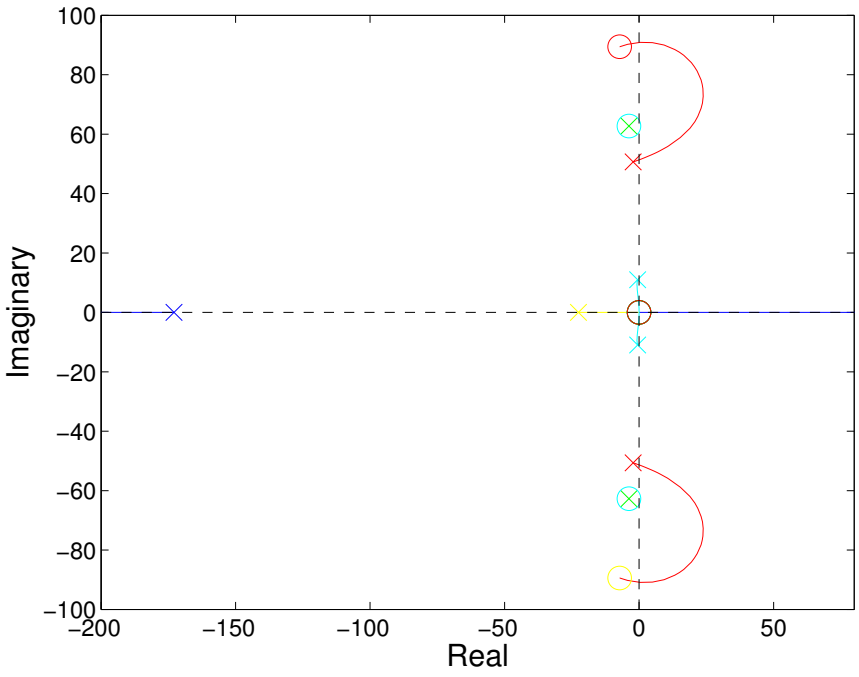

Figure 2.1: Root locus of the mass damping system without a lowpass filter or any other modifications.

(I refer to this as the s^4 model throughout this work). While this portion of the system is none causal, the transfer function for θ/v is relative degree 2 so that the overall system $\ddot{x}/v$ is relative degree zero.

The model change does not change the interpretation of the root locus results and the results from the old model are included in Chapter 3 for comparison.

Chapter 4 overlays Bode plots from each of the two models. The results are somewhat inconclusive. Each model can do a decent job of recreating the system dynamics over the frequency range considered (though the coefficients for the two models vary significantly). Bode data would need to be taken over a wider range to show which model correctly predicts the high frequency roll off (or lack there of) for the system. This higher frequency data is difficult to obtain because of the band width of the hydraulic actuator.

Figure 2.1 shows a root locus for the mass damping control system without a low pass filter or any modifications. This system is unstable for most values of K_a. There is no value for K_a for which this system has better vibration response than the system without acceleration feedback.

One interesting thing to note about this system is that it is relative degree two, however as $K_a \rightarrow \infty$, two of the poles are approaching $\pm\infty$ along the real axis. This is the result of the coefficient B_2 in the flexible base transefer function (equation 2.4) being negative. This makes all of the numerator coefficients in the tranfer function $\ddot{x}/d$ negative, resulting in a root locus with an angle criterion of 0° rather than 180°.

Figure 2.2 zooms in on the portion of the locus near the origin.

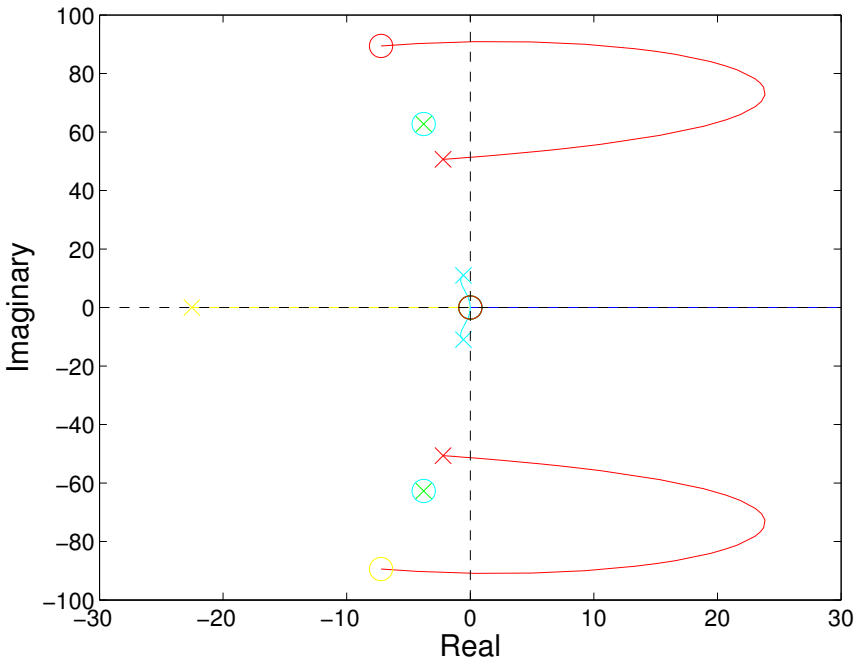

Figure 2.2: Root locus of the mass damping system without a lowpass filter or any other modifications. (Zooming in on Figure 2.1.)

$$\frac{x}{\ddot{\theta}} = \frac{B_1\phi_1(L)}{s^2 + 2\zeta_1\omega_1 s + w_1^2} + \frac{B_2\phi_2(L)}{s^2 + 2\zeta_2\omega_2 s + \omega_2^2} \tag{2.4}$$

With B_2 negative, it seemed necessary to investigate the system with the gain K_a negative as well. The root locus for the unmodified system with K_a negative is shown in Figure 2.3. This system is also unstable for most values of K_a and there is not value for K_a for which this system has a better vibration response than the open loop system.

With the system not exhibiting better performance than the system without acceleration feedback for any values (positive or negative) of K_a, it seemed necessary to investigate the potential improvement in system performance that could be attained if either of the two deviations from the theoretical mass damping system could be fixed.

Figure 2.4 zooms in on the portion of the locus near the origin.

Figures 2.5 and 2.7 show the changes to the root locus resulting from eliminating the phase difference between the pure integrator model and the experimentally determined actuator transfer function near the second natural frequency of the base. This root locus simulates the system response if the actuator was actually a pure integrator. This eliminates one second order zero and one zero order pole from the root locus. At first glance it would appear that this has simply eliminated the pole/zero pairs at $-3.77 \pm 62.7j$. Actually, this has eliminated the zeros at $-3.77 \pm 62.7j$ and the poles at $-2.21 \pm 50.7j$. The poles at $-3.77 \pm 62.7j$ then follow trajectories similar to those that the poles at $-2.21 \pm 50.7j$ used to follow in the system shown in Figures 2.1 and

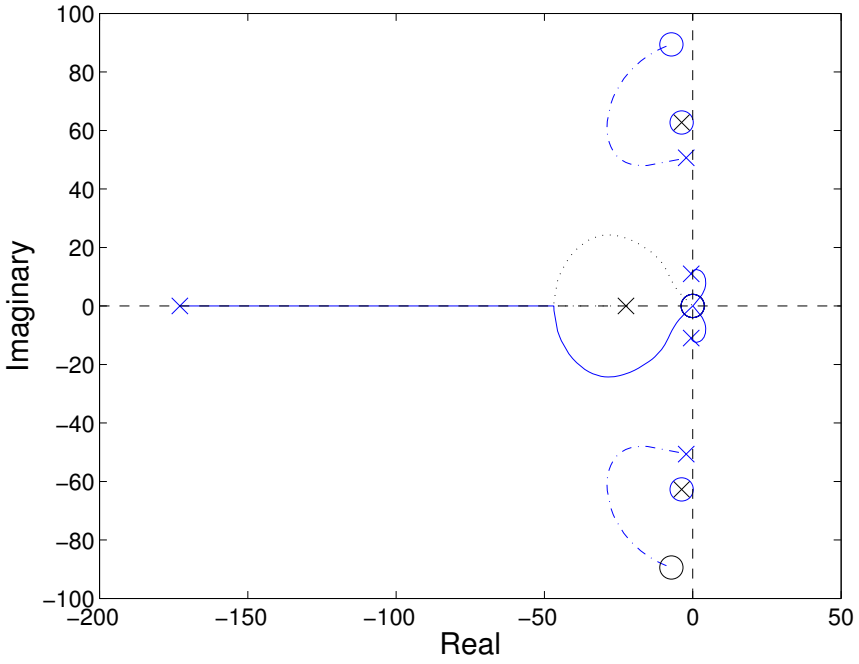

Figure 2.3: Root locus of the mass damping system without a lowpass filter or any other modifications. The gain K_a is negative.

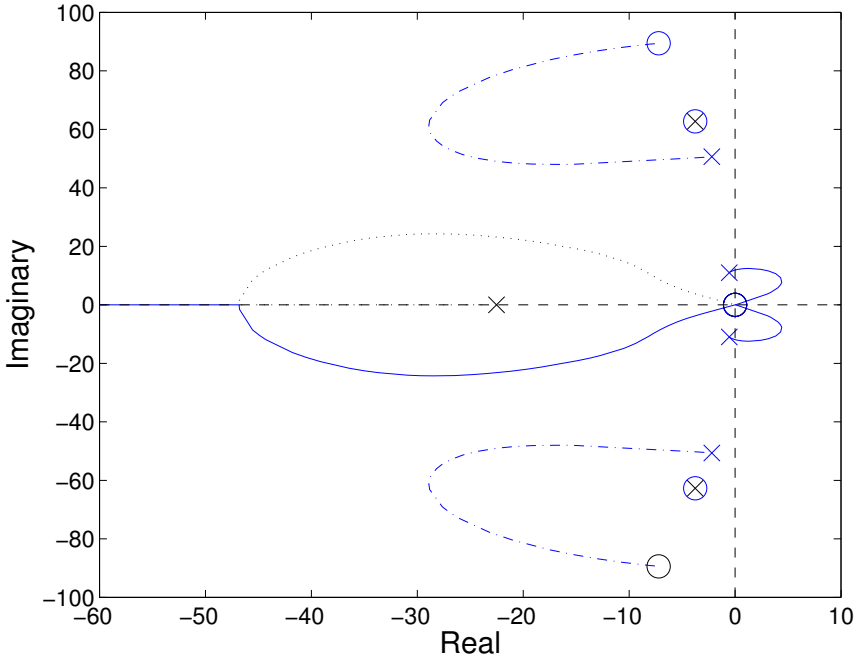

Figure 2.4: Root locus of the mass damping system without a lowpass filter or any other modifications. The gain K_a is negative. (Zooming in on Figure 2.3.)

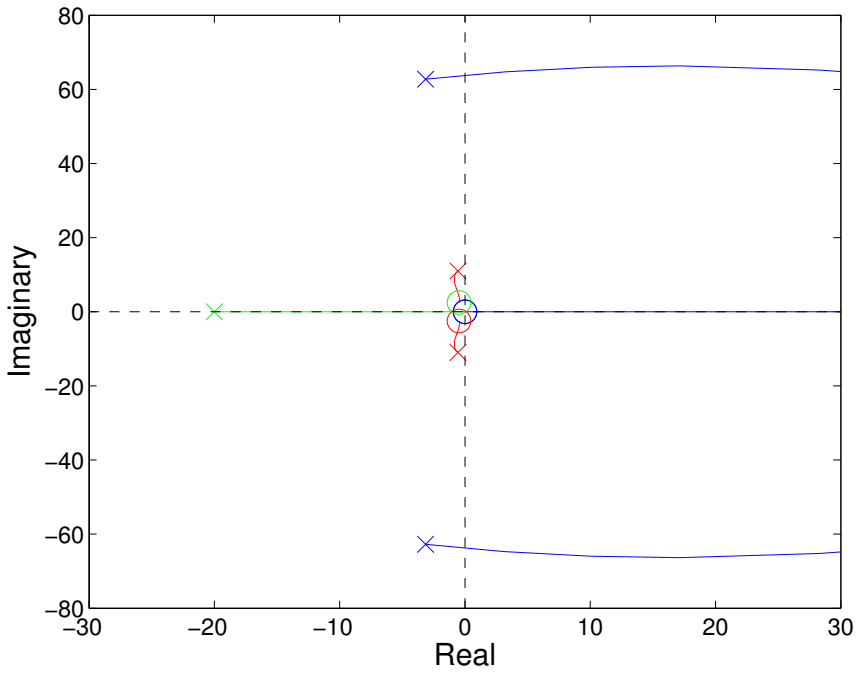

Figure 2.5: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed).

2.3.

The similarities between Figures 2.1 and 2.5 and between Figures 2.3 and 2.7 lead to the somewhat counterintuitive conclusion that the 90° difference between expected and actual phase near the second natural frequency of the base does not have a significant impact on the stability of the system. The result seems counterintuitive from a vibrations stand point: if the system is supposed to generate interaction forces the simulate damping, a 90° phase error should be significant.

Figures 2.9 and 2.11 show the effects on the root locus of switching from accelerometers to sensors that could somehow sense the position of the base. If suitable sensors could be found, this would eliminate another difference between the actual system and the theoretical ideal that this system was designed around. While this difference again seems intuitively significant, the effects of installing base position sensors on the root locus are not obviously beneficial. This change eliminates a pair of zeros at the origin and changes the system from relative degree one to relative degree three. Figure 2.9 shows the root locus with gain K_a positive and Figure 2.11 shows the root locus with K_a negative. In both cases, the system is unstable for most values of K_a and no value of K_a can be found for which the vibration response of the system is significantly improved over the system without vibration feedback.

Figures 2.13 and 2.15 show the root locus for the system with both of the previously mentioned deviations from the ideal corrected (the actuator is a pure integrator and the system has

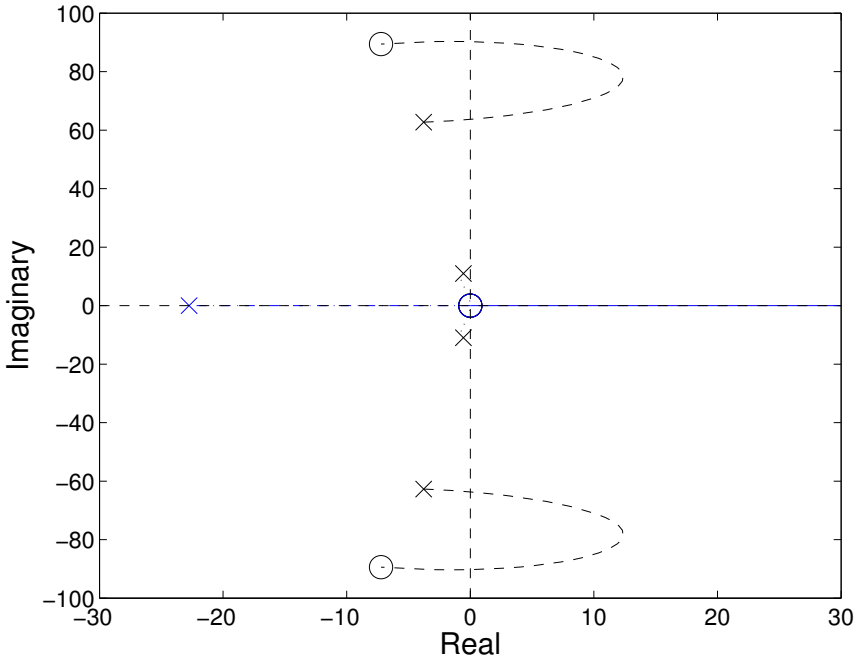

Figure 2.6: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed). (Zooming in on Figure 2.5.)

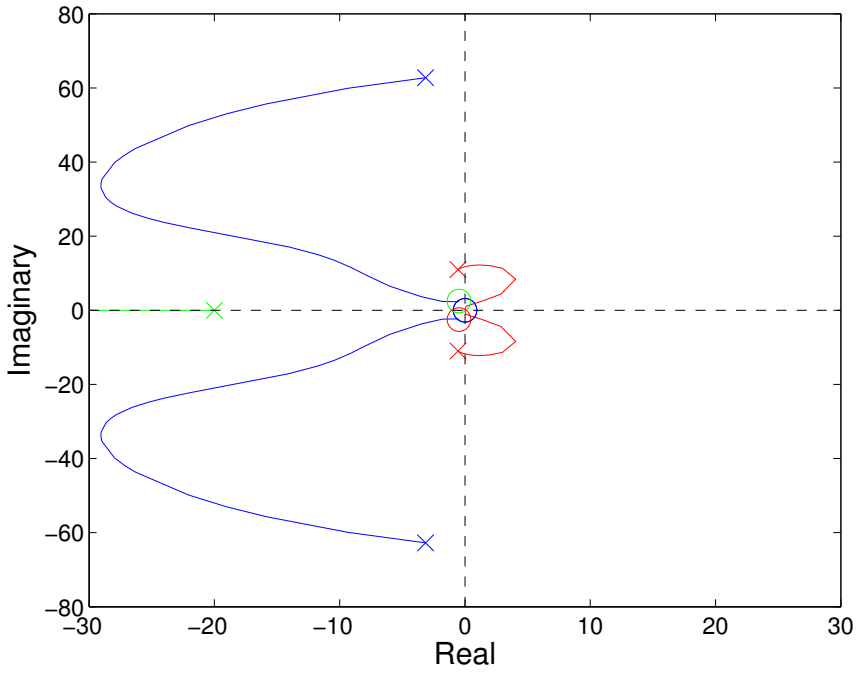

Figure 2.7: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed). The gain K_a is negative.

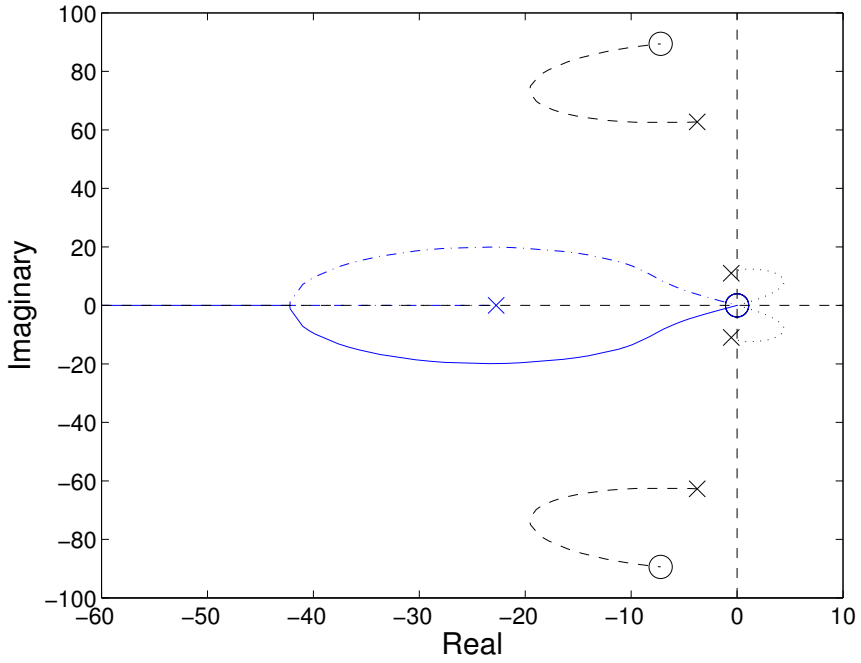

Figure 2.8: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed). The gain K_a is negative. (Zooming in on Figure 2.7.)

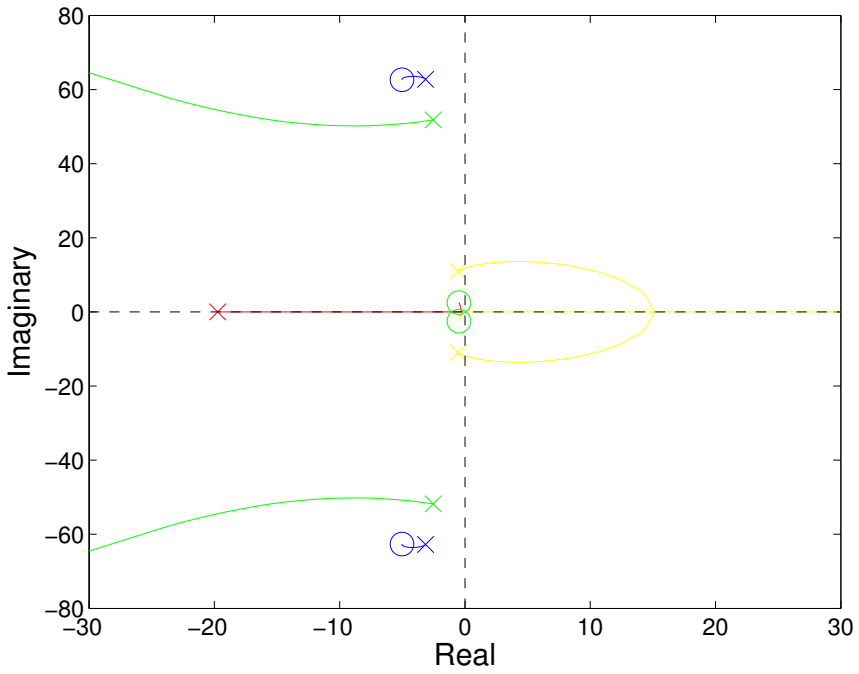

Figure 2.9: Root locus of the mass damping system with a base position sensor but without a lowpass filter.

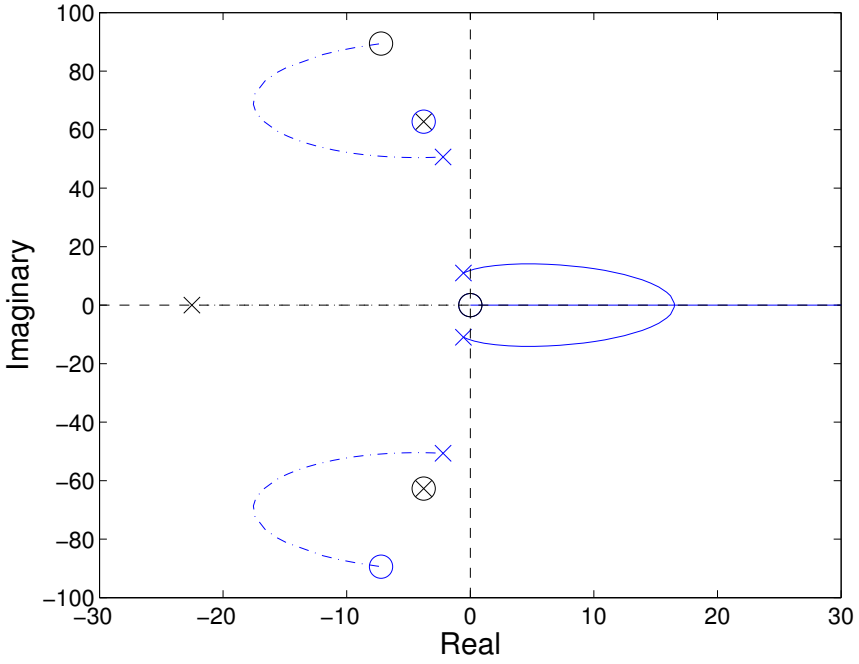

Figure 2.10: Root locus of the mass damping system with a base position sensor but without a lowpass filter. (Zooming in on Figure 2.9.)

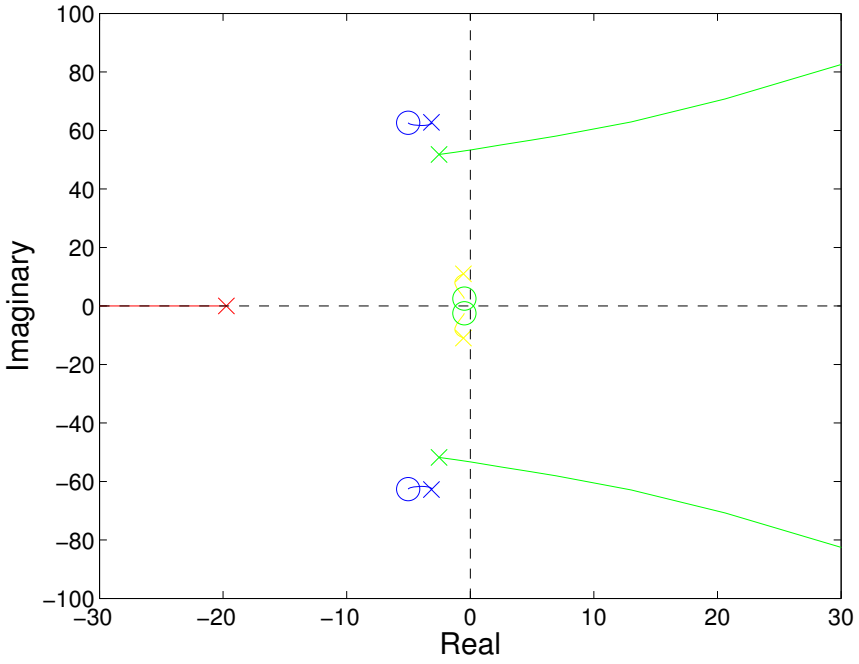

Figure 2.11: Root locus of the mass damping system with a base position sensor but without a lowpass filter. The gain K_a is negative.

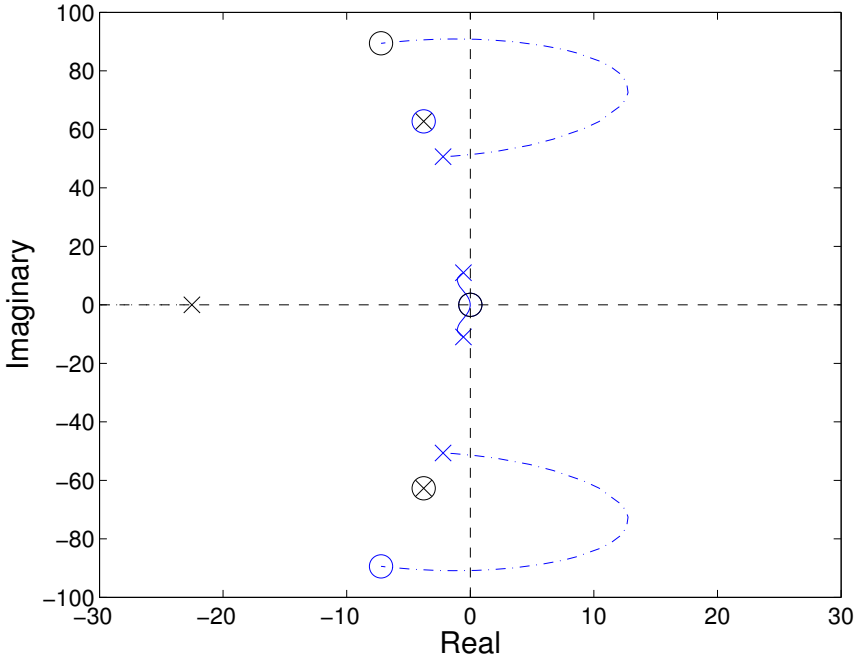

Figure 2.12: Root locus of the mass damping system with a base position sensor but without a lowpass filter. The gain K_a is negative. (Zooming in on Figure 2.11.)

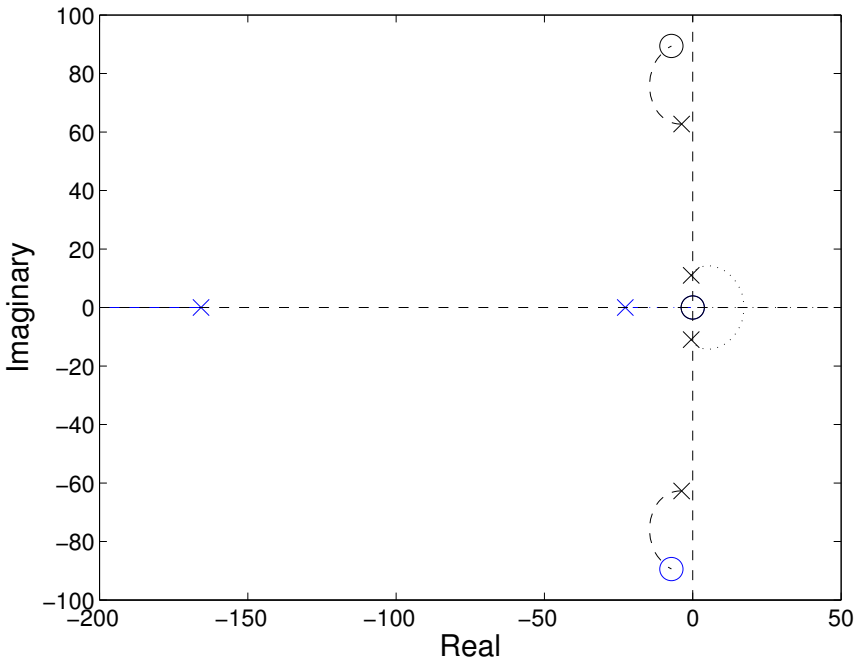

Figure 2.13: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter.

base position sensors). The system is still unstable for most values of K_a and the performance is still not improved over the system with no active vibration suppression.

Figure 2.17 shows the root locus of the system with the current design for a low-pass filter on the accelerometer signal (a 2^{nd} order Butterworth filter with a cutoff frequency of 2Hz).

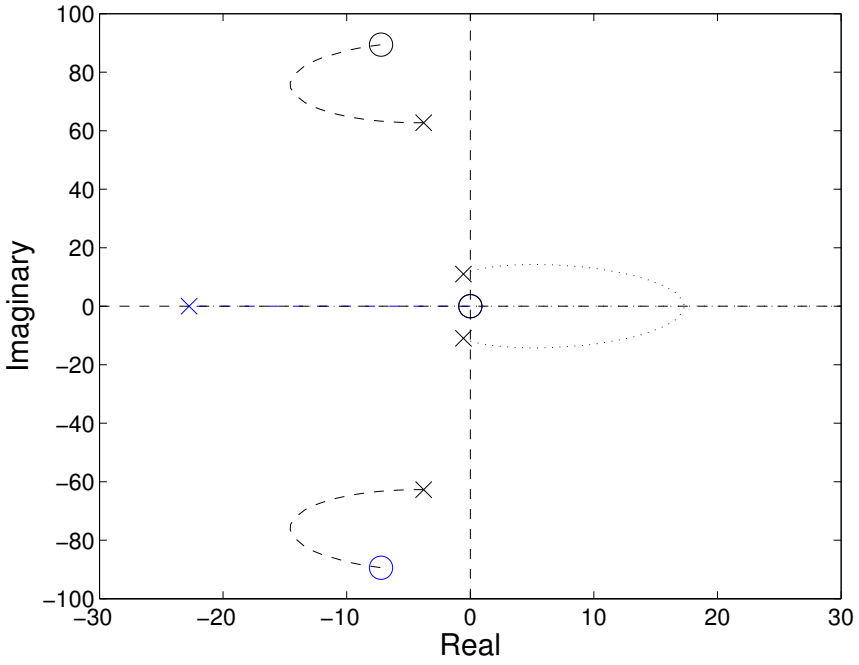

Figure 2.14: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter. (Zooming in on Figure 2.13.)

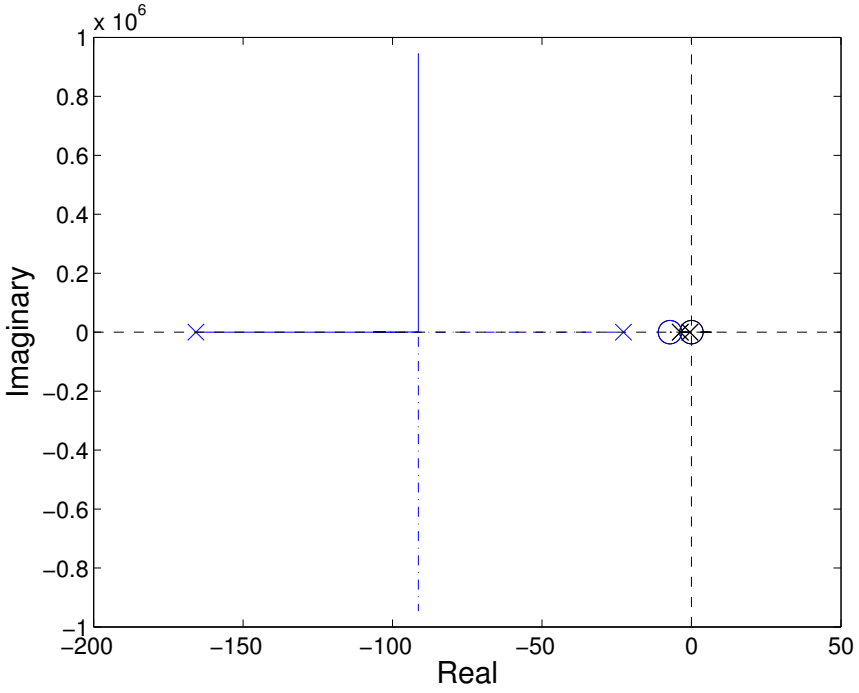

Figure 2.15: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter. The gain K_a is negative.

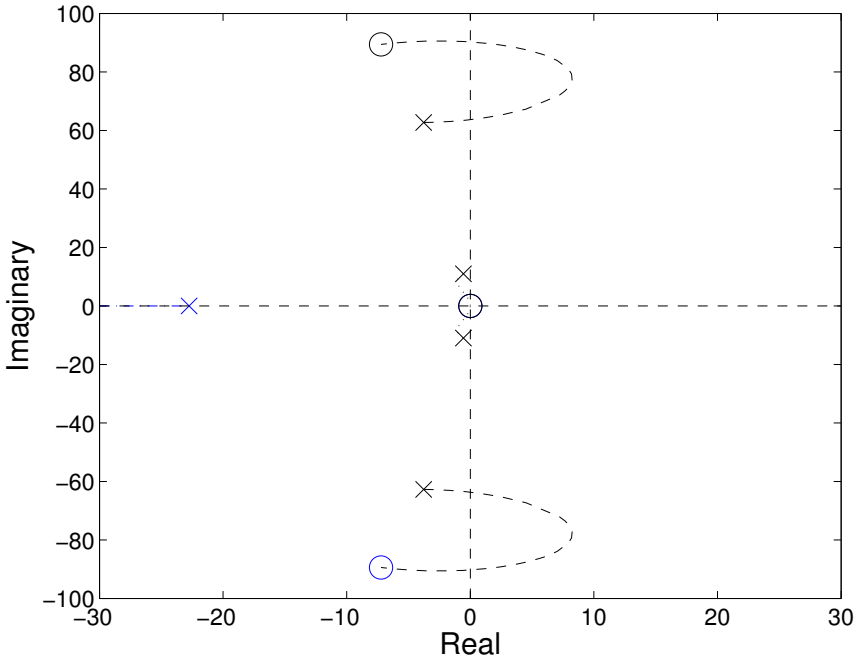

Figure 2.16: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter. The gain K_a is negative. (Zooming in on Figure 2.15.)

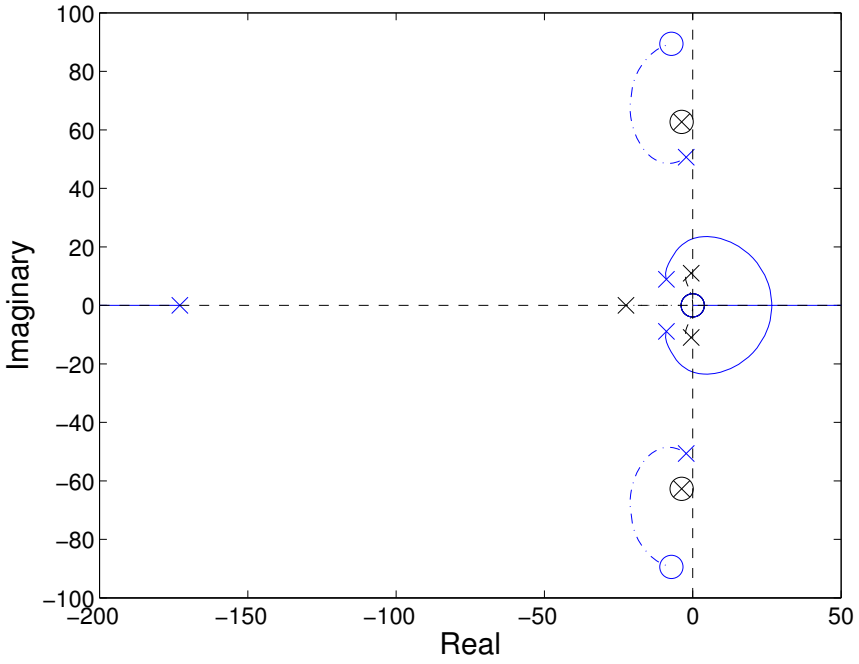

Figure 2.17: Root locus of the base system with a lowpass filter.

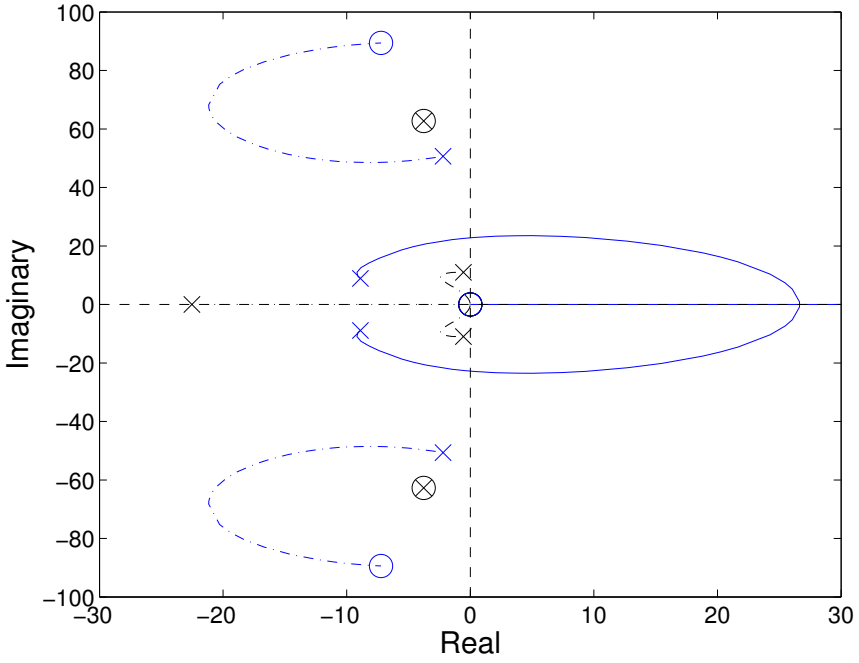

Figure 2.18: Root locus of the base system with a lowpass filter. (Zooming in on Figure 2.17.)

B.1 Matlab File Used

B.1.1 Matlab File Summary

This file uses the Matlab file `rlocus_full_story_s_4_callfunc.m` This file generates bode plots for many of the system modifications considered (accelerometers vs. base position sensors, removing unmodeled actuator dynamics, etc).

I believe this file was eventually replaced by `call_rlocus_pseudos_s_4.m`, which allows for easier comparing of one system against another.

The transfer function for $\ddot{x}/\theta$ used in this file has an s^4 term in its numerator.

B.1.2 Verbatim Matlab Text

`rlocus_full_story_callfunc_editted_verb_apndx.pdf`

Chapter 3

Root Locus Analysis: s^2 Model for Flexible Base TF

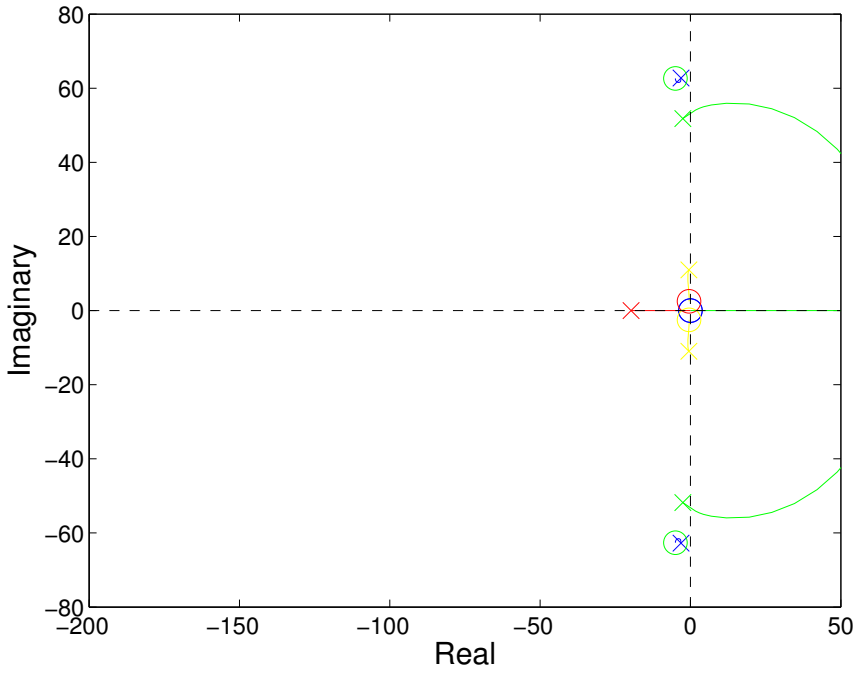

Figure 3.1: Root locus of the mass damping system without a lowpass filter or any other modifications.

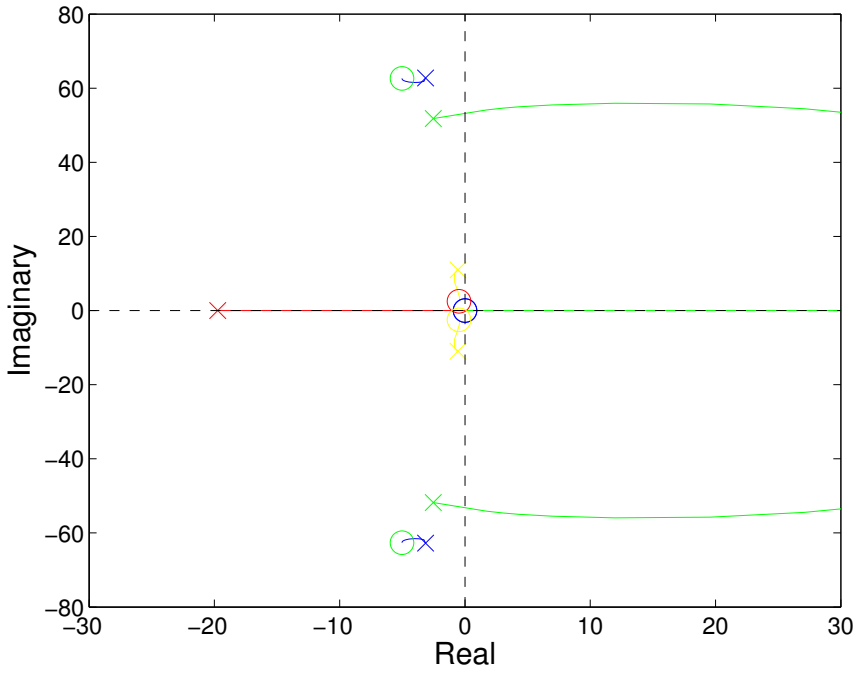

Figure 3.2: Root locus of the mass damping system without a lowpass filter or any other modifications. (Zooming in on Figure 3.1.)

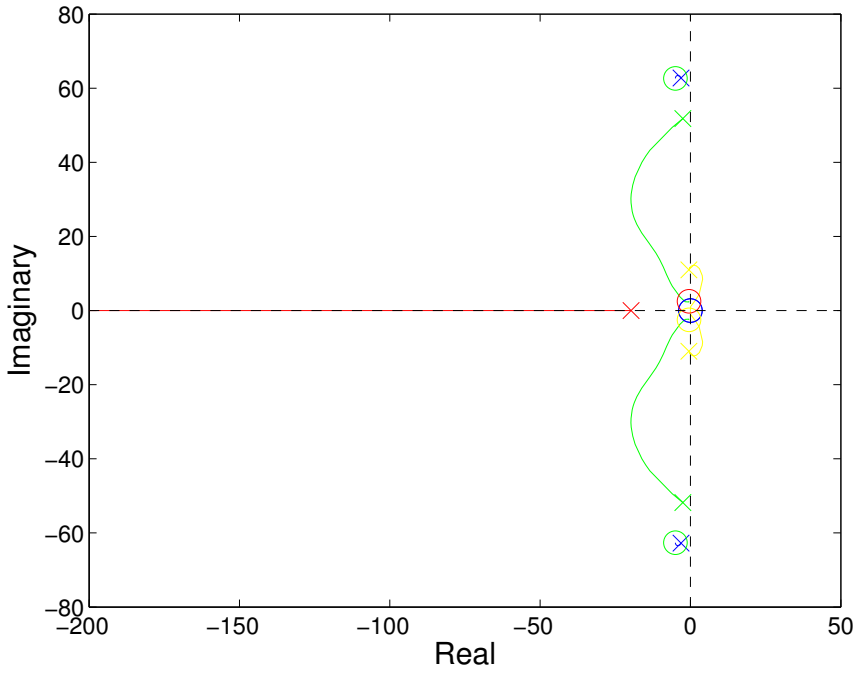

Figure 3.3: Root locus of the mass damping system without a lowpass filter or any other modifications. The gain K_a is negative.

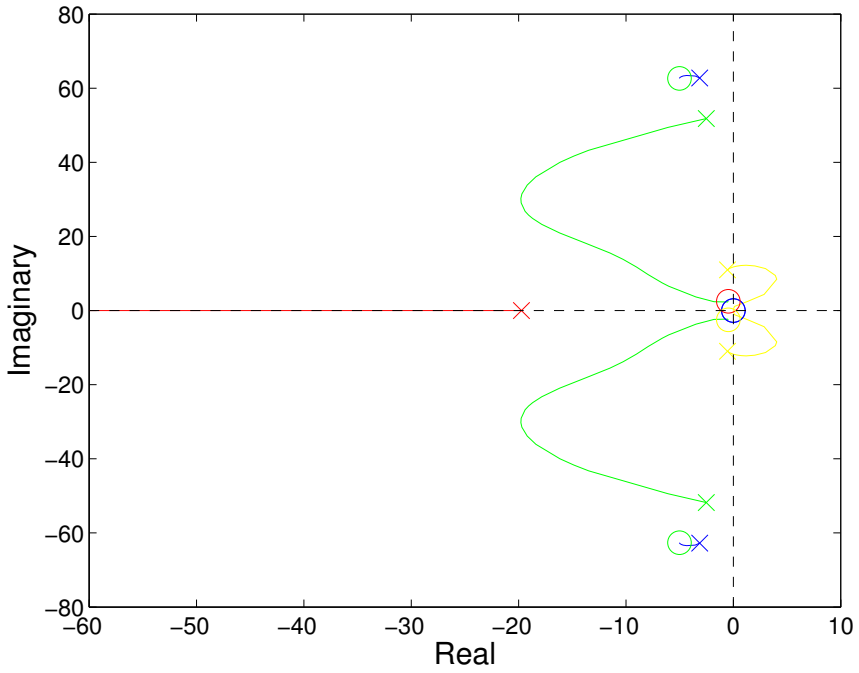

Figure 3.4: Root locus of the mass damping system without a lowpass filter or any other modifications. The gain K_a is negative. (Zooming in on Figure 3.3.)

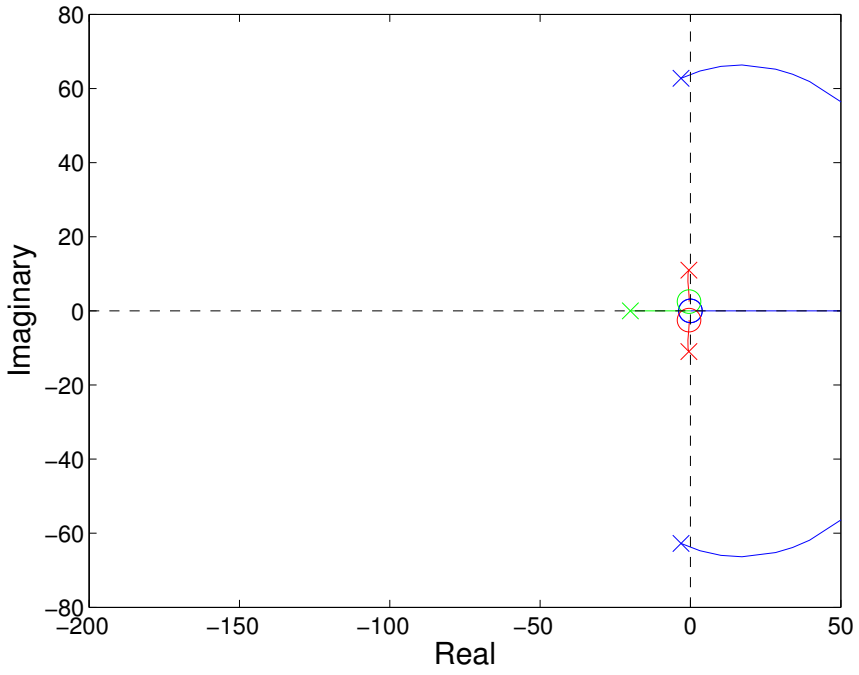

Figure 3.5: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed).

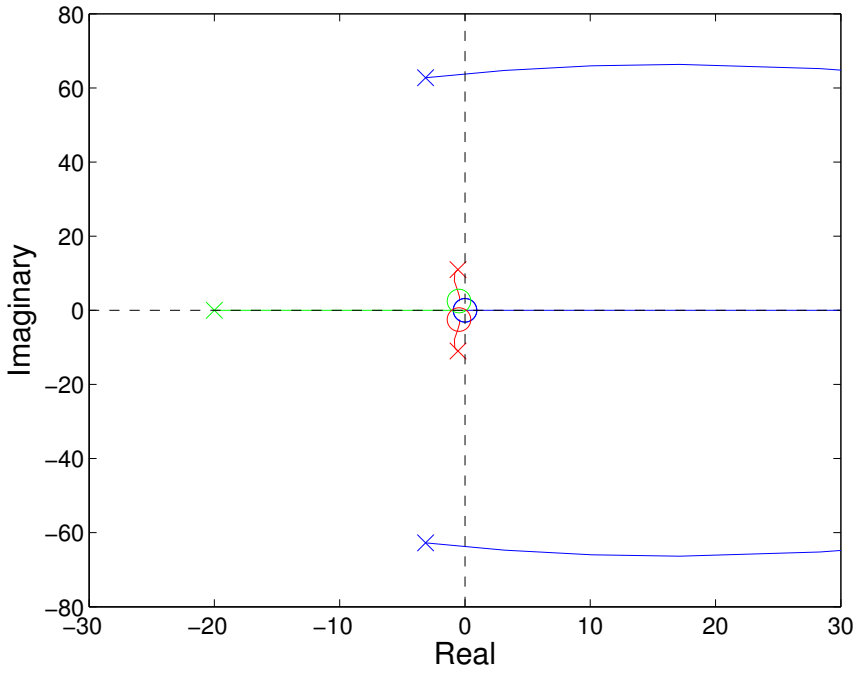

Figure 3.6: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed). (Zooming in on Figure 3.5.)

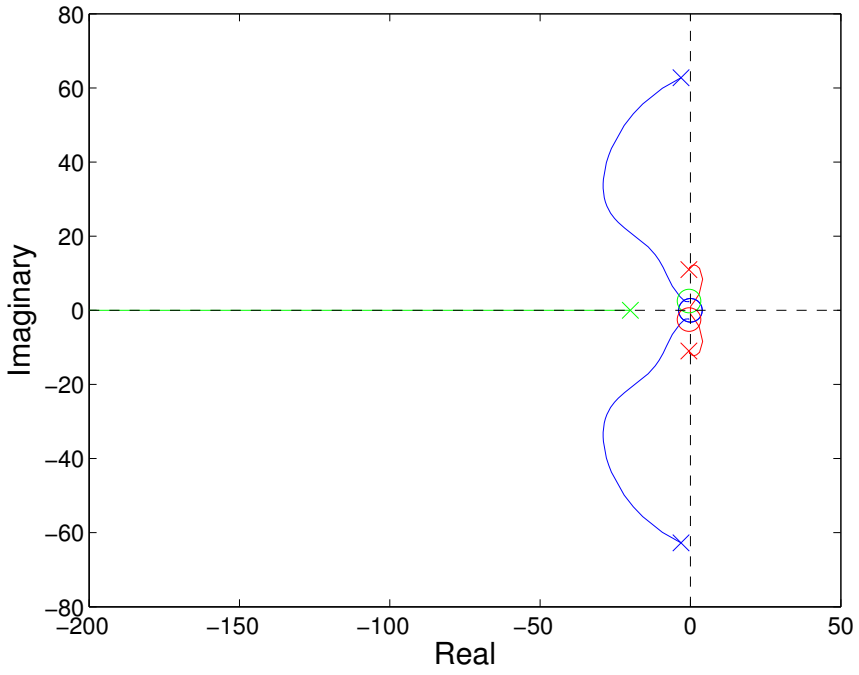

Figure 3.7: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed). The gain K_a is negative.

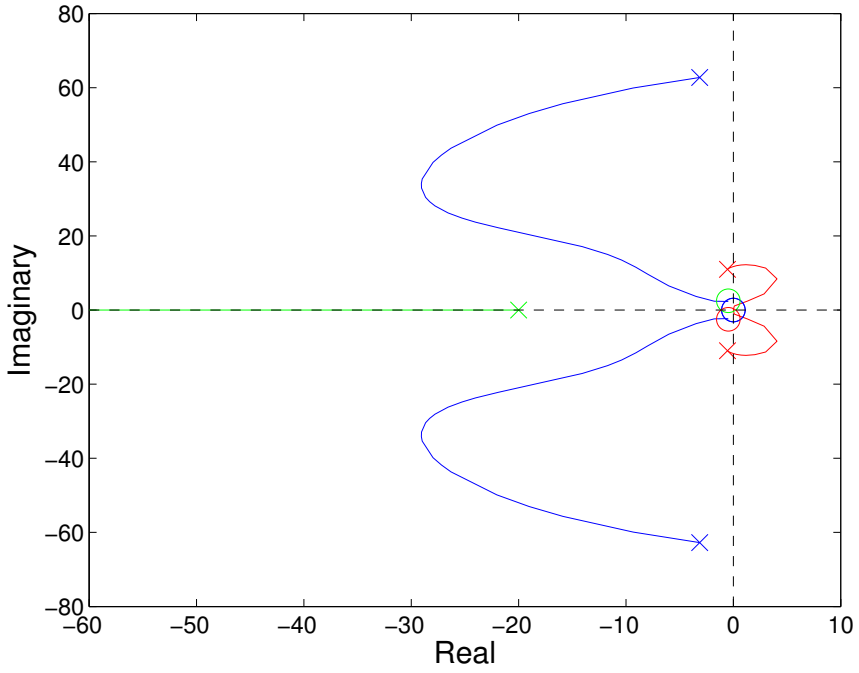

Figure 3.8: Root locus of the mass damping system with an actuator that is a pure integrator and a first order lag but without a lowpass filter (i.e. the second order pole and zero in the actuator tranfer function have some how been removed). The gain K_a is negative. (Zooming in on Figure 3.7.)

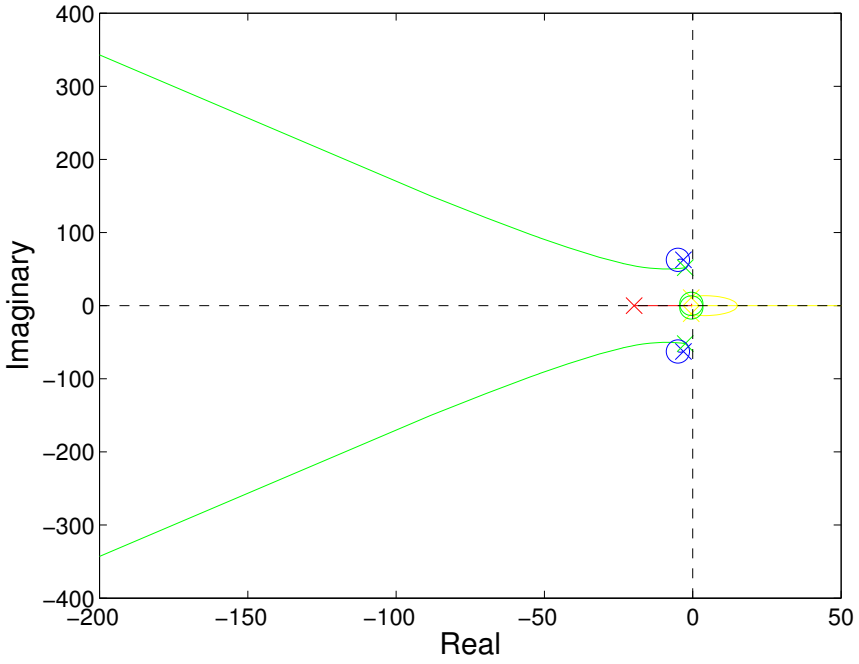

Figure 3.9: Root locus of the mass damping system with a base position sensor but without a lowpass filter.

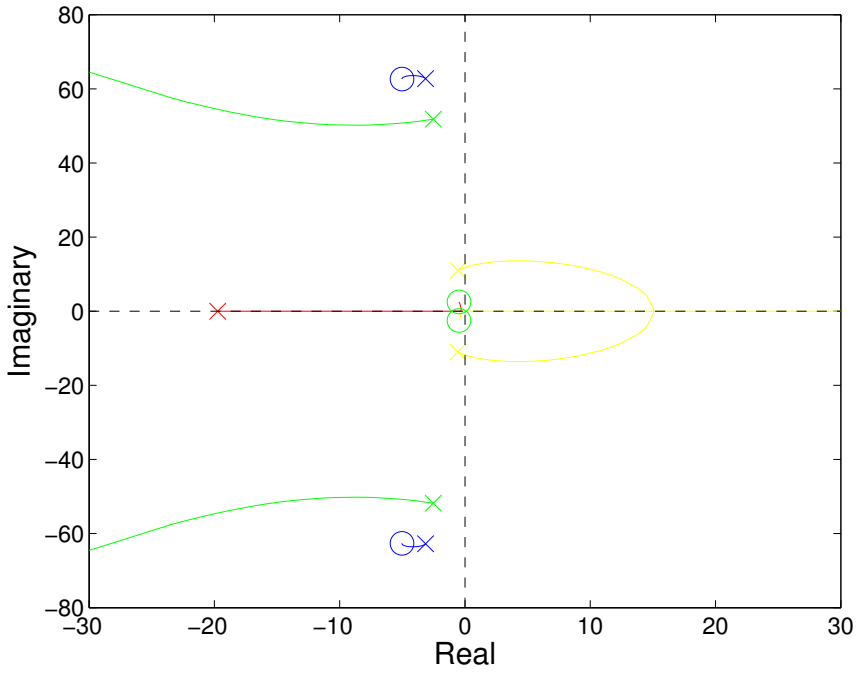

Figure 3.10: Root locus of the mass damping system with a base position sensor but without a lowpass filter. (Zooming in on Figure 3.9.)

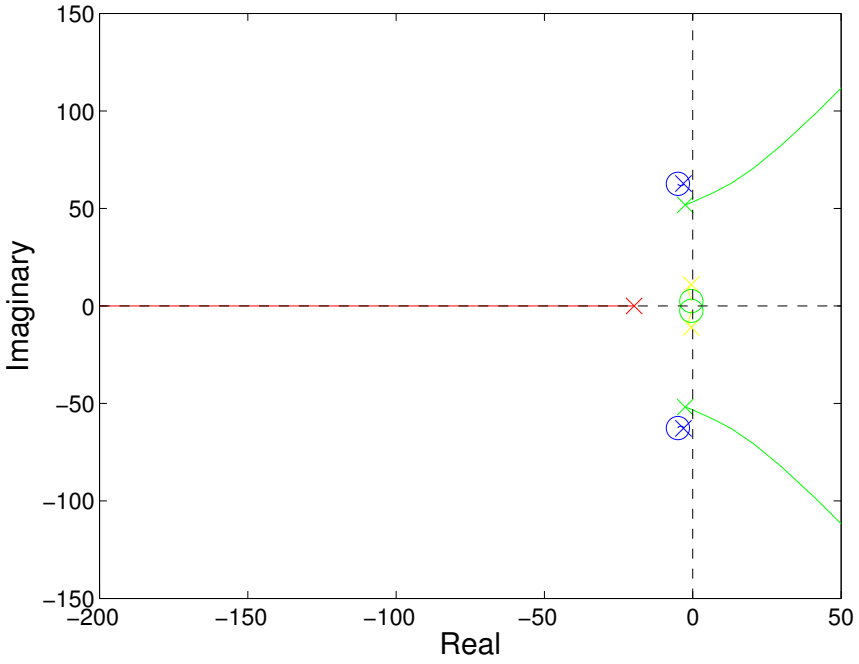

Figure 3.11: Root locus of the mass damping system with a base position sensor but without a lowpass filter. The gain K_a is negative.

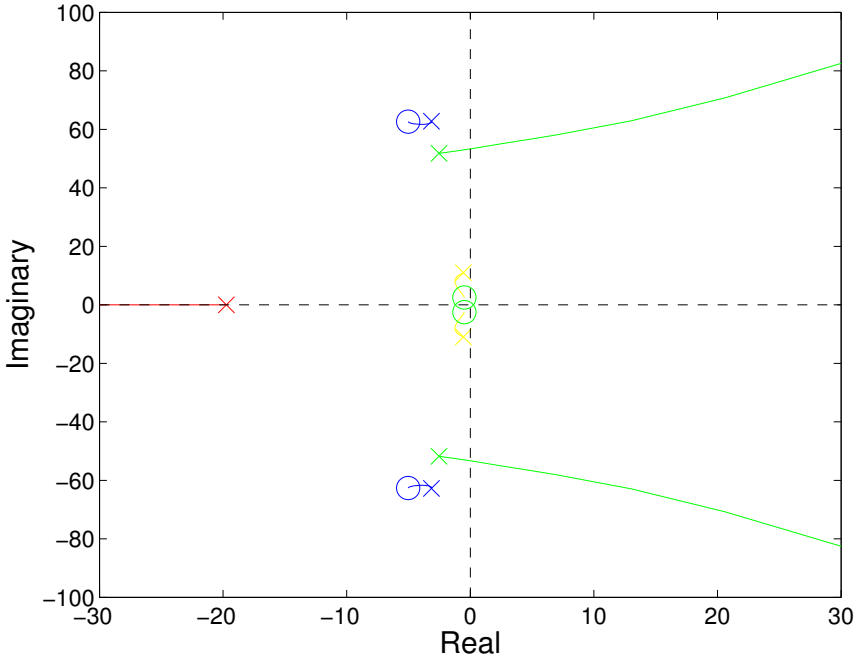

Figure 3.12: Root locus of the mass damping system with a base position sensor but without a lowpass filter. The gain K_a is negative. (Zooming in on Figure 3.11.)

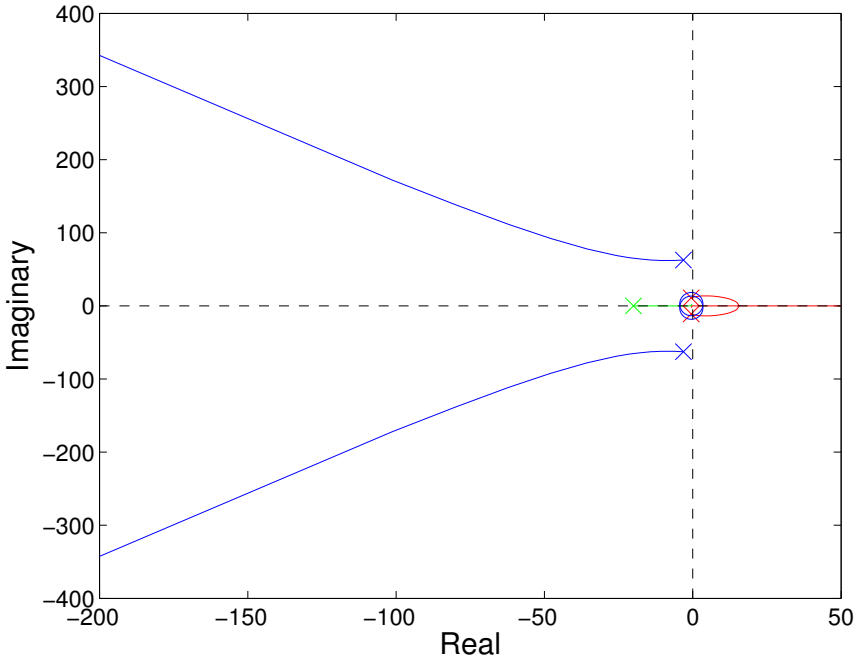

Figure 3.13: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter.

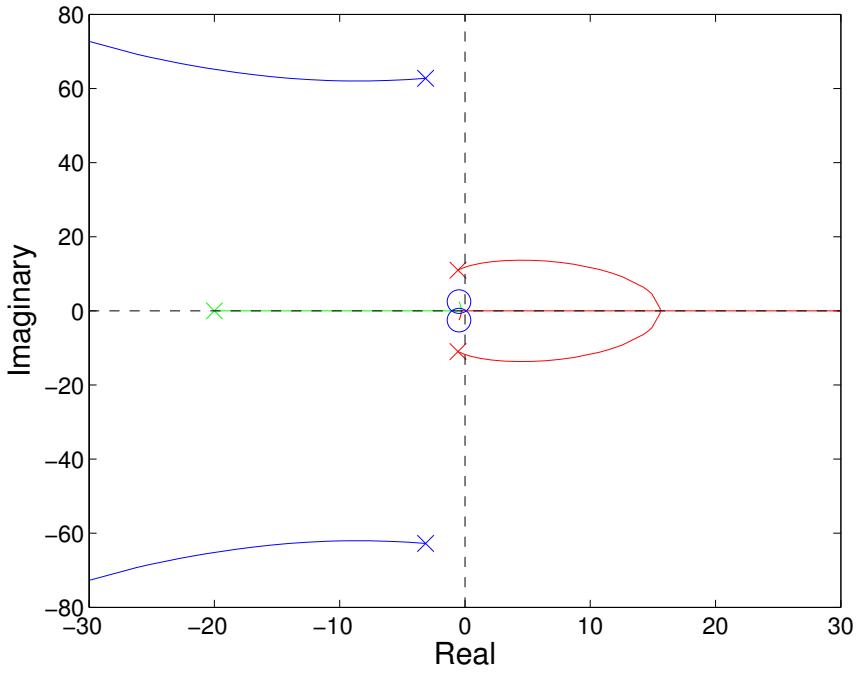

Figure 3.14: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter. (Zooming in on Figure 3.13.)

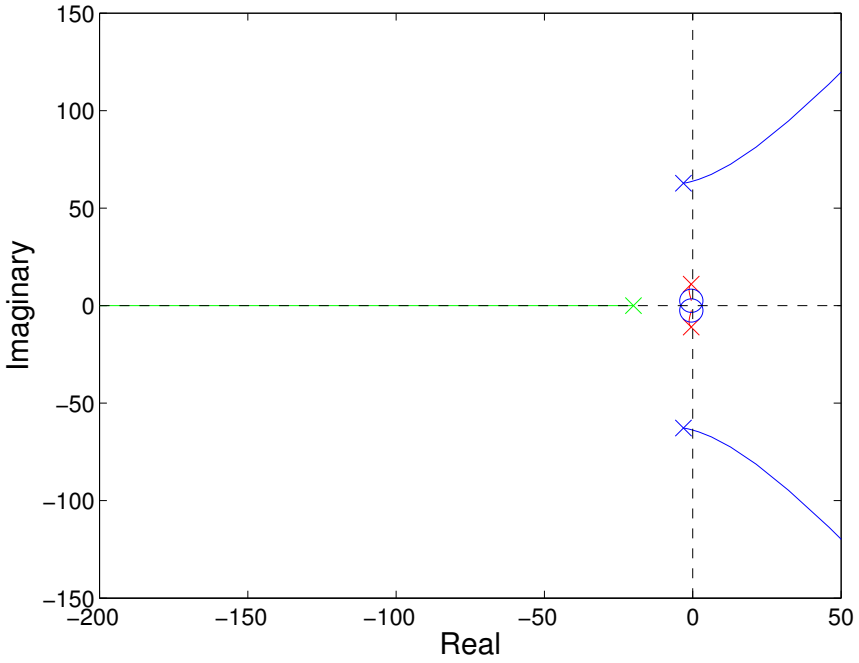

Figure 3.15: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter. The gain K_a is negative.

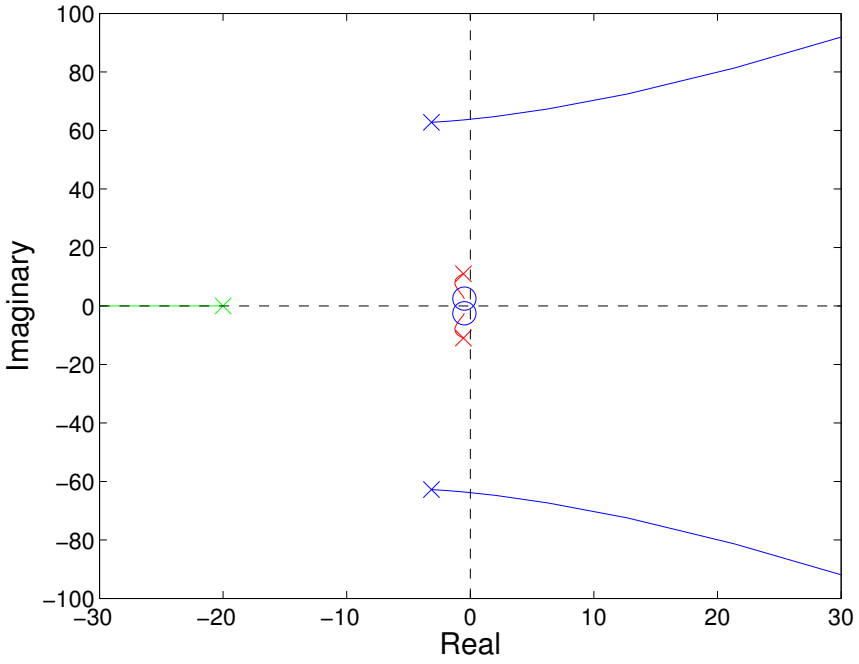

Figure 3.16: Root locus of the mass damping system with a base position sensor and second order dynamics removed from actuator but without a lowpass filter. The gain K_a is negative. (Zooming in on Figure 3.15.)

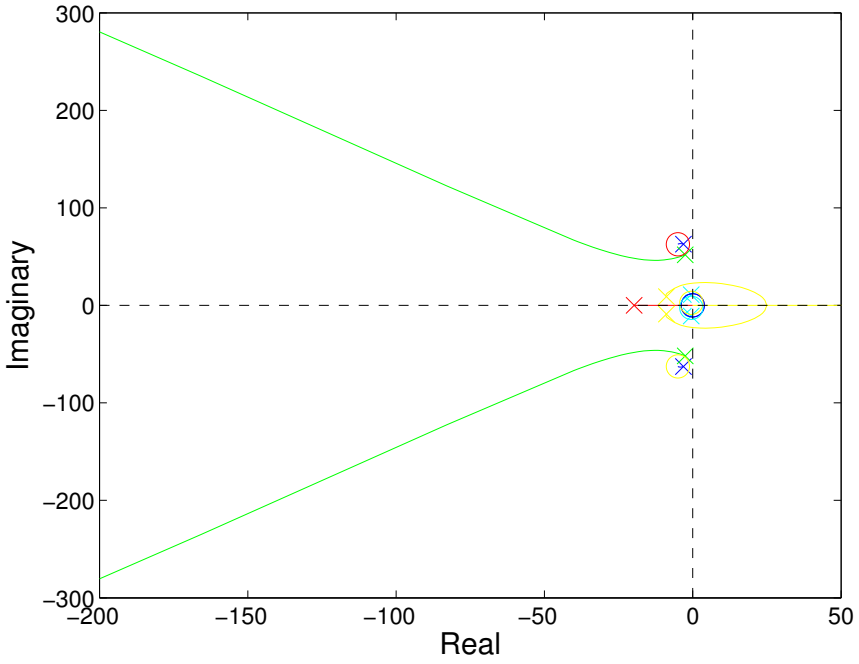

Figure 3.17: Root locus of the base system with a lowpass filter.

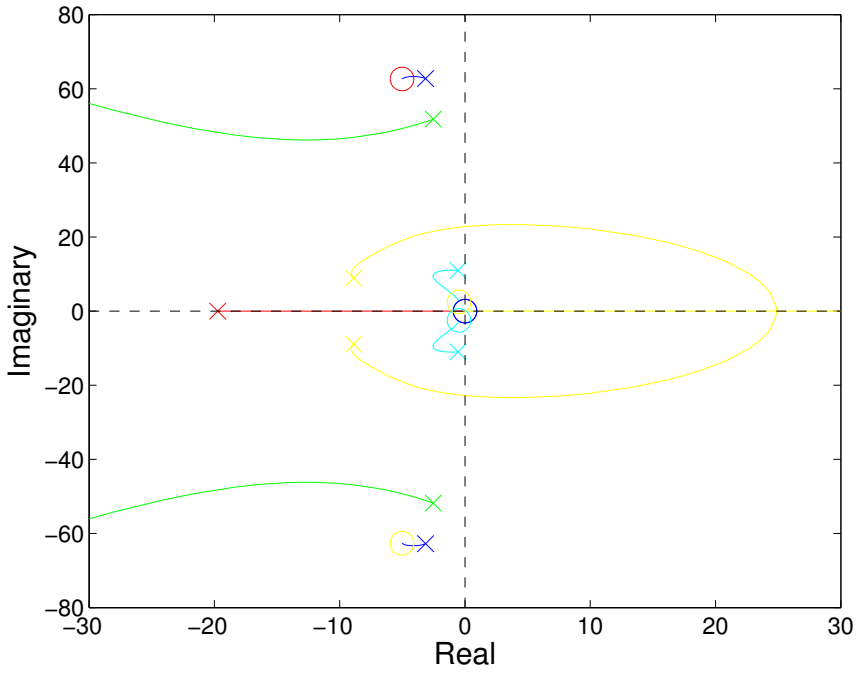

Figure 3.18: Root locus of the base system with a lowpass filter. (Zooming in on Figure 3.17.)

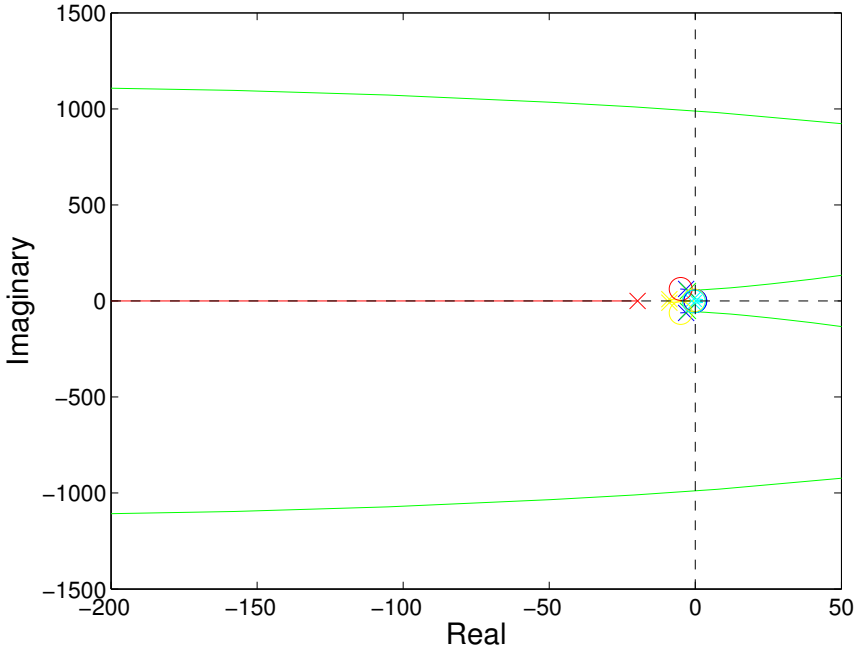

Figure 3.19: Root locus of the base system with a lowpass filter. The gain K_a is negative.

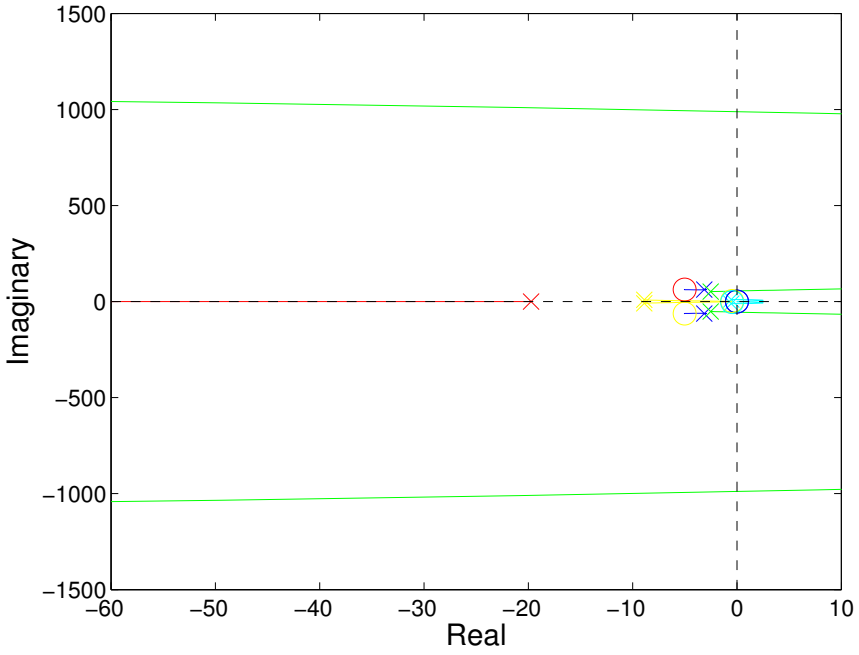

Figure 3.20: Root locus of the base system with a lowpass filter. The gain K_a is negative. (Zooming in on Figure 3.19.)

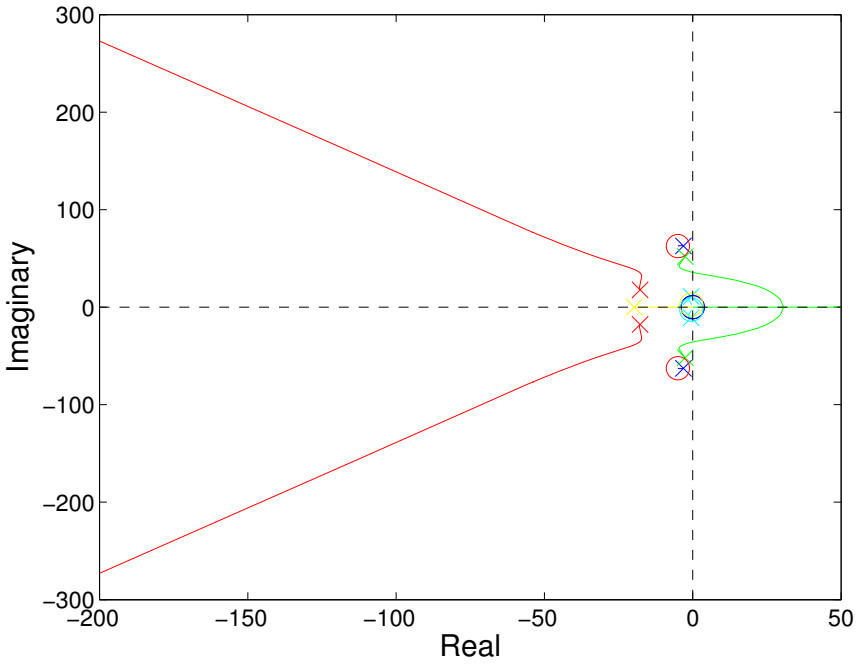

Figure 3.21: Root locus of the base system with a lowpass filter with a corner frequency of4Hz.

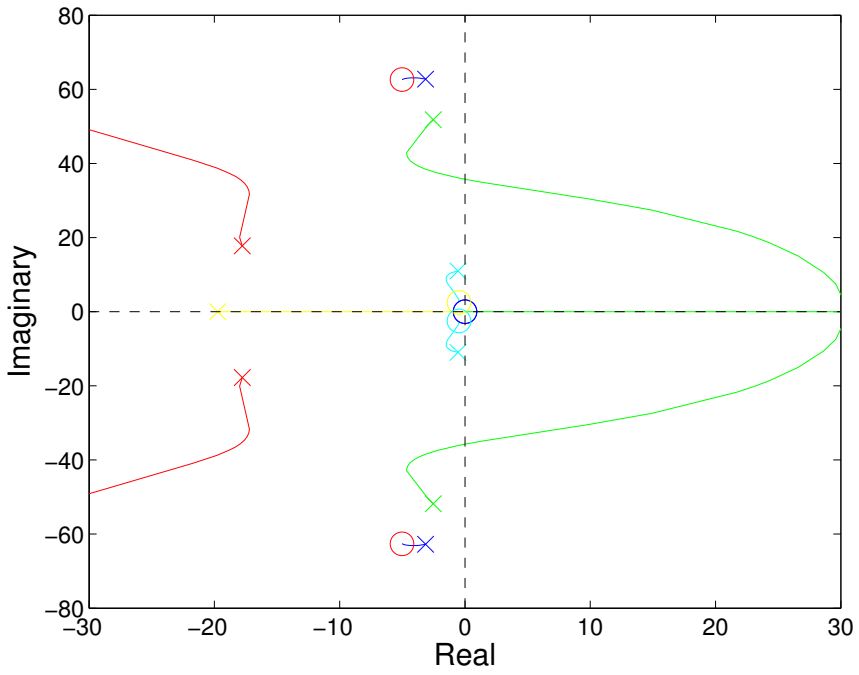

Figure 3.22: Root locus of the base system with a lowpass filter with a corner frequency of4Hz. (Zooming in on Figure 3.21.)

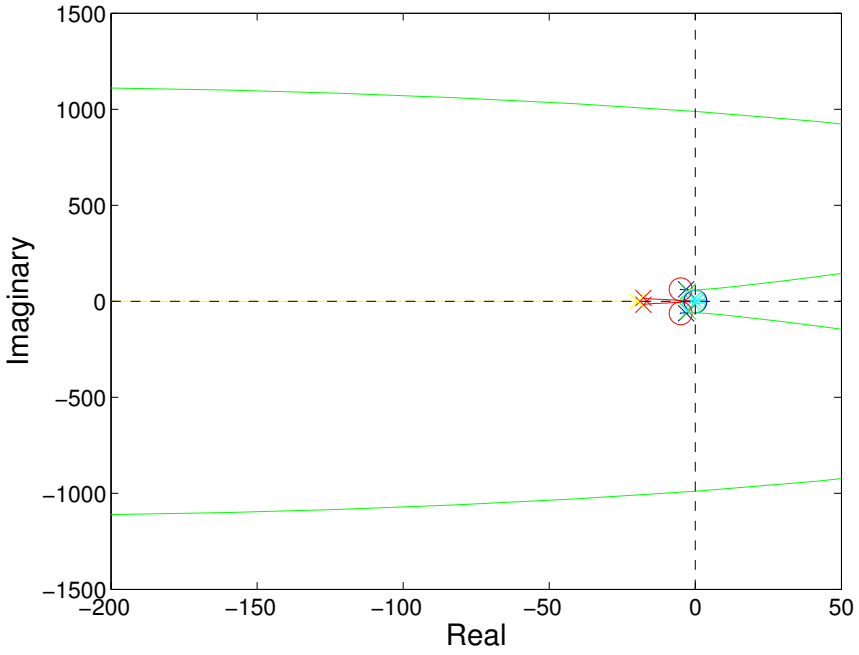

Figure 3.23: Root locus of the base system with a lowpass filter with a corner frequency of4Hz. The gain K_a is negative.

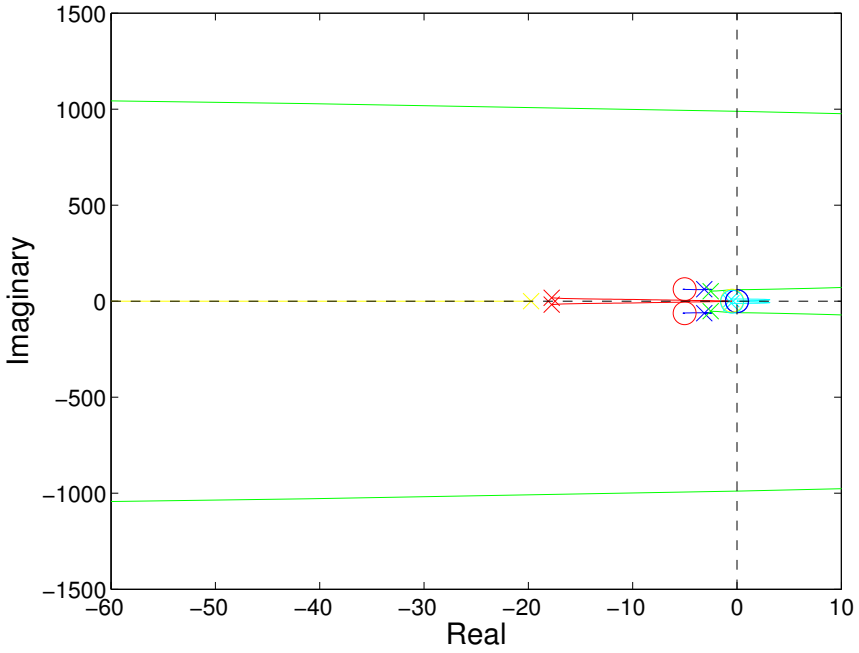

Figure 3.24: Root locus of the base system with a lowpass filter with a corner frequency of4Hz. The gain K_a is negative. (Zooming in on Figure 3.23.)

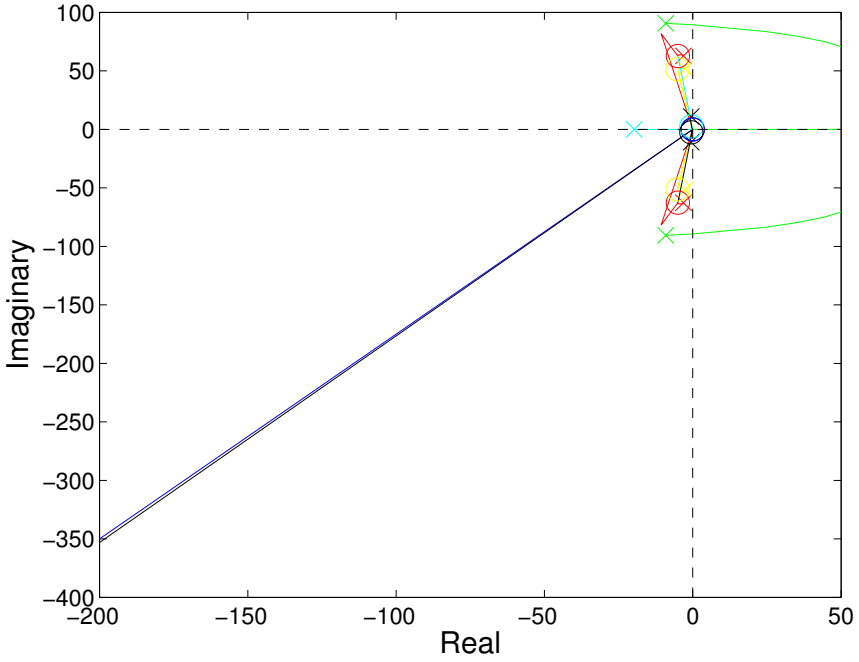

Figure 3.25: Root locus of the base system with a bandpass filter.

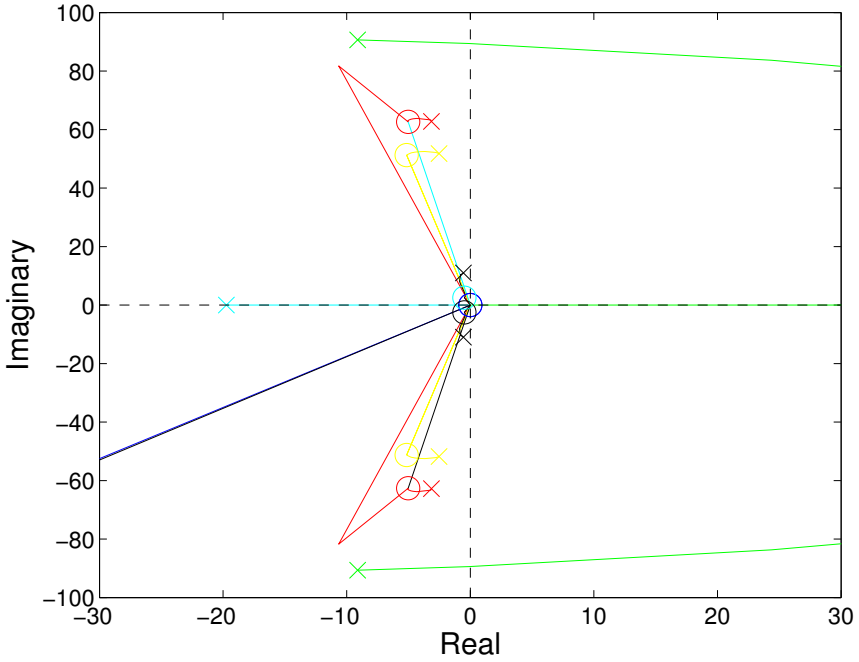

Figure 3.26: Root locus of the base system with a bandpass filter. (Zooming in on Figure 3.25.)

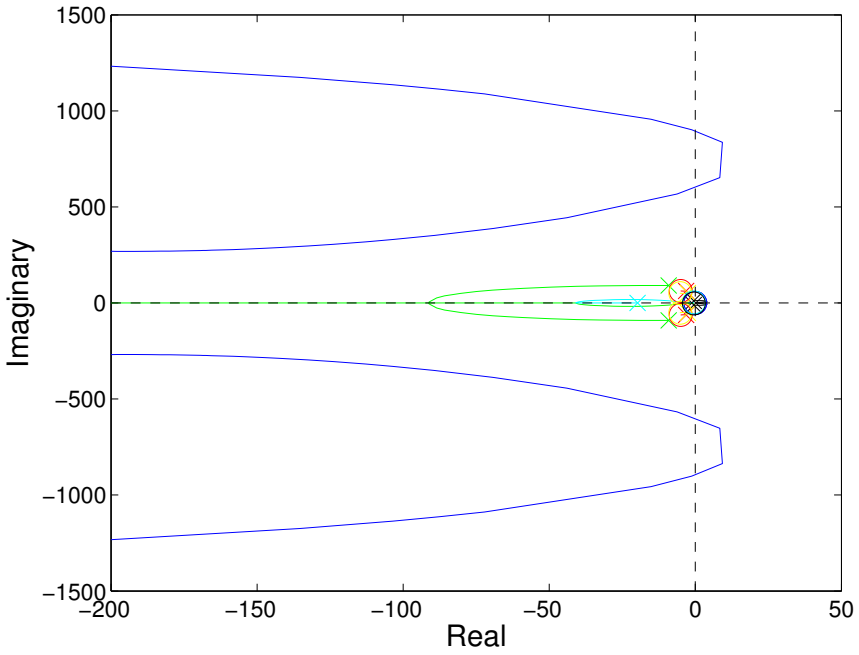

Figure 3.27: Root locus of the base system with a bandpass filter. The gain K_a is negative.

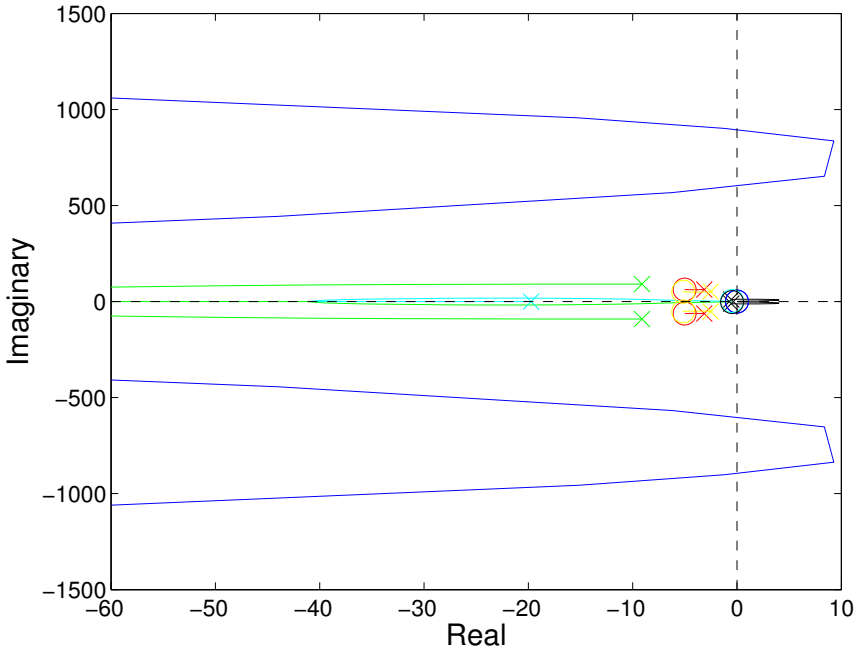

Figure 3.28: Root locus of the base system with a bandpass filter. The gain K_a is negative. (Zooming in on Figure 3.27.)

C.1 Matlab File Used

C.1.1 Matlab File Summary

This file uses the Matlab file `rlocus_full_story_s_2_callfunc.m` This function generates several root loci simulating various system modifications for SAMII (base position sensing, removing actuator dynamics, etc.).

The transfer function $\ddot{x}/\theta$ has an s^2 term in its numerator.

This file was replaces `rlocus_full_story.m`.

C.1.2 Verbatim Matlab Text

`rlocus_full_story_callfunc_s_2_verb_apndx.pdf`

Chapter 4

Comparing s^2 and s^4 Models for the Flexible Base TF

The results reported here are from fitting the data with a model that has an s^2 term in the numerator of the transfer function $\ddot{x}/\theta$. Figures 4.1-4.3 compare the results of curve fitting SAMII's Bode data with a phase weight of 0.1 with models that have s^2 and s^4 terms in the transfer function numerator ($\ddot{x}/\theta$).

The results of this comparison are somewhat inconclusive: both models are able to recreate the system dynamics over the frequency change considered. Figure 4.3 shows that over the frequency range from 20-40Hz, the s^4 model seems to better capture the lack of roll off. This is a small frequency range however and it is near the end of the bandwidth of the input swept sine signal. Also this is near the end of the useful bandwidth of the hydraulic actuator - so it is difficult to excite the system out past this range where the two models begin to differ significantly because of the roll off of the s^2 model.

Curve fitting criteria:

Curve fit set to loop 20 times. Each loop runs the Matlab optimization for a maximum of 500 iterations. The Krauss convergence criteria was set to a maximum percent change in any coeffiecent of 0.01 and an error function maximum percent change of 0.01. These criteria needed to be met for 5 consecutive loops. (The percent changes refer to changes in the values at the end of one loop versus the values at the end of the previous loop where each loop represents 500 iterations.)

Met Matlab convergence criteria. The percent error change on the last loop was -5.4434e-

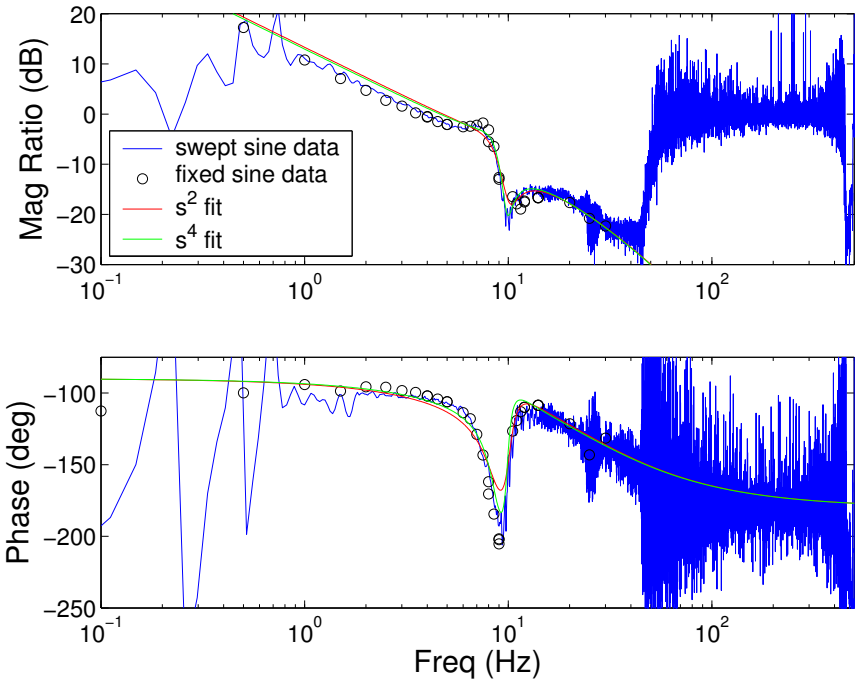

Figure 4.1: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position. The phase weight for this optimization was 0.1.

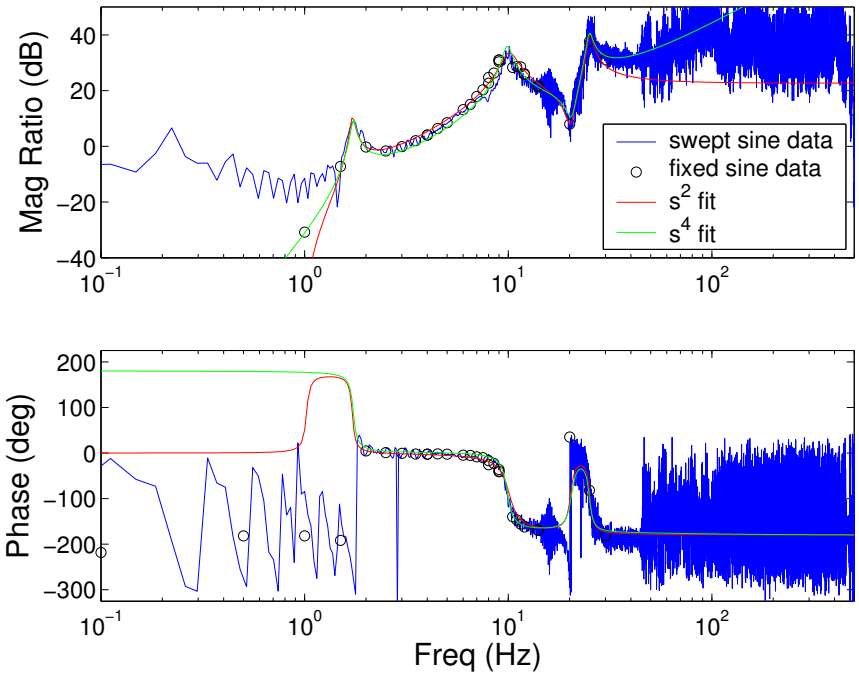

Figure 4.2: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration. The phase weight for this optimization was 0.1.

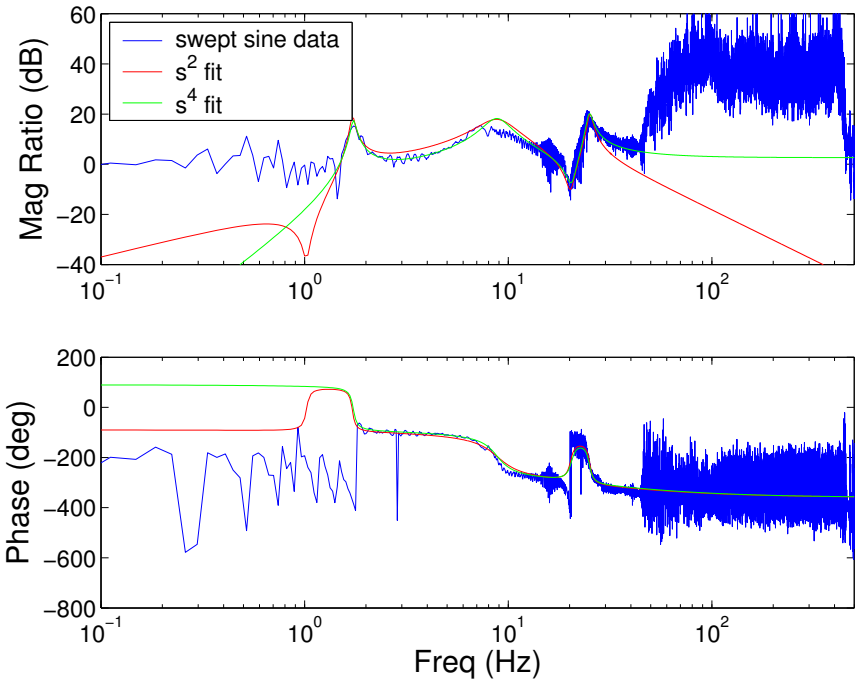

Figure 4.3: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 voltage and the output is base acceleration. The phase weight for this optimization was 0.1.

010% and the percent change in the coefficents was

$$\%\Delta x = \begin{bmatrix} 5.86829e-005 \\ -3.81885e-005 \\ 0.000239918 \\ -9.71491e-006 \\ 0.000186499 \\ -9.51278e-005 \\ 2.81043e-007 \\ 0.000661983 \\ -1.32802e-005 \\ -0.000149074 \\ 0.000103456 \\ -1.03743e-005 \\ -8.48818e-005 \end{bmatrix} \tag{4.1}$$

The output coefficent values were

$$\bar{x} = \begin{bmatrix} 29.1021 \\ 55.7324 \\ 0.202146 \\ 63.3685 \\ 0.084229 \\ 158.514 \\ 10.7661 \\ 0.0249317 \\ 158.082 \\ 0.0284088 \\ 0.168324 \\ -8.02654 \\ -5.77622 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ \omega_3 \\ \zeta_3 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{4.2}$$

These plots were generated by the Matlab file `main_s2_bode_fit_12_03_03.m`, which calls the files `load_data_s2_m3_12_03_03.m`, `run_optim_s2_m3_12_03_03.m`, and `genbodeplots_s2_12_03_03.m`. The optimization uses the error file `samiierr.m` which depends on the model file `samiimodel.m`. The curve fit results are written to the ascii file `bodefit_s2_12_03_03_pw=0_1.txt`.

All of these files are in the folder `C:\ryan\GT\Research\SAMII\curve_fitting\Nov03\s2_fitting`.

D.1 Matlab Files Used

D.1.1 Main Matlab File

The main Matlab file is `main_s2_bode_fit_12_03_03.m` This file is used to demonstrate why the numerator of the transfer function between base acceleration and theta (angular position) needs an s^4 term in the numerator. This is done by curve fitting a model with only an s^2 term in the numerator and comparing the curve fit to the experimental Bode data.

This file calls the files `load_data_s2_m3_12_03_03.m`, `run_optim_s2_m3_12_03_03.m` and `genbodeplots_s2_12_03_03.m`.

D.1.2 Additional Matlab Files

`genbodeplots_s2_12_03_03.m`

This file generates the bode plots for one set of curve fitting results (i.e. one phase weight). This file is called by `main_s2_bode_fit_12_03_03.m`.

Results from fits done with s^2 and s^4 terms in the numerator of $\ddot{x}/\theta$ are overlaid.

`load_data_s2_m3_12_03_03.m`

This file loads experimental swept and fixed sine data. The swept sine data is loaded into global variables used by the curve fitting cost function. This file is called by `main_s2_bode_fit_12_03_03.m`.

`run_optim_s2_m3_12_03_03.m`

This function runs the Matlab function fminsearch for a given set of optmization paramters (maximum nmber of iterations, etc.) and then outputs the results to a text file. This function is called by `main_s2_bode_fit_12_03_03.m` and uses the file `samiierrs2.m` as the cost function for fminsearch.

The model used in the optimization has an s^2 term in the numerator of the transfer function between $\ddot{x}/\theta$.

`samiierrs2.m`

This function defines the cost function for the curve fit. The cost function is the sum of the squared magnitude error in dB plus a phase weighting term (scalar) times the sum of the squared phase error in degrees.

errout=samiierrs2(coeffsin,phaseweight)

The first input (coeffsin) is a vector of input coeffiecients. The second input is the phase weight.

This function is called by `run_optim_s2_m3_12_03_04.m` and calls the function `samiimodels2.m`, which outputs the bode magnitudes and phases for both the actuator and flexible base models based on the vector of input coeffiecents.

`samiimodels2.m`

This function is called by `samiierrors2.m` and `genbodeplots_s2_12_03_03.m`. This function takes a vector of coeffiecients as an input and outputs the bode magnitudes and phases for both the actuator and flexible base models.

This function has an s^2 rather than s^4 term in the numerator of the transfer function between base acceleration and theta (angular position) for comparison purposes.

coeffsin=varargin{1};

varargout{1}=act_fit_mag;
varargout{2}=base_fit_mag;
varargout{3}=act_fit_ph;
varargout{4}=base_fit_ph;

`samiimodels4.m`

This function is called by `genbodeplots_s2_12_03_03.m`. This function takes a vector of coeffiecients as an input and outputs the bode magnitudes and phases for both the actuator and flexible base models.

This function has an s^4 term in the numerator of the transfer function between base acceleration and theta (angular position) for comparison purposes.

coeffsin=varargin{1};

varargout{1}=act_fit_mag;
varargout{2}=base_fit_mag;
varargout{3}=act_fit_ph;
varargout{4}=base_fit_ph;

D.1.3 Verbatim Matlab Files

`s2_vs_s4_curvefitting_verb_apndx.pdf`

Chapter 5

State-space Representation with Base Acceleration and Angular Position Outputs

A state-space representation of the system was needed to perform pole placement design. This was done by starting with the transfer function model of the system and working toward a controllable canonical form.

This manipulation of the system model was done twice: this chapter performs that analysis for one accelerometer output attached at the end of the arm. Chapter 7 performs this same analysis but with two accelerometers used to provide modal feedback.

5.1 Transfer Function Manipulation

The transfer funciton for θ/d can be written as

$$\frac{\theta}{d} = \frac{{\omega_d}^2 \left(s^2 + 2\zeta_2\omega_2 s + {\omega_2}^2\right)\tau}{s{\omega_2}^2 \left(s^2 + 2\zeta_d\omega_d s + {\omega_d}^2\right)(s+\tau)} \tag{5.1}$$

The transfer function for $\ddot{x}/\theta$ can be written as

$$\frac{\ddot{x}}{\theta} = \frac{s^4 B_1 {\omega_1}^2}{s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2} + \frac{s^4 B_2 {\omega_2}^2}{s^2 + 2\zeta_2\omega_2 s + {\omega_2}^2} \tag{5.2}$$

multiplying equation 5.2 by equation 5.1 allows the transfer function between $\ddot{x}/d$ to be written as

$$\frac{\ddot{x}}{d} = \frac{\left(\left({\omega_2}^2 s^2 + 2{\omega_2}^2\zeta_1\omega_1 s + {\omega_1}^2{\omega_2}^2\right)s^3 B_2 + \left({\omega_1}^2{\omega_2}^2 + {\omega_1}^2 s^2 + 2{\omega_1}^2\zeta_2\omega_2 s\right)s^3 B_1\right){\omega_d}^2\tau s}{\left(s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2\right)s{\omega_2}^2\left(s^2 + 2\zeta_d\omega_d s + {\omega_d}^2\right)(s+\tau)} \tag{5.3}$$

For the sake of the state-space representation, we will use the following common denominator for the tranfer functions

$$D = \left(s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2\right)s{\omega_2}^2\left(s^2 + 2\zeta_d\omega_d s + {\omega_d}^2\right)(s+\tau) \tag{5.4}$$

The numerator and denominator of the transfer function θ/d would need to be mulplied by the term

$$\frac{D}{D_\theta} = s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2 \tag{5.5}$$

Expanding the denominator gives

$$\begin{aligned} D = {\omega_2}^2 s^6 &+ \left(2{\omega_2}^2\zeta_1\omega_1 + {\omega_2}^2\tau + 2{\omega_2}^2\zeta_d\omega_d\right) s^5 \\ &+ \left(4{\omega_2}^2\zeta_1\omega_1\zeta_d\omega_d + 2{\omega_2}^2\zeta_1\omega_1\tau + {\omega_1}^2{\omega_2}^2 + {\omega_2}^2{\omega_d}^2 + 2{\omega_2}^2\zeta_d\omega_d\tau\right) s^4 \\ &+ \left(2{\omega_2}^2{\omega_1}^2\zeta_d\omega_d + {\omega_2}^2{\omega_1}^2\tau + {\omega_2}^2{\omega_d}^2\tau + 2{\omega_2}^2\zeta_1\omega_1{\omega_d}^2 + 4{\omega_2}^2\zeta_1\omega_1\zeta_d\omega_d\tau\right) s^3 \\ &+ \left(2{\omega_2}^2{\omega_1}^2\zeta_d\omega_d\tau + {\omega_2}^2{\omega_1}^2{\omega_d}^2 + 2{\omega_2}^2\zeta_1\omega_1{\omega_d}^2\tau\right) s^2 \\ &+ {\omega_2}^2 s{\omega_1}^2{\omega_d}^2\tau \end{aligned} \tag{5.6}$$

Expanding the numerator for θ/d gives

$$\begin{aligned} N_\theta = s^4\tau{\omega_d}^2 &+ \left(2\tau{\omega_d}^2\zeta_1\omega_1 + 2\tau{\omega_d}^2\zeta_2\omega_2\right) s^3 \\ &+ \left({\omega_1}^2{\omega_d}^2\tau + 4\tau{\omega_d}^2\zeta_2\omega_2\zeta_1\omega_1 + {\omega_2}^2{\omega_d}^2\tau\right) s^2 \\ &+ \left(2\tau{\omega_d}^2{\omega_1}^2\zeta_2\omega_2 + 2{\omega_2}^2\zeta_1\omega_1{\omega_d}^2\tau\right) s \\ &+ {\omega_2}^2{\omega_1}^2{\omega_d}^2\tau \end{aligned} \tag{5.7}$$

Expanding the numerator for $\ddot{x}/d$ gives

$$\begin{aligned} N_x = \left(B_1{\omega_1}^2 + B_2{\omega_2}^2\right){\omega_d}^2\tau s^6 &+ \left(2{\omega_2}^2\zeta_1\omega_1 B_2 + 2{\omega_1}^2\zeta_2\omega_2 B_1\right){\omega_d}^2\tau s^5 \\ &+ \left(B_2{\omega_2}^2{\omega_1}^2 + B_1{\omega_1}^2{\omega_2}^2\right){\omega_d}^2\tau s^4 \end{aligned} \tag{5.8}$$

5.2 Controllable Cannonical Form

For a system with the transfer function

$$\frac{y}{u} = \frac{b_n s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} \tag{5.9}$$

the controllable cannonical realization would be

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-1} \end{bmatrix} \tag{5.10}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{5.11}$$

$$\mathbf{C} = \begin{bmatrix} b_0 - b_n a_0 & b_1 - b_n a_1 & \cdots & b_{n-1} - b_n a_{n-1} \end{bmatrix} \tag{5.12}$$

$$\mathbf{D} = b_n \tag{5.13}$$

For this system $n = 6$ and the coefficients of the denominator polynomial are

$$a_0 = 0 \tag{5.14}$$

$$a_1 = {\omega_2}^2 {\omega_1}^2 {\omega_d}^2 \tau \tag{5.15}$$

$$a_2 = 2{\omega_2}^2 {\omega_1}^2 \zeta_d \omega_d \tau + {\omega_2}^2 {\omega_1}^2 {\omega_d}^2 + 2{\omega_2}^2 \zeta_1 \omega_1 {\omega_d}^2 \tau \tag{5.16}$$

$$a_3 = 2{\omega_2}^2 {\omega_1}^2 \zeta_d \omega_d + {\omega_2}^2 {\omega_1}^2 \tau + {\omega_2}^2 {\omega_d}^2 \tau + 2{\omega_2}^2 \zeta_1 \omega_1 {\omega_d}^2 + 4{\omega_2}^2 \zeta_1 \omega_1 \zeta_d \omega_d \tau \tag{5.17}$$

$$a_4 = 4{\omega_2}^2 \zeta_1 \omega_1 \zeta_d \omega_d + 2{\omega_2}^2 \zeta_1 \omega_1 \tau + {\omega_1}^2 {\omega_2}^2 + {\omega_2}^2 {\omega_d}^2 + 2{\omega_2}^2 \zeta_d \omega_d \tau \tag{5.18}$$

$$a_5 = 2{\omega_2}^2 \zeta_1 \omega_1 + {\omega_2}^2 \tau + 2{\omega_2}^2 \zeta_d \omega_d \tag{5.19}$$

$$a_6 = {\omega_2}^2 \tag{5.20}$$

because $a_6 \neq 1$ all of the coefficients (a_n and b_n) must be divided by a_6 before being plugged into the matix representation.
With θ as the output, the coefficients of the numerator polynomial are

$$b_0 = {\omega_2}^2 {\omega_1}^2 {\omega_d}^2 \tau \tag{5.21}$$

$$b_1 = 2\tau {\omega_d}^2 {\omega_1}^2 \zeta_2 \omega_2 + 2{\omega_2}^2 \zeta_1 \omega_1 {\omega_d}^2 \tau \tag{5.22}$$

$$b_2 = {\omega_1}^2 {\omega_d}^2 \tau + 4\tau {\omega_d}^2 \zeta_2 \omega_2 \zeta_1 \omega_1 + {\omega_2}^2 {\omega_d}^2 \tau \tag{5.23}$$

$$b_3 = 2\tau {\omega_d}^2 \zeta_1 \omega_1 + 2\tau {\omega_d}^2 \zeta_2 \omega_2 \tag{5.24}$$

$$b_4 = \tau {\omega_d}^2 \tag{5.25}$$

With $\ddot{x}$ as the output, the coefficients of the numerator polynomial are

$$b_0 = 0 \tag{5.26}$$

$$b_1 = 0 \tag{5.27}$$

$$b_2 = 0 \tag{5.28}$$

$$b_3 = 0 \tag{5.29}$$

$$b_4 = \tau {\omega_d}^2 B_2 {\omega_2}^2 {\omega_1}^2 + \tau {\omega_d}^2 B_1 {\omega_1}^2 {\omega_2}^2 \tag{5.30}$$

$$b_5 = 2{\omega_d}^2 \tau {\omega_2}^2 \zeta_1 \omega_1 B_2 + 2{\omega_d}^2 \tau {\omega_1}^2 \zeta_2 \omega_2 B_1 \tag{5.31}$$

$$b_6 = {\omega_d}^2 \tau B_1 {\omega_1}^2 + {\omega_d}^2 \tau B_2 {\omega_2}^2 \tag{5.32}$$

Finding the transfer function θ/d from the matrices according to

$$\frac{\theta}{d} = \mathbf{C}\left(s\mathbf{I} - \mathbf{A}\right)^{-1}\mathbf{B} + D \tag{5.33}$$

gives

$$\frac{\theta}{d} = \frac{\left(s^2 + 2\zeta_2\omega_2 s + {\omega_2}^2\right)\tau{\omega_d}^2}{{\omega_2}^2\left(s^2 + 2s\zeta_d\omega_d + {\omega_d}^2\right)\left(s+\tau\right)s} \tag{5.34}$$

giving us back what we started with and proving that the state-space representation is correct. Similarly the transfer function $\ddot{x}/d$ from the matrices is

$$\frac{\ddot{x}}{d} = \frac{\left(s^3\left(s^2{\omega_2}^2 + 2\zeta_1\omega_1 s{\omega_2}^2 + {\omega_1}^2{\omega_2}^2\right)B_2 + s^3\left({\omega_1}^2{\omega_2}^2 + 2{\omega_1}^2\zeta_2 s\omega_2 + s^2{\omega_1}^2\right)B_1\right){\omega_d}^2\tau s}{s{\omega_2}^2\left(s^2 + 2s\zeta_d\omega_d + {\omega_d}^2\right)\left(s+\tau\right)\left(s^2 + 2s\zeta_1\omega_1 + {\omega_1}^2\right)} \tag{5.35}$$

which is exactly the transfer function in equation 5.3. The state space matrices are given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -{\omega_1}^2{\omega_d}^2\tau & \frac{-2{\omega_2}^2{\omega_1}^2\zeta_d\omega_d\tau - {\omega_1}^2{\omega_2}^2{\omega_d}^2 - 2{\omega_2}^2\zeta_1\omega_1{\omega_d}^2\tau}{{\omega_2}^2} & \frac{-2{\omega_2}^2{\omega_1}^2\zeta_d\omega_d - {\omega_2}^2{\omega_1}^2\tau - {\omega_2}^2{\omega_d}^2\tau - 2{\omega_2}^2\zeta_1\omega_1{\omega_d}^2 - 4{\omega_2}^2\zeta_1\omega_1\zeta_d\omega}{{\omega_2}^2} \end{bmatrix} \tag{5.36}$$

$$\mathbf{C} = \begin{bmatrix} {\omega_1}^2{\omega_d}^2\tau & \frac{2\tau{\omega_d}^2{\omega_1}^2\zeta_2\omega_2 + 2{\omega_2}^2\zeta_1\omega_1{\omega_d}^2\tau}{{\omega_2}^2} & \frac{{\omega_1}^2{\omega_d}^2\tau + 4\tau{\omega_d}^2\zeta_2\omega_2\zeta_1\omega_1 + {\omega_2}^2{\omega_d}^2\tau}{{\omega_2}^2} \\ 0 & -\frac{\left({\omega_d}^2\tau B_1{\omega_1}^2 + {\omega_d}^2\tau B_2{\omega_2}^2\right){\omega_1}^2{\omega_d}^2\tau}{{\omega_2}^2} & -\frac{\left({\omega_d}^2\tau B_1{\omega_1}^2 + {\omega_d}^2\tau B_2{\omega_2}^2\right)\left(2{\omega_2}^2{\omega_1}^2\zeta_d\omega_d\tau + {\omega_1}^2{\omega_2}^2{\omega_d}^2 + 2{\omega_2}^2\zeta_1\omega_1{\omega_d}^2\tau\right)}{{\omega_2}^4} \end{bmatrix} \tag{5.37}$$

$$\mathbf{D} = \begin{bmatrix} 0 \\ \frac{{\omega_d}^2\tau B_1{\omega_1}^2 + {\omega_d}^2\tau B_2{\omega_2}^2}{{\omega_2}^2} \end{bmatrix} \tag{5.38}$$

This state space system representation is output to the m-file `ccfaccelwnums.m`.

E.1 Matlab Files Used

E.1.1 Main Matlab File

The main Matlab file is `ccfaccel.m` This file starts with the transfer functions for the hydraulic actuator and base acceleration including the first two modes of the flexible base. From there it derives a controllable canonical state space representation of the system. This file outputs the results of its derivation to a LaTeX file for easy readability. The output file is `ccfaccel.tex`.

The transfer functions used in this derivation include an s^4 term in the numerators of the transfer functions between the base accleration and the theta input (angular position) (i.e. $\ddot{x}/\theta$).

It also creates a Matlab m-file that defines the state space matrices in terms of the variables used in this derivation. Editting this file so that it begins with numerically defining each of these variables (i.e. w1=2*pi*10), gives an m-file that has the properly defined state space representation of the system. The output m-file generated by this file is `ccfaccelwnums.m` and the editted version is `ccfaccelwnums_editted.m` which actually calls the file `curvefitparams_s_4_sixth` where the parameters are all defined.sy

E.1.2 Additional Matlab Files

`ccfaccelwnums.m`

This file was created by the m-file C:\ryan\GT\Research\SAMII\statespace\ccfaccel.m and it contains a CCF state-space model for SAMII based on SISO transfer functions about a nominal operating point.

E.1.3 Verbatim Matlab Files

`ccfaccel_verb_apndx.pdf`

Chapter 6

Verifying State-space Model by Comparison with Transfer Function Model

The state-space representation developed in Chatper 5 is verified by overlaying Bode and root locus plots from the transfer function based model and the state-space model.

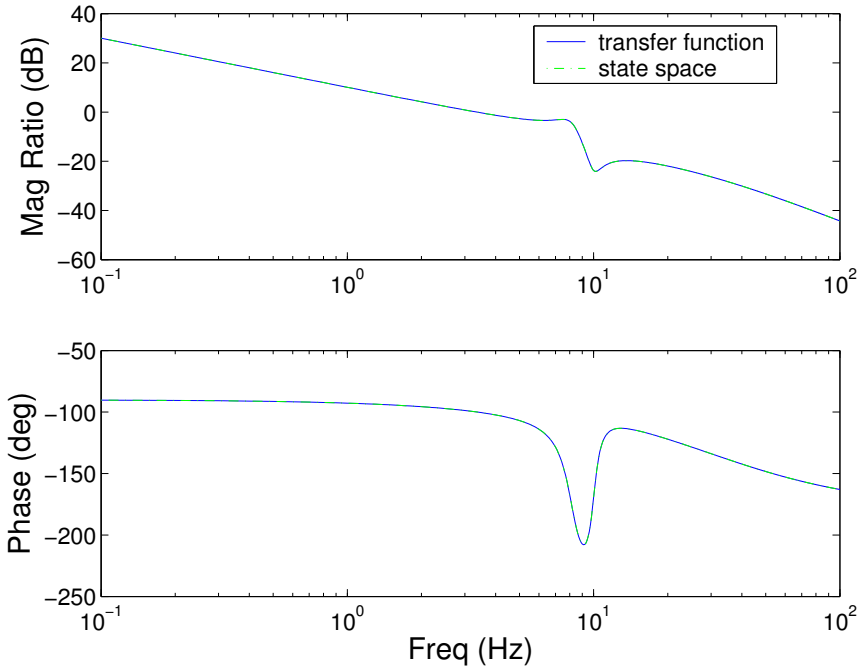

Figure 6.1: Comparison of Bode diagrams for the hydraulic actuator based on the transfer function based model and the state space model.

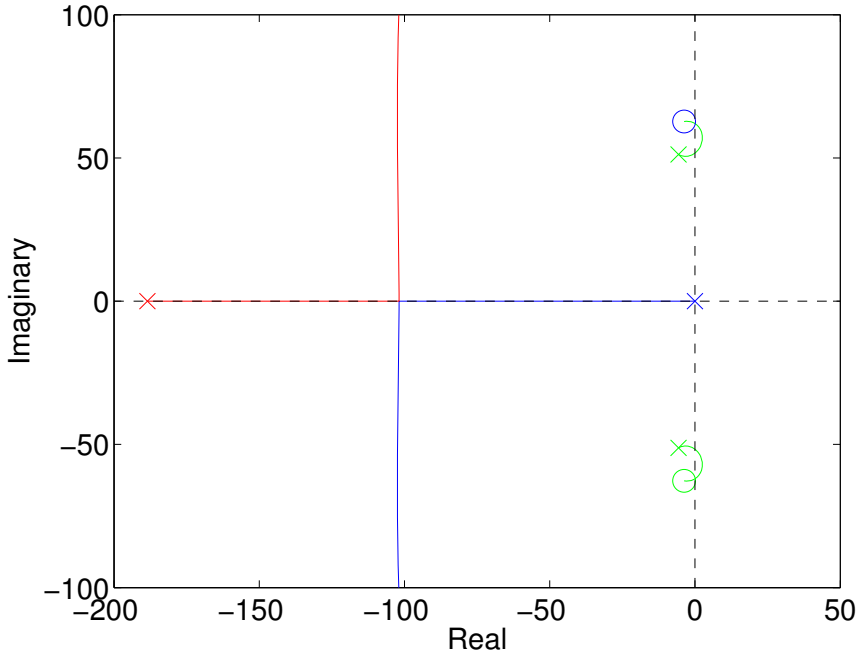

Figure 6.2: Root locus of the hydraulic actuator from the transfer function based model.

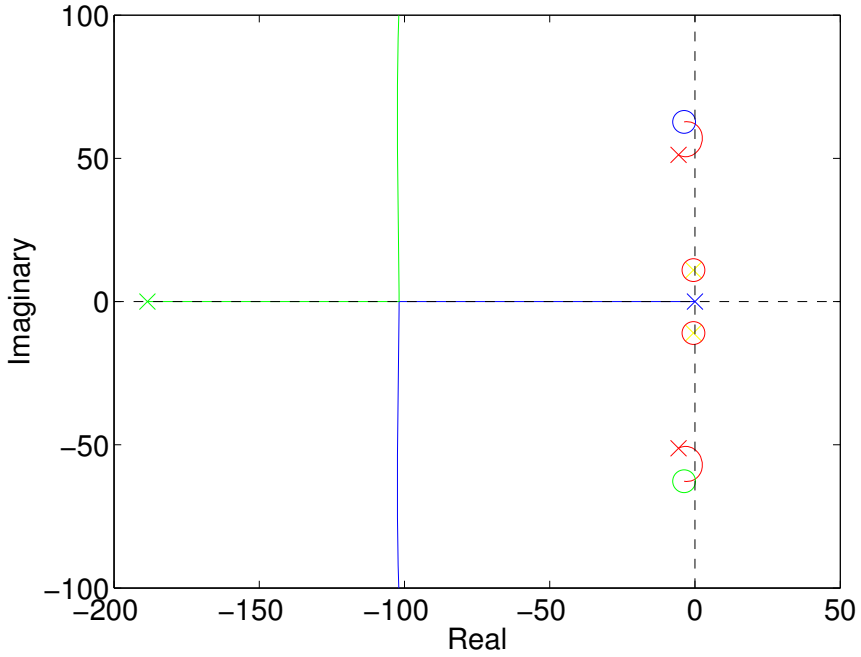

Figure 6.3: Root locus of the hydraulic actuator from the state space model.

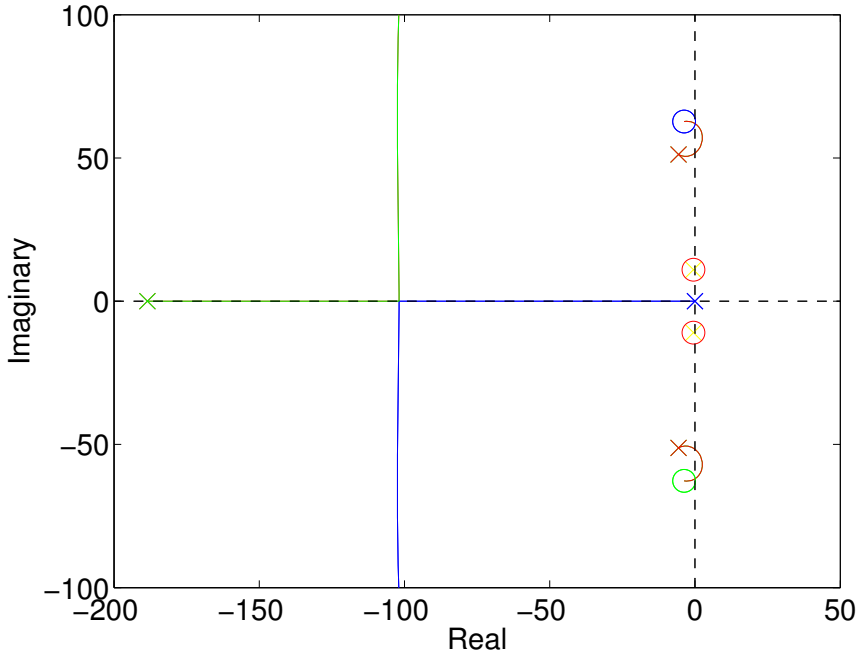

Figure 6.4: Overlay of root loci of the hydraulic actuator from the transfer function based model and the state space model.

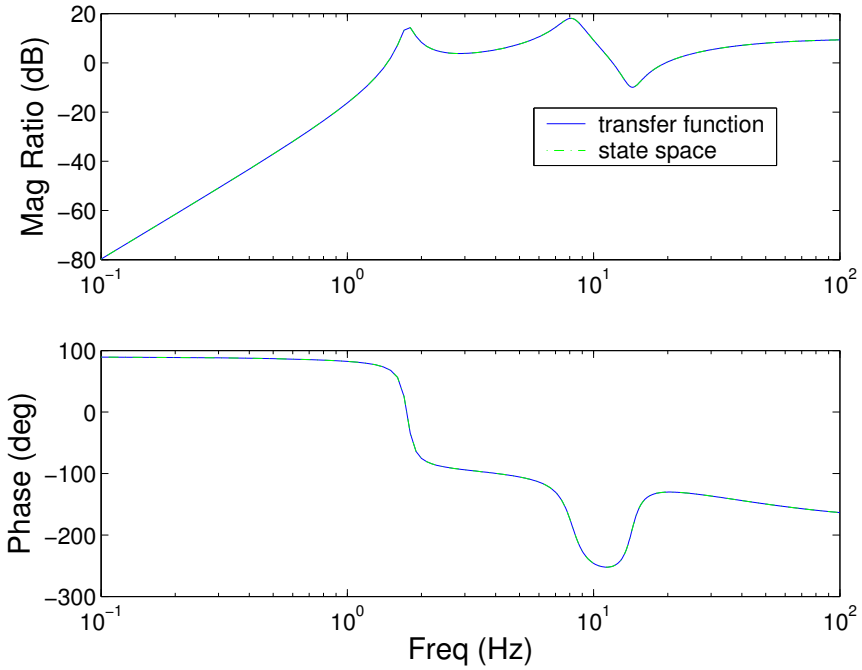

Figure 6.5: Comparison of Bode diagrams for the flexible base based on the transfer function based model and the state space model.

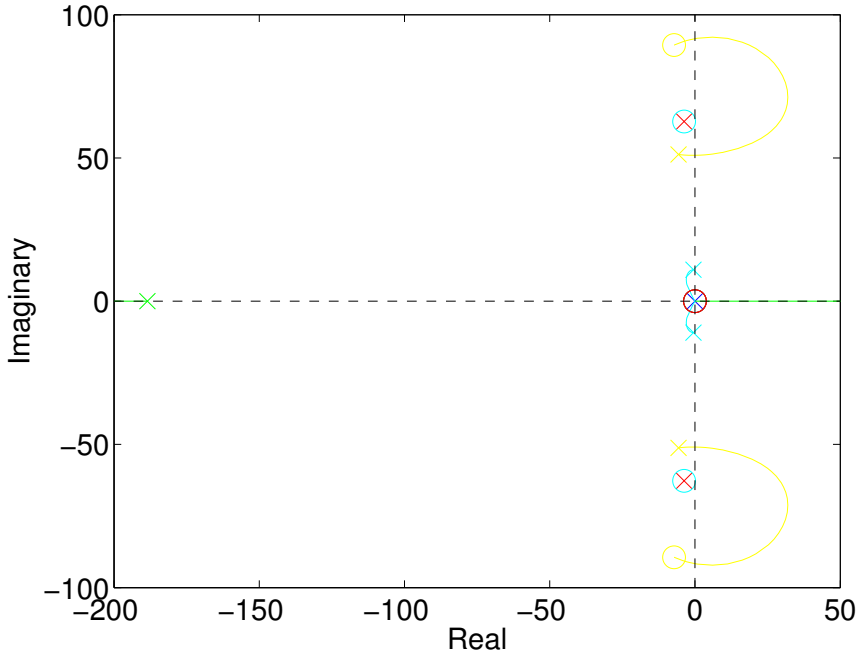

Figure 6.6: Root locus of the flexible base from the transfer function based model.

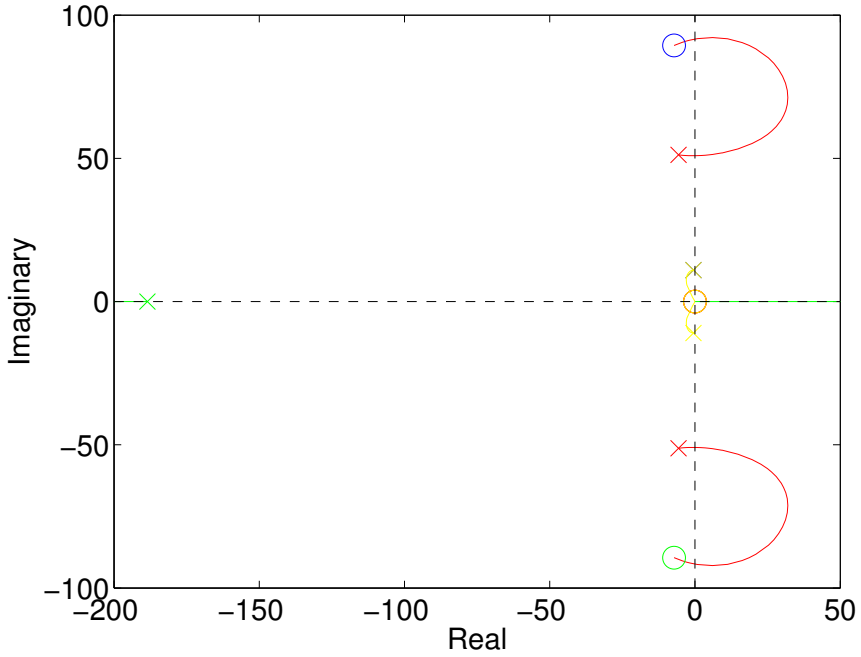

Figure 6.7: Root locus of the flexible base from the state space model.

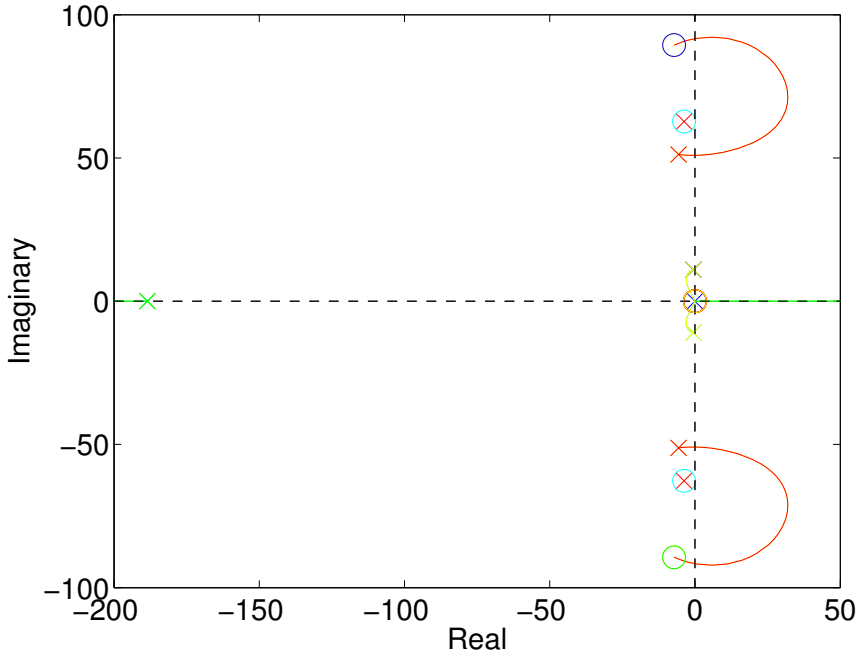

Figure 6.8: Overlay of root loci of the flexible base from the transfer function based model and the state space model.

F.1 Matlab File Used

F.1.1 Matlab File Summary

This file uses the Matlab file `ccfaccelwnums_editted.m` A porition of this file was created by the m-file `C:\ryan\GT\Research\SAMII\statespace\ccfaccel.m` and contains a CCF state-space model for SAMII based on SISO transfer functions about a nominal operating point.

This file calls `curvefitparams_s_4_sixth` to define the system parameters and then generates bode plots and root loci to verify that the state space system is correctly recreating the transfer function based system.

F.1.2 Verbatim Matlab Text

`ccfaccel_tf_ss_comp_verb_apndx.pdf`

Chapter 7

State-space Representation with Two Accelerometers Providing Modal Feedback

A controllable canonical representation of the system with 2 accelerometers providing modal feedback is developed in this chapter based on a transfer function based model.

7.1 Transfer Function Manipulation

The transfer funciton for θ/d can be written as

$$\frac{\theta}{d} = \frac{{\omega_d}^2 \left(s^2 + 2\zeta_2\omega_2 s + {\omega_2}^2\right)\tau}{s{\omega_2}^2 \left(s^2 + 2\zeta_d\omega_d s + {\omega_d}^2\right)(s+\tau)} \tag{7.1}$$

The transfer function for $\ddot{q}_1/\theta$ can be written as

$$\frac{\ddot{q}_1}{\theta} = \frac{s^4 B_1 {\omega_1}^2}{s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2} \tag{7.2}$$

The transfer function for $\ddot{q}_2/\theta$ can be written as

$$\frac{\ddot{q}_2}{\theta} = \frac{s^4 B_2 {\omega_2}^2}{s^2 + 2\zeta_2\omega_2 s + {\omega_2}^2} \tag{7.3}$$

multiplying equation 7.2 by equation 7.1 allows the transfer function between $\ddot{q}_1/d$ to be written as

$$\frac{\ddot{q}_1}{d} = \frac{s^4 {\omega_d}^2 \left(s^2 + 2\zeta_2\omega_2 s + {\omega_2}^2\right)\tau B_1 {\omega_1}^2}{s{\omega_2}^2 \left(s^2 + 2\zeta_d\omega_d s + {\omega_d}^2\right)(s+\tau)\left(s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2\right)} \tag{7.4}$$

multiplying equation 7.3 by equation 7.1 allows the transfer function between $\ddot{q}_2/d$ to be written as

$$\frac{\ddot{q}_2}{d} = \frac{s^4 {\omega_d}^2 \tau B_2}{s\left(s^2 + 2\zeta_d\omega_d s + {\omega_d}^2\right)(s+\tau)} \tag{7.5}$$

where the terms $(s^2 + 2\zeta_2\omega_2 s + {\omega_2}^2)$ and ω_2^2 have canceled between the numerator and denominator.
For the sake of the state-space representation, we will use the following common denominator for the tranfer functions

$$D = s{\omega_2}^2 \left(s^2 + 2\zeta_d\omega_d s + {\omega_d}^2\right)(s+\tau)\left(s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2\right) \tag{7.6}$$

The numerator and denominator of the transfer function θ/d would need to be mulplied by the term

$$\frac{D}{D_\theta} = s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2 \tag{7.7}$$

Similarly, the numerator and denominator of the transfer function $\ddot{q}_2/d$ would need to be mulplied by the term

$$\frac{D}{D_{q2}} = {\omega_2}^2\left(s^2 + 2\zeta_1\omega_1 s + {\omega_1}^2\right) \tag{7.8}$$

Expanding the denominator gives

$$\begin{aligned} D = {} & s^6{\omega_2}^2 + \left(2{\omega_2}^2\zeta_d\omega_d + 2{\omega_2}^2\zeta_1\omega_1 + {\omega_2}^2\tau\right)s^5 \\ & + \left(2{\omega_2}^2\zeta_d\omega_d\tau + 4{\omega_2}^2\zeta_d\omega_d\zeta_1\omega_1 + {\omega_2}^2{\omega_d}^2 + 2{\omega_2}^2\tau\zeta_1\omega_1 + {\omega_2}^2{\omega_1}^2\right)s^4 \\ & + \left({\omega_2}^2{\omega_d}^2\tau + 2{\omega_2}^2{\omega_d}^2\zeta_1\omega_1 + {\omega_2}^2\tau{\omega_1}^2 + 4{\omega_2}^2\zeta_d\omega_d\tau\zeta_1\omega_1 + 2{\omega_2}^2\zeta_d\omega_d{\omega_1}^2\right)s^3 \\ & + \left({\omega_2}^2{\omega_d}^2{\omega_1}^2 + 2{\omega_2}^2{\omega_d}^2\tau\zeta_1\omega_1 + 2{\omega_2}^2\zeta_d\omega_d\tau{\omega_1}^2\right)s^2 \\ & + s{\omega_2}^2{\omega_d}^2\tau{\omega_1}^2 \end{aligned} \tag{7.9}$$

Expanding the numerator for θ/d gives

$$\begin{aligned} N_\theta = {} & s^4{\omega_d}^2\tau + \left(2{\omega_d}^2\tau\zeta_1\omega_1 + 2{\omega_d}^2\tau\zeta_2\omega_2\right)s^3 \\ & + \left({\omega_d}^2\tau{\omega_1}^2 + 4{\omega_d}^2\tau\zeta_2\omega_2\zeta_1\omega_1 + {\omega_2}^2{\omega_d}^2\tau\right)s^2 \\ & + \left(2{\omega_d}^2\tau\zeta_2\omega_2{\omega_1}^2 + 2{\omega_2}^2{\omega_d}^2\tau\zeta_1\omega_1\right)s \\ & + {\omega_2}^2{\omega_d}^2\tau{\omega_1}^2 \end{aligned} \tag{7.10}$$

Expanding the numerator for $\ddot{q}_1/d$ gives

$$N_{q1} = {\omega_d}^2\tau B_1{\omega_1}^2 s^6 + 2{\omega_d}^2\tau\zeta_2\omega_2 B_1{\omega_1}^2 s^5 + {\omega_2}^2{\omega_d}^2\tau B_1{\omega_1}^2 s^4 \tag{7.11}$$

Expanding the numerator for $\ddot{q}_2/d$ gives

$$N_{q2} = s^6{\omega_d}^2\tau B_2{\omega_2}^2 + 2s^5{\omega_d}^2\tau B_2{\omega_2}^2\zeta_1\omega_1 + s^4{\omega_d}^2\tau B_2{\omega_2}^2{\omega_1}^2 \tag{7.12}$$

7.2 Controllable Cannonical Form

For a system with the transfer function

$$\frac{y}{u} = \frac{b_n s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} \tag{7.13}$$

the controllable cannonical realization would be

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-1} \end{bmatrix} \tag{7.14}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{7.15}$$

$$\mathbf{C} = \begin{bmatrix} b_0 - b_n a_0 & b_1 - b_n a_1 & \cdots & b_{n-1} - b_n a_{n-1} \end{bmatrix} \tag{7.16}$$

$$\mathbf{D} = b_n \tag{7.17}$$

For this system $n = 6$ and the coefficients of the denominator polynomial are

$$a_0 = 0 \tag{7.18}$$

$$a_1 = {\omega_2}^2 {\omega_d}^2 \tau {\omega_1}^2 \tag{7.19}$$

$$a_2 = {\omega_2}^2 {\omega_d}^2 {\omega_1}^2 + 2{\omega_2}^2 {\omega_d}^2 \tau \zeta_1 \omega_1 + 2{\omega_2}^2 \zeta_d \omega_d \tau {\omega_1}^2 \tag{7.20}$$

$$a_3 = {\omega_2}^2 {\omega_d}^2 \tau + 2{\omega_2}^2 {\omega_d}^2 \zeta_1 \omega_1 + {\omega_2}^2 \tau {\omega_1}^2 + 4{\omega_2}^2 \zeta_d \omega_d \tau \zeta_1 \omega_1 + 2{\omega_2}^2 \zeta_d \omega_d {\omega_1}^2 \tag{7.21}$$

$$a_4 = 2{\omega_2}^2 \zeta_d \omega_d \tau + 4{\omega_2}^2 \zeta_d \omega_d \zeta_1 \omega_1 + {\omega_2}^2 {\omega_d}^2 + 2{\omega_2}^2 \tau \zeta_1 \omega_1 + {\omega_2}^2 {\omega_1}^2 \tag{7.22}$$

$$a_5 = 2{\omega_2}^2 \zeta_d \omega_d + 2{\omega_2}^2 \zeta_1 \omega_1 + {\omega_2}^2 \tau \tag{7.23}$$

$$a_6 = {\omega_2}^2 \tag{7.24}$$

because $a_6 \neq 1$ all of the coefficients (a_n and b_n) must be divided by a_6 before being plugged into the matix representation.
With θ as the output, the coefficients of the numerator polynomial are

$$b_0 = {\omega_2}^2 {\omega_d}^2 \tau {\omega_1}^2 \tag{7.25}$$

$$b_1 = 2{\omega_d}^2 \tau \zeta_2 \omega_2 {\omega_1}^2 + 2{\omega_2}^2 {\omega_d}^2 \tau \zeta_1 \omega_1 \tag{7.26}$$

$$b_2 = {\omega_d}^2 \tau {\omega_1}^2 + 4{\omega_d}^2 \tau \zeta_2 \omega_2 \zeta_1 \omega_1 + {\omega_2}^2 {\omega_d}^2 \tau \tag{7.27}$$

$$b_3 = 2{\omega_d}^2 \tau \zeta_1 \omega_1 + 2{\omega_d}^2 \tau \zeta_2 \omega_2 \tag{7.28}$$

$$b_4 = {\omega_d}^2 \tau \tag{7.29}$$

With $\ddot{q}_1$ as the output, the coefficients of the numerator polynomial are

$$b_0 = 0 \tag{7.30}$$

$$b_1 = 0 \tag{7.31}$$

$$b_2 = 0 \tag{7.32}$$

$$b_3 = 0 \tag{7.33}$$

$$b_4 = {\omega_2}^2 {\omega_d}^2 \tau B_1 {\omega_1}^2 \tag{7.34}$$

$$b_5 = 2{\omega_d}^2 \tau \zeta_2 \omega_2 B_1 {\omega_1}^2 \tag{7.35}$$

$$b_6 = {\omega_d}^2 \tau B_1 {\omega_1}^2 \tag{7.36}$$

With $\ddot{q}_2$ as the output, the coefficients of the numerator polynomial are

$$b_0 = 0 \tag{7.37}$$

$$b_1 = 0 \tag{7.38}$$

$$b_2 = 0 \tag{7.39}$$

$$b_3 = 0 \tag{7.40}$$

$$b_4 = {\omega_d}^2 \tau B_2 {\omega_2}^2 {\omega_1}^2 \tag{7.41}$$

$$b_5 = 2{\omega_d}^2 \tau B_2 {\omega_2}^2 \zeta_1 \omega_1 \tag{7.42}$$

$$b_6 = {\omega_d}^2 \tau B_2 {\omega_2}^2 \tag{7.43}$$

Finding the transfer function θ/d from the matrices according to

$$\frac{\theta}{d} = \mathbf{C} \left(s\mathbf{I} - \mathbf{A} \right)^{-1} \mathbf{B} + D \tag{7.44}$$

gives

$$\frac{\theta}{d} = \frac{\left(s^2 + 2\zeta_2 \omega_2 s + {\omega_2}^2 \right) {\omega_d}^2 \tau}{{\omega_2}^2 \left(s^2 + 2s\zeta_d \omega_d + {\omega_d}^2 \right) \left(s + \tau \right) s} \tag{7.45}$$

giving us back what we started with and proving that the state-space representation is correct. Similarly the transfer function $\ddot{q}_1/d$ from the matrices is

$$\frac{\ddot{q}_1}{d} = \frac{{\omega_d}^2 \tau {\omega_1}^2 s^3 \left(s^2 + {\omega_2}^2 + 2s\zeta_2 \omega_2 \right) B_1}{{\omega_2}^2 \left(s^2 + 2s\zeta_d \omega_d + {\omega_d}^2 \right) \left(s + \tau \right) \left(s^2 + 2s\zeta_1 \omega_1 + {\omega_1}^2 \right)} \tag{7.46}$$

which is exactly the transfer function in equation 7.4 but with an s cancelled between the numerator and denominator. Similarly the transfer function $\ddot{q}_2/d$ from the matrices is

$$\frac{\ddot{q}_2}{d} = \frac{B_2 s^3 {\omega_d}^2 \tau}{\left(s^2 + 2s\zeta_d \omega_d + {\omega_d}^2 \right) \left(s + \tau \right)} \tag{7.47}$$

which is exactly the transfer function in equation 7.5 but with an s cancelled between the numerator and denominator. The state space matrices are given by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -{\omega_d}^2\tau{\omega_1}^2 & \frac{-{\omega_2}^2{\omega_d}^2{\omega_1}^2-2{\omega_2}^2{\omega_d}^2\tau\zeta_1\omega_1-2{\omega_2}^2\zeta_d\omega_d\tau{\omega_1}^2}{{\omega_2}^2} & \frac{-{\omega_2}^2{\omega_d}^2\tau-2{\omega_2}^2{\omega_d}^2\zeta_1\omega_1-{\omega_2}^2\tau{\omega_1}^2-4{\omega_2}^2\zeta_d\omega_d\tau\zeta_1\omega_1-2{\omega_2}^2\zeta_d\omega_d\omega \end{bmatrix} \tag{7.48}$$

$$\mathbf{C} = \begin{bmatrix} {\omega_d}^2\tau{\omega_1}^2 & \frac{2{\omega_d}^2\tau\zeta_2\omega_2{\omega_1}^2+2{\omega_2}^2{\omega_d}^2\tau\zeta_1\omega_1}{{\omega_2}^2} & \frac{{\omega_d}^2\tau{\omega_1}^2+4{\omega_d}^2\tau\zeta_2\omega_2\zeta_1\omega_1+{\omega_2}^2{\omega_d}^2\tau}{{\omega_2}^2} \\ 0 & -\frac{{\omega_d}^4\tau^2B_1{\omega_1}^4}{{\omega_2}^2} & -\frac{{\omega_d}^2\tau B_1{\omega_1}^2\left({\omega_2}^2{\omega_d}^2{\omega_1}^2+2{\omega_2}^2{\omega_d}^2\tau\zeta_1\omega_1+2{\omega_2}^2\zeta_d\omega_d\tau{\omega_1}^2\right)}{{\omega_2}^4} & -\frac{{\omega_d}^2\tau B_1{\omega_1}^2\left({\omega_2}^2{\omega_d}^2\tau \\ 0 & -{\omega_d}^4\tau^2B_2{\omega_1}^2 & -\frac{{\omega_d}^2\tau B_2\left({\omega_2}^2{\omega_d}^2{\omega_1}^2+2{\omega_2}^2{\omega_d}^2\tau\zeta_1\omega_1+2{\omega_2}^2\zeta_d\omega_d\tau{\omega_1}^2\right)}{{\omega_2}^2} & -\frac{{\omega_d}^2\tau B_2\left({\omega_2}^2{\omega_d}^2\tau+ \end{bmatrix} \tag{7.49}$$

$$\mathbf{D} = \begin{bmatrix} 0 \\ \frac{{\omega_d}^2\tau B_1{\omega_1}^2}{{\omega_2}^2} \\ {\omega_d}^2\tau B_2 \end{bmatrix} \tag{7.50}$$

This state space system representation is output to the m-file `ccfmodalwnums.m`.

F.1 Matlab Files Used

F.1.1 Main Matlab File

The main Matlab file is `ccfmodal.m` This file starts with the transfer functions for the hydraulic actuator and the first two modes of the flexible base. From there it derives a controllable canonical state space representation of the system. This file outputs the results of its derivation to a \LaTeX file for easy readability. The output file is `ccfmodal.tex`.

The transfer functions used in this derivation include an s^4 term in the numerators of the transfer functions between the modal acclerations and the theta input (angular position) (i.e. $\ddot{q}_i/\theta$).

It also creates a Matlab m-file that defines the state space matrices in terms of the variables used in this derivation. Editting this file so that it begins with numerically defining each of these variables (i.e. w1=2*pi*10), gives an m-file that has the properly defined state space representation of the system. The output m-file generated by this file is `ccfmodalwnums.m` and the editted version is `ccfmodalwnums_editted.m` which actually calls the file `curvefitparams_s_4_sixth` where the parameters are all defined.sy

F.1.2 Additional Matlab Files

`ccfmodalwnums.m`

This file was created by the m-file C:\ryan\GT\Research\SAMII\statespace\ccfmodal.m and it contains a CCF state-space model for SAMII based on SISO transfer functions about a nominal operating point.

`ccfmodalwnums_editted.m`

A porition of this file was created by the m-file `C:\ryan\GT\Research\SAMII\statespace\ccfmodal.m` and contains a CCF state-space model for SAMII based on SISO transfer functions about a nominal operating point. This file calls `curvefitparams_s_4_sixth` to define the system parameters and then generates bode plots and root loci to verify that the state space system is correctly recreating the transfer function based system.

This file also includes a pole placement controller design for SAMII that was proposed in my IEEE Aerospace conference paper. The controller is designed in continuous time and does not include an observer (i.e. all states are assumed measurable and availible for full state feedback - obviously this is not the case, but it is used as a starting point to verify that the controller would work well if full state feedback was availible).

F.1.3 Verbatim Matlab Files

`ccfmodal_verb_apndx.pdf`

Chapter 8

Initial Digital Pole Placement Controller/Observer Design

This file documents the design of a digital state feedback observer controller for SAMII. The open loop pole locations for SAMII operating around a nominal configuration of $(-90°, 90°, 90°, 0°, 0°, 0°)$ are

$$p_{ol} = \begin{bmatrix} 0 \\ -188.5 \\ -6.237 + 52.01i \\ -6.237 - 52.01i \\ -0.4361 + 10.55i \\ -0.4361 - 10.55i \end{bmatrix} \tag{8.1}$$

The poles for a system having unity θ feedback and no vibration suppression are

$$p_{\theta fb} = \begin{bmatrix} -169.8 \\ -2.986 + 51.56i \\ -2.986 - 51.56i \\ -25.19 \\ -0.4361 + 10.55i \\ -0.4361 - 10.55i \end{bmatrix} \tag{8.2}$$

The desired pole locations for the state feedback system being designed are

$$p_{des} = \begin{bmatrix} -169.8 \\ -36.52 - 36.52i \\ -36.52 + 36.52i \\ -25.19 \\ -7.466 - 7.466i \\ -7.466 + 7.466i \end{bmatrix} \tag{8.3}$$

Figure 8.1 plots the real vs. imaginary parts of these poles.
Figure 8.2 plots the real vs. imaginary parts of the digital poles.

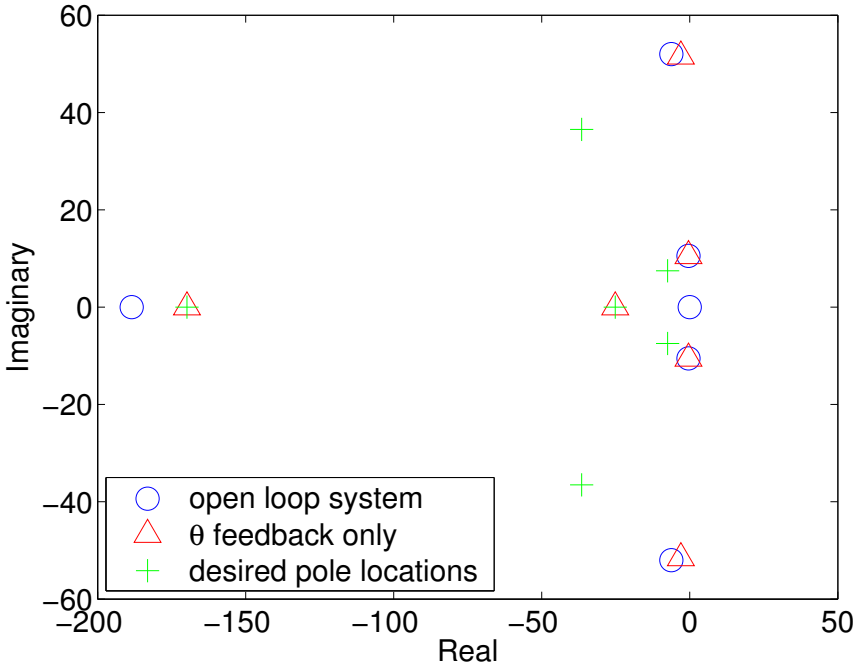

Figure 8.1: Pole locations for the open loop system, a controller for SAMII that has only θ feedback (i.e. no vibration suppression), and the desired pole locations for a full state feedback system.

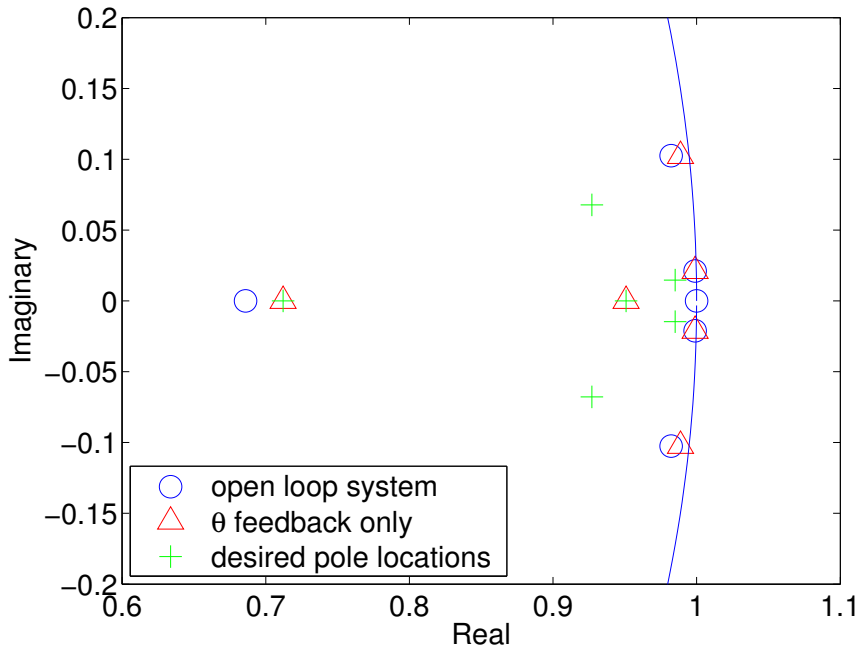

Figure 8.2: Digital pole locations for the open loop system, a controller for SAMII that has only θ feedback (i.e. no vibration suppression), and the desired pole locations for a full state feedback system.

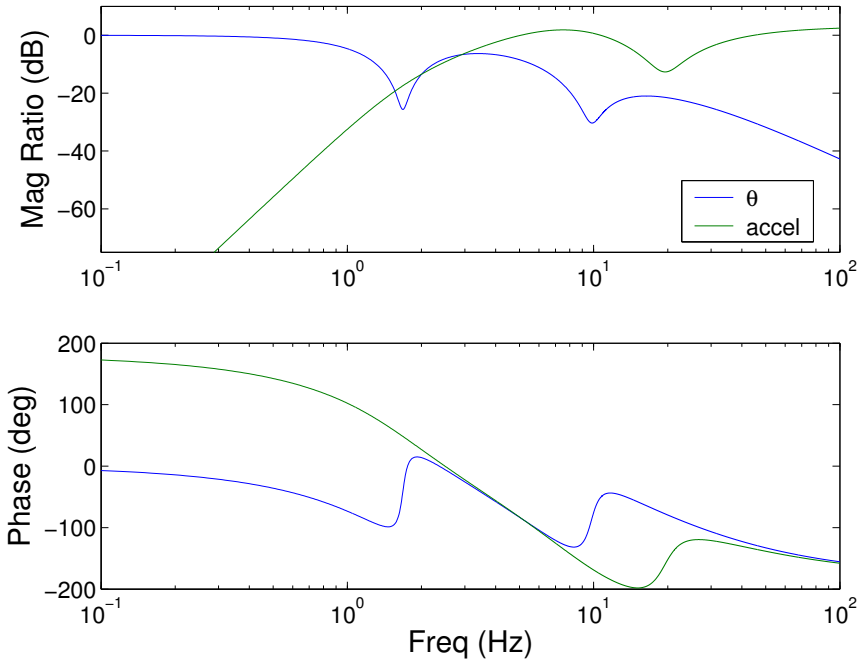

Figure 8.3: Bode plots for θ/v and $\ddot{x}/\theta$ for the SAMII control system with the desired closed loop poles.

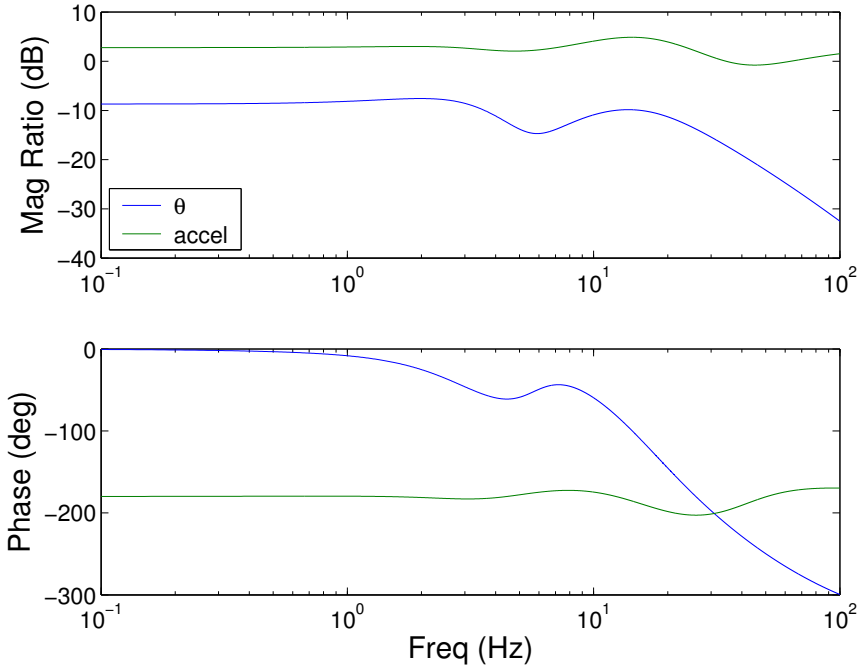

Figure 8.4: Bode plots for an observer system designed by pole placement.

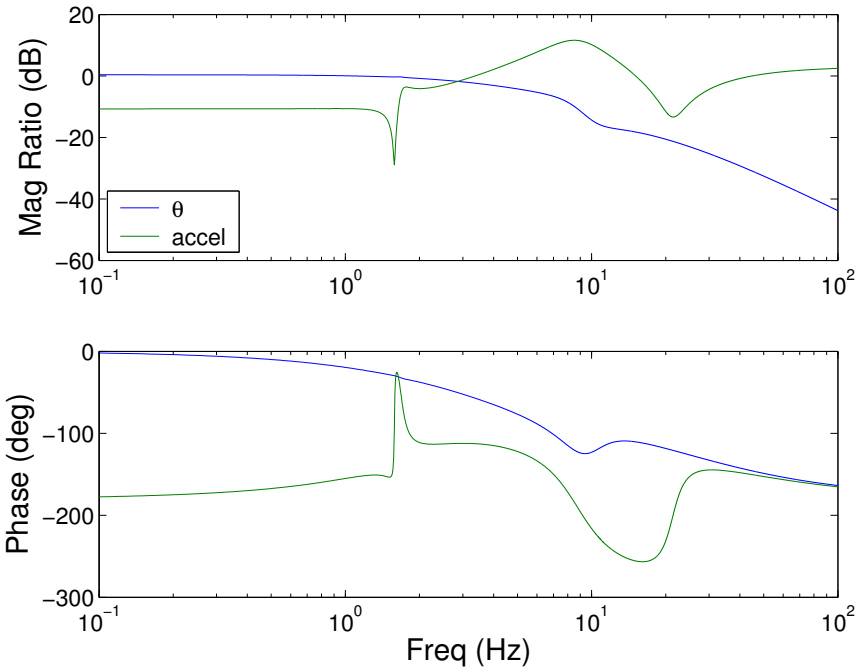

Figure 8.5: Bode plots for an observer system based on Kalman design.

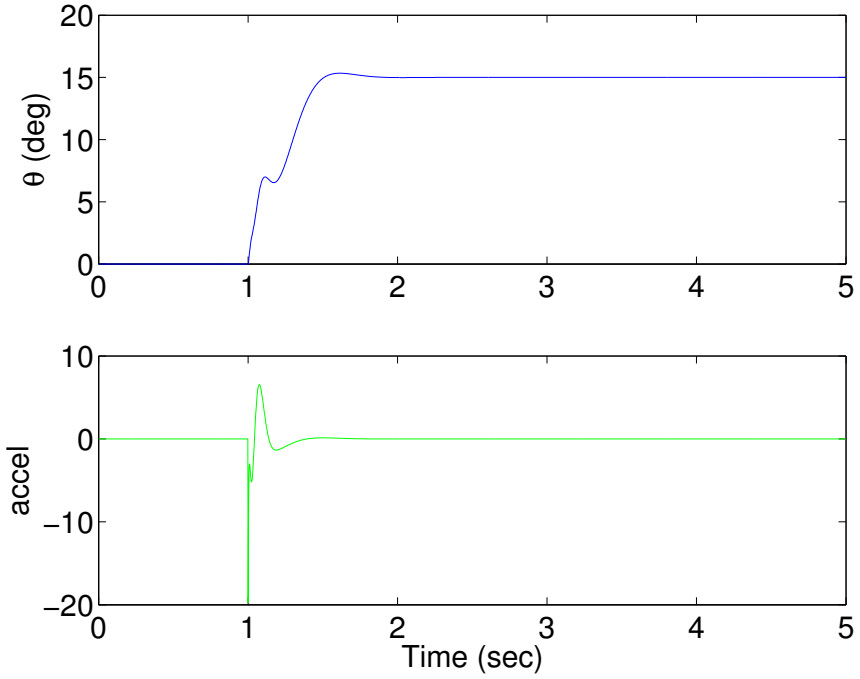

Figure 8.6: Step response for the state feedback controller observer system with no noise.

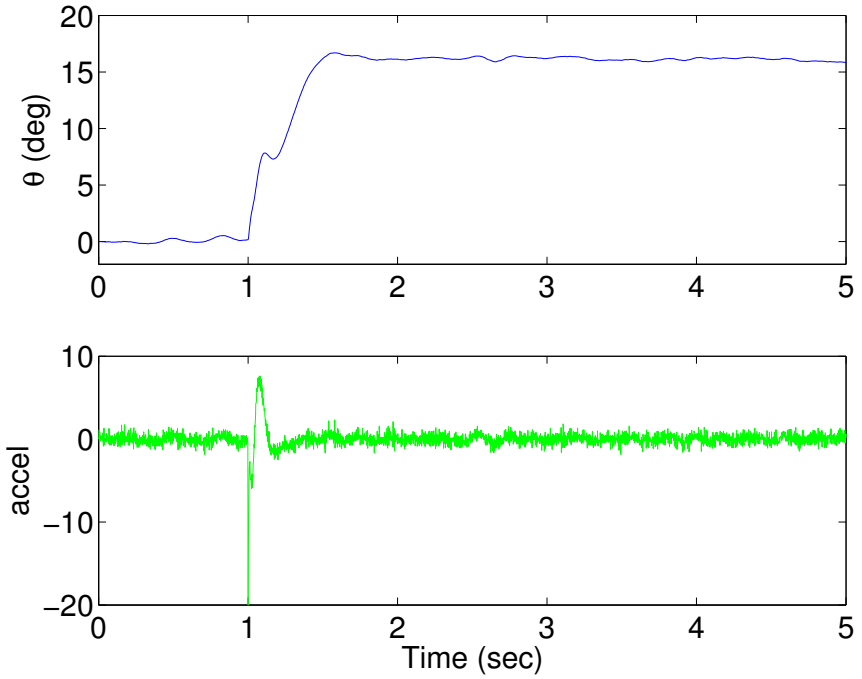

Figure 8.7: Step response for the state feedback controller observer system from a Simulink simulation with accelerometer noise.

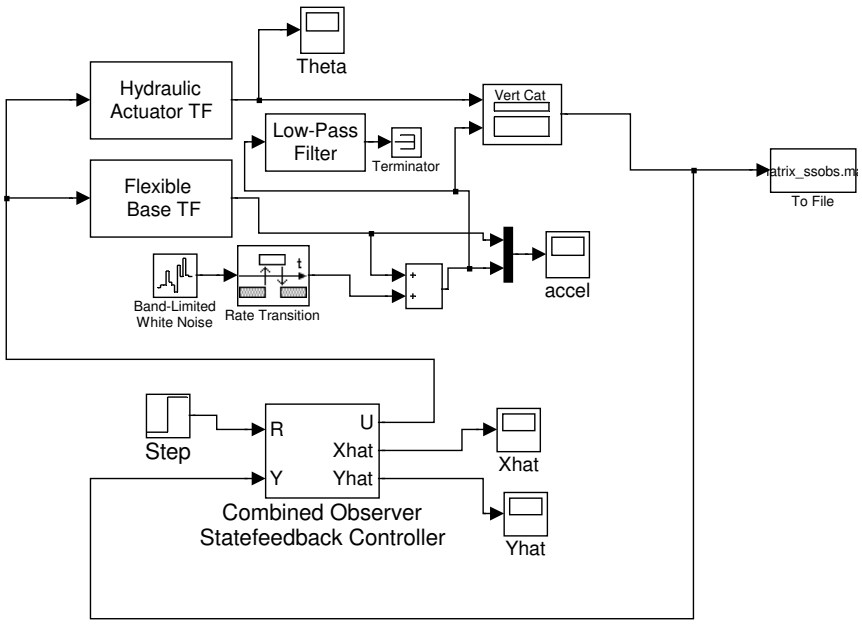

Figure 8.8: Block diagram of the Simulink simulation.

Figure 8.3 shows a bode plot for θ/v for the closed loop state feedback system with the desired pole locations.

Figure 8.4 shows a bode plot for observer system designed by pole placement.

Figure 8.5 shows a bode plot for observer system designed as a Kalman filter.

Figure 8.6 shows the step response of the state feedback controller observer without any noise.

Figure 8.7 shows the step response of the state feedback controller observer system from a simulation in Simulink with accelerometer noise.

Figure 8.8 shows the block diagram of the Simulink simulaiton.

While the acceleration signals from Figures 8.6 and 8.7 look promising, this controller made the third mode of the flexible base unstable when implemented experimentally. This third mode instability lead me to refit a wider frequency range of my bode data, so that the third mode was included in my models. Once I had the new models, I began redesigning a new controller similar to this one but considering the third mode.

G.1 Matlab Files Used

G.1.1 Main Matlab File

The main Matlab file is `digital_ss_11_05_03.m` This file designs a digital pole placement state feedback controller/observer for SAMII for a system model with 2 modes of the flexible base. This controller seems to work well in simulation, though it does have some noise sensitity concerns. When implemented experimentally, this controller made the third mode of the system go unstable. This instability is kept in check by the fact that the thrid mode (around 30Hz) is near the limit of the bandwidth of the system. However, this controller does not really work in practice. This lead me to curve fit a wider frequency range of my bode data to get a model that includes the third mode, so that I could design and test a new controller.

It outputs a text file that can be cut and pasted into Simulink that includes digital A,B,C,D,K, and L matrices. The digital system is found using the Matlab c2d function with the 'zoh' option. The K and L matrices are found by performing pole placement on the digital system.

The desired pole locations for the state feedback controller were found by first finding the eigenvalues of the system with unity θ feedback (this gives good response as far as the speed of getting θ to the desired position, but it is fast enough to excite significant vibration if nothing is done to suppression vibration). The real poles of this system with unity θ feedback are left unchanged. The complex poles are then moved to a pole location having a specified damping ratio, but the same magnitude as the lightly damped pole. The calculation of the desired pole locations is handled by the function `clpolelocs.m`.

These desired pole locations were then mapped onto desired digital pole locations using $z = e^{sT}$.

The observer pole locations were found two ways: first placing them at twice as fast as the controller poles ($p_L = 2p_c$) in the continuous domain (before mapping to the z domain) and then by designing a Kalman filter using the Matlab `kalman` function.

The parameters used for the system model are from a curve fit that does not consider phase error with a model with an s^4 term in the numerator of the transfer function of $\ddot{x}/\theta$. The model is sixth order and considers the first two modes of the flexible base. The system parameters are loaded by the function `loadparams_ss_11_05_03.m`;

This function loads data from the mat-file `matrix_ssobs.mat` which is the output of the Simulink file `matrix_ssobs_obs_subsys_11_10_03.mdl` that simulates the state feedback controller with accelerometer noise.

G.1.2 Additional Matlab Files

`clpolelocs.m`

```
A=varargin{1};
B=varargin{2};
C=varargin{3};
deszeta=varargin{4};
```

varargout{1}=eigcl;

This function finds the desired closed loop poles of a state feedback controller for SAMII by first finding the poles with unity theta feedback and then moving any complex poles to the same natural frequency but with the specified damping ratio.

`loadparams_ss_11_05_03.m`

This function loads system parameters from a curve fit that does not consider phase error with a model with an s^4 term in the numerator of the transfer function of $\ddot{x}/\theta$. The model is sixth order and considers the first two modes of the flexible base.

`matrix_ssobs.mat`

`matrix_ssobs_obs_subsys_11_10_03.mdl`

G.1.3 Verbatim Matlab Files

`digital_ss_design_m2_editted_verb_apndx.pdf`

Chapter 9

System ID with 3 Flexible Base Modes

The third mode instability seen when implementing the controller in Chapter 8 lead me to refit a wider frequency range of my Bode data and include the third mode in my models.

In this chapter, I also considered including the phase error in my optimization (in my previous curve fitting efforts, the phase was predicted fairly well by the models even though it was not considered in the error function). The overall error I was trying to minimize was the sum of the squared magnitude error in dB plus a phase weight times the sum of the squared phase error in degrees. Results are shown for phase weights of 0-0.5 by steps of 0.1. Figure 9.1 shows that with phase error not considered (phase weight=0), the actuator model does not predict the phase near 10Hz as well as previous models (with only 2 modes).

9.1 Introduction

The data used for the curve fitting in this system i.d. work was generated using a swept sine input to SAMII's joint 2. This was done with SAMII in a nominal configuration of joint 1 = -90°, joint 2 = 90°, and joint 3 = 90°, and joints 4-6 = 0°. This puts SAMII in a configuration where joint 2 positions the second link vertical and joint 3 and the end effector point north.

The swept sine data is based on 3 averages. The swept sine input has frequency content from 0.1-40Hz.

The swept sine data is saved in the Matlab file `swept_sine_bode_08_25_03.mat`. Fixed sine data is saved in the file `fixed_sine_bode_08_22_03.mat`. The fixed sine data is overlaid for comparison only and is not used in any calculations.

The model used for the curve fitting can be found in the file `samiimodel.m`.

The hydraulic actuator was modeled with the transfer function

$$\frac{\theta}{v} = \frac{K_1\omega_p^2\left(s^2 + 2\zeta_z\omega_z s + \omega_z^2\right)\tau}{s\omega_z^2\left(s^2 + 2\zeta_p\omega_p s + \omega_p^2\right)(s+\tau)} \tag{9.1}$$

This model was implemented using the Matlab code:

```
hyd_act=tf(k1*wp^2*[1,2*wz*zz,wz^2],wz^2*[1,2*wp*zp1,wp^2,0]);
```

```
fo_lag=tf(tau,[1 tau]);
hyd_act=hyd_act*fo_lag;
```

The flexible base was modeled by with the transfer function

$$\frac{\ddot{x}}{\theta} = s^4 \left(\frac{B_1}{s^2 + 2\zeta_1\omega_1 s + \omega_1^2} + \frac{B_2}{s^2 + 2\zeta_2\omega_2 s + \omega_2^2} + \frac{B_3}{s^2 + 2\zeta_3\omega_3 s + \omega_3^2} \right) \tag{9.2}$$

where $\omega_2 = \omega_z$ and $\zeta_2 = \zeta_z$.

This model was implemented using the Matlab code:

```
mode1=tf(1,[1,2*z1*w1,w1^2]);
mode2=tf(1,[1,2*z2*w2,w2^2]);
mode3=tf(1,[1,2*z3*w3,w3^2]);
qd=tf([1 0 0 0 0],1);
flexb=qd*(B1*mode1+B2*mode2+B3*mode3);
```

The curve fitting was done using the Matlab function `fminsearch`. The error function defined the error as the squared sum of the magnitude errors in dB from the swept sine data for both the actuator and the flexible base plus the squared sum of the phase error (for both the actuator and the flexible base) times a phase weighting factor. This error function was implemented using Matlab code:

```
ev1=20*log10(mean_io_mr)-20*log10(act_fit_mag);
ev2=20*log10(mean_a2_j2a_mr)-20*log10(base_fit_mag);
phe1=mean_io_ph-act_fit_ph;
phe2=mean_a2_j2a_ph-base_fit_ph;

evt=[ev1;ev2;phaseweight*phe1;phaseweight*phe2];

errout=sum(evt.^2);
```

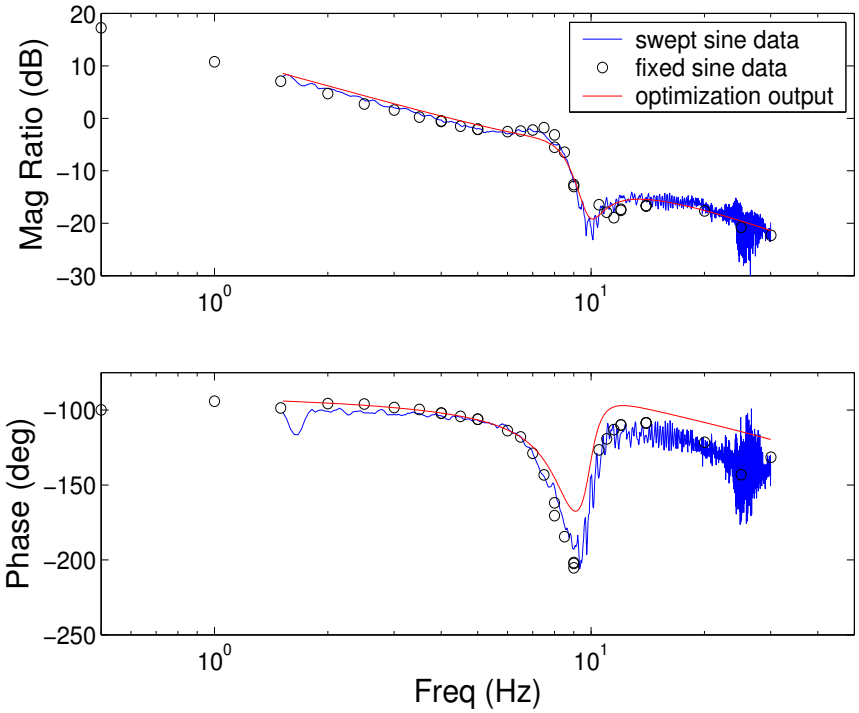

Figure 9.1: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position. The phase weight for this optimization was 0.

9.2 Phase Weight=0 (phase error not considered)

Figures 9.1 and 9.2 show the results of curve fitting SAMII's Bode data with a phase weight of 0.

Curve fitting criteria:

Curve fit set to loop 20 times. Each loop runs the Matlab optimization for a maximum of 500 iterations. The Krauss convergence criteria was set to a maximum percent change in any coeffiecent of 0.01 and an error function maximum percent change of 0.01. These criteria needed to be met for 5 consecutive loops. (The percent changes refer to changes in the values at the end of one loop versus the values at the end of the previous loop where each loop represents 500 iterations.)

Met Krauss convergence criteria.

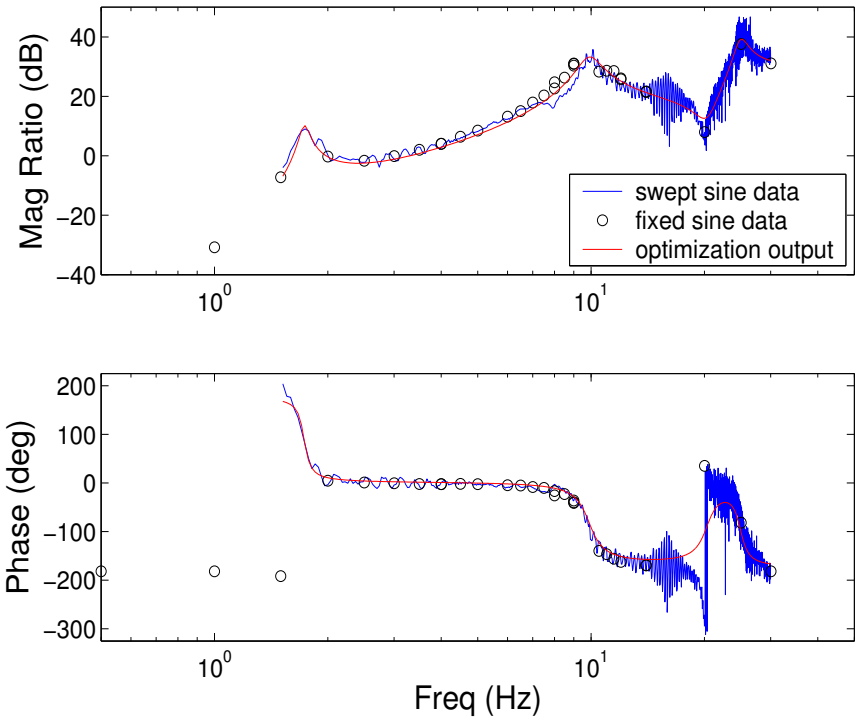

Figure 9.2: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration. The phase weight for this optimization was 0.

The output coefficent values were

$$\bar{x} = \begin{bmatrix} 25.4406 \\ 54.6046 \\ 0.163755 \\ 62.1284 \\ 0.0677148 \\ 294.127 \\ 10.8724 \\ 0.0281329 \\ 156.561 \\ 0.0395 \\ -0.00153797 \\ 0.00159421 \\ 0.000276836 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ \omega_3 \\ \zeta_3 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{9.3}$$

These plots were generated by the Matlab file `main_bode_fit_11_14_03.m`, which calls the files `load_data_s4_m3_11_14_03.m`, `run_optim_s4_m3_11_14_03.m`, and `genbodeplots_11_14_03.m`. The optimization uses the error file `samiierr.m` which depends on the model file `samiimodel.m`. The curve fit results are written to the ascii file `bodefit_11_14_03_pw=0.txt`.

All of these files are in the folder `C:\DocumentsandSettings\ryan\MyDocuments\GT\Research\SAMII\curve_fitting\Nov03`.

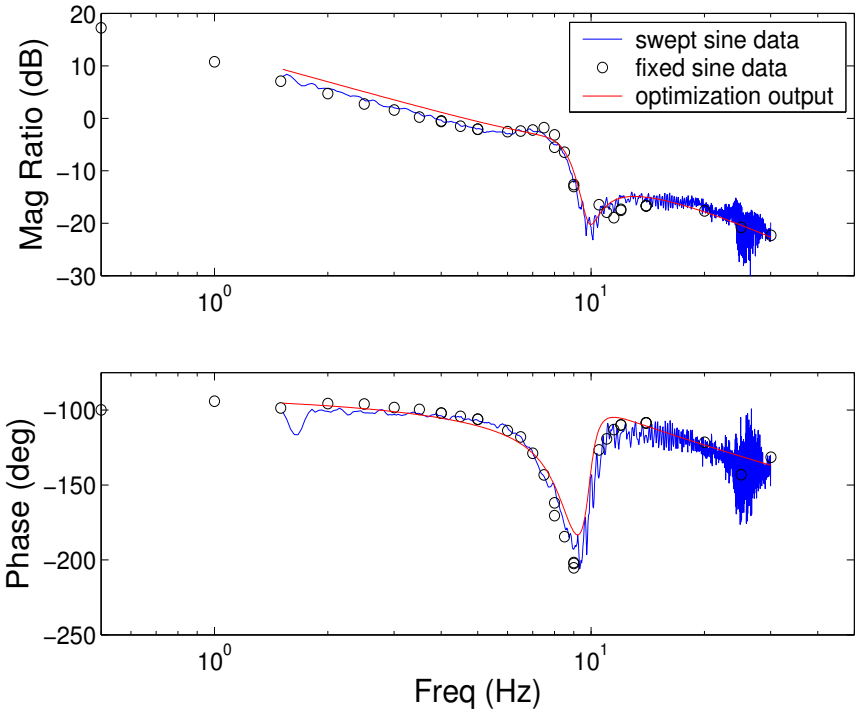

Figure 9.3: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position. The phase weight for this optimization was 0.1.

9.3 Phase Weight=0.1

Figures 9.3 and 9.4 show the results of curve fitting SAMII's Bode data with a phase weight of 0.1.

Curve fitting criteria:

Curve fit set to loop 20 times. Each loop runs the Matlab optimization for a maximum of 500 iterations. The Krauss convergence criteria was set to a maximum percent change in any coeffiecent of 0.01 and an error function maximum percent change of 0.01. These criteria needed to be met for 5 consecutive loops. (The percent changes refer to changes in the values at the end of one loop versus the values at the end of the previous loop where each loop represents 500 iterations.)

Met Matlab convergence criteria.

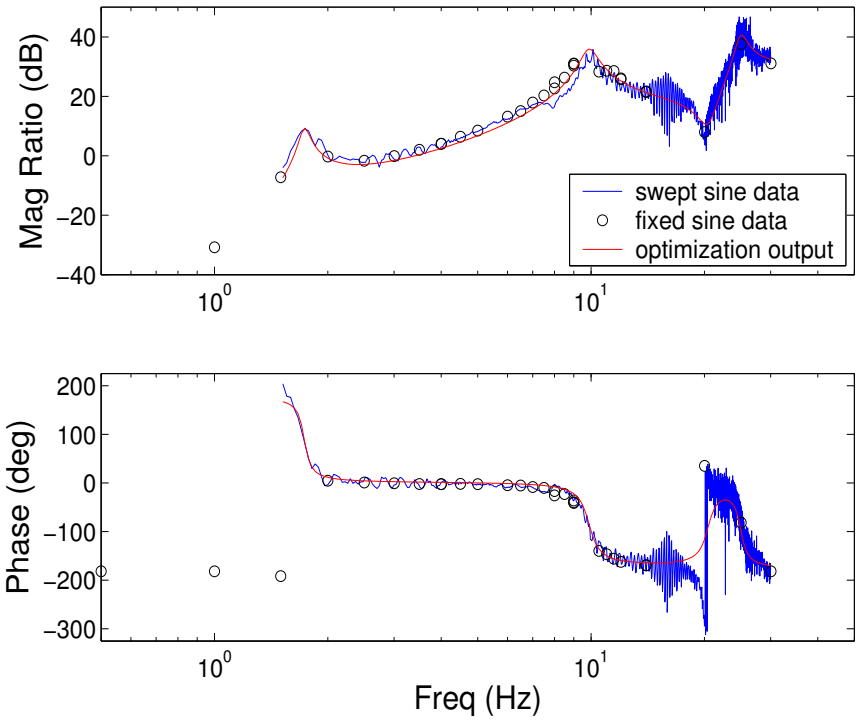

Figure 9.4: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration. The phase weight for this optimization was 0.1.

The output coefficent values were

$$\bar{x} = \begin{bmatrix} 27.8962 \\ 55.2835 \\ 0.139813 \\ 62.0634 \\ 0.0486851 \\ 158.869 \\ 10.8862 \\ 0.0298125 \\ 156.814 \\ 0.0336399 \\ -0.00146074 \\ 0.00156505 \\ 0.000280877 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ \omega_3 \\ \zeta_3 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{9.4}$$

These plots were generated by the Matlab file `main_bode_fit_11_14_03.m`, which calls the files `load_data_s4_m3_11_14_03.m`, `run_optim_s4_m3_11_14_03.m`, and `genbodeplots_11_14_03.m`. The optimization uses the error file `samiierr.m` which depends on the model file `samiimodel.m`. The curve fit results are written to the ascii file `bodefit_11_14_03_pw=0_1.txt`.

All of these files are in the folder `C:\DocumentsandSettings\ryan\MyDocuments\GT\Research\SAMII\curve_fitting\Nov03`.

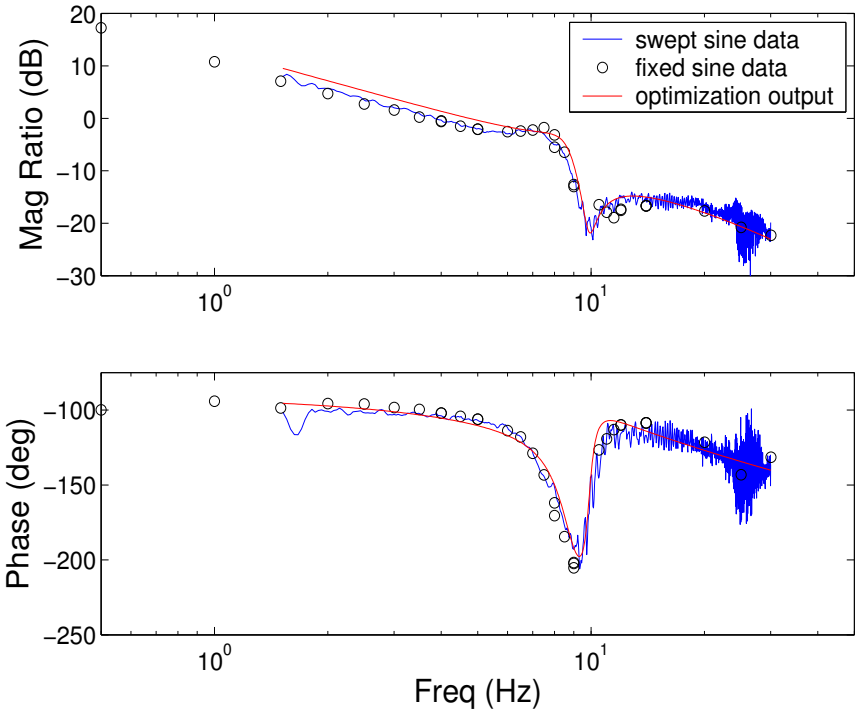

Figure 9.5: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position. The phase weight for this optimization was 0.2.

9.4 Phase Weight=0.2

Figures 9.5 and 9.6 show the results of curve fitting SAMII's Bode data with a phase weight of 0.2.

Curve fitting criteria:

Curve fit set to loop 20 times. Each loop runs the Matlab optimization for a maximum of 500 iterations. The Krauss convergence criteria was set to a maximum percent change in any coeffiecent of 0.01 and an error function maximum percent change of 0.01. These criteria needed to be met for 5 consecutive loops. (The percent changes refer to changes in the values at the end of one loop versus the values at the end of the previous loop where each loop represents 500 iterations.)

Met Matlab convergence criteria. The percent error change on the last loop was -1.22381e-

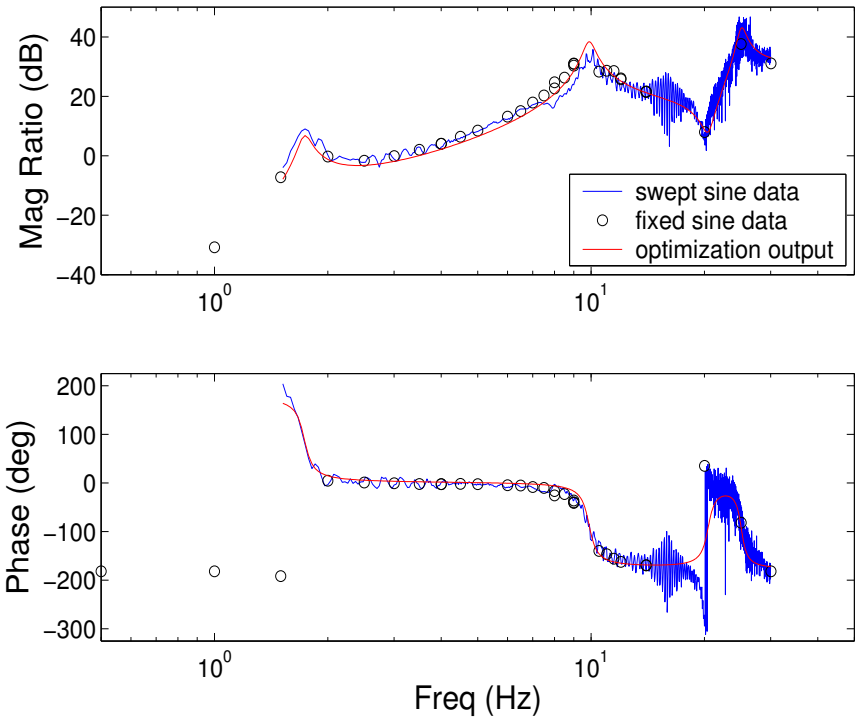

Figure 9.6: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration. The phase weight for this optimization was 0.2.

008% and the percent change in the coefficents was

$$\%\Delta x = \begin{bmatrix} -0.000597686 \\ -3.59537e-005 \\ -0.00116798 \\ -7.01124e-005 \\ 0.000828878 \\ 0.000139129 \\ -3.03098e-005 \\ -0.00251684 \\ -5.67693e-005 \\ -0.000785406 \\ 3.36303e-005 \\ 0.000319589 \\ 0.000429127 \end{bmatrix} \tag{9.5}$$

The output coefficent values were

$$\bar{x} = \begin{bmatrix} 28.4998 \\ 55.2627 \\ 0.113717 \\ 62.0611 \\ 0.035692 \\ 143.207 \\ 10.8871 \\ 0.0382545 \\ 157.501 \\ 0.0258995 \\ -0.00140709 \\ 0.00153227 \\ 0.000291275 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ \omega_3 \\ \zeta_3 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{9.6}$$

These plots were generated by the Matlab file `main_bode_fit_11_14_03.m`, which calls the files `load_data_s4_m3_11_14_03.m`, `run_optim_s4_m3_11_14_03.m`, and `genbodeplots_11_14_03.m`. The optimization uses the error file `samiierr.m` which depends on the model file `samiimodel.m`. The curve fit results are written to the ascii file `bodefit_11_14_03_pw=0_2.txt`.

All of these files are in the folder `C:\DocumentsandSettings\ryan\MyDocuments\GT\Research\SAMII\curve_fitting\Nov03`.

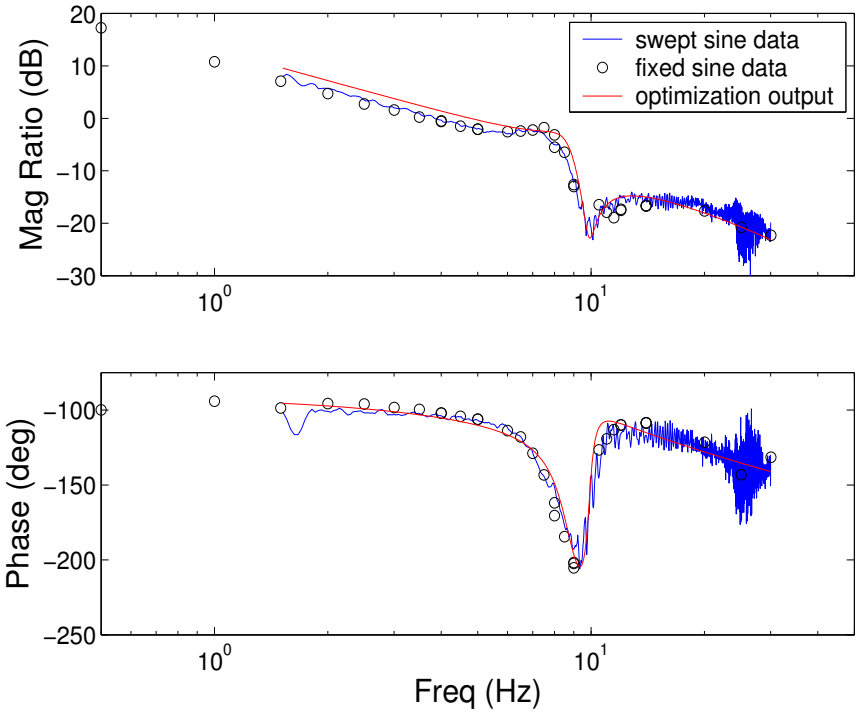

Figure 9.7: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position. The phase weight for this optimization was 0.3.

9.5 Phase Weight=0.3

Figures 9.7 and 9.8 show the results of curve fitting SAMII's Bode data with a phase weight of 0.3.

Curve fitting criteria:

Curve fit set to loop 20 times. Each loop runs the Matlab optimization for a maximum of 500 iterations. The Krauss convergence criteria was set to a maximum percent change in any coeffiecent of 0.01 and an error function maximum percent change of 0.01. These criteria needed to be met for 5 consecutive loops. (The percent changes refer to changes in the values at the end of one loop versus the values at the end of the previous loop where each loop represents 500 iterations.)

Met Matlab convergence criteria. The percent error change on the last loop was -2.78414e-

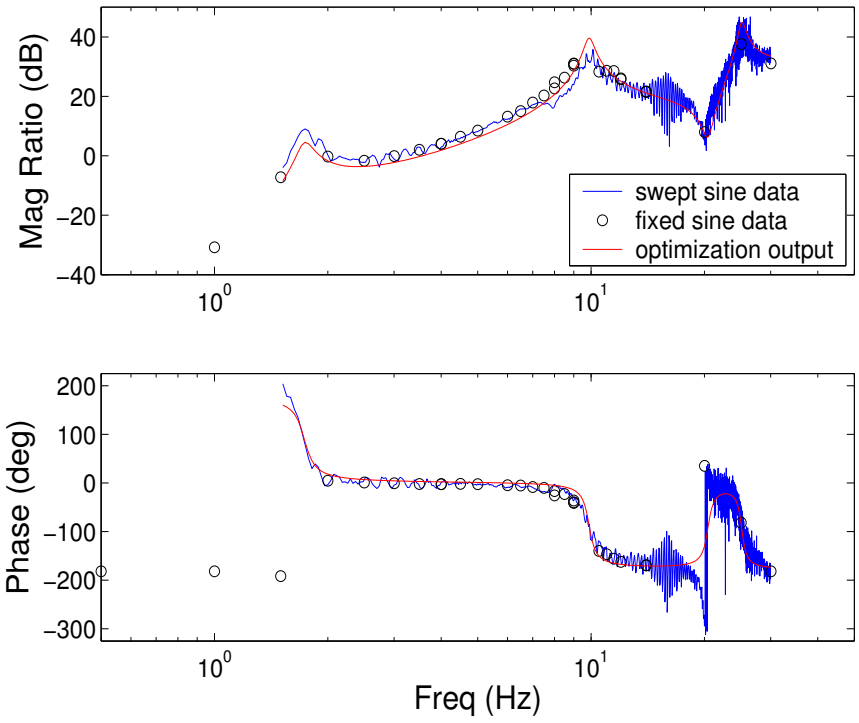

Figure 9.8: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration. The phase weight for this optimization was 0.3.

009% and the percent change in the coefficents was

$$\%\Delta x = \begin{bmatrix} 0.000224966 \\ 3.80211e-005 \\ 0.000826888 \\ 3.40805e-005 \\ 0.000671968 \\ -0.00055353 \\ 9.18664e-005 \\ 0.000327665 \\ 5.17764e-006 \\ -1.6552e-005 \\ -0.000180637 \\ -8.92491e-005 \\ 0.000213391 \end{bmatrix} \tag{9.7}$$

The output coefficent values were

$$\bar{x} = \begin{bmatrix} 28.6391 \\ 55.2783 \\ 0.103414 \\ 62.0692 \\ 0.0303846 \\ 139.245 \\ 10.8884 \\ 0.047849 \\ 157.829 \\ 0.0219657 \\ -0.00135041 \\ 0.0014989 \\ 0.000303583 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ \omega_3 \\ \zeta_3 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{9.8}$$

These plots were generated by the Matlab file `main_bode_fit_11_14_03.m`, which calls the files `load_data_s4_m3_11_14_03.m`, `run_optim_s4_m3_11_14_03.m`, and `genbodeplots_11_14_03.m`. The optimization uses the error file `samiierr.m` which depends on the model file `samiimodel.m`. The curve fit results are written to the ascii file `bodefit_11_14_03_pw=0_3.txt`.

All of these files are in the folder `C:\DocumentsandSettings\ryan\MyDocuments\GT\Research\SAMII\curve_fitting\Nov03`.

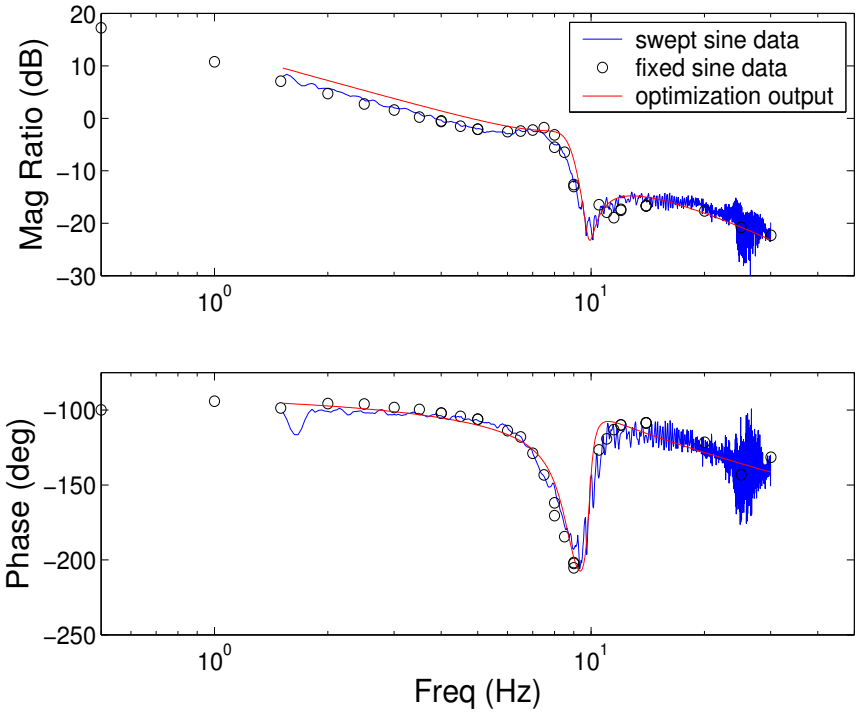

Figure 9.9: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position. The phase weight for this optimization was 0.4.

9.6 Phase Weight=0.4

Figures 9.9 and 9.10 show the results of curve fitting SAMII's Bode data with a phase weight of 0.4.

Curve fitting criteria:

Curve fit set to loop 20 times. Each loop runs the Matlab optimization for a maximum of 500 iterations. The Krauss convergence criteria was set to a maximum percent change in any coeffiecent of 0.01 and an error function maximum percent change of 0.01. These criteria needed to be met for 5 consecutive loops. (The percent changes refer to changes in the values at the end of one loop versus the values at the end of the previous loop where each loop represents 500 iterations.)

Met Matlab convergence criteria. The percent error change on the last loop was -8.49406e-

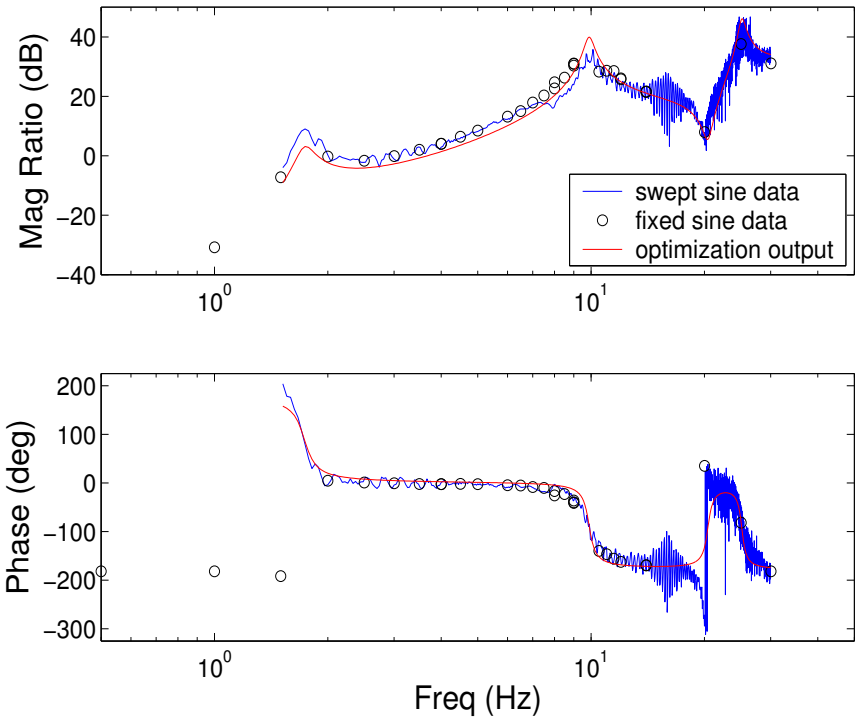

Figure 9.10: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration. The phase weight for this optimization was 0.4.

009% and the percent change in the coefficents was

$$\%\Delta x = \begin{bmatrix} -0.000311367 \\ -7.3114e-005 \\ -0.00155954 \\ -2.21957e-005 \\ -0.000444943 \\ 0.000691911 \\ -0.00010933 \\ -0.00139417 \\ 1.49292e-005 \\ -0.00173694 \\ 2.70344e-006 \\ 0.00025895 \\ 0.000997821 \end{bmatrix} \tag{9.9}$$

The output coefficent values were

$$\bar{x} = \begin{bmatrix} 28.6947 \\ 55.2662 \\ 0.0992186 \\ 62.0713 \\ 0.0284038 \\ 137.869 \\ 10.8883 \\ 0.0533673 \\ 157.982 \\ 0.0199391 \\ -0.0012746 \\ 0.00145774 \\ 0.000318233 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} K_1 \\ \omega_p \\ \zeta_p \\ \omega_z \\ \zeta_z \\ \tau \\ \omega_1 \\ \zeta_1 \\ \omega_3 \\ \zeta_3 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{9.10}$$

These plots were generated by the Matlab file `main_bode_fit_11_14_03.m`, which calls the files `load_data_s4_m3_11_14_03.m`, `run_optim_s4_m3_11_14_03.m`, and `genbodeplots_11_14_03.m`. The optimization uses the error file `samiierr.m` which depends on the model file `samiimodel.m`. The curve fit results are written to the ascii file `bodefit_11_14_03_pw=0_4.txt`.

All of these files are in the folder `C:\DocumentsandSettings\ryan\MyDocuments\GT\Research\SAMII\curve_fitting\Nov03`.

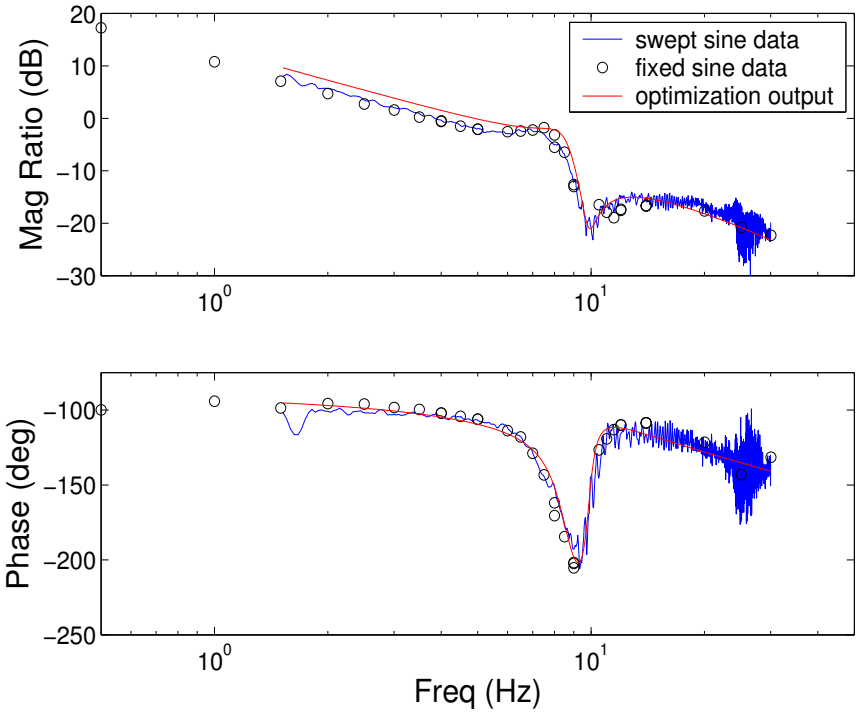

Figure 9.11: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position. The phase weight for this optimization was 0.5.

9.7 Phase Weight=0.5

Figures 9.11 and 9.12 show the results of curve fitting SAMII's Bode data with a phase weight of 0.5.

Curve fitting criteria:

Curve fit set to loop 20 times. Each loop runs the Matlab optimization for a maximum of 300 iterations. The Krauss convergence criteria was set to a maximum percent change in any coeffiecent of 0.01 and an error function maximum percent change of 0.01. These criteria needed to be met for 5 consecutive loops. (The percent changes refer to changes in the values at the end of one loop versus the values at the end of the previous loop where each loop represents 300 iterations.)

Did not meet either the Krauss or the Matlab convergence criteria. The percent error change

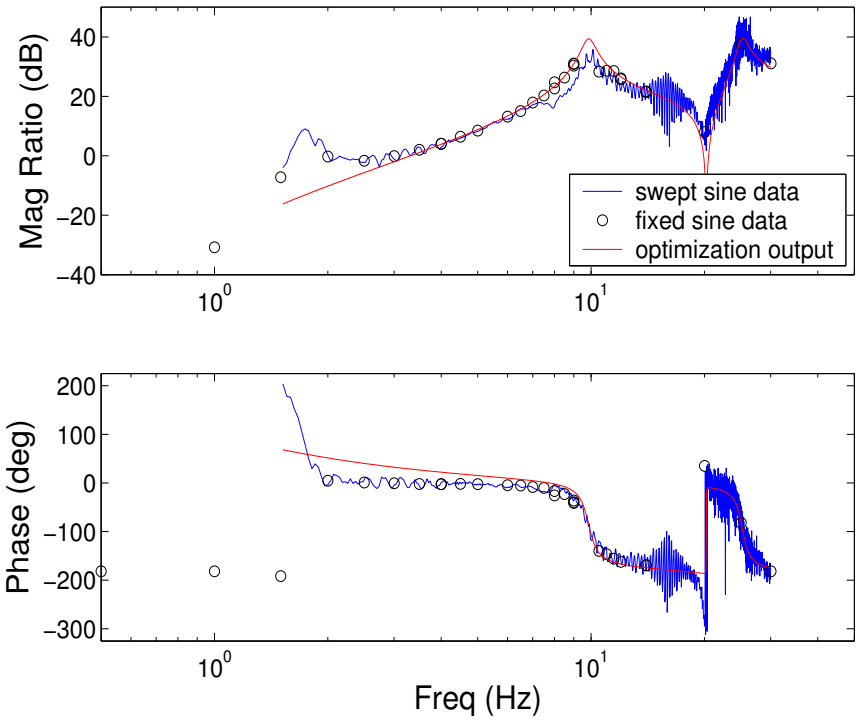

Figure 9.12: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration. The phase weight for this optimization was 0.5.

on the last loop was -0.0001149% and the percent change in the coefficents was

$$\%\Delta x = \begin{bmatrix} -0.0296606 \\ -0.00466991 \\ -0.152182 \\ -0.00983466 \\ 0.000984358 \\ -0.0053648 \\ 0.0435498 \\ -0.00604467 \\ 0.00255982 \\ 0.212486 \\ -0.00446356 \\ -0.00490934 \\ -0.0201934 \end{bmatrix} \tag{9.11}$$

The output coefficent values were

$$\bar{x} = \begin{bmatrix} 28.8249 \\ 54.3453 \\ 0.104566 \\ 61.8723 \\ 0.0406231 \\ 143.094 \\ 7.08669 \\ 0.817514 \\ 158.322 \\ 0.0325469 \\ -0.00218181 \\ 0.00199658 \\ 0.000247715 \end{bmatrix} \quad \text{where} \quad \bar{x} = \begin{bmatrix} k1 \\ wp \\ zp1 \\ wz \\ zz \\ tau \\ w1 \\ z1 \\ w3 \\ z3 \\ B1 \\ B2 \\ B3 \end{bmatrix} \tag{9.12}$$

These plots were generated by the Matlab file `main_bode_fit_11_14_03.m`, which calls the files `load_data_s4_m3_11_14_03.m`, `run_optim_s4_m3_11_14_03.m`, and `genbodeplots_11_14_03.m`. The optimization uses the error file `samiierr.m` which depends on the model file `samiimodel.m`. The curve fit results are written to the ascii file `bodefit_11_14_03_pw=0_5.txt`.

All of these files are in the folder `C:\DocumentsandSettings\ryan\MyDocuments\GT\Research\SAMII\curve_fitting\Nov03`.

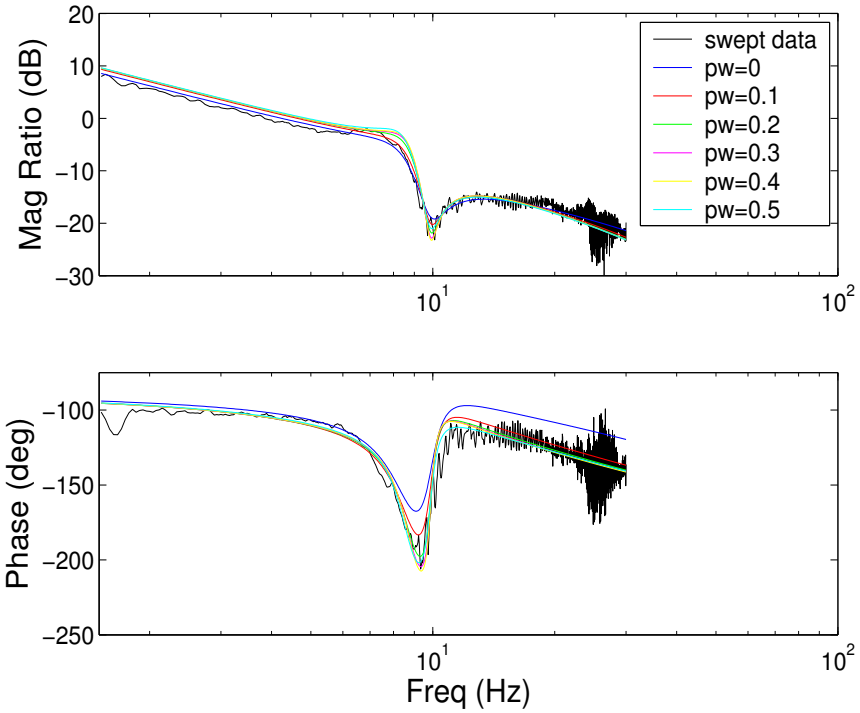

Figure 9.13: Comparison of bode plots from model and experimental data for the hydraulic actuator. The input is the voltage into the servo-valve of joint 2. The output is joint 2 angular position.

9.8 Overlay of All Phase Weights

Figures 9.13 and 9.14 overlay the results of curve fitting SAMII's Bode data with various phase weights.

Figure 9.14 shows that as phase weight is increased, amplitude error near the first mode (1.7Hz) is traded off against phase error near the zero between the second and third modes (20Hz). For the case of phase weight=0.5, the first mode is completely missed while the phase near 20Hz is matched very well. I am going to use the parameter estimates from a phase weight of 0.1 as a starting point in my controller design.

H.1 Matlab Files Used

H.1.1 Main Matlab File

The main Matlab file is `main_bode_fit_11_14_03.m` This is the current (as of 11/24/03) main file for Bode curve fitting. It calls several other files to run an optimization that reduces the error between model and experimetnal bode curves. The error includes the squared sum

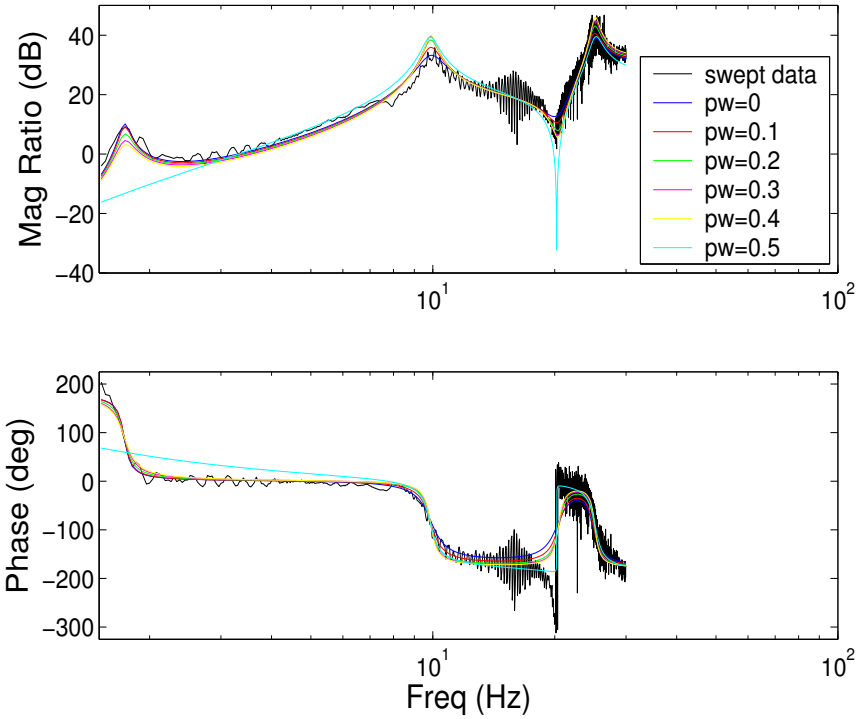

Figure 9.14: Comparison of bode plots from model and experimental data for the flexible base. The input is joint 2 angular position and the output is base acceleration.

of magnitude errors in dB and a weighting factor times the phase error in degrees. The model includes the first three modes of the flexible base.

This file calls the files `run_optim_s4_m3_11_14_03` and `genbodeplots_11_14_03.m`.

H.1.2 Additional Matlab Files

`curvefitparams_s4_m3.m`

As of 11/24/03, this function is not used in this directory.

`fixed_sine_bode_08_22_03.mat`

`genbodeplots_11_14_03.m`

This file generates the bode plots for one set of curve fitting results (i.e. one phase weight). This file is called by `main_bode_fit_11_14_03.m`.

`load_data_s4_m3_11_14_03.m`

This file loads experimental swept and fixed sine data. The swept sine data is loaded into global variables used by the curve fitting cost function. This file is called by `main_bode_fit_11_14_03.m` and `overlay_fits.m`.

`overlay_fits.m`

This file loads the results from curve fitting Bode data with various phase weights. It than overlays the Bode curves from the various weights to allow the best fit to be selected. To overlay the plots, it calls the function `overlaybodeplots_11_14_03.m`.

`overlaybodeplots_11_14_03.m`

This file has 3 inputs: starting figure index, coeffscell, and pwcell. The function overlays the bode plots from a cell array of curve fit output coeffiecents. The cell array of phase weigths (pwcell) is used only for the legend entries. This file is called by `overlay_fits.m` and is used to overlay the output of running `main_bode_fit_11_14_03.m` multiple times with different phase weights.

`run_optim_s4_m3_11_14_03.m`

This function runs the Matlab function fminsearch for a given set of optmization paramters (maximum nmber of iterations, etc.) and then outputs the results to a text file. This function is called by `main_bode_fit_11_14_03.m` and uses the file `samiierr.m` as the cost function for fminsearch.

`samiierr.m`

This function defines the cost function for the curve fit. The cost function is the sum of the squared magnitude error in dB plus a phase weighting term (scalar) times the sum of the squared phase error in degrees.

errout=samiierr(coeffsin,phaseweight)

The first input (coeffsin) is a vector of input coeffiecients. The second input is the phase weight.

This function is called by `run_optim_s4_m3_11_14_04.m` and calls the function `samiimodel.m`, which outputs the bode magnitudes and phases for both the actuator and flexible base models based on the vector of input coeffiecents.

`samiimodel.m`

This function is called by `samiierror.m`. This function takes a vector of coeffiecients as an input and outputs the bode magnitudes and phases for both the actuator and flexible base models.

coeffsin=varargin{1};

varargout{1}=act_fit_mag;
varargout{2}=base_fit_mag;
varargout{3}=act_fit_ph;
varargout{4}=base_fit_ph;

`swept_sine_bode_08_25_03.mat`

H.1.3 Verbatim Matlab Files

`main_bode_fit_s4_m3_editted_verb_apndx.pdf`

Chapter 10

Pole Canceling Controller Design

The basic idea behind this controller design is to cancel the lightly damped flexible system and replace them with more highly damped poles. A block diagram of the system is shown in Figure 10.1. This will be done by designing the pole canceling compensator shown in Figure 10.1.

One of the pairs of second order poles that need to be canceled are the closed loop poles of the inner feedback loop that controls the angular position of θ_2. This pair of poles is given by

$$cancel_{il} = \begin{bmatrix} -4.214 - 52.84i \\ -4.214 + 52.84i \end{bmatrix} \tag{10.1}$$

The other two pairs of second order poles that need to be canceled are from the first and third mode of the flexible base (the second mode poles get canceled by the numerator of the hydraulic actuator transfer function). These poles are given by

$$cancel_{fb1,3} = \begin{bmatrix} -0.3245 - 10.88i \\ -0.3245 + 10.88i \\ -5.275 - 156.7i \\ -5.275 + 156.7i \end{bmatrix} \tag{10.2}$$

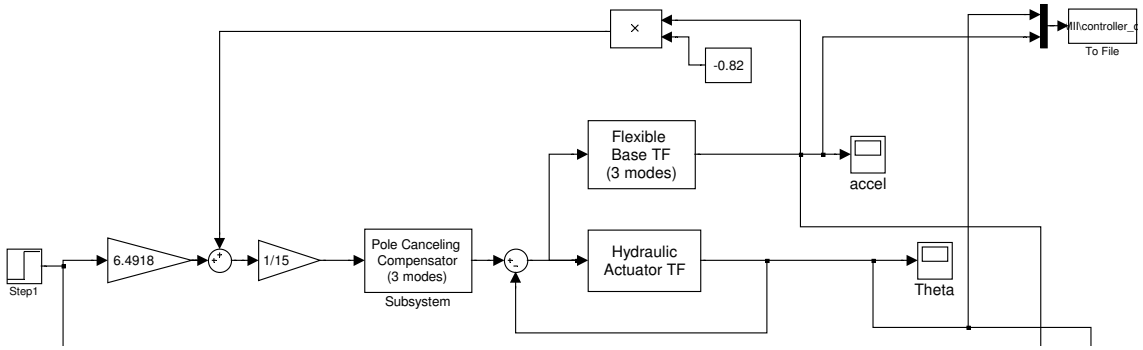

Figure 10.1: Desired vs. actual joint 2 angle for a controller with no vibration suppression.

The compensator will have these poles that we are seeking to cancel as its zeros and the desired replacement poles as its poles. The desired pole locations are

$$\begin{aligned} f_1 &= 3 \\ f_2 &= 8 \\ f_3 &= 10 \end{aligned} \tag{10.3}$$

(10.4)

where all of the f's, are in Hz and

$$\begin{aligned} \zeta_1 &= 1 \\ \zeta_2 &= 1 \\ \zeta_3 &= 1 \end{aligned} \tag{10.5}$$

(10.6)

These desired pole natural frequencies and damping ratios are turned into pole locations by using the relationship

$$d_p = -\zeta\omega \pm i\sqrt{1-\zeta^2} \tag{10.7}$$

These desired poles are given by

$$d_p = \begin{bmatrix} -18.85 \\ -18.85 \\ -50.27 \\ -50.27 \\ -62.83 \\ -62.83 \end{bmatrix} \tag{10.8}$$

The desired poles are specified to be critically damped, and then the overall gain of the compensator is chosen using root locus so that the closed loop poles having damping of approximately 0.7. The desired poles and zeros for the compensator are used to find the polynomials for the numerator and denominator of the compensator transfer function by using the Matlab function `poly`.

The compensator transfer function is given by

$$G_c = \frac{s^6 + 19.63s^5 + (2.762e+4)s^4 + (2.57e+5)s^3 + (7.251e+7)s^2 + (7.293e+7)s + (8.189e+9)}{s^6 + 263.9s^5 + (2.799e+4)s^4 + (1.515e+6)s^3 + (4.37e+7)s^2 + (6.299e+8)s + (3.544e+9)} \tag{10.9}$$

Figures 10.2 and 10.3 show the root locus for the system. These loci stop at the chosen compensator gain value of 0.0666. This value was chosen to place the closed loop system poles at locations with $\zeta \approx 0.7$. (Note that Figure 10.3 is simply zooming in on Figure 10.2 near the origin.)

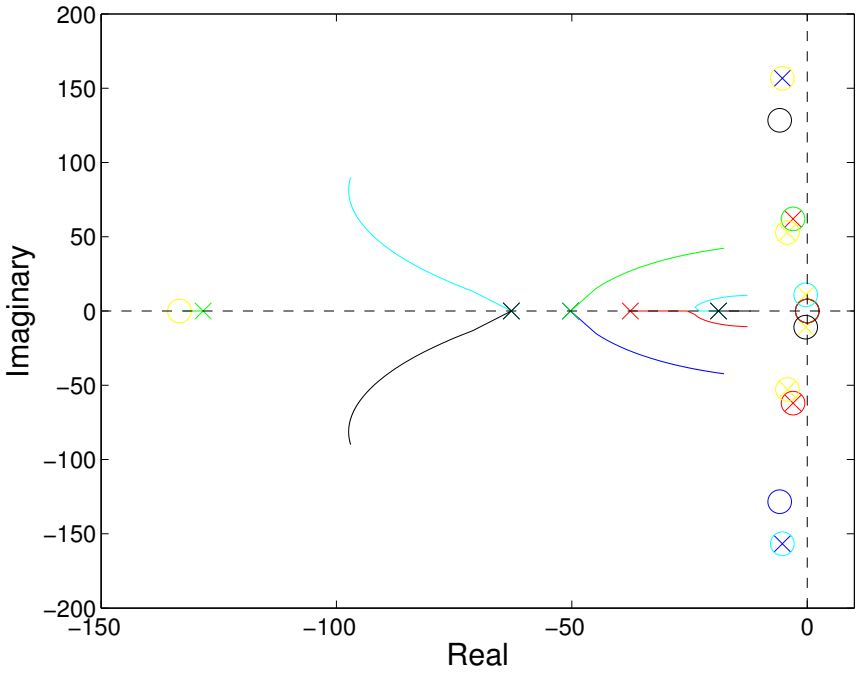

Figure 10.2: Root locus of the mass damping system without a lowpass filter or any other modifications.

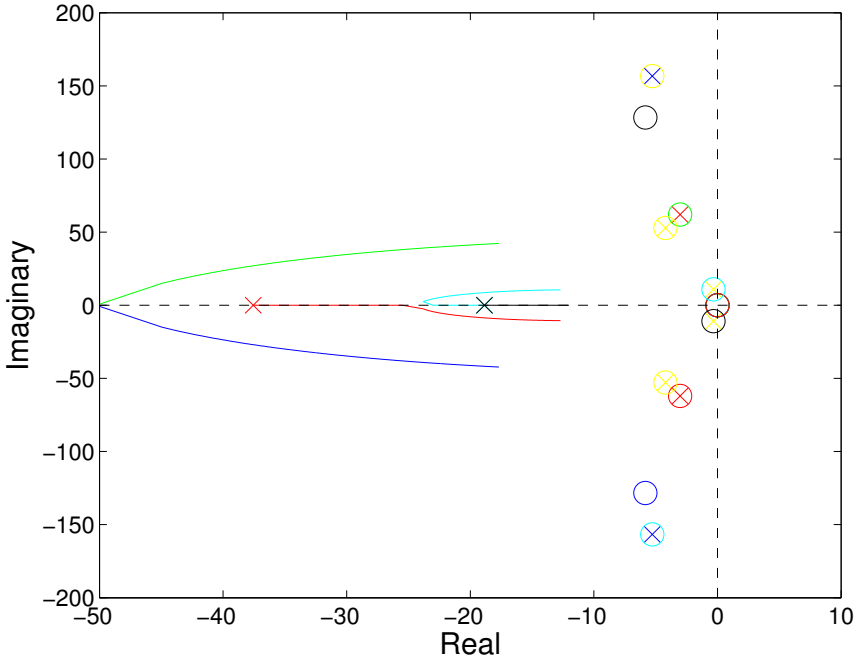

Figure 10.3: Root locus of the mass damping system without a lowpass filter or any other modifications (zooming in on Figure 10.2).

I.1 Matlab Files Used

I.1.1 Main Matlab File

The main Matlab file is `main_pole_cancellation_11_25_03.m` This function is the main function used to design a controller based on canceling the lightly damped poles of SAMII. The initial design will be based on canceling and placing the first two lightly damped poles in a model that includes the first three modes of the flexible base. If the third mode goes unstable as a result (I suspect it will), then the controller will be redesigned to cancel the third mode as well.

I.1.2 Additional Matlab Files

`pole_cancel_loci.m`

This function is very similar to the file `C:\ryan\GT\Research\SAMII\Ryan_SAMII_Wincon\system_id\without_force_torque_08_20_03\analysis\rlocus_pole_cancel_pseudo_s_4.m`. It will go further in that it will simulate the response with the third mode of the flexible base included in the model.

The basic idea is that the controller will cancel the lightly damped poles of the flexible base and replace them with poles with significantly higher damping.

I.1.3 Verbatim Matlab Files

`pcancel_design_editted_verb_apndx.pdf`

Chapter 11

Pole Canceling Controller Implementation

This chapter shows results from implementing the controller designed in Chapter 10. Simulation and experimental results are overlaid.

This controller does a good job of not exciting vibration by changing how a step change in desired angle is input into the system, but it is not capable of quickly damping out a vibration disturbance. In this sense, this controller seems to act like an input shaper.

Figure 11.5 shows the result of moving SAMII without the pole canceling compensator being used and then turning the pole canceling compensator on to see if it quickly damps the vibration. Obviously this didn't work very well.

Figure 11.6 shows that the simulation does a pretty good job of predicting the θ_2 response. Figure 11.7 shows that the simulation does a decent good job of predicting the acceleration resonse when viewed from the axis settings of Figure 11.4. Figure 11.8 shows that the simulation is not as good when the response is zoomed in on.

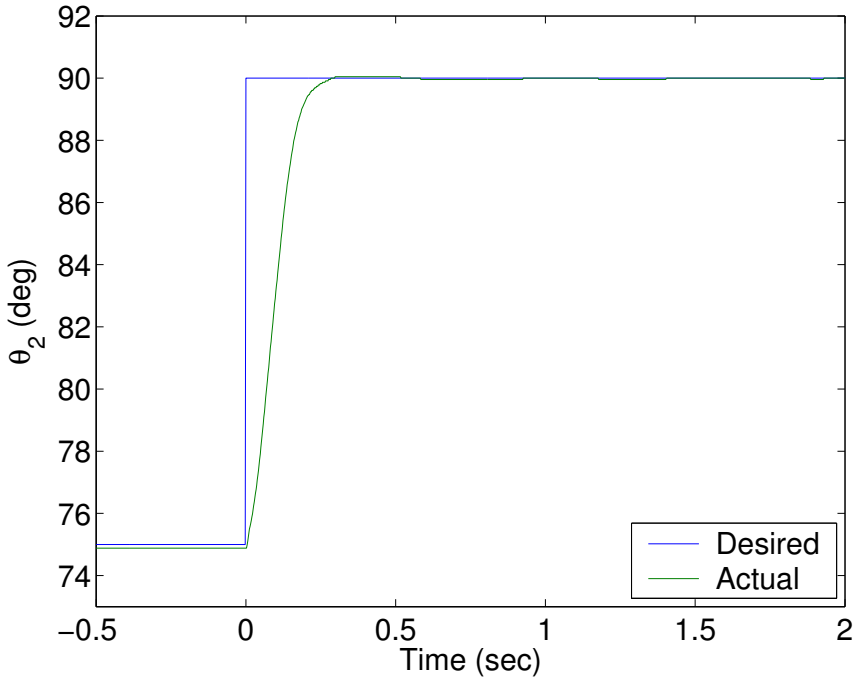

Figure 11.1: Desired vs. actual joint 2 angle for a controller with no vibration suppression.

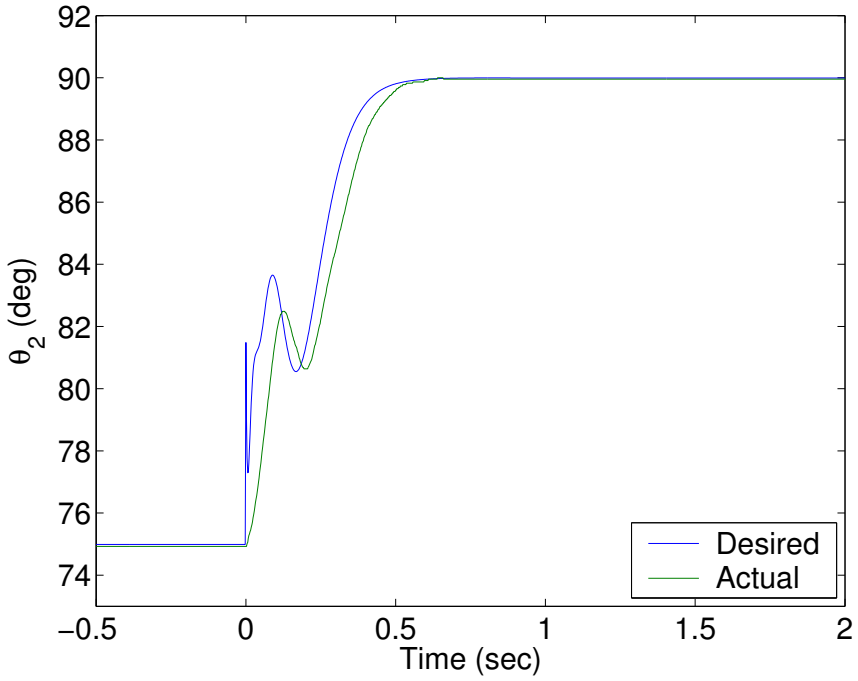

Figure 11.2: Desired vs. actual joint 2 angle for a controller with a pole canceling compensator.

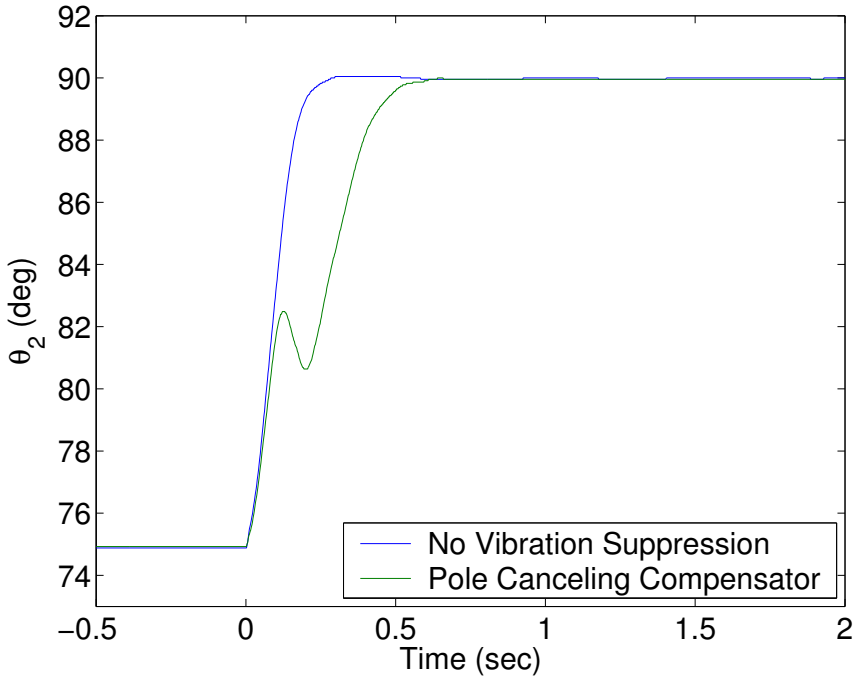

Figure 11.3: Actual joint 2 angle without vibration suppression vs. a controller with a pole canceling compensator.

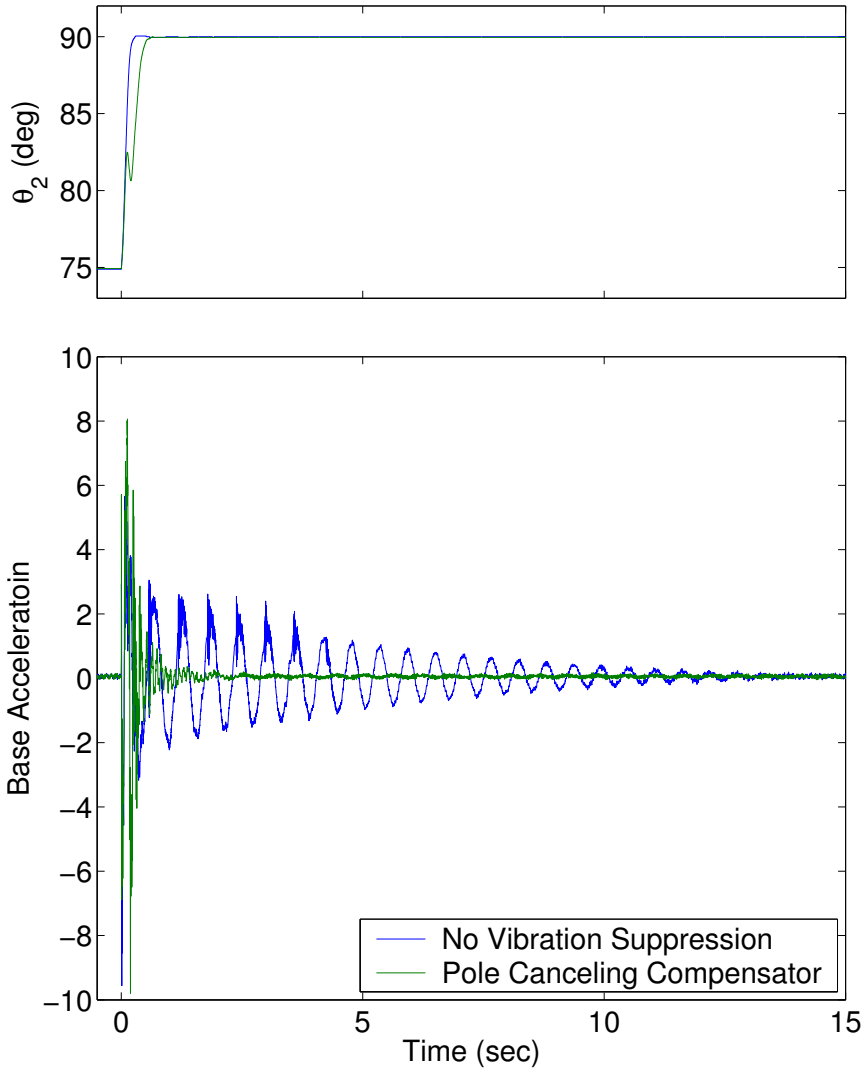

Figure 11.4: Base acceleration with and without a controller with a pole canceling compensator.

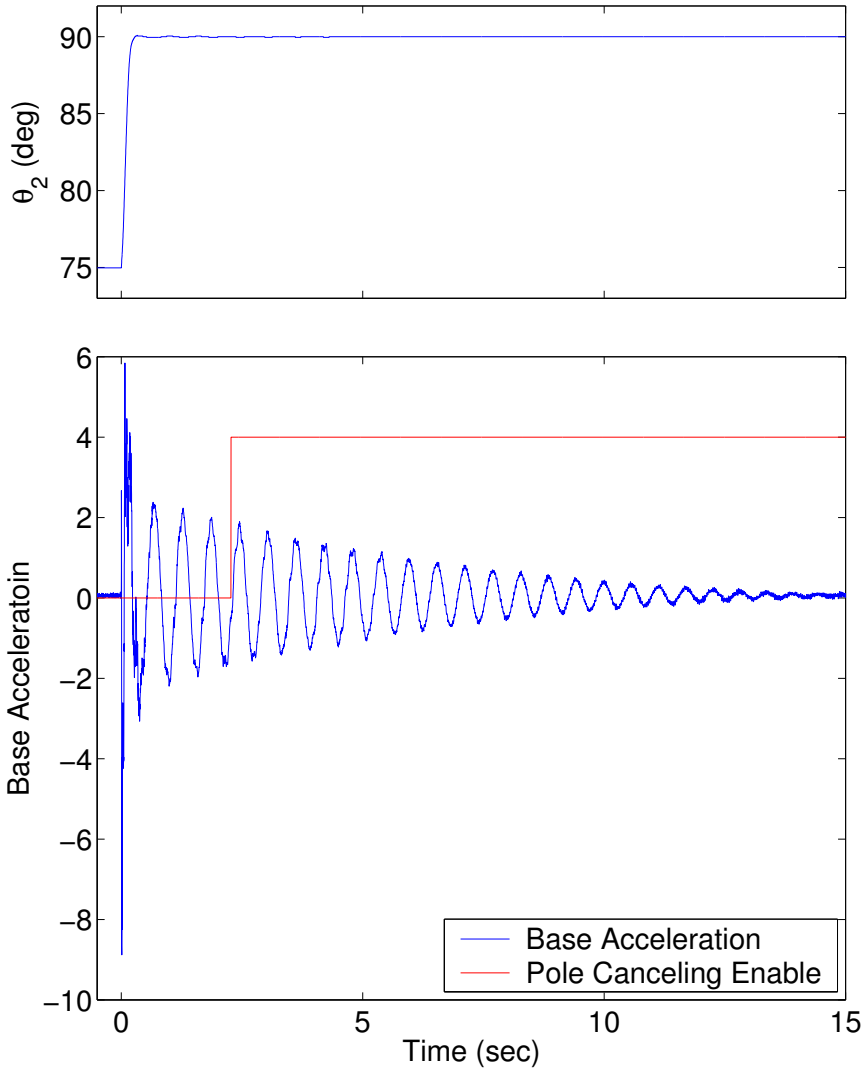

Figure 11.5: An unsuccessful attempt to recreate the Loper plot with the pole canceling controller.

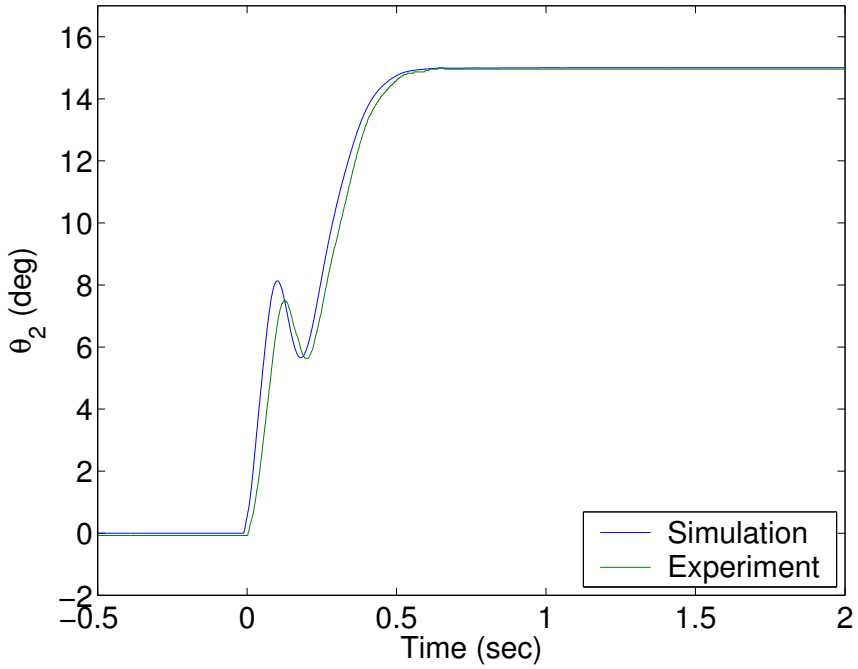

Figure 11.6: Simulated vs. experimental response for a system with a pole canceling compensator.

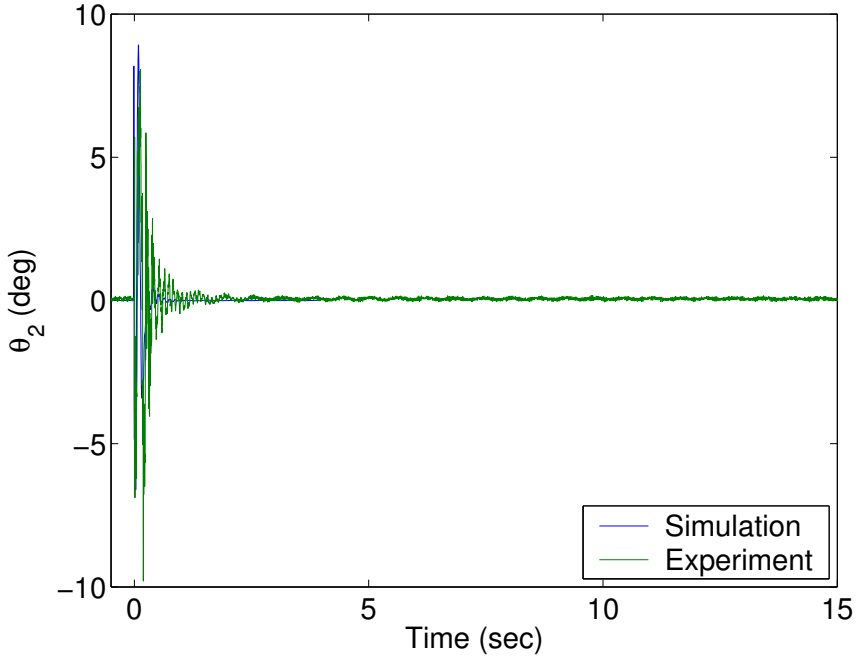

Figure 11.7: Simulated vs. experimental response for a system with a pole canceling compensator (zooming in). These are the same x-axis settings as shown in Figure 11.4.

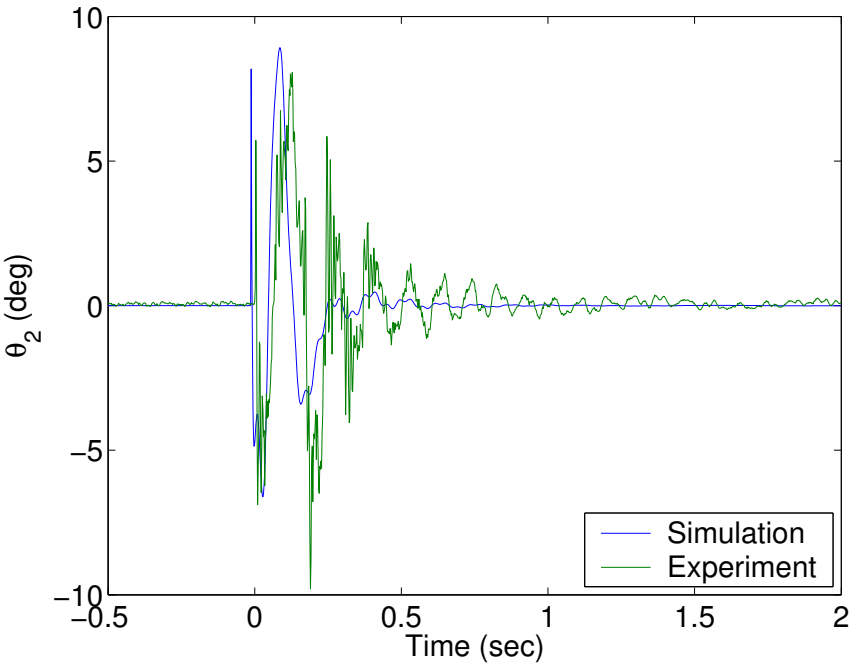

Figure 11.8: Simulated vs. experimental response for a system with a pole canceling compensator (zooming in). These are the same x-axis settings as shown in Figure 11.4.

J.1 Matlab Files Used

J.1.1 Main Matlab File

The main Matlab file is `load_pcancel.m` This function loads data from testing done with a pole canceling SISO controller on 11/26/03. It then overlays the data from testing done with and without this pole cancelation controller. It also plots data attempting to use this new controller to recreate the Loper plot where the vibration suppression controller is turned on and the vibration quickly goes away. This controller is essentially acting as input shaper, so the result is not as impressive as the plot from Loper's thesis.

The design of this controller is done in the Matlab file `main_pole_cancellation_11_25_03.m`, which calls the file `pole_cancel_loci.m`.

This file also overlays experimental and simulation data. The simulation data comes from the Simulink file `pole_cancel_m3_sim.mdl`.

J.1.2 Additional Matlab Files

`bigstep_with_pcancel.mat`

`compensator.mdl`

`loper_attempt.mat`

`pole_cancel_m3_sim.mdl`

`pole_cancel_switching_sim.mdl`

`simdata.mat`

`step_no_pcancel.mat`

`step_with_pcancel.mat`

J.1.3 Verbatim Matlab Files

`pcancel_overlay_verb_apndx.pdf`