

Interpolation of Second Order Dynamical Systems

by Ryan Ritch, mentored by Dr. Christopher Beattie

This semester, as an undergraduate research project, I worked with my mentor on coding and analyzing an algorithm for reducing second order MIMO system. The algorithm was taken from the article “Interpolatory Projection Methods for Structure-Preserving Model Reduction” by C. Beattie and S. Gugercin. It proved to be fairly challenging project for me, both from a mathematical and a programming perspective.

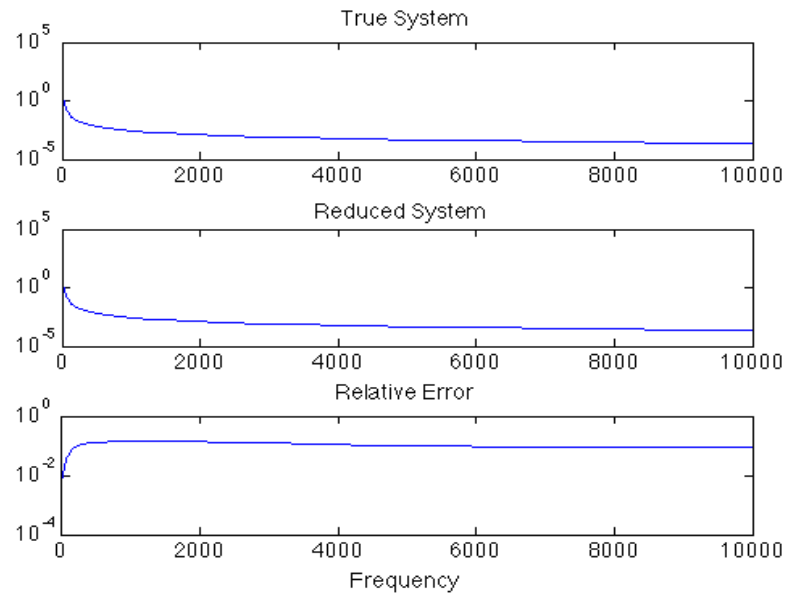
The project began with a fairly substantial learning curve for me. While I initially used MATLAB to code the algorithm from the paper without too much difficulty, to gain a basic understanding of what the algorithm did and how it worked, I had to do a significant amount of background learning. It took me several weeks both to learn about Laplace transforms and to become comfortable enough with the linear algebra that the algorithm used to proceed. Once I felt comfortable enough to proceed, I finished coding the algorithm and set up the appropriate conditions for it to run. As an input, I chose to use either the second-order system with exact condenser distribution from “Krylov-based Model Reduction of Second-Order Systems with Proportional Damping” once again by C. Beattie and S. Gugercin or a simple wave motion model. Aside from the matrices M , G , and K that defined the model itself, I chose most of the other parameters to be either random matrices or simple values (e.g. 1, 0.01).

No sooner than I had finished coding the algorithm, I encountered a difficulty that plagued me throughout the course of the semester. I discovered that unless I chose my input conditions very carefully, the algorithm would not full rank output matrices. Upon careful inspection, we discovered that this was due to floating point errors introduced by performing matrix operations on matrices with extremely small entries. We managed to correct this by orthogonalizing the reduction matrix V_r before reducing the model itself. This change corrected the problem in most cases, however the algorithm still occasionally failed to produce full rank outputs.

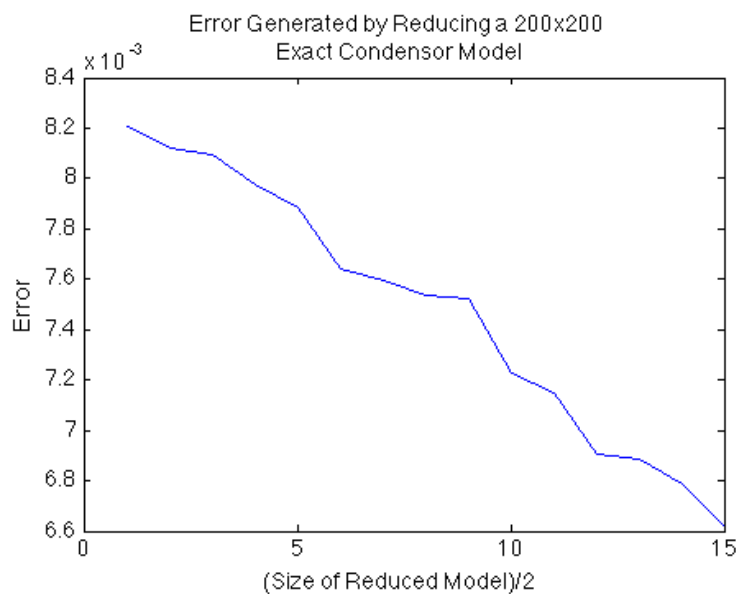
Once I had coded the algorithm and fixed any errors I could find, I began to test how effectively it functioned. I used two different methods. The first was Bode plots. While they were useful to compare the error between the original and the reduced system at different frequencies, they were slightly cumbersome to work with, especially when trying to compare a large number of different test conditions. For this reason, I coded a second method for analyzing error: state space models. By converting both the original and the reduced model to MATLAB state space models, I was able to find the relative error between the two as a single output number, making it quite easy to compare how well

the algorithm had performed over a large number of initial conditions.

For the most part, the algorithm performed fairly well and behaved as would be expected. The error that it generated was usually relatively small even when reducing fairly large systems (800x800) to relatively small systems (20x20). The following graph shows a Bode plot for reducing a 200x200 exact condenser system to a 20x20 system, as well as the error resulting from the reduction. As I



increased the size of the reduced system, the resulting error decreased in a fairly linear fashion. The algorithm exhibited this behavior over a wide range of initial conditions for both the exact condenser and the wave motion model. The following graph uses state space models to demonstrate this decrease in error.



Much of the rest of my time was spent attempting to improve the accuracy of the reduction by finding better guesses for the interpolation points used in the algorithm. When I initially coded the model, I chose values these points like $\{0.01, 0.02, \dots\}$ for lack of any better values to choose. My first attempt at refining these guesses involved deriving the interpolation points from the eigenvalues of a matrix generated by the M, G, and K matrices of the model I was reducing. I spent several weeks tweaking this code in various ways, but I eventually rejected it for two reasons. First, I began to have the same problem that I had initially encountered: the algorithm failed to generate full-rank reduced matrices. Second, even when it did perform as expected, the error that the reduction generated was usually extremely close to the error that was generated from my “initial guess” interpolation points.

I then embarked on a much more elaborate method of generating better interpolation points. My mentor showed me an iterative algorithm taken from the paper “ H_2 Model Reduction for Large-Scale Linear Dynamical Systems” by S. Gugercin, Dr. A. C. Antoulas, and C. Beattie that, given an initial set of interpolation points, was guaranteed to converge to the optimal interpolation points for a first order system. His idea was to take a second order system, reduce it using the algorithm that I had already coded, convert the second order model to a first order model, and then use IRKA, the new algorithm, to find the optimal interpolation points for this new system. His hope was that if enough iterations were run, this process would converge to the optimal interpolation points for the given second order model. Coding this process proved to be rather difficult, as it involved coding a new reduction method, IRKA, and then ensuring that it interfaced correctly with previously coded algorithm. While I managed to write all of the code, unfortunately, I was unable to completely debug it by the end of the semester. The code I have written currently fails at any number of places depending on the initial parameters: sometimes the second-order reduction algorithm fails to create full-rank matrices, sometime IRKA fails to converge, and sometimes the first iteration succeeds but somehow the interpolation points that it generates have an odd number of complex numbers which breaks the process. I believe that the problem lies somehow in my handling of complex conjugates somewhere in the process. It would be my hope that someone would pick up where I have left off and fix what I have started. I would be very curious to see if the process actually did generate ideal interpolation points for the second order model or if there were a mathematical flaw somewhere in the process that prevented it from working. If it were to function as we desired, it could potentially significantly increase the accuracy with which we were able to reduce second order models.

While it is unfortunate that I could not complete the final algorithm, I believe that I have accomplished a significant amount this semester. In addition to coding and extensively testing the second order model reduction algorithm, I believe that I made a good start on the more advanced

algorithm using IRKA. Equally importantly, I believe, I was exposed to a broad range of new mathematical topics that I would most likely not have otherwise seen during my undergraduate education. As such, I am grateful that I have had this experience to with real-world problems that are relevant to mathematics today. It is my hope that the work that I have done this semester will directly benefit my mentor in his research and perhaps eventually indirectly benefit the mathematical community as a whole.