**Project #3 - Numerical Methods - Spring ‘18**

*Created By: Liam Ayers & Clarisa Leu-Rodriguez*

*Group #2 - May 31st, 2018*

*Dr. Willett - Tacoma Community College*

**Purpose of Project:**

The purpose of this project is to estimate the position of a vehicle’s center of mass and the total distance traveled by the vehicle given simulated time and acceleration data of a vehicle’s suspension system. We will be using different methods of numerical modeling and numerical integration discussed in class. This project will be done using MATLAB.

**Background Information & Description:**

The job of a vehicle’s suspension system is to maximize the friction between the tries and the road surface. Suspension systems provide steering ability with good handling and ensure the comfort of passengers in the vehicle. The study of the forces at work on a moving car is called, “vehicle dynamics,” and is a rather complex field of Physics to study. The magnitude and quality of the transmission of force from a deformity on the ground through the vehicle depend on many factors, such as, tire pressure, width, design (suppleness), frame design (material, rigidity, geometry), and the suspension system itself (stiffness, travel).

In this project, we’re going to analyze given time and acceleration data from a vehicle’s suspension system and numerically estimate the position of the vehicle’s center of mass (which is the position of the system). We’re are also going to estimate the total distance that the center of mass moves. For this project, we’ll assume the initial velocity (V0) is -1.6 m/s and the initial position (S0) is 0 m.

For this project, we will be using three different methods of numerical modeling and we will be using the trapezoidal rule for all numerical integration.

**Methodology:**

Method One - Using the Trapezoidal Rule Directly on the Acceleration Data

As the second antiderivative of acceleration is position, for the first method, we used the trapezoidal rule directly on the given acceleration data twice to estimate the position data. The MATLAB implementation for trapezoidal numerical integration was found in Chapter Nineteen of the class textbook, as well as in the class code files on WAMAP.

This method was relatively easy to implement and required only a few lines of code. To delegate this task, we created a helper function called *trapHelper.m* which performed the trapezoidal rule twice on any given set of parameters.

Method Two - Using a Cubic Spline to Model the Given Data & Then Numerically Integrating

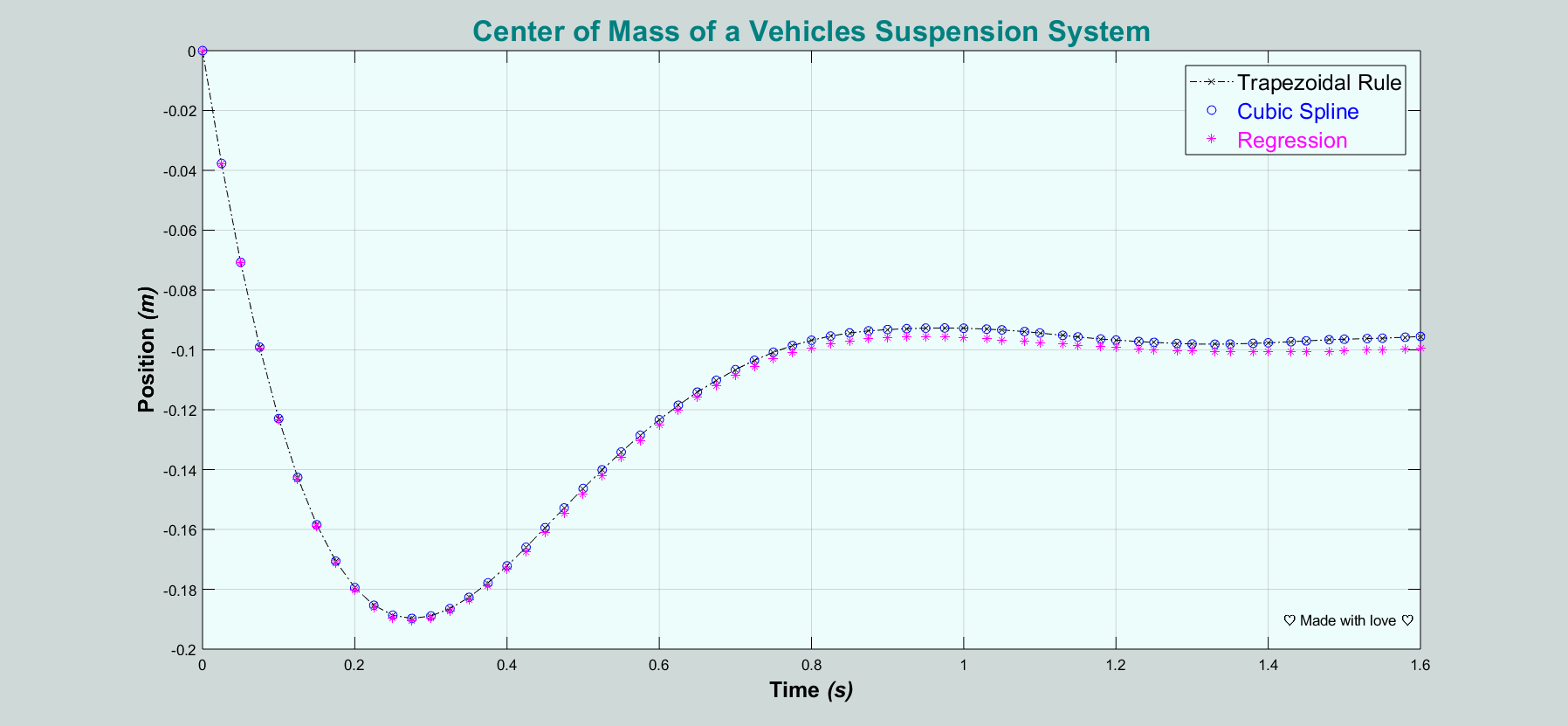
For the second method, we used the built in *spline()* command on the simulated data (*project3data.txt*) to create a cubic spline model. We then called our helper function *trapHelper.m* directly on our spline. This method was also relatively easy to implement and required only a few lines of code.

Method Three - Using Regression to Model the Given Data

Given that a damped oscillator should have an acceleration modeled by the equation: , we used regression on the data to determine the values constants A, B, C, and D. We then called our helper function *trapHelper.m* directly on our non-linear model. This method required creating another *.m* file to be used by the method *fminsearch(),* but was also relatively easy to implement.

Finding the Total Distance Traveled of the System

To find the total distance traveled of the system for each method, we used the same process of integrating the absolute value of the velocity data using *trapHelper.m*.

**Results:**

Graphical Analysis

The above graph contains the positions graphs for methods one through three. After analysis of the graph, we noticed that method one and two are essentially overlapping one another and don’t differ by much over time. This is because a spline will interpolate through all points in the data set, so it was expected that methods one and two would return essentially the same results. Method three does differ from the other two methods but not by much.

Comparing the Total Computed Distance Traveled by the System

Methods one and two produced the same total distance traveled of approximately 0.294931m. Method three produced a total distance traveled of approximately 0.291829m. This results in a relative error of about 1.05% compared to the first two methods. The standard deviation of the three different distances computed was 0.001791m, meaning that all three values were very close to each other.

Comparing Run Times & Determining the Most Efficient Method

On average, after ten trials:

* Method one had a run-time of approximately 0.0002654s.
* Method two had a run-time of approximately 0.0006542s.
* Method three had a run-time of approximately 0.006409s.

Method two took three times as long to run in comparison method one. Method three took ten times as long to run in comparison method two.

In conclusion, we felt that the first method (using the trapezoidal rule twice directly on the data) is the most efficient method as it was the easiest to implement (requiring only a few lines of code) and took the shortest amount of time to run. It was also very precise in comparison with the other methods, especially method two. The standard deviation of run times was approximately 0.0045s, meaning that there wasn’t a huge difference in the run times of each method and all were efficient in retrospect.

**Conclusion:**

The most efficient method was the first method (using the trapezoidal rule twice directly on the data) and didn’t differ much from method two (modeling the acceleration data with a spline). This was to be expected as a spline interpolates all points in a set. The average distance traveled by the system was approximately 0.29m with a standard deviation of 0.002m and the average run time was approximately 0.0047s with a standard deviation of 0.004s.

Our favorite part of this project was learning more about numerical integration and vehicle suspension systems. The most challenging aspect of this project was trying to determine which method was the most efficient as the true position data of the system was not known. However, to determine the best method we compared ease of set up and run times.