Numerical Methods

ECE 3040 – Fall 2021 – Dr. Abhilash Pandya

Alfarook Saleh

AccessID : gw5880 | Group # 10

Abstract

This project is about developing a GUI that can create a model of a pendulum and compare the data collected to the Theory of Pendulums. The GUI program can measure and save data from the swing of a pendulum, take necessary user inputs for functionality, and process and save that data through numerical methods to form a model.

Introduction

With today’s world constantly improving technology, it’s hard to find anything that isn’t automated in some way or form. Wherever there’s automation, there is likely the programming of code involved. Automation plays such an enormous role in our everyday lives that most people can’t live without it. Considering the grave importance of code, it is essential for an engineer to be able to understand, write, and use some level of code in order to be successful.

In this report, I will demonstrate my knowledge of the graphical user interface (GUI) my team and I developed by

* Explaining key functions of the GUI’s code that collect and process data through the Matrix Inverse, LU Decomposition, and Gauss Seidel numerical methods.
* Using the GUI to graph the swing of a pendulum, the derivative of the swing, a model of the swing, and comparing it to the Theory of Pendulums.
* Showing how the GUI will only accept correct user inputs through checks, and how the GUI changes accordingly to prevent user error.

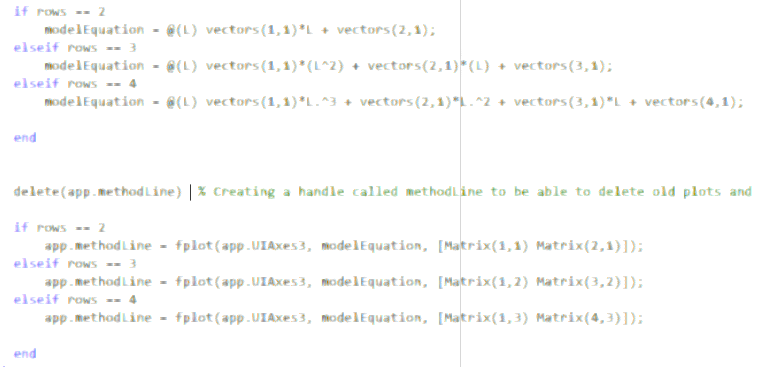
Problem Description

The goal of this project is to build off of Project 1 to design an interactive system that can automatically take and save distance and time data from a swinging pendulum of different lengths, and automatically calculate the data’s derivative and period. The system will then utilize user inputs to process the calculated data using the Matrix Inverse, LU Decomposition, and Gauss Seidel numerical methods to generate a 1st, 2nd, or 3rd order model of the pendulum swings. This generated model is then compared to the Theory of Pendulums and its error is calculated and displayed.

Numerical Solutions

First, I will describe how the program utilizes functions to calculate and graph the model of a pendulum swing. Our program is able to use three separate numerical methods to do this. The first function is the MatrixInverse function. It takes in a matrix and its corresponding right hand side (RHS) matrix, and outputs the equation of the calculated model and its coefficients.

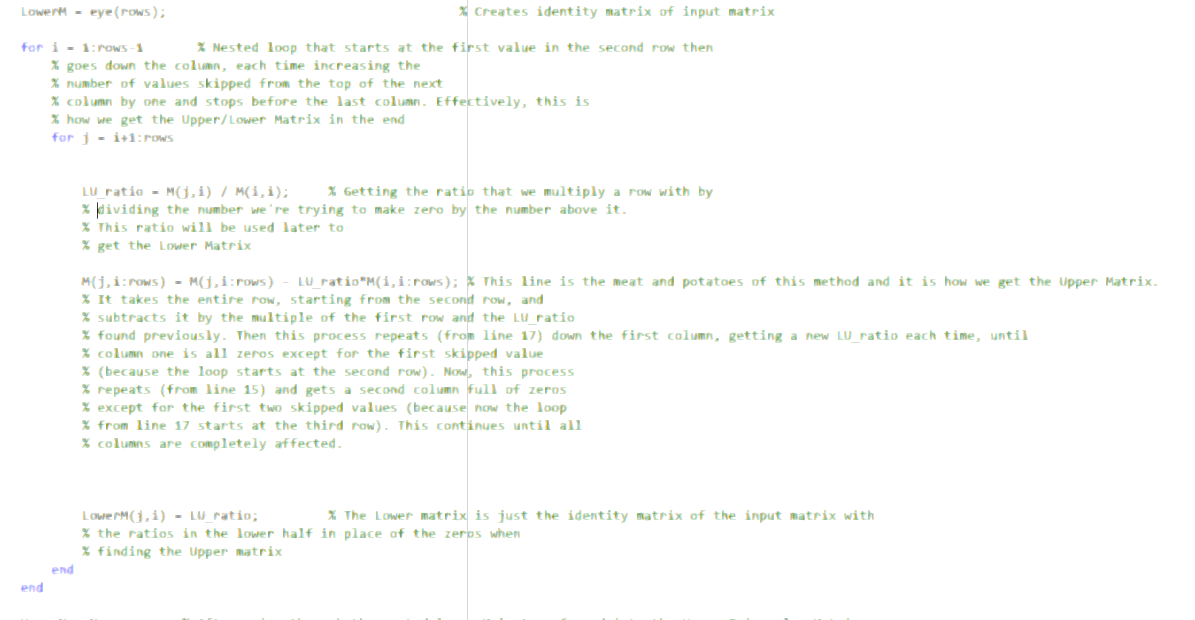
The code is simple enough. It reads the size of the matrix to ensure the matrix is square, otherwise it will stop the function. Then, the calculation method is simply one line of code that MATLAB excels at. The vectors matrix, A.K.A the coefficients, is simply equal to the inverse of the input matrix multiplied by the input RHS matrix. MATLAB has a built-in function that automatically finds the inverse of a matrix and can easily find the dot product of matrices. All that’s left is a simple if statement that controls the order of the equation for the model based on the number of rows of the inputted matrix. This if statement is repeated in the LU\_Decomposition function so the correct order for the model equation is used for each matrix. Also at the end of the MatrixInverse and LU\_Decomposition functions, is another if statement that graphs the correct effective range of the calculated model equation. This prevents the user from accidentally using ineffective data. Below is the code that is repeated that controls the order of the model equation and how it’s plotted.



Moving onto the LU\_Decomposition function, it also only takes in a matrix and its RHS, and checks to make sure the matrix is square by setting an error command if the rows and columns of the input matrix aren’t equal. This process is repeated across all numerical method functions to ensure the method functions properly.

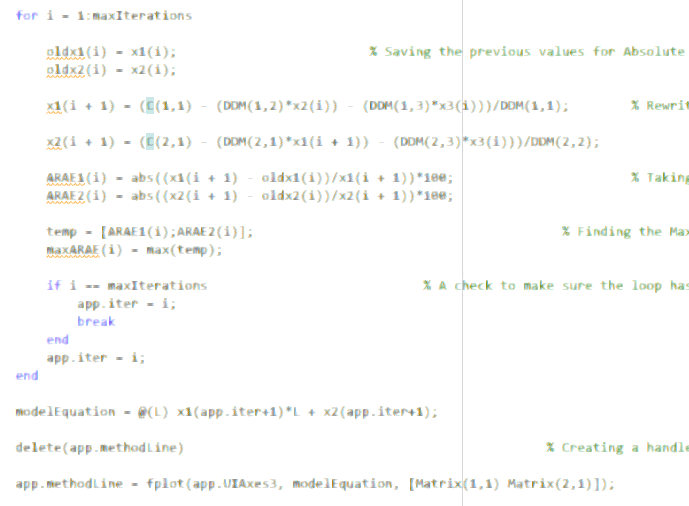
To solve for LU Decomposition, I start off by initializing an identity matrix equal to the size of the input matrix. This will become the lower triangular matrix after going through the function. Then, we utilize a nested loop that starts at the first value in the second row and goes down the column, each time increasing the number of values skipped from the top of the next column by one, and stops before the last column. The purpose of this top-down approach is to divide the lower numbers by the top number, record that specific ratio, multiply the entire row by the ratio, and subtract it by the top row. Then, the ratio is recorded into the identity matrix initialized earlier into its corresponding index for the lower triangular matrix. This process loops, each time skipping the next value in a column by one, and goes through all the columns except for the last one, thus creating an upper and lower triangular matrix. The lower matrix is made after all the LU ratios have been recorded and placed into the identity matrix. The upper matrix is made when subtracting a row by its LU ratio multiplied by the top level row.

Now that the lower and upper matrices are found, we can move onto some basic calculations using MATLAB’s inverse function that find the coefficients required for the model equation. It is here that we use the same if statements as shown above to correctly allocate the model equation and plot to the correct data. Below is the nested loop used for the LU\_Decomposition function as well as detailed comments.



Finally, we will talk about the Gauss Seidel method. This function actually utilizes another function inside of it that can automatically sort a given matrix and its RHS matrix into its diagonally dominant form IF it is possible to do so. This sorter function, called DDSorter, uses two nested loops, one for finding the absolute sum of the rows of a matrix, and another for finding the largest number in a row, testing for diagonal dominance, and swapping out the row with another row to sort out for the diagonally dominant matrix. It has its checks to make sure if its possible for a matrix to pass diagonal dominance. One such check is the check to see if there’s a number in a row that is larger than or equal to the absolute sum of the rest. If not, then diagonal dominance is impossible. A ‘count’ variable is incremented for every time a dominant number is found in a row. If the count is equal to the number of rows by the end of the loop, that means there is definitely a dominant number in every row and diagonal dominance is possible. Another check is in place to see if the dominant number is greater than the absolute sum of the rest of the values in the row. This check would indicate a strictly diagonally dominant matrix that is guaranteed to converge. By the end of the second nested loop, the input matrix and the RHS should be in it’s diagonally dominant form and is returned to the user if it passes the checks and the user is informed if it is strictly dominant or not.

Now to the actual Gauss Seidel method, it is actually three separate functions, each one increasing in the order of the model equation, as I couldn’t figure out how to make a single function that could accept any size square matrix. My issue came when trying input the DDSorted matrix and RHS into a universal equation that will increase or decrease in size according to the number of rows of the input matrix. Running out of time, I quickly copy pasted the Gauss Seidel function three times, each nearly identical except having modified lines to accept different level order matrices. The Gauss Seidel function take in a matrix, its RHS, a guess RHS matrix, and max iterations, and outputs the model equation calculated, the relative error, and the coeffecients of the model. The calculation for this function is pretty straightforward when not having to worry about different sized matrices. Following the Gauss Seidel method to the tee, I have a for loop that doesn’t go above the input max iterations, sets one side of the equation equal to the first coefficient and calculates that with the input guesses, repeats for all the coefficients, and finds the max absolute relative approximate error. When max iterations is reached, the model equation is generated with the coefficients found and plotted. Below is the code used for the 1st order Gauss Seidel method. Matrix C is the diagonally dominant RHS matrix found previously. Matrix DDM is, intuitively, diagonally dominant matrix.

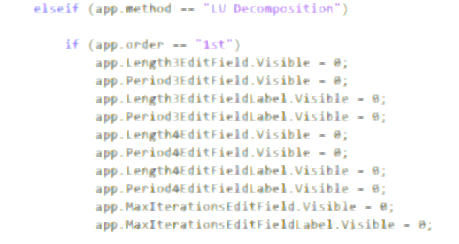


The GUI can also find the ideal period for a given length by using a dropdown bar to indicate what input length the user wants to use. It also gives the percent error of the calculated gravitational constant compared to the actual constant. This is simply done in essentially three lines of code shown below. Text

Description automatically generated

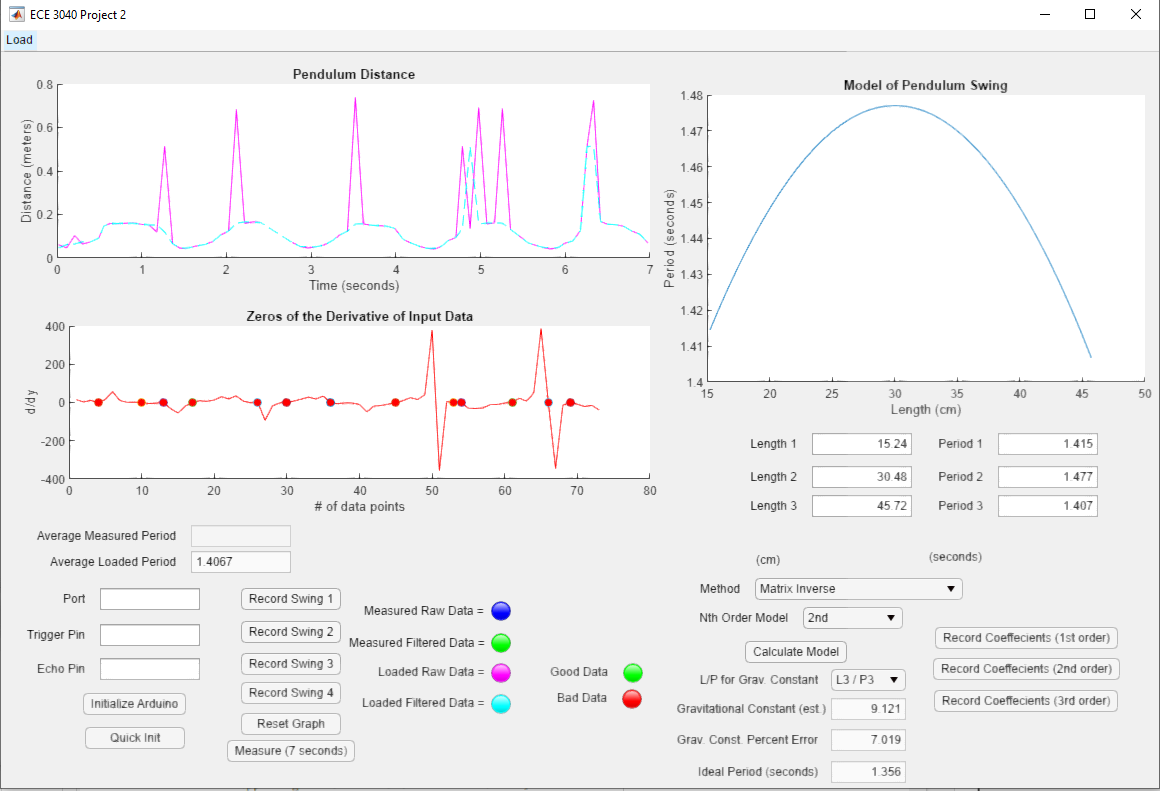
The first line calculates the gravitational constant by using the theory of pendulums but utilizing the user input data. Skipping the second line for ease of use, the third line then takes the error of the calculated constant by comparing it to 9.81, the actual gravitational constant, and displaying that in an edit field value in front of the user. The last line calculates the ideal period using the Theory of Pendulums formula and the inputted length in centimeters. This too is displayed in front of the user via edit field value.

Now that the calculations are out of the way, let’s discuss how the interface works. This part isn’t too complicated. It is mostly just functions set off by a button click or global variables being set to one or zero. Everything in the GUI can be referenced as a global variable prefixed by “app.”. Let’s take the method, order of the method, and input variables for example. Normally, there should be four lengths and four periods displayed for the user to input values into. However, this is only for 3rd order calculations. When going to 1st order and choosing a method that is not Gauss Seidel, the max iterations, length 3 and 4, and period 3 and 4 disappear by setting a value from one to zero. Below is the code that does specifically that.



Everything user interface related follows some sort of command as shown above.

Now let’s get into some GUI action. Using the load menu in the top left of the application, I can load previously measured data and it is automatically graphed in different colors and styles for the user to be able to differentiate the lines, and the period is automatically calculated and displayed below the graph. Additionally, the derivative is automatically calculated and its zeros are plotted for the user to easily distinguish where the slope changed from positive to negative. The user can then take those calculated periods and input them into the model equation maker alongside the corresponding lengths on the right side of the application. When the user clicks the calculate button, the selected order and numerical method will be used to generate the model of a pendulum in the graph above. Additionally, clicking a menu item from the dropdown list below the calculate button will display the corresponding length and period’s calculated gravitational constant, its percent error, and the ideal period for the given length. Below is an example of the GUI at work.



Conclusion

This report summarizes the main points to be taken from Project 2. It builds upon the previous GUI from Project 1 in order to be able to calculate a model equation, display it, and compare it to the model of the Theory of Pendulums. In order to do so, the program must be able to take length and period inputs and use one of the three numerical methods shown in this project to calculate and plot the model equation of the given parameters. After doing so, the user can compare their data to the Theory of Pendulums using the dropdown list near the bottom right of the application.

This project was especially insightful on how numerical methods can be applied to data in order to create a model to interpolate new data from. It also gives a glimpse of how code and software can be manipulated to serve the needs of an engineer. The knowledge I’ve learned from this project is cemented into my brain from how much effort was put into formulating and correcting the syntax to make this application work. I could absolutely see myself using the knowledge I’ve learned from this project in a professional setting.