EAS230 Engineering Computations
Fall 2024

Design and Analysis of a Tuning Fork

Group Name: MATLORDS
Partner #1: Nathan Jachlewski, Partner Section: B
Partner #2: William Dove, Partner Section: C

Part	**Description of the breakdown of tasks**	
	Partner # 1: 50%	Partner # 2: 50%
A	Completed Part A	
B		Completed Part B
C	Completed Numerical Solution	Completed Analytical Solution
D	Completed Part D	
E		Completed Part E
F	Split Part F	Split Part F

Table of Contents

List of Tables and Figures

1. **Summary/Abstract**

The objective of this project was to determine the deflection, or vibration, of a tuning fork at various points along its length over time. The program created can calculate deflection for tuning forks with variable geometries, materials, desired frequencies and desired octaves. The deflection is calculated using data for the frequencies of various notes, as well as the equations governing the Euler-Bernoulli Beam Theory. First, positions along the length of the arm and timesteps following an initial set of conditions were discretized into vector quantities of lengths M and N, respectively. An M by N matrix, W, was used to model the deflection as a function of space along the M dimension and time along the N dimension. First, the deflection was calculated analytically, using equations 6-9 to calculate $W(x_m, t_n)$ up to a specified time, *tmax*. Next, the deflection was additionally calculated numerically, using finite difference analysis. Given a set of initial conditions, equations 10-14 were used to determine the deflection for each point x_m one timestep at a time, based on the previous timestep. Finally, the results for each calculation method were displayed graphically and numerically given sample input conditions.

2. **Introduction**

This project aims to provide a manufacturer of musical instruments with a model for production of tuning forks. A tuning fork is a metal instrument designed to vibrate at a certain frequency when struck, and therefore the resulting pitch can be used to tune other instruments. The resonant frequency of a tuning fork depends on variables such as the material and dimensions of the instrument. These properties will affect the fork's density, polar moment, cross-sectional area and Young's modulus which all will alter the resonant frequency. To create the necessary model using the given information, a series of MATLAB scripts and functions were used to calculate the frequency, length, Young's modulus, density, polar moment of area, cross-sectional area, and cost of the tuning fork, as well as the displacement along the fork's length as a function of space and time. In the first script, necessary parameters were determined using input data. In the second and third scripts, the displacement was calculated both analytically and numerically using necessary parameters and equations. In the fourth and fifth scripts, the resulting data was displayed both graphically and numerically.

3. Description of code

Part A: TuningForkParams.m

- A function file that receives parameters for the desired note, octave, width, material, and shape of the tuning fork and outputs the material's frequency, length, Young's modulus, density, polar moment of area, cross-sectional area, and cost.
- Frequency, Young's modulus, density, and cost were gathered from a given table (materials.xlsx) and were converted as necessary to the correct units. Following this, length, polar moment, and cross-sectional area were calculated using the given equations (equation 1 and table 1). For each string input, a switch case structure was used to determine a numerical value for use in later calculations.

Part B: AnalyticalSolnParams.m

- To calculate the deflection of the tuning fork analytically, multiple coefficients and variables needed to be calculated beforehand.
- This function file takes inputs of the tuning fork's length, L, the desired number of terms in the summation given by equation 5, Q (this will affect accuracy; for the purposes of this project Q=12 terms was used), and the number of discrete distances contained in the vector **x,** M.
- Given this information, the file outputs values for z_k, β_k and φ_k using equations 7a and 7b as well as the function $f(z) = \cosh(z)\cos(z) + 1$.

Part C: TuningForkSoln.m

- A function file that creates a matrix, W, which contains information regarding the deflection of the tuning fork with respect to time.
- The function takes input parameters of length, Young's modulus, density, polar moment, cross-sectional area, total time, the number of elements in the distance of x, the number of elements in the time vector t, and the solver the user wants to select (analytical or numerical). The function then outputs a vector, **x,** representing the nodes along the length of the fork, a second vector, **t**, which represents the discrete time points in time, and the deflection matrix, W.
- There were two distinct ways of calculating the W matrix:
- **Analytical Solver:**
  - The analytical solver makes use of the Euler-Bernoulli Beam Theorem and the equation $W(\boldsymbol{x_m}, \boldsymbol{t}_n)$**,** given by equation 6, which models the deflection of a beam, W, at different points along the beam (represented by discrete distance intervals in column vector **x**,) and different instances in time (represented by discrete timestep intervals in row vector **t**).

  - To make use of this equation, multiple variables previously calculated in part B were used. The mode amplitudes, c_k, can be calculated through left division given the initial conditions of W (equation 5) and evaluating $W(\boldsymbol{x_m}, \boldsymbol{t_n})$ (equation 6) at $\boldsymbol{t_1} = 0$.
- **Numerical Solver:**
  - The numerical solution uses finite difference analysis and defines the system through partial differential equations. This method created a matrix of coefficients, A, such that A**x**=**b**, with **x** being a column of W, and **b** being a vector depending on the previous columns of W. This method calculates each column of the W matrix using left division of A\b (**b** depending on the previous columns of W, starting with the initial deflection at $\boldsymbol{t_1} = 0$). To create A and **b**, the given equations 10-14 were used. A for-loop structure was used to calculate the next column one at a time, changing the vector b to reflect the appropriate column of W and calculating the following column until the W matrix was complete.

Part D: PlotResults.m

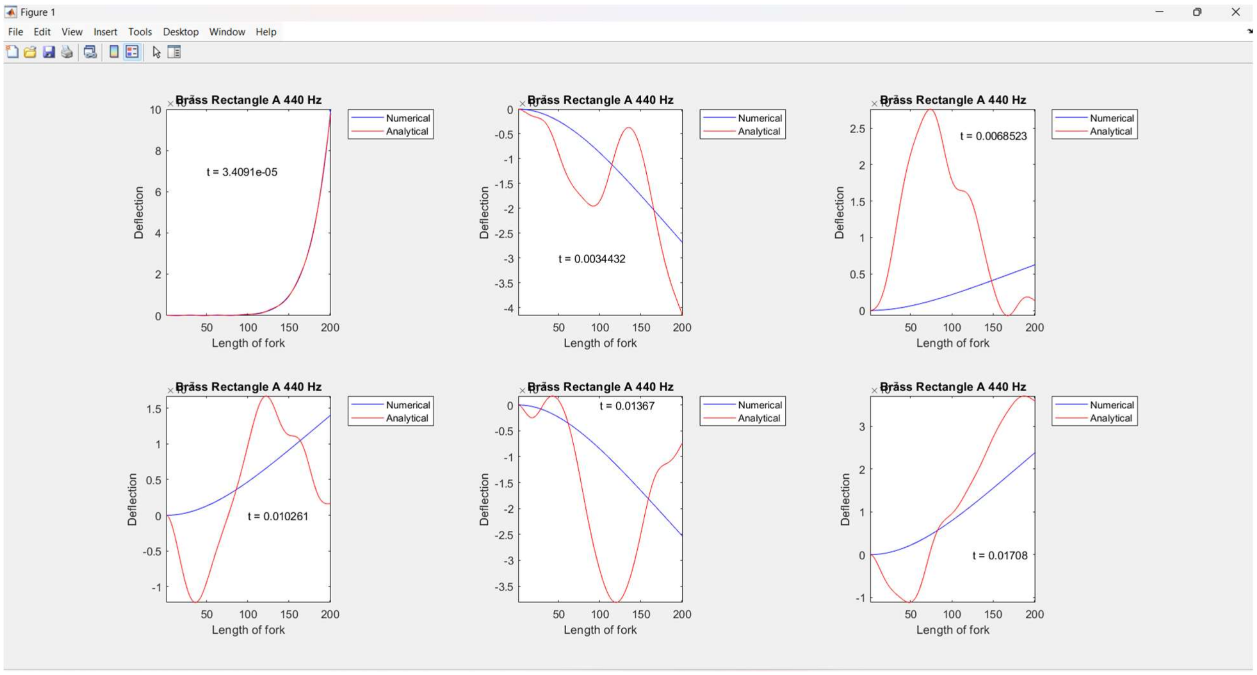
- This script file will create 4 figures, the first 3 figures each giving a visual about the deflection of the tuning fork and the 4th shows the relationship between material, octave, and cost. The files TuningForkSol.m along with TuningForkParams.m were used to determine the W matrix of deflection. Then depending on whether the plot was depicting a certain time or certain point on the tuning fork, the graph would plot a column or row of the W matrix. To create the correct axes, the minimum and maximum values of the graphs sampled, and the x and y limits were matched with them. Moreover, to create multiple subplots in one figure a for-loop structure was used where each iteration filled in another subplot.

Part E: PrintResults.m

- This script makes use of the function from part A to determine the frequency, cost and length of two different tuning forks for different octaves. The first tuning fork was a brass rectangular tuning fork with a width of 5mm, and a note of A. The second had the same conditions, but was designed for the note C.
- Given this information, a table was generated using the built-in MATLAB fprintf() function, determining that the cost of the first tuning fork was slightly higher than the cost of the second for all octaves, among other numerical data.
- This data was also saved to two different ACII .dat files using the MATLAB writematrix() function.

4. **Results**

Figure (1)



Both methods for calculating deflection were compared at different times.

Figure (2)

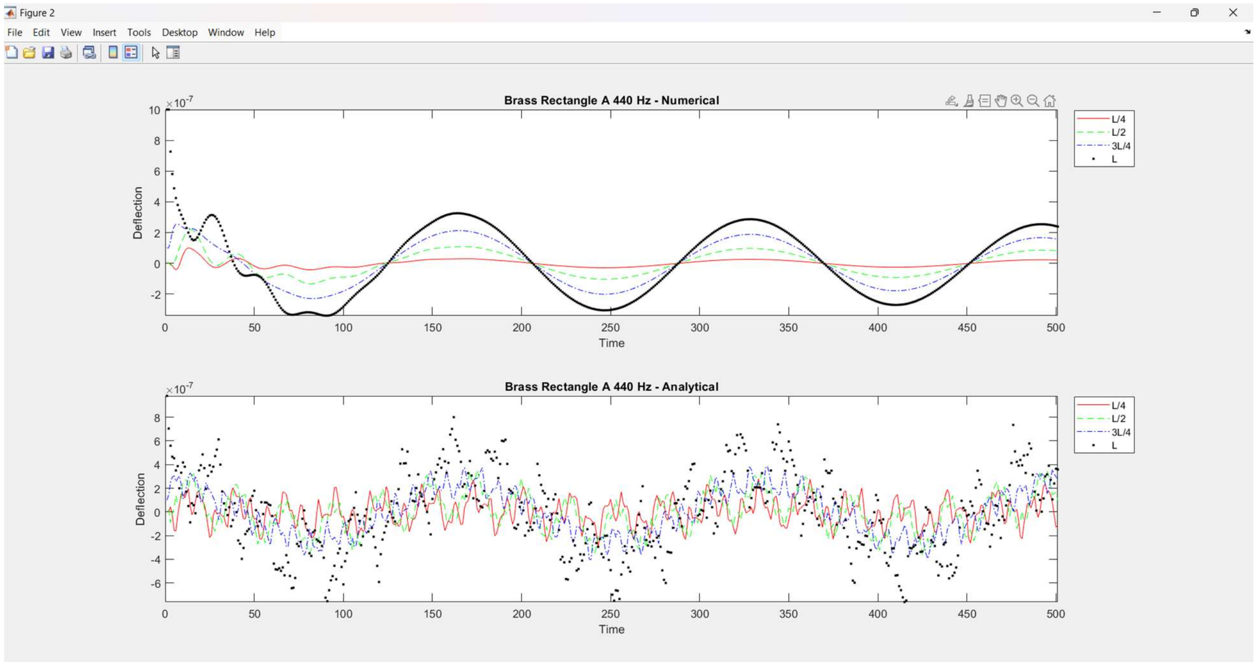


Figure (2) uses both methods for calculating deflection at different points along the tuning fork.

Figure (3)

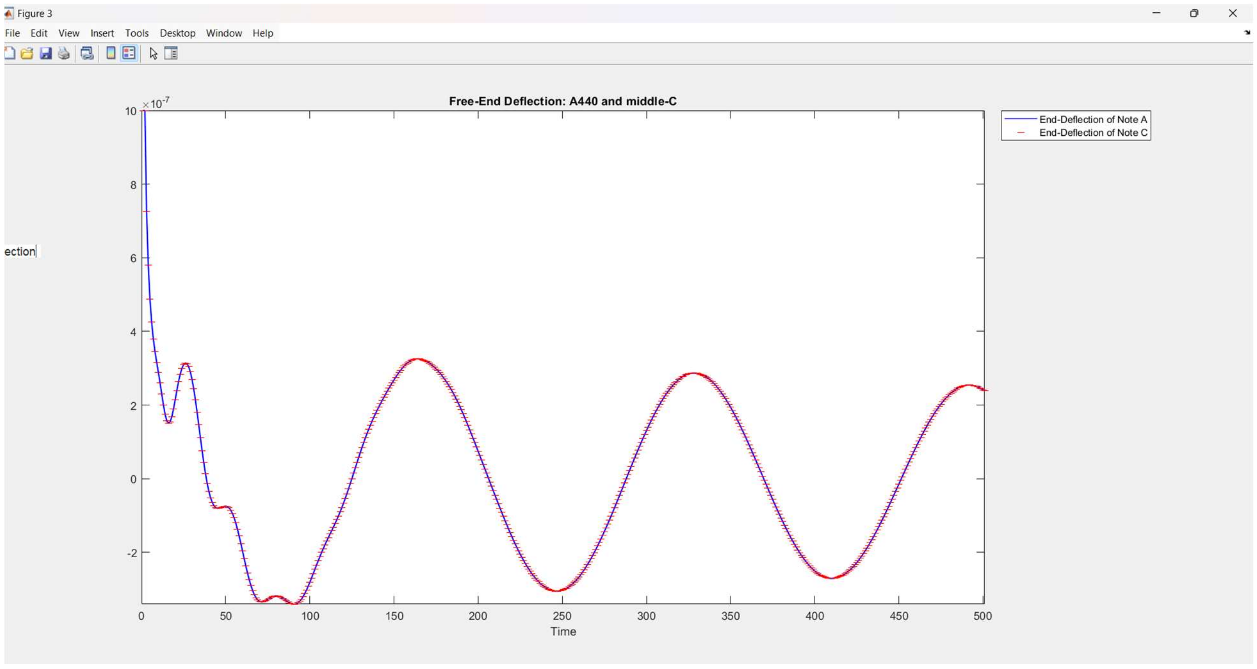
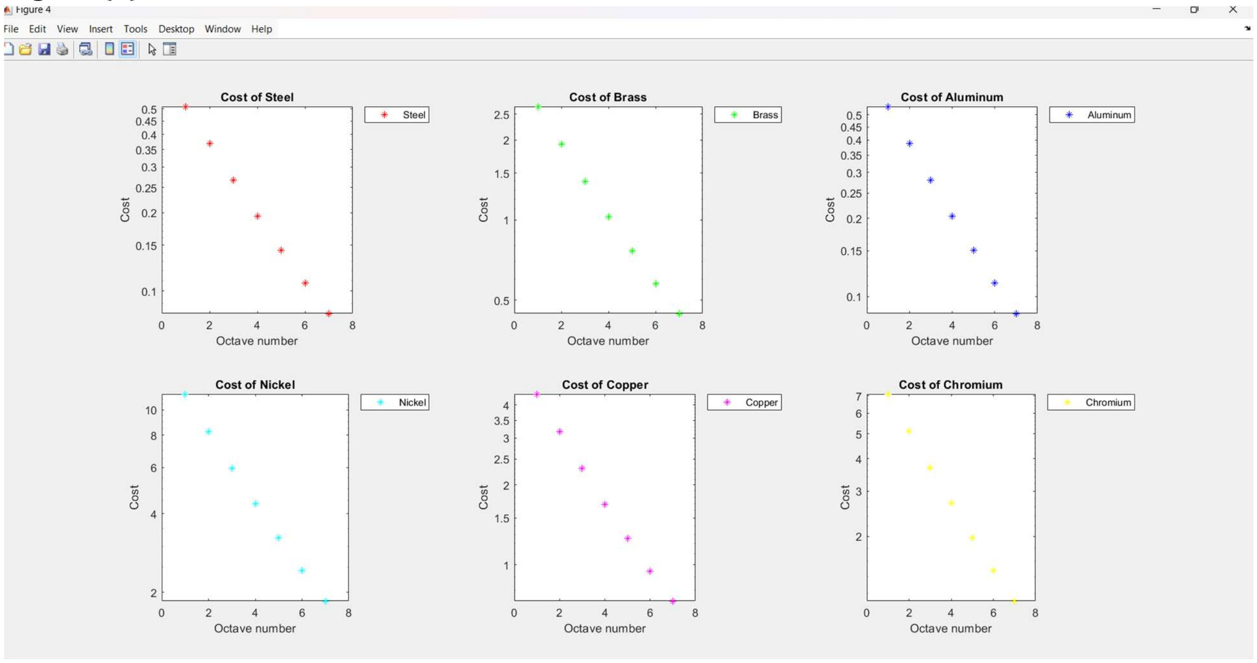


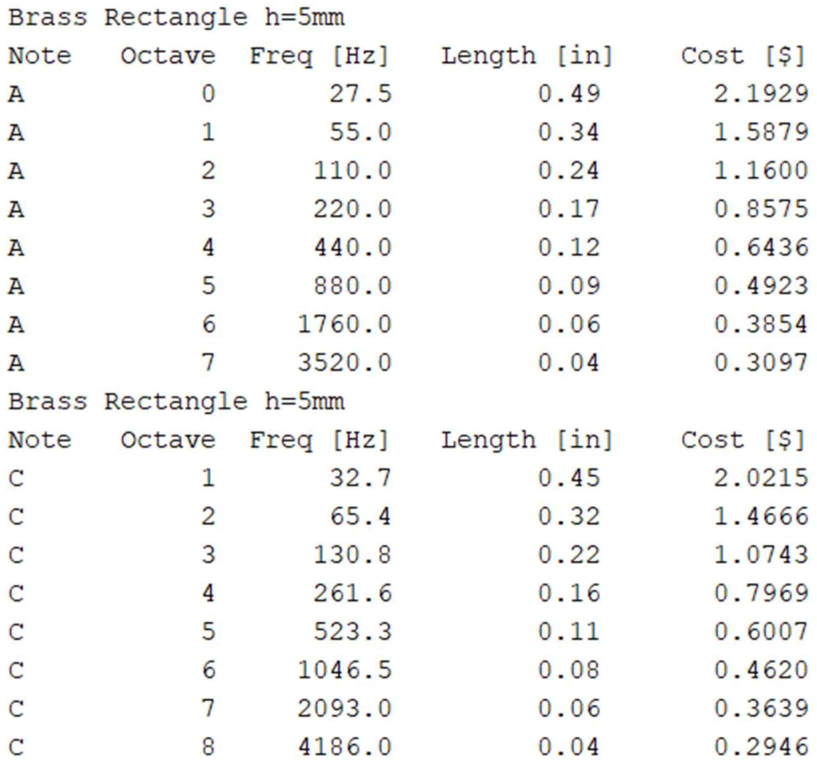
Figure (3) Compares the deflection of two different notes using the Numerical method

Figure (4)



Determined the cost of different tuning fork materials at different octaves.

Figure (5)



```
Brass Rectangle h=5mm
Note   Octave  Freq [Hz]   Length [in]    Cost [$]
A           0       27.5          0.49      2.1929
A           1       55.0          0.34      1.5879
A           2      110.0          0.24      1.1600
A           3      220.0          0.17      0.8575
A           4      440.0          0.12      0.6436
A           5      880.0          0.09      0.4923
A           6     1760.0          0.06      0.3854
A           7     3520.0          0.04      0.3097
Brass Rectangle h=5mm
Note   Octave  Freq [Hz]   Length [in]    Cost [$]
C           1       32.7          0.45      2.0215
C           2       65.4          0.32      1.4666
C           3      130.8          0.22      1.0743
C           4      261.6          0.16      0.7969
C           5      523.3          0.11      0.6007
C           6     1046.5          0.08      0.4620
C           7     2093.0          0.06      0.3639
C           8     4186.0          0.04      0.2946
```

Data for Different Note Tuning Forks from MATLAB Terminal

5. **Conclusions**

In conclusion, each file is built on top of one another. The functions in parts A and B were called in Part C, and Parts A and C were called in parts E and F. Our code continuously built on top of itself. We created ample comments throughout the code to facilitate a clear understanding of the code's design and purpose. Moreover, we used concise syntax with for-loops, accomplishing the task using minimal lines of code. Finally, we named each variable after the information it held within, making the code understandable to others. Our good programming practices improved our code by allowing each teammate to help one another since it was clear what each piece of code did. The most challenging aspect of this project was debugging our code, since a small bug such as a neglected negative sign could and did cause the entire function to produce stray results. The least challenging aspects were reading from matrices and doing calculations from the data. MATLAB as a coding language is built to do these tasks very well, which makes data processing very efficient.