

ENGO363: Estimation and Statistical Testing

Term 2024 – Lab 1

**Title:** Numerical computations in C/C++ and data  
visualization in MATLAB/Python

|  |  |
| --- | --- |
| Course instructor: | Dr. Ivan Detchev |
| Teaching Assistant: | Michael Blois |
| Date: | January 28, 2024 |
| Student: | Wesam Omran |
| Student ID: | 30133992 |

Academic Integrity Statement:

I Wesam Omran certify that this is my own work, which has been done expressly for this course, either without the assistance of any other party or where appropriate I have acknowledged the work of others. Further, I have read and understood the section in the university calendar on plagiarism/cheating/other academic misconduct and I am aware of the implications thereof. WO

Table of Contents

[List of Figures iii](#_Toc157721017)

[List of Tables iii](#_Toc157721018)

[List of Equations iv](#_Toc157721019)

[1 Introduction 1](#_Toc157721020)

[2 Methodology 1](#_Toc157721021)

[3 Results 4](#_Toc157721022)

[3.1 Tables 4](#_Toc157721023)

[3.2 Figures 5](#_Toc157721024)

[4 Discussion 6](#_Toc157721025)

[4.1 Alignment with Observed Data 6](#_Toc157721026)

[4.2 Residual Analysis 6](#_Toc157721027)

[4.3 Questions 6](#_Toc157721028)

[4.3.1 Units of Matrix : 6](#_Toc157721029)

[4.3.2 Units of Vector : 6](#_Toc157721030)

[4.3.3 Units of Vector : 6](#_Toc157721031)

[4.3.4 Units of Vector : 6](#_Toc157721032)

[5 Conclusion 7](#_Toc157721033)

[5.1 Learning outcomes 7](#_Toc157721034)

[6 References 8](#_Toc157721035)

[7 Appendix: 9](#_Toc157721036)

[7.1 Code 9](#_Toc157721037)

[7.2 Full Tables 17](#_Toc157721038)

# List of Figures

|  |  |
| --- | --- |
| Figure Name | Figure Number |
| Z(t) Vs Time | Figure 1 |
| Estimated Residuals Vs Time | Figure 2 |

# List of Tables

|  |  |
| --- | --- |
| Table Names | Table Numbers |
| Lab Observations | Table 1 |
| Estimated Unknown Parameters | Table 2 |
| Estimated Residuals | Table 3 |

# List of Equations

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

# Introduction

In this lab, we were assigned to use data taken by Dr. Detchev for calculating, estimating and analyzing the deflection of concrete specimens. The use C or C++ to perform all the calculations as C++ contains libraries that help and is an Object-Oriented Programming language that can help organize the code structure. Also, the use of Python or MATLAB to visualize the data estimated by C++. Using the programming tools will help us estimate the deflection amplitude.

# Methodology

The program was divided into numerous functions to enhance the readability of the code. First, the important libraries were imported (included) to the program to allow the utilization of them. The libraries included:

* “iostream” header file
* “fstream” header file
* “string” header file
* “cmath” header file
* Eigen folder

Those header files and folders help with writing the code with higher efficiency. Also, since there isn’t a lot of namespaces to use in this project, it is safe to make use of shortcuts of using namespaces, i.e. “using namespace std”.

Then the program was designed to begin with initializing all necessary matrices and vectors. This ensures easy accessibility and manageability to make changes as the program progresses. The following are the variables that need to be declared:

* Observation Vector Z(t)
* Time Vector t
* Design Matrix A
* Misclosure Vector w
* Normal Matrix N
* Normal Vector
* Vector
* Vector
* Vector
* Vector

Additionally, the program sets 2 constants that will be needed for calculations during the processing stage. The angular frequency using equation (1). The second constant is going to store the number of rows found in the file that the data is extracted from.

Noting the installation path of “lab1\_data\_2024” file from D2L will be required for the following steps.

The program proceeds with writing a function that will read and populate vectors t and Z(t). Using “ifstream” library, it creates an object that will read file passed on by parameter in the function. It checks to see if the file is opened properly, if not it will print a statement to the console and exit the code. Next, it counts the number of lines in the file and store it to the global variable “rows”. The code ensures that the sizes of vectors and are correct by resizing them according to the number of rows. It iterates through each row and column, storing values in vectors and using a loop. Finally it will close the file.

Next, is populating the Design Matrix A. The code ensures that the size of Matrix A is correct by resizing it with the rows global variable. Using a loop, it will iterate through each row and column, updating their values according to equation (3).

Following that, is populating the Misclosure Vector w. The code ensures that the size of Vector w is correct by resizing it with the rows global variable. Then, using equation (4), it will populate vector w.

Then a function to compute the Normal Matrix N. This is done by using equation (5).

Subsequently, a function to compute the Normal Vector . It ensures that the size of Vector is correct by resizing it with the number of rows. Using equation (6), it computes the values for vector .

A function to estimate the Unknown Parameters. First, it populates the Vector using equation (8). Then, using equation (7), it computes the values of Vector . Lastly, using equation (9), it computes the values of Vector .

A function to estimate the Residuals. It ensures the size of Vector is correct by resizing it with the number of rows. Using equation (10), it computes the values of Vector .

Finally, the program contains a function to write to 2 different files that will take in 2 string arguments. One to write the values of Vector . The other to write the values of Vector . To do that, it creates 2 objects of type “ofstream” that will create 2 files with parameters taken respectively. It checks if output files are open, if not, prints out a message to the console.Then, it initializes 2 different variables that each will be used to loop through their respective vectors, Vector , and Vector . In the loops, the code outputs each row to a line in the created file, ensuring that the files are closed when it is done.

In the “main” function, the program declares a string with value of the “lab1\_data\_2024” file path. This will be used as a parameter in the function responsible for reading the file. Then, executes the remaining functions up until the function responsible for writing the files. Then, it declares 2 strings with values responsible for the names of the files that will be created. It uses these as parameters in the function responsible for writing the files.

After the results of the calculations were made and written to 2 separate files. Python was used, with the help of the following libraries, to visualize the data and compare them using graphical implementations:

* Numpy
* Matlibplot

The program starts with importing the libraries needed. Then by creating variables and . The values of these variables are set using the function loadtext(). Then setting the constant . Finally, it calculates the value of the smooth curve using equation (2). Finally, using matplotlib library, it creates 2 plots, first one comparing the smooth curve of Vs and the actual points of Vs , and the second one is plotting the estimated residuals Vs .

# Results

The tables below show the results of the computations done in the program written in C++. The calculations were made through using the data in the dataset *Table 1 – Lab Observations*.

## Tables

Table – Lab Observations

|  |  |  |
| --- | --- | --- |
| Observation # | Time [ss] | Z(t) [mm] |
| 1 | 0 | -68.740 |
| 2 | 0.063 | -67.860 |
| 3 | 0.125 | -67.280 |
| 4 | 0.188 | -68.250 |
| … | … | … |
| … | … | … |
| 78 | 4.813 | -75.470 |
| 79 | 4.875 | -74.520 |
| 80 | 4.938 | -72.110 |
| 81 | 5.000 | -69.570 |

Note: Full Table will be included in Appendi Full Tables.

Table – Estimated Unknown Parameters

|  |  |
| --- | --- |
| Parameters | Estimated Unknown Parameters |
| a | 4.38221 |
| b | 5.9514 |
| c | -74.9966 |

Table – Estimated Residuals

| Observation # | Time [ss] | Estimated Residuals |
| --- | --- | --- |
| 1 | 0 | -0.30516 |
| 2 | 0.063 | 0.0388176 |
| 3 | 0.125 | -0.409591 |
| 4 | 0.188 | -0.42042 |
| … | … | … |
| … | … | … |
| 78 | 4.813 | -1.29769 |
| 79 | 4.875 | 0.633027 |
| 80 | 4.938 | 0.934821 |
| 81 | 5.000 | 0.524844 |

Note: Full Table will be included in Appendix: Full Tables.

## Figures

The figures below show a visualization of the calculations made and comparing them.

A graph of a graph

Description automatically generated

Figure Shows the smooth representation of the Observed data with respect to Time VS the Actual points of the Observed data with respect to Time.

A graph showing a number of blue dots

Description automatically generated with medium confidence

Figure Shows the Estimated Residuals with respect to Time.

# Discussion

## Alignment with Observed Data

The harmonic model's effectiveness is initially supported by the alignment of estimated parameters with the observed data (Table 1). Figure 3.1 visually represents this alignment, affirming the model's capability to capture the deflection properties of the concrete specimens. The precision of this alignment underscores the model's reliability in representation.

## Residual Analysis

Figure 3.2 portrays the residuals, indicating deviations from the model's predictions. While the residuals are generally distributed around zero, suggesting a fair precision of the model, specific areas reveal possible systematic errors or limitations. These subtleties prompt critical questions about potential improvements to enhance overall accuracy.

The estimated residuals (Table 3) highlight areas where the model may fall short in predicting deflection accurately. Potential systematic errors warrant a thoughtful consideration of model refinement or additional data processing, because with more data, the analysis could be more accurate with outliers being a lot less impactful. This analysis is crucial for ensuring the model's reliability and applicability to real-world scenarios.

## Questions

### Units of Matrix :

The units in columns of the design matrix A are dimensionless. Since first and second columns are calculated by and , where is in and is in , then that will result in a unitless quantity.

### Units of Vector :

The units of matrix is in , because the matrix is simply the negative of matrix .

### Units of Vector :

The units of Vector are in , because it is the result of the inverse of normal vector multiplied by the normal equation vector , which has units of .

### Units of Vector :

The vector of residuals 𝑣̂, which represents the differences between the predicted and observed deflections, will also have units of .

# Conclusion

In this study, we applied the linear fundamental harmonic equation to estimate deflections in concrete specimens using laser transducer measurements. Through rigorous data fitting and parameter estimation in C/C++, that resulted in gaining several key ideas.

The linear fundamental harmonic equation was used, and the sinusoidal and cosine terms showed a good fit with the data that was observed. The dynamic behaviour of the concrete specimens during testing was well-represented by the model.

The unknown parameters and were successfully estimated, providing valuable information about the amplitude and phase of the deflection. These estimations, carried out through the computation of the normal equation matrix and vector, contribute to a comprehensive understanding of the structural response.

The residual analysis provided insight into any systematic flaws or model constraints. These observations highlight the significance of continuous progress in deflection estimation accuracy and call for further considerations for data processing and model modification.

A reliable and adaptable method was demonstrated by the combination of Python for data visualisation and C++ for parameter estimate and data manipulation. This combination of programming languages facilitated efficient calculations and insightful visualizations.

To sum up, this research offers a comprehensive examination of deflection estimation and identifies areas that require improvement. As we continue to explore the complexities of structural behaviour, the results shown here support the continued search for accuracy and consistency in deflection modelling.

## Learning outcomes

In this lab I was able to develop my knowledge and understanding of handling input and output from and to text files in C++. Learned how to use the Eigen library and carefully and correctly incorporate into my code and utilize its power of handling matrices. I was also able to apply my knowledge from other courses in Python to generate graphs to visualize the processed data using Numpy and Matlibplot libraries. Finally, learning how to write a proper lab report is crucial to me as an aspiring engineer.

If I had more time, I would create header classes for the calculation and population functions, and their respective C++ files, which would be separate from the main function file where the program is being executed. That would make the code more readable and manageable. I would also try to make use of the MatrixXf constructor in the Eigen class, since it would be more generic and be more optimized.

# References

**[1] Eigen, "Eigen: A C++ template library for linear algebra" [Online]. Available:** <https://eigen.tuxfamily.org/index.php?title=Main_Page>.

**[2] W3Schools, "C++ Files" [Online]. Available:** <https://www.w3schools.com/cpp/cpp_files.asp>.

**[3] D2L, “ENGOXXX\_LabZ\_LastNameFirstinitial\_yyyymmdd” [Online]. Available:** <https://d2l.ucalgary.ca/d2l/le/content/569767/viewContent/6313876/View>.

# Appendix:

## Code



























## Full Tables

Table 1 – Lab Observations

| Observation # | Time [ss] | Z(t) [mm] |
| --- | --- | --- |
| 1 | 0.000 | -68.740 |
| 2 | 0.063 | -67.860 |
| 3 | 0.125 | -67.280 |
| 4 | 0.188 | -68.250 |
| 5 | 0.250 | -71.590 |
| 6 | 0.313 | -73.800 |
| 7 | 0.375 | -76.780 |
| 8 | 0.438 | -78.620 |
| 9 | 0.500 | -80.260 |
| 10 | 0.563 | -82.430 |
| 11 | 0.625 | -82.020 |
| 12 | 0.688 | -81.180 |
| 13 | 0.750 | -80.030 |
| 14 | 0.813 | -76.660 |
| 15 | 0.875 | -74.320 |
| 16 | 0.938 | -70.420 |
| 17 | 1.000 | -69.020 |
| 18 | 1.063 | -66.920 |
| 19 | 1.125 | -67.790 |
| 20 | 1.188 | -68.250 |
| 21 | 1.250 | -70.530 |
| 22 | 1.313 | -74.150 |
| 23 | 1.375 | -76.550 |
| 24 | 1.438 | -78.510 |
| 25 | 1.500 | -81.030 |
| 26 | 1.563 | -82.820 |
| 27 | 1.625 | -82.740 |
| 28 | 1.688 | -81.330 |
| 29 | 1.750 | -80.000 |
| 30 | 1.813 | -76.370 |
| 31 | 1.875 | -74.260 |
| 32 | 1.938 | -70.270 |
| 33 | 2.000 | -69.540 |
| 34 | 2.063 | -67.640 |
| 35 | 2.125 | -68.340 |
| 36 | 2.188 | -68.910 |
| 37 | 2.250 | -70.800 |
| 38 | 2.313 | -72.940 |
| 39 | 2.375 | -75.590 |
| 40 | 2.438 | -78.670 |
| 41 | 2.500 | -81.190 |
| 42 | 2.563 | -81.860 |
| 43 | 2.625 | -82.030 |
| 44 | 2.688 | -81.390 |
| 45 | 2.750 | -79.210 |
| 46 | 2.813 | -76.650 |
| 47 | 2.875 | -74.430 |
| 48 | 2.938 | -70.800 |
| 49 | 3.000 | -69.330 |
| 50 | 3.063 | -68.040 |
| 51 | 3.125 | -67.490 |
| 52 | 3.188 | -68.700 |
| 53 | 3.250 | -70.570 |
| 54 | 3.313 | -72.840 |
| 55 | 3.375 | -75.540 |
| 56 | 3.438 | -78.920 |
| 57 | 3.500 | -80.840 |
| 58 | 3.563 | -82.380 |
| 59 | 3.625 | -82.310 |
| 60 | 3.688 | -81.230 |
| 61 | 3.750 | -79.810 |
| 62 | 3.813 | -76.720 |
| 63 | 3.875 | -73.640 |
| 64 | 3.938 | -71.130 |
| 65 | 4.000 | -68.140 |
| 66 | 4.063 | -68.040 |
| 67 | 4.125 | -67.940 |
| 68 | 4.188 | -69.420 |
| 69 | 4.250 | -70.040 |
| 70 | 4.313 | -72.710 |
| 71 | 4.375 | -76.100 |
| 72 | 4.438 | -78.370 |
| 73 | 4.500 | -80.910 |
| 74 | 4.563 | -82.120 |
| 75 | 4.625 | -82.370 |
| 76 | 4.688 | -81.230 |
| 77 | 4.750 | -79.450 |
| 78 | 4.813 | -75.470 |
| 79 | 4.875 | -74.520 |
| 80 | 4.938 | -72.110 |
| 81 | 5.000 | -69.570 |

*Table 3 – Estimated Residuals*

| Observation # | Time [ss] | Estimated Residuals [mm] |
| --- | --- | --- |
| 1 | 0.000 | -0.30516 |
| 2 | 0.063 | 0.0388176 |
| 3 | 0.125 | -0.409591 |
| 4 | 0.188 | -0.42042 |
| 5 | 0.250 | 0.975654 |
| 6 | 0.313 | 0.574575 |
| 7 | 0.375 | 0.673857 |
| 8 | 0.438 | -0.197937 |
| 9 | 0.500 | -0.687962 |
| 10 | 0.563 | 0.258061 |
| 11 | 0.625 | -0.28353 |
| 12 | 0.688 | -0.142701 |
| 13 | 0.750 | 0.651225 |
| 14 | 0.813 | -0.107696 |
| 15 | 0.875 | 0.433022 |
| 16 | 0.938 | -0.755184 |
| 17 | 1.000 | -0.0251594 |
| 18 | 1.063 | -0.901182 |
| 19 | 1.125 | 0.100409 |
| 20 | 1.188 | -0.42042 |
| 21 | 1.250 | -0.084347 |
| 22 | 1.313 | 0.924574 |
| 23 | 1.375 | 0.443856 |
| 24 | 1.438 | -0.307938 |
| 25 | 1.500 | 0.0820376 |
| 26 | 1.563 | 0.648061 |
| 27 | 1.625 | 0.43647 |
| 28 | 1.688 | 0.00729932 |
| 29 | 1.750 | 0.621226 |
| 30 | 1.813 | -0.397695 |
| 31 | 1.875 | 0.373023 |
| 32 | 1.938 | -0.905182 |
| 33 | 2.000 | 0.494841 |
| 34 | 2.063 | -0.181182 |
| 35 | 2.125 | 0.650409 |
| 36 | 2.188 | 0.239579 |
| 37 | 2.250 | 0.185652 |
| 38 | 2.313 | -0.285427 |
| 39 | 2.375 | -0.516145 |
| 40 | 2.438 | -0.147939 |
| 41 | 2.500 | 0.242037 |
| 42 | 2.563 | -0.31194 |
| 43 | 2.625 | -0.27353 |
| 44 | 2.688 | 0.0673 |
| 45 | 2.750 | -0.168773 |
| 46 | 2.813 | -0.117693 |
| 47 | 2.875 | 0.543025 |
| 48 | 2.938 | -0.375181 |
| 49 | 3.000 | 0.284842 |
| 50 | 3.063 | 0.218818 |
| 51 | 3.125 | -0.199592 |
| 52 | 3.188 | 0.0295783 |
| 53 | 3.250 | -0.0443491 |
| 54 | 3.313 | -0.385429 |
| 55 | 3.375 | -0.566147 |
| 56 | 3.438 | 0.102059 |
| 57 | 3.500 | -0.107964 |
| 58 | 3.563 | 0.20806 |
| 59 | 3.625 | 0.0064704 |
| 60 | 3.688 | -0.0926993 |
| 61 | 3.750 | 0.431228 |
| 62 | 3.813 | -0.0476922 |
| 63 | 3.875 | -0.246974 |
| 64 | 3.938 | -0.0451802 |
| 65 | 4.000 | -0.905157 |
| 66 | 4.063 | 0.218819 |
| 67 | 4.125 | 0.250408 |
| 68 | 4.188 | 0.749578 |
| 69 | 4.250 | -0.57435 |
| 70 | 4.313 | -0.51543 |
| 71 | 4.375 | -0.00614804 |
| 72 | 4.438 | -0.447942 |
| 73 | 4.500 | -0.0379647 |
| 74 | 4.563 | -0.0519403 |
| 75 | 4.625 | 0.0664706 |
| 76 | 4.688 | -0.0926987 |
| 77 | 4.750 | 0.0712293 |
| 78 | 4.813 | -1.29769 |
| 79 | 4.875 | 0.633027 |
| 80 | 4.938 | 0.934821 |
| 81 | 5.000 | 0.524844 |