3801ICT Numerical Algorithms
Assignment Part Two

Ethan Leet
s5223103
ethan.leet@griffithuni.edu.au

June 7, 2022

Contents

0 Introduction

0.1 Overview

This assignment involved writing c++ code to solve four Numerical Algorithm based problems and solving one problem by hand. Test cases were then created to prove the validity of the algorithm used for each of the four coding problems. Each question is broken down into several sub sections throughout this report. The sub sections include; problem statement, algorithm description, complexity analysis, and experiments and results.

0.2 Compilation and Runtime

All programs included in this submission compile without errors nor warnings using clang++ and the c++ standard library 14. Each program contains a make file which will compile the program with all its required dependencies. There is also a make file in the parent folder which will make every program in this submission and place the binaries in the respective question folders. The following command can be executed to run the 'make' command:

make

If make is not available please check the respective questions makefile to extract the command used to compile individual programs with their respective binaries.

After successful compilation programs can then be executed with the commands described below, where the question number is used in place of x. Note that in order to execute the following command you must be inside the question folder.

./questionx

1 Systems of Linear Equations

1.1 Problem Statement

By hand, use Gauss Elimination, LU Decomposition and the Gauss-Jordan method to solve the following system of linear equations to six significant figures. Show each step of your calculations.

$$0.1x_1 + 7x_2 - 0.3x_3 = -19.3$$

$$0.3x_1 - 0.2x_2 + 10x_3 = 71.4$$

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85$$

1.2 Gauss Elimination

$$\left(\begin{array}{ccc|c} 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \\ 3 & -0.1 & -0.2 & 7.85 \end{array}\right)$$

$R_2 \to R_2 - 3R_1$
$R_3 \to -30R_1$

$$\left(\begin{array}{ccc|c} 0.1 & 7 & -0.3 & -19.3 \\ 0 & -21.2 & 10.9 & 129.3 \\ 0 & -210.1 & 8.8 & 586.8 \end{array}\right)$$

$R_3 \to R_3 - \frac{2101}{212} R_2$

$$\left(\begin{array}{ccc|c} 0.1 & 7 & -0.3 & -19.3 \\ 0 & -21.2 & 10.9 & 129.3 \\ 0 & 0 & -99.2231 & -694.612 \end{array}\right)$$

$\Rightarrow x_3 = \frac{693.612}{99.2231} = 6.99043$

$\Rightarrow x_2 = \frac{129.3 - 10.9x_3}{-21.2} = -2.49974$

$\Rightarrow x_1 = \frac{-19.3 + 0.3x_3 - 7x_2}{0.1} = 2.98333$

If we sub these values for x back into our original equations we get the following:

$0.1 \times 2.98333 + 7 \times -2.49974 - 0.3 \times 6.99043 = -19.2970$

$0.3 \times 2.98333 - 0.2 \times -2.49974 + 10 \times 6.99043 = 71.2993$

$3 \times 2.98333 - 0.1 \times -2.49974 - 0.2 \times 6.99043 = 7.80188$

1.3 LU Decomposition

$$\begin{pmatrix} 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \\ 3 & -0.1 & -0.2 \end{pmatrix}$$

$R_2 \rightarrow R_2 - 3R_1 = l_{2,1}$
$R_3 \rightarrow R_3 - 30R_1 = l_{3,1}$

$$\begin{pmatrix} 0.1 & 7 & -0.3 \\ 0 & -21.2 & 10.9 \\ 0 & -210.1 & 8.8 \end{pmatrix}$$

$R_3 \rightarrow R_3 - \frac{2101}{212}R_2 = l_{3,2}$

$$\begin{pmatrix} 0.1 & 7 & -0.3 \\ 0 & -21.2 & 10.9 \\ 0 & 0 & -99.2231 \end{pmatrix} = U$$

$$A = \begin{pmatrix} 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \\ 3 & -0.1 & -0.2 \end{pmatrix}, L = \begin{pmatrix} 1 & 0 & 0 \\ l_{2,1} & 1 & 0 \\ l_{3,1} & l_{3,2} & 1 \end{pmatrix}$$

$$\Rightarrow L = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 30 & \frac{2101}{212} & 1 \end{pmatrix}, X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, Z = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}, C = \begin{pmatrix} -19.3 \\ 71.4 \\ 7.85 \end{pmatrix}$$

$\Rightarrow LZ = C$
$\Rightarrow z_1 = -19.3$
$\Rightarrow z_2 = 71.4 - 3z_1 = 71.4 - 3 \times -19.3 = 129.3$
$\Rightarrow z_3 = 7.85 - 30z_1 - \frac{2101}{212}z_2 = 7.85 - 30 \times -19.3 - \frac{2101}{212}129.3 = -694.562$

$\Rightarrow UX = Z$

$\Rightarrow x_3 = \frac{694.562}{99.2231} = 7$

$\Rightarrow x_2 = \frac{129.3 - 10.9x_3}{-21.2} = \frac{129.3 - 10.9 \times 7}{-21.2} = -2.5$

$\Rightarrow x_1 = \frac{-19.3 + 0.33x_3 - 7x_2}{0.1} = \frac{-19.3 + 0.3 \times 7 - 7 \times -2.5}{0.1} = 3$

If we sub these values for x back into our original equations we get the following:

$0.1 \times 3 + 7 \times -2.5 - 0.3 \times 7 = -19.3$

$0.3 \times 3 - 0.2 \times -2.5 + 10 \times 7 = 71.4$

$3 \times 3 - 0.1 \times -2.5 - 0.2 \times 7 = 7.85$

1.4 Gauss-Jordan

$$\left(\begin{array}{ccc|c} 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \\ 3 & -0.1 & -0.2 & 7.85 \end{array}\right)$$

$R_1 \to R_1/0.1$
$R_2 \to R_2 - 0.3R_1$
$R_3 \to R_3 - 3R_1$

$$\left(\begin{array}{ccc|c} 1 & 70 & -3 & -193 \\ 0 & 21.2 & 10.9 & 129.3 \\ 0 & -210.1 & -0.2 & 585.85 \end{array}\right)$$

$R_2 \to R_2/-21.2$
$R_3 \to R_3 + 210.1R_2$

$$\left(\begin{array}{ccc|c} 1 & 70 & -3 & -193 \\ 0 & 1 & -0.514151 & -6.09906 \\ 0 & 0 & -99.2231 & -694.563 \end{array}\right)$$

$R_3 \to R_3/-99.2231$
$R_1 \to R_1 - 70R_2$

$$\left(\begin{array}{ccc|c} 1 & 0 & 32.9906 & 233.9342 \\ 0 & 1 & -0.514151 & -6.09906 \\ 0 & 0 & 1 & 7.00001 \end{array}\right)$$

$R_2 \to R_2 + 0.51415R_3$
$R_1 \to R_1 - 32.9906R_3$

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & 2.99967 \\ 0 & 1 & 0 & -2.5 \\ 0 & 0 & 1 & 7.00001 \end{array}\right)$$

If we sub these values for x back into our original equations we get the following:

$0.1 \times 2.99967 + 7 \times -2.5 - 0.3 \times 7.00001 = -19.3$

$0.3 \times 2.99967 - 0.2 \times -2.5 + 10 \times 7.00001 = 71.3991$

$3 \times 2.99967 - 0.1 \times -2.5 - 0.2 \times 7.00001 = 7.84901$

1.5 Summary

In testing each of the above methods, every calculation was rounded to six significant figures as requested in the problem statement. It can be seen that when any rounding occurs error of a small tolerance can be seen for the Gauss Elimination and Gauss-Jordan methods. This is due to the amount of calculations needed for both causing eventual overflow or underflow in the intermediate and finals results. It can be seen that this error does not appear in the LU decomposition method.

Assuming some appropriate error tolerance is allowed, all methods provide correct answers and are easily found. The Gauss Elimination method is the easier method to conduct by hand followed by the Gauss-Jordan and LU Decomposition methods.

2 Trajectory of a Thrown Ball

2.1 Problem Statement

The trajectory of a thrown ball is defined by the (x, y) coordinates that can be modelled as:

$$y = (tan\theta_0)x - \frac{g}{2v_0^2 cos^2\theta_0}x^2 + y_0$$

Programmatically using C++ find the appropriate initial angle θ_0, if the initial velocity v_0 = 20 m/s and the distance to the catcher x is 35 m. Note that the ball leaves the thrower's hand at an elevation of y_0 = 2m and the catcher receives it at an elevation of 1 m. Express the final result in degrees. Use a value of $9.81 m/s^2$ for g.

2.2 Algorithm Description

To solve this problem it was observed that setting the equation to zero and finding the roots of the new equation will approximate values for θ.

The new equation is as follows:

$$(tan\theta_0)x - \frac{g}{2v_0^2 cos^2\theta_0}x^2 + y_0 - y = 0$$

To approximate the roots of the equation the Bisection method was used. The Bisection method operates by systematically reducing the interval of $[x_0, x_1]$ into equal parts based on a midpoint calculation. It then performs a simple test and discards half the function. This process repeats until a root of the function is found. It assumes that the function $f(x)$ is continuous on the limits of $[x_0, x_1]$ and that $f(x_0) \times f(x_1) < 0$.

In order to use this method the above conditions were confirmed to be true; the limits $[x_0, x_1]$ were clamped at 0 and $\frac{\pi}{2}$. It can be seen that with a lower limit than 0 the ball would be thrown or dropped on the ground, and with a higher limit of $\frac{\pi}{2}$ the ball would be thrown directly up or backwards.

Algorithm 1 Bisection method

loop
  $MidPoint \leftarrow \frac{x_0+x_1}{2}$
  $f_{MidPoint} \leftarrow f(MidPoint)$
  if $f(x_0) \times f_{MidPoint} < 0$ **then**
    $x_1 \leftarrow MidPoint$
  end if
  if $f(x_0) \times f_{MidPoint} > 0$ **then**
    $x_0 \leftarrow MidPoint$
  end if
end loop

2.3 Complexity

2.3.1 Space Complexity

The Bisection method updates the function intervals and the new root value upon each traversal. As such, its space complexity can be considered constant and of order $O(1)$.

2.3.2 Time Complexity

The run-time complexity for the Bisection method is bounded by the iterations needed to meet the defined tolerance, in this case 0.001%. If we assume the amount of iterations needed is n the method runs in linear time. The program therefore runs in $2n$ time and is of order $O(n)$.

2.4 Experiments and Results

Before the Bisection method was chosen two other root-finding methods were considered; Newton-Raphson and Secant methods. The Newton-Raphson method was decided to be too hard as a derivative value is needed. The Secant method was initially tried however it did not find any of the two roots present due to the fact that the Secant method does not always converge.

As the $[x_0, x_1]$ limits were known, the Bisection method was chosen as the implemented method. Using the initial clamps as mentioned above the first root of $\theta_1 = 27.2039$ degrees was found. To find the second root the method had to be clamped at $[\theta_1, \frac{\pi}{2}]$ which pplankuced $\theta_2 = 61.1598$ degrees.

It was theorized that other approximation methods could be used to solve this type of problem, such as the Golden Section Search method. This approach was not chosen due to the simplicity of the Bisection method and knowing the $[x_0, x_1]$ limits. If the limits were unknown, and the Secant and Newton-Raphson methods were also unable to be used the Golden Section Search method would be an appropriate tool to solve like problems.

3 Space Shuttle Thrust

3.1 Problem Statement

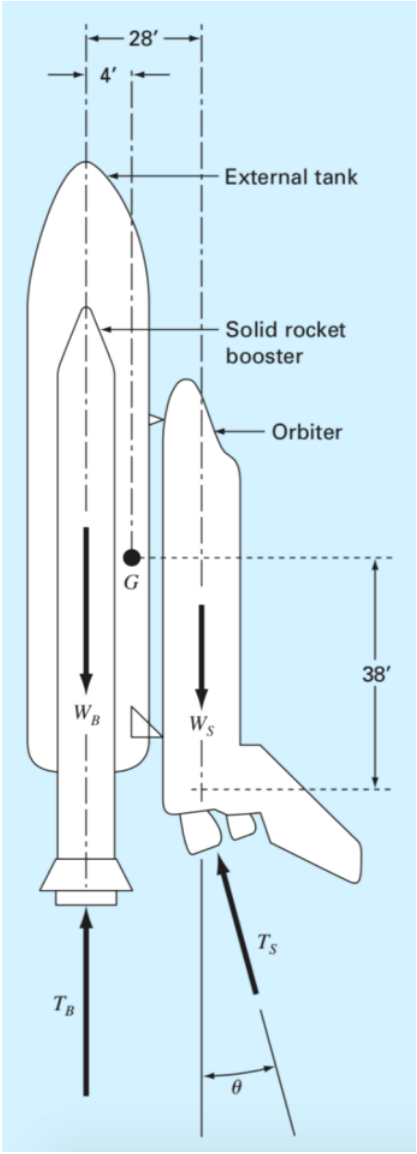
The space shuttle, at lift-off from the launch pad, has four forces acting on it, which are shown on the following diagram. The combined weight of the two solid rocket boosters and external fuel tank is $WB = 1.66310^6$ lb of which 1.30010^6 lb is fuel which burns at a constant rate for 127 seconds. The weight of the orbiter with a full payload is $WS = 0.2310^6$ lb. The combined thrust of the two solid rocket boosters is $TB = 5.3010^6$ lb. The combined thrust of the three liquid fuel orbiter engines is $TS = 1.12510^6$ lb.

At lift-off, the orbiter engine thrust is directed at angle θ to make the resultant moment acting on the entire craft assembly (external tank, solid rocket boosters, and orbiter) equal to zero. With the resultant moment equal to zero, the craft will not rotate about its mass centre G at lift-off. With these forces, the craft will have a resultant force with components in both the vertical and horizontal direction. The vertical resultant force component is what allows the craft to lift off from the launch pad and fly vertically.
The horizontal resultant force component causes the craft to fly horizontally. The resultant moment acting on the craft will be zero when θ is adjusted to the proper value. If this angle is not adjusted properly, and there is some resultant moment acting on the craft, the craft will tend to rotate about it's mass centre.

1. Derive the moment equation about point G, the centre of mass.

2. Write a C++ program to solve for the angle θ using the Secant method to find the root of the moment equation. Terminate your iterations when the value of θ has better than five significant figures.

3. T_B is constant over the 127 seconds so the fuel is burned at a constant rate which means the moment about point G resulting from T_B increases over time. This requires continual adjustment to θ so there is no rotation. Determine the appropriate values for θ at suitable intervals and interpolate these values using Lagrange Interpolating Polynomials to obtain an equation giving how θ should be changed during the 127 seconds so that the moment equation is always zero.


28′
4′
External tank
Solid rocket
booster
Orbiter
G
38′
W_B
W_S
T_S
T_B
θ

3.2 Algorithm Description

The first step of this problem is to define an appropriate moment equation given the forces described in the problem statement. As such, the moment equation about point G, the centre of mass, can be described as:

$$m(G) = (4T_B + 24W_S + 38T_S sin(\theta)) - (4W_B + 24T_S sin(\theta))$$

The Secant Method uses two initial guesses of the root x_0 and x_1 to find a more accurate estimation of the root. It operates under the assumption that two initial points are given and that $f(x_0) - f(x_1) \neq 0$. This method uses a current approximation x_{i-1} and x_i to find a better approximation x_{i+1} using the below equation. It will continue to estimate a more accurate guess of the root x_{i+1} until some predefined tolerance is met.

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

$$f(x_{i+1}) = f(x_i) + (x_{i+1} - x_i)\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

A root implies $f(x_{i+1}) = 0$ so,

$$0 = f(x_i) + (x_{i+1} - x_i)\frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

$$x_{i+1}(f(x_i) - f(x_{i-1})) = x_i(f(x_i) - f(x_{i-1})) - f(x_i)(x_i - x_{i-1})$$

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

To achieve a tolerance of five significant figures the following error calculation was used where n is the amount of significant figures required:

$$\epsilon_s = (0.5 \times 10^{(2-n)})\%$$

The two initial guesses of the root x_0 and x_1 were defined as 0 and $\frac{\pi}{2}$ respectively. These values were identified as the widest possible clamping points for the thrust components before a backwards thrust would be implied. As an example, if a thrust had a value $> \theta = \frac{\pi}{2}$, the thrust would be in the opposite direction the component is facing, and therefore invalid.

To calculate the moment about point G as T_B increases over time, θ was continuously calculated over the period the fuel was being burned (127 seconds). At each time interval t the above secant method was used to evaluate θ at the new $weight = W_B - \frac{127}{W_B - fuel} \times t$. This resulted in $127(x, y)$ data-points which could be used to interpolate using Lagrange Interpolating Polynomials.

The Lagrange Polynomial is the summation of terms where each term is; equal to f() at a data point, equal to zero at all other data-points and, each term is an n^{th} degree polynomial:

$$p_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

$$L_i(x) = {}_{k=0,k\neq i}^{n} \frac{x - x_i}{x_i - x_k}$$

Using the interpolated values, Linear Regression could then be conducted to show how θ should be changed during the time interval so the moment equation is always $zero$. To accurately fit and find a unique regression line for a set of data the strategy is to minimize the sum of the squares of the residuals between the measured-y and the y-calculated with the linear model.
In order to minimise error:

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_i x_i)^2$$

Then, we can find normal equations which can be solved simultaneously:

$$(1) = \sum y_i = n a_0 + (\sum x_i) a_i$$

$$(2) = \sum y_i x_i = (\sum x_i) a_0 + (\sum x_i^2) a_i$$

$$a_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_1}{n \sum x_i^2 - (\sum x_i)^2}$$

Finally, using (1), a_0 can be expressed as $a_0 = \bar{y} - a_i \bar{x}$ Where $\bar{y}$ and $\bar{x}$ are mean values. Thus, resulting in a linear equation showing how θ should be changed during the 127 seconds so that the moment equation is always zero.

3.3 Complexity

3.3.1 Space Complexity

The program is bounded in space by the size of the terms used by the Lagrange Interpolating Polynomials. In this case T, where T is the amount of intervals in $time = t$ the fuel takes to burn to 0. Therefore, a total space complexity of $O(T)$ can be expected for this program.

3.3.2 Time Complexity

To Assess the time complexity first consider each individual function:

1. Secant Method
   This function iterates until some predefined tolerance is met, 0.001% in this particular case. As such, the time complexity for this program is $O(i)$ where i is the amount of iterations used to reach the tolerance.

2. Lagrange Interpolating Polynomials Method
   The Lagrange Interpolating Polynomials operates on two sets of data-points (x, y) of size $n = x = y$. For every x data-point traversed the entire set of y data-points are traversed, totalling a space complexity of $O(n^2)$.

3. Linear Regression Method
   This final function traverses the set of points passed to it three times. Firstly to work out the summations, secondly to work out the S_t value and finally to approximate the S_r value. However, the size of the data-set passed to the function to be regressed depends on the amount of terms that in your model. As an example, two points is a linear model, $three$ points is quadratic and $four$ points is a cubic model. It is known that for a general model between two and $three$ points will provide the regression prediction. Therefore, the amount of data-points passed to it will be minute, and the function can be considered to operate in constant time.

As such, this program is bounded by the Lagrange Interpolating Polynomials Method and can be expected to operate in $O(n^2)$ time.

3.4 Experiments and Results

Using the approach described in Section 3.2, the root of the moment equation (in radians) was found to be $\theta = 0.155184$.

Several experiments were conducted using Lagrange Interpolating Polynomials to find an accurate equation for how θ should change over time. The results are stated below:

1. 2-Point Estimation
   Using the points (0, 127) $\theta(63)$ was predicted to be 0.0980555 radians using the Lagrange Estimate. The true $\theta(63)$ was calculated to be 0.0650833 which results in a true error of 0.00841687%.

   The formula to describe how θ should change over time was solved to be $y = -0.00090681x + 0.155184$ with $S_t = 0.00663148$, $S_r = 0$, $r^2 = 1$ and the correlation coefficient $= 1$.

2. 3-Point Estimation
   Using the points (0, 63, 127) $\theta(100)$ was predicted to be 0.0650608 radians using the Lagrange Estimate. The true $\theta(100)$ was calculated to be 0.0650833 which results in a true error of 0.000345351%.

   The formula to describe how θ should change over time was solved to be $y = -0.000906845x + 0.155464$ with $S_t = 0.00663258$, $S_r = 3.83647 \times 10^{-7}$, $r^2 = 0.999942$ and the correlation coefficient $= 0.999971$.

3. 4-Point Estimation
   Using the points (0, 42, 84, 127) $\theta(100)$ was predicted to be 0.0650829 radians using the Lagrange Estimate. The true $\theta(100)$ was calculated to be 0.0650833 which results in a true error of $7.23703 \times 10^{-6}\%$.

   The formula to describe how θ should change over time was solved to be $y = -0.000906754x + 0.155552$ with $S_t = 0.00735659$, $S_r = 4.16014 \times 10^{-7}$, $r^2 = 0.999943$ and the correlation coefficient $= 0.999972$.

4. 5-Point Estimation
   Using the points (0, 32, 64, 96, 127) $\theta(100)$ was predicted to be 0.0650833 radians using the Lagrange Estimate. The true $\theta(100)$ was calculated to be 0.0650833 which results in a true error of $8.30202 \times 10^{-8}\%$.

   The formula to describe how θ should change over time was solved to be $y = -0.000906678x + 0.155591$ with $S_t = 0.00831398$, $S_r = 4.39601e \times 10^{-7}$, $r^2 = 0.999947$ and the correlation coefficient $= 0.999974$.

It can be seen that when using Lagrange Interpolating Polynomials to predict data-points a more accurate result will be achieved when the set of given data-points is larger. When more points are supplied, the interpolating function is able to more accurately predict a point estimation. As the data is constant and linear, the estimation on a less amount of points is sufficient within some tolerance.

When estimating a regression line for θ the two-point estimation achieved the most accurate result. This is due to the data and decrease in θ being reduced at a constant linear rate over time. It can be seen that as more points are being used to estimate the line equation the error increases slightly. If the data was not linear and constant this increase in error will be more evident for points greater than three, and is due to an over-fit of the data.

4 Multiple Linear Regression

4.1 Problem Statement

Flow through a tube is related to the diameter and the angle to the horizontal by the following equation:

$$F = aD^bS^c$$

The following table shows results of nine experiments:

Experiment	D	S	F
1	1	0.001	1.4
2	2	0.001	8.3
3	3	0.001	24.2
4	1	0.01	4.7
5	2	0.01	28.9
6	3	0.01	84.0
7	1	0.05	11.1
8	2	0.05	69.0
9	3	0.05	200.0

Using multiple linear regression find F when D = 2.5 and S = 0.025

4.2 Algorithm Description

The first step for this problem involved filling the Multiple Linear Regression matrix with the $\log_{10}$ values of those given in the above table.

The Multiple Linear Regression matrix is described as follows, where n is the size of the data-set:

$$\begin{bmatrix} n & \sum x_{1i} & \sum x_{2i} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i}x_{2i} \\ \sum x_{2i} & \sum x_{1i}x_{2i} & \sum x_{2o}^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_{1i}y_i \\ \sum x_{2i}y_i \end{bmatrix}$$

Due to the original formula having two powers b and c, the $\log_{10}$ of the original values were calculated and used as per the power rule:

$$y = \alpha_2 x^{\beta_2}$$

$$\Rightarrow \log_{10} y = \beta_2 \log_{10} x + \log_{10} \alpha_2$$

The Gaussian Elimination method was then used on the resulting Multiple Linear Regression matrix to find the constants a, b and c. This method transforms the matrix problem into a triangular system by using forward elimination, the function finally uses backwards substitution to solve for the three unknown constants.

The result returned by the Gaussian Elimination function is a 3×1 matrix containing the now known constants a, b and c. The final step is to remove the $\log_{10}$ from a by raising it to the power of 10 $\Rightarrow a^{10}$.

Finally, F can be found when $D = 2.5$ and $S = 0.025$.

4.3 Complexity

4.3.1 Space Complexity

In total there are 9 data-points for $three$ variables totalling $3n$ space, where n is the amount of datapointas present. In addition, three matrices were initialised to hold the values and results of the Gaussian Elimination method and are of size n, $\frac{n}{3}$and $\frac{n}{3}$ respectively. In total, a complexity of $3n + \frac{2n}{3}$ can be expected and is of order $O(n)$.

4.3.2 Time Complexity

The Multiple Linear Regression function iterates the entire data-set as well as an internal iteration to fill each element in the matrix. The internal iteration is executed $four$ times each iteration and is bounded by the amount of unknown constants present in the problem, 3 in this case. As such, the total complexity for this function can be described as $n \times (4 \times 3)$ and is of order $O(n)$. However, as $(4 \times 3) > n$ for this particular problem, a more accurate complexity can be described in polynomial time.

The Gaussian Elimination function traverses the unknown constant matrix twice for both the forward elimination and backward substitution steps. As such, a total complexity of $n \times n + n \times n = 2n^2$ can be expected and is of order $O(n^2)$.

4.4 Experiments and Results

After performing Gaussian Elimination the unknown constants a, b and c were calculated as $55.9718, 2.61584$ and 0.536779 respectively.

Each experiment was then confirmed with these values of a, b and c and produced the following results:

Experiment	Result	Actual
1	1.37287	1.4
2	8.41547	8.3
3	24.3056	24.2
4	4.72509	4.7
5	28.9639	28.9
6	83.6537	84.0
7	11.2099	11.1
8	68.7148	69.0
9	198.462	200.0

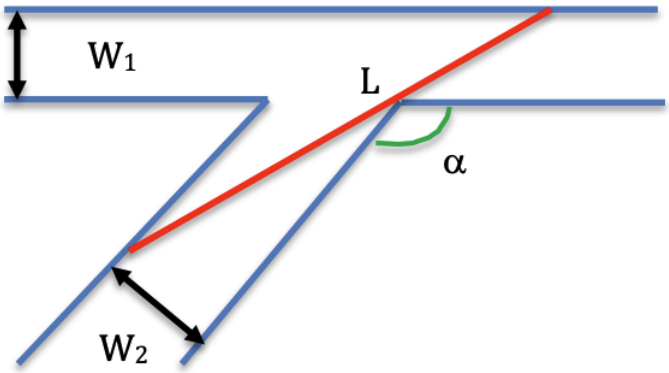
It is noted that the results are close however not exact when compared to the initial values given. It is expected that this is due to calculation errors in round-off within the Gaussian Elimination method as well as when the $\log_{10}$ is performed. If exact results were needed, a different approach such as LU Decomposition should be considered.

Finally, when $D = 2.5$ and $S = 0.025$ $F = 84.9113$.

5 Maximum Length of a plank Around a Corner

5.1 Problem Statement

The maximum length L of a plank that can negotiate the corner shown in the sketch below is a function of w_1, w_2 and α. Write a C++ program to develop a plot of L for a range of a values assuming w_1 and w_2 have fixed values.

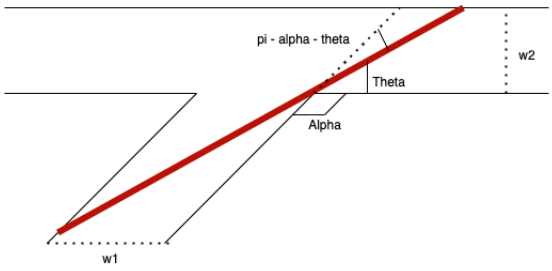


5.2 Algorithm Description

It was noticed that this problem is a trigonometric puzzle. Instead of finding the largest plank that can negotiate a corner, the shortest plank such that the plank is touching both walls and the corner was found. Next, an equation was derived the the length of the plank. It was noticed that the smallest plank such that all three points were being touched was when the plank was at the left corner. When the plank negotiates this corner, it is clear and is therefore a plank that can pass the corner.

In order to negotiate this left corner, a formula for the plank was derived:

$$\frac{w_1}{\sin\theta} + \frac{w_2}{\sin\pi - \alpha - \theta}$$



Therefore, the maximum length plank that can negotiate the right corner for some α is dependant on the theta value such that the plank can negotiate the left corner.

A Golden Section Search method was implemented to find the optimal theta for every α and the results recorded. The Golden Section search (below) is a maximisation search. As we require a minimum, the above function was negated in order to find the minima.

The Golden Section Search uses two points which are known to contain a maxima x_l and x_u as well as 'the golden ratio' $R = 0.61803$. The method systematically reduces x_l and x_u using the below equations until the maxima is found. The function is first evaluated at x_1 and x_2. If $f(x_1) > f(x_2)$ then the domain to the left of x_2 can be eliminated and x_2 becomes the new x_l and x_1 becomes the new x_2 and d is updated. If $f(2) > f(x_1)$ then the domain to the right of x_1 can be eliminated and x_1 becomes the new x_u and x_2 becomes the new x_1 and d is updated.

$$d = R(x_u - x_l)$$
$$x_1 = x_l + d$$
$$x_2 = x_u - d$$

This process repeats until some predefined tolerance is met and will return the maximum of the function.

5.3 Complexity

5.3.1 Space Complexity

The Golden Section Search function operates on several initialised variables and can be considered to operate in constant space. Whilst the remainder of the program does not contribute any run time space it does save the results of every search to a text file. This space requirement is bounded by the amount of steps from 0 to π and can be considered linear.

5.3.2 Time Complexity

The Golden Section Search searches for a maxima until some predefined tolerance is met. As such, the time complexity can be considered linear and is bounded by $O(n)$ where n is the amount of iterations needed to find the maxima.

This function is executed $step = 0.01$ times from 0 to π. As such, a total run time for this program is linear and is bounded by $18000(n)$ or $O(n)$.

5.4 Experiments and Results

To confirm the validity of this algorithm α was plotted from 0 to π and tested across ranging w_1 and w_2 values. When $\alpha = 0$ the length of rod was confirmed to be $w_1 + w_2$ and it was observed that for all different but constant values of w_1 and w_2 the ladder grows exponentially as α approaches π.