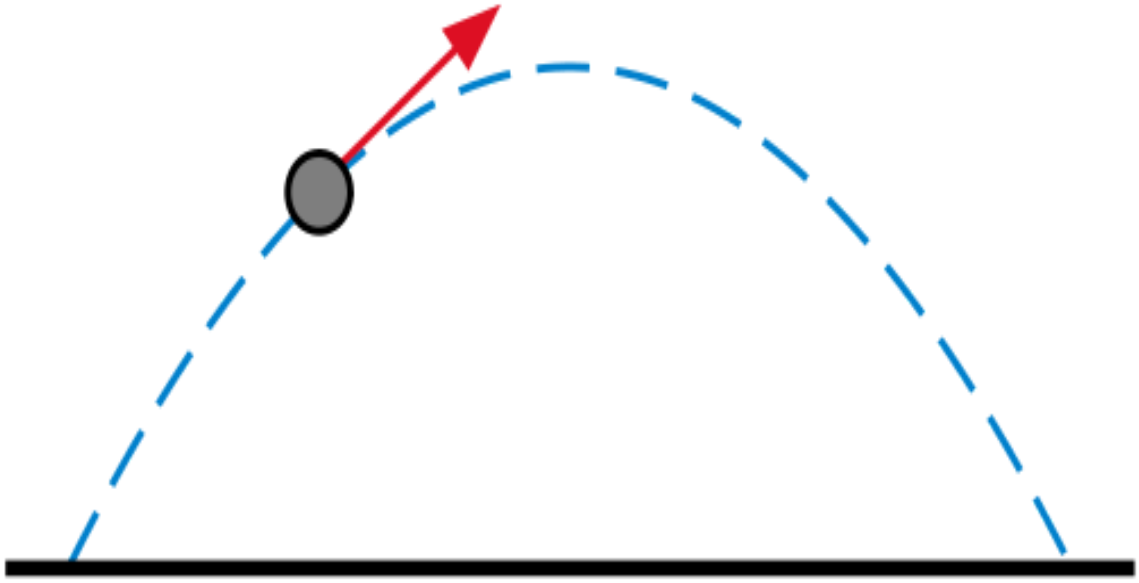
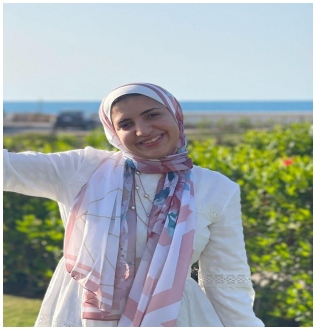
The American University in Cairo
School of Sciences & Engineering
ENGR 3202 - Spring 2021
Dr. ElKhayam Dorra



Group 3: Projectile Toolbox Report

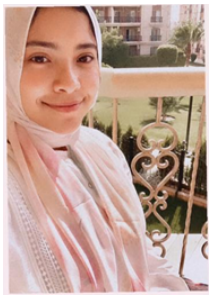
Table Of Contents

List Of Contributors



Ashrakat ElKhalifa (900170051)

- For the first milestone, I conducted O(h^2) method for tool 2 and compute the resultant velocity
- For the second milestone, I worked on understanding the shooting method and shared in applying its code in addition to the rest of tool 5. I also shared in writing the main function of the whole program. Moreover, I wrote the related parts to my work on the report on the section of verification and manual for code.



Nouran Mahmoud (900170863)

- For the first milestone, I understood and wrote code for O(h^4) numerical differentiation in tool 2 and got the resultant angle.
- For the second milestone, I contributed in the construction of tool 5 and I understood the shooting method and how to apply it and shared in writing its code. Also, I shared in writing the main.m file which is the main runner of the program. Finally, I wrote the verification parts for tool 2 and tool 5.

Omar ElAyat (900182568)

- For the first milestone, I coded tool 1, troubleshooted it, tested it on random values with known outputs. Then ran it on the given values in the project statement
- For the second milestone, I coded tool 4, troubleshooted it, and followed the same methodologies of testing I used in the first milestone. I also tested it on the given data and analyzed the results in comparison to the results obtained from online simulations. Moreover, I wrote the parts that explain and analyze the technicalities and the outputs of the two tools I coded.

Ramy ElGendi (900170269)

- For the first milestone, I worked on tool 3, writing and testing the MATLAB code for both the trapezoidal rule and simpson's ⅓ rule, and I created the x vs z graph to display the tool's results.
- For the second milestone, I worked on writing the project's report. I did most of the contents of the report except the parts where every tool developer had to explain specific technicalities about his/her code.

Abstract

This paper aims to create a projectile toolbox while creating MATLAB tools; the project resulted in 5 tools. The first and second tool aimed at calculating the total length of the path of the projectile and the resultant velocity and angle of the projectile versus time respectively, these are calculated given the number of positions x and z versus time of the projectile using numerical tools such as quadratic interpolation, Gauss Jordan, Numerical Integration and Differentiation. The third tool calculates the position (x,y) versus time (t) given the number of points describing velocity (vx, vz) vs time (t) alongside the projectile's path by utilizing numerical integration methods. Fourth tool addresses resultant velocity and angle of the projectile versus time when given the mass, diameter, initial velocity and initial angle of the projectile using ODEs. The last tool combines all the aforementioned tools by using the initial velocity, or initial angle and target position to identify the total length of the projectile path, the position of the projectile at each time step and the resultant velocity and direction at each time step.

Background Review Of Problem

The problem addressed here is firing a projectile with the aim of it landing on a specific target, in this research we disregard the impact of air resistance and use the below differential equations to calculate acceleration. It is important to note that the Vx and VZ are the horizontal and vertical velocity components, t is time and g is gravitational acceleration. Thus the change of horizontal velocity over time is 0 and the change of vertical velocity over time is represented by acceleration of gravity which creates a parabolic trajectory.

$$\frac{dv_x}{dt} = 0 \qquad \frac{dv_z}{dt} = -g$$

By using ordinary differential equations we are able to calculate the initial velocity and initial angle above the horizontal axis which is needed for the projectile to reach the intended target, however, there will exist air resistance when the projectile is targeted at a far target.

Thus, the equations are modified to put into account air resistance and so the model becomes a non-linear boundary value system of ordinary differential equations as seen below.

$$m\frac{dv_x}{dt} = -\frac{1}{2}C_D A_f \rho_{Air} v_x v \qquad m\frac{dv_z}{dt} = -mg - \frac{1}{2}C_D A_f \rho_{Air} v_z v$$

Moreover, the velocity resulting from the projectile is represented by the below equation.

$$v = \sqrt{v_x^2 + v_z^2}$$

_____In the equations above, m is the mass of the projectile and the left hand side of the equation represents the drag force in the direction of i. Af in particular is the silhouette area of the projectile pAir is the density of Air and Cd is a dimensionless drag coefficient. IN a simpler case this drag coefficient can be constant however it is important to know that it depends on shape, velocity and density of air and the projectile. Here, we will proceed with the assumption that the drag coefficient can be correlated to Reynold's number Re as follows.

$$C_D = \frac{24}{Re}, \qquad when\ Re < 0.2$$

$$C_D = \frac{21.12}{Re} + \frac{6.3}{\sqrt{Re}} + 0.25, \qquad when\ 0.2 \le Re < 2000$$

where,

$$Re = \frac{\rho v D}{\mu}$$

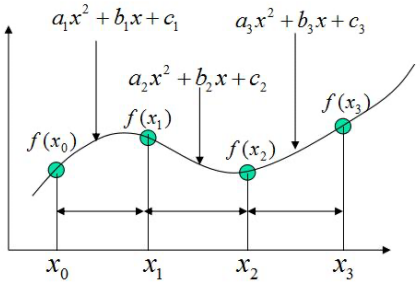
For calculating the length we need, we integrate the equations of the splines in a manner so that each interval according to the arc length formula known is as the equation below which will be used for the functions of the first tool.

$$l = \int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx$$

Background Review Of Methods

Quadratic Splines

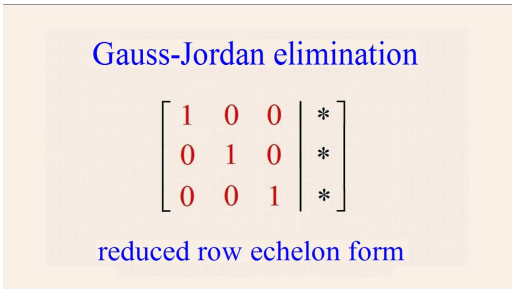
In quadratic splines, each adjacent pairs of points are connected with a quadratic line in form of $ax^2 + bx + c = f(x)$. Here, a set of unknowns a,b, and c will be solved using gauss-jordan elimination code [3].



[1]

Gauss-Jordan Elimination

In Gauss-Jordan, the coefficients of a system of linear equations are transformed into a matrix form and reduced into "reduced row-echelon form" which eliminates the need for back substitution (unlike gauss elimination) [3].



[2]

$O(h^2)$ Numerical Differentiation

This is called the two-point central difference method for the first derivative. It uses one point ahead of X_i and one pint behind it to estimate the first derivative with a higher accuracy (O(h^2)) than the normal forward and backward methods. Formula is: $f'(x) = \frac{f(x+h)-f(x-h)}{2h}$, where h here is the step value [3].

$O(h^4)$ Numerical Differentiation

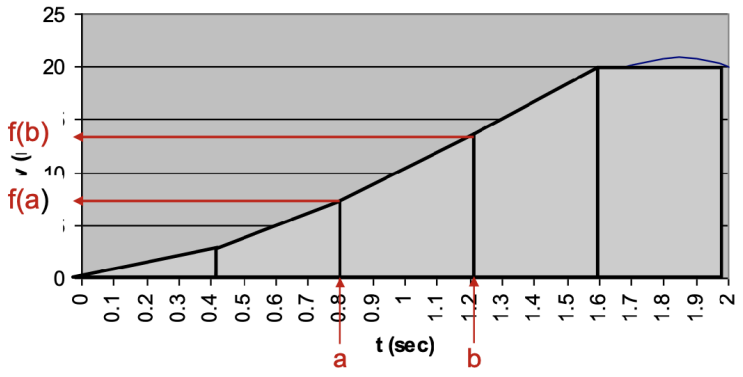
This is called the four-point central difference method for the first derivative. It uses two points ahead of X_i and two pints behind it to estimate the first derivative with a higher accuracy (O(h^4)) than the two-point central difference method. This is the formula used for it:

$$f'(x_i) = \frac{-f(x_{i+2}) + 8f(x_{i+1}) - 8f(x_{i-1}) + f(x_{i-2})}{12h}$$

[3]

Trapezoidal Rule

In trapezoidal rule, we divide up the area under the curve into trapezoids and by estimating the area of the trapezoids, the integral can be evaluated as $Area = (b - a) * \frac{f(a)+f(b)}{2}$. The approximation becomes more and more accurate as the number of partitions increase [3].



[3]

Simpson's ⅓ Rule

This rule is more accurate than the trapezoidal rule since it uses a higher order polynomial to connect the points. In ⅓, the function used is a polynomial of 2nd degree.

$$= \int_a^b f_2(x)dx$$

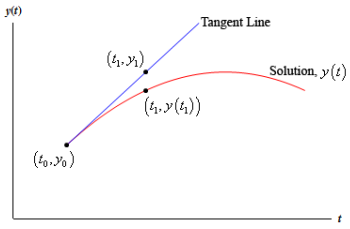
$$= \int_{x_0}^{x_2} \left[\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_o) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2) \right]$$

$$\cong \frac{h}{3}\left[f(x_o) + 4f(x_1) + f(x_2) \right]$$

[3]

Euler's Method

This is a one step method that uses the slope of a function to predict new values of the function at another point. $y_{i+1} = y_i + f(x_i, y_i) * h$, where h is the step value. The truncation error is: $E_a = \frac{f'(x_i, y_i)}{2!} * h^2$[3].



[4]

RK4's Method

This method is also known as the classic Runge-Kutta Method. It provides a means of achieving the accuracy of a Taylor series approach without calculating higher derivatives [3].

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

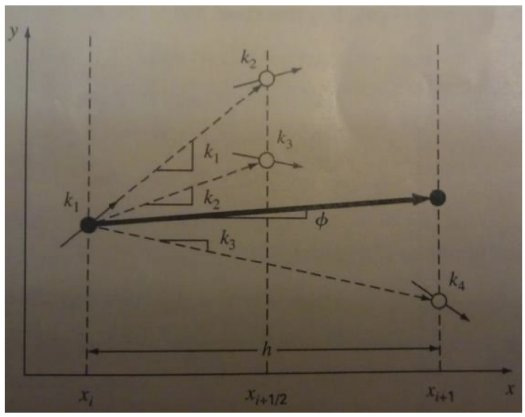
Where:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right)$$

$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h\right)$$

$$k_4 = f\left(x_i + h, y_i + k_3 h\right)$$



[3]

Shooting Method

This method is used to solve boundary value problems by reducing it to a system of initial value problems. After that a trial and error will be implemented to solve for the equations.

Example

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0$$

$$T(0) = 40 \qquad T(10) = 200$$

$$\frac{dT}{dx} = z$$

(E27.1.2)

$$\frac{dz}{dx} = h'(T - T_a)$$

[5]

We will then need to guess for value z(0), we can take it for instance to be 10 and then use euler method or RK4 method to find a solution for all values. Then try again using another guess for z(0) for instance z(0) = 200. We can then use linear interpolation to find a good guess for z(0) that will yield the boundary conditions. [5]

Functions Validation

Tool 1

Tool 1 was divided into 3 functions: Quadratic splines generator, Gauss jordan, and length of the trajectory computer. Each function was tested and validated separately, then the whole tool was gathered and tested.

- For the quadratic splines functions, simple functions with expected behaviors were imputed and the accuracy of the results were compared to the expected results. Ex: x^3 , a*x^2 + b*x+c, e^x , log(x)

  This example was taken from the slides:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 16 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 64 & 8 & 1 \\ 4 & 1 & 0 & -4 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 1 & 0 & -8 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} a_1 \\ b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 4.3 \\ 4.3 \\ 6.1 \\ 6.1 \\ 9.0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}$$

x	f(x)
0	1.0
2	4.3
4	6.1
8	9.0

```
>> t = [ 0 2 4 8]

t =

     0     2     4     8

>> x = [1 4.3 6.1 9]

x =

    1.0000    4.3000    6.1000    9.0000

>> [a,b]=Quad_splines2(t,x)

a =

     0     1     0     0     0     0     0     0
     2     1     0     0     0     0     0     0
     0     0     4     2     1     0     0     0
     0     0    16     4     1     0     0     0
     0     0     0     0     0    16     4     1
     0     0     0     0     0    64     8     1
     1     0    -4    -1     0     0     0     0
     0     0     8     1     0    -8    -1     0


b =

    1.0000    4.3000    4.3000    6.1000    6.1000    9.0000         0         0
```

Note: the first column is assumed to be 0 (a0)

- For the Gauss jordan function, systems of nonlinear equations examples from the slides and the problem sets, with which answers are known, were utilized. The final coefficients matrix was compared with the answers of these examples

  For example, using the same lecture's problem shown above, the splines coefficients were found to be:

  Interval 1:

  $f(x) = 1.65x + 1$

  Interval 2:

  $f(x) = -0.375x^2 + 3.15x - 0.5$

  Interval 3:

  $f(x) = 0.14375x^2 - x + 7.8$

```
>> t = [ 0 2 4 8]

t =

     0     2     4     8

>> x = [1 4.3 6.1 9]

x =

    1.0000    4.3000    6.1000    9.0000

>> coeff(t,x)

ans =

         0    1.6500    1.0000
   -0.3750    3.1500   -0.5000
    0.1437   -1.0000    7.8000
```

- For the length of the trajectory function, the same testing method of the first function was used, simple functions with known lengths were tested, and had their results compared with the known values.

  This part was tested using t = 0:10, x = exp(t)

  Theoretical length = 22025.93

  Measured length:

```
ans =

   2.2026e+04
```

Tool 2

For the validation of tool 2, we used known values from a question in problem set 6 which we solved manually and checked for the result using both O(h^2) and O(h^4).
The question:

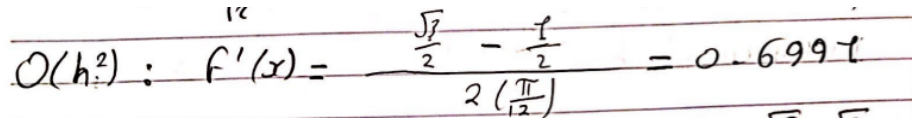
5. *High accuracy Differentiation formulas [20 points]* . Using Taylor series.
(a) Prove the following centered finite difference formula that is $O(h^4)$ for the first derivative

$$f'(x_i)=\frac{-f(x_{i+2})+8f(x_{i+1})-8f(x_i-1)+f(x_{i-2})}{12h}+O(h^4)$$

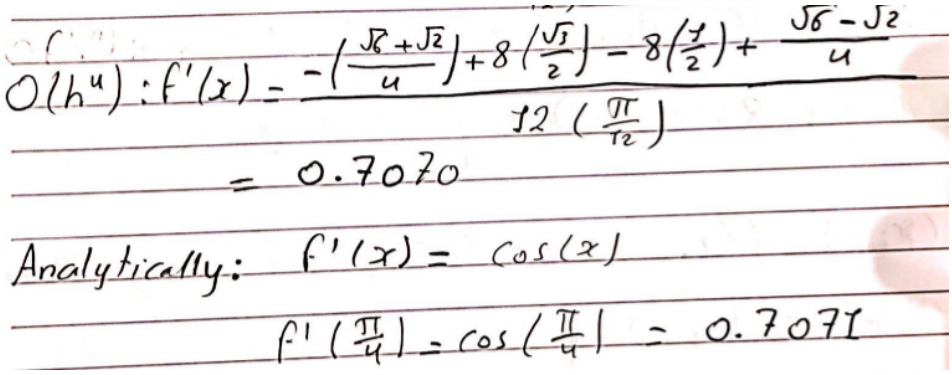
(b) Compute the centered finite difference approximation of $O(h^2)$ and $O(h^4)$ for the first derivative of $y=\sin x$ at $x=\pi/4$ using the value of $h=\pi/12$. Calculate the true percent relative error in both cases.

****

The manual answer for y'(pi/4) using O(h^2):



The manual answer for y'(pi/4) using O(h^2):



The code answers for y'(x) O(h^2):

```
Please Enter tool number from 1 to 5: 2
Please Enter the array t [0 pi/12 pi/6 pi/4 pi/3 5*pi/12]
         0    0.2618    0.5236    0.7854    1.0472    1.3090

Please Enter the array X [0 0.259 0.5 0.707 0.866 0.966]
         0    0.2590    0.5000    0.7070    0.8660    0.9660

Please Enter the array Z [0 0 0 0 0 0]
Please Enter the method (2 for O(h^2) or 4 for O(h^4))2
The velocity in x:
    1.0237    0.9549    0.8556    0.6990    0.4947    0.2693
```

The code answers for y'(x) O(h^4):

```
Please Enter tool number from 1 to 5: 2
Please Enter the array t [0 pi/12 pi/6 pi/4 pi/3 5*pi/12]
         0    0.2618    0.5236    0.7854    1.0472    1.3090

Please Enter the array X [0 0.259 0.5 0.707 0.866 0.966]
         0    0.2590    0.5000    0.7070    0.8660    0.9660

Please Enter the array Z [0 0 0 0 0 0]
Please Enter the method (2 for O(h^2) or 4 for O(h^4))4
warning! This method has a limation for not being able to calculate the velocity and angle for the first and last two points
The velocity in x:
         0         0    0.8652    0.7070         0         0
```

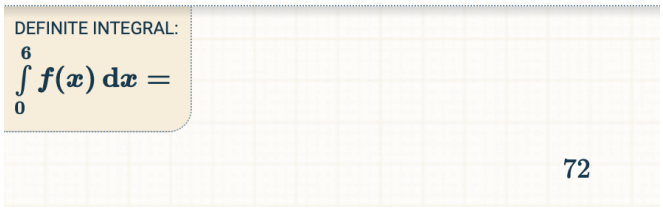
Tool 3

Tool 3 Code & Results:

```
T =   [0     1  2  3   4   5   6];
x =   [0     1  4  9   16  25  36];

disp("Trapezoidal: ");
disp(trapezoidal(x,T));
disp("Simpson's 1/3: ");
disp(simpson1_3(x,T));
```

```
Trapezoidal:
        0    0.5000    3.0000    9.5000   22.0000   42.5000   73.0000

Simpson's 1/3:
        0    0.3333    2.6667    7.0000   21.3333   35.0000   72.0000

>>
```

Online Integral Calculator:



From the results above, after entering plot points for the function x^2 from 0 to 6 in our tool 3's MATLAB code, the final results for trapezoidal was 73, and for simpson's ⅓ was 72. The answer using online integral is 72 which is exactly the same as Simpson's ⅓ results, but close enough to trapezoidal rule's result. This shows us that the simpson's method is more accurate than the trapezoidal since it uses a higher polynomial equation to connect points together.

Tool 3 Code & Results:

```
% Given Data
T =   [0     1  2   3   4   5   6    7   8    9   10];
x =   [0     1  8   27  64  125 216  343 512  729  1000];

disp("Trapezoidal: ");
disp(trapezoidal(x,T));
disp("Simpson's 1/3: ");
disp(simpson1_3(x,T));
```

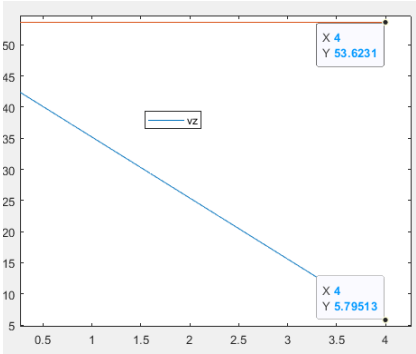
```
Trapezoidal:
   1.0e+03 *

        0    0.0005    0.0050    0.0225    0.0680    0.1625    0.3

Simpson's 1/3:
   1.0e+03 *

        0    0.0003    0.0040    0.0157    0.0640    0.1270    0.3
```

Online Integral Calculator:

From the results above, after entering plot points for the function x^3 from 0 to 10 in our tool 3's MATLAB code, the final results for trapezoidal was 2525, and for simpson's ⅓ was 2500. The answer using online integral is 2500.

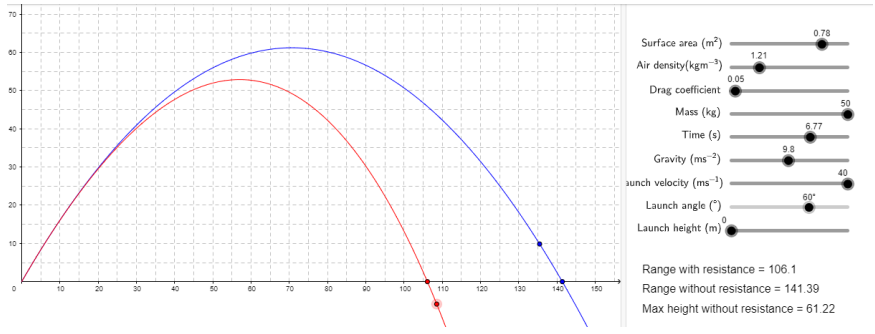
Tool 4

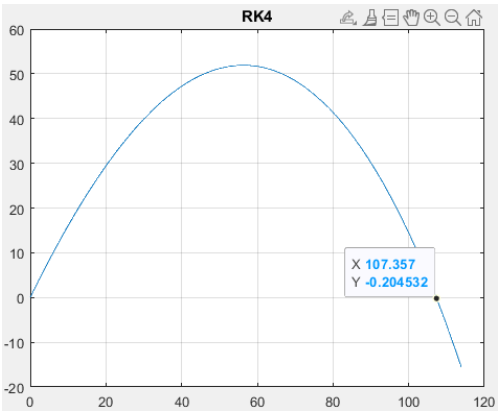
This tool was implemented in phases and each phase was tested separately before putting the whole tool together. First, the projectile velocity was calculated without air resistance, and the results were compared to the results obtained from an online simulation with the same initial criteria.

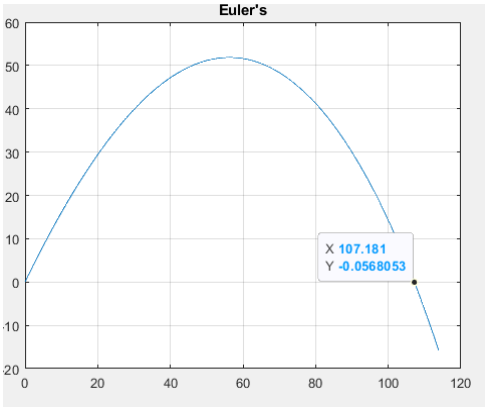


Time	4 sec
Velocity	53.9325 m/s
Horizontal velocity	53.6 m/s
Vertical velocity	5.77 m/s

For euler's and RK4 parts, the outputs of an online simulation (geogebra) with the same initial conditions were compared to both methods results. As, the simulation only shows the position without showing the velocity, Tool3 was used to sketch the path of the projectile and compare its final position with the simulation.







Tool 5

The four tools are verified separately as shown above. As for the shooting method, it is verified after obtaining the initial velocity by using it to x positions at each time and comparing the last value for x position with the targeted value given as an input.

Graphical Flow Chart

Code Manual

Tool 1

This tool is composed of three functions that call one another to execute the code. The user only has to call the function called “fcts” with a 1-D array of x points and 1-D array of z points and a step. The output of the function is the length of the function z(x).
The following example shows an example of the function z = sqrt(x)+x where x is from 0 to 10.

$$\int_0^{10} \sqrt{1+\left(\frac{1}{2\sqrt{x}}+1\right)^2}\,dx$$

Approximation:

16.60022668634096

```
t =

     0     1     2     3     4     5     6     7     8     9    10

>> x = sqrt(t)+t

x =

         0    2.0000    3.4142    4.7321    6.0000    7.2361    8.4495    9.6458   10.8284   12.0000   13.1623

>> fcts(t,x,0.1)

coxm =

         0    2.0000         0
   -0.5858    3.1716   -0.5858
    0.4894   -1.1292    3.7150
   -0.5393    5.0430   -5.5434
    0.5074   -3.3307   11.2041
   -0.5301    7.0441  -14.7329
    0.5129   -5.4715   22.8138
   -0.5265    9.0800  -28.1163
    0.5154   -7.5900   38.5636
   -0.5247   11.1312  -45.6818


ans =

   16.6897
```

The output shown above is the length of the function and the coefficients of the quadratic splines with the As in column 1, Bs in column 2 and Cs in column 3.

Tool 2

You will first need to run the main file. You can enter 2 in order to use tool 2 as it will be shown in the prompt. The program will then ask you to enter array t, array x, array z respectively. Make sure that the three arrays will have the same size otherwise the program will stop and an error will be printed to screen. You also need to make sure the difference between values in the time array are consistent or an error will be printed indicating that h is not consistent. The program will then ask you to enter the method. You can enter 2 for O(h^2) method or 4 for O (h^4) method as it will appear in the prompt. After that, the program will compute Vx, Vz, resultant velocity and angle and output them to console

Input Example Screenshot:

```
Please Enter tool number from 1 to 5: 2
Please Enter the array t [0 0.1 0.2 0.3 0.4 0.5 0.6]
Please Enter the array X [0 2.1 4.3 6.1 8.5 11 13.2]
Please Enter the array Z [0 1.5 2.7 3.8 2.4 0.8 -0.5]
Please Enter the method (2 for O(h^2) or 4 for O(h^4))2
```

Output Example Screenshot:

```
The velocity in x:
   20.5000   21.5000   20.0000   21.0000   24.5000   23.5000   20.5000

The velocity in z:
   16.5000   13.5000   11.5000   -1.5000  -15.0000  -14.5000  -11.5000

The resultant velocities are:
   26.3154   25.3870   23.0705   21.0535   28.7272   27.6134   23.5053

The angles are:
    0.6777    0.5607    0.5218   -0.0713   -0.5494   -0.5528   -0.5112
```

Tool 3

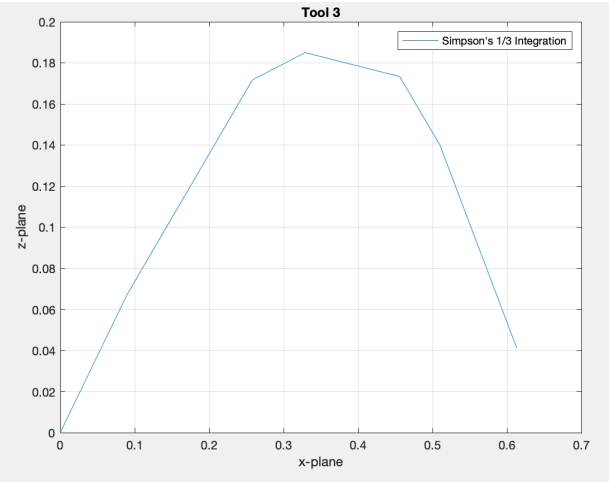
Before running the code, make sure to add the correct data points for x and y to a variable in the following format: “x = [0 23 42 12 …..]”, and add the time to another variable in the format: “t = [0 23 42 12 …..]”. Then call either of the two functions in the following format: “trapezoidal(x,t)” or “simpson1_3(x,t)”.

Attached, there is a demo.m file to demonstrate how to use the functions. All you need to do is change the “T”, “V_x”,”V_z” values, and after running the code, you will be prompted to enter s or t to pick which method to use. Note that V_x is used for x-values, V_z is for z-values, and T is for time values.

Screenshots using demo.m:

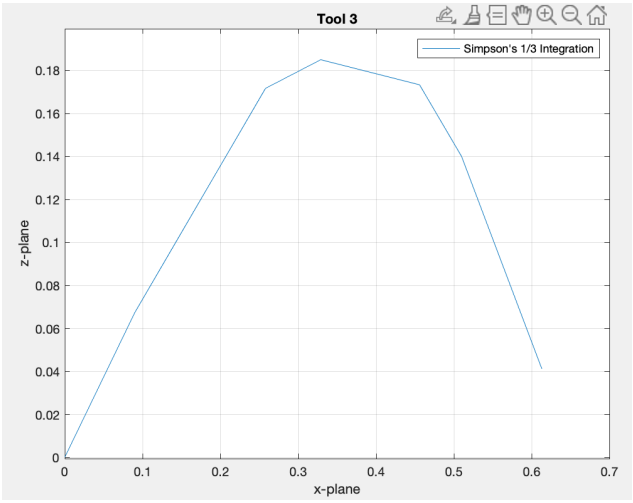
Result For Simpsons ⅓ (1)

```
Do you want to use (Trapezoidal Integration (t) or Simpson's 1/3 Integration (s). Enter t or s:
s

x_s =

         0    0.0900    0.2580    0.3290    0.4560    0.5100    0.6133

z_s =

         0    0.0673    0.1717    0.1850    0.1733    0.1400    0.0410
```



Result For Trapezoidal (2)

```
Do you want to use (Trapezoidal Integration (t) or Simpson's 1/3 Integration (s). Enter t or s:
t

x_t =

         0    0.1350    0.2570    0.3635    0.4550    0.5360    0.6130

z_t =

         0    0.1010    0.1665    0.1865    0.1690    0.1190    0.0355
```



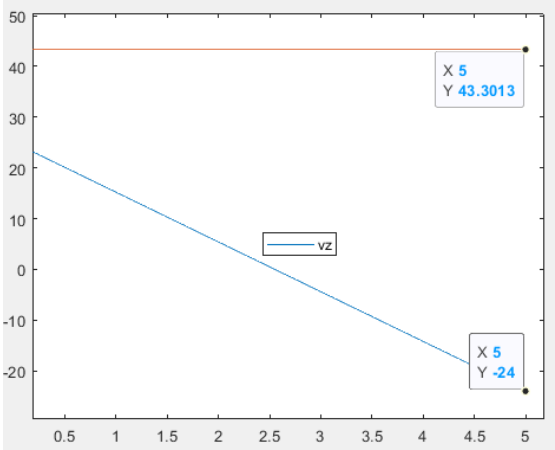
Tool 4

The program asks the user to input the following values, respectively, in order to set the initial conditions of the projectile: initial velocity, Launch angle, time of stoppage, step, mode of air resistance, diameter of the projectile, mass of the projectile. The output is two 1-D arrays of Vx and Vz

Remarks:

- Launch angle is in radians, mass is in Kg, diameter is in m^2
- If the user wants to ignore air resistance, he/she should enter “0”. If they want to incorporate air resistance, they should enter “ 1” for Euler's method and “2” for RK4.

In this example, the user ignored air resistance “0” and set the initial velocity to 50, the angle to 30, the stoppage time at 5 s, and a step of 0.01.



As dvx/dt = 0, the x-component of velocity equals vi*cos(theta) throughout the simulation period. since , vi = 50 and theta= 30. Then, 43.3 which approximately equals the simulated results. Also, as dvz/dt = -9.8 , it was analytically found that vz at t = 5 is -24, which is also exact to the simulated results.

Tool 5

You will first need to run the main file. You can enter 5 to use tool 5 as shown in the prompt. The program will then ask you to enter the initial angle, total time, time step, math diameter, the method for tool 4 whether using air resistance and whether euler or RK4. The program then asks the user to enter 0 for using tool 5 without shooting or 1 to use this tool with shooting method. If the user chooses 0 (with no shooting method) the program will ask the user to enter an initial value for the velocity. The program will then pass this initial velocity and angle

to tool 4 to calculate the vx and vz and then use them to calculate vr and and angle and display them. The program will then pass them tool 3 to calculate the corresponding x and z. Then x and z are passed to the tool to compute the length of the path and the length is printed to the console.

On the other hand, if you chose 1 to use tool 5 with shooting, the program will ask you to enter the value for your target in the x position. It will then use shooting method to estimate the initial velocity and angle and consequently path results to tool 4 followed by tool 3 followed by to calculate the length of the path

Input Example Screenshot:

```
Please Enter tool number from 1 to 5: 5
Please Enter the initial angle pi/4
Please Enter the total time 0.6
Please Enter h value (time step)0.1
Please Enter the mass 10
Please Enter the diameter 0.05
Please Enter whether you want resistance or not (0 for no air resistance or 1 for Euler or 2 for RK4)2
Please Enter whether you want to use shooting method or not (0 for no shooting or 1 for shooting)0
Please Enter the initial velocity25
```

Output Example Screenshot:

```
The total length of the projectile is:
   16.2293
```

Technical Challenges

For Tool 3, there is a truncation error for both trapezoidal rule and simpson's ⅓. There is also an approximation error since we are taking results for up to 4 decimal places.

Trapezoidal Truncation Error:

$$E_a = -\frac{(b-a)^3}{12} f''(\xi)$$

$f''(\xi)$ is a measure of linearity

$a < \xi < b$ [3]

Simpson's ⅓ Truncation Error:

$$E_a = -\frac{(b-a)^5}{2880} f^{(4)}(\xi)$$ [3]

Conclusion

The functions created resulted in the success of the five tool objectives thus, the research succeeded in measuring total length of path of projectile, the resultant velocity and angle versus time, the position against time and hitting a specific target by varying the information given by the user. And while the tools and functions took into account air resistance, the application of such a model in real life is more complex and includes more variables that are constantly interchangeable, for example air resistance cannot always be easily calculated, gravitational forces may change from one area or one height to another, viscosity of air in the atmosphere is also something that can be varying and hard to calculate. Thus, while the used functions and tools strive to produce the best possible results, there remains a margin of error and a chance for increased accuracy given more complex tools that would calculate all needed variables given the circumstances of conducting the experiment.

References

[1] https://www.youtube.com/watch?v=UhhfKXBnnEo

[2] https://www.programmersought.com/article/47074724055/

[3] Doctor ElKhayam Dorra's slides for Engineering Analysis and Computation, Spring 2021.

[4] https://tutorial.math.lamar.edu/classes/de/eulersmethod.aspx

[5] https://textflow.mheducation.com/parser.php?secload=2