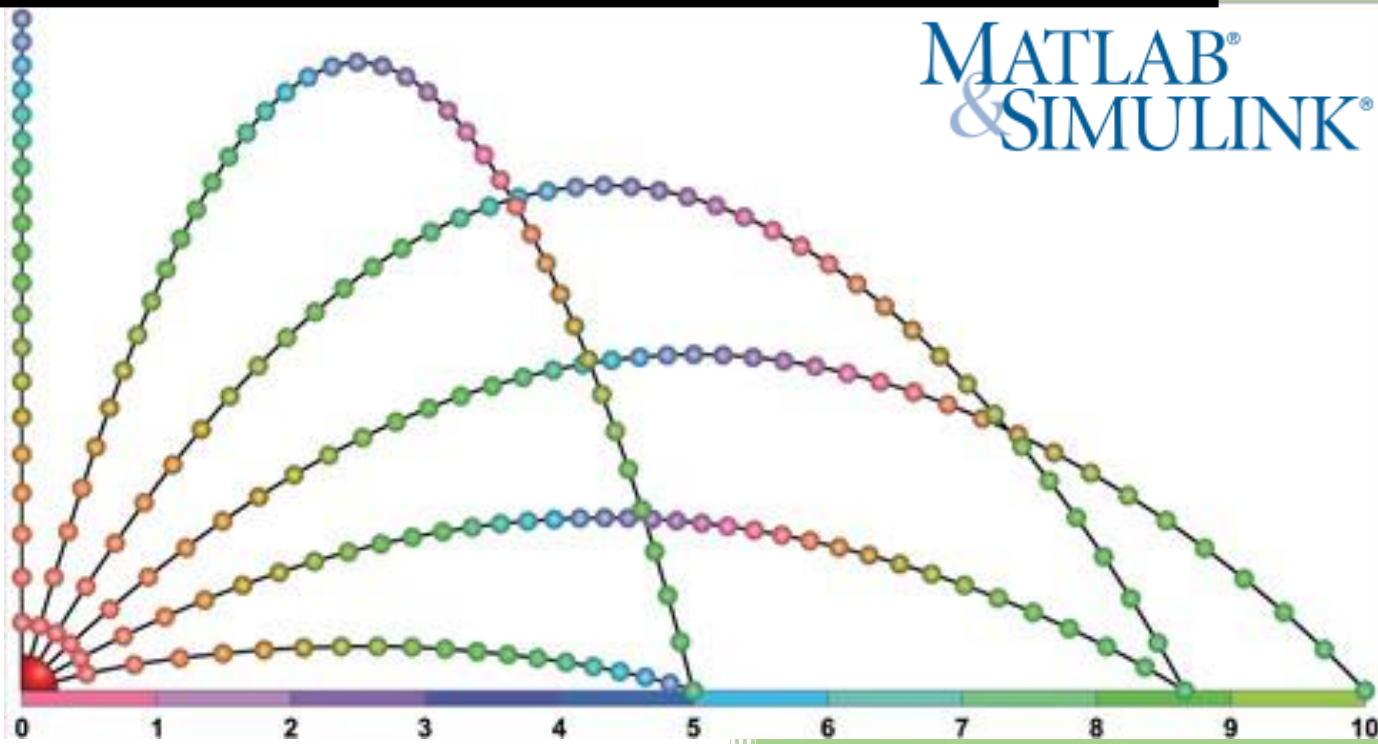


2021 Spring

“Projectile Toolbox”



The American University in Cairo

ENGR3202 Project

Engineering Analysis & Computation I

Under the Supervision of

Dr. ElKhayam Dorra

Abstract

When it comes to firing or casting an object, we aim to be able to precisely decide on its projectile path and landing position. In standard conditions, the projectile equations become so simple after ignoring the effects of air resistance. However, when it comes to firing an object for a long distance, or when projectile's accuracy must be so high; effects of air resistance are too large to be ignored. With things like the frontal area of projectile, density of air, and drag coefficient controlling the impact of air resistance, it becomes too hard to combine all that data and variables manually, but instead using computers for simulations and data presentation. So, with using MATLAB software package, it would aid us hugely with visualizing the expected behavior of the projectile.

In this paper, we would be developing a projectile toolbox that can calculate the total length of the projectile path, calculating the projectile's angle and velocity regarding the flight time, calculating the exact position of the fired object at any certain time, and then combining this all together with applying the shooting method to hit a specific landing point. However, the present codes in this paper are adequately short and well-documented those they could be modified and adapted to according to the main purpose of the user.

Keywords: Computational Simulation, Projectile Motion, MATLAB, Shooting Method.

Background review of the problem

When we fire a projectile such that it hits a specific target. We usually ignore air resistance, that is why we end up with 2 simple differential equations describing the acceleration.

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

The acceleration vector in the horizontal direction equals to 0 in respect to time, while the acceleration component in vertical direction equals to the magnitude of the gravitational acceleration. As these equations would mostly result in parabolic trajectory.

With this set of ordinary differential equations, the initial velocity and initial angle above the horizontal which are necessary for the projectile to hit a specific landing point could be calculated. But when Air resistance value is huge to be neglected, these equations get much more complicated such that.

The nonlinear boundary value equation in Z direction would be:

$m \frac{dv_z}{dt} = -\frac{1}{2} C_D A_f \rho_{Air} v_z v$, whereas m is the mass of the projectile, and all the right side is called the drag force.

On the other hand, the equation in X direction would be:

$m \frac{dv_x}{dt} = -\frac{1}{2} C_D A_f \rho_{Air} v_x v$, A_f refers to the frontal area of projected object, ρ_{Air} is the density of the air, and C_D is drag coefficient. Although we can assume the drag coefficient to be constant in simple cases, in other ones, it can be correlated to Reynold's number (Re) following the following equations:

$C_D = \frac{24}{Re}$ only when Re is less than 0.2. However, when Re is less than 2000 and more than or equal to 0.2. The equation becomes, $C_D = \frac{21.12}{Re} + \frac{6.3}{\sqrt{Re}} + 0.25$. In all these equations, we assume Re to equal $\frac{\rho v D}{\mu}$, with ρ referring to the density of the air (the medium) which equals to 1.204 Kg/m³, and the μ to be the viscosity of the air in the atmosphere with value of 18.13 Kg/m-s. And v to be the resultant velocity of the system calculated by $\sqrt{v_x^2 + v_z^2}$ and D to be the diameter of the projected object.

Background on Methods

Numerical Differentiation

Employ Taylor series expansions to derive finite-divided-difference approximations of derivatives. Uses forward, backward, or centered approximations to estimate differentials, with the centered method being the most accurate.

Algorithm:

- Taking n data points where n should be equal to or more than 2.
- The used numerical differentiation could be $O(h^2)$ or $O(h^4)$ methods.
- If method= $O(h^2)$, the first centered difference method is used for all center points, however, for the first and last points, the forward and backward difference methods are used, respectively. All these methods give $O(h^2)$ error.
- if method= $O(h^4)$. the second centered difference method is used for all center points $O(h^4)$. For the second and second last points, the first centered difference method is used. For the first and last points the forward and backward difference methods are used, respectively. These methods give $O(h^2)$ error which is the maximum achievable accuracy.
- Below is a list of equations used:

Forward difference method: $O(h^2)$

$$f'(x_i) = \frac{-f(x_i + 2) + 4f(x_i + 1) - 3f(x_i)}{2h}$$

Backward difference method: $O(h^2)$

$$f'(x_i) = \frac{3f(x_i) - 4f(x_{i-1}) + f(x_{i-2})}{2h}$$

First centered difference method: $O(h^2)$

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

First centered difference method: $O(h^4)$

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

Gauss-Jordan

It is a method that can be used to solve systems of linear equations. It relies upon the three elementary row operations to convert matrix into reduced-row echelon form. The conditions that should be satisfied are: 0's rows are at the bottom of the matrix, first non-zero element in each row "pivot" is to the right of the leading entry of the row above it and all other entries in the column containing a leading 1 are 0's.

Algorithm:

- Apply partial pivoting, i.e., searching for the row with the biggest coefficient in the "k" column.

Projectile Toolbox

- Store row number.
- Swapping elements of wrong "pivot" row and the row with the biggest coefficient in the same column.
- Modifying coefficients above and below pivot coefficient. The elements in the row above or lower to the pivot are set to zero and other elements in the same rows are subtracted by: the value in the pivot column in the same row multiplied by the value in the column above the currently being modified element.

Numerical Integration

It is a method that can be used to estimate the value of definite integrals when it's hard to solve the closed form of this integral; and when an approximate answer for this integral would be enough. There are many different types of integration method, however the most used techniques for numerical integration are the midpoint rule, trapezoidal rule, and Simpson's rule. solve systems of linear equations. In our code, we would be only using Simpson's 1/3 rule and trapezoidal rule.

Algorithm:

A) Simpson's 1/3 rule

- Defining the Interval Width - dividing the interval into equal steps.
- Since MATLAB indices starts with one not zero, even will start from 3 and odd starts from 2.
- Add even sets, then Add odd sets.

B) Trapezoidal rule

- Defining the Interval Width - dividing the interval into equal steps.
- Calculating the summation of intermediate points.
- Calculating the total area of all trapezoids "Multiple Trapezoidal Rule".

Euler's Method

It is a very common and reliable method mathematics and computational science that can be used to approximate the value of differential equations. When it comes to the ordinary differential equations with a given initial value, a first order numerical procedure of Euler's Method could be used.

Algorithm:

- Numerically integrate above from $x = a$ to $x = b$ using the given step size (with a and b being the boundary points).
- Defining the function that would be integrated, and store the resulted function in a new variable.
- Then we are going to loop the code from 1 to $n-1$ with the defined step size for X values, and use a function feval which relies on the main function and the value of an increment $x(i)$ to find y value across the graph.
- Then repeat the same steps as of last one, but instead in the integrated function instead

Shooting Method

When it comes to understanding the Shooting Method, topics such ODE boundary value and PDE are directly related and crucial in understanding the main concept. Although it's not a usual thing to face problems with initial conditions, when we have a system of ODE with derivative value or/and solutions at more than one point, let's say two points, we call this system a two-point boundary value problem. However, at more complicated levels, like wave equations and shooting or marching methods, the problems may have n number of initial conditions (or boundary conditions) and we call it nth order boundary-value or initial-value equation.

The Concept:

A) Boundary-Value problem

The Standard shape of Boundary Value Problem is shown below, in which we also assume the value of t to be within the range of a and b (exclusive)

$$y'' = f(t, y, y')$$

In this form, we can again assume f to be the following, then we can notice that partial derivative of f in respect to y equals to $q(t)$, while the partial derivative of f in respect to y' equals to $p(t)$

$$f(t, y, y') = p(t) y' + q(t) y + r(t)$$

The partial derivatives are continuous if there is a constant M for $q(t)$ and $p(t)$ that $q(t) > 0$ for all t belongs to $[a, b]$ and $|p(t)| \leq M$. Then f has a unique solution that $y = y(t)$ on $a \leq t \leq b$.

B) Linear Shooting Problem

To solve such types of problems effectively, we would be using Runge–Kutta scheme. Such problems are usually subjected to boundary conditions, that decides on how the function is dealt with. Let us take the following equation as an example,

$$\frac{d^2 y}{dx^2} + y = 0$$

With conditions that, $y(a) = A$ and $y(b) = B$, then we gonna assume that $y(0) = \alpha$, where we are free to choose α for now, we then integrate from 0 to $\frac{\pi}{3}$: if $y(\frac{\pi}{3})$ is not equal to $1/2$ then we can adjust the value of α . In order to do this, we use the Newton–Raphson scheme (Discussed later on).

Functions Validation

Splines

TABLE 18.1

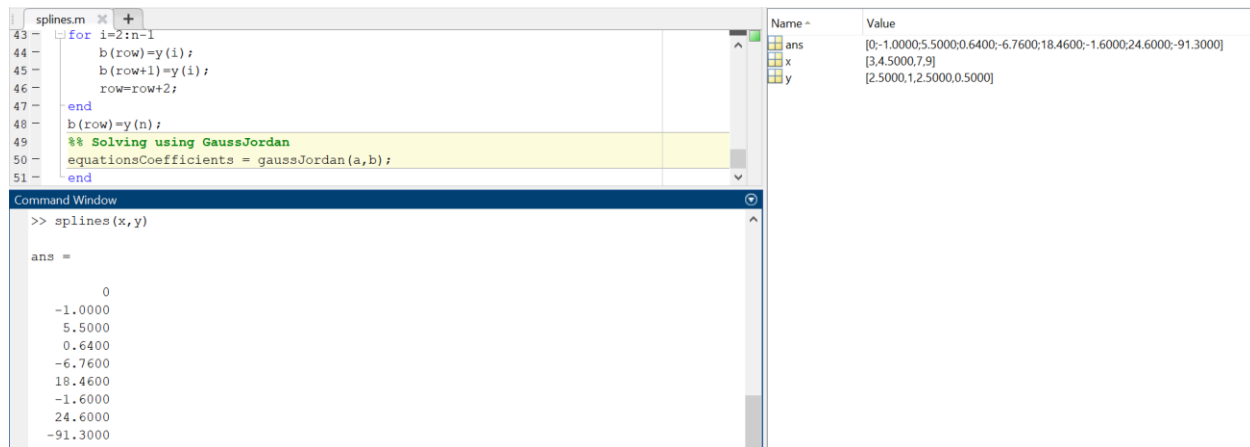
Data to be fit with spline functions.

| x | $f(x)$ |
|-----|--------|
| 3.0 | 2.5 |
| 4.5 | 1.0 |
| 7.0 | 2.5 |
| 9.0 | 0.5 |

These equations can be solved using techniques from Part Three, with the results:

$$\begin{aligned}
 a_1 &= 0 & b_1 &= -1 & c_1 &= 5.5 \\
 a_2 &= 0.64 & b_2 &= -6.76 & c_2 &= 18.46 \\
 a_3 &= -1.6 & b_3 &= 24.6 & c_3 &= -91.3
 \end{aligned}$$

Fig. Spline Interpolation



Numerical Differentiation

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

$$f'(0.5) = \frac{0.6363 - 1.1035}{2 * 0.25} = -0.9344$$

Projectile Toolbox

```

differential.m
17-         vx(i)=(x(i+1)-x(i-1))/(2*h);
18-     end
19-     elseif method==4
20-         for i=3:n-2
21-             vx(i)=(-x(i+2)+8*x(i+1)-8*x(i-1)+x(i-2))/(12*h);
22-         end

```

Command Window

```

x =
    1.2000    1.1035    0.9250    0.6363    0.2000

>> t=[0,0.25,0.5,0.75,1]

t =
    0    0.2500    0.5000    0.7500    1.0000

>> differential(t,x,2)

ans =
   -0.2219   -0.5500   -0.9344   -1.4500   -2.0406

```

fx >> |

$$O(h^2)$$

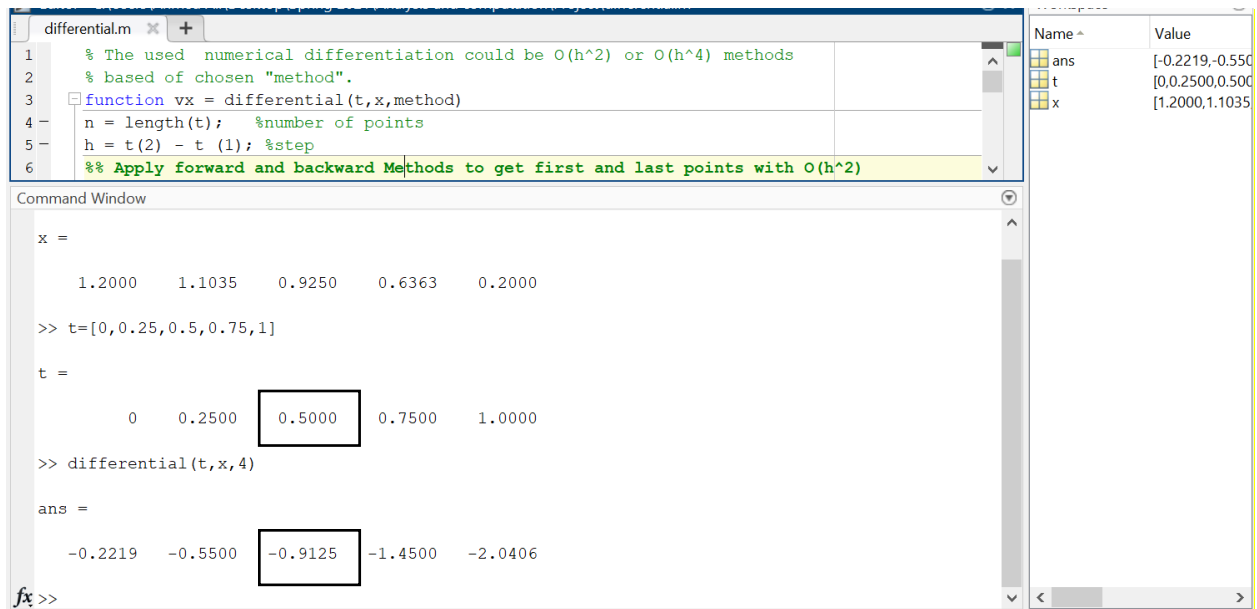
Solution. The data needed for this example is

| | |
|------------------|--------------------------|
| $x_{i-2} = 0$ | $f(x_{i-2}) = 1.2$ |
| $x_{i-1} = 0.25$ | $f(x_{i-1}) = 1.1035156$ |
| $x_i = 0.5$ | $f(x_i) = 0.925$ |
| $x_{i+1} = 0.75$ | $f(x_{i+1}) = 0.6363281$ |
| $x_{i+2} = 1$ | $f(x_{i+2}) = 0.2$ |

The centered difference of accuracy $O(h^4)$ is computed as (Fig. 23.3)

$$f'(0.5) = \frac{-0.2 + 8(0.6363281) - 8(1.1035156) + 1.2}{12(0.25)} = -0.9125 \quad \varepsilon_t = 0\%$$

Projectile Toolbox



$O(h^4)$

Gauss-Jordan

Gauss-Jordan Method

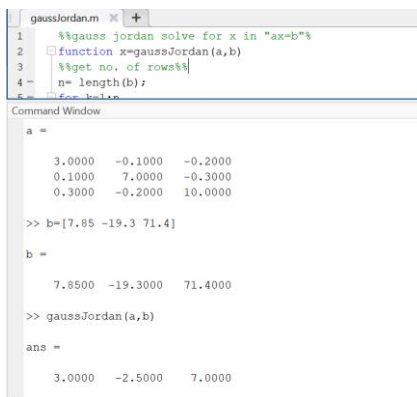
Problem Statement. Use the Gauss-Jordan technique to solve the same system as in Example 9.5:

$$\begin{aligned} 3x_1 - 0.1x_2 - 0.2x_3 &= 7.85 \\ 0.1x_1 + 7x_2 - 0.3x_3 &= -19.3 \\ 0.3x_1 - 0.2x_2 + 10x_3 &= 71.4 \end{aligned}$$

Finally, the x_3 terms can be reduced from the first and the second equations to give

$$\begin{bmatrix} 1 & 0 & 0 & 3.0000 \\ 0 & 1 & 0 & -2.5000 \\ 0 & 0 & 1 & 7.0000 \end{bmatrix}$$

Fig. Gauss-Jordan



Numerical Integration (Trapezoidal Rule)

Multiple-Application Trapezoidal Rule

Problem Statement. Use the two-segment trapezoidal rule to estimate the integral of

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

from $a = 0$ to $b = 0.8$. Employ Eq. (21.13) to estimate the error. Recall that the correct value for the integral is 1.640533.

Solution. $n = 2$ ($h = 0.4$):

$$f(0) = 0.2 \quad f(0.4) = 2.456 \quad f(0.8) = 0.232$$

$$I = 0.8 \frac{0.2 + 2(2.456) + 0.232}{4} = 1.0688$$

```
trapezoidalRule.m  x +
1  %% Calculates integral for a given set using Trapezoidal Rule
2  function I = trapezoidalRule (x,y)
3  n=length(x);
4  a=x(1);
5  b=x(n);

Command Window

>> x=[0,0.4,0.8]

x =

    0    0.4000    0.8000

>> y=[0.2,2.456,0.232]

y =

    0.2000    2.4560    0.2320

>> trapezoidalRule(x,y)

ans =

    1.0688
```

Numerical Integration (Simpson's 1/3 Rule)

Multiple-Application Version of Simpson's 1/3 Rule

Problem Statement. Use Eq. (21.18) with $n = 4$ to estimate the integral of

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

from $a = 0$ to $b = 0.8$. Recall that the exact integral is 1.640533.

Solution. $n = 4$ ($h = 0.2$):

$$f(0) = 0.2 \quad f(0.2) = 1.288$$

$$f(0.4) = 2.456 \quad f(0.6) = 3.464$$

$$f(0.8) = 0.232$$

From Eq. (21.18),

$$I = 0.8 \frac{0.2 + 4(1.288 + 3.464) + 2(2.456) + 0.232}{12} = 1.623467$$

```
simpsRule.m  x +
6  w = (b-a)/(n-1); % Interval Width - dividing the inter
7  evenSum=0;
8  oddSum=0;

Command Window

>> y=[0.2,1.288,2.456,3.464,0.232]

y =

    0.2000    1.2880    2.4560    3.4640    0.2320

>> x=[0,0.2,0.4,0.6,0.8]

x =

    0    0.2000    0.4000    0.6000    0.8000

>> simpsRule(x,y)

ans =

    1.6235
```

ODE (Euler's Method)

Euler's Method

Problem Statement. Use Euler's method to numerically integrate Eq. (PT7.13):

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8.5$$

25.1 EULER'S METHOD

from $x = 0$ to $x = 4$ with a step size of 0.5. The initial condition at $x = 0$ is $y = 1$. If that the exact solution is given by Eq. (PT7.16):

$$y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

Solution. Equation (25.2) can be used to implement Euler's method:

$$y(0.5) = y(0) + f(0, 1)0.5$$

where $y(0) = 1$ and the slope estimate at $x = 0$ is

$$f(0, 1) = -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$$

Therefore,

$$y(0.5) = 1.0 + 8.5(0.5) = 5.25$$

| Name | Value |
|------|--------|
| ans | 5.2500 |
| fx | 8.5000 |
| h | 0.5000 |
| x0 | 0 |
| x1 | 0.5000 |
| y0 | 1 |

```

>> fx=8.5

fx =

    8.5000

>> eulerMethod(h,x1,x0,y0,fx)

ans =

    5.2500
    
```

ODE (RK4 Method)

Compare both values at $t=1$.

There is a small difference due to rounding values in MATLAB (rounding error) but they yield the same exact results up to third decimal place.

RK4 Method

$$\frac{dv_x}{dt} = -\frac{1}{2} C_D \rho_f \frac{v_{air}^2}{m} - V_{xc} V$$

$C_D = 0.2569$ $\rho_f = 1.204$
 $\rho_f = 0.0491$ $V = 50$
 $V_{xc} = 35.3553$ $m = 30$ $h = 0.5$

$K_1 = f(0, 35.3553) = -0.1265583597$
 $K_2 = f(0.5, 35.36162792) = -0.371689413$
 $K_3 = f(0.5, 35.16948153) = -0.4474508774$
 $K_4 = f(0.5, 34.90784912) = -0.4474508774$

$V_{x2+1} = 35.3553 + \frac{1}{6} (K_1 + 2K_2 + 2K_3 + K_4)$
 $V_{x2+1} = 35.3553 + \frac{1}{6} (2.51218475)$
 $V_{x2+1} = 34.93660254 \approx 34.937$

Matlab Solution $\rightarrow 34.9374$
 ≈ 34.937

| | | | | | | | | | | |
|--------------------------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Time | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Velocity in x-direction in m/s | 35.3553 | 34.9374 | 34.5789 | 34.2622 | 33.9654 | 33.6647 | 33.3393 | 32.9747 | 32.5624 | 32.0990 |

Projectile Toolbox

Toolbox Running Without Taking Air Resistance into Account.

Problem 1

An object is launched at a velocity of 20 m/s in a direction making an angle of 25° upward with the horizontal.

b) The time of flight is the interval of time between when projectile is launched: t_1 and when the projectile touches the ground: t_2 . At $t = t_1$ and $t = t_2$, $y = 0$ (ground). Hence

$$V_0 \sin(\theta) t - (1/2) g t^2 = 0$$

Solve for t

$$t(V_0 \sin(\theta) - (1/2) g t) = 0$$

two solutions

$$t = t_1 = 0 \text{ and } t = t_2 = 2 V_0 \sin(\theta) / g$$

$$\text{Time of flight} = t_2 - t_1 = 2 (20 \sin(25^\circ)) / 9.8 = \boxed{1.72 \text{ seconds}}$$

d) The object hits the ground at $t = t_2 = 2 V_0 \sin(\theta) / g$ (found in part b above)

The components of the velocity at t are given by

$$V_x = V_0 \cos(\theta) \quad V_y = V_0 \sin(\theta) - g t$$

The components of the velocity at $t = 2 V_0 \sin(\theta) / g$ are given by

$$V_x = V_0 \cos(\theta) = 20 \cos(25^\circ) \quad V_y = V_0 \sin(25^\circ) - g (2 V_0 \sin(25^\circ) / g) = -V_0 \sin(25^\circ)$$

The magnitude V of the velocity is given by

$$V = \sqrt{V_x^2 + V_y^2} = \sqrt{(20 \cos(25^\circ))^2 + (-V_0 \sin(25^\circ))^2} = V_0 = \boxed{20 \text{ m/s}}$$

c) In part c) above we found the time of flight $t_2 = 2 V_0 \sin(\theta) / g$. The horizontal range is the horizontal distance given by x at $t = t_2$.

$$\text{range} = x(t_2) = V_0 \cos(\theta) t_2 = 2 V_0 \cos(\theta) V_0 \sin(\theta) / g = V_0^2 \sin(2\theta) / g = 20^2 \sin(2(25^\circ)) / 9.8 = \boxed{31.26 \text{ meters}}$$

Find the maximum height by substituting t by 0.86 seconds in the formula for y

$$\text{maximum height } y(0.86) = 20 \sin(25^\circ)(0.86) - (1/2) (9.8) (0.86)^2 = \boxed{3.64 \text{ meters}}$$

problemsphysics.com

MATLAB

```
-----Tool4-----
*****
i, this is tool 4 in projectile toolbox
ool 4 computes Resultant velocity and angle of the projectile versus time using ODEs
nter Initial Velocity in m/s: 20
nter Initial Angle in degrees: 25
nter Travel Time for The Projectile in seconds: 1.72
ow many Vx and Vz values you want to identify in this interval? 9
o you want to print vx or vz?
nter 1 for vx
nter 2 for vz
nter 3 for vx and vz
nter any other number to neglect printing either

f you want to print resultant velocity and angle at each time stamp enter 11
o you want to account for air resistance?
nter 1 To ignore air resistance
nter 2 To incorporate air resistance
```

| | | | | | | | | | |
|--------------------------------|---------|---------|---------|---------|---------|---------|----------|----------|----------|
| Time | 0 | 0.2150 | 0.4300 | 0.6450 | 0.8600 | 1.0750 | 1.2900 | 1.5050 | 1.7200 |
| Velocity in x-direction in m/s | 18.1262 | 18.1262 | 18.1262 | 18.1262 | 18.1262 | 18.1262 | 18.1262 | 18.1262 | 18.1262 |
| Velocity in z-direction in m/s | 8.4524 | 6.3432 | 4.2341 | 2.1249 | 0.0158 | -2.0934 | -4.2025 | -6.3117 | -8.4208 |
| Resultant Velocity in m/s | 20.0000 | 19.2040 | 18.6141 | 18.2503 | 18.1262 | 18.2466 | 18.6070 | 19.1936 | 19.9867 |
| Angle in degrees | 25.0000 | 19.2874 | 13.1479 | 6.6862 | 0.0498 | -6.5879 | -13.0534 | -19.1986 | -24.9181 |

Projectile Toolbox

-----Tool3-----

Hi, this is tool 3 in projectile toolbox

Tool 3 computes Position (x,z) versus time (t)

Which Method do you want to use

Enter 1 to pick Trapezoidal rule.

Enter 2 to pick Simpson's 1/3 rule.

2

Time

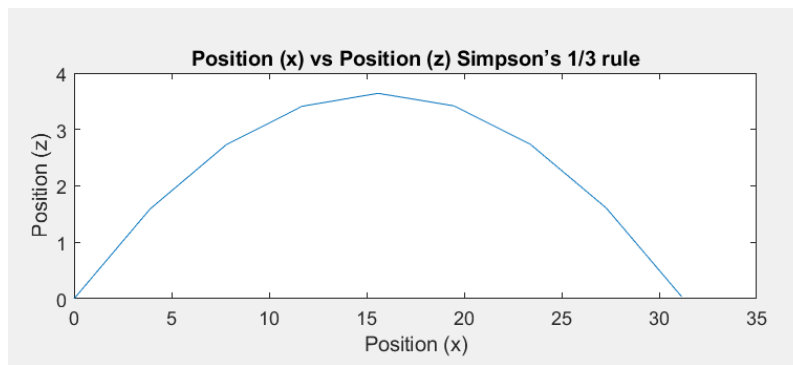
| | | | | | | | | |
|---|--------|--------|--------|--------|--------|--------|--------|--------|
| 0 | 0.2150 | 0.4300 | 0.6450 | 0.8600 | 1.0750 | 1.2900 | 1.5050 | 1.7200 |
|---|--------|--------|--------|--------|--------|--------|--------|--------|

X-position

| | | | | | | | | |
|---|--------|--------|---------|---------|---------|---------|---------|---------|
| 0 | 3.8971 | 7.7942 | 11.6914 | 15.5885 | 19.4856 | 23.3827 | 27.2799 | 31.1770 |
|---|--------|--------|---------|---------|---------|---------|---------|---------|

Z-position

| | | | | | | | | |
|---|--------|--------|--------|--------|--------|--------|--------|--------|
| 0 | 1.5905 | 2.7276 | 3.4112 | 3.6413 | 3.4180 | 2.7411 | 1.6109 | 0.0271 |
|---|--------|--------|--------|--------|--------|--------|--------|--------|



-----Tool1-----

Hi, this is tool 1 in projectile toolbox

Tool 1 computes the total length of the path of the projectile in meters

Total Length of the projectile's path in m is:

32.2846

-----Tool2-----

Hi, this is tool 2 in projectile toolbox

Tool 2 computes the resultant velocity and angle of the projectile versus time

Choose Degree of Error: 2 for $O(h^2)$ and 4 for $O(h^4)$

if you want to calculate resultant velocity enter 1

if you want to calculate angle of the projectile enter 2

if you want to calculate both enter 3

3

Time

| | | | | | | | | |
|---|--------|--------|--------|--------|--------|--------|--------|--------|
| 0 | 0.2150 | 0.4300 | 0.6450 | 0.8600 | 1.0750 | 1.2900 | 1.5050 | 1.7200 |
|---|--------|--------|--------|--------|--------|--------|--------|--------|

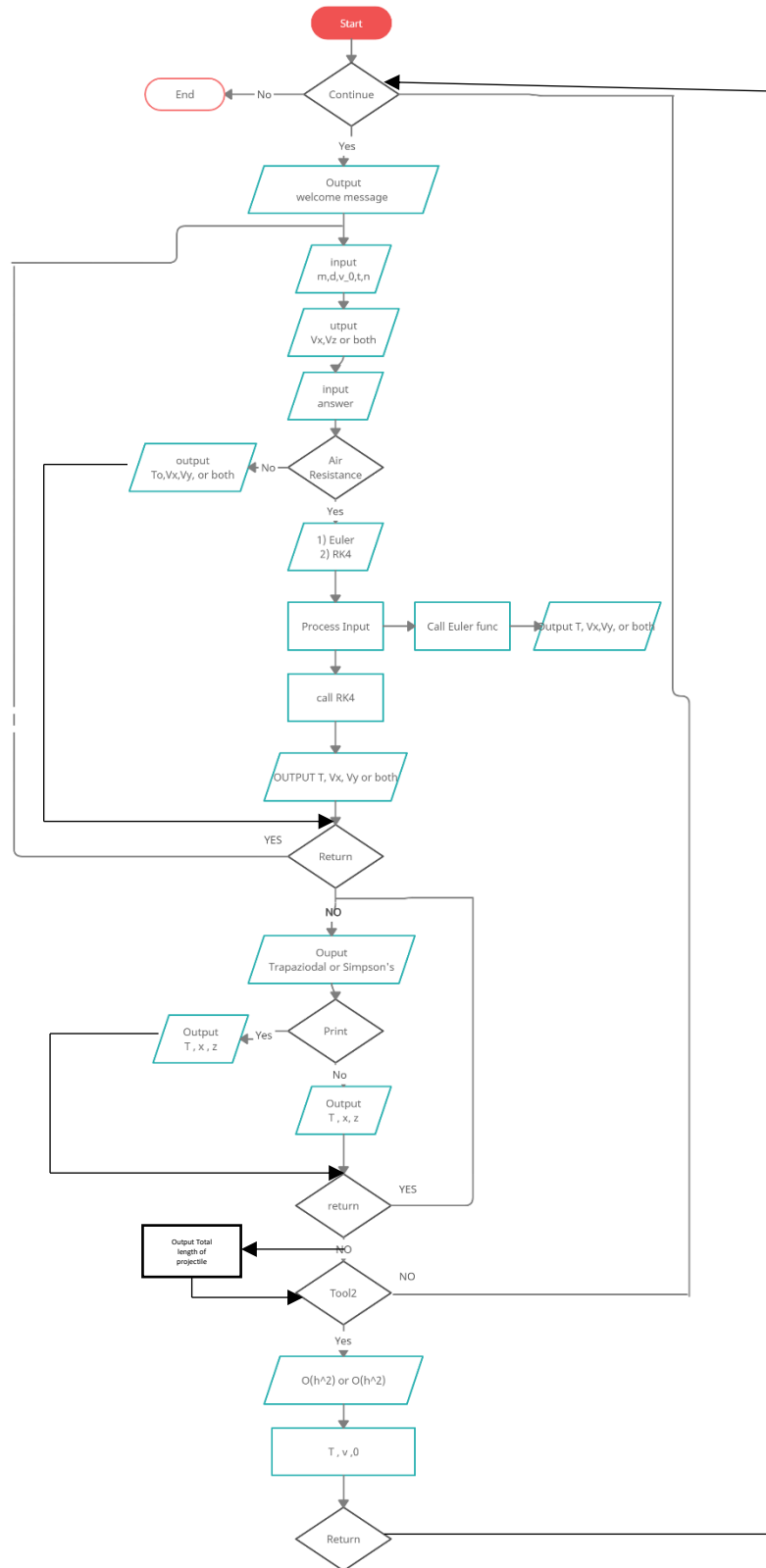
Resultant Velocity

| | | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 20.0000 | 19.2040 | 18.6141 | 18.2503 | 18.1262 | 18.2466 | 18.6070 | 19.1936 | 19.9867 |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|

Angle Of Projectile

| | | | | | | | | |
|---------|---------|---------|--------|--------|---------|----------|----------|----------|
| 25.0000 | 19.2874 | 13.1479 | 6.6862 | 0.0498 | -6.5879 | -13.0534 | -19.1986 | -24.9181 |
|---------|---------|---------|--------|--------|---------|----------|----------|----------|

Flow Chart



User Manual

- 1- After installing the zip file, decompress it and open main.m file using MATLAB software.
- 2- Run the program using run icon in editor toolbar or type main in command window.
- 3- A welcome message should appear showing program objective and tool 4 executes.
- 4- In tool 4 you are first required to enter some data:

- Initial velocity of projectile in m/s.
- Initial angle of projection in degrees.
- The time you target for simulation in seconds.
- Number of x-positions and z-positions you aim to get between the time 0sec and simulation time you choose.

```
>> main
Hello This is a Projectile Toolbox of 4 tools to help you:
  1- Identify the resultant velocity and direction of the projectile at each time step.
  2- Identify the position of the projectile at each time step.
  3- Identify the total length of the projectile's path.
*****
-----Tool4-----
*****
Hi, this is tool 4 in projectile toolbox
Tool 4 computes Resultant velocity and angle of the projectile versus time using ODEs
Enter Initial Velocity in m/s: 30
Enter Initial Angle in degrees: 45
Enter Travel Time for The Projectile in seconds: 5
How many Vx and Vz values you want to identify in this interval? 9
```

- 5- You are asked whether you want to print either velocities at x-axis or velocities at z-axis or both corresponding to the time intervals you choose and you are asked whether you want to print resultant velocity and angles through the time intervals.
- 6- Then you have a choice to account for air resistance in your calculations or neglect it.
- 7- If you choose to account for air resistance you have to give:

- Mass of the projectile in kg.
- Diameter of the projectile in meters.

and have the choice to use either Euler's method or RK4 method for calculation.

```
Do you want to print vx or vz?
Enter 1 for vx
Enter 2 for vz
Enter 3 for vx and vz
Enter any other number to neglect printing either
3
If you want to print resultant velocity and angle at each time stamp enter 11
Do you want to account for air resistance?
Enter 1 To ignore air resistance
Enter 2 To incorporate air resistance
2
Enter 1 for Euler's Method
Enter 2 for RK4 Method
2
Enter Mass of Projectile in kg: 2
Enter Diameter of Projectile in m: 0.5
```

- 8- You get output in this format below and you are asked if you would like to re-run tool 4 again or leave tool 4:

Projectile Toolbox

```

Time
    0    0.6250    1.2500    1.8750    2.5000    3.1250    3.7500    4.3750    5.0000

Velocity in x-direction in m/s
    21.2132    16.7033    14.1656    12.4634    11.0242    9.5948    8.1565    6.7815    5.5391

Velocity in z-direction in m/s
    21.2132    11.0355    3.5948    -2.6637    -8.1819    -12.8964    -16.6690    -19.4970    -21.5065

Resultant Velocity in m/s
    30.0000    20.0196    14.6146    12.7449    13.7287    16.0741    18.5575    20.6427    22.2084

Angle in degrees
    45.0000    33.4517    14.2393    -12.0638    -36.5822    -53.3511    -63.9265    -70.8212    -75.5571

x If you wish to rerun tool4 enter 1 --- If you wish to proceed enter any other number |

```

- 9- If you exit you are redirected to tool3 where you get position (x,z) versus time (t).
- 10- You are given the choice to use either Trapezoidal rule or Simpsons 1/3 rule.
- 11- Then you are asked if you want to plot x-position against y-position at different times.
- 12- If you choose to print the graph, a new window will open showing it in half of the window, to account for another graph with the other method if you rerun the tool again to facilitate comparing between them if you wish.
- 13- You are asked whether you want to rerun tool 3 again or exit and proceed the program.

```

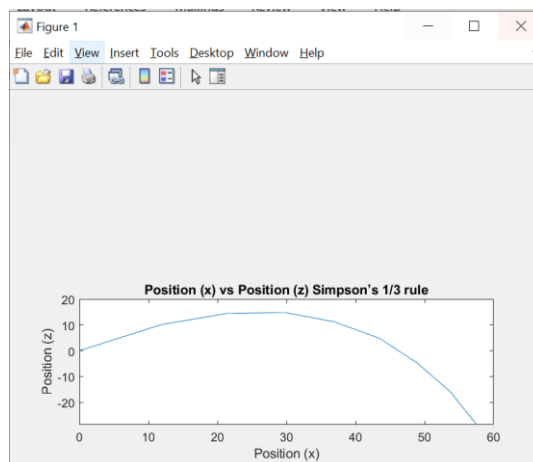
*****
-----Tool3-----
*****
Hi, this is tool 3 in projectile toolbox
Tool 3 computes Position (x,z) versus time (t)
Which Method do you want to use
Enter 1 to pick Trapezoidal rule.
Enter 2 to pick Simpson's 1/3 rule.
2
Time
    0    0.6250    1.2500    1.8750    2.5000    3.1250    3.7500    4.3750    5.0000

X-position
    0    11.8489    21.2900    29.7300    36.9241    43.5122    48.9157    53.7210    57.5712

Z-position
    0    10.0777    14.3646    14.8175    11.1892    4.7575    -4.7351    -15.8819    -28.6955

Do you which to plot x-position versus y-position?
Enter 1 to plot the graph.
Enter any other number to proceed.
1
If you wish to rerun tool3 enter 1 --- If you wish to proceed enter any other number |

```



Projectile Toolbox

14- Tool 1 is executed directly after Tool3 and it prints the total length of the projectile's path in meters.

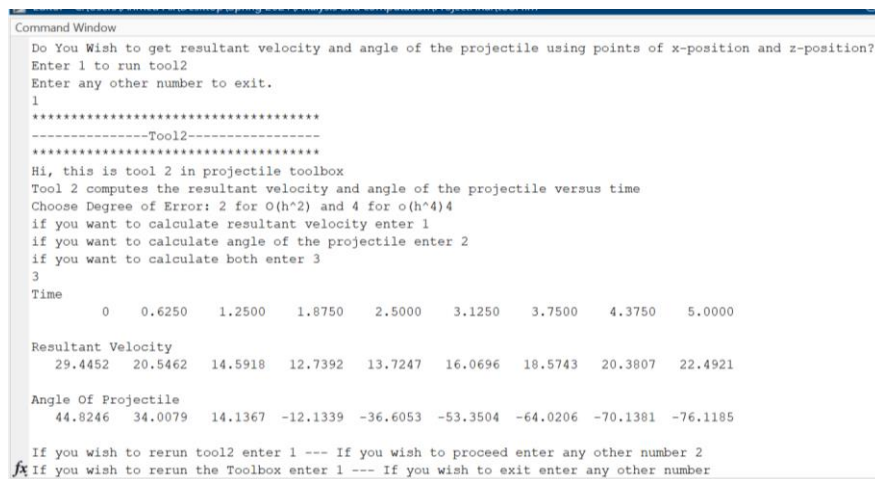
```
*****
-----Tool1-----
*****
Hi, this is tool 1 in projectile toolbox
Tool 1 computes the total length of the path of the projectile in meters
Total Length of the projectile's path in m is:
88.6518
```

15- Tool 2 is an optional tool, so you are asked whether you want to use it or not. You could use it to get resultant velocity and angle of the projectile using points of x-position and z-position you obtained in tool 3.

16- In tool 2 we are using numerical differentiation and you have the choice to choose either $O(h^2)$ or $O(h^4)$ degree of error for your calculations.

17- Then you are asked whether you want to calculate resultant velocity, angle of projectile or both.

18- You are asked whether you want to rerun tool 2 or not. You can use that to calculate the results for another degree of error.



```
Command Window
Do You Wish to get resultant velocity and angle of the projectile using points of x-position and z-position?
Enter 1 to run tool2
Enter any other number to exit.
1
*****
-----Tool2-----
*****
Hi, this is tool 2 in projectile toolbox
Tool 2 computes the resultant velocity and angle of the projectile versus time
Choose Degree of Error: 2 for O(h^2) and 4 for o(h^4)4
if you want to calculate resultant velocity enter 1
if you want to calculate angle of the projectile enter 2
if you want to calculate both enter 3
3
Time
0    0.6250    1.2500    1.8750    2.5000    3.1250    3.7500    4.3750    5.0000

Resultant Velocity
29.4452    20.5462    14.5918    12.7392    13.7247    16.0696    18.5743    20.3807    22.4921

Angle Of Projectile
44.8246    34.0079    14.1367    -12.1339    -36.6053    -53.3504    -64.0206    -70.1381    -76.1185

If you wish to rerun tool2 enter 1 --- If you wish to proceed enter any other number 2
fx If you wish to rerun the Toolbox enter 1 --- If you wish to exit enter any other number
```

19- Finally, you are asked whether you want to rerun the whole program again or exit.

Difficulties and Sources of error

Difficulties

The main difficulty that faced us during programming the code understood the concept behind the equations themselves, from understanding the equations and different numerical methods to applying them and putting them in action through programming. Algorithmic errors, some of the codes were programmed several times and went through more than 1 test to make sure it is working effectively. Also, the flow chart section was a bit complicated, as it was hard to make several charts for each tool separately, with each code containing more than 50 lines. the utilization of the code in different parts was quite tricky too. However, initially, it looked like breaking down the code to a few other tools would make it easier, but when it came to putting them all together in work took much extra effort.

Sources of error

- 1- MATLAB rounds values leading to some rounding errors in the results.
- 2- Drag coefficient calculation should be for Reynold's number of less than 2000, but sometimes Reynold's calculation yields more than 2000 missing up with velocities calculations and it sometimes the values seem unrealistic under certain inputs (diameter and initial velocity since they account for changes in Reynold's number).
- 3- Truncation and rounding errors in differential and integral numerical methods.
- 4- Very large values of Reynold's number will terminate the program.

Conclusion

With the five tools being finalized, the resulting computational analysis is expected to meet the given instructional and research-based objectives. Therefore, fundamental principles of kinematics and mechanics from introductory physics and engineering, along with different numerical analysis methods, ranging from simple metrics to advanced ODE Techniques, are simulated and utilized together to set up the Projectile Toolbox. Using the powerful and extensible MATLAB features, with its numerical and graphical tools included, many challenging aspects of this Projectile toolbox were huge simplified to serve the objectives of this paper. And with the help of this toolbox, it would be possible to calculate the total length of the path of the projectile along with the resultant velocity and angel along with the flight; moreover, the position of the fired object could be determined at any point in the 2D plane, with all that combined along with some variables related to the air resistance, this toolbox could be utilized to hit a specific target. In light of the positive findings of this work, future work on this area is expected to expand these simulations to cover other theoretical and experimental topics.

References

- 1- Chapra, Steven C, and Raymond P. Canale. Numerical Methods for Engineers. Boston: McGraw-Hill Higher Education, 2006.
- 2- Adam, B., & Hashim, M. (204AD, October). SHOOTING METHOD IN SOLVING BOUNDARY VALUE PROBLEM.
- 3- Differential Equations - Boundary Value Problems. (2020, May).
<https://tutorial.math.lamar.edu/classes/de/BoundaryValueProblem.aspx>.
- 4- Gladwell, I. (2008, July). Boundary value problem. Scholarpedia.
http://www.scholarpedia.org/article/Boundary_value_problem#:~:text=A%20Boundary%20value%20problem%20is,two%2Dpoint%20boundary%20value%20problem.
- 5- Yetilmezsoy, K., & Mungan, C. E. (2018). MATLAB time-based simulations of projectile motion, pendulum oscillation, and water discharge. European Journal of Physics, 39(6), 065803.
<https://doi.org/10.1088/1361-6404/aadaee>
- 6- Solutions and Explanations to Projectile Problems
http://www.problemsphysics.com/mechanics/projectile/projectile_solution.html#Solution_to_Problem_1
- 7- EM375 MECHANICAL ENGINEERING EXPERIMENTATION PROJECTILE MOTION WITH AIR RESISTANCE