<u>Lab-III</u>

Devansh Shukla
I18PH021

1 Aim

Write and execute an octave program to simulate/solve projectile motion.

2 Theory

3 Program

3.1 Simple pendulum

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Projectile
%
% Program to solve/simulate the motion of a particle in projectile motion
% Author: Devansh Shukla I18PH021
% 2nd Feb, 2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

graphics_toolkit gnuplot
pkg load symbolic

% set the symbolic vars
syms x0 y0 t g v0 th theta;

% Analytical Solution for projectile motion
% Time of flight
T = 2 * v0 * sin(theta) / g;

% Max height reached
H = v0**2 * sin(theta)**2 / (2 * g);

% Range
R = v0**2 * sin(2*theta) / (2 * g);


% Initial parameters
g = 9.81; % m/s/s
v0 = 20;
th = pi/4;
x0 = 0;
y0 = 0;

% Positions
x = v0 * cos(th) * t
y = v0 * sin(th) * t - 0.5*g*t*t

% velocities
vx = v0 * cos(th)
vy = v0 * sin(th) - g * t


t0 = input("Enter t0 ")
tf = input("Enter tf ")

x = y = [];
m = 1;
% loop for numerically computing the motion
% NOTE: the loop will break when y < 0
for t=t0:0.1:tf
    x(m) = v0 * cos(th) * t;
    y(m) = v0 * sin(th) * t - 0.5*g*t*t;
    if (y(m) < 0)
        display("BREAK")
        break
    endif
    m = m + 1;
endfor

% plotting the trajectory
figure()
hold on
grid on
set(gca,'XMinorTick','on','YMinorTick','on')
plot(x, y, "marker", "+", "linewidth", 2)
xlabel("x[m]")
ylabel("y[m]")
title("Trajectory")
ylim([0, 12])
print("-dpng", "proj_trajectory.png")
hold off

% computing positions for multiple thetas to contrast their ranges
```

```
th = [pi/16, pi/8, pi/4]

% plotting multiple trajectories
figure()
hold on;
grid on;
set(gcf, 'PaperSize', [6, 3]);
set(gca,'XMinorTick','on','YMinorTick','on')

% loop for computing multiple trajectories
for i=1:3
    x = y = [];
    m = 1;
    for t=0:0.1:3
        x(m) = v0 * cos(th(i)) * t;
        y(m) = v0 * sin(th(i)) * t - 0.5*g*t*t;
        if (y(m) < 0)
            display("BREAK")
            break
        endif
        m = m + 1;
    endfor
    plot(x, y, "marker", "+", "linewidth", 2);
    drawnow
endfor

xlabel("x[m]");
ylabel("y[m]");
title("Trajectory");
legend("theta=pi/16", "theta=pi/8", "theta=pi/4", "location", "northeastoutside")
legend boxoff
ylim([0, 12])
set(gcf, 'renderer', 'painters');
print("-dpng", "proj_th_vs_traj.png");
hold off;
```

4 Results

4.1 Terminal output

```
(escape) devansh@ds:~/GitHub/Vault/OctaveLab/Programs/outputs$ octave ../Projectile.m
Symbolic pkg v2.9.0: Python communication link active, SymPy v1.5.1.
warning: passing floating-point values to sym is dangerous, see "help sym"
warning: called from
    double_to_sym_heuristic at line 50 column 7
    sym at line 379 column 13
    mtimes at line 63 column 5
    ../Projectile.m at line 34 column 3
x = (sym) 10·√2·t
warning: passing floating-point values to sym is dangerous, see "help sym"
warning: called from
    double_to_sym_heuristic at line 50 column 7
    sym at line 379 column 13
    mtimes at line 63 column 5
    ../Projectile.m at line 35 column 3
warning: passing floating-point values to sym is dangerous, see "help sym"
warning: called from
    double_to_sym_heuristic at line 50 column 7
    sym at line 379 column 13
    mtimes at line 63 column 5
    ../Projectile.m at line 35 column 3
y = (sym)

         2
    981·t
  - ────── + 10·√2·t
     200

vx =  14.142
warning: passing floating-point values to sym is dangerous, see "help sym"
warning: called from
    double_to_sym_heuristic at line 50 column 7
    sym at line 379 column 13
    mtimes at line 63 column 5
    ../Projectile.m at line 39 column 4
warning: passing floating-point values to sym is dangerous, see "help sym"
warning: called from
    double_to_sym_heuristic at line 50 column 7
    sym at line 379 column 13
    minus at line 55 column 5
    ../Projectile.m at line 39 column 4
vy = (sym)

    981·t
  - ───── + 10·√2
     100

Enter t0 0
t0 = 0
Enter tf 10
tf =  10
BREAK
th =

   0.19635   0.39270   0.78540

BREAK
BREAK
BREAK
```

4.2 Plots

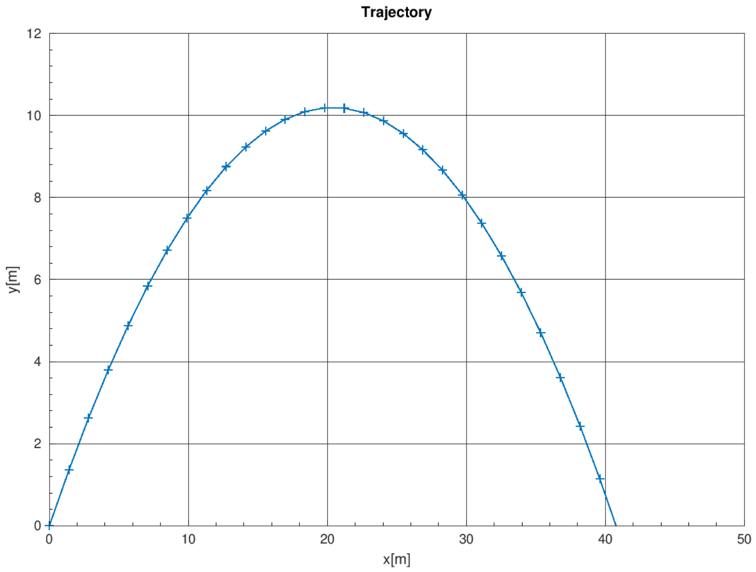

Figure 1: Inital parameters: $v_0 = 20\ m/s$; $\theta = \pi/4$

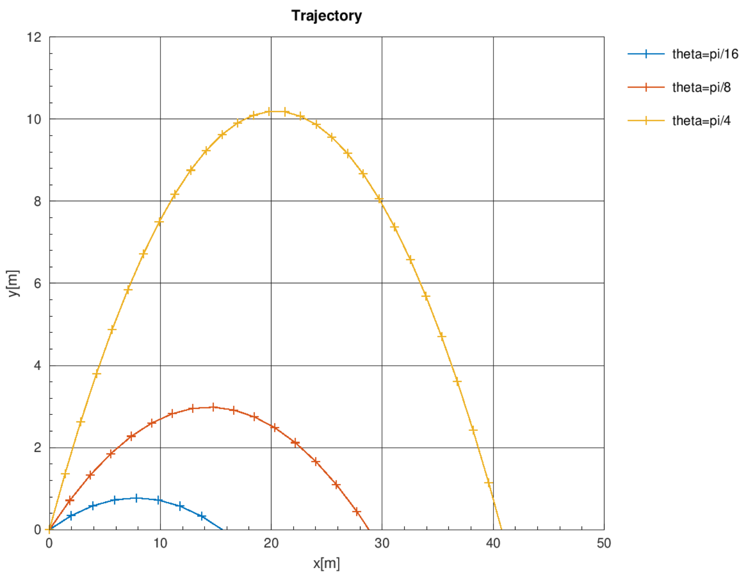

Figure 2: Trajectories at $\theta = \pi/16,\ \pi/8,\ \pi/4$

5 Remarks

The programs can be used to trace and simulate the motion of any particle in uniform circular motion by defining the required parameters.

The parameters computed numerically and via the programs are in agreement.