Lab-V

Devansh Shukla
I18PH021

1 Aim

Write and execute an octave program to simulate/solve motion of a particle in a plane in polar coordinates.

2 Theory

3 Program

3.1 Simple pendulum

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DampedSimpleHarmonicMotion
%
% Program to solve/simulate motion of a particle in uniform circular motion
% in polar coordinates.
% Author: Devansh Shukla I18PH021
% 9th March, 2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

graphics_toolkit gnuplot
pkg load symbolic

% set the symbolic variables
syms theta0 t t0 R omega

% equations for position theta and x,y for plotting
th = theta0 + omega * (t-t0)
x = R * cos(theta0 + omega * (t-t0));
y = R * sin(theta0 + omega * (t-t0));

% pretty output
fprintf("Position\n");
fprintf("x = "); pretty(x)
fprintf("y = "); pretty(y)
fprintf("theta = "); pretty(th)
fprintf("\n")

% Initial conditions
omega = input("Enter omega ");
R = input("Enter radius ");
t0 = input("Enter t0 ");
tf = input("Enter tf ");
theta0 = pi/4

theta = pos_x = pos_y = [];

% Loop for numerically calculating positions
idx = 1;
for t=t0:0.05:tf
    pos_x(idx) = eval(x);
    pos_y(idx) = eval(y);
    theta(idx) = eval(th);
    idx = idx + 1;
endfor

% Plot the trajectory and positions
figure();
set(gcf, 'PaperSize', [6, 3]);
hold on;
grid on;
set(gca,'XMinorTick','on','YMinorTick','on')
plot(pos_x, pos_y, "linewidth", 2);
xlabel("x[m]");
ylabel("y[m]");
title("Trajectory");
xlim([-3, 3])
ylim([-3, 3])
set(gcf, 'renderer', 'painters');
print("-dpng", "polar_cart_traj.png");
legend boxoff
hold off;

radius = zeros(length(theta), 1) + R;
figure();
polar(theta, radius, "x");
title("Trajectory");
set(gca, "rtick", 0.5:R+1, "ttick", 0:20:340);
set(gcf, 'renderer', 'painters');
print("-dpng", "polar_traj.png");
legend boxoff
hold off;
```

```
theta_plot = atan(pos_y ./ pos_x);
figure();
set(gcf, 'PaperSize', [6, 3]);
hold on;
grid on;
set(gca,'XMinorTick','on','YMinorTick','on')
plot(theta_plot, "linewidth", 2);
plot([0 150], [pi/2 pi/2], "color", "red", "linewidth", 2);
plot([0 150], [-pi/2 -pi/2], "color", "red", "linewidth", 2);
xlabel("Time(s) [1=0.05s]");
ylabel("theta");
title("theta vs time");
xlim([0 150])
legend("theta", "pi/2", "-pi/2")
legend boxoff
set(gcf, 'renderer', 'painters');
print("-dpng", "polar_theta.png");
hold off;
```

4 Results

4.1 Terminal output

```
(escape) devansh@ds:~/GitHub/Vault/OctaveLab/Programs/outputs$ octave ../MotionOnAPlanePolar.m
Symbolic pkg v2.9.0: Python communication link active, SymPy v1.5.1.
th = (sym) ω·(t - t₀) + θ₀
Position
x =   R·cos(ω·(t - t₀) + θ₀)
y =   R·sin(ω·(t - t₀) + θ₀)
theta =   ω·(t - t₀) + θ₀

Enter omega 2
Enter radius 2
Enter t0 0
Enter tf 6
theta0 =  0.78540
```

4.2 Plots

Inital paramters:

- $R = 2.0\ m$
- $\omega = 2.0\ rad/s$
- $\theta_0 = \pi/4$
- $t_0 = 0.0s$
- $t_f = 6.0s$

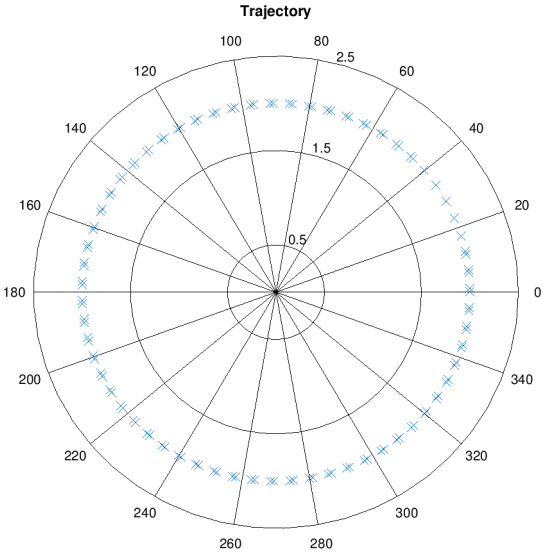

Figure 1: Polar plot

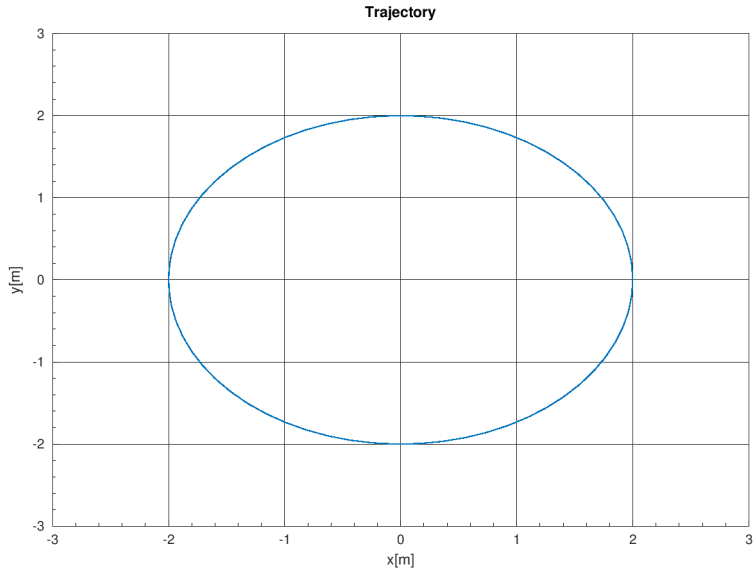

Figure 2: Polar transformed to cartesian coordinates

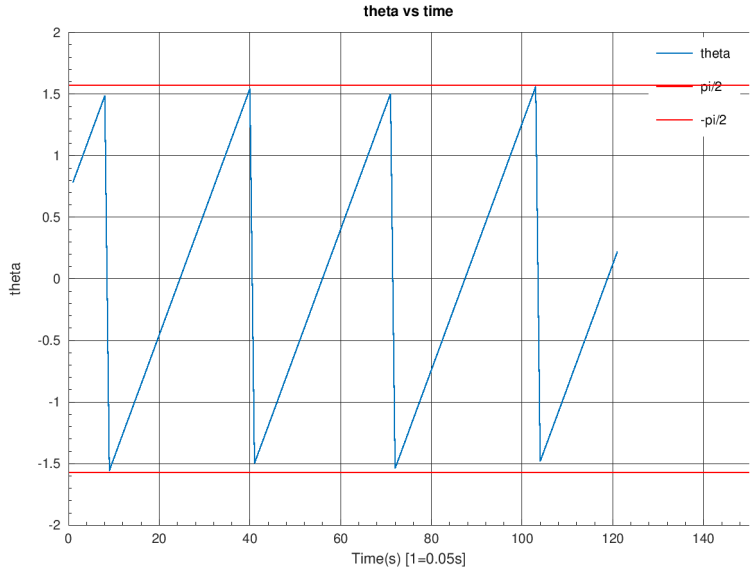

Figure 3: θ vs time

5 Remarks

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