Animations using Position Vectors in MATLAB

Cluster Innovation Centre
University of Delhi

Kritika Verma
Siddhartha Mahajan

July 2022

Month Long Project submitted for the paper
Linearity in Nature: Engineering through Linear Algebra

Certificate of Originality

The work embodied in this report entitled **"Animations using Position Vectors in MATLAB"** has been carried out by **Kritika Verma** and **Siddhartha Mahajan** for the paper **"Linearity in Nature: Engineering through Linear Algebra"**. I declare that the work and language included in this project report is free from any kind of plagiarism.

(Name and Signature)

Acknowledgement

With a deep sense of gratitude, we express our dearest indebtedness of Dr.Sonam Tanwar and Dr.Harendra Pal Singh for their support throughout the duration of our project. We would like to thank them for giving us the opportunity to do this wonderful project. Their learned advice and constant encouragement has helped us to complete this project. It is a privilege for us to be their students.

Animations using Position Vectors in MATLAB

by

Kritika Verma
BTech ITMI 2021-25, Cluster Innovation Center,
University of Delhi

Siddhartha Mahajan
BTech ITMI 2021-25, Cluster Innovation Center,
University of Delhi

This report focuses on plotting differential equations and discrete dynamical systems using MATLAB. Along with animation of these systems which depicts their growth with respect to time.

Index

1 Introduction

1.1 Background and Context

MATLAB is a programming and numeric computing platform used by students and professionals alike. It has various tools and functions for plotting as well as calculating. It is also one of the most efficient software to work with **Linear Algebra and its Applications** . And one its key features which this report focuses on is matrices, dynamic system, their plotting and animation of those systems changing with time. We've taken a few of linear equations as well as matrices and tried to animate them using the different tools in MATLAB.

1.2 Scope and Objectives

- Making animations using MATLAB, which are extractable in any video format.
- Making a GUI from scratch in MATLAB
- Use different scripts as functions in the main script.
- Plot and animate a solved differential equation.
- Plot and animate a saddle point of an equation.
- Plot and animate some aesthetically pleasing figures.
- To enhance our skills in LaTeX.
- Plot discrete dynamical systems in MATLAB.
- Learn how to collaborate better on group projects.

1.3 Achievements

- We made animations using MATLAB, which were extract-able in any video format.
- We made a GUI from scratch using MATLAB and used different scripts as functions in the main script.
- We were able to show a plot of a solved differential equation, a saddle point of an equation and some aesthetically pleasing figures.
- We enhanced our skills in LaTeX.
- We were able to plot discrete dynamical systems in MATLAB.
- We have learnt to collaborate better on group projects.

2 Formulation of the Problem

2.1 Problem Statement

Though, we know how plotting works in MATLAB. It's hard to see the trajectory of a particle by looking at the already formed graph. The same can be said about discrete dynamical systems that change with time. A better method of plotting these can be with the help of animation which will give a more vivid idea of how the system of equation works.

2.2 Methodology

All animations and videos are just a lot of images stitched together. Just like in a flip-book, if changes are made little by little, and then the pictures are flipped by really fast, it looks like an animation. We tried to do the same thing in MATLAB.

To make an animation in MATLAB, we followed the following steps:

1. Run a simulation and generate all the data points.
2. Draw/render/plot the scenario at time
$$t_k$$
.
3. Take a snapshot of the scenario, store it.
4. Advance time to
$$t_k + 1$$
.

5. Repeat the process from Step 2.
6. Make a movie from the acquired snapshots.

We constructed a GUI in MATLAB to show animations/plots of the following:

1. **Flower1**
   We found a parameterized equation for a flower which is:
   $$x = 3cos(cos(7.94round(t))(1.2)(t * cos(14.34t)$$
   $$y = 3sin(14.34t)sin(14.34t)sin(7.94t)$$
   We plotted these equations with t varying from -2 to 2 in 100 steps and took a snapshot at each step. Saving these snapshots and running them with a frame rate of 20, gave us an extract-able video.
2. **Flower2**
   We found another parameterized equation for a flower which is:
   $$x = 2cos(t) + cos(5t)$$
   $$y = 2sin(t) + sin(5t)$$
   We plotted these equations with t varying from to 2pi in 100 steps and took a snapshot at each step. Saving these snapshots and running them with a frame rate of 20, gave us an extract-able video.
3. **Spheres**
   We found a parameterized equation for a spiral single parameter sphere which is:
   $$x = sin(t)sin(ct)$$
   $$y = sin(t)cos(ct)$$
   $$z = cos(t)$$

We ran a loop for c=1 to 100 and varied t from 0 to 2pi in 100 steps and took a snapshot at each step. Saving these snapshots and running them with a frame rate of 20, gave us an extract-able video. Analyzing the data we got, we found that c=85 and c=49 gave really interesting results.

4. **Discrete Dynamical System**
   The set of equations we chose was from a dynamic system that consists of complex Eigenvalues. So, theoretically, we should get a spiral. And since we took a negative exponential our system should tend towards 0. The equations we used were:

$$x = [-1.5sin(5t) + 3cos(5t)]e^{-2t}$$
$$y = [3cos(5t) + 6sin(5t)]e^{-2t}$$

   We plotted these equations with t varying from 0 to 2pi in 100 steps and took a snapshot at each step. Saving these snapshots and running them with a frame rate of 20, gave us an extract-able video.

5. **Differential Equations**
   We solved a differential equation using Eigen Vector and got these following parameterized equations:

$$x = 3e^{-5t} + 2e^{-2t}$$
$$y = 6e^{-5t} - 2e^{-2t}$$

   We plotted these equations with t varying from 0 to 2pi in 100 steps and took a snapshot at each step. Saving these snapshots and running them with a frame rate of 20, gave us an extract-able video.

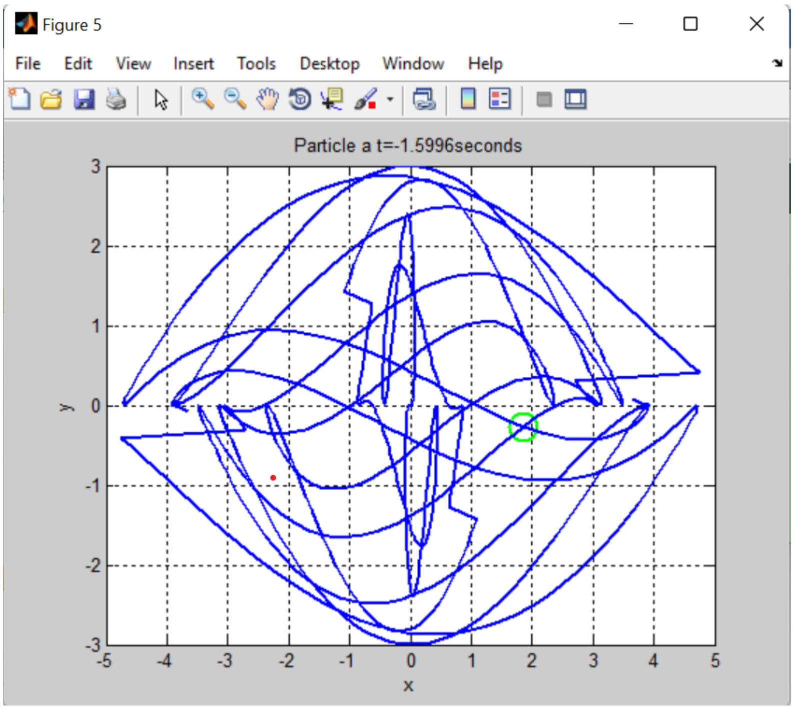
2.3 Results

We got the following results:

1. **Flower1**

$$x = 3cos(cos(7.94round(t))(1.2)(t * cos(14.34t)$$

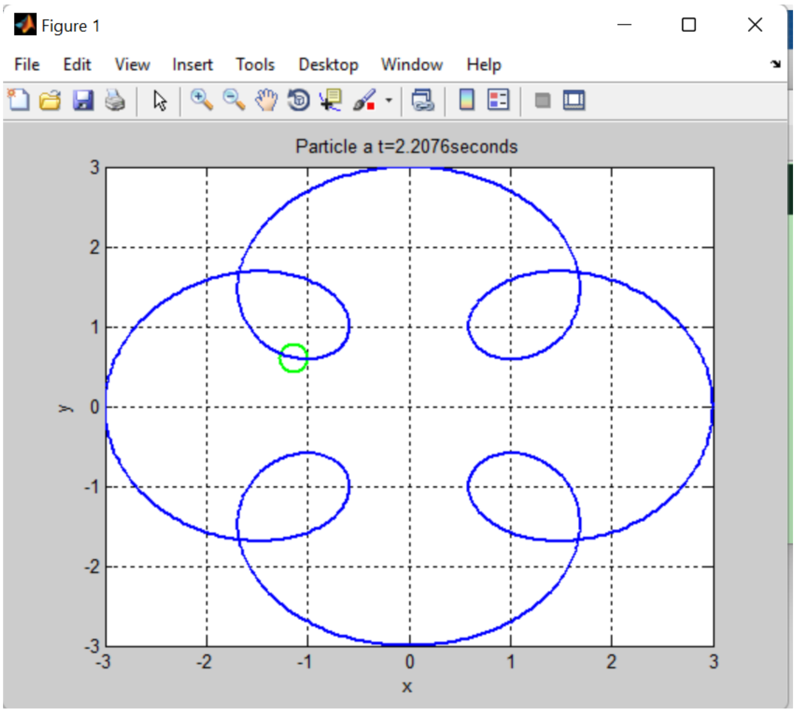
$$y = 3sin(14.34t)sin(14.34t)sin(7.94t)$$



2. **Flower2**

$$x = 2cos(t) + cos(5t)$$

$$y = 2sin(t) + sin(5t)$$



3. Spheres

$$x = sin(t)sin(ct)$$

$$y = sin(t)cos(ct)$$

$$z = cos(t)$$

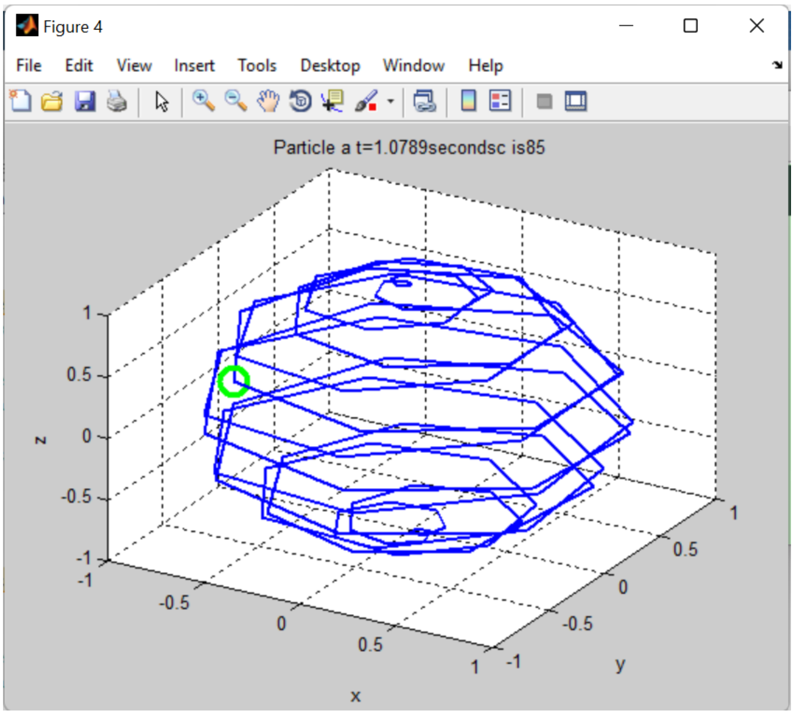


Figure 4
File Edit View Insert Tools Desktop Window Help
Particle a t=1.0789secondsc is85
1
0.5
0
-0.5
-1
z
-1
-0.5
0
0.5
1
x
-1
-0.5
0
0.5
1
y

c=85

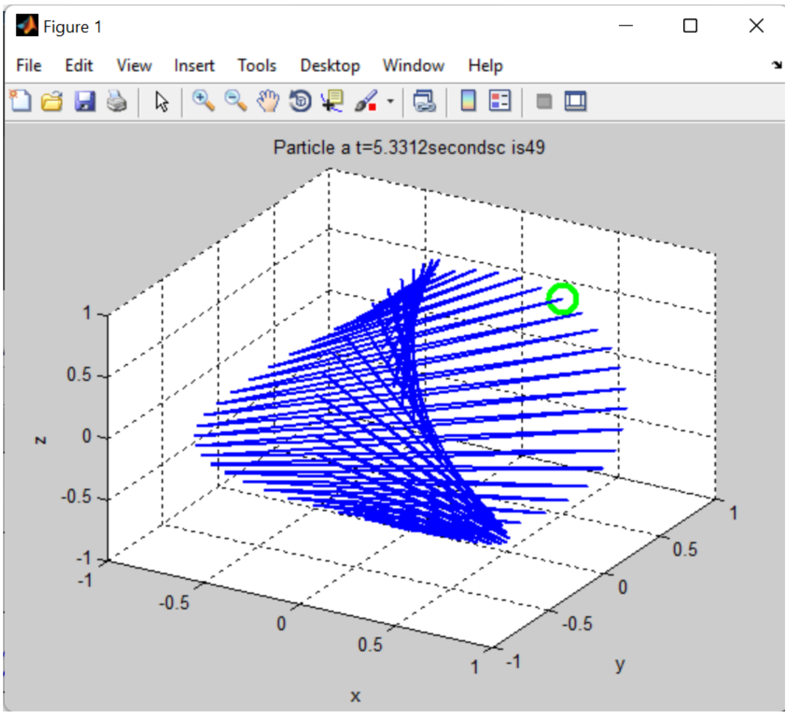
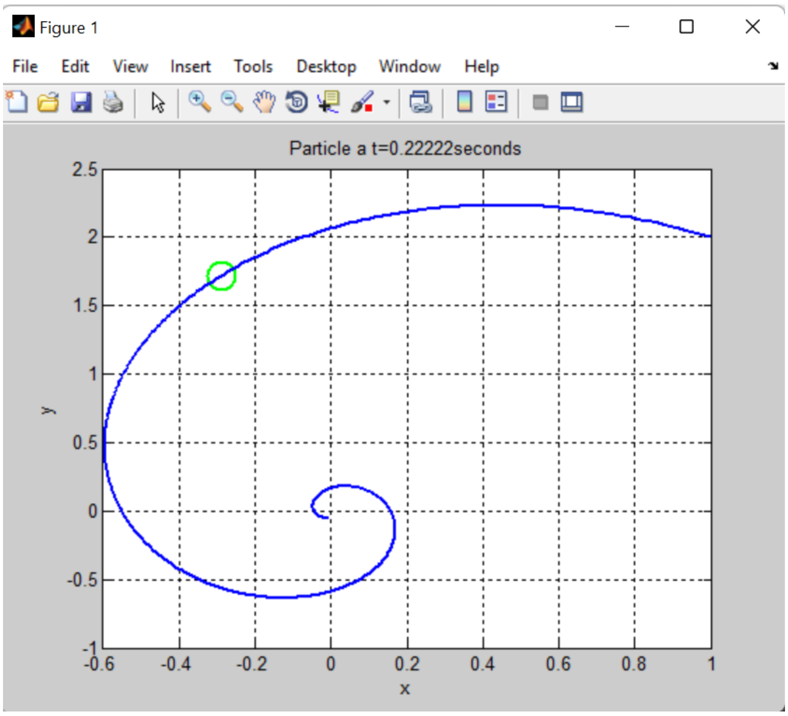


Figure 1
File Edit View Insert Tools Desktop Window Help
Particle a t=5.3312secondsc is49
1
0.5
0
-0.5
-1
z
-1
-0.5
0
0.5
1
x
-1
-0.5
0
0.5
1
y

c=49

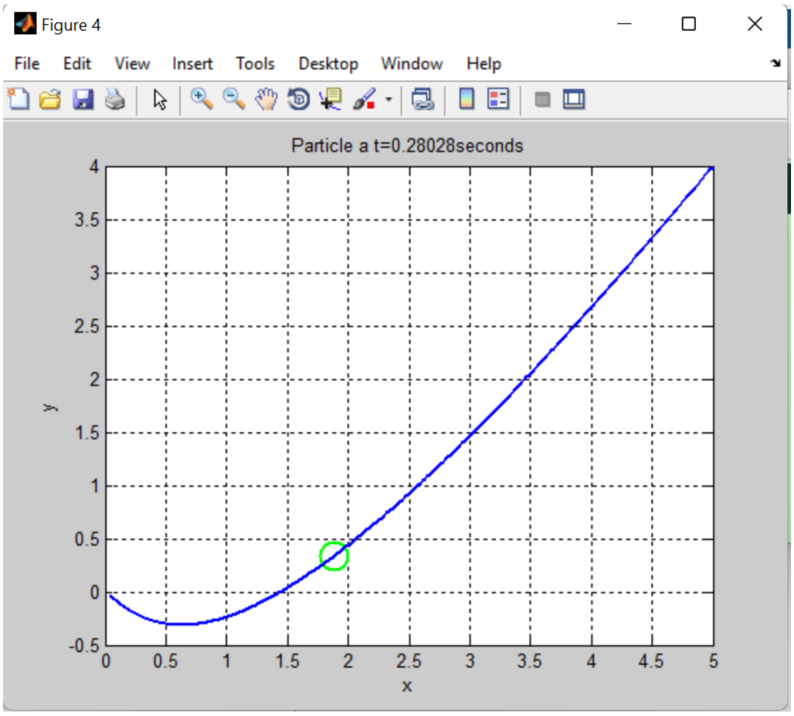
4. Discrete Dynamical System

$$x = [-1.5sin(5t) + 3cos(5t)]e^{-2t}$$
$$y = [3cos(5t) + 6sin(5t)]e^{-2t}$$

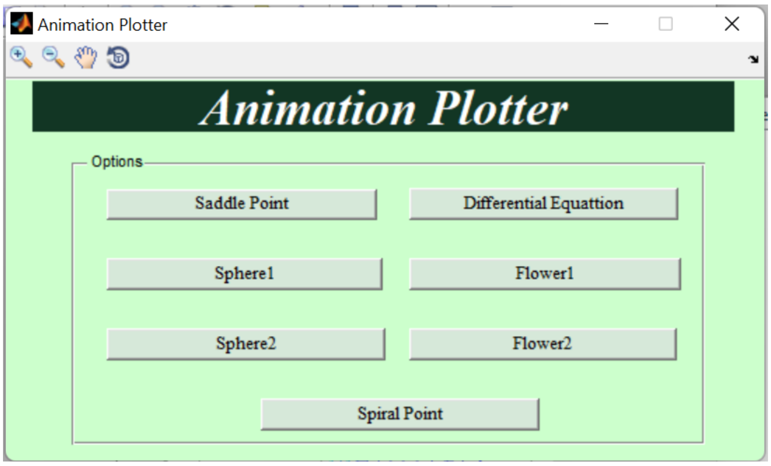


5. Differential Equations

$$x = 3e^{-5t} + 2e^{-2t}$$
$$y = 6e^{-5t} - 2e^{-2t}$$



6. Final GUI



Our final GUI has 7 push buttons for different options

3 Conclusion

We arrived at the following conclusions:

- We can animate and plot equations and system of equations in MATLAB.
- We can make downloadable videos from the animations.
- We can easily make a basic GUI in MATLAB.
- We can follow the trajectory of a point with respect to time.
- We can plot the solutions of Discrete Dynamical Systems and Differential Equations and animate them using MATLAB.

4 References

- https://elepa.files.wordpress.com/2013/11/fifty-famous-curves.pdf
- https://in.mathworks.com/help/matlab/ref/plot.html
  https://www.youtube.com/ChristopherLum
- https://sgeos.github.io/3d/parametric/math/
- https://lifethroughamathematicianseyes.wordpress.com /2014/11/13/parametric-equations/
- https://matlab.mathworks.com/
- https://www.youtube.com/watch?v=Ta1uhGEJFBE
- Linear Algebra and Its Applications by David C. Lay, Steven R. Lay and Judi J. Mcdonald

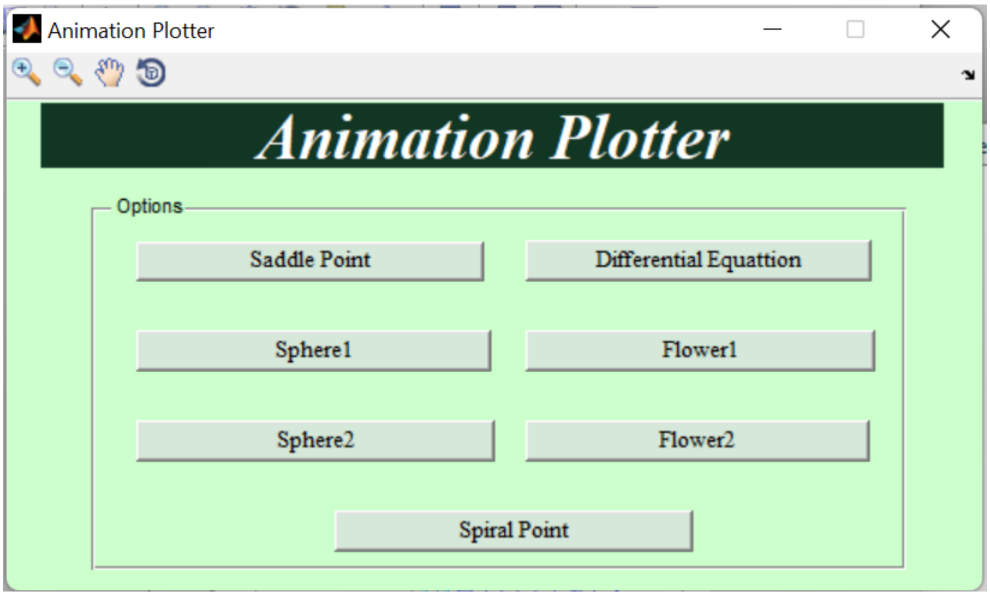
5 Appendix

Matlab-GUI

```
function varargout = Kritika(varargin)
%KRITIKA M-file for Kritika.fig
%      KRITIKA, by itself, creates a new KRITIKA or raises the ...
    existing
%      singleton*.
%
%      H = KRITIKA returns the handle to a new KRITIKA or the ...
    handle to
%      the existing singleton*.
%
%      KRITIKA('Property','Value',...) creates a new KRITIKA ...
    using the
%      given property value pairs. Unrecognized properties are ...
    passed via
%      varargin to Kritika_OpeningFcn.  This calling syntax ...
    produces a
%      warning when there is an existing singleton*.
%
%      KRITIKA('CALLBACK') and KRITIKA('CALLBACK',hObject,...) ...
    call the
%      local function named CALLBACK in KRITIKA.M with the given ...
    input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI ...
    allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Kritika

% Last Modified by GUIDE v2.5 20-Jul-2022 01:38:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Kritika_OpeningFcn, ...
                   'gui_OutputFcn',  @Kritika_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Kritika is made visible.
```

```
function Kritika_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for Kritika
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Kritika wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Kritika_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in Sphere1button.
function Sphere1button_Callback(hObject, eventdata, handles)
Sphere()
% hObject    handle to Sphere1button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
Differential()
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in Flower1button.
function Flower1button_Callback(hObject, eventdata, handles)
Flower()
% hObject    handle to Flower1button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
Sphere2()
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton6.
```

```
function pushbutton6_Callback(hObject, eventdata, handles)
ode1()
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in Flower2button.
function Flower2button_Callback(hObject, eventdata, handles)
Flower2()
% hObject    handle to Flower2button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


function pushbutton8_Callback(hObject, eventdata, handles)
Spiral()
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```



Sphere1

```
function[y]=Sphere()

%% Step 1: Generate Data
    c=85;
    t=linspace(0, 2*pi, 100);
    x=sin(t).*sin(c.*t);
    y=sin(t).*cos(c.*t);
    z=cos(t);

    %%Step 2:
    figure

        for k=1:length(t)
            clf

            t_k=t(k);
            x_k=x(k);
            y_k=y(k);
            z_k=z(k);

            plot3(x_k,y_k,z_k, 'go', 'LineWidth', 3, ...
                'MarkerSize', 15)

            hold on
            plot3(x,y,z,'b-', 'LineWidth',2)

            grid on
            xlabel('x')
            ylabel('y')
            zlabel('z')
            title(['Particle a t=', num2str(t_k),'seconds','c ...
                is',num2str(c)])
            view([30 35])



        %drawnow

        %pause(0.2)

            movieVector(k)=getframe;




    end
end
```

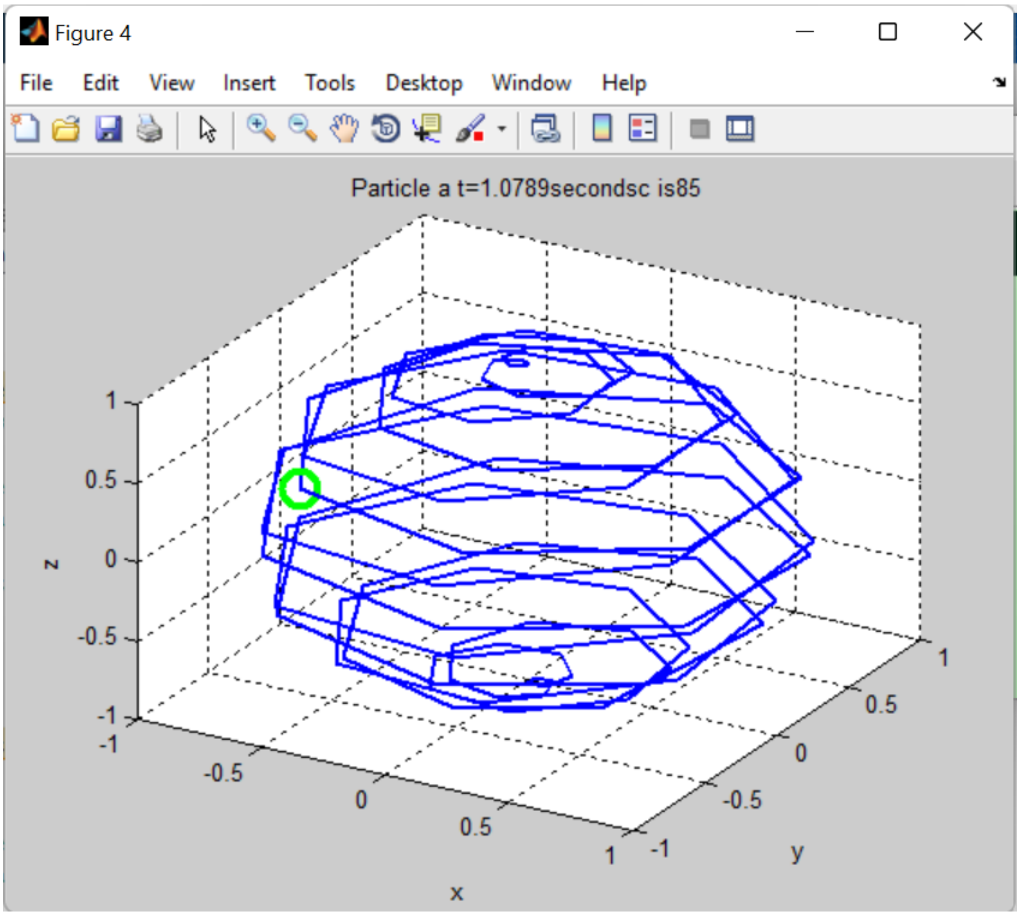


Figure 4
File Edit View Insert Tools Desktop Window Help
Particle a t=1.0789secondsc is85
z
x
y
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
-1
-0.5
0
0.5
1

Sphere2

```
function[y]=Sphere2()

%% Step 1: Generate Data
    c=49;
    t=linspace(0, 2*pi, 100);
    x=sin(t).*sin(c.*t);
    y=sin(t).*cos(c.*t);
    z=cos(t);

    %%Step 2:
    figure

        for k=1:length(t)
            clf

            t_k=t(k);
            x_k=x(k);
            y_k=y(k);
            z_k=z(k);

            plot3(x_k,y_k,z_k, 'go', 'LineWidth', 3, ...
                'MarkerSize', 15)

            hold on
            plot3(x,y,z,'b-', 'LineWidth',2)

            grid on
            xlabel('x')
            ylabel('y')
            zlabel('z')
            title(['Particle a t=', num2str(t_k),'seconds','c ...
                is',num2str(c)])
            view([30 35])



        %drawnow

        %pause(0.2)

            movieVector(k)=getframe;




    end
end
```

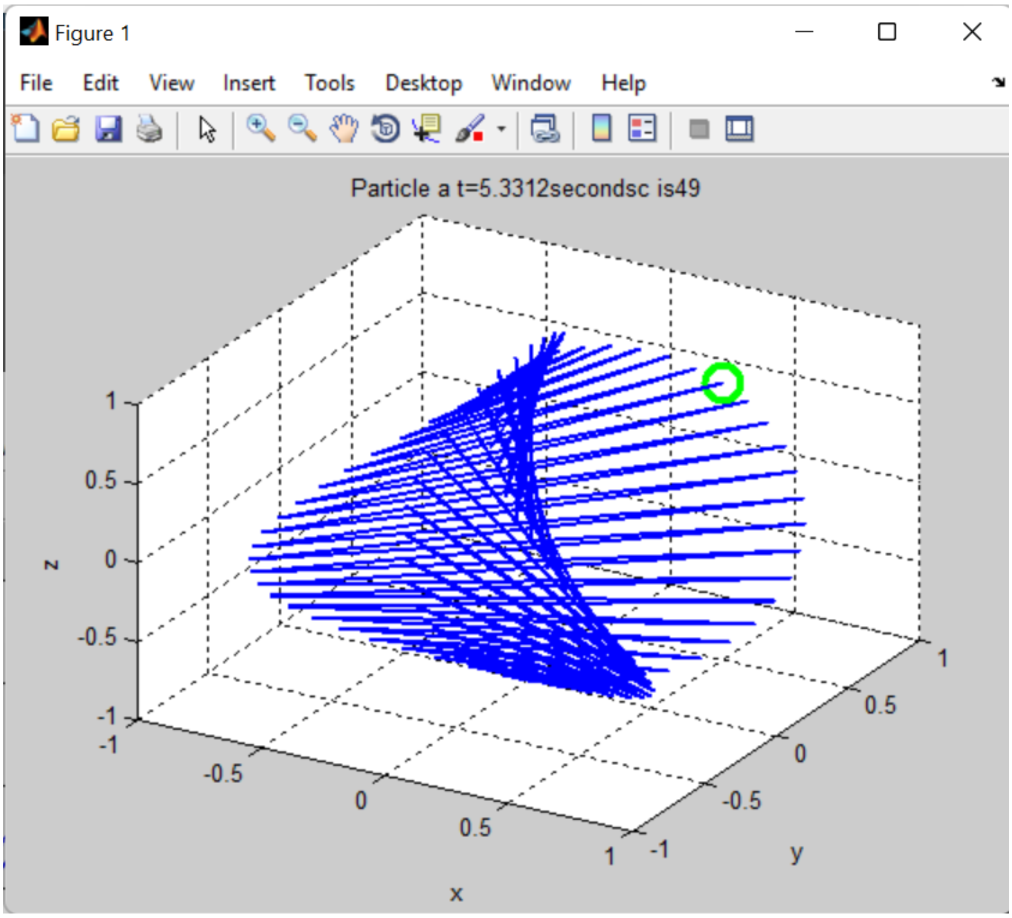


Figure 1
File Edit View Insert Tools Desktop Window Help
Particle a t=5.3312secondsc is49
1
0.5
0
-0.5
-1
z
-1
-0.5
0
0.5
1
x
-1
-0.5
0
0.5
1
y

Flower1

```
function[y]=Flower()

    %% Step 1: Generate Data
    t=linspace(-2, 2, 1000);
    x=3.*cos(cos(7.94.*round(t))).*(1.2).*(t.*cos(14.34.*t));
    y=3.*sin(14.34.*t).*sin(14.34.*t).*sin(7.94.*t);


    %%Step 2:
    figure

    for k=1:length(t)
        clf

        t_k=t(k);
        x_k=x(k);
        y_k=y(k);


        plot(x_k,y_k, 'go', 'LineWidth', 2, 'MarkerSize', 15)

        hold on
        plot(x,y,'b-', 'LineWidth',2)

        grid on
        xlabel('x')
        ylabel('y')
        zlabel('z')
        title(['Particle a t=', num2str(t_k),'seconds'])
        view([0 0 5])



        %drawnow

        %pause(0.2)

        movieVector(k)=getframe;




    end
end
```

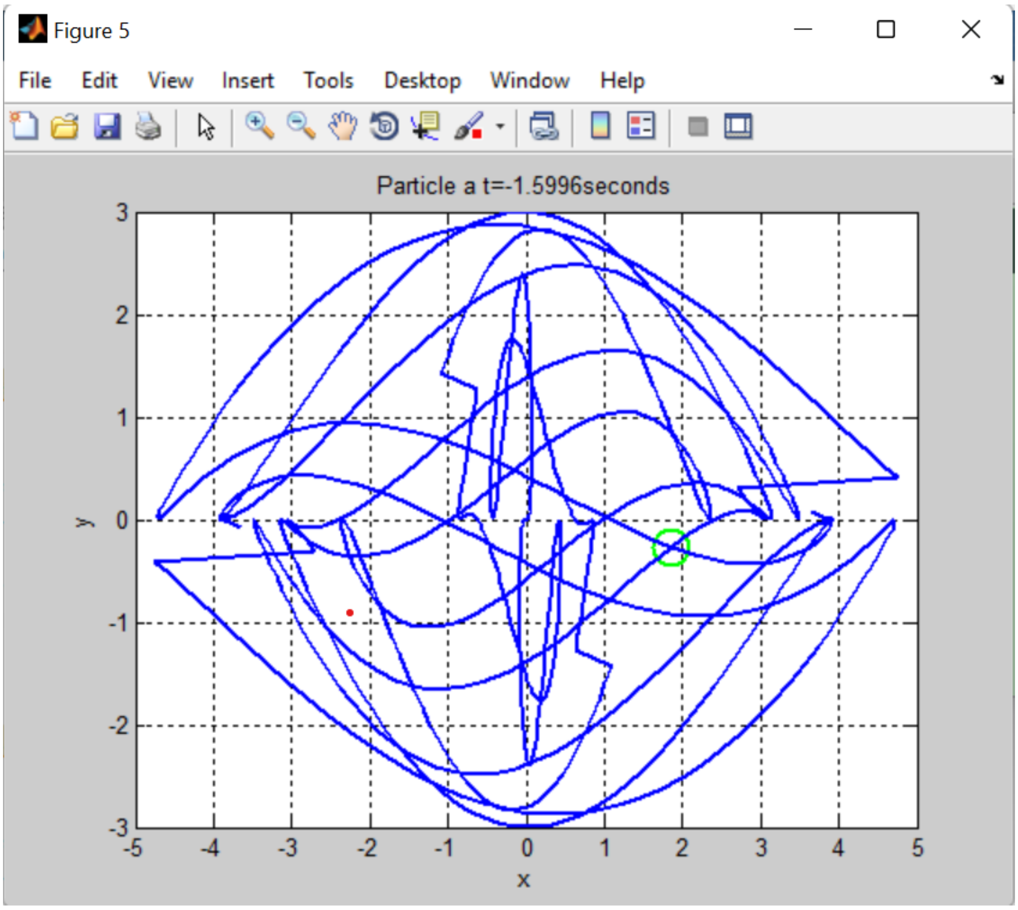


Figure 5
File Edit View Insert Tools Desktop Window Help
Particle a t=-1.5996seconds
y
x
3
2
1
0
-1
-2
-3
-5
-4
-3
-2
-1
0
1
2
3
4
5

Flower2

```

function[y]=Flower2()

    %% Step 1: Generate Data
    t=linspace(0, 2*pi, 1000);
    x=2.*cos(t)+cos(5.*t)
    y=2.*sin(t)+sin(5.*t)


    %%Step 2:
    figure

    for k=1:length(t)
        clf

        t_k=t(k);
        x_k=x(k);
        y_k=y(k);


        plot(x_k,y_k, 'go', 'LineWidth', 2, 'MarkerSize', 15)

        hold on
        plot(x,y,'b-', 'LineWidth',2)

        grid on
        xlabel('x')
        ylabel('y')
        zlabel('z')
        title(['Particle a t=', num2str(t_k),'seconds'])
        view([0 0 5])



        %drawnow

        %pause(0.2)

        movieVector(k)=getframe;




    end
end
```

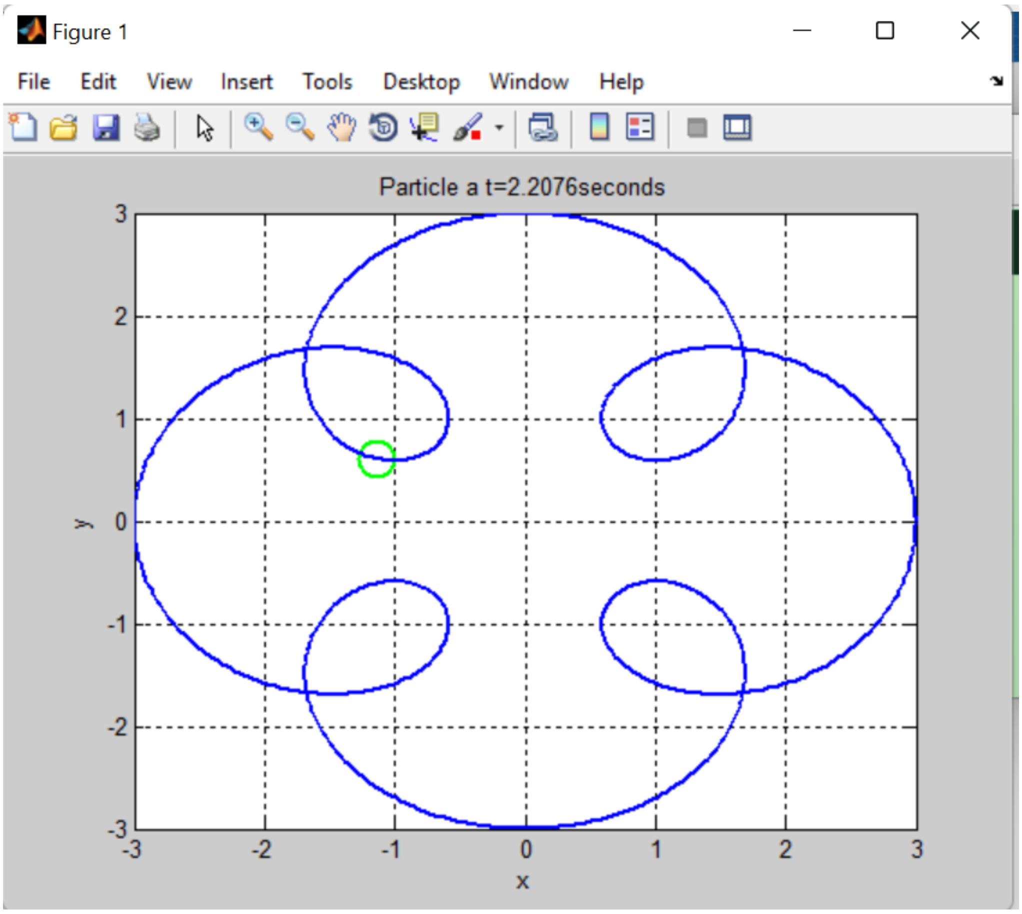
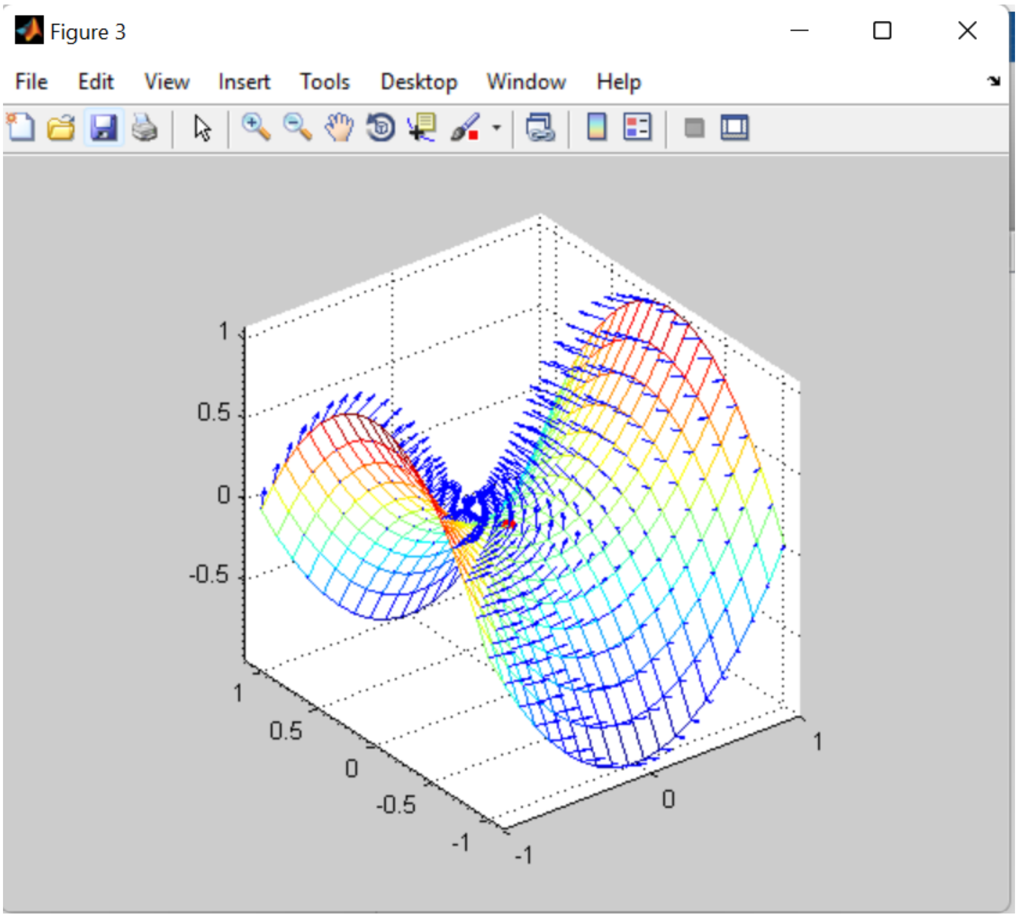


Figure 1
File Edit View Insert Tools Desktop Window Help
Particle a t=2.2076seconds
y
x
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

Saddle Point

```
function[y]=ode1()
    figure
    % generate some data
    [X,Y] = meshgrid( linspace(-1,1,20) );
    Z = X.^2-Y.^2;
    [nx,ny,nz] = surfnorm(X,Y,Z);          % normal vectors
    [az,el,rho] = cart2sph(nx,ny,nz);      % find azimuth and elevation
    [¬,ix] = max(el(:));                   % find maximum elevation
    mesh(X,Y,Z)
    hold on
    scatter3(X(ix),Y(ix),Z(ix),50,'r','filled')       % saddle point
    quiver3(X,Y,Z,nx,ny,nz,'b')                % show normal vectors
    hold off
    axis equal
end
```



Solved Differential Equation

```
function[y]=Differential()
    %% Step 1: Generate Data
    t=linspace(0,2 , 1000);
    x=3.*exp(-5.*t)+2.*exp(-2.*t)
    y=6.*exp(-5.*t)-2.*exp(-2.*t)


    %%Step 2:
    figure

    for k=1:length(t)
        clf

        t_k=t(k);
        x_k=x(k);
        y_k=y(k);


        plot(x_k,y_k, 'go', 'LineWidth', 2, 'MarkerSize', 15)

        hold on
        plot(x,y,'b-', 'LineWidth',2)

        grid on
        xlabel('x')
        ylabel('y')
        zlabel('z')
        title(['Particle a t=', num2str(t_k),'seconds'])

        view([0 0 0.0001])

        %drawnow

        %pause(0.2)

        movieVector(k)=getframe;




    end
end
```

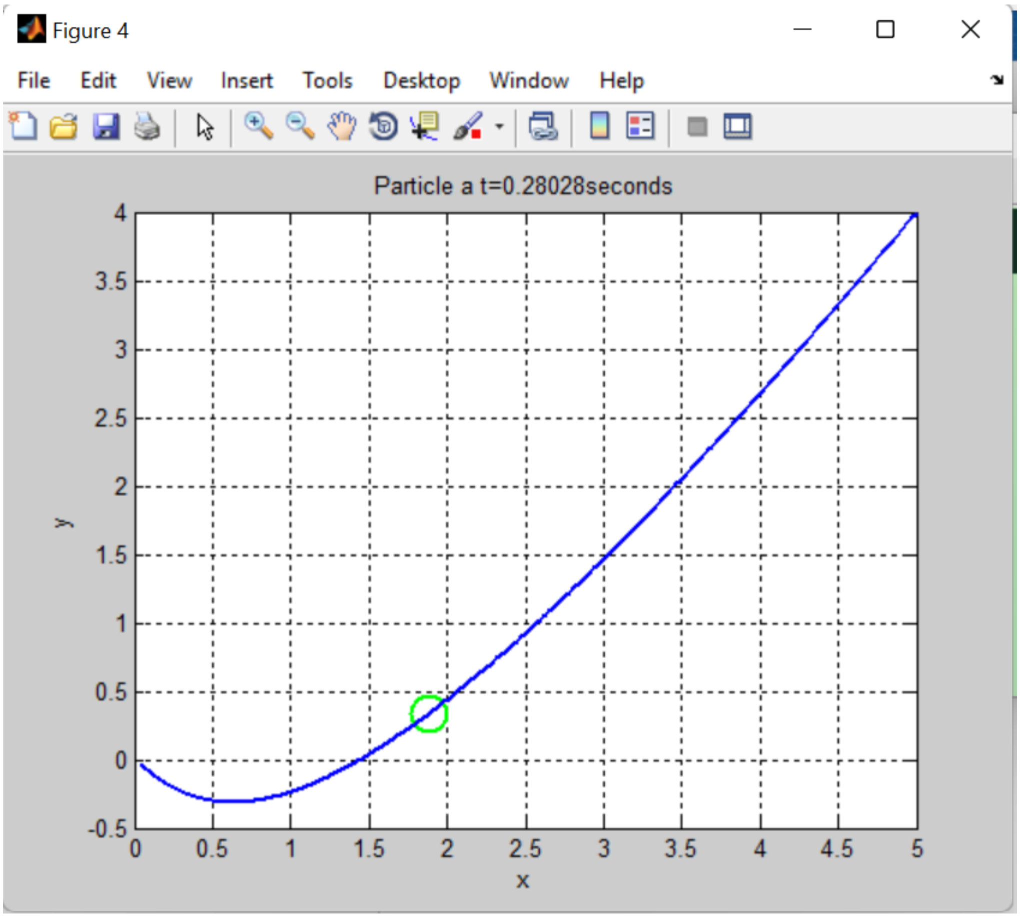


Figure 4
File Edit View Insert Tools Desktop Window Help
Particle a t=0.28028seconds
y
x
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

Spiral Point of a dynamical system with complex eigen-values

```
function[y]=Spiral()
    %% Step 1: Generate Data
    t=linspace(0,2 , 1000);
    for a=1:20
        for b=1:20
            x=[-(a).*sin(5.*t) + (b).*cos(5.*t)].*exp(-2.*t);
            y=[(2.*a).*cos(5.*t) + (2.*b).*sin(5.*t)].*exp(-2.*t);


    %%Step 2:
    figure

    for k=1:length(t)
        clf

        t_k=t(k);
        x_k=x(k);
        y_k=y(k);


        plot(x_k,y_k, 'go', 'LineWidth', 2, 'MarkerSize', 15)

        hold on
        plot(x,y,'b-', 'LineWidth',2)

        grid on
        xlabel('x')
        ylabel('y')
        zlabel('z')
        title(['Particle a t=', num2str(t_k),'seconds'])

        view([0 0 0.0001])

        %drawnow

        %pause(0.2)

        movieVector(k)=getframe;


    end
        end

    end
end
```

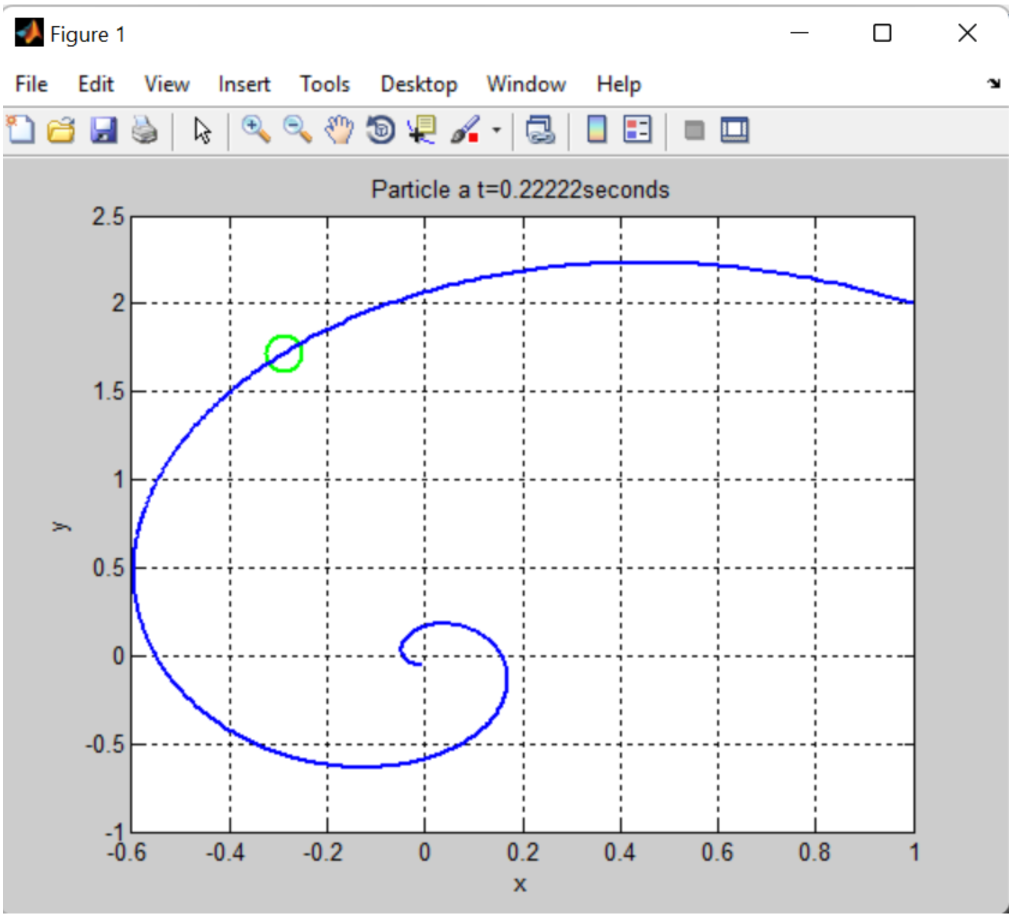


Figure 1
File Edit View Insert Tools Desktop Window Help
Particle a t=0.22222seconds
2.5
2
1.5
1
0.5
0
-0.5
-1
y
-0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
x

Added Code to turn animations into extract-able videos

```
mywriter= VideoWriter('Sphere','MPEG-4');
mywriter.FrameRate=20;

open(mywriter);
writeVideo(mywriter, movieVector);
close(mywriter);
```