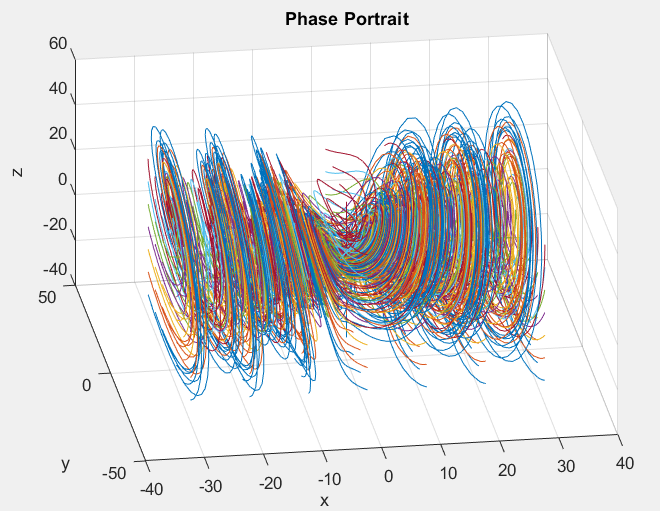
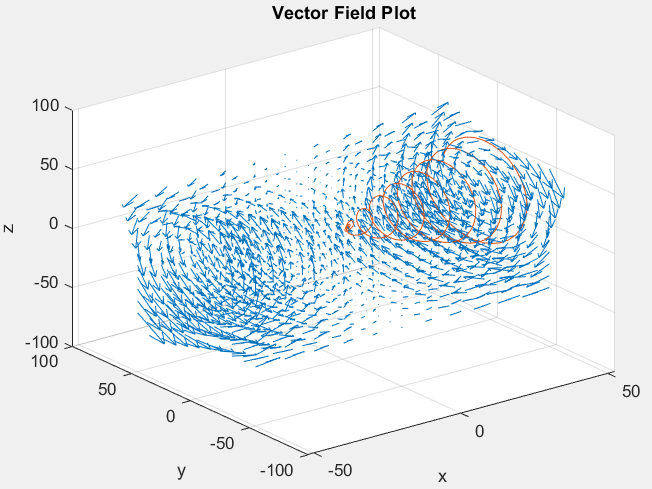
ESE 448, HW 1

1. Lorenz equations with parameters

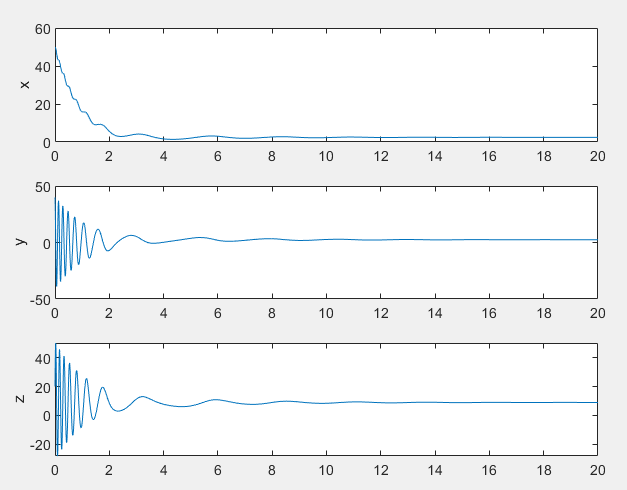
1.1 **Phase Portrait Plot**

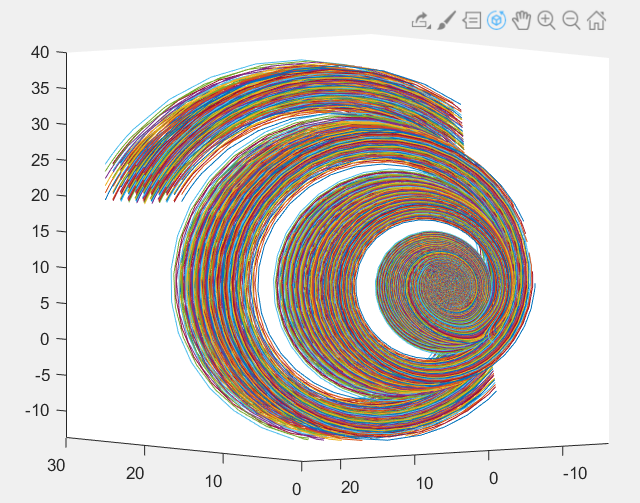


1.2 **Vector Field Plot**



1.3 **Time Series Plot**



*Does the system exhibit chaos?*

No. The figure to the right shows a plot of a grouping with slightly varied initial conditions, which follow similar behavior.

*Does the system oscillate?*

Yes. As shown in the time series plot in part 1.3, the and components have damped oscillations as they approach the stable critical point.

*Are there multiple modes of behavior?*

Yes. There is different behavior in the different directions, (Damped oscillations in , simple damping in as the dominant modes).

*Do trajectories approach a critical point?*

Yes. As shown in the time series in 1.3, the critical point is (2.44, 2.45, 8.97)

%parameters of Lorenz Eq'ns

rho = 10;

sigma = 1.1;

beta = 2/3;

%define system of ODEs

f = @(t,x) [sigma\*(x(2)-x(1));x(1)\*(rho-x(3))-x(2);x(1)\*x(2)-beta\*x(3)];

%Create grid for phase portrait (x,y,z have same values)

x = -30:10:30;

%size of gride

len = length(x)^3;

%create empty array to hold ode45 sol'ns for all points in grid

t\_phase = zeros(len,4000,1);

X\_phase = zeros(len,4000,3);

%nth data point (n=1:len)

n=1;

for i=1:length(x)

for j=1:length(x)

for k=1:length(x)

%solve for each point on grid

[t,X]= ode45(f, [0 20], [x(i) x(j) x(k)]);

soln\_len = length(t);

%append 0s to sol'n to make it same dims as t\_phase, X\_phase

t(numel(t\_phase(n,:))) = 0;

X(numel(t\_phase(n,:)),:) = 0;

%store the values of the sol'n

t\_phase(n,:) = t;

t\_phase(n,1) = soln\_len; %stores the num steps of the sol'n

X\_phase(n,:,:) = X;

n=n+1;

end

end

end

% PHASE PORTRAIT

figure;

%plots all sol'ns

for n=1:len

%plots only the number of steps in the sol'n for each initial condition

plot3(X\_phase(n,1:t\_phase(n,1),1), X\_phase(n,1:t\_phase(n,1),2), X\_phase(n,1:t\_phase(n,1),3));

hold on;

end

grid on;

title('Phase Portrait');

xlabel('x');

ylabel('y');

zlabel('z');

%VECTOR FIELD

%create grid

[x,y,z] = meshgrid(-50:10:50);

%calculate force vector at each point

u = sigma.\*(y-x); %x\_dot

v = x.\*(rho-z); %y\_dot

w = x.\*y-beta.\*z; %z\_dot

%calculate one random sol'n to plot for reference in field graph

[t,X] = ode45(f, [0,20], [50,40,20]);

figure;

%quiver plots vector field for each point in the mesh

quiver3(x,y,z,u,v,w,3);

hold on;

plot3(X(:,1), X(:,2), X(:,3));

grid on;

title('Vector Field Plot');

xlabel('x');

ylabel('y');

zlabel('z');

%TIME SERIES

%uses sol'n from ode45 in vector field

figure;

%subplots x,y,z over time for sol'n

subplot(3,1,1);

plot(t,X(:,1));

ylabel('x');

subplot(3,1,2);

plot(t,X(:,2));

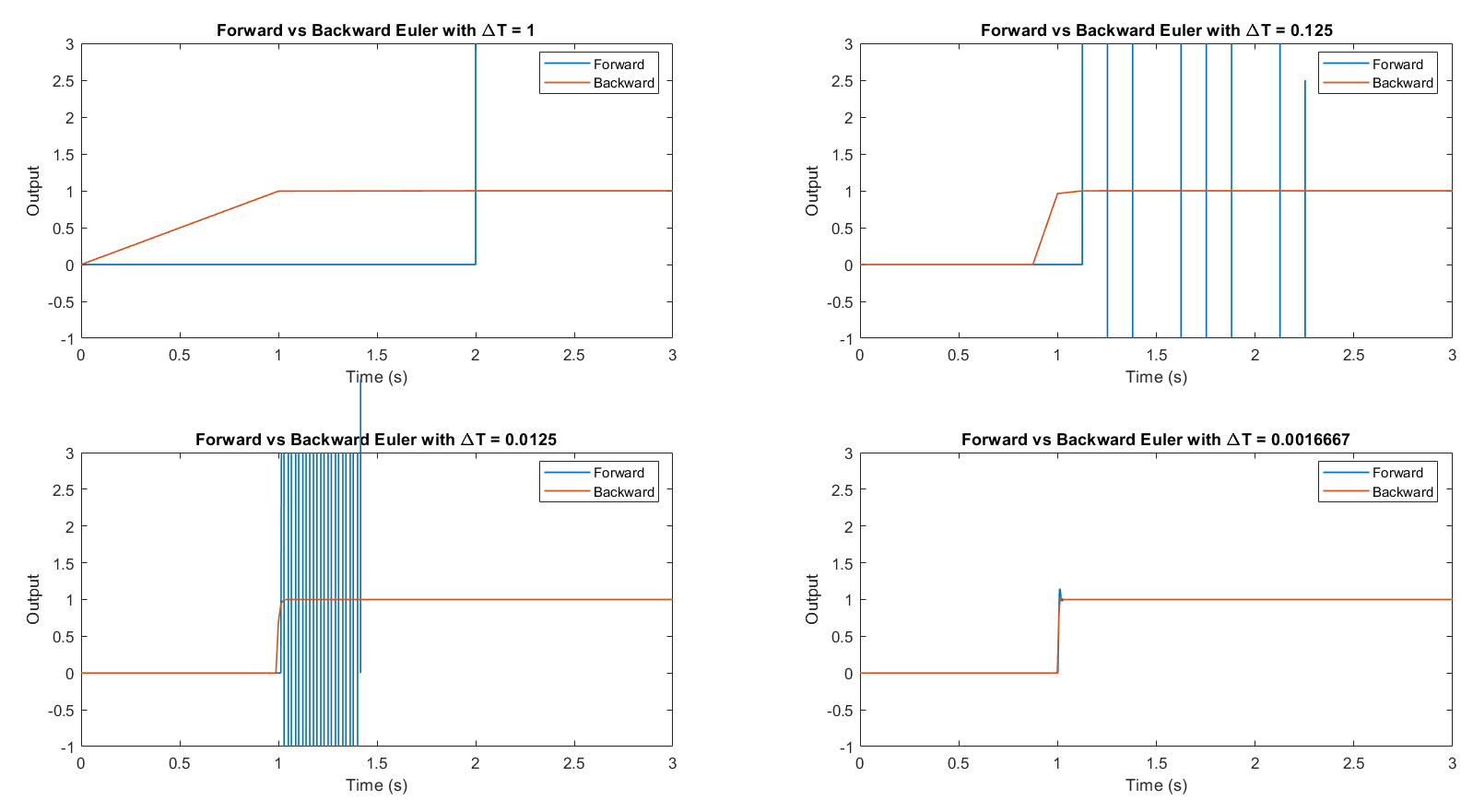
ylabel('y');

subplot(3,1,3);

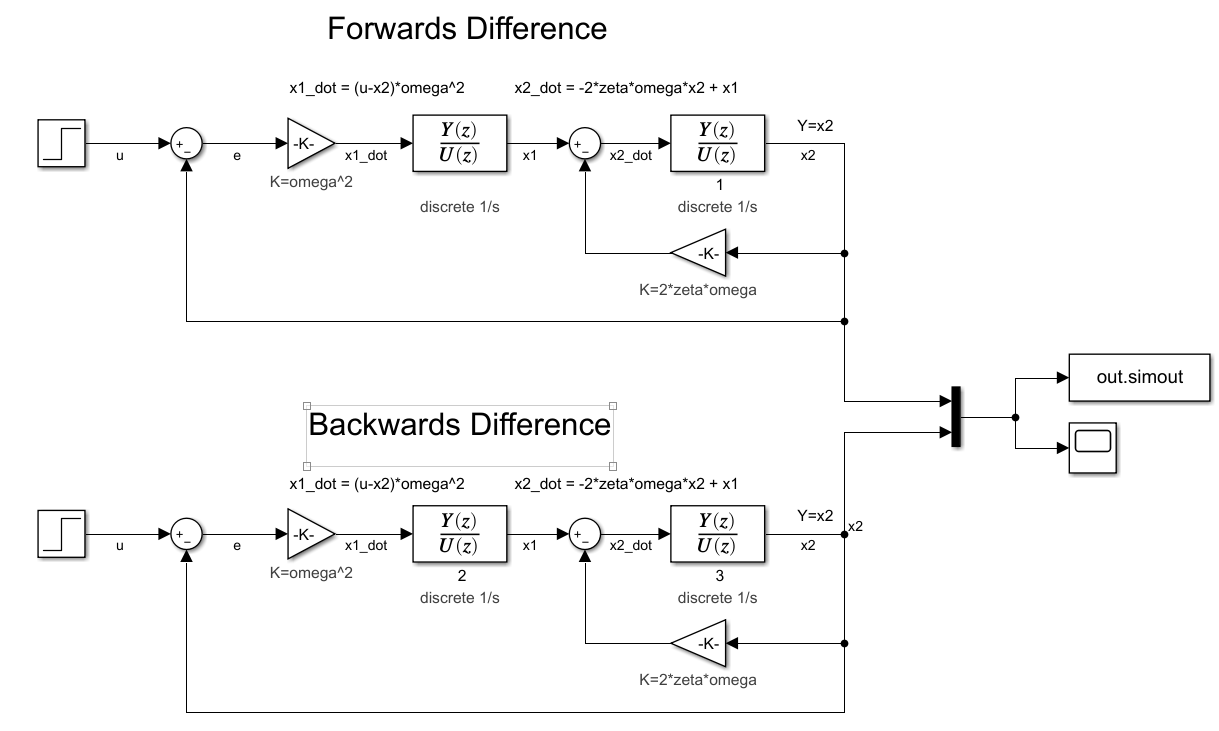
plot(t,X(:,3));

ylabel('z');

2.1

For this problem, I wrote a script that ran the forward and backward euler transforms for each time step in and plotted them next to eachother. The backward difference was stable for all timesteps, but the forwards one was stable only for the smallest.

Simulink Model:



%define constants and array for dT

zeta = 1/sqrt(2);

omega = 294;

dT = [1 1/8 1/80 1/600];

figure('Color', [1,1,1]);

for i=1:length(dT)

%define z for ith timestep

z = tf('z',dT(i));

%forwards and backwards transform to replace s w/ z

s\_f = @(z) (z-1)/dT(i);

s\_b = @(z) (1-z^-1)/dT(i);

%get integrator transfer functions in terms of z

Hf = 1/s\_f(z);

Hb = 1/s\_b(z);

%get numerator and denominator arrays for simulink values

[num\_f,dem\_f] = tfdata(Hf, 'v');

[num\_b,dem\_b] = tfdata(Hb, 'v');

%run simulink model from [0 3] seconds and capture the outputs

output = sim('ESE\_448\_HW\_1\_simulink\_model', [0 3]);

%ith subplot is (half width, half height, ith position)

subplot(2,2,i);

%plot forwards

plot(output.simout.Time, output.simout.Data(:,1),'LineWidth',1);

hold all;

%plot backwards

plot(output.simout.Time, output.simout.Data(:,2),'LineStyle', '-','LineWidth',1);

xlabel('Time (s)');

ylabel('Output');

%limit x, y so unstable parts aren't graphed

xlim([0 3]);

ylim([-1 3]);

title(['Forward vs Backward Euler with \DeltaT = ' num2str(dT(i))]);

legend({'Forward', 'Backward'},'Location','Best');

end