No.

Date

ME564 L19

recall Lorentz equation

↳ sensitive dependance on initial condition

a slight perturbation could lead to a total different state

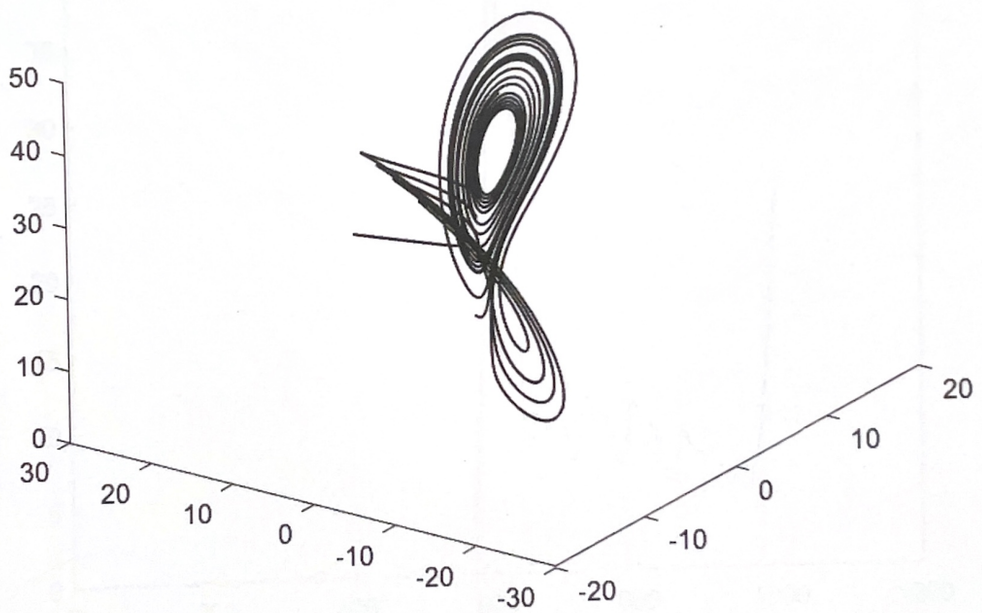
↳ Chaotic

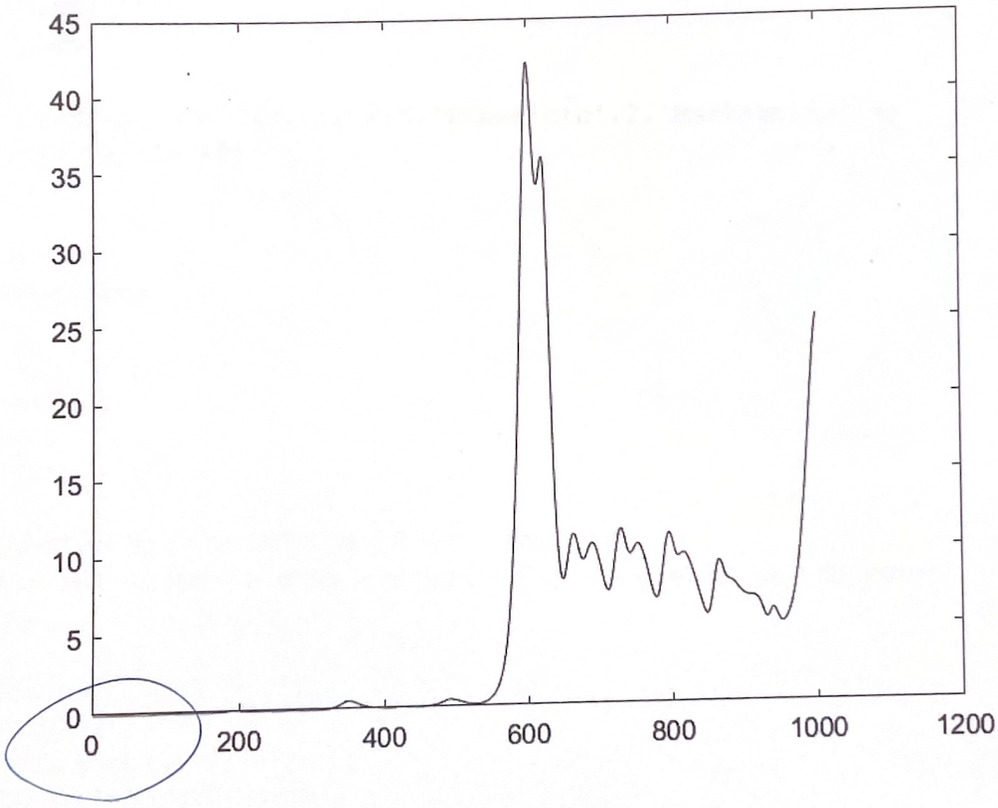
Matlab interpreted language.

```
close all; clear all; clc;

% Lorenz's parameters (chaotic)
sigma = 10;
beta = 8/3;
rho = 28;

% Initial condition
y0=[-8; 8; 27];
eps = [0.001; 0.001; 0.001];

% Compute trajectory
dt =0.01;
tspan=[0:dt:10];

% Y(:,1)=y0;
% yin = y0;
% for i=1:length(tspan)
%     time = i*dt;
%     yout = rk4singlestep(@(t,y)lorenz(t,y,sigma,beta,rho),dt,time,yin);
%     Y = [Y yout];
%     yin = yout;
% end
[t,Y] = ode45(@(t,y)lorenz(t,y,sigma,beta,rho),tspan,y0);
plot3(Y(:,1),Y(:,2),Y(:,3),'b', 'LineWidth', 1)

hold on
[t,y] = ode45(@(t,y)lorenz(t,y,sigma,beta,rho),tspan,y0 + eps);
plot3(y(:,1),y(:,2),y(:,3),'r', 'LineWidth', 1)

for i = 1:1001
    diff(i) = norm(Y(i,:)-y(i,:));
end

figure
plot(diff)


function yout = rk4singlestep(fun,dt,t0,y0)

f1 = fun(t0,y0);
% size(f1)
% size(y0)
f2 = fun(t0+dt/2,y0+(dt/2)*f1);
f3 = fun(t0+dt/2,y0+(dt/2)*f2);
f4 = fun(t0+dt,y0+dt*f3);

yout = y0 + (dt/6)*(f1+2*f2+2*f3+f4);
end
```

→ chaos, diverge slightly after awhile (w/ different initial condition.

```
function dy = lorenz(t,y,sigma,beta,rho)
% y is a three dimensional state-vector
dy = [
sigma*(y(2)-y(1));
y(1)*(rho-y(3))-y(2);
y(1)*y(2)-beta*y(3);
];
end
```


50
40
30
20
10
0
30
20
10
0
-10
-20
-30
-20
-10
0
10
20



exponetial growth even @ the beginning.

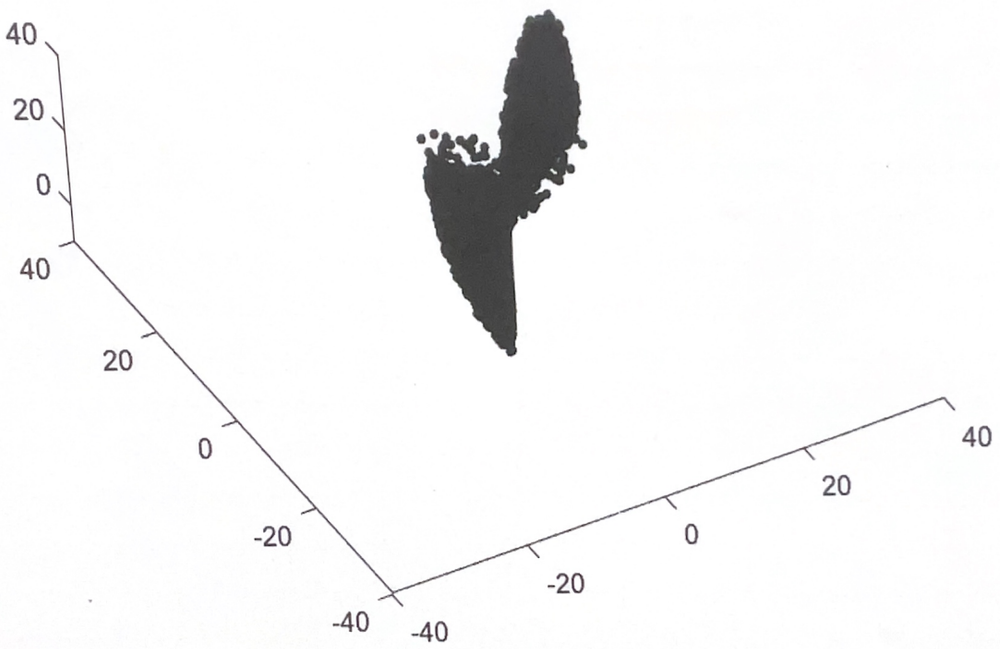
```
clear all

% Lorenz's parameters (chaotic)
sigma = 10;
beta = 8/3;
rho = 28;

% Initial condition - large cube of points
xvec = -20:2:20;
yvec = -20:2:20;
zvec = -20:2:20;
[x0,y0,z0] = meshgrid(xvec,yvec,zvec);
yIC(1,:,:,:) = x0;
yIC(2,:,:,:) = y0;
yIC(3,:,:,:) = z0;

plot3(yIC(1,:),yIC(2,:),yIC(3,:),'r.','LineWidth',2,'MarkerSize',4)
axis([-40 40 -40 40 -40 40])
view(20,40);
drawnow

%% Compute trajectory
dt =0.01;
duration = 4
tspan=[0,duration];
L = duration/dt;
yparticles = yIC;

% this code is slow because MATLAB is not compiled
% we use nested for loops to step through every single IC in the cube
% one at a time...
for step=1:L
    time = step*dt
    for i=1:length(xvec)
        for j=1:length(yvec)
            for k=1:length(zvec)
                yin = yparticles(:,i,j,k);
                yout = rk4singlestep(@(t,y)lorenz(t,y,sigma,beta,rho),dt,time,yin);
                yparticles(:,i,j,k) = yout;
            end
        end
    end
    plot3(yparticles(1,:),yparticles(2,:),yparticles(3,:),'r.','LineWidth',↙
2,'MarkerSize',4)
    view(20,40);
    axis([-40 40 -40 40 -10 40])
    drawnow
end
```

```
function yout = rk4singlestep(fun,dt,t0,y0)

f1 = fun(t0,y0);
% size(f1)
% size(y0)
f2 = fun(t0+dt/2,y0+(dt/2)*f1);
f3 = fun(t0+dt/2,y0+(dt/2)*f2);
f4 = fun(t0+dt,y0+dt*f3);

yout = y0 + (dt/6)*(f1+2*f2+2*f3+f4);
end

function dy = lorenz(t,y,sigma,beta,rho)
% y is a three dimensional state-vector
dy = [
sigma*(y(2)-y(1));
y(1)*(rho-y(3))-y(2);
y(1)*y(2)-beta*y(3);
];
end
```

```
clear all

% Lorenz's parameters (chaotic)
sigma = 10;
beta = 8/3;
rho = 28;

% Initial condition 1 - Large cube of data
y0=[0 0 0];
xvec = -20:2:20;
yvec = -20:2:20;
zvec = -20:2:20;
% % Initial condition 2 - small cube around initial condition from last class
% y0=[-8; 8; 27];
% xvec = -1:.1:1;
% yvec = -1:.1:1;
% zvec = -1:.1:1;
% % Initial condition 3 - even smaller cube around initial condition
% y0=[-8; 8; 27];
% xvec = -.1:.01:.1;
% yvec = -.1:.01:.1;
% zvec = -.1:.01:.1;
[x0,y0,z0] = meshgrid(xvec+y0(1),yvec+y0(2),zvec+y0(3));
yIC(1,:,:,:) = x0;
yIC(2,:,:,:) = y0;
yIC(3,:,:,:) = z0;

plot3(yIC(1,:),yIC(2,:),yIC(3,:),'r.','LineWidth',2,'MarkerSize',10)
axis([-40 40 -40 40 -40 40])
view(20,40);
drawnow

%% Compute trajectory
dt =0.01;
duration = 4
tspan=[0,duration];
L = duration/dt;
yin = yIC;

for step = 1:L
    time = step*dt
    yout = rk4singlestep(@(t,y)lorenz3D(t,y,sigma,beta,rho),dt,time,yin);
    yin = yout;
    plot3(yout(1,:),yout(2,:),yout(3,:),'r.','LineWidth',2,'MarkerSize',10)
    view(20+360*step/L,40);
    axis([-40 40 -40 40 -10 40])
    drawnow
end
```

```
function dy = lorenz3D(t,y,sigma,beta,rho)
% y is a three dimensional state-vector
dy = [
sigma*(y(2,:,:,:)-y(1,:,:,:));
y(1,:,:,:).*(rho-y(3,:,:,:))-y(2,:,:,:);
y(1,:,:,:).*y(2,:,:,:)-beta*y(3,:,:,:);
];
end

function yout = rk4singlestep(fun,dt,t0,y0)

f1 = fun(t0,y0);
% size(f1)
% size(y0)
f2 = fun(t0+dt/2,y0+(dt/2)*f1);
f3 = fun(t0+dt/2,y0+(dt/2)*f2);
f4 = fun(t0+dt,y0+dt*f3);

yout = y0 + (dt/6)*(f1+2*f2+2*f3+f4);
end
```

matlab handles matrix multiplication faster!


40
20
0
40
20
0
-20
-40
-40
-20
0
20
40