Contents

```
clear; close all; clc;
```

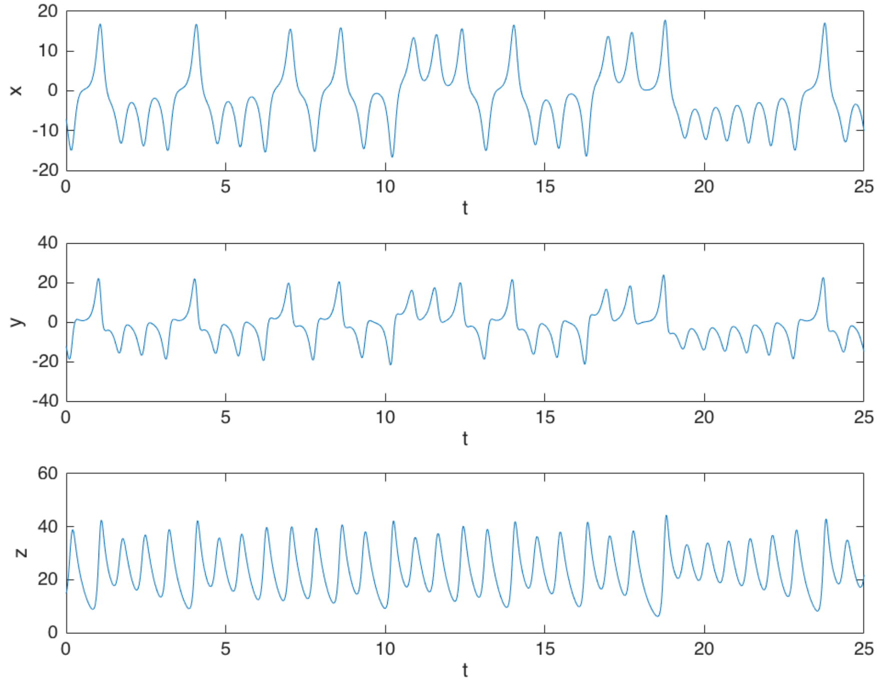
parameters for the Lorenz equation

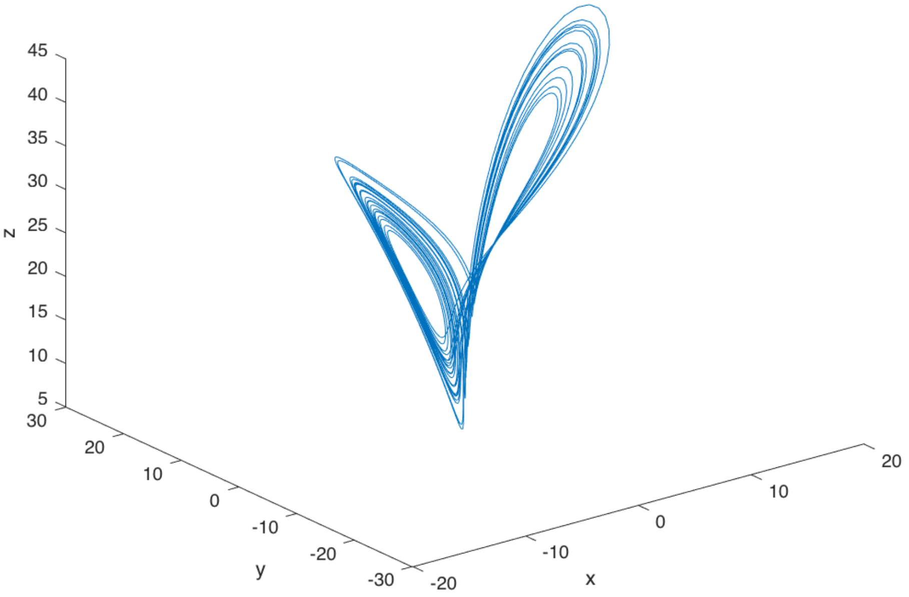
see wikipedia page for more details

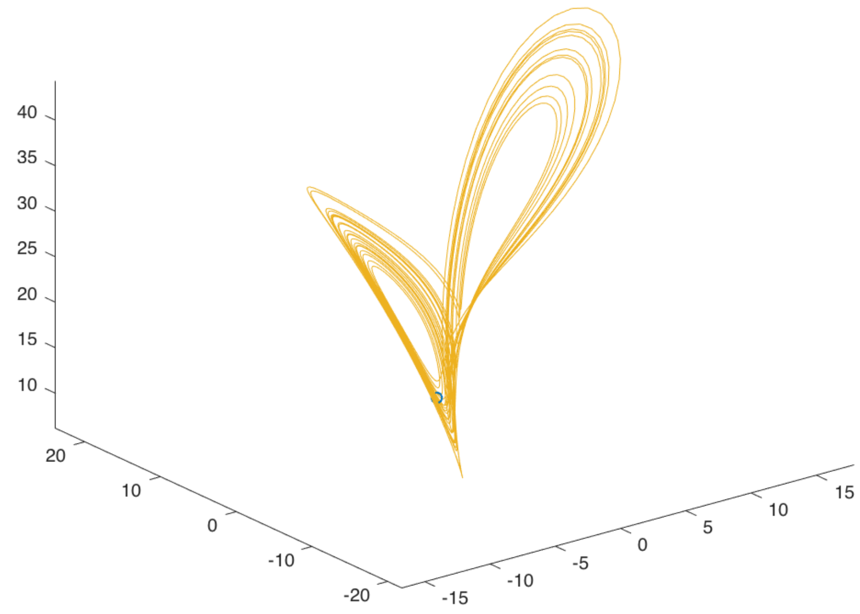
```
param.rho = 28;
param.sigma = 10;
param.beta = 8/3;
```

simulate for some initial conditions

```
tMax = 25;
tSpan = linspace(0,tMax,tMax*100); % time to simulate

stateVar0 = [-7.152697136677542; % some initial conditions close to the chaotic attractor
 -12.240110163291554;
  14.803748615043197];
options = odeset('reltol',1e-8,'abstol',1e-8);

[tList,stateVarList] = ode45(@odeLorenz,tSpan,stateVar0,options,param);

figure(1);
subplot(311); plot(tList,stateVarList(:,1)); xlabel('t'); ylabel('x');
subplot(312); plot(tList,stateVarList(:,2)); xlabel('t'); ylabel('y');
subplot(313); plot(tList,stateVarList(:,3)); xlabel('t'); ylabel('z');

figure(2);
plot3(stateVarList(:,1),stateVarList(:,2),stateVarList(:,3));
xlabel('x'); ylabel('y'); zlabel('z');

figure(3); % xlabel('x'); ylabel('y'); zlabel('z'); hold on;
comet3(stateVarList(:,1),stateVarList(:,2),stateVarList(:,3),0.001);
```


20
10
0
-10
-20
x
0
5
10
15
20
25
t
40
20
0
-20
-40
y
0
5
10
15
20
25
t
60
40
20
0
z
0
5
10
15
20
25
t

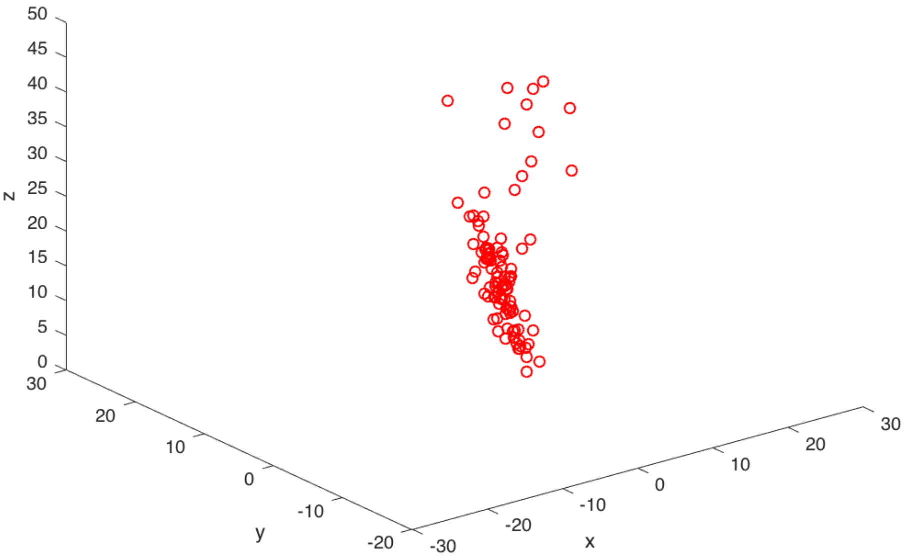

45
40
35
30
25
20
15
10
5
z
30
20
10
0
-10
-20
-30
y
-20
-10
0
10
20
x



simulating LOTs of initial conditions starting very close to each other

```
tMax = 15;
tSpan = linspace(0,tMax,tMax*25); % time to simulate

stateVar0 = [-7.152697136677542; % some initial conditions close to the chaotic attractor
 -12.240110163291554;
  14.803748615043197];
numInitialConditions = 100;
stateVar0Matrix = repmat(stateVar0,1,numInitialConditions) + ...
    1e-2*randn(3,numInitialConditions); % a whole bunch of initial conditions VERY close to each other
stateVarStore = cell(numInitialConditions,1);
for i = 1:numInitialConditions
    stateVar0 = stateVar0Matrix(:,i);
    [tList,stateVarStore{i}] = ode45(@odeLorenz,tSpan,stateVar0,options,param);
end
figure(4);
for j = 1:length(tList)
    a = [];
    for i = 1:numInitialConditions
        a = [a; stateVarStore{i}(j,1),stateVarStore{i}(j,2),stateVarStore{i}(j,3)];
    end
     plot3(a(:,1),a(:,2),a(:,3),'ro'); hold on;
     xlabel('x'); ylabel('y'); zlabel('z');
     title('Evolution of many initial conditions starting very close to each other');
     xlim([-30 30]); ylim([-20 30]); zlim([0 50]);
%      pause
    pause(0.01);
    hold off;
end
```

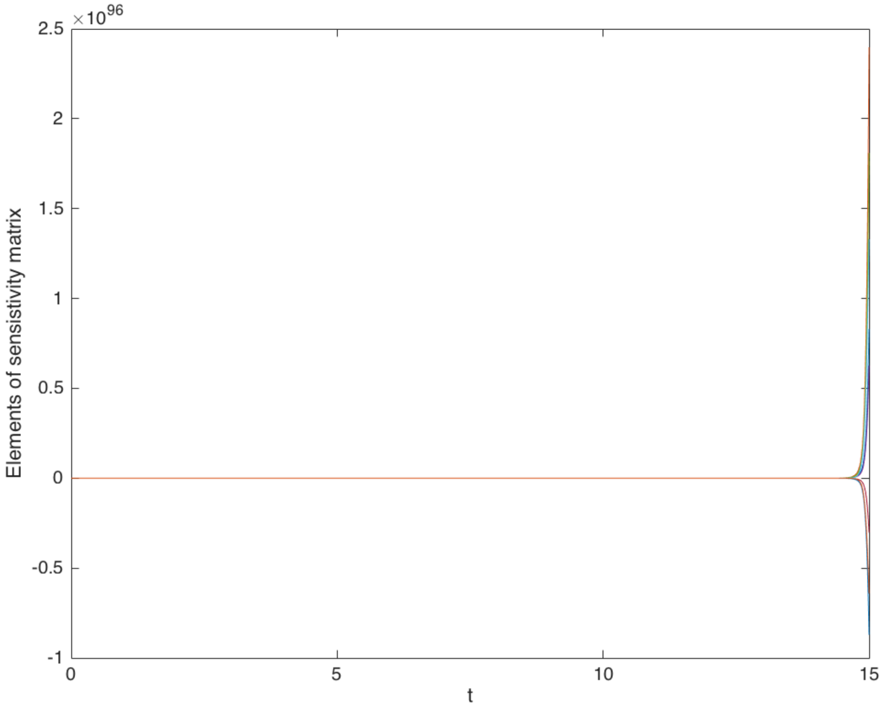
Evolution of many initial conditions starting very close to each other



simulating the sensitivity equations along with the original ODE

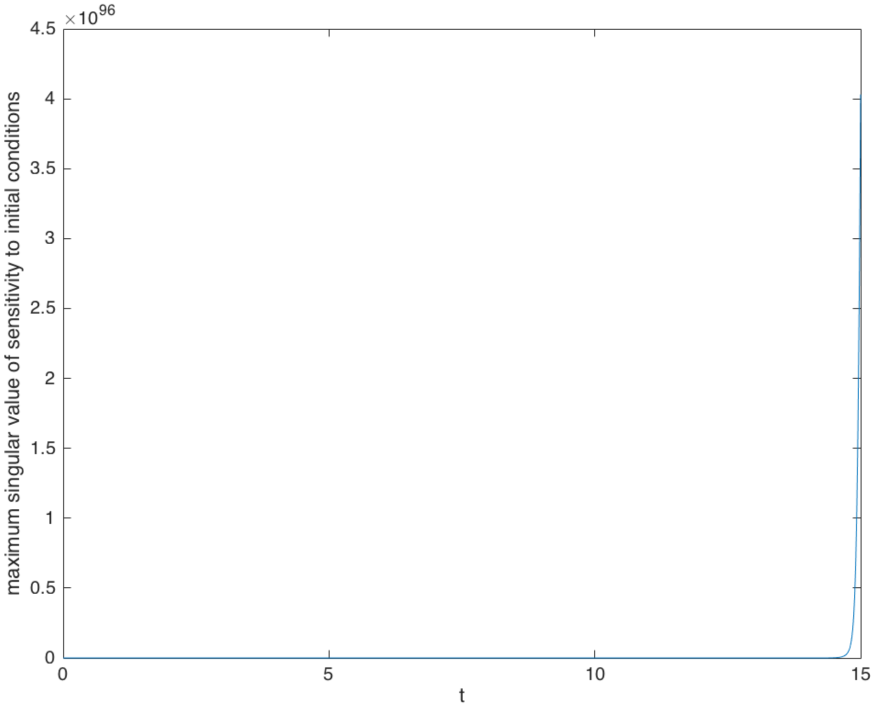
thus, we are simulating 12 ODEs instead of 3 ODEs

```
tMax = 15;
tSpan = linspace(0,tMax,tMax*100); % time to simulate

DsDs0_0 = eye(3); % how initial conditions change wrt initial conditions

% append state to sensitivity matrix, so they can be simultaneously simulated
% of course, the 3x3 matrix needs to be reshaped into a 9x1 matrix, so that
% ode45 can handle it
stateAndSensitivity0 = [stateVar0; reshape(DsDs0_0,numel(DsDs0_0),1)];

[tList,stateAndSensitivityList] = ode45(@odeLorenzWithSensitivity,tSpan,stateAndSensitivity0,options,param);

% plot the individual sensitivity matrix components
figure(5);
plot(tList,stateAndSensitivityList(:,4:12)');
xlabel('t'); ylabel('Elements of sensistivity matrix');

% note that the sensitivity equations characterize the expansion for
% initial conditions that are arbitrarily close to each other. This
% expansion grows exponentially. But the expansion slows down when the
% initial conditions are further apart, for instance, manifested in
% boundedness of the trajectories.
```



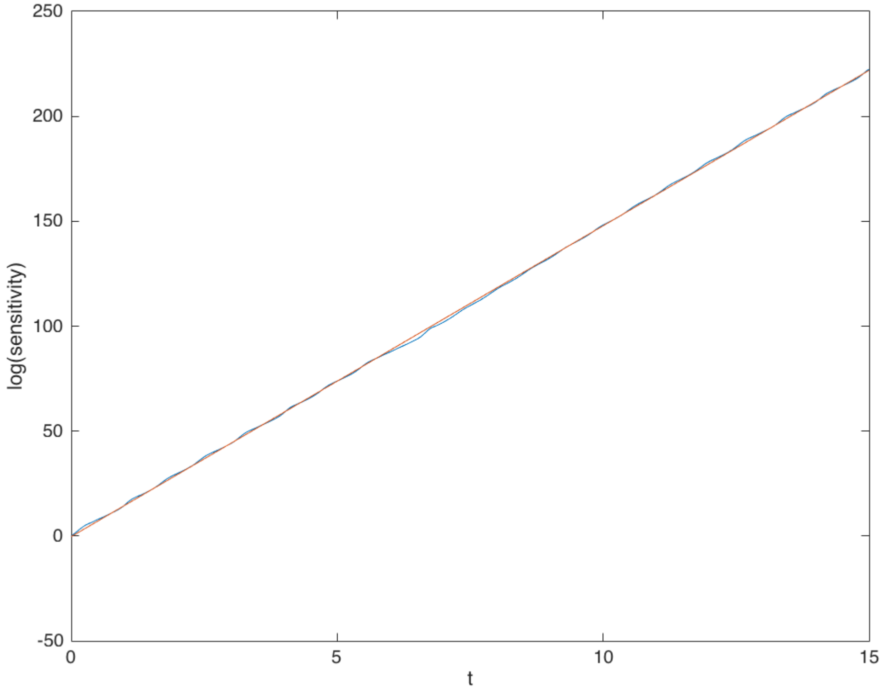
plot the maximum singular value of the sensitivity matrix as a function of time

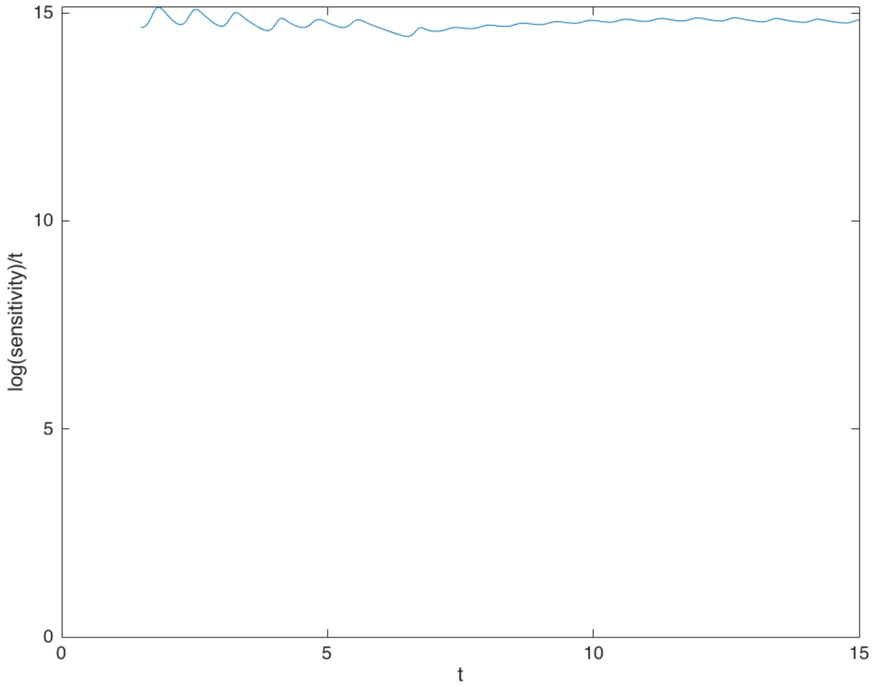
```
maxSingularValueList = [];
for i = 1:length(tList)
   DsDs0 = stateAndSensitivityList(i,4:12);
   DsDs0 = reshape(DsDs0,3,3);
   maxSingularValueList = [maxSingularValueList; max(svd(DsDs0))];
end

figure(6);
plot(tList,maxSingularValueList);
xlabel('t'); ylabel('maximum singular value of sensitivity to initial conditions');
```



```
figure(7);
logSensitivity = log(maxSingularValueList);
plot(tList,logSensitivity);
xlabel('t'); ylabel('log(sensitivity)');

% time averaged maximal Lyapunov exponent is roughly the slope of the linear fit to the log of the
% max singular value of the sensitivity matrix (eventually)
coeffsRegression = regress(logSensitivity,[ones(size(tList)) tList]); % doing a linear fit
LyapunovExponentEstimate1 = coeffsRegression(2)
hold on; plot(tList,coeffsRegression(1)+coeffsRegression(2)*tList)

% maximal Lyapunov exponent is the limit of the 1/t times log of the
% max singular value of the sensitivity matrix
figure(8);
temp = logSensitivity(0.1*end:end)./tList(0.1*end:end);
plot(tList(0.1*end:end),temp);
xlabel('t'); ylabel('log(sensitivity)/t');
ylim([0 max(temp)]);
LyapunovExponentEstimate2 = temp(end)
```

```
LyapunovExponentEstimate1 =

  14.813254568940248


LyapunovExponentEstimate2 =

  14.829463308501168
```





Published with MATLAB® R2015b