Massachusetts Institute of Technology
Department of Mechanical Engineering

2.003J/1.053J Dynamics & Control I

Fall 2007

Homework 8 Solution

Problem 8.1 : Nonlinear parametric pendulum

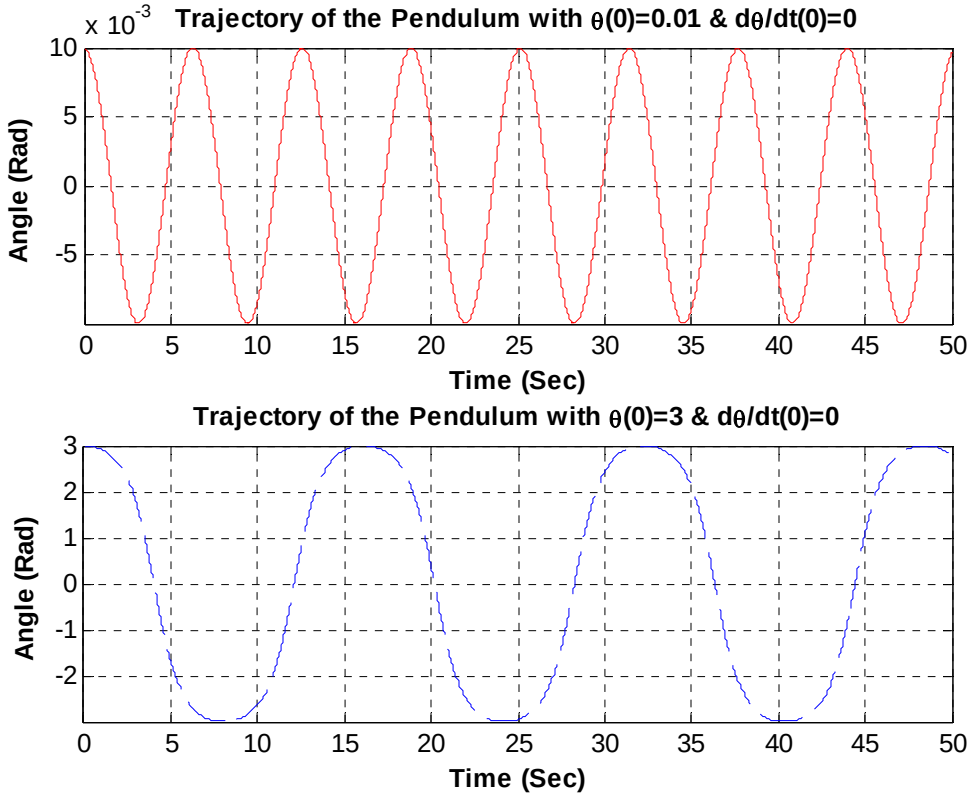
i)

Nonlinear pendulum without damping and force can be simplified to $\frac{d^2\theta}{dt^2} + \omega_0^2 \sin\theta = 0$.

Simulation results with given two initial conditions (small angle motion and large angle motion) are compared (See the below codes and figure). Runge-Kutta method is used, and simulation time is up to 50 second. For the increase of simulation accuracy, maximum time step is set to 0.01 second. In the figure, for a), small angular one is definitely sinusoidal, and its period (6.28 second) corresponds to the reciprocal of the natural frequency (1 rad/sec=1/2π Hz) of the pendulum, and it confirms the period is independent of the initial conditions, and $\sin\theta \approx \theta$ is a good approximation since natural frequency is closely related to the period of the response with no force in linear system. However, the period increases as the initial angle becomes larger. For b), the shape in the figure is not sinusoidal any more though it's still periodic. Therefore, the period is related to the initial condition for the initial condition with large angle.

```
function P81_i
% Problem 8.1 : Nonlinear parametric pendulum
% Part i. Unforced and undamped pendulum

% Define some constants
omega0=1;   % Natural frequency of the pendulum

% Define some parameters for using Runge-Kutta metod
```

```
tspan=[0 50];    % Time span to be obtained
% Choose time step smaller than 0.01 sec
optn=odeset('InitialStep',1e-2,'MaxStep',1e-2);

% Simulation with small angle (theta(0)=0.01 radian)
[T1,Y1]=ode45(@(t,x) Npp(t,x,omega0),tspan,[0.01 0],optn);

% Simulation with large angle (theta(0)=3 radian)
[T2,Y2]=ode45(@(t,x) Npp(t,x,omega0),tspan,[3 0],optn);

% Plot and compare simulation results
figure(1);
subplot(2,1,1); plot(T1,Y1(:,1),'r'); grid on; axis tight;
xlabel('\bfTime (Sec)'); ylabel('\bfAngle (Rad)');
title('\bfTrajectory of the Pendulum with \theta(0)=0.01 &
d\theta/dt(0)=0');
subplot(2,1,2); plot(T2,Y2(:,1),'b--'); grid on; axis tight;
xlabel('\bfTime (Sec)'); ylabel('\bfAngle (Rad)');
title('\bfTrajectory of the Pendulum with \theta(0)=3 &
d\theta/dt(0)=0');
end

function dx=Npp(t,x,omega0)
% Describe nonlinear parametric pendulum motion
dx(1,1)=x(2);
dx(2,1)=-omega0^2*sin(x(1));
end
```

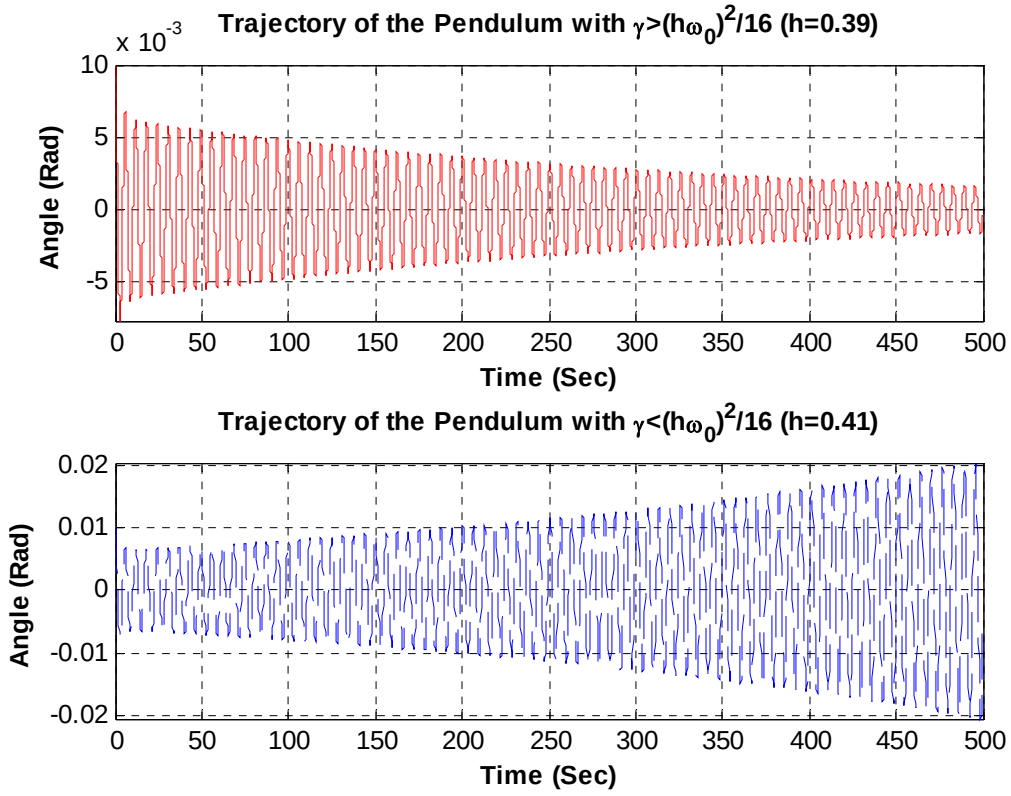


ii)

From $\gamma^2 < (h\omega_0)^2/16$, the pendulum become stable when $h > 0.4$ with given parameters in this problem. To verify the above prediction. Two h values are chosen: the ones slightly smaller and larger the critical value $h = 0.4$ ($h = 0.39$ and $h = 0.41$) From the below figure generated by the following m-codes, it is shown that the pendulum with $h = 0.39$ becomes stable since the angular position approaches to zero as time goes on (top plot), but the one with $h = 0.41$ turns out to be unstable since the angular motion increases (bottom plot).

```
function P81_ii
% Problem 8.1 : Nonlinear parametric pendulum
% Part ii. Stability

% Define some constants
omega0=1;       % Natural frequency of the pendulum
gamma=0.1;      % Damping coefficient
```

```
h=[0.39 0.41];  % Two amplitudes of forcing

% Define some parameters for using Runge-Kutta metod
tspan=[0 500];     % Time span to be obtained
theta0=[0.01,0];    % Initial conditions

% Choose time step smaller than 0.01 sec
optn=odeset('InitialStep',1e-2,'MaxStep',1e-2);

% Simulation with the amplitude of forcing smaller than critical value
[T1,Y1]=ode45(@(t,x) Npp(t,x,gamma,omega0,h(1)),tspan,theta0,optn);

% Simulation with the amplitude of forcing larger than critical value
[T2,Y2]=ode45(@(t,x) Npp(t,x,gamma,omega0,h(2)),tspan,theta0);

% Plot and compare simulation results
figure(1);
subplot(2,1,1); plot(T1,Y1(:,1),'r'); grid on; axis tight;
xlabel('\bfTime (Sec)'); ylabel('\bfAngle (Rad)');
title('\bfTrajectory of the Pendulum with \gamma>(h\omega_0)^2/16
(h=0.39)');
subplot(2,1,2); plot(T2,Y2(:,1),'b--'); grid on; axis tight;
xlabel('\bfTime (Sec)'); ylabel('\bfAngle (Rad)');
title('\bfTrajectory of the Pendulum with \gamma<(h\omega_0)^2/16
(h=0.41)');
end

function dx=Npp(t,x,gamma,omega0,h)
% Describe nonlinear parametric pendulum motion
dx(1,1)=x(2);
dx(2,1)=-2*gamma*x(2)-omega0^2*(1+h*cos(2*omega0*t))*sin(x(1));
end
```



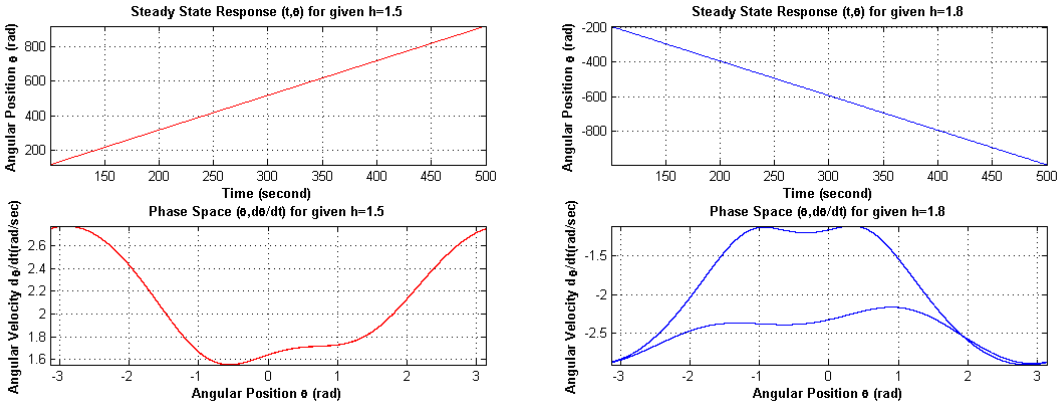
iii)

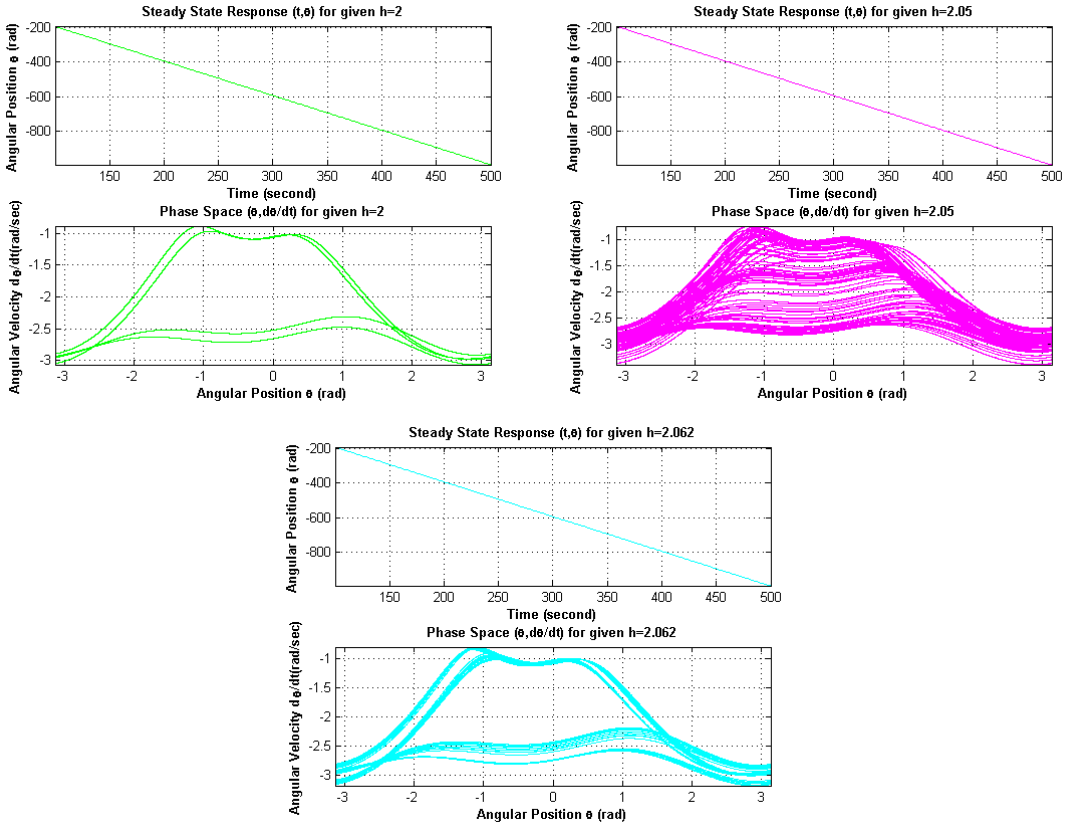
With the below m-codes, 5 plots are represented with different h (But, same initial conditions chosen arbitrarily). For plotting time series, all of them look so similar; the absolute values of slopes are almost same. However, their differences are apparent when they are plotted in the phase space. At $h = 1.50$, a given angular position has only one corresponding angular velocity (single curve), which means the period of the response is 2π second since $\omega_0 = 1$ rad/sec. At $h = 1.80$, a given angular position has two corresponding angular velocities (two curves). It means that the pendulum follows one of two trajectories first, and the other next time. So, the pendulum rotates along the same trajectory with every other rotation. It confirms that the period is now doubled, compared with the one at $h = 1.50$. (If you plot the response along the angular position from -2π to 2π, the trajectory is just single curve.) It is called 'period doubling', and its period is 4π second. Similar phenomenon (period doubling) happens again at $h = 2.00$. (Its period is 8π second.) For $h = 2.05$, its period is close to 128π second or more since so many period doubling happen between $h = 2.00$ and $h = 2.05$. Between $h = 2.05$ and

$h = 2.62$, it period decreases down to about 32π second. It's a general behavior of period change in the nonlinear parametric pendulum.

```
function P81_iii
% Problem 8.1 : Nonlinear parametric pendulum
% Part iii. Period doubling

% Define some constants
omega0=1;      % Natural frequency of the pendulum
gamma=0.1;     % Damping coefficient
h=[ 1.50 1.80 2.00 2.05 2.062];  % Five amplitudes of forcing

% Define some parameters for using Runge-Kutta metod
tspan=[0 500];     % Time span to be obtained
theta0=[3,0];    % Initial conditions

% Choose time step smaller than 0.01 sec
optn=odeset('InitialStep',1e-2,'MaxStep',1e-2);

% Define plot color set
col={'r' ; 'b' ; 'g' ; 'm' ; 'c'};

for i=1:length(h)
   % Simulation with different h
   [T,Y]=ode45(@(t,x) Npp(t,x,gamma,omega0,h(i)),tspan,theta0,optn);
   % Remove transient response (remove data smaller than 100 second)
   Y=Y(find(T>=100),:);
   T=T(find(T>=100),:);
   % Generate figure
   figure(i);
   % Plot steady state response for given h
   subplot(2,1,1); plot(T,Y(:,1),col{i});
   grid on; axis tight;
   xlabel('\bfTime (second)');
   ylabel('\bfAngular Position \theta (rad)');
```

```
    title(['\bfSteady State Response (t,\theta) for given h='
num2str(h(i))]);
    % Plot phase space for given h
    % Use mod for the angular position (-pi<=theta<=+pi)
    subplot(2,1,2); p=plot(pi-mod(Y(:,1),2*pi),Y(:,2),[col{i} '.']);
    set(p,'MarkerSize',2);
    grid on; axis tight;
    xlabel('\bfAngular Position \theta (rad)');
    ylabel('\bfAngular Velocity d\theta/dt(rad/sec)');
    title(['\bfPhase Space (\theta,d\theta/dt) for given h='
num2str(h(i))]);
end
end

function dx=Npp(t,x,gamma,omega0,h)
% Describe nonlinear parametric pendulum motion
dx(1,1)=x(2);
dx(2,1)=-2*gamma*x(2)-omega0^2*(1+h*cos(2*omega0*t))*sin(x(1));
end
```





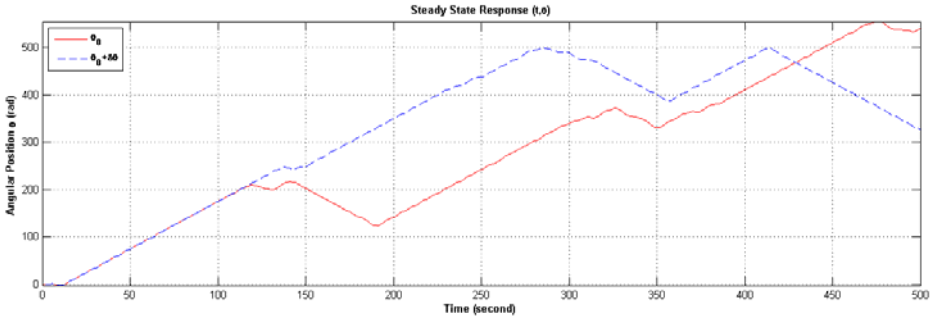
iv)

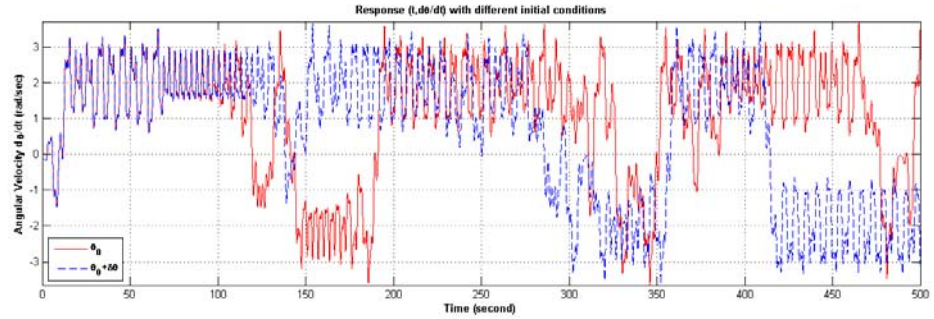
First figure contains the time responses of angular position and angular velocity of the pendulum, and second one represents phase space of the pendulum's response. (They are generated with following m-codes.) If the system is deterministic, the small change of initial conditions makes the system response change a little. However, the system responses are very different even with very similar initial conditions if system is chaotic. In the case of our nonlinear parametric pendulum case, the system is either deterministic or chaotic, depending on its some parameters. In this problem, the amplitude of forcing decides characteristics of pendulum's response. Therefore, the pendulum with $h = 2.20$ is chaotic and it shows sensitive dependence on initial conditions, based on figures below.

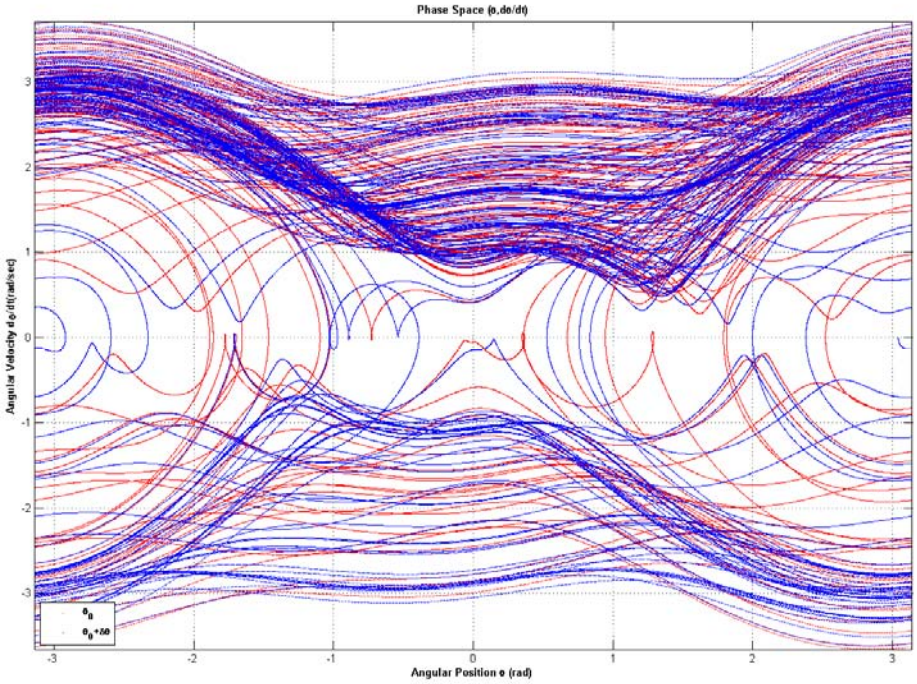
```
function P81_iv
% Problem 8.1 : Nonlinear parametric pendulum
% Part iv. Sensitive dependence on initial conditions
```

```
% Define some constants
omega0=1;      % Natural frequency of the pendulum
gamma=0.1;     % Damping coefficient
h=2.20;        % Amplitude of forcing

% Define some parameters for using Runge-Kutta metod
tspan=[0 500];     % Time span to be obtained
theta0=[1e-1,0];    % Initial conditions
dtheta0=[0,1e-8];   % Change of initial conditions

% Choose time step smaller than 0.01 sec
optn=odeset('InitialStep',1e-2,'MaxStep',1e-2);

% Simulation with given initial condition
[T,Y]=ode45(@(t,x) Npp(t,x,gamma,omega0,h),tspan,theta0,optn);
% Simulation with given initial condition + delta
[Td,Yd]=ode45(@(t,x) Npp(t,x,gamma,omega0,h),tspan,theta0+dtheta0,optn);
% Generate figure
figure(1);
% Plot steady state response for given h
subplot(2,1,1); plot(T,Y(:,1),'r-',Td,Yd(:,1),'b--');
grid on; axis tight;
xlabel('\bfTime (second)');
ylabel('\bfAngular Position \theta (rad)');
title('\bfSteady State Response (t,\theta)');
legend('\bf\theta_0','\bf\theta_0+\delta\theta', ...
   'Location','NorthWest');
subplot(2,1,2); plot(T,Y(:,2),'r-',Td,Yd(:,2),'b--');
grid on; axis tight;
xlabel('\bfTime (second)');
ylabel('\bfAngular Velocity d\theta/dt (rad/sec)');
title('\bfResponse (t,d\theta/dt) with different initial conditions');
legend('\bf\theta_0','\bf\theta_0+\delta\theta', ...
   'Location','SouthWest');
```

```
% Plot phase space
% Use mod for the angular position (-pi<=theta<=+pi)
figure(2);
p=plot(pi-mod(Y(:,1),2*pi),Y(:,2),'ro', ...
   pi-mod(Yd(:,1),2*pi),Yd(:,2),'bs');
set(p,'MarkerSize',1);
grid on; axis tight;
xlabel('\bfAngular Position \theta (rad)');
ylabel('\bfAngular Velocity d\theta/dt(rad/sec)');
title('\bfPhase Space (\theta,d\theta/dt)');
legend('\bf\theta_0','\bf\theta_0+\delta\theta', ...
   'Location','SouthWest');
end

function dx=Npp(t,x,gamma,omega0,h)
% Describe nonlinear parametric pendulum motion
dx(1,1)=x(2);
dx(2,1)=-2*gamma*x(2)-omega0^2*(1+h*cos(2*omega0*t))*sin(x(1));
end
```





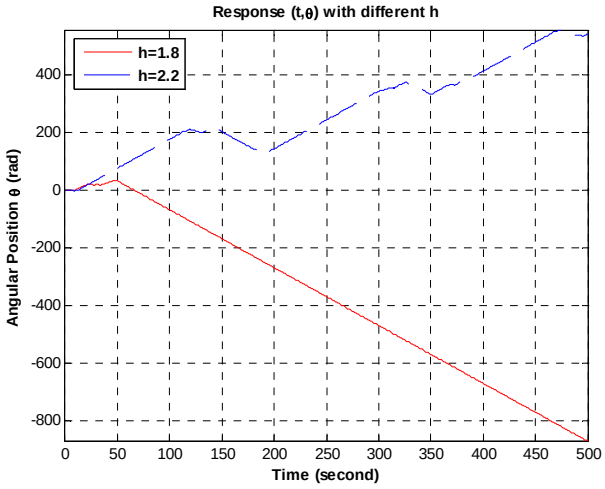


v)

With the below m-codes, the comparisons with different h are shown. As you see plots, they seem to have similar response at the beginning. However, the responses are quite different as time goes on. For $h = 1.80$, its response becomes steady and predictable. (the pendulum seems to rotate clockwise constantly) However, the response with $h = 2.20$ is neither stabilized nor predictable. The pendulum rotates counterclockwise constantly, and it suddenly changes the rotating direction. Therefore, $h = 2.20$ makes pendulum's response chaotic.

```
function P81_v
% Problem 8.1 : Nonlinear parametric pendulum
% Part v. Sensitive dependence on initial conditions

% Define some constants
omega0=1;       % Natural frequency of the pendulum
gamma=0.1;      % Damping coefficient
```

```
h=[1.80 2.20];        % Two amplitudes of forcing

% Define some parameters for using Runge-Kutta metod
tspan=[0 500];     % Time span to be obtained
theta0=[1e-1,0];    % Initial conditions

% Choose time step smaller than 0.01 sec
optn=odeset('InitialStep',1e-2,'MaxStep',1e-2);

% Simulation with h=1.8
[T1,Y1]=ode45(@(t,x) Npp(t,x,gamma,omega0,h(1)),tspan,theta0,optn);
% Simulation with h=2.2
[T2,Y2]=ode45(@(t,x) Npp(t,x,gamma,omega0,h(2)),tspan,theta0,optn);
% Generate figure
figure(1);
% Plot steady state response for given h
plot(T1,Y1(:,1),'r-',T2,Y2(:,1),'b--');
grid on; axis tight;
xlabel('\bfTime (second)');
ylabel('\bfAngular Position \theta (rad)');
title('\bfResponse (t,\theta) with different h');
legend(['\bfh=' num2str(h(1))],['\bfh=' num2str(h(2))], ...
    'Location','NorthWest');
end

function dx=Npp(t,x,gamma,omega0,h)
% Describe nonlinear parametric pendulum motion
dx(1,1)=x(2);
dx(2,1)=-2*gamma*x(2)-omega0^2*(1+h*cos(2*omega0*t))*sin(x(1));
end
```



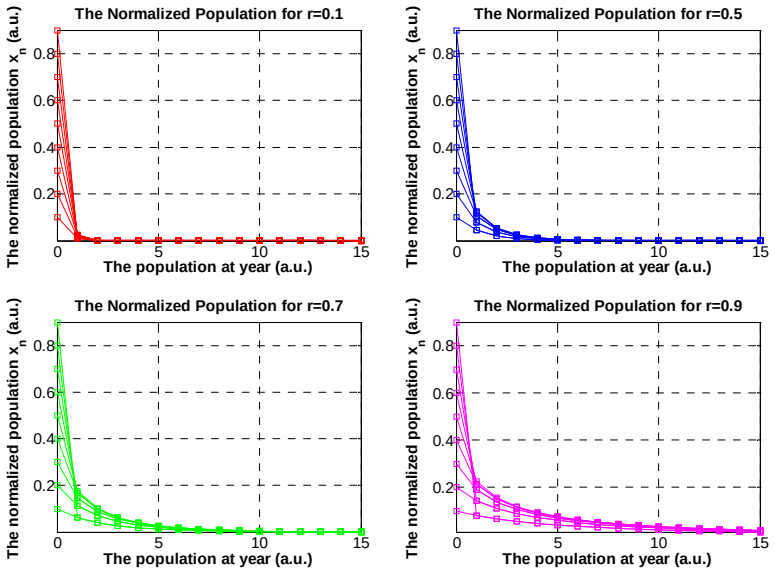
Problem 8.2 : The growth/decay of population of animal species

i)

The following figures with m-codes described as below represent the normalized population with respect to year n with different initial populations. Each subplot has different r (0.1, 0.5, 0.7, and 0.9). As seen in the plots, for arbitrary r between 0 and 1, all the populations approach to zero as time goes on regardless of their initial conditions.

```
function P82_i
% Problem 8.2 : The growth/decay of population of animal species
% i. Show x_n converges to zero when 0<r<1.

% Choose arbitrary r between 0 and 1
r=[0.1 0.5 0.7 0.9];
% Set initial population matrix
x0=[0.1:0.1:0.9]';
% Set the population year
nf=15;
% Define color setr
col={'r' ; 'b' ; 'g' ; 'm'};

figure(1);
for j=1:length(r)
```

```
    % Set the normalized population matrix
    % row: different initial populatipon
    % column: population year n
    xn=zeros(length(x0),nf);
    % Calculate the quadratic recurrence equation
    x=x0;
    for i=1:nf
        xn(:,i)=r(j)*x.*(1-x);
        x=xn(:,i);
    end
    % Plot the normalized population for n
    subplot(2,2,j);
    h=plot(0:nf,[x0 xn],[col{j} 's-']);
    set(h,'MarkerSize',6);
    set(gca,'FontSize',18);
    grid on; axis tight;
    xlabel('\bfThe population at year (a.u.)');
    ylabel('\bfThe normalized population x_n (a.u.)');
    title(['\bfThe Normalized Population for r=' num2str(r(j))]);
end
end
```



ii)

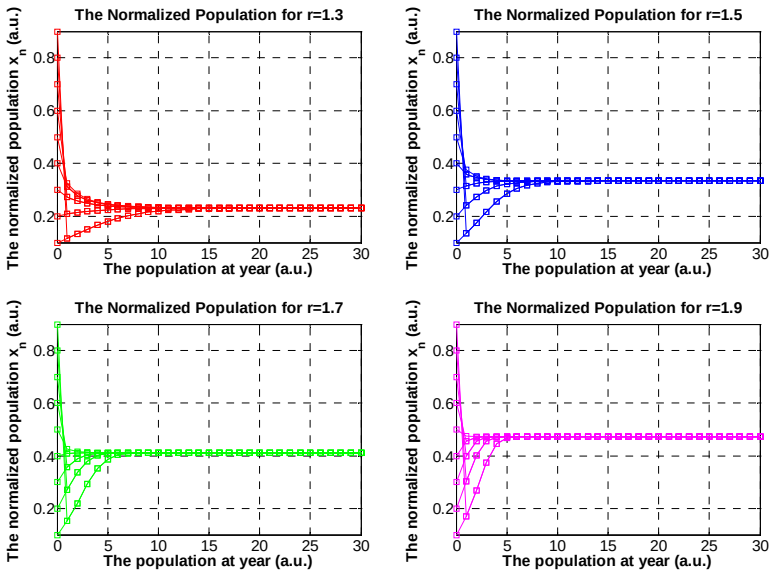
In the same way as i), each subplot has different r (1.3, 1.5, 1.7, and 1.9). with these r's, below plots show the normalized population as time goes on. Unlike i), the normalized population don't approach to zero any more. Following output lists the final values which the populations approach to.

```
1. r=1.300000
the population approaches to 0.230754
2. r=1.500000
the population approaches to 0.333333
3. r=1.700000
the population approaches to 0.411765
4. r=1.900000
the population approaches to 0.473684
```

With given r, each final values are same to $\frac{r-1}{r}$. Therefore, the final values for r between 1 and 2 converge to $\frac{r-1}{r}$.

```
function P82_ii
% Problem 8.2 : The growth/decay of population of animal species
% ii. Show x_n converges to (r-1)/r when 1<r<2.

% Choose arbitrary r between 1 and 2
r=[1.3 1.5 1.7 1.9];
% Set initial population matrix
x0=[0.1:0.1:0.9]';
% Set the population year
nf=30;
% Define color setr
col={'r' ; 'b' ; 'g' ; 'm'};
```

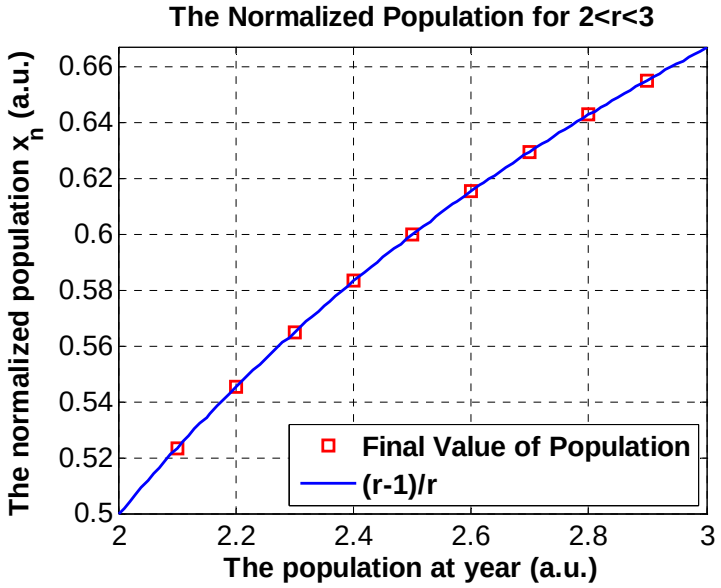
```
figure(1);
for j=1:length(r)
   % Set the normalized population matrix
   % row: different initial populatipon
   % column: population year n
   xn=zeros(length(x0),nf);
   % Calculate the quadratic recurrence equation
   x=x0;
   for i=1:nf
      xn(:,i)=r(j)*x.*(1-x);
      x=xn(:,i);
   end
   % Plot the normalized population for n
   subplot(2,2,j);
   figure(1);
   h=plot(0:nf,[x0 xn],[col{j} 's-']);
   set(h,'MarkerSize',6);
   set(gca,'FontSize',18);
   grid on; axis tight;
   xlabel('\bfThe population at year (a.u.)');
   ylabel('\bfThe normalized population x_n (a.u.)');
   title(['\bfThe Normalized Population for r=' num2str(r(j))]);
   % Display final values which the population approaches to
   fprintf('%d. r=%f\n',j,r(j));
   fprintf('the population approaches to %f\n',mean(xn(:,end)));
end
end
```



iii) What happens for values of r between 2 and 3 ?

```
function P82_iii
% Problem 8.2 : The growth/decay of population of animal species
% iii. What happens to x_n  when 2<r<3.

% make series of r between 2 and 3
r=2.10:0.10:2.90;
% Set initial population matrix
x0=[0.1:0.1:0.9]';
% Set the population year
nf=100;
for j=1:length(r)
    % Set the normalized population matrix
    % row: different initial populatipon
    % column: population year n
    xn=zeros(length(x0),nf);
    % Calculate the quadratic recurrence equation
    x=x0;
    for i=1:nf
        xn(:,i)=r(j)*x.*(1-x);
```

```
        x=xn(:,i);
    end
    xf(j)=mean(xn(:,end));
end
% Plot the normalized population for n
figure(1);

% Generate series
r1=[2:0.01:3];
xf1=(r1-1)./r1;
%
h=plot(r,xf,'rs',r1,xf1,'b-');
set(h,'MarkerSize',6,'LineWidth',2);
set(gca,'Fontsize',15);
legend('\bfFinal Value of Population','\bf(r-1)/r', ...
    'Location','SouthEast');
grid on; axis tight;
    xlabel('\bfThe population at year (a.u.)');
    ylabel('\bfThe normalized population x_n (a.u.)');
    title(['\bfThe Normalized Population for r=' num2str(r(j))]);
end
```



iv)

Below m-codes generate four plots whenever period-doubling happens as r increases from 2.90 to 3.60. As seen in the figure, each normalized population converges to an oscillation between 2^k different values. Another figure shows how many values exists in single oscillation with respect to r. In addition to figures, the following output for m-code below is shown.

```
at r=2.900000, number of population =1
at r=3.000000, number of population =2
at r=3.450000, number of population =4
at r=3.545000, number of population =8
at r=3.565000, number of population =16
at r=3.569000, number of population =28
at r=3.570000, number of population =69
at r=3.571000, number of population =129
at r=3.572000, number of population =150
at r=3.573000, number of population =174
……
……
```

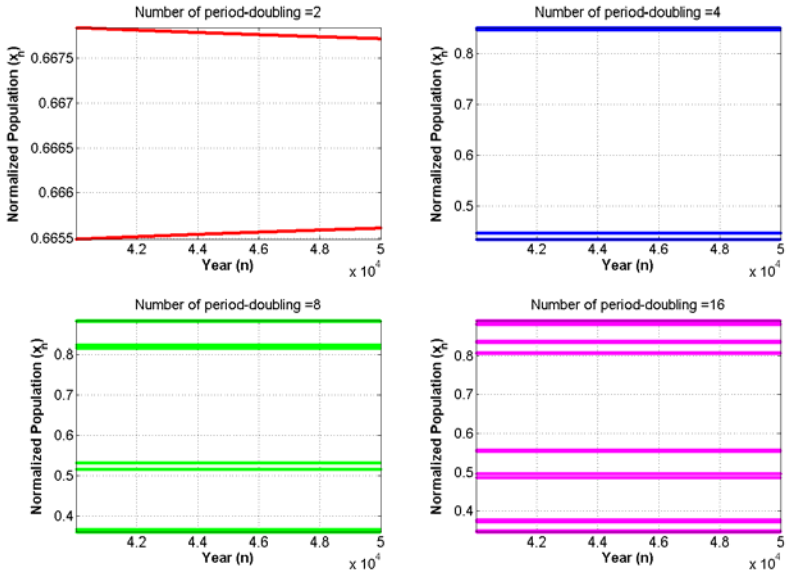
The number of different final populations is displayed doubled as r increases (2-period oscillation at r=3.000, 4-period oscillation at r=3.450, 8-period oscillation at r=3.455, and 16-period oscillation at r=3.565). When r is close to 3.57, period-doubling happens any more since the number of different final populations increases randomly (look like chaotic), and no finite-period oscillation doesn't seem to exist. Therefore, 3.000, 3.450, 3.455, and 3.565 are the first four values of r where period doubling occurs.

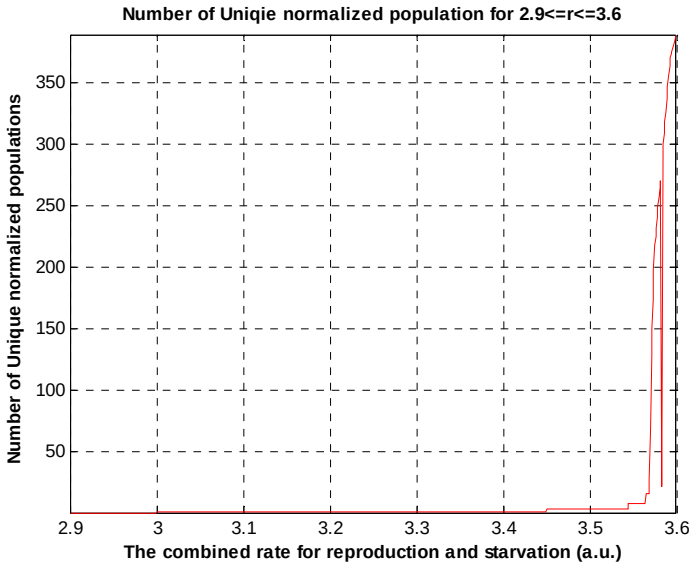
```
function P82_iv
% Problem 8.2 : The growth/decay of population of animal species
% iv. period-doubling
```

```
% make series of r between 2.90 and 3.60
% with step size of 0.001
r=2.90:0.001:3.60;
% Set initial population
x0=0.5;
% Define number of different final populations
n=zeros(1,length(r));
% Counter for number of period doubling happening
j=1;
% Define color setr
col={'r' ; 'b' ; 'g' ; 'm'};
% define figure
figure(1);
% Simulation with different r
for i=1:length(r)
   % get 10000 values of final population
   % 1. Simulation up to 40000 secs
   % 2. Collect populations for next 10000 secs
   %   as final population.
   xn=LogiEqn(r(i),x0);

   % Discard numbers below 0.001
   xf=fix(xn*1000);
   % Extract unique populations
   xf=unique(xf);
   % Count number of unique populations
   n(i)=length(xf);
   if i>1
     % if period-doubling happens, display r
     if n(i)~=n(i-1)
         fprintf('at r=%f, number of population =%d\n', ...
            r(i),n(i));
         % if the first 4 period-dobling, plot n vs. xn
         if j<=4
            subplot(2,2,j);
```

```
            plot([40001:50000],xn,[col{j} '.']);
            axis tight; grid on;
            set(gca,'FontSize',16);
            xlabel('\bfYear (n)');
            ylabel('\bfNormalized Population (x_n)');
            title(['Number of period-doubling =' num2str(n(i))]);
            j=j+1;
         end
      end
   % For starting point,
   else
      fprintf('at r=%f, number of population =%d\n', ...
         r(i),n(i));
   end
end
% Plot r vs. number of unique populations
figure(2);
plot(r,n,'r-');
grid on; axis tight;
xlabel('\bfThe combined rate for reproduction and starvation (a.u.)');
ylabel('\bfNumber of Unique normalized populations');
title('\bfNumber of Uniqie normalized population for 2.9<=r<=3.6');
end

function y=LogiEqn(r,x)
% Calculate quadratic reccureance relations
% x_n+1=r*x_n

% Calculate population for first 40000 years
for i=1:40000
   xn=r*x.*(1-x);
   x=xn;
end
% Define variable to store population for next 10000 years
y=zeros(length(x),10000);
```

```
% Calculate population for next 10000 years
for i=1:10000
    xn=r*x.*(1-x);
    x=xn;
    y(:,i)=xn;
end
end
```



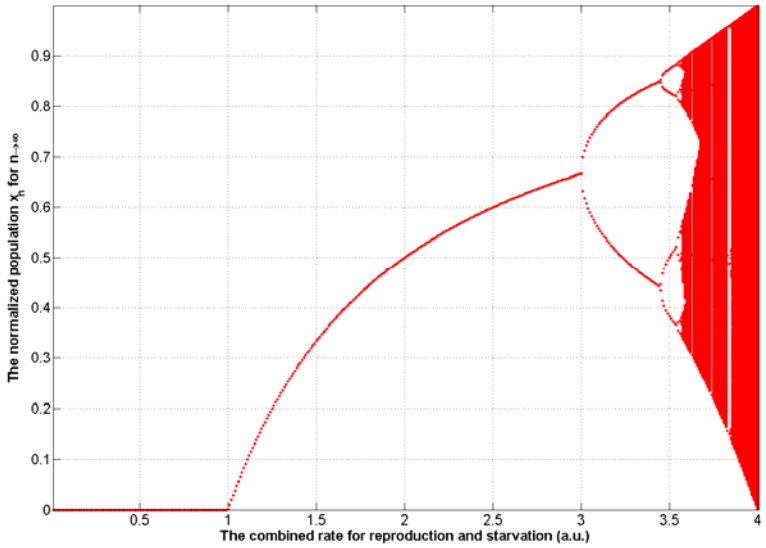


v)

Following m-codes are for creating a rough map, and the figure represent a rough map of the asymptotic values of x_n for varying r.

```
function P82_v
% Problem 8.2 : The growth/decay of population of animal species
% v.Plot r vs. x_infinity for logistic equation


% define the series of r from 0.01 to 4 with time step of 0.01
r=0.01:0.01:4;
% Calculate x_n when n approach to the infinity.
for i=1:length(r)
   % Obtain x_n for different very large n
   y=LogiEqn(r(i),0.1);
   % Prepare the plot
   x=r(i)*ones(length(y),1);
   % Generate the plot x_infinity distribution for the given r
   plot(x,y,'r.'); hold on;
end
hold off;
axis tight; grid on;
set(gca,'FontSize',16);
xlabel('\bfThe combined rate for reproduction and starvation (a.u.)');
ylabel('\bfThe normalized population x_n for n\rightarrow\infty');
end

function y=LogiEqn(r,x0)
%
% Calculate quadratic reccureance relations
% x_n+1=r*x_n
%

for i=1:40000
   xn=r*x0.*(1-x0);
```

```
    x0=xn;
end
y=zeros(length(x0),10000);
for i=1:10000
    xn=r*x0.*(1-x0);
    x0=xn;
    y(:,i)=xn;
end
end
```



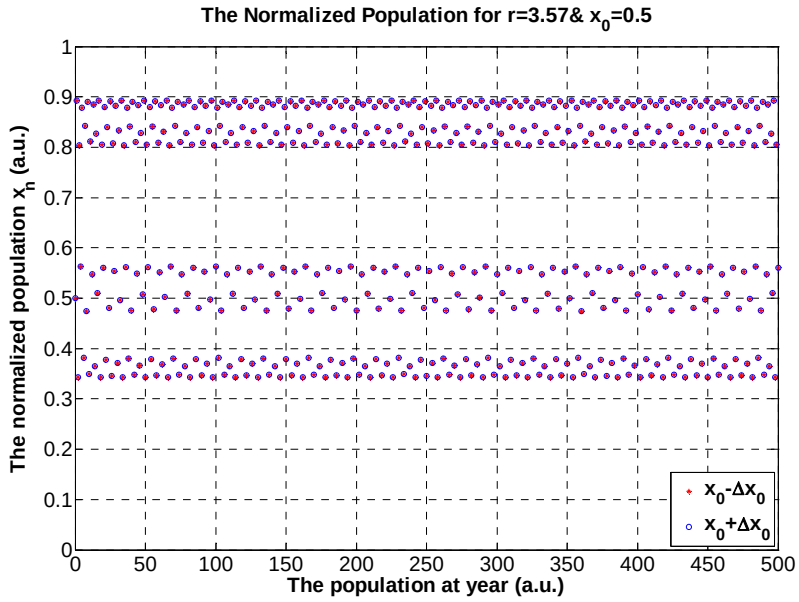
vi)

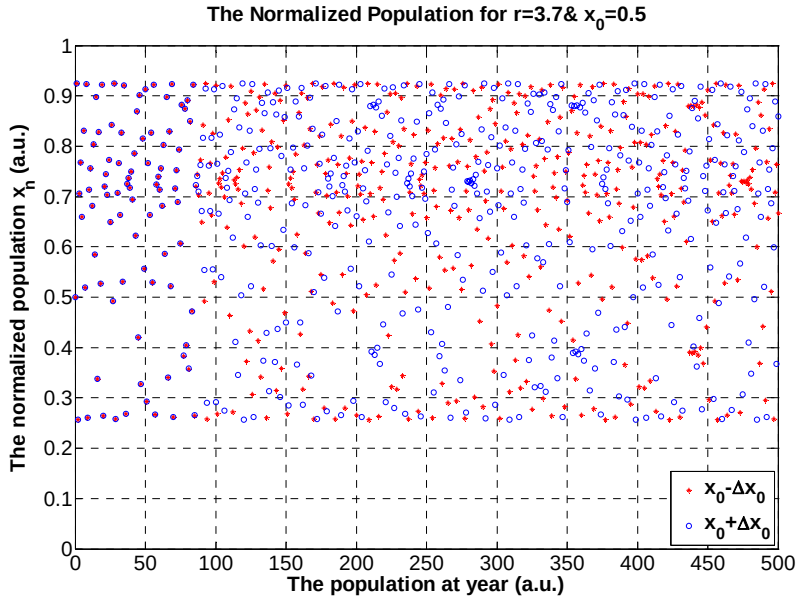
Even if there is no oscillation, the sensitive dependence on initial conditions doesn't happen when initial populations $x_0 \pm \Delta x_0 = 0.5 \pm 10^{-5}$ are chosen at $r = 3.57$. However, the output of population for very very small amount of difference between two initial populations ($x_0 \pm \Delta x_0 = 0.5 \pm 10^{-5}$) at $r = 3.7$ is quite different, and it confirms that there is sensitive dependence on initial conditions. The m-codes and graphs for those outputs are shown as below.

```
function P82_vi
% Problem 8.2 : The growth/decay of population of animal species
```

```
% vi. sensitive dependence on initial conditions

% Set two r's given in problem
r=[3.57 3.7];
% Set initial population
x0=0.5;
% set small difference of initial population
dx0=0.00001;
for i=1:length(r)
   % calcuate population with x0-dx0
   xn1=LogiEqn(r(i),x0-dx0);
   % calcuate population with x0-dx0
   xn2=LogiEqn(r(i),x0+dx0);
   % plot and compare the results with above initial populations
   figure(i);
   plot(0:length(xn1),[x0;xn1],'r*',0:length(xn2),[x0;xn2],'bo');
   d=axis(); d(3:4)=[0 1]; axis(d); grid on;
   set(gca,'FontSize',24);
   xlabel('\bfThe population at year (a.u.)');
   ylabel('\bfThe normalized population x_n (a.u.)');
   title(['\bfThe Normalized Population for r=' num2str(r(i)) '& x_0='
num2str(x0)]);
   legend('\bfx_0-Deltax_0','\bfx_0+\Deltax_0','Location','SouthEast');
end
end

function xn=LogiEqn(r,x0)
%
% Calculate quadratic reccureance relations
% x_n+1=r*x_n
%
nf=500;
xn=zeros(nf,1);
x=x0;
for i=1:nf
```

```
    xn(i)=r*x.*(1-x);
    x=xn(i);
end
end
```





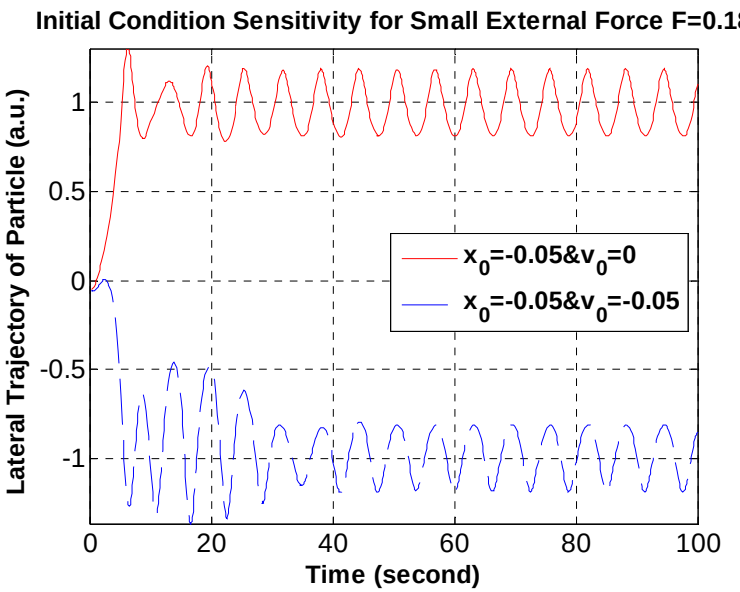
Problem 8.3 : Double-Well Potential System

i)

Two different initial conditions are used to observe their response. All the m-codes and figures are shown as below. First of all, there responses are quite different, depending on

initial conditions. Second, their steady state responses are oscillatory. Finally, even though particles with different initial conditions stay in one of two wells, their periods are didn't change.

```
function P83_i
% Problem 8.3 : Double-Well Potential System
% i. Initial Condition Sensitivity for Small External Force

% Define some parameters
x01=[-0.05 0];         % First initial condition
x02=[-0.05 -0.05];      % Second initial condition
F=0.18;               % Amplitude of external force
% Solve equation of motion with the first initial conditions
[T1,X1]=ode45(@(t,x)DoubleWell(t,x,F),[0,100],x01);
% Solve equation of motion with the second initial conditions
[T2,X2]=ode45(@(t,x)DoubleWell(t,x,F),[0,100],x02);
% Plot and compare above two time response
plot(T1,X1(:,1),'r-',T2,X2(:,1),'b--');
grid on; axis tight;
set(gca,'FontSize',14);
xlabel('\bfTime (second)');
ylabel('\bfLateral Trajectory of Particle (a.u.)');
title('\bfInitial Condition Sensitivity for Small External Force
F=0.18');
legend('\bfx_0=-0.05&v_0=0','\bfx_0=-0.05&v_0=-0.05');
end

function dx=DoubleWell(t,x,F)
% Describe euation of motion
% in the double-well potetial system
dx(1,1)=x(2);
dx(2,1)=F*cos(t)-0.25*x(2)+x(1)-x(1)^3;
end
```

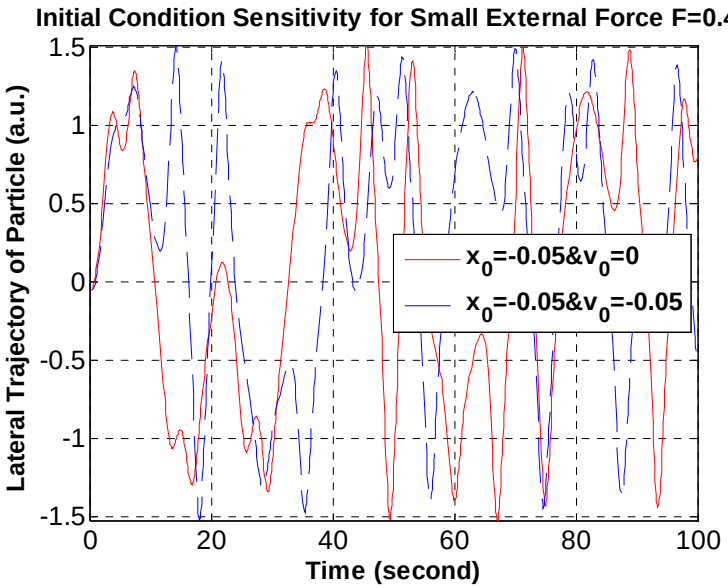


ii)

In the same way as i), we compare particle trajectories for different initial conditions. Their responses are quite different finally, even though their transient responses are a little similar. They are not periodic any more, they are irrelevant, and seem to be chaotic.

```
function P83_ii
% Problem 8.3 : Double-Well Potential System
% i. Initial Condition Sensitivity for Large External Force

% Define some parameters
x01=[-0.05 0];         % First initial condition
x02=[-0.05 -0.05];      % Second initial condition
F=0.40;               % Amplitude of external force
% Solve equation of motion with the first initial conditions
[T1,X1]=ode45(@(t,x)DoubleWell(t,x,F),[0,100],x01);
% Solve equation of motion with the second initial conditions
[T2,X2]=ode45(@(t,x)DoubleWell(t,x,F),[0,100],x02);
% Plot and compare above two time response
plot(T1,X1(:,1),'r-',T2,X2(:,1),'b--');
grid on; axis tight;
set(gca,'FontSize',14);
```

```
xlabel('\bfTime (second)');
ylabel('\bfLateral Trajectory of Particle (a.u.)');
title('\bfInitial Condition Sensitivity for Small External Force
F=0.40');
legend('\bfx_0=-0.05&v_0=0','\bfx_0=-0.05&v_0=-0.05');
end

function dx=DoubleWell(t,x,F)
% Describe euation of motion
% in the double-well potetial system
dx(1,1)=x(2);
dx(2,1)=F*cos(t)-0.25*x(2)+x(1)-x(1)^3;
end
```

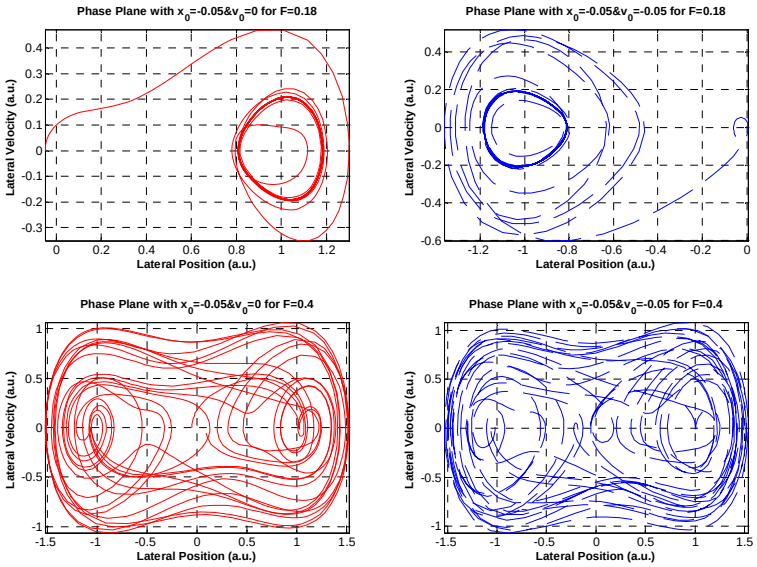


iii)

Time responses obtained in i) and ii) are converted into position and velocity of particle trajectories. Phase planes for different F and initial conditions are shown, which are generated by following m-codes. For $F = 0.18$, they shows periodicity in steady state, but phase planes with $F = 0.40$ don't seems to have periodicity, even if the responses are independent of time. (See below Figures)

```
function P83_iii
% Problem 8.3 : Double-Well Potential System
% iii. Phase Plane for different initial conditions
%     and the amplitude of external force

% Define some parameters
x01=[-0.05 0];         % First initial condition
x02=[-0.05 -0.05];      % Second initial condition
F=[0.18 0.40];         % Amplitude of external force
% Create figure
figure(1);
for i=1:length(F)
   % Solve equation of motion with the first initial conditions
   [T1,X1]=ode45(@(t,x)DoubleWell(t,x,F(i)),[0,200],x01);
   % Plot phase plane for the first initial conditions
   subplot(2,2,i*2-1); plot(X1(:,1),X1(:,2),'r-');
   grid on; axis tight;
   set(gca,'FontSize',14);
   xlabel('\bfLateral Position (a.u.)');
   ylabel('\bfLateral Velocity (a.u.)');
   title(['\bfPhase Plane with x_0=-0.05&v_0=0 for F=' num2str(F(i))]);
   % Solve equation of motion with the second initial conditions
   [T2,X2]=ode45(@(t,x)DoubleWell(t,x,F(i)),[0,200],x02);
   % Plot phase plane for the second initial conditions
   subplot(2,2,i*2); plot(X2(:,1),X2(:,2),'b--');
   grid on; axis tight;
   set(gca,'FontSize',14);
   xlabel('\bfLateral Position (a.u.)');
   ylabel('\bfLateral Velocity (a.u.)');
   title(['\bfPhase Plane with x_0=-0.05&v_0=-0.05 for F='
num2str(F(i))]);
end
end

function dx=DoubleWell(t,x,F)
```

```
% Describe euation of motion
% in the double-well potetial system
dx(1,1)=x(2);
dx(2,1)=F*cos(t)-0.25*x(2)+x(1)-x(1)^3;
end
```

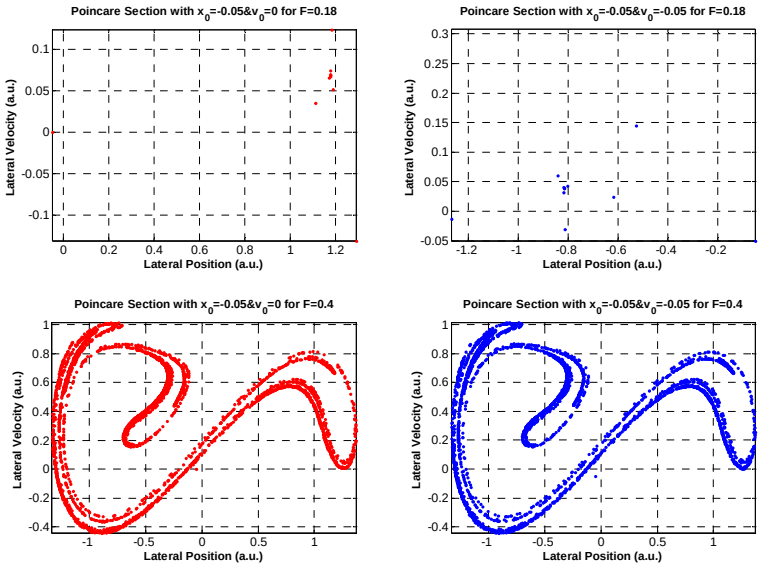


iv)

Sometimes, Poincare section is quite useful in nonlinear dynamics analysis. We choose phase which is identical to the multiple of 2π in the phase of oscillating external force. For small external force ($F = 0.18$), Poincare section have several points for both initial conditions, but the position of points in Poincare section are completely different, depending on initial conditions. Therefore, the phases of responses are synchronized with those of external forces, and they are also periodic, but the position and velocity of particles during the oscillation depend on initial conditions. For large external force($F = 0.40$), Their shapes for different initial conditions in Poincare section are almost identical, independent of initial condition, even though they didn't show any similarities in both time response and phase plane analysis. It means that the position and velocity pairs of the particle at the multiple of period for external force are same, independent of their pair's sequence. Therefore, it confirms that time responses in ii) are not chaotic, and they are relevant each other.

```
function P83_iv
% Problem 8.3 : Double-Well Potential System
% iv. Poincare section for different initial conditions
%    and the amplitude of external force

% Define some parameters
x01=[-0.05 0];         % First initial condition
x02=[-0.05 -0.05];      % Second initial condition
F=[0.18 0.40];         % Amplitude of external force
% Create figure
figure(1);
for i=1:length(F)
   % Solve equation of motion with the first initial conditions
   [T1,X1]=ode45(@(t,x)DoubleWell(t,x,F(i)),[0:2*pi:20000],x01);
   % Plot phase plane for the first initial conditions
   subplot(2,2,i*2-1); plot(X1(:,1),X1(:,2),'r.');
   grid on; axis tight;
   set(gca,'FontSize',14);
   xlabel('\bfLateral Position (a.u.)');
   ylabel('\bfLateral Velocity (a.u.)');
   title(['\bfPoincare Section with x_0=-0.05&v_0=0 for F='
num2str(F(i))]);
   % Solve equation of motion with the second initial conditions
   [T2,X2]=ode45(@(t,x)DoubleWell(t,x,F(i)),[0:2*pi:20000],x02);
   % Plot phase plane for the second initial conditions
   subplot(2,2,i*2); plot(X2(:,1),X2(:,2),'b.');
   grid on; axis tight;
   set(gca,'FontSize',14);
   xlabel('\bfLateral Position (a.u.)');
   ylabel('\bfLateral Velocity (a.u.)');
   title(['\bfPoincare Section with x_0=-0.05&v_0=-0.05 for F='
num2str(F(i))]);
end
end
```

```
function dx=DoubleWell(t,x,F)
dx(1,1)=x(2);
dx(2,1)=F*cos(t)-0.25*x(2)+x(1)-x(1)^3;
end
```

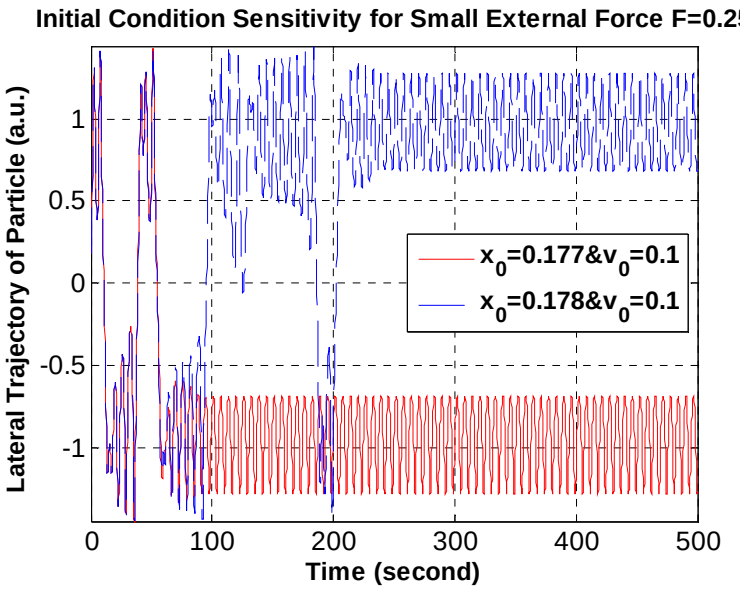


v)

Very small different initial condition (position difference is only 0.001, whereas velocity is same) give completely different response and final well where the particle stays at the end. At the beginning, their transient responses are almost same since they have very similar initial condition, but one of their response ($x_0 = 0.178$) goes to another well suddenly due to some nonlinearity. (See following figure.) m-codes is also provided as below.

```
function P83_v
% Problem 8.3 : Double-Well Potential System
% v. Large Initial Condition Sensitivity

% Define some parameters
x01=[0.177 0.1];        % Initial condition for i)
x02=[0.178 0.1];     % Initial condition for ii)
```

```
F=0.25;                 % Amplitude of external force
% Solve equation of motion with initial conditions for i)
[T1,X1]=ode45(@(t,x)DoubleWell(t,x,F),[0,500],x01);
% Solve equation of motion with initial conditions for ii)
[T2,X2]=ode45(@(t,x)DoubleWell(t,x,F),[0,500],x02);
% Plot and compare above two time response
plot(T1,X1(:,1),'r-',T2,X2(:,1),'b--');
grid on; axis tight;
set(gca,'FontSize',14);
xlabel('\bfTime (second)');
ylabel('\bfLateral Trajectory of Particle (a.u.)');
title('\bfInitial Condition Sensitivity for Small External Force
F=0.25');
legend('\bfx_0=0.177&v_0=0.1','\bfx_0=0.178&v_0=0.1');
end

function dx=DoubleWell(t,x,F)
% Describe euation of motion
% in the double-well potetial system
dx(1,1)=x(2);
dx(2,1)=F*cos(t)-0.25*x(2)+x(1)-x(1)^3;
end
```

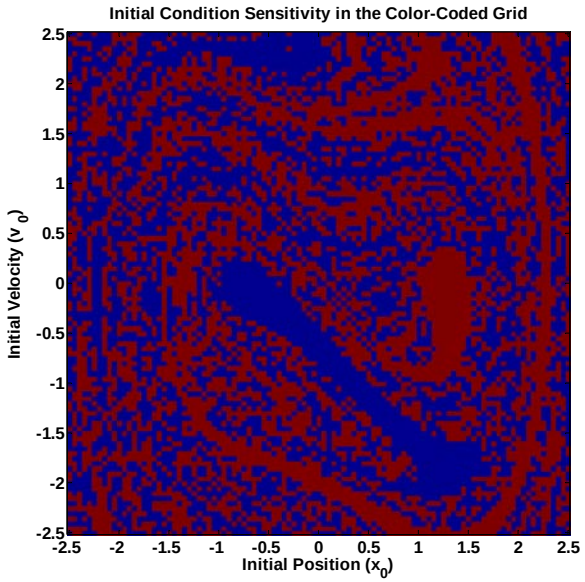


vi)

Following m-codes let you know how to generate color-coded grid, and display it as an image. See the below figure shows the result of initial condition sensitivity for $F = 0.25$. As expected, the results of which well where the particle stays eventually quite depend on the initial conditions in some range of initial condition.

```
function P83_vi
% Problem 8.3 : Double-Well Potential System
% vi. Plot the color-coded grids
%    for different initial conditions.

% Define number of steps
nx=50; ny=50;
% Define range of initial conditions
mx=2.5; my=2.5;
% Generate full range of initial conditions
x=[-nx:nx]/nx*mx;
y=[-ny:ny]/ny*my;
% Define Grid (101 X 101)
o=zeros(length(x),length(y));
% Calculate the final position o particle,
% depending on the initial conditions
for i=1:length(x)
   for j=1:length(y)
      % Set initial condition
      x0=[x(i);y(j)];
      % Calculate trajectory of particle
      [T,X]=ode45(@DoubleWell,[0,500],x0);
      % Determine which well the particle stays finally
      o(i,j)=sign(X(end,1));
   end
end
% Plot grid as an image with color code
% Swap row and column (Matrix transpose)
o=o';
```

```
imagesc(o);
axis image;
set(gca,'YDir','normal')
set(gca,'XTick',[1:10:101])
set(gca,'XTickLabel',[-2.5:0.5:2.5])
set(gca,'YTick',[1:10:101])
set(gca,'YTickLabel',[-2.5:0.5:2.5])
set(gca,'FontSize',20);
set(gca,'FontWeight','bold');
xlabel('\bfInitial Position (x_0)');
ylabel('\bfInitial Velocity (v_0)');
title('\bfInitial Condition Sensitivity in the Color-Coded Grid');
end

function dx=DoubleWell(t,x)
% Describe euation of motion
% in the double-well potetial system
dx(1,1)=x(2);
dx(2,1)=0.25*cos(t)-0.25*x(2)+x(1)-x(1)^3;
end
```



Note: Below figures shows initial condition sensitivity with same range of initial conditions as iv), but use different steps (201 X 201 for left one, and 901 X 901 for right one.)