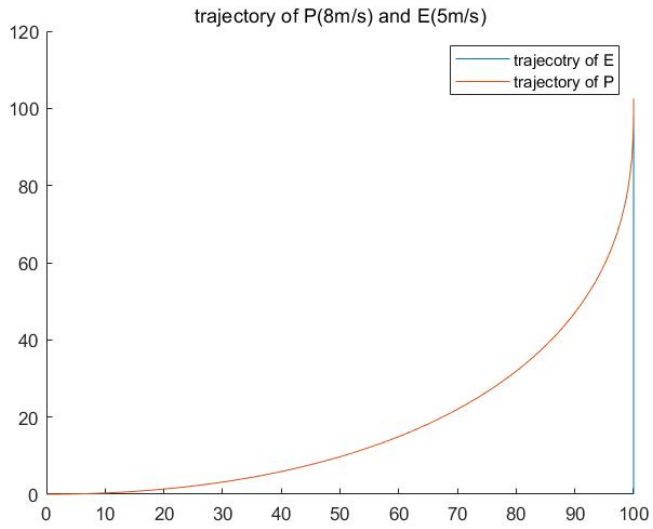
For 1 we use following codes.

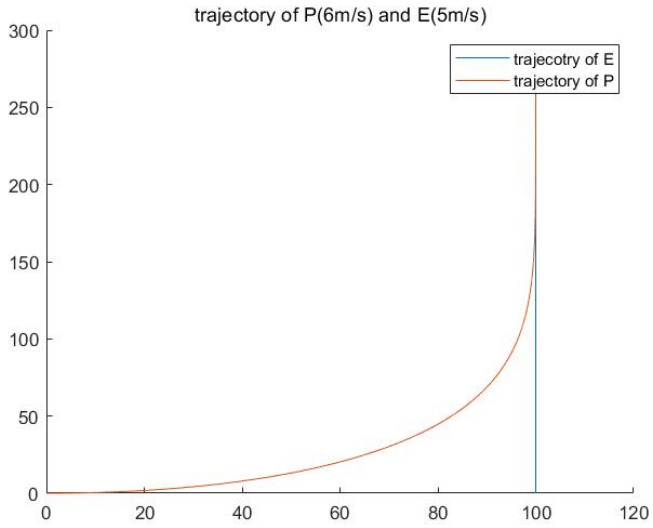
```
clc;
clear all;
VT = 5;
Vu = 6;
theta_E = pi/2;
time = 0:0.01:100;
finPos = 0;
opts = odeset('Events', @(t,y)StopEventPos(t,y,finPos), 'RelTol', 1e-6, 'AbsTol', 1e-6);
odefun = @(t,y) deriv(t,y,VT,Vu,theta_E);   % Anonymous derivative function with A
tspan = time;
f0 = [100;0];
[T,R_beta] = ode45(odefun,tspan,f0,opts);   % Pass in column vector initial value

len_tra = length(T);
traj_E = zeros(len_tra,2);
traj_E(:,1) = 100*ones(len_tra,1);
traj_E(:,2) = VT*T;

R =R_beta(:,1);
beta = R_beta(:,2);
x = traj_E(:,1)-R.*cos(beta);
y = traj_E(:,2)-R.*sin(beta);

figure(1)
hold on
plot(traj_E(:,1),traj_E(:,2));
plot(x,y);
title("trajectory of P(11m/s) and E(5m/s)")
legend("trajecotry of E","trajectory of P")

figure(2)
plot(T,R);
title("R versus time(P=6m/s)")

gama = Vu/VT;
d_beta = abs( (beta-pi/2).^(2-gama));
dd_beta = abs( (beta-pi/2).^(3-2*gama));
figure(3)
subplot(2,1,1)
plot(T(1:end-1),d_beta(1:end-1))
title("beta^* versus time(P=6m/s)")
```

```
subplot(2,1,2)
plot(T(1:end-1),dd_beta(1:end-1))
title("beta^{**} versus time(P=6m/s)")

function dy = deriv(t,y,VT,Vu,theta_E)
%direct DP
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
dR = VT*cos(beta-theta_E)-Vu;
d_beta = -VT*sin(beta-theta_E)/R;
dy = [dR;d_beta];
end

function [value, isterminal, direction] = StopEventPos(t, y, pos)
    value      = y(1) - pos;
    isterminal = 1;    % Stop the integration
    direction  = 0;
end
```

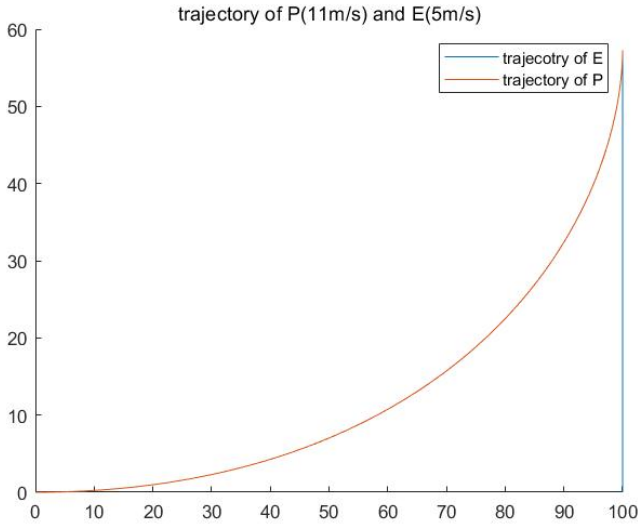
And the pictures look like below:



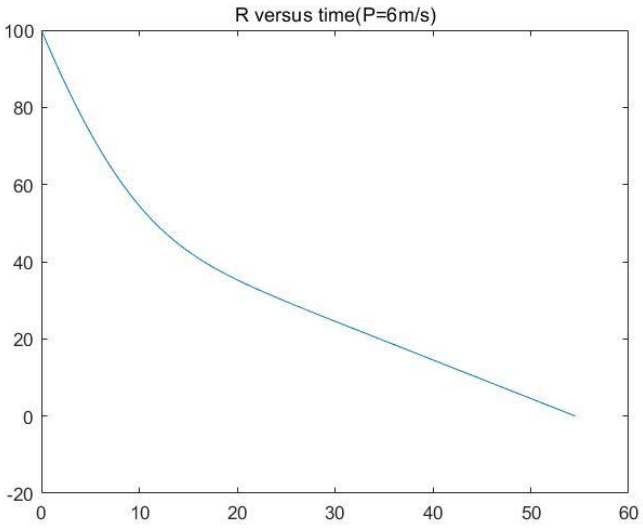
P1: trajectory of P(8m/s) and E(5m/s)



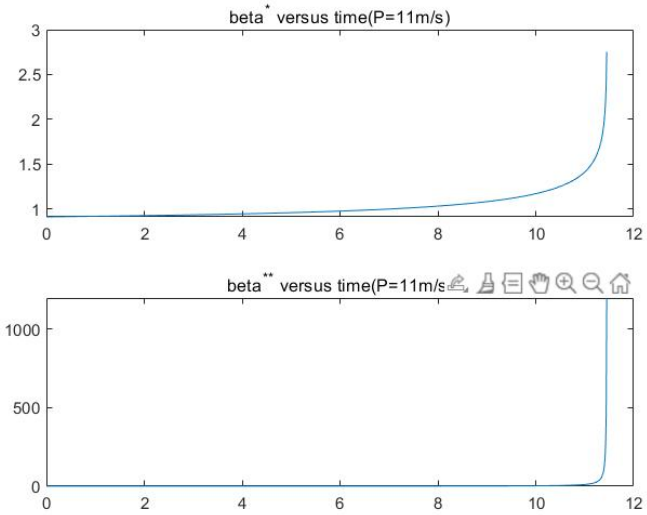
P2: trajectory of P(6m/s) and E(5m/s)



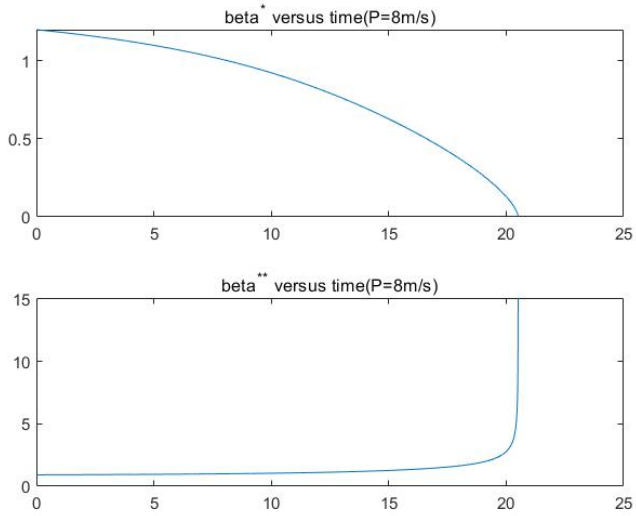
P3: trajectory of P(11m/s) and E(5m/s)



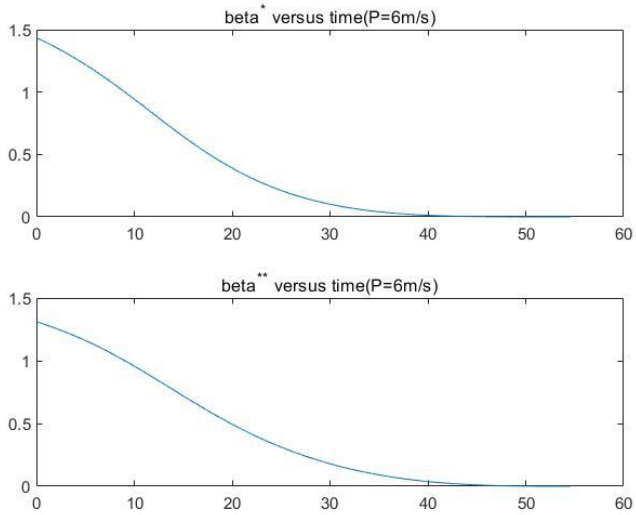
P4: R versus time(P=6m/s)



P5: $\dot{\beta}$ and $\ddot{\beta}$ versus time(P=11m/s)



P6: $\dot{\beta}$ and $\ddot{\beta}$ versus time(P=8m/s)



P7: $\dot{\beta}$ and $\ddot{\beta}$ versus time(P=6m/s)

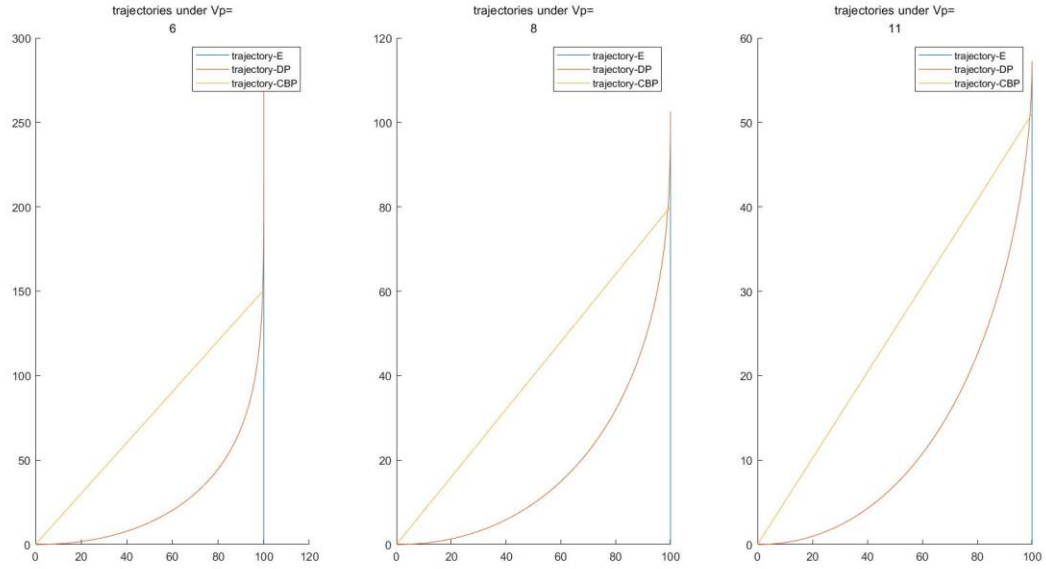
For 2 we use following codes.

```
clc;
clear all;
VT = 5;
Vui = [6,8,11];
Vu = 6;
theta = asin(VT/Vu);
theta_E = pi/2;
time = 0:0.01:100;
finPos = 0;
opts = odeset('Events', @(t,y)StopEventPos(t,y,finPos), 'RelTol', 1e-6, 'AbsTol', 1e-6);
tspan = time;

figure(1)
R_DP_all = {};
T_DP_all = {};
R_CBP_all = {};
T_CBP_all = {};
for i=1:3
Vu = Vui(i);
theta = asin(VT/Vu);
%%
%calculate DP
odefun = @(t,y) deriv(t,y,VT,Vu,theta_E);   % Anonymous derivative function with A
f0 = [100;0];
[T,R_beta] = ode45(odefun,tspan,f0,opts);

len_tra = length(T);
traj_E = zeros(len_tra,2);
traj_E(:,1) = 100*ones(len_tra,1);
traj_E(:,2) = VT*T;
R = R_beta(:,1);
beta = R_beta(:,2);
x_DP = traj_E(:,1)-R.*cos(beta);
y_DP = traj_E(:,2)-R.*sin(beta);
R_DP_all{i} = R;
T_DP_all{i} = T;
%%
%to calculate CBP
beta_CBP = 0;
ode_CBP = @(t,R) CBP(t,R,VT,Vu,theta_E,theta,beta_CBP);
R0 = 100;
[T_CBP,R_CBP] = ode45(ode_CBP,tspan,R0,opts);
```

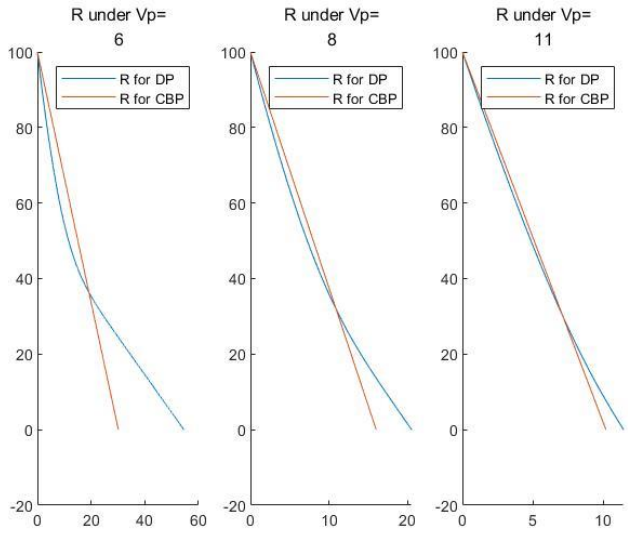
```
len_tra_CBP = length(T_CBP);
traj_E_CBP = zeros(len_tra_CBP,2);
traj_E_CBP(:,1) = 100*ones(len_tra_CBP,1);
traj_E_CBP(:,2) = VT*T_CBP;
x_CBP = traj_E_CBP(:,1)-R_CBP.*cos(beta_CBP);
y_CBP = traj_E_CBP(:,2)-R_CBP.*sin(beta_CBP);
R_CBP_all{i} = R_CBP;
T_CBP_all{i} = T_CBP;
%%
%plot
subplot(1,3,i)
hold on
plot(traj_E(:,1),traj_E(:,2));
plot(x_DP,y_DP);
plot(x_CBP,y_CBP);
legend("trajectory-E","trajectory-DP","trajectory-CBP")
title("trajectories under Vp=",int2str(Vu))
hold off
end

figure(2)
for i=1:3
    subplot(1,3,i)
    hold on
    plot(T_DP_all{i},R_DP_all{i});
    plot(T_CBP_all{i},R_CBP_all{i});
    legend("R for DP","R for CBP")
    Vu = Vui(i);
    title("R under Vp=",int2str(Vu))
    hold off
end

%%
%functions
function dy = deriv(t,y,VT,Vu,theta_E)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
dR = VT*cos(beta-theta_E)-Vu;
d_beta = -VT*sin(beta-theta_E)/R;
dy = [dR;d_beta];
end

function dR = CBP(t,R,VT,Vu,theta_E,theta,beta)
```

```
%y(1) = R, y(2) = beta
dR = VT*cos(beta-theta_E)-Vu*cos(beta-theta);
end

function [value, isterminal, direction] = StopEventPos(t, y, pos)
    value      = y(1) - pos;
    isterminal = 1;    % Stop the integration
    direction  = 0;
end
```

And the pictures look like below:



P8: trajectory of E,P with DP and CBP under Vp = [6,8,11]m/s



P9: R with DP and CBP under Vp = [6,8,11]m/s versus time

As Vp increases, the difference between the times of captured obtained for DP and CBP shrinks

For 3 we use following codes.

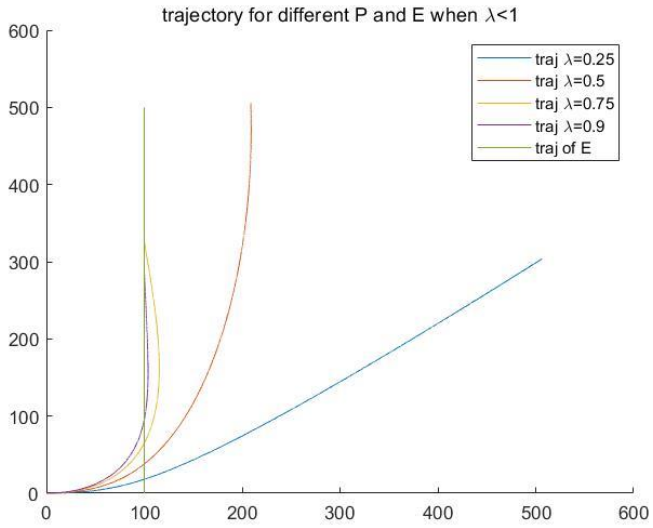
```
clc;
clear all;
c0 = 0;
VT = 5;
Vu = 6;
theta_E = pi/2;
time = 0:0.01:100;
tspan = time;
theta = asin(VT/Vu);
finPos = 0;
opts = odeset('Events', @(t,y)StopEventPos(t,y,finPos), 'RelTol', 1e-6, 'AbsTol', 1e-6);
lambdas_1 = [0.25, 0.5, 0.75, 0.9];
lambdas_2 = [1, 2, 5, 50];
lambdas =    [0.25, 0.5, 0.75, 0.9, 1, 2, 5, 50];

xpos_PP_all = {};
ypos_PP_all = {};
x_CBP_all = {};
y_CBP_all = {};
traj_E1 = zeros(1,1);
traj_E2 = zeros(1,1);
R_PP_all = {};
R_CBP_all = {};

for i=1:length(lambdas)
    lambda = lambdas(i);
    %%
    %to calculate PP
    %beta_PP = 0;
    ode_PP = @(t,y) PP(t,y,VT,Vu,theta_E,lambda);
    y0 = [100;0];
    [T_PP,y_PP] = ode45(ode_PP,tspan,y0,opts);
    R =y_PP(:,1);
    R_PP_all{i} = R;
    beta = y_PP(:,2);
    len_tra_PP = length(T_PP);
    traj_E_PP = zeros(len_tra_PP,2);
    traj_E_PP(:,1) = 100*ones(len_tra_PP,1);
    traj_E_PP(:,2) = VT*T_PP;
    xpos_PP = traj_E_PP(:,1)-R.*cos(beta);
    ypos_PP = traj_E_PP(:,2)-R.*sin(beta);
    xpos_PP_all{i}=xpos_PP;
```

```
    ypos_PP_all{i}=ypos_PP;

    if(i<length(lambdas_1))
        if(length(traj_E_PP(:,1))>length(traj_E1(:,1)))
            traj_E1 = traj_E_PP;
        end
    else
        if(length(traj_E_PP(:,1))>length(traj_E2(:,1)))
            traj_E2 = traj_E_PP;
        end
    end

    %%
    %to calculate CBP
    beta_CBP = 0;
    ode_CBP = @(t,R) CBP(t,R,VT,Vu,theta_E,theta,beta_CBP);
    R0 = 100;
    [T_CBP,R_CBP] = ode45(ode_CBP,tspan,R0,opts);
    R_CBP_all{i} = R_CBP;
    len_tra_CBP = length(T_CBP);
    traj_E_CBP = zeros(len_tra_CBP,2);
    traj_E_CBP(:,1) = 100*ones(len_tra_CBP,1);
    traj_E_CBP(:,2) = VT*T_CBP;
    x_CBP = traj_E_CBP(:,1)-R_CBP.*cos(beta_CBP);
    y_CBP = traj_E_CBP(:,2)-R_CBP.*sin(beta_CBP);
    x_CBP_all{i} = x_CBP;
    y_CBP_all{i}=y_CBP;
    if(i<length(lambdas_1))
        if(length(traj_E_CBP(:,1))>length(traj_E1(:,1)))
            traj_E1 = traj_E_CBP;
        end
    else
        if(length(traj_E_CBP(:,1))>length(traj_E2(:,1)))
            traj_E2 = traj_E_CBP;
        end
    end

end

%%
%PP and CBP <1
figure(1) %trajectory
hold on
for i=1:length(lambdas_1)
```

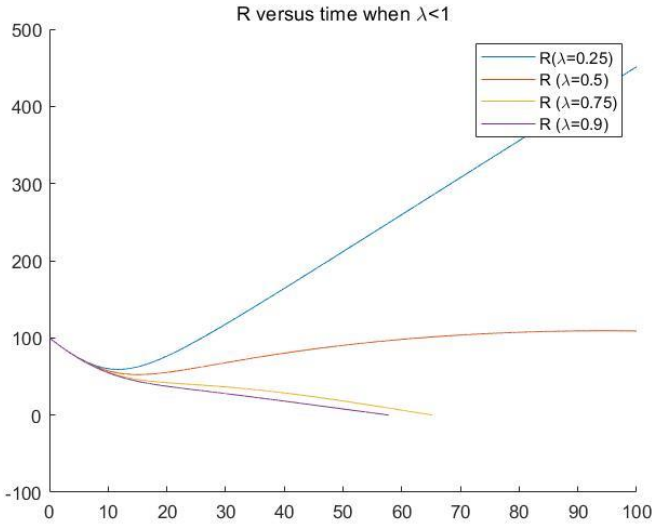
```
    plot(xpos_PP_all{i},ypos_PP_all{i})
end
plot(traj_E1(:,1),traj_E1(:,2))
hold off
legend('traj \lambda=0.25','traj \lambda=0.5','traj \lambda=0.75','traj \lambda=0.9','traj of E')
title('trajectory for different P and E when \lambda<1')

figure(2) %R
hold on
for i=1:length(lambdas_1)
    plot((0:1:(length(R_PP_all{i})-1))/100 , R_PP_all{i});
end
hold off
legend('R(\lambda=0.25)','R (\lambda=0.5)','R (\lambda=0.75)','R (\lambda=0.9)')
title('R versus time when \lambda<1')


%%
%PP and CBP >1
figure(3) %trajectory
hold on
len = length(lambdas_1);
for i=len+1:1:len+length(lambdas_2)
    plot(xpos_PP_all{i},ypos_PP_all{i});
end
plot(x_CBP_all{i},y_CBP_all{i},'--');
plot(traj_E2(:,1),traj_E2(:,2))

legend('PP traj \lambda=1','PP traj \lambda=2',...
    'PP traj \lambda=5','PP traj \lambda=50',...
   'CBP traj', 'traj of E');
title('trajectory for different P and E when \lambda>1')
hold off


figure(4) %R
hold on
for i=len+1:1:len+length(lambdas_2)
    plot((0:1:(length(R_PP_all{i})-1))/100 , R_PP_all{i});
end
plot((0:1:(length(R_CBP_all{i})-1))/100,R_CBP_all{i},'--');
hold off
legend('R(\lambda=1)','R (\lambda=2)','R (\lambda=5)','R (\lambda=50)','R CBP')
title('R versus time when \lambda>1')
```

```
function dR = CBP(t,R,VT,Vu,theta_E,theta,beta)
dR = VT*cos(beta-theta_E)-Vu*cos(beta-theta);
end

function dy = PP(t,y,VT,Vu,theta_E,lambda)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
theta = lambda*beta;
d_R = VT*cos(beta-theta_E)-Vu*cos(beta-theta);
d_beta = -(VT*sin(beta-theta_E)-Vu*sin(beta-theta))/R;
dy = [d_R;d_beta];
end

function dy = deriv(t,y,VT,Vu,theta_E)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
dR = VT*cos(beta-theta_E)-Vu;
d_beta = -VT*sin(beta-theta_E)/R;
dy = [dR;d_beta];
end

function [value, isterminal, direction] = StopEventPos(t, y, pos)
    value      = y(1) - pos;
    isterminal = 1;    % Stop the integration
    direction  = 0;
end
```

And the pictures look like below:

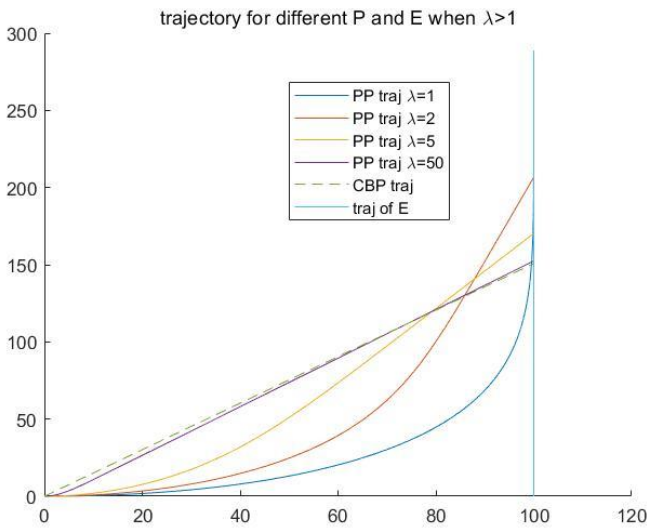


P10: trajectory for different P and E when $\lambda < 1$

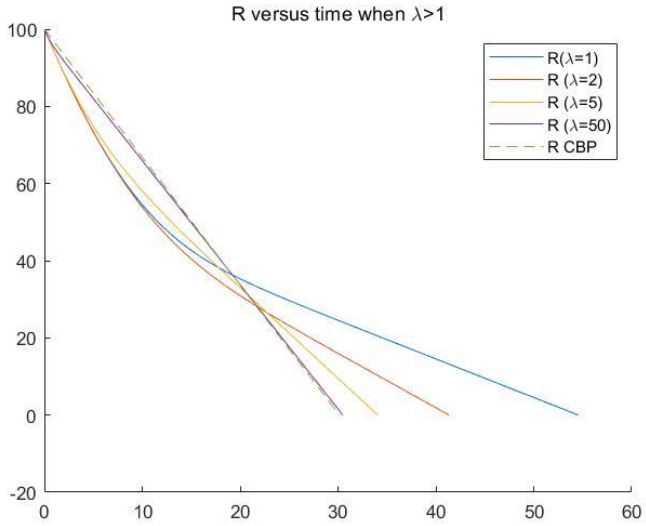


P11:R versus time when $\lambda < 1$

Trajectory of P overshoot and as I increase λ, overshoot becomes small and cconverge quickly



P12:trajectory of different P and E when $\lambda > 1$



P13:R versus time when $\lambda > 1$

As I increase λ, the trajectory of PP and R of PP go close to what happens in CBP. P converge quickly to E and R drop quickly as well

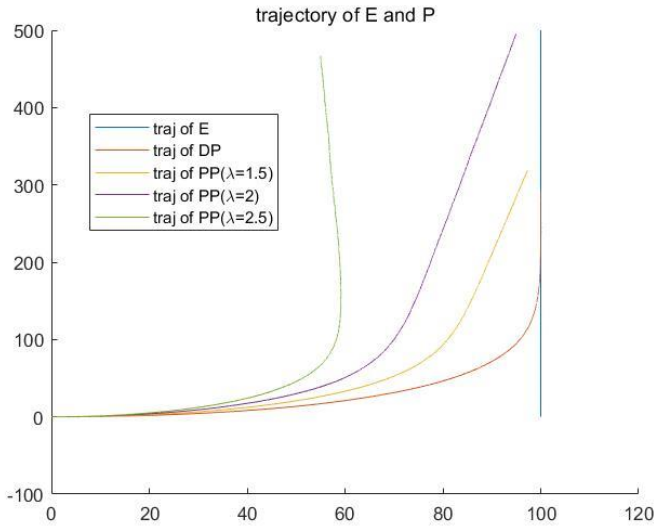
For 4 we use following codes.

```
clc;
clear all;
VT = 5;
Vu = 6;
theta_E = pi/2;
time = 0:0.01:100;
finPos = 5;
opts = odeset('Events', @(t,y)StopEventPos(t,y,finPos), 'RelTol', 1e-6, 'AbsTol', 1e-6);
odeDP = @(t,y) DP_noise(t,y,VT,Vu,theta_E);   % Anonymous derivative function with A
tspan = time;
f0 = [100;0];
[T,R_beta] = ode45(odeDP,tspan,f0,opts);   % Pass in column vector initial value

len_tra = length(T);
traj_E = zeros(len_tra,2);
traj_E(:,1) = 100*ones(len_tra,1);
traj_E(:,2) = VT*T;

R_DP =R_beta(:,1);
beta_DP = R_beta(:,2);
x = traj_E(:,1)-R_DP.*cos(beta_DP);
y = traj_E(:,2)-R_DP.*sin(beta_DP);

lams = [1.5,2,2.5];
xpos_PP_all = {};
ypos_PP_all = {};
R_PP_all = {};
for i=1:1:length(lams)
    lambda = lams(i);
    ode_PP = @(t,y) PP_noise(t,y,VT,Vu,theta_E,lambda);
    y0 = [100;0];
    [T_PP,y_PP] = ode45(ode_PP,tspan,y0,opts);
    R_PP =y_PP(:,1);
    R_PP_all{i} = R_PP;
    beta_PP = y_PP(:,2);
    len_tra_PP = length(T_PP);
    traj_E_PP = zeros(len_tra_PP,2);
    traj_E_PP(:,1) = 100*ones(len_tra_PP,1);
    traj_E_PP(:,2) = VT*T_PP;
    if(len_tra_PP>length(traj_E))
        traj_E = traj_E_PP;
    end
```

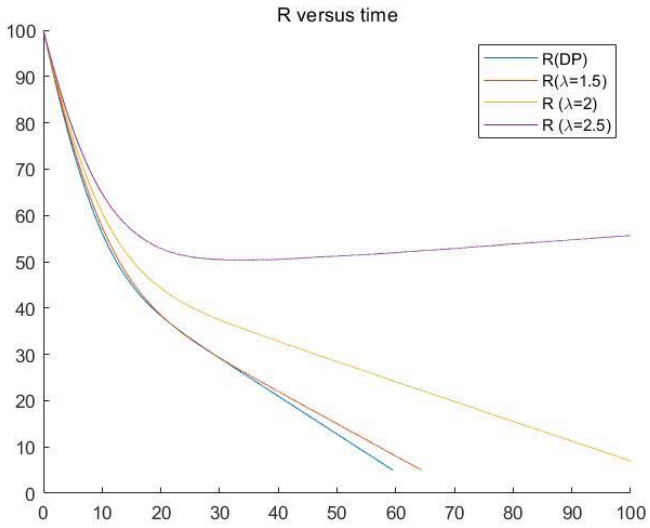
```
    xpos_PP = traj_E_PP(:,1)-R_PP.*cos(beta_PP);
    ypos_PP = traj_E_PP(:,2)-R_PP.*sin(beta_PP);
    xpos_PP_all{i} = xpos_PP;
    ypos_PP_all{i} = ypos_PP;
end


figure(1)
hold on
plot(traj_E(:,1),traj_E(:,2));
plot(x,y);
for i=1:length(xpos_PP_all)
    x_pp = xpos_PP_all{i};
    y_pp = ypos_PP_all{i};
    plot(x_pp,y_pp);
end
legend('traj of E','traj of DP', 'traj of PP(\lambda=1.5)', 'traj of PP(\lambda=2)', 'traj of
PP(\lambda=2.5)');
title("trajectory of E and P")

figure(2)
hold on
plot((0:1:length(R_DP)-1)/100, R_DP);
for i=1:length(R_PP_all)
    plot((0:1:length(R_PP_all{i})-1)/100,R_PP_all{i});
end
legend('R(DP)','R(\lambda=1.5)','R (\lambda=2)','R (\lambda=2.5)')
hold off
title("R versus time")

function dy = PP(t,y,VT,Vu,theta_E,lambda)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
theta = lambda*beta;
d_R = VT*cos(beta-theta_E)-Vu*cos(beta-theta);
d_beta = -(VT*sin(beta-theta_E)-Vu*sin(beta-theta))/R;
dy = [d_R;d_beta];
end

function dy = PP_noise(t,y,VT,Vu,theta_E,lambda)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
```

```
theta = lambda*(beta+0.25*randn);
d_R = VT*cos(beta-theta_E)-Vu*cos(beta-theta);
d_beta = -(VT*sin(beta-theta_E)-Vu*sin(beta-theta))/R;
dy = [d_R;d_beta];
end


function dy = DP_noise(t,y,VT,Vu,theta_E)
%direct DP
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
theta = beta+0.25*randn;
dR = VT*cos(beta-theta_E)-Vu*cos(beta-theta);
d_beta = -(VT*sin(beta-theta_E)-Vu*sin(beta-theta))/R;
dy = [dR;d_beta];
end

function [value, isterminal, direction] = StopEventPos(t, y, pos)
    value      = y(1) - pos;
    isterminal = 1;    % Stop the integration
    direction  = 0;
end
```

And the pictures look like below:



P14: trajectory of E and P with noise



P15: R versus time with noise

As I increase λ, the trajectory of P deviate away and converge slowly, even diverge. And R decrease slowly, even increase.

The reason is that the increase of λ mmultipliesthe noise on our control angle θ_p, and it gives a huge error.

For 5 we use following codes.

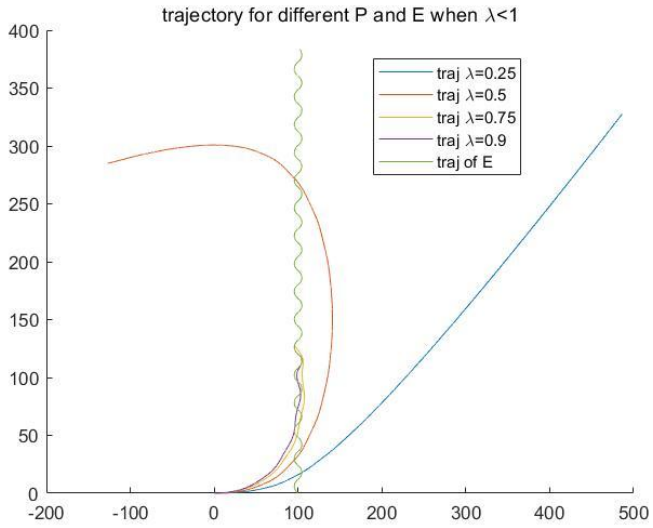
```
clc;
clear all;
c0 = 0;
VT = 5;
Vu = 6;
theta_E = pi/2;
time = 0:0.01:100;
tspan = time;
theta = asin(VT/Vu);
finPos = 0;
opts = odeset('Events', @(t,y)StopEventPos(t,y,finPos), 'RelTol', 1e-6, 'AbsTol', 1e-6);
lambdas_1 = [0.25, 0.5, 0.75, 0.9];
lambdas_2 = [1, 2, 5, 50];
lambdas =    [0.25, 0.5, 0.75, 0.9, 1, 2, 5, 50];

xpos_PP_all = {};
ypos_PP_all = {};
R_PP_all = {};
x_CBP_all = {};
y_CBP_all = {};
traj_E1 = zeros(1,1);
traj_E2 = zeros(1,1);
R_CBP_all = {};

for i=1:length(lambdas)
    lambda = lambdas(i);
    %%
    %to calculate PP
    %beta_PP = 0;
    ode_PP = @(t,y) PP(t,y,VT,Vu,theta_E,lambda);
    y0 = [100;0];
    [T_PP,y_PP] = ode45(ode_PP,tspan,y0,opts);
    R =y_PP(:,1);
    R_PP_all{i} = R;
    beta = y_PP(:,2);
    len_tra_PP = length(T_PP);

    ode_traj = @(t,s) cal_traj_E(t,s,VT,theta_E);
    [T_traj_E,traj_E] = ode45(ode_traj, T_PP, y0,opts);
    %theta_tao = cos(T_PP)+pi/2;
    %traj_E = zeros(len_tra_PP,2);
    %traj_E(:,1) = 100*ones(len_tra_PP,1);
```

```
    %traj_E(:,2) = VT*T_PP;
    xpos_PP = traj_E(:,1)-R.*cos(beta);
    ypos_PP = traj_E(:,2)-R.*sin(beta);
    xpos_PP_all{i}=xpos_PP;
    ypos_PP_all{i}=ypos_PP;

    if(i<length(lambdas_1))
        if(length(traj_E(:,1))>length(traj_E1(:,1)))
            traj_E1 = traj_E;
        end
    else
        if(length(traj_E(:,1))>length(traj_E2(:,1)))
            traj_E2 = traj_E;
        end
    end

    %%
    %to calculate CBP
    beta_CBP = 0;
    ode_CBP = @(t,R) CBP(t,R,VT,Vu,theta_E,theta,beta_CBP);
    R0 = 100;
    [T_CBP,R_CBP] = ode45(ode_CBP,tspan,R0,opts);
    R_CBP_all{i} = R_CBP;
    len_tra_CBP = length(T_CBP);

    [T_traj_E_CBP,traj_E_CBP] = ode45(ode_traj, T_CBP, y0,opts);
    %traj_E_CBP = zeros(len_tra_CBP,2);
    %traj_E_CBP(:,1) = 100*ones(len_tra_CBP,1);
    %traj_E_CBP(:,2) = VT*T_CBP;
    x_CBP = traj_E_CBP(:,1)-R_CBP.*cos(beta_CBP);
    y_CBP = traj_E_CBP(:,2)-R_CBP.*sin(beta_CBP);
    x_CBP_all{i} = x_CBP;
    y_CBP_all{i}=y_CBP;
    if(i<length(lambdas_1))
        if(length(traj_E_CBP(:,1))>length(traj_E1(:,1)))
            traj_E1 = traj_E_CBP;
        end
    else
        if(length(traj_E_CBP(:,1))>length(traj_E2(:,1)))
            traj_E2 = traj_E_CBP;
        end
    end

end
```

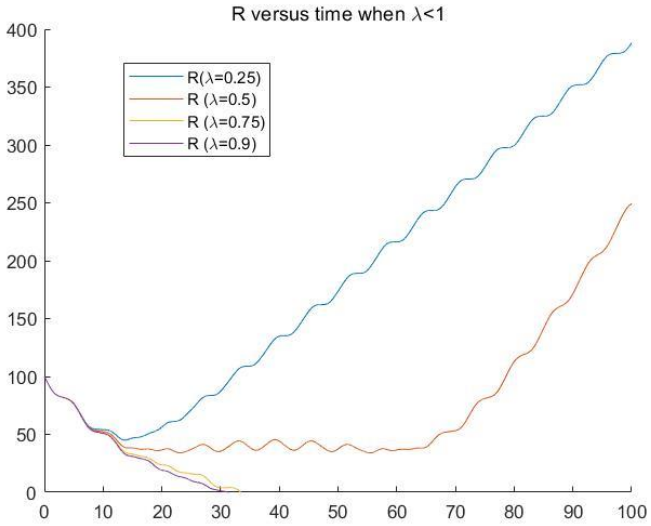
```
%%
%PP and CBP <1
figure(1) %trajectory
hold on
for i=1:length(lambdas_1)
    plot(xpos_PP_all{i},ypos_PP_all{i})
end
plot(traj_E1(:,1),traj_E1(:,2))
hold off
legend('traj \lambda=0.25','traj \lambda=0.5','traj \lambda=0.75','traj \lambda=0.9','traj of E')
title('trajectory for different P and E when \lambda<1')

figure(2) %R
hold on
for i=1:length(lambdas_1)
    plot((0:1:(length(R_PP_all{i})-1))/100 , R_PP_all{i});
end
hold off
legend('R(\lambda=0.25)','R (\lambda=0.5)','R (\lambda=0.75)','R (\lambda=0.9)')
title('R versus time when \lambda<1')


%%
%PP and CBP >1
figure(3) %trajectory
hold on
len = length(lambdas_1);
for i=len+1:1:len+length(lambdas_2)
    plot(xpos_PP_all{i},ypos_PP_all{i});
end
plot(x_CBP_all{i},y_CBP_all{i},'--');
plot(traj_E2(:,1),traj_E2(:,2))

legend('PP traj \lambda=1','PP traj \lambda=2',...
    'PP traj \lambda=5','PP traj \lambda=50',...
   'CBP traj', 'traj of E');
title('trajectory for different P and E when \lambda>1')
hold off


figure(4) %R
hold on
for i=len+1:1:len+length(lambdas_2)
```

```
    plot((0:1:(length(R_PP_all{i})-1))/100 , R_PP_all{i});
end
plot((0:1:(length(R_CBP_all{i})-1))/100,R_CBP_all{i},'--');
hold off
legend('R(\lambda=1)','R (\lambda=2)','R (\lambda=5)','R (\lambda=50)','R CBP')
title('R versus time when \lambda>1')

function ds = cal_traj_E(t,s,VT,theta_E)
    dx = VT*cos(theta_E+cos(t));
    dy = VT*sin(theta_E+cos(t));
    ds = [dx;dy];
end

function dR = CBP(t,R,VT,Vu,theta_E,theta,beta)
dR = VT*cos(beta-theta_E-cos(t))-Vu*cos(beta-theta);
end

function dy = PP(t,y,VT,Vu,theta_E,lambda)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
theta = lambda*beta;
d_R = VT*cos(beta-theta_E-cos(t))-Vu*cos(beta-theta);
d_beta = -(VT*sin(beta-theta_E-cos(t))-Vu*sin(beta-theta))/R;
dy = [d_R;d_beta];
end

function dy = deriv(t,y,VT,Vu,theta_E)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
dR = VT*cos(beta-theta_E-cos(t))-Vu;
d_beta = -VT*sin(beta-theta_E-cos(t))/R;
dy = [dR;d_beta];
end

function [value, isterminal, direction] = StopEventPos(t, y, pos)
    value      = y(1) - pos;
    isterminal = 1;    % Stop the integration
    direction  = 0;
end
```

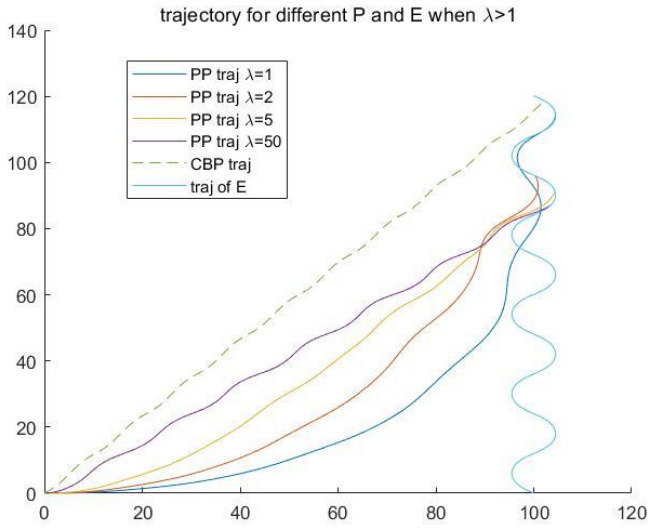
And the pictures look like below:



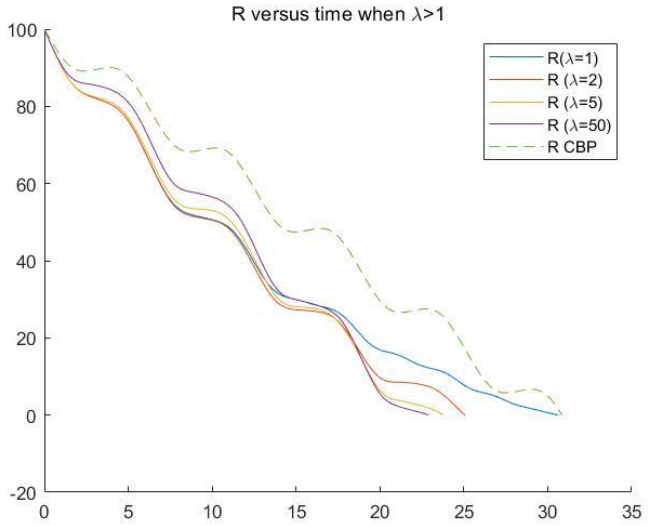
P16: trajectory for different P and E when $\lambda < 1$



P17: R versus time when $\lambda < 1$



P18: trajectory for different P and E when $\lambda > 1$



P19: R versus time when $\lambda > 1$

It is quite different from what in Problem3. If $\lambda > 1$, almost all PP behave better than CBP and when increase λ, PP will get away rather than converge to the trajectory of CBP. And if λ is too small, it will get a larger overshoot.

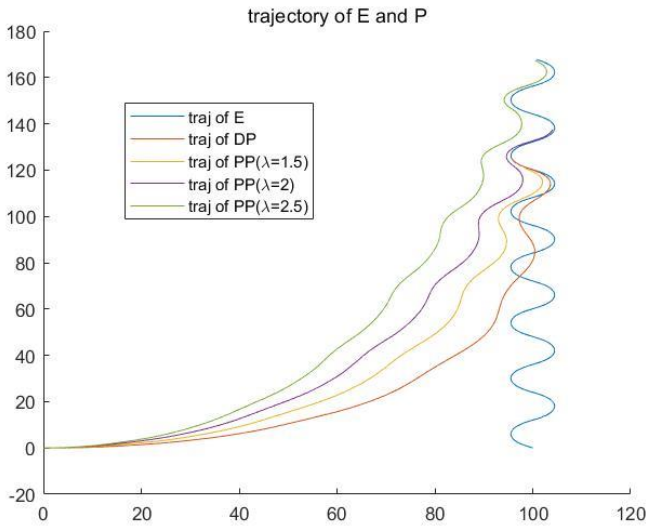
For 6 we use following codes.

```
clc;
clear all;
VT = 5;
Vu = 6;
theta_E = pi/2;
time = 0:0.01:100;
finPos = 0.5;
opts = odeset('Events', @(t,y)StopEventPos(t,y,finPos), 'RelTol', 1e-6, 'AbsTol', 1e-6);
odeDP = @(t,y) DP_noise(t,y,VT,Vu,theta_E);   % Anonymous derivative function with A
tspan = time;
f0 = [100;0];
[T,R_beta] = ode45(odeDP,tspan,f0,opts);   % Pass in column vector initial value

ode_traj = @(t,s) cal_traj_E(t,s,VT,theta_E);
[T_traj_E,traj_E] = ode45(ode_traj, T, f0,opts);
len_tra = length(T);

R_DP =R_beta(:,1);
beta_DP = R_beta(:,2);
x = traj_E(:,1)-R_DP.*cos(beta_DP);
y = traj_E(:,2)-R_DP.*sin(beta_DP);

lams = [1.5,2,2.5];
xpos_PP_all = {};
ypos_PP_all = {};
R_PP_all = {};

for i=1:1:length(lams)
    lambda = lams(i);
    ode_PP = @(t,y) PP_noise(t,y,VT,Vu,theta_E,lambda);

    y0 = [100;0];
    [T_PP,y_PP] = ode45(ode_PP,tspan,y0,opts);

    ode_traj = @(t,s) cal_traj_E(t,s,VT,theta_E);
    [T_traj_E,traj_E_PP] = ode45(ode_traj, T_PP, y0,opts);
    if(length(traj_E_PP)>length(traj_E))
        traj_E = traj_E_PP;
    end
    R_PP =y_PP(:,1);
    R_PP_all{i} = R_PP;
```

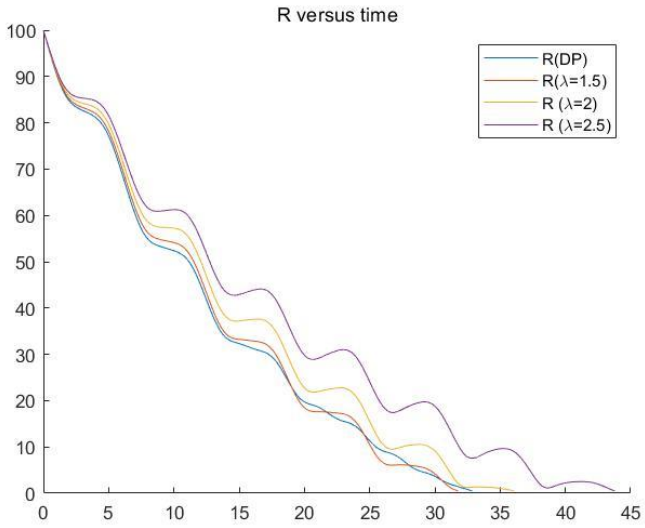
```
    beta_PP = y_PP(:,2);
    len_tra_PP = length(T_PP);
    %traj_E_PP = zeros(len_tra_PP,2);
    %traj_E_PP(:,1) = 100*ones(len_tra_PP,1);
    %traj_E_PP(:,2) = VT*T_PP;
    xpos_PP = traj_E_PP(:,1)-R_PP.*cos(beta_PP);
    ypos_PP = traj_E_PP(:,2)-R_PP.*sin(beta_PP);
    xpos_PP_all{i} = xpos_PP;
    ypos_PP_all{i} = ypos_PP;
end


figure(1)
hold on
plot(traj_E(:,1),traj_E(:,2));
plot(x,y);
for i=1:length(xpos_PP_all)
    x_pp = xpos_PP_all{i};
    y_pp = ypos_PP_all{i};
    plot(x_pp,y_pp);
end
legend('traj of E','traj of DP', 'traj of PP(\lambda=1.5)', 'traj of PP(\lambda=2)', 'traj of
PP(\lambda=2.5)');
title("trajectory of E and P")

figure(2)
hold on
plot((0:1:length(R_DP)-1)/100, R_DP);
for i=1:length(R_PP_all)
    plot((0:1:length(R_PP_all{i})-1)/100,R_PP_all{i});
end
legend('R(DP)','R(\lambda=1.5)','R (\lambda=2)','R (\lambda=2.5)')
hold off
title("R versus time")


function dy = PP_noise(t,y,VT,Vu,theta_E,lambda)
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
theta = lambda*(beta+0.25*randn);
d_R = VT*cos(beta-theta_E-cos(t))-Vu*cos(beta-theta);
d_beta = -(VT*sin(beta-theta_E-cos(t))-Vu*sin(beta-theta))/R;
dy = [d_R;d_beta];
```

```
end


function dy = DP_noise(t,y,VT,Vu,theta_E)
%direct DP
%y(1) = R, y(2) = beta
R = y(1);
beta = y(2);
theta = beta+0.25*randn;
dR = VT*cos(beta-theta_E-cos(t))-Vu*cos(beta-theta);
d_beta = -(VT*sin(beta-theta_E-cos(t))-Vu*sin(beta-theta))/R;
dy = [dR;d_beta];
end

function ds = cal_traj_E(t,s,VT,theta_E)
    dx = VT*cos(theta_E+cos(t));
    dy = VT*sin(theta_E+cos(t));
    ds = [dx;dy];
end

function [value, isterminal, direction] = StopEventPos(t, y, pos)
    value      = y(1) - pos;
    isterminal = 1;    % Stop the integration
    direction  = 0;
end
```

And the picture look like below:



P20: trajectory of E and P



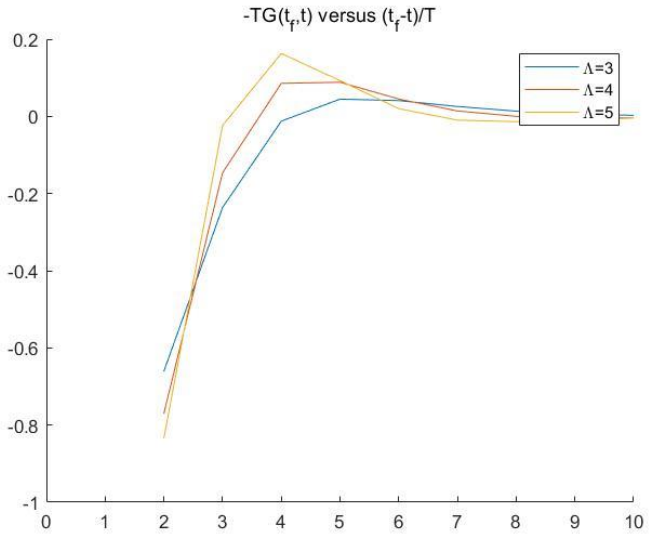
P21: R versus time

It is different from problem4 because all trajectory converge

For 7a we use following codes.

```
clc;
clear all;
Lams = [3,4,5];
t = 0:0.001:9.999;
tf_star = 10;
T = 0.001;
opts = odeset( 'RelTol', 1e-6, 'AbsTol', 1e-6);

figure(1)
hold on
for i=1:length(Lams)
    lam = Lams(i);
    ode_p = @(t,p) deriv_p(t,p,lam,tf_star,T);
    tp = 9.999-t;
    p0 = [1;0];
    [T_P,p] = ode45(ode_p,tp,p0,opts);
    p_tao = flip(p); %from zeros to tf*
    tao = flip(T_P); %from zeros to tf*
    G_tf_tao = lam*p_tao(:,2)./(T*(tf_star-tao));
    TG_minus = -T*G_tf_tao; %from zeros to tf*
    %flip(TG_minus);
    x_axis = (tf_star-tao)/T;
    plot(x_axis(1:end-1),TG_minus(1:end-1));
end
hold off
legend('\Lambda=3','\Lambda=4','\Lambda=5');
xlim([0,10]);
title('-TG(t_f,t) versus (t_f-t)/T')

function dp = deriv_p(t,p,lam,tf_star,T)
    p1 = p(1);
    p2 = p(2);
    dp1 = lam/(T*(tf_star-t))*p2;
    dp2 = -p1+p2/T;
    dp = [dp1;dp2];
end
```

And the picture look like below:



P22: $-TG(t_f, t)$ $versus$ $\frac{t_f - t}{T}$

7b:

Interpret 7a: if evader manuvers at the very close time when the pursuer catch it, basically very close to 0 of X-axis in P22(very little of time-to-go), then the miss distance will be very large. After that, as time-to-go goes larger, the miss distance will decrease to 0, then overshoot a little bit. Finally, it goes to 0 gradually.

Picture(a): Evader maneuvers at 0 and only maneuvers once. The normalized time-to-go time is (0.5-0)/0.001 = 500. So from P22 we can know that for all Λ the miss distance is moderately

small. And The larger the Λ, the smaller the distance. Because that's what shown in P22

Picture(c): Almost the same as A except that evader maneuvers at normalized time-to-go 1/0.001 = 1000. So the miss distance is smaller.

Picture(e): The same, miss distance is even smaller.

Picture(b): The final maneuver is almost at 0.4, normalized time-to-go 0.1/0.001=100. Then get a larger miss distance than (a)

Picture(d):The final maneuver is almost 0.8 to 0.9, normalized time-to-go is larger, get a less miss distance than (b).

Picture(f):The final maneuver is almost 4.5~4.6, normalized time-to-go is the largest, get the least distance.