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)

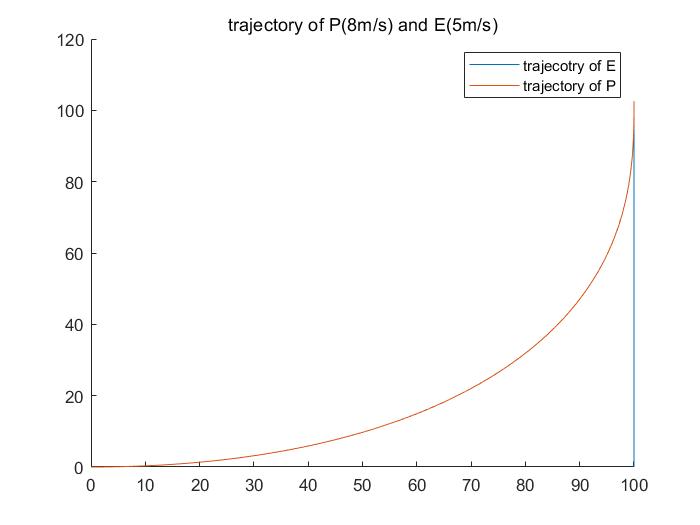
value = y(1) - pos;

isterminal = 1; % Stop the integration

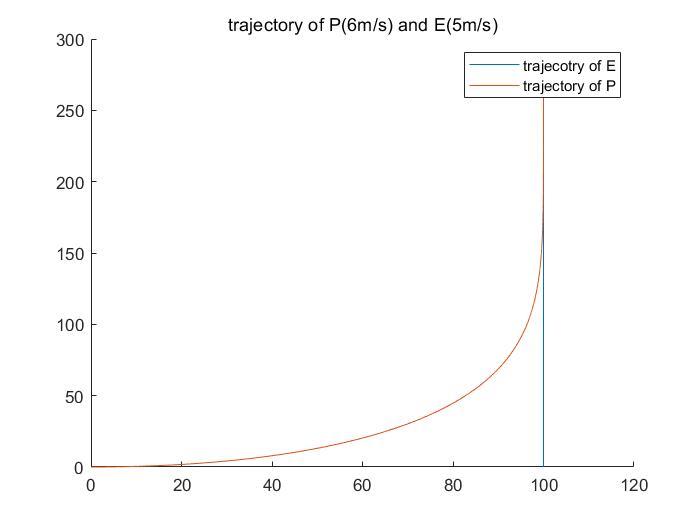
direction = 0;

end

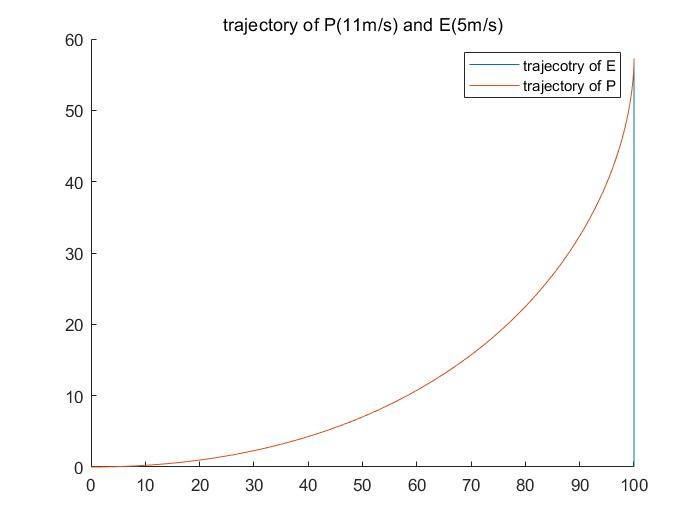
And the pictures look like below:



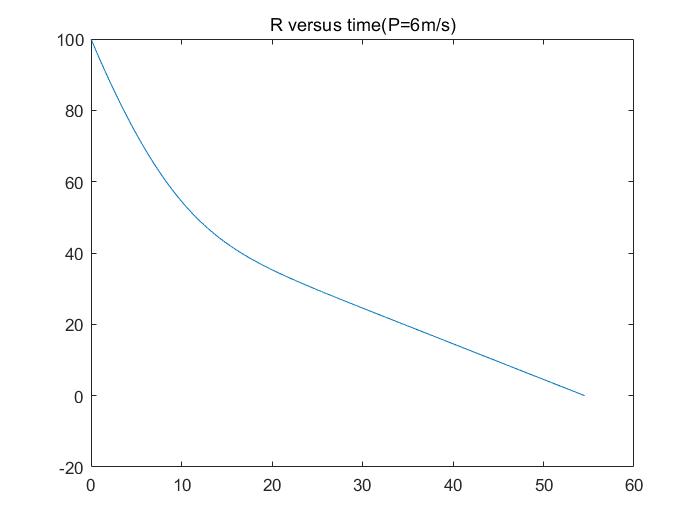
P1: trajectory of P(8m/s) and E(5m/s)



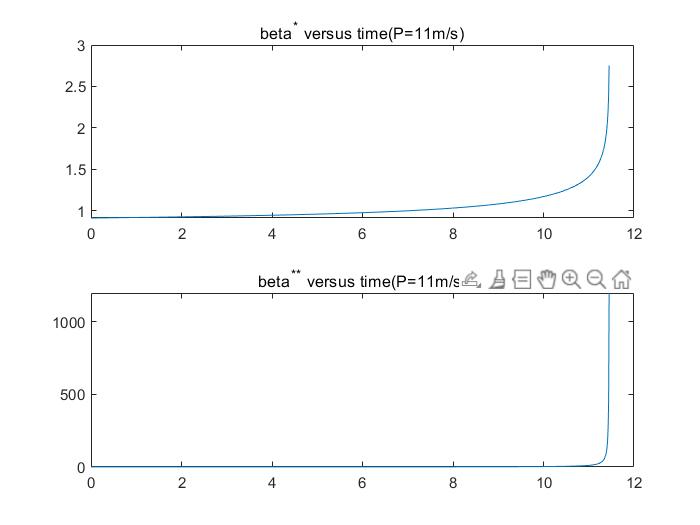
P2: trajectory of P(6m/s) and E(5m/s)

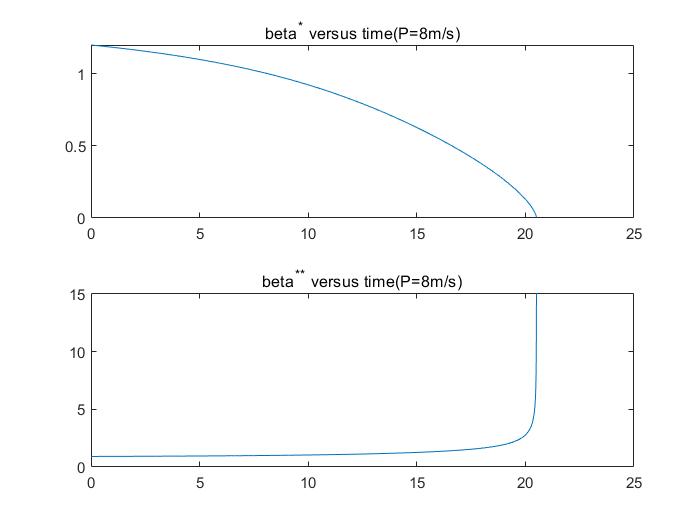


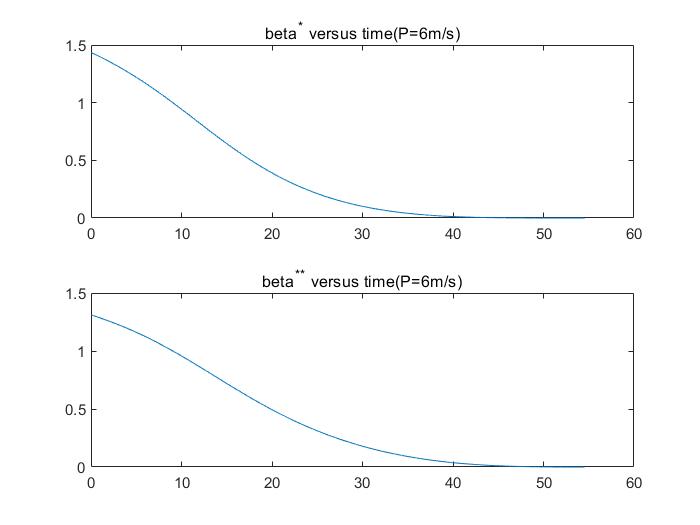
P3: trajectory of P(11m/s) and E(5m/s)



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



P5:   and   versus time(P=11m/s)

P6:   and   versus time(P=8m/s)

P7:   and   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)

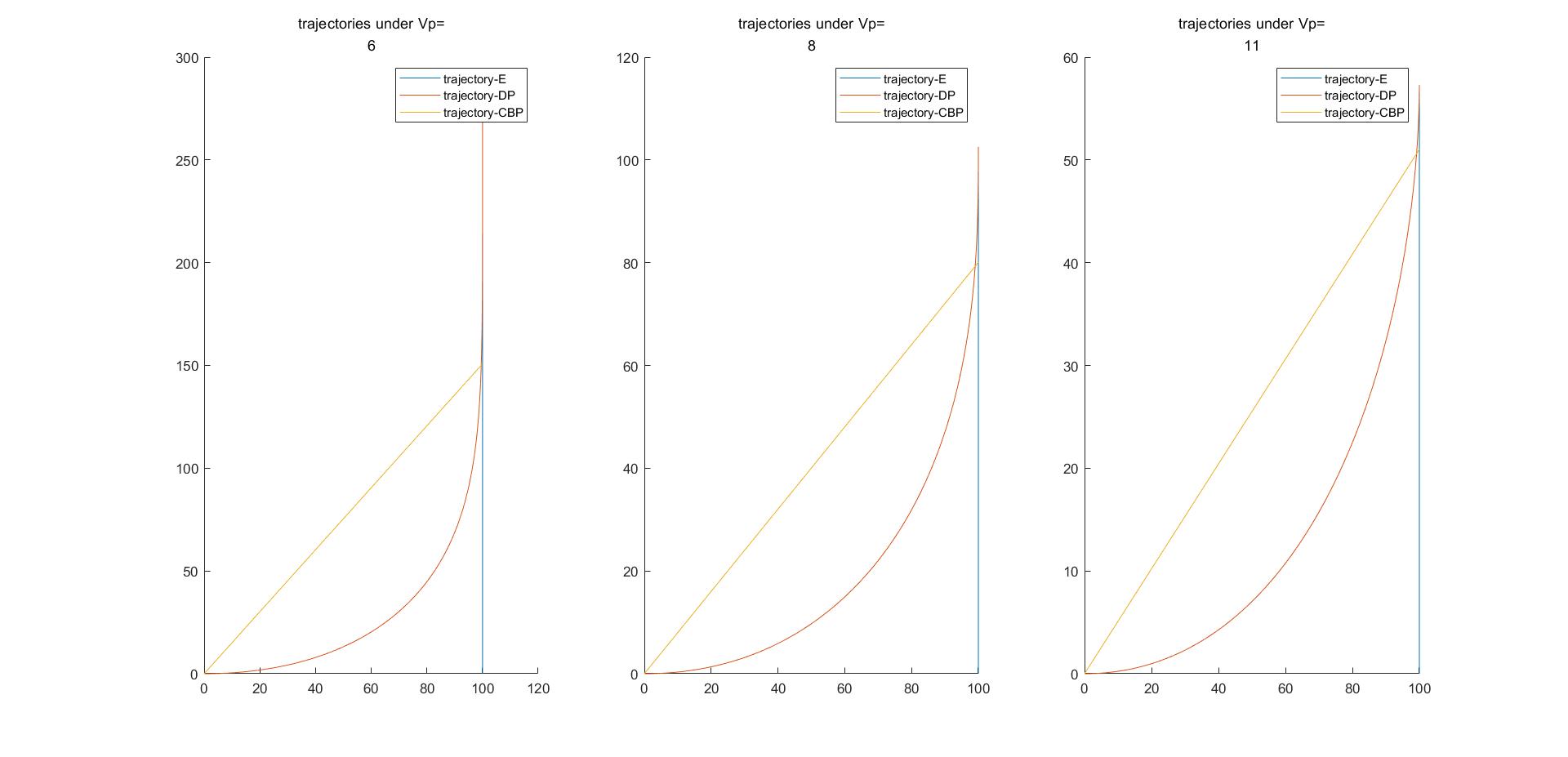
value = y(1) - pos;

isterminal = 1; % Stop the integration

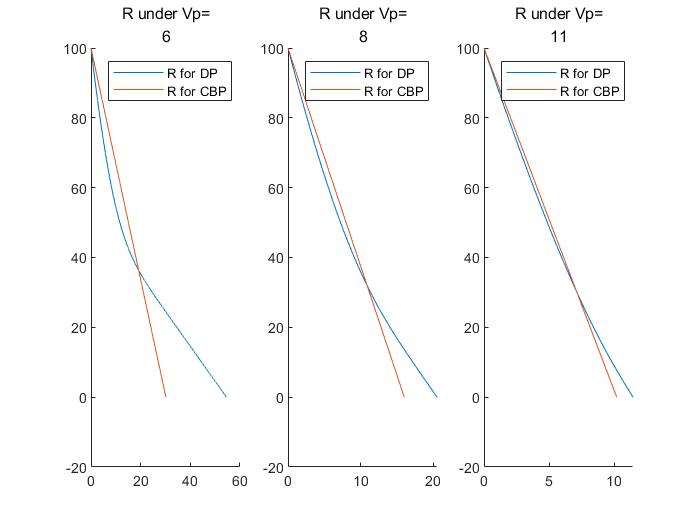
direction = 0;

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)

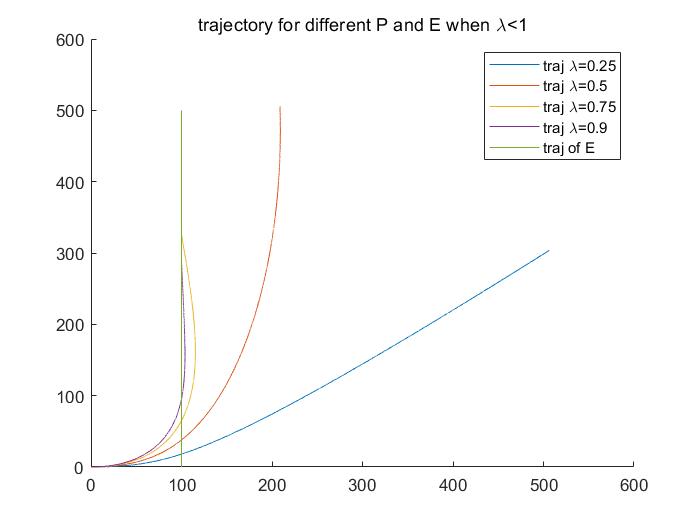
value = y(1) - pos;

isterminal = 1; % Stop the integration

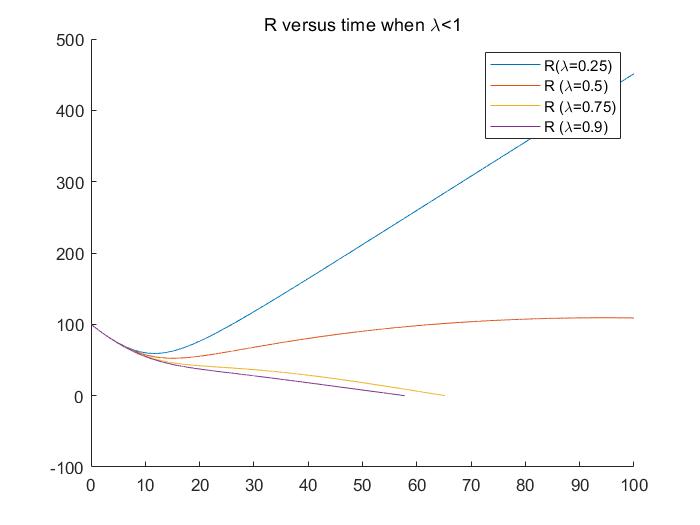
direction = 0;

end

And the pictures look like below:

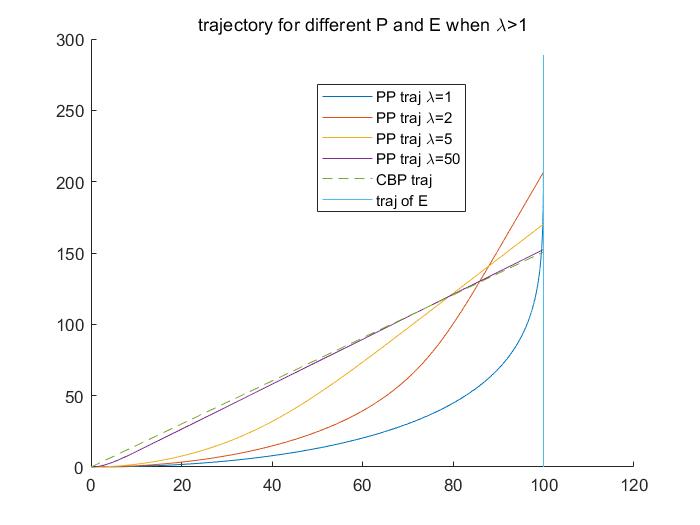


P10: trajectory for different P and E when

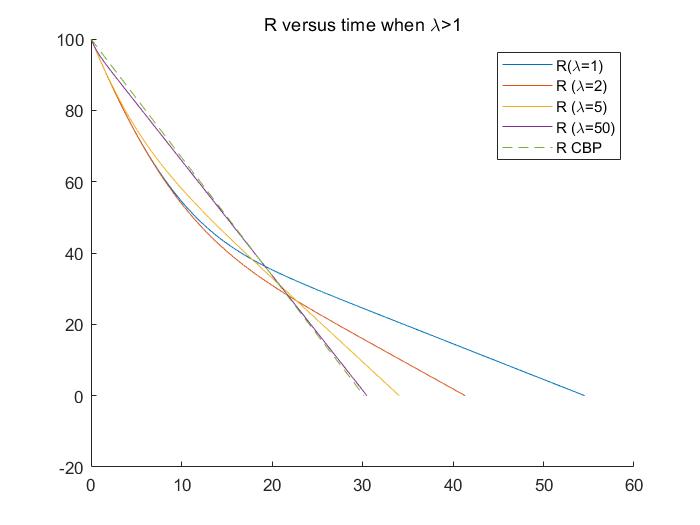


P11:R versus time when

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



P12:trajectory of different P and E when



P13:R versus time when

As I increase , 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)

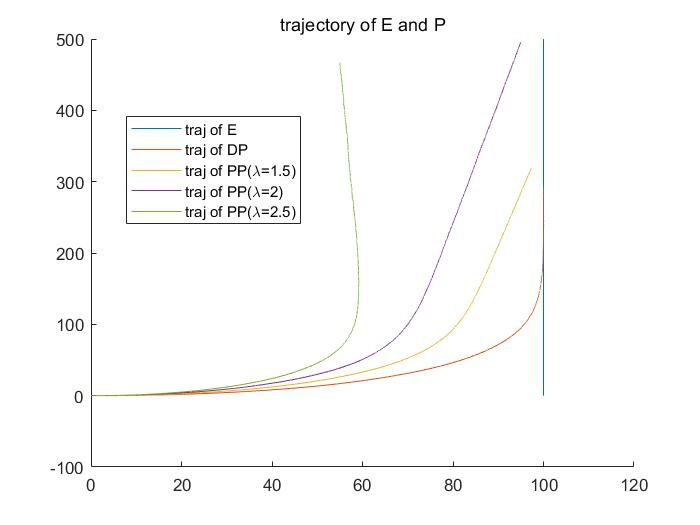
value = y(1) - pos;

isterminal = 1; % Stop the integration

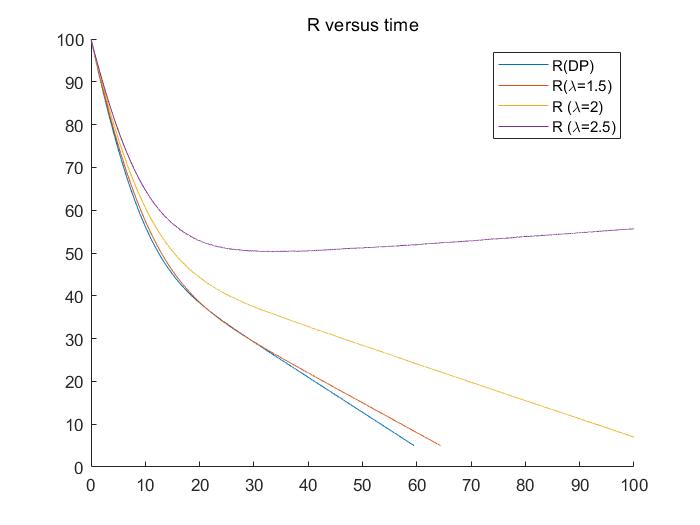
direction = 0;

end

And the pictures look like below:



P14: trajectory of E and P with noise



P15: R versus time with noise

As I increase , the trajectory of P deviate away and converge slowly, even diverge. And R decrease slowly, even increase. The reason is that the increase of mmultipliesthe noise on our control angle , 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)

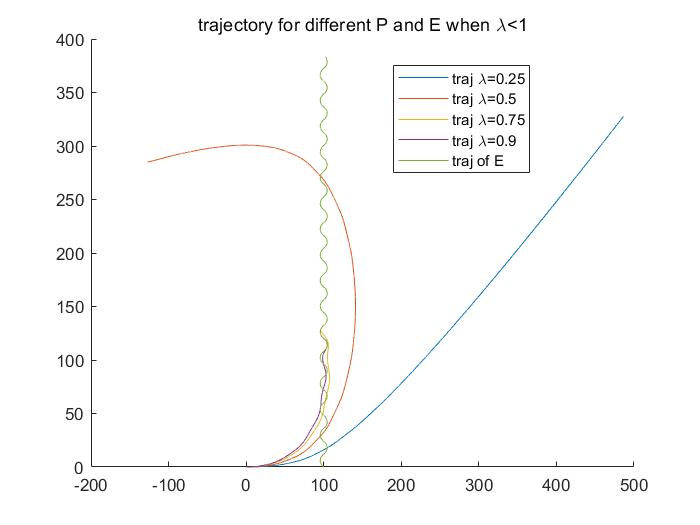
value = y(1) - pos;

isterminal = 1; % Stop the integration

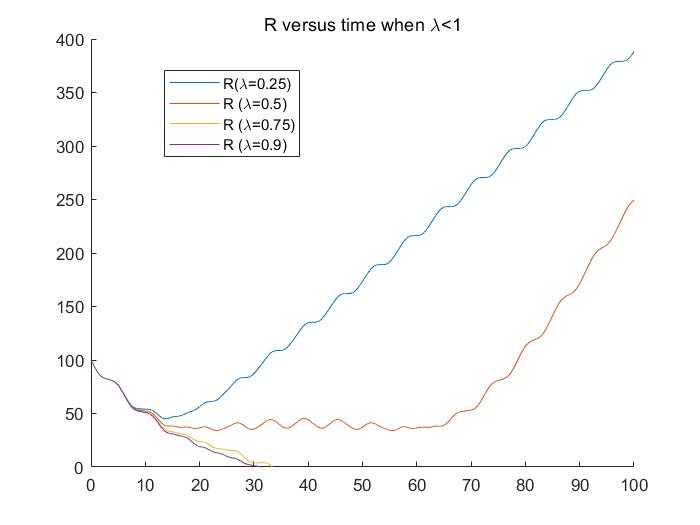
direction = 0;

end

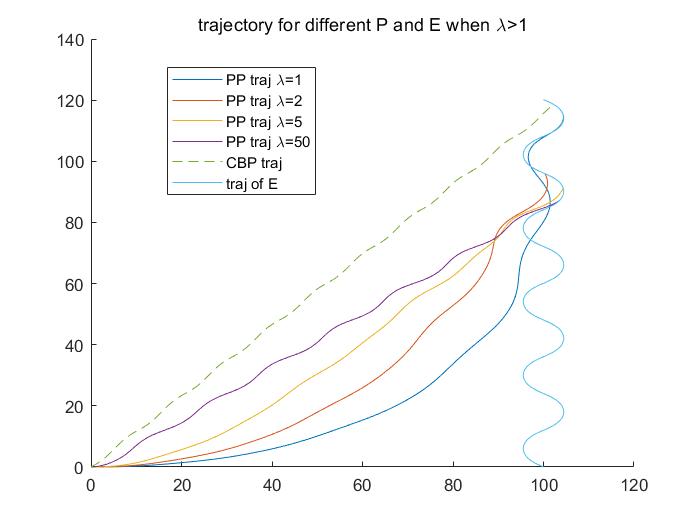
And the pictures look like below:



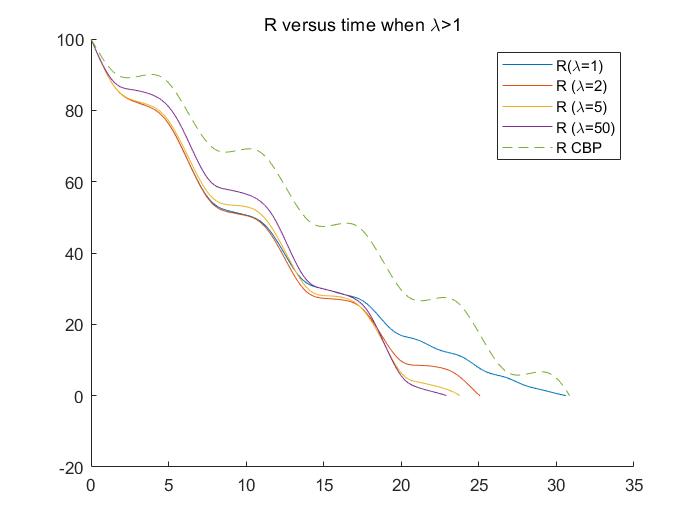
P16: trajectory for different P and E when



P17: R versus time when



P18: trajectory for different P and E when



P19: R versus time when

It is quite different from what in Problem3. If , almost all PP behave better than CBP and when increase , PP will get away rather than converge to the trajectory of CBP. And if 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)

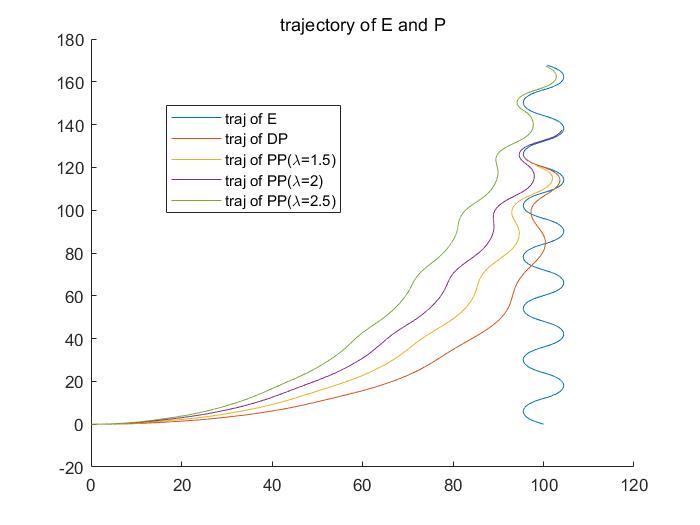
value = y(1) - pos;

isterminal = 1; % Stop the integration

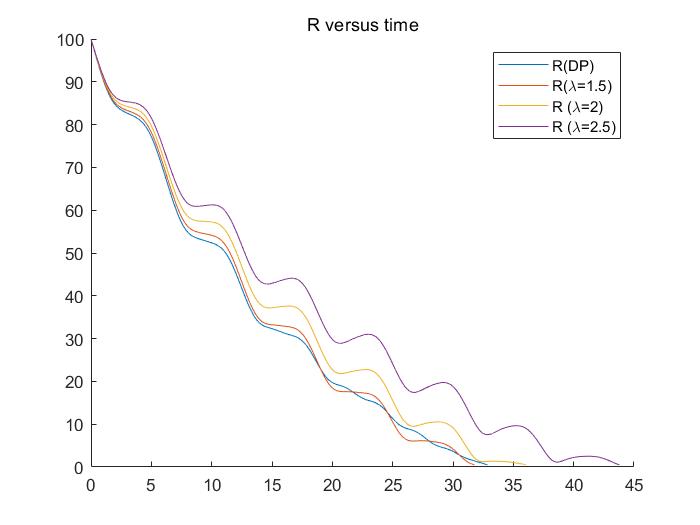
direction = 0;

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);

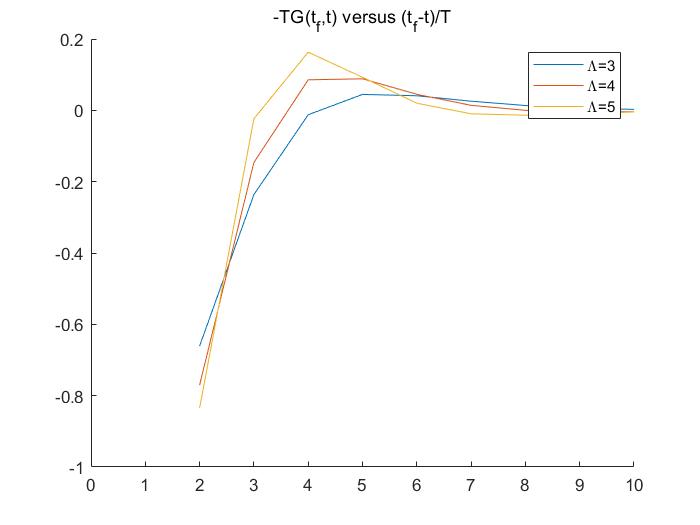
dp1 = lam/(T\*(tf\_star-t))\*p2;

dp2 = -p1+p2/T;

dp = [dp1;dp2];

end

And the picture look like below:



P22:

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 the miss distance is moderately small. And The larger the , 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.