**Appendix**

|  |  |
| --- | --- |
| **Part 1** | **Matlab codes and plots** |
| **Part 2** | **MAtlab codes and plots** |

**Part 1**

1. **R = 1(unit step)**

function hw5q34

%system data

A = [0 1; 1 -1];

B = [0; 1];

nx = size(A,2);

nu = size(B,2);

Xs0 = [16; 0]; %initial position at 16

tfinal = 5; %final time 5 sec

dt = 0.01;

tvec = 0:dt:tfinal;

rc = 1;

Q = 10\*eye(nx); R = eye(nu);

%checking for controllability

CMAT = ctrb(A,B);

if rank(CMAT) == nx

fprintf('\n This system is controllable \n');

else

return;

end

%checking for detectability

C = sqrt(Q)';

OMAT = obsv(A,C);

if rank(OMAT) == nx

fprintf('\n This system is observable \n');

else

return;

end

fprintf('\n Finite Time LQR \n')

fprintf('\n Cost Function = 1/2 Xf^T\*Qf\*Xf + 1/2 int\_0^tfinal (X^T Q X + U^T R U)dt \n');

Qf = [1000000 0; 0 0];

Phist = timeRicSol(A,B,Q,R,Qf,tvec); %solving the DARE and storing solution

X0 = Xs0;

Y(1,:) = X0'; %storing states

for ct = 1:length(tvec)-1

P = reshape(Phist(ct,:), nx, nx);

[tv, yv] = ode45(@sysde, [tvec(ct) tvec(ct+1)], X0, [], A, B, Q, R, P, rc);

Y(ct+1,:) = yv(end,:);

X0 = yv(end,:);

end

%evaluating control history for plots

for ct = 1:length(tvec)

P = reshape(Phist(ct,:), nx, nx);

Xs = Y(ct,:)';

U(ct,1) = -R\(B'\*P)\*(Xs);

end

%plotting results

figure(1)

plot(tvec, Y(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Finite Horizon');

figure(2)

plot(tvec, Y(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Finite Horizon');

figure(3)

plot(tvec, U, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Finite Horizon');

clc; fprintf('\n Press any key to continue \n'); pause; fprintf('\n Finite vs Infinite Time LQT \n')

Pinf = care(A,B,Q,R); %CARE can also be supplied with more arguments such as R,S,E etc. MATLAB suggests using icare

X0 = Xs0;

[tv, yinf] = ode45(@sysde, tvec, X0, [], A, B, Q, R, Pinf, rc);

%Evaluating control history to generate plots

for ct = 1:length(tvec)

Xinf = yinf(ct,:)';

Uinf(ct,1) = -R\(B'\*Pinf)\*(Xinf);

end

figure(4)

plot(tvec, yinf(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Infinite Horizon');

figure(5)

plot(tvec, yinf(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Infinite Horizon');

figure(6)

plot(tvec, Uinf, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Infinite Horizon');

%comparison of finite vs. infinite time control

figure(7)

plot(tvec, Y(:,1), tvec, yinf(:,1), 'r-.', 'linewidth',2); title('Comparison of X1')

legend('Finite time', 'Infinite Time')

figure(8)

plot(tvec, Y(:,2), tvec, yinf(:,2), 'r-.', 'linewidth',2); title('comparison of X2')

legend('Finite time', 'Infinite Time')

figure(9)

plot(tvec, U, tvec, Uinf,'r-.', 'linewidth',2); title('Control comparison')

legend('Finite time', 'Infinite Time')

end

%local functions

function xd = sysde(t, x, A, B, Q, R, P, rc)

K = R\(B'\*P);

u = -K\*(x - [rc; 0]);

xd = A\*x + B\*u;

end

function Phist = timeRicSol(A,B,Q,R,Qf,tvec)

P0 = Qf; pvec0 = reshape(P0, numel(P0), 1); % solving the matrix differential equation starting at tf and moving backwards

[Thist, Phist] = ode45(@ricDE, tvec, pvec0, [], A, B, Q, R);

Phist = flipud(Phist);

end

function pdvec = ricDE(t, pvec, A, B, Q, R)

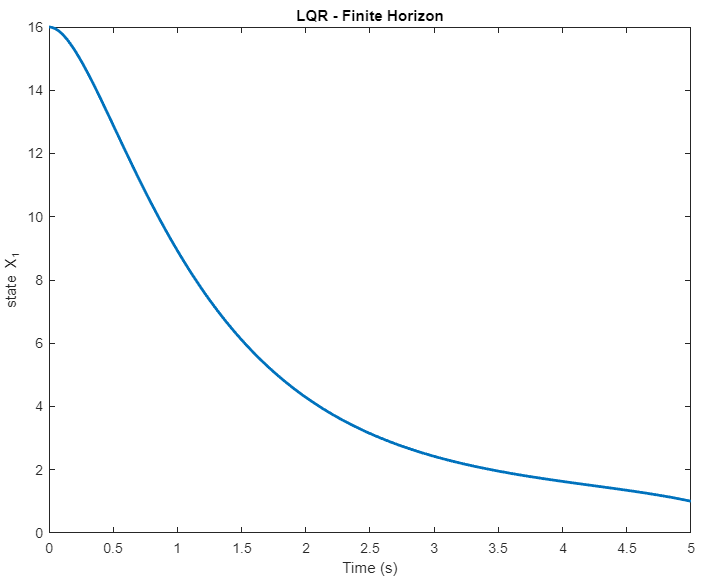
P = reshape(pvec, size(A));

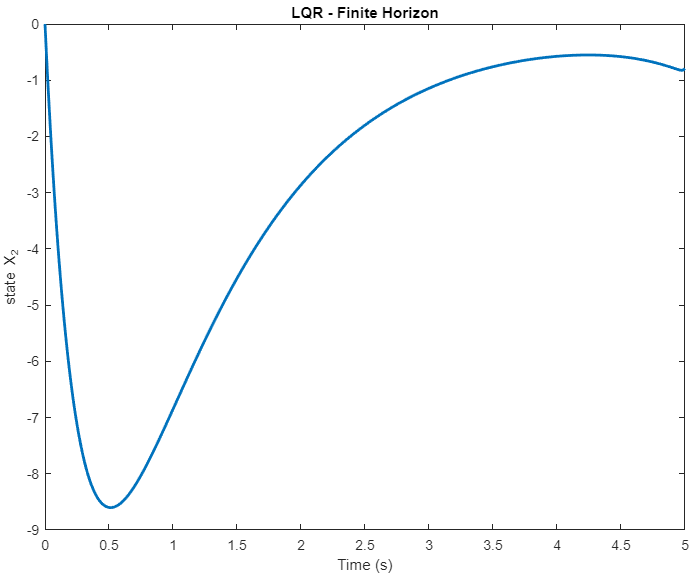
PD = P\*A + A'\*P - P\*B\*(R\(B'\*P)) + Q;

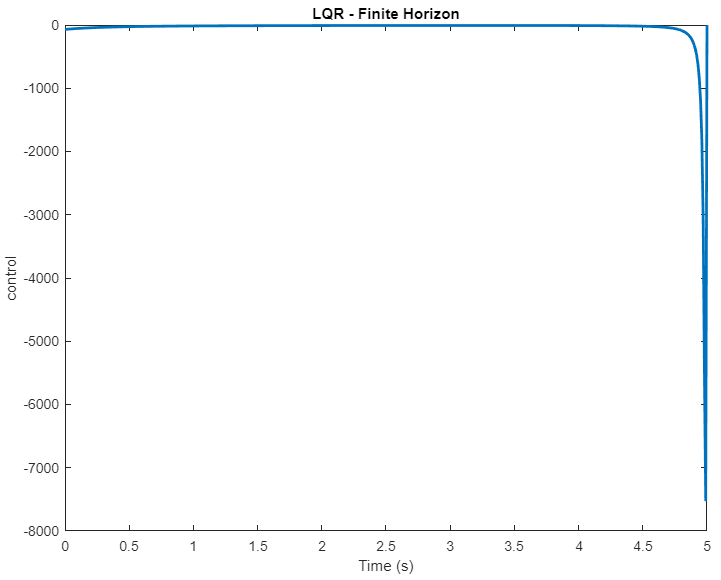
pdvec = reshape(PD, numel(PD), 1);

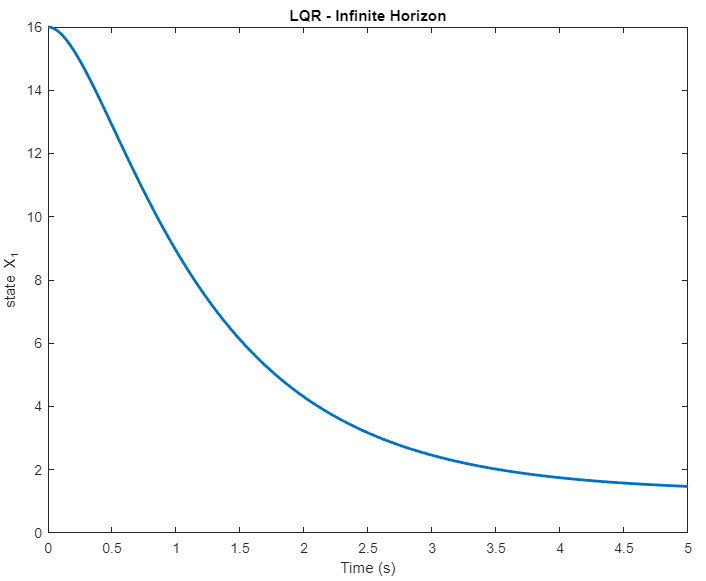
end

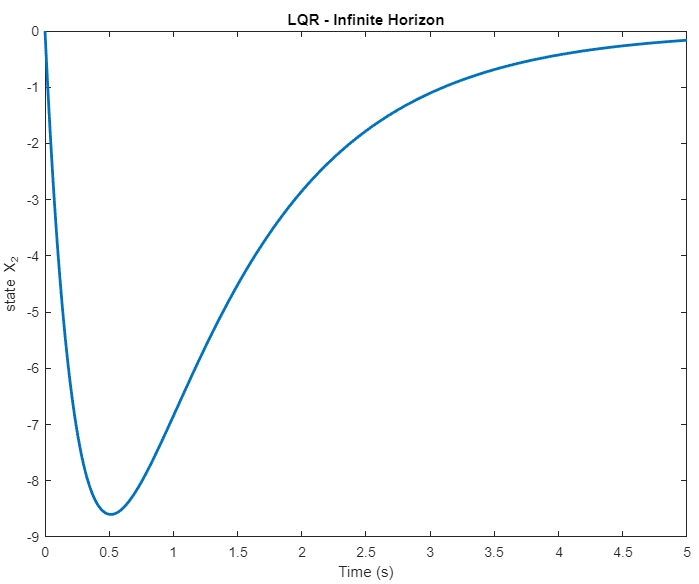
**Plots**

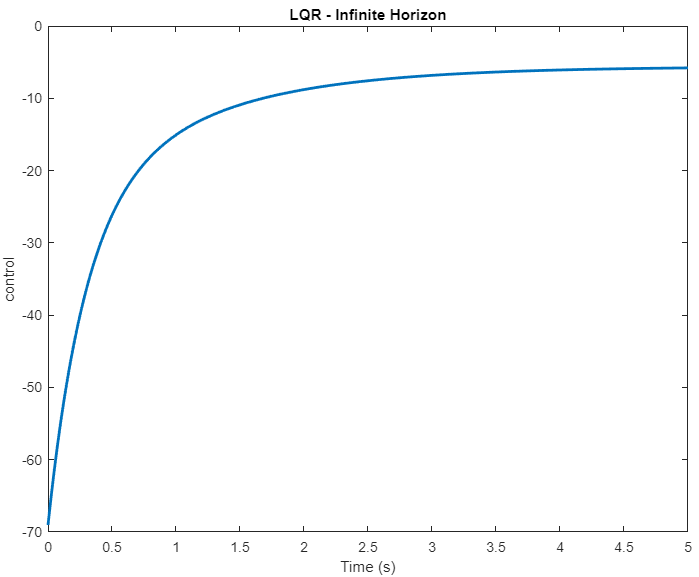
****

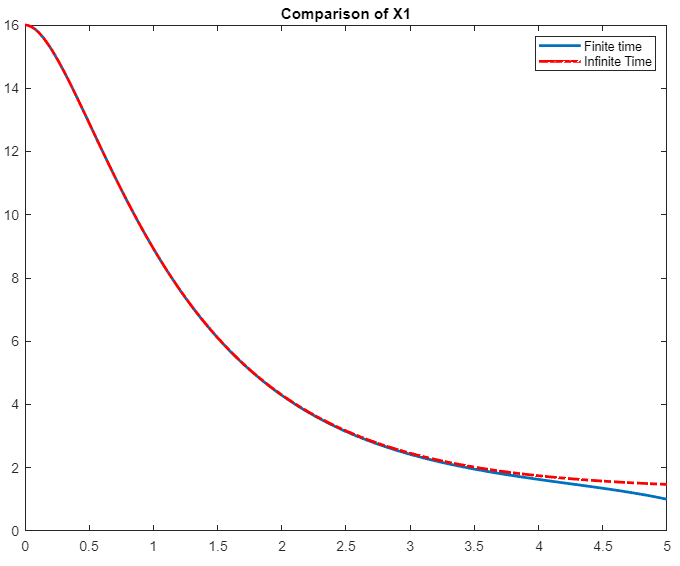
****

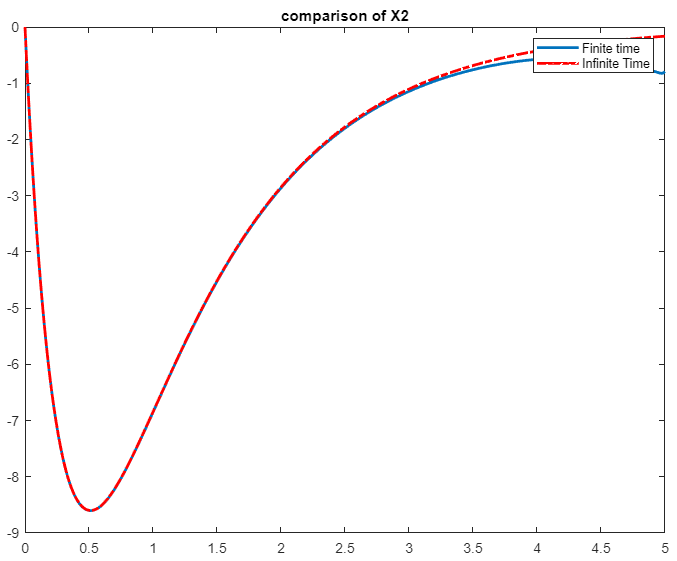
****

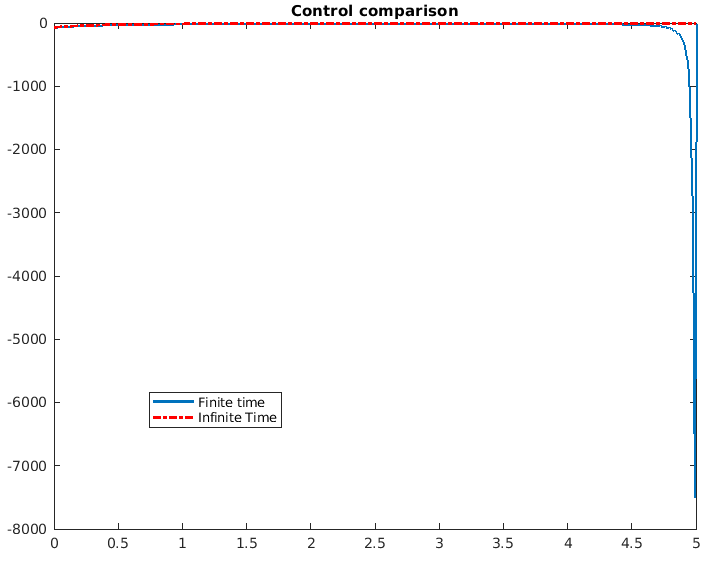
****

****

****

****

****

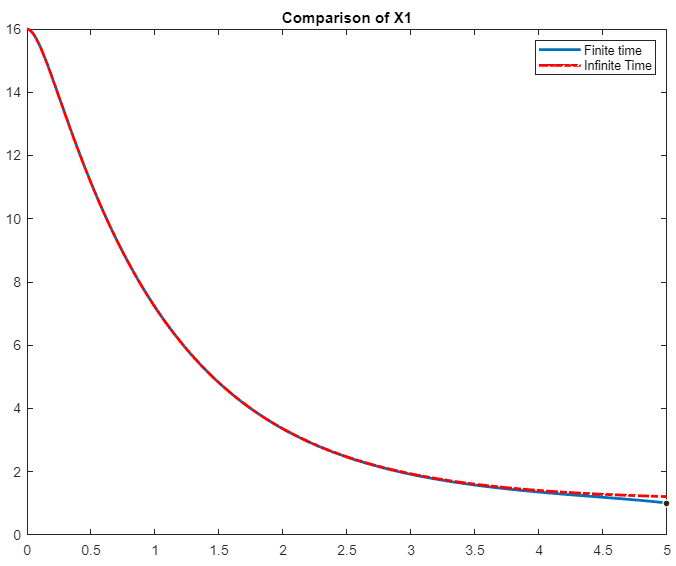
****

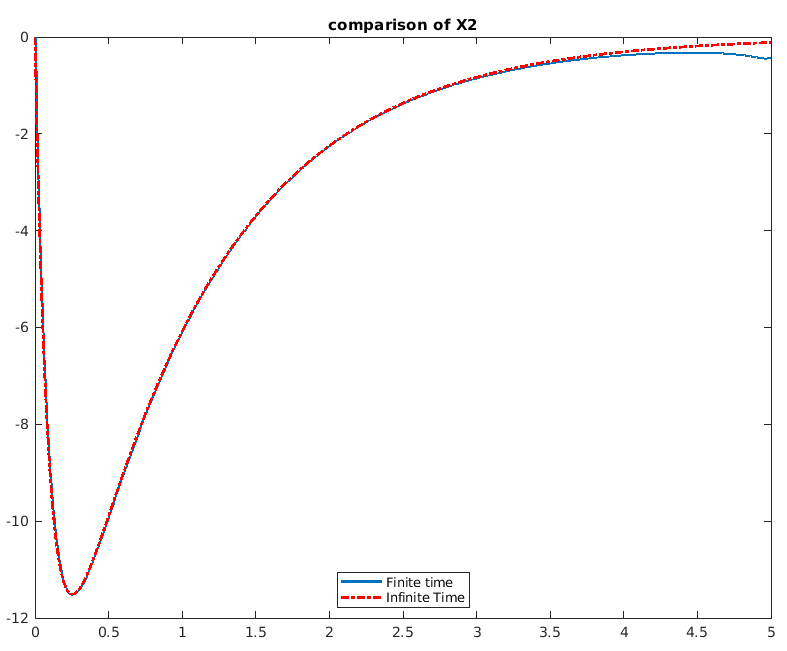
Such high fluctuations in control values may not be desirable so changing the Q and Qf matrices and defining them as:

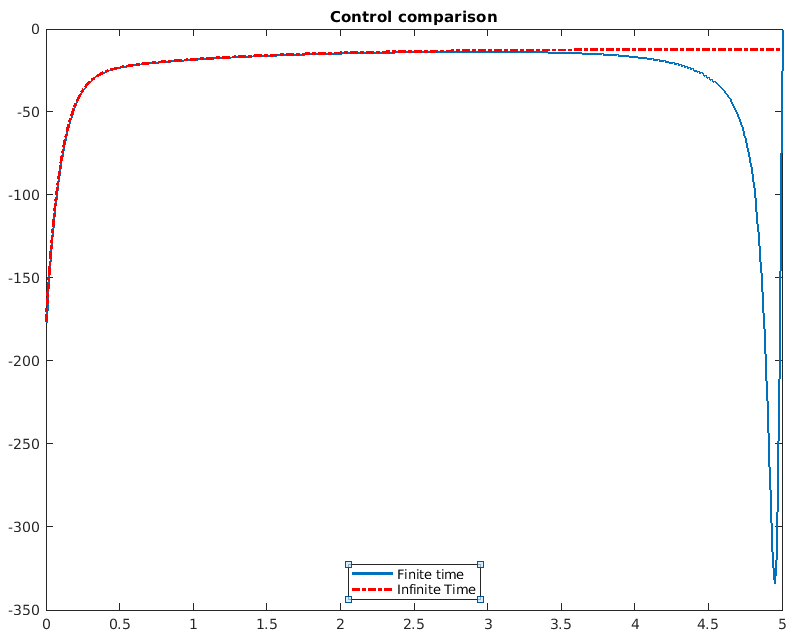
Q = 10\*eye(nx); R = eye(nu);

Qf = [1000000 0; 0 0];

**Now the comparison plots are**

****

****

****

These control values seem more acceptable. However, the Q matrices can be further modified according to the system.

1. **R=sin(t)**

function hw5q3

%system data

A = [0 1; 1 -1];

B = [0; 1];

nx = size(A,2);

nu = size(B,2);

Xs0 = [16; 0]; %initial position at 16

tfinal = 5; %final time 5 sec

dt = 0.01;

tvec = 0:dt:tfinal;

Q = 100\*eye(nx); R = eye(nu);

%checking for controllability

CMAT = ctrb(A,B);

if rank(CMAT) == nx

fprintf('\n This system is controllable \n');

else

return;

end

%checking for detectability

C = sqrt(Q)';

OMAT = obsv(A,C);

if rank(OMAT) == nx

fprintf('\n This system is observable \n');

else

return;

end

fprintf('\n Finite Time LQR \n')

fprintf('\n Cost Function = 1/2 Xf^T\*Qf\*Xf + 1/2 int\_0^tfinal (X^T Q X + U^T R U)dt \n');

Qf = [10000 0; 0 0];

Phist = timeRicSol(A,B,Q,R,Qf,tvec); %solving the DARE and storing solution

X0 = Xs0;

Y(1,:) = X0'; %storing states

for ct = 1:length(tvec)-1

P = reshape(Phist(ct,:), nx, nx);

rc= sin(ct);

[tv, yv] = ode45(@sysde, [tvec(ct) tvec(ct+1)], X0, [], A, B, Q, R, P, rc);

Y(ct+1,:) = yv(end,:);

X0 = yv(end,:);

end

%evaluating control history for plots

for ct = 1:length(tvec)

P = reshape(Phist(ct,:), nx, nx);

Xs = Y(ct,:)';

U(ct,1) = -R\(B'\*P)\*(Xs);

end

%plotting results

figure(1)

plot(tvec, Y(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Finite Horizon');

figure(2)

plot(tvec, Y(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Finite Horizon');

figure(3)

plot(tvec, U, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Finite Horizon');

clc; fprintf('\n Press any key to continue \n'); pause; fprintf('\n Finite vs Infinite Time LQT \n')

Pinf = care(A,B,Q,R); %CARE can also be supplied with more arguments such as R,S,E etc. MATLAB suggests using icare

X0 = Xs0;

[tv, yinf] = ode45(@sysde, tvec, X0, [], A, B, Q, R, Pinf, rc);

%Evaluating control history to generate plots

for ct = 1:length(tvec)

Xinf = yinf(ct,:)';

Uinf(ct,1) = -R\(B'\*Pinf)\*(Xinf);

end

figure(4)

plot(tvec, yinf(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Infinite Horizon');

figure(5)

plot(tvec, yinf(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Infinite Horizon');

figure(6)

plot(tvec, Uinf, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Infinite Horizon');

%comparison of finite vs. infinite time control

figure(7)

plot(tvec, Y(:,1), tvec, yinf(:,1), 'r-.', 'linewidth',2); title('Comparison of X1')

legend('Finite time', 'Infinite Time')

figure(8)

plot(tvec, Y(:,2), tvec, yinf(:,2), 'r-.', 'linewidth',2); title('comparison of X2')

legend('Finite time', 'Infinite Time')

figure(9)

plot(tvec, U, tvec, Uinf,'r-.', 'linewidth',2); title('Control comparison')

legend('Finite time', 'Infinite Time')

end

%local functions

function xd = sysde(t, x, A, B, Q, R, P, rc)

K = R\(B'\*P);

u = -K\*(x - [rc; 0]);

xd = A\*x + B\*u;

end

function Phist = timeRicSol(A,B,Q,R,Qf,tvec)

P0 = Qf; pvec0 = reshape(P0, numel(P0), 1); % solving the matrix differential equation starting at tf and moving backwards

[Thist, Phist] = ode45(@ricDE, tvec, pvec0, [], A, B, Q, R);

Phist = flipud(Phist);

end

function pdvec = ricDE(t, pvec, A, B, Q, R)

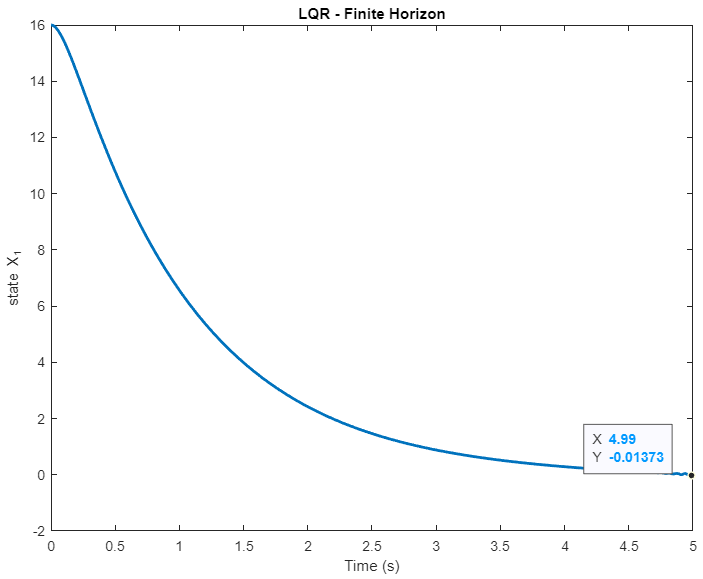
P = reshape(pvec, size(A));

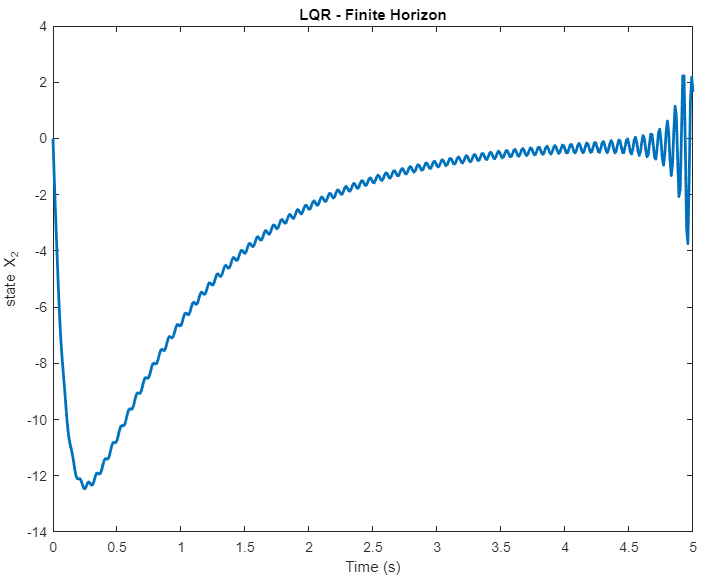
PD = P\*A + A'\*P - P\*B\*(R\(B'\*P)) + Q;

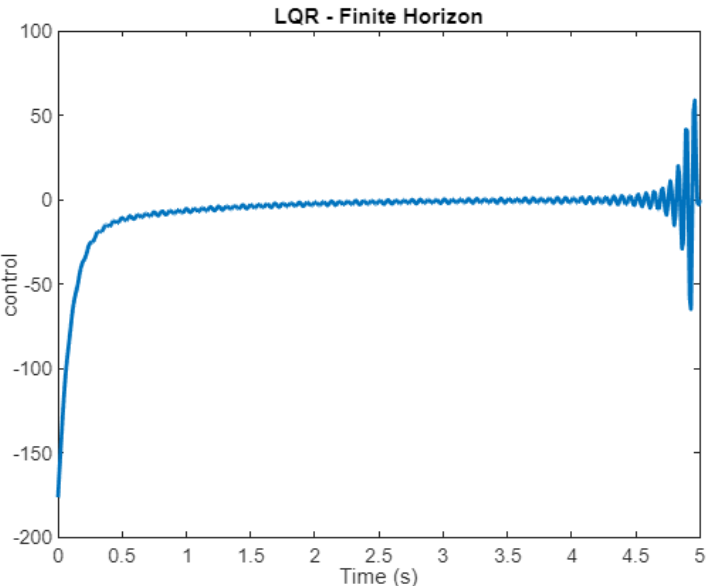
pdvec = reshape(PD, numel(PD), 1);

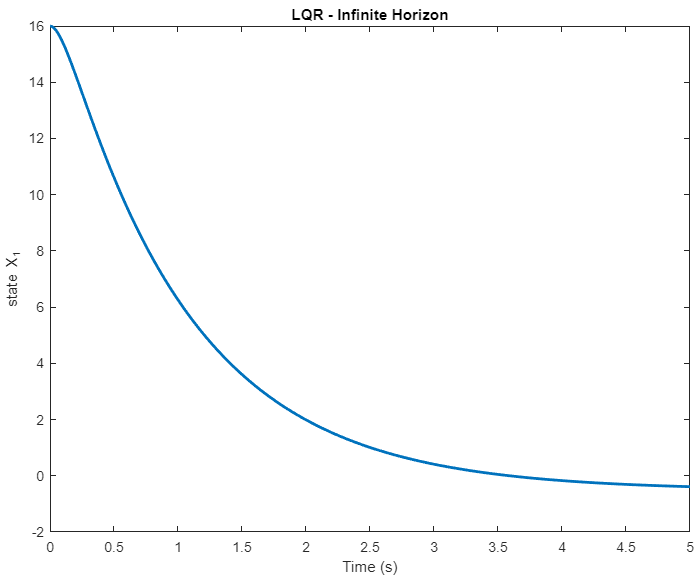
end

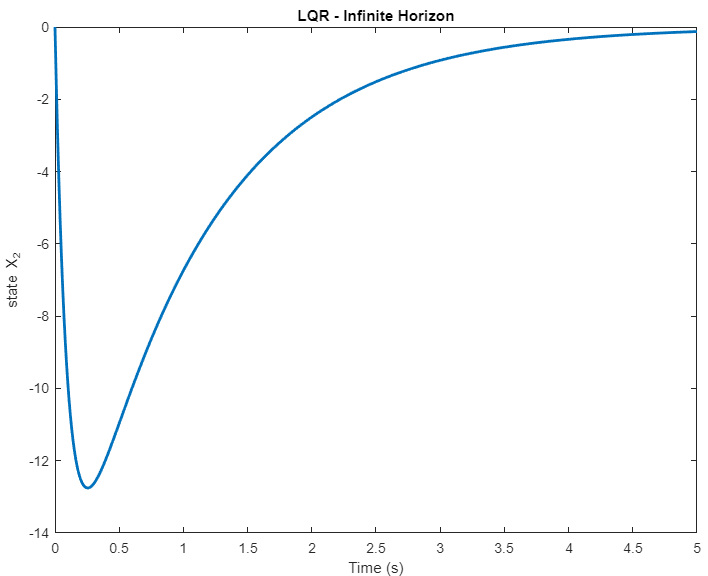
**Plots**

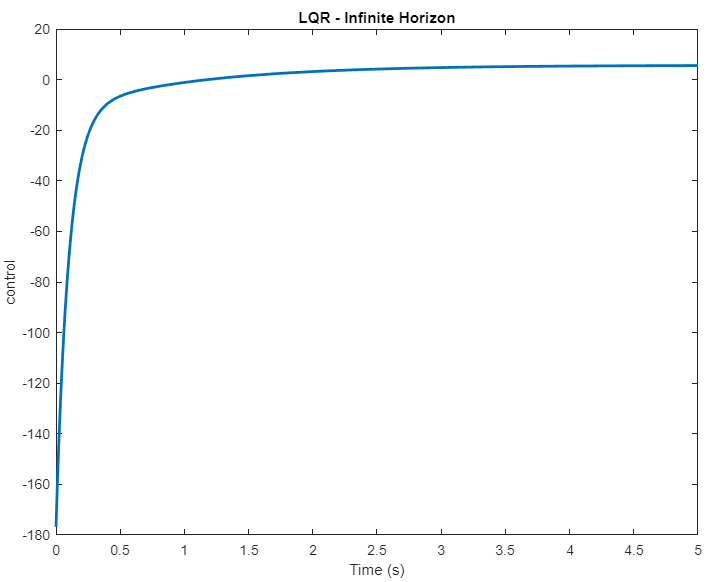
****

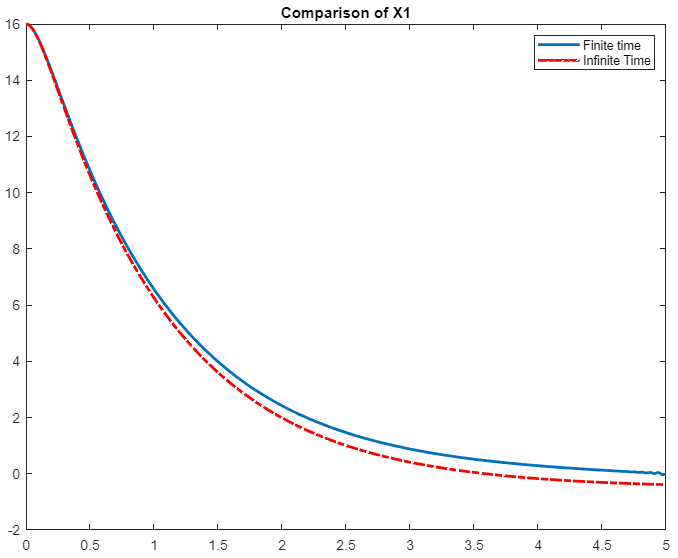
****

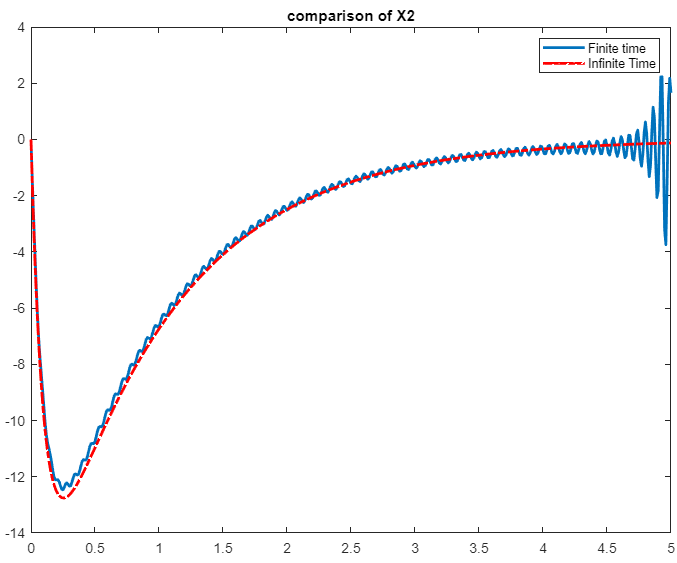
****

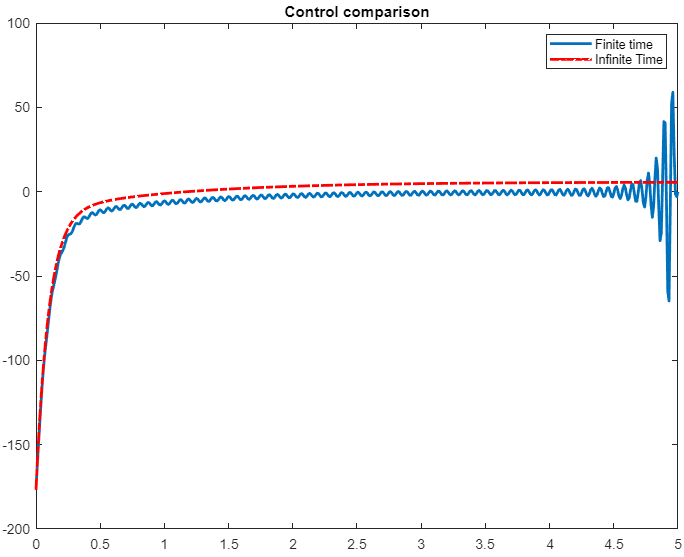
****

****

****

****

****

****

**Part 2**

1. **R=1(step input)**

function hw5q4

%system data

A = [0 1; 1 -1];

B = [0; 1];

nx = size(A,2);

nu = size(B,2);

Xs0 = [32; 0]; %initial position at 16

tfinal = 5; %final time 5 sec

dt = 0.01;

tvec = 0:dt:tfinal;

rc=1;

Q = 100\*eye(nx); R = eye(nu);

%checking for controllability

CMAT = ctrb(A,B);

if rank(CMAT) == nx

fprintf('\n This system is controllable \n');

else

return;

end

%checking for detectability

C = sqrt(Q)';

OMAT = obsv(A,C);

if rank(OMAT) == nx

fprintf('\n This system is observable \n');

else

return;

end

fprintf('\n Finite Time LQR \n')

fprintf('\n Cost Function = 1/2 Xf^T\*Qf\*Xf + 1/2 int\_0^tfinal (X^T Q X + U^T R U)dt \n');

Qf = [10000 0; 0 0];

Phist = timeRicSol(A,B,Q,R,Qf,tvec); %solving the DARE and storing solution

X0 = Xs0;

Y(1,:) = X0'; %storing states

for ct = 1:length(tvec-1)-1 %not evaluating for final point

P = reshape(Phist(ct,:), nx, nx);

[tv, yv] = ode45(@sysde, [tvec(ct) tvec(ct+1)], X0, [], A, B, Q, R, P, rc);

Y(ct+1,:) = yv(end,:);

X0 = yv(end,:);

end

%evaluating control history for plots

for ct = 1:length(tvec-1) %not taking the final point

P = reshape(Phist(ct,:), nx, nx);

Xs = Y(ct,:)';

U(ct,1) = -R\(B'\*P)\*(Xs);

end

for ct=5

Xs=1; %fixing the final state

end

%plotting results

figure(1)

plot(tvec, Y(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Finite Horizon');

figure(2)

plot(tvec, Y(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Finite Horizon');

figure(3)

plot(tvec, U, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Finite Horizon');

clc; fprintf('\n Press any key to continue \n'); pause; fprintf('\n Finite vs Infinite Time LQT \n')

Pinf = care(A,B,Q,R); %CARE can also be supplied with more arguments such as R,S,E etc. MATLAB suggests using icare

X0 = Xs0;

[tv, yinf] = ode45(@sysde, tvec, X0, [], A, B, Q, R, Pinf, rc);

%Evaluating control history to generate plots

for ct = 1:length(tvec)

Xinf = yinf(ct,:)';

Uinf(ct,1) = -R\(B'\*Pinf)\*(Xinf);

end

figure(4)

plot(tvec, yinf(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Infinite Horizon');

figure(5)

plot(tvec, yinf(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Infinite Horizon');

figure(6)

plot(tvec, Uinf, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Infinite Horizon');

%comparison of finite vs. infinite time control

figure(7)

plot(tvec, Y(:,1), tvec, yinf(:,1), 'r-.', 'linewidth',2); title('Comparison of X1')

legend('Finite time', 'Infinite Time')

figure(8)

plot(tvec, Y(:,2), tvec, yinf(:,2), 'r-.', 'linewidth',2); title('comparison of X2')

legend('Finite time', 'Infinite Time')

figure(9)

plot(tvec, U, tvec, Uinf,'r-.', 'linewidth',2); title('Control comparison')

legend('Finite time', 'Infinite Time')

end

%local functions

function xd = sysde(t, x, A, B, Q, R, P, rc)

K = R\(B'\*P);

u = -K\*(x - [rc; 0]);

xd = A\*x + B\*u;

end

function Phist = timeRicSol(A,B,Q,R,Qf,tvec)

P0 = Qf; pvec0 = reshape(P0, numel(P0), 1); % solving the matrix differential equation starting at tf and moving backwards

[Thist, Phist] = ode45(@ricDE, tvec, pvec0, [], A, B, Q, R);

Phist = flipud(Phist);

end

function pdvec = ricDE(t, pvec, A, B, Q, R)

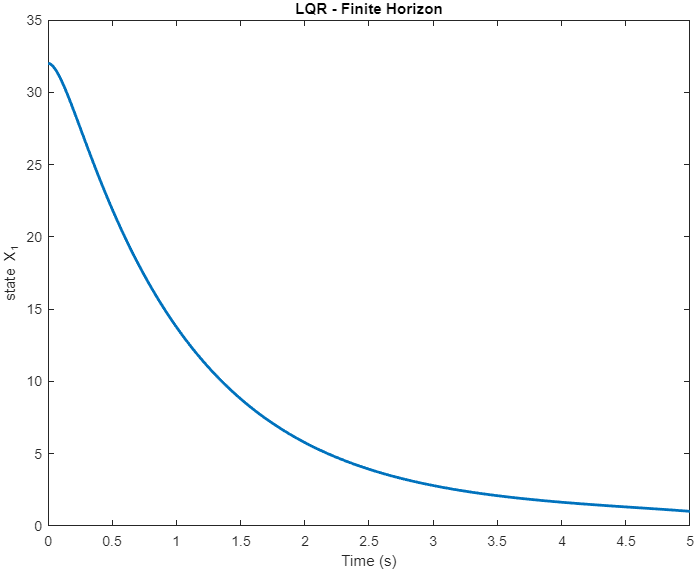
P = reshape(pvec, size(A));

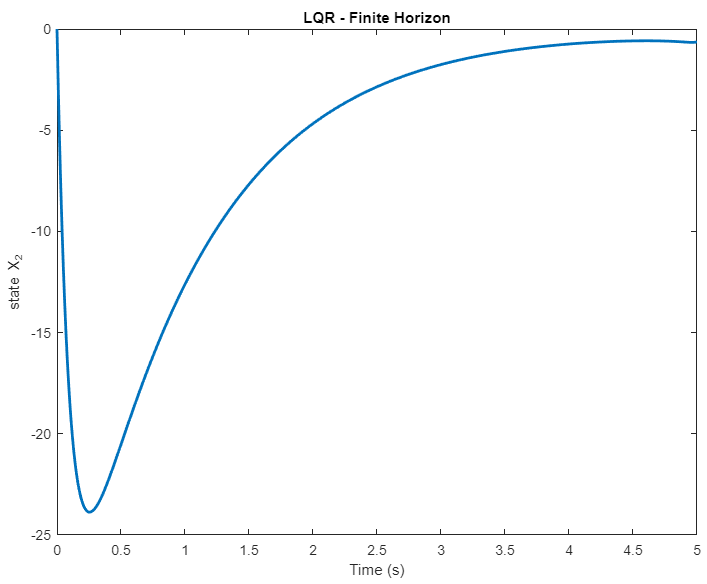
PD = P\*A + A'\*P - P\*B\*(R\(B'\*P)) + Q;

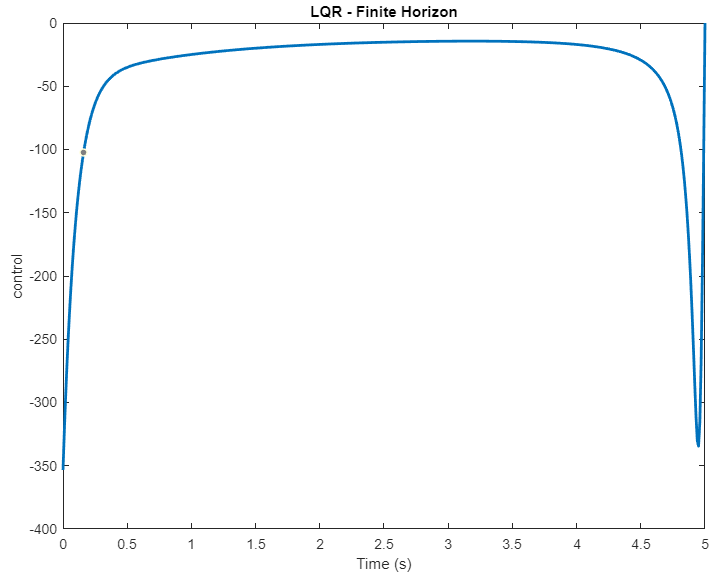
pdvec = reshape(PD, numel(PD), 1);

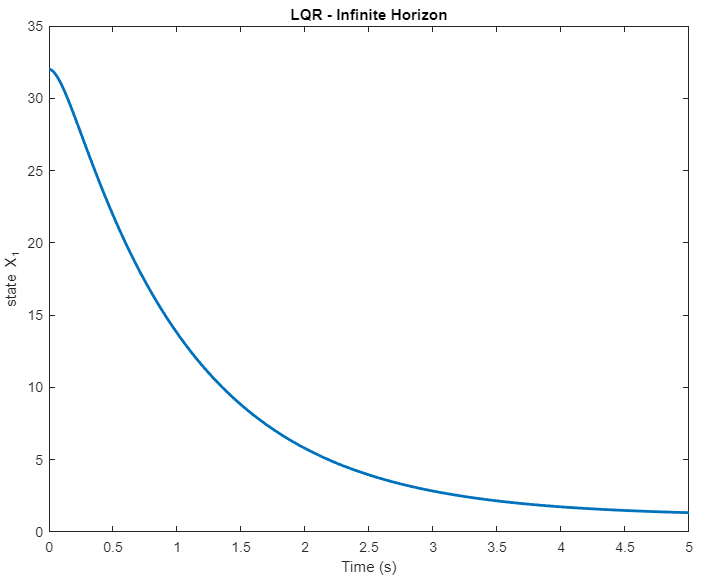
end

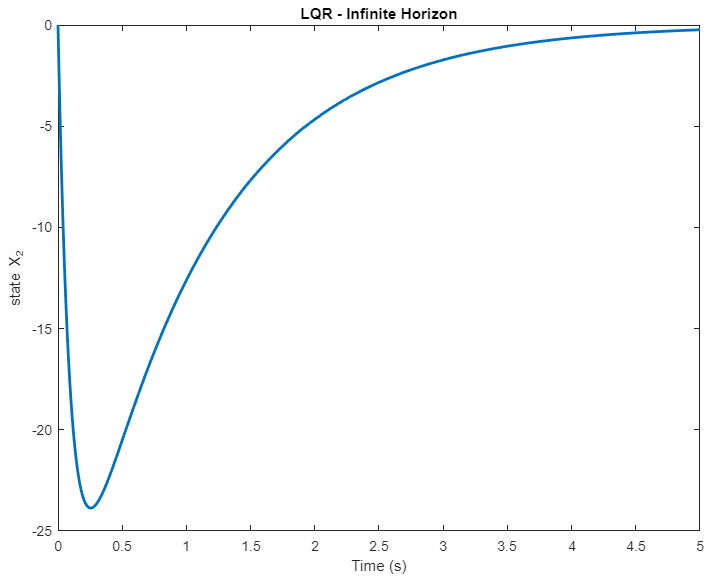
**Plots**

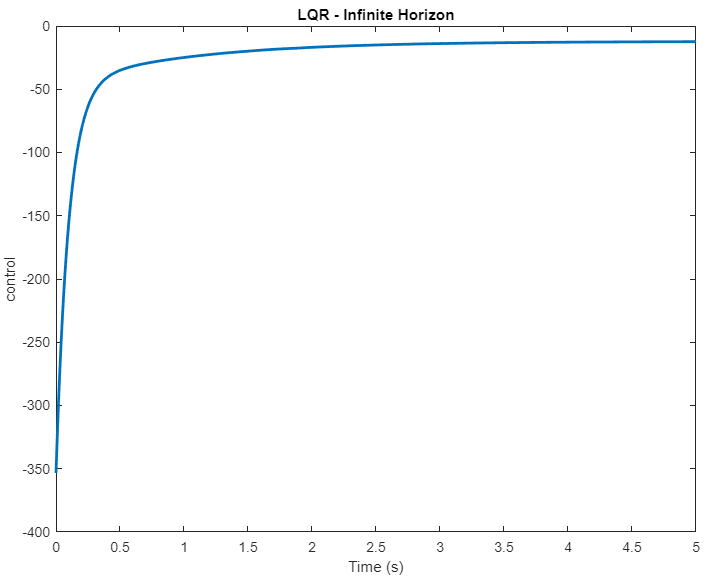
****

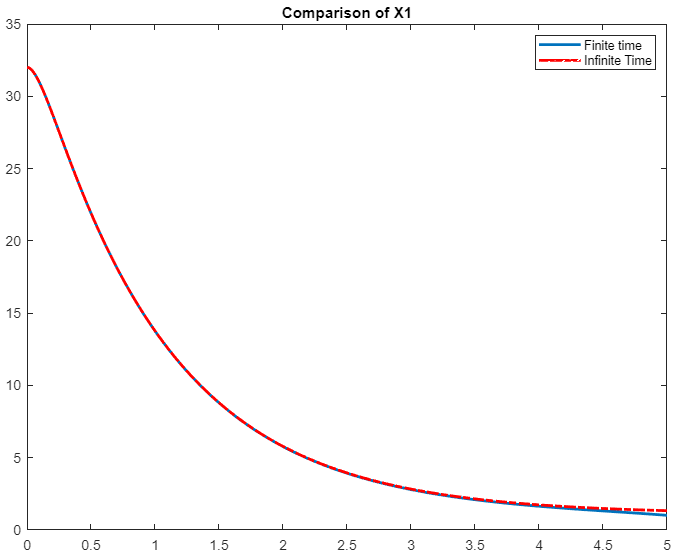
****

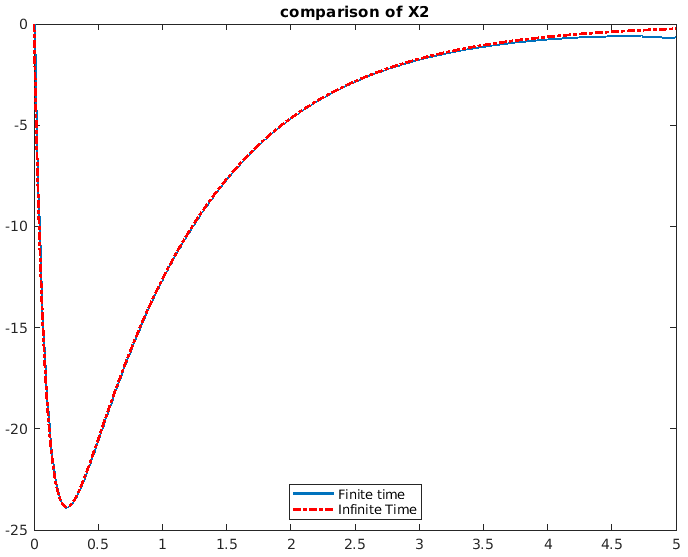
****

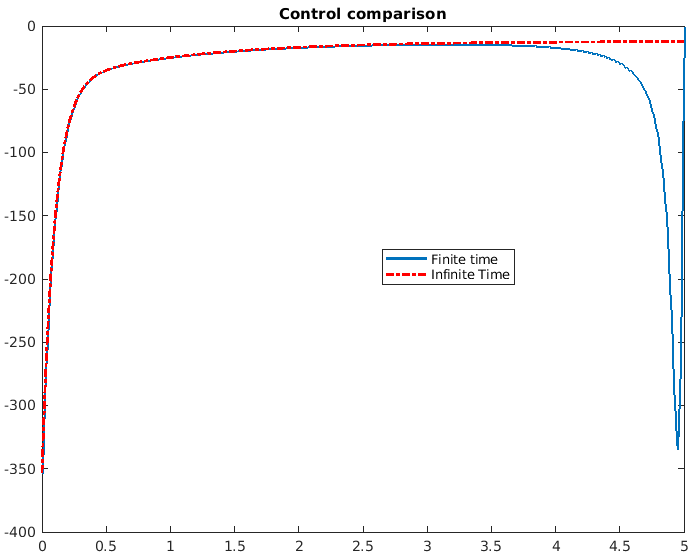
****

****

****

****

****

****

1. **R=sin(t)**

function hw5q4

%system data

A = [0 1; 1 -1];

B = [0; 1];

nx = size(A,2);

nu = size(B,2);

Xs0 = [32; 0]; %initial position at 16

tfinal = 5; %final time 5 sec

dt = 0.01;

tvec = 0:dt:tfinal;

Q = 100\*eye(nx); R = eye(nu);

%checking for controllability

CMAT = ctrb(A,B);

if rank(CMAT) == nx

fprintf('\n This system is controllable \n');

else

return;

end

%checking for detectability

C = sqrt(Q)';

OMAT = obsv(A,C);

if rank(OMAT) == nx

fprintf('\n This system is observable \n');

else

return;

end

fprintf('\n Finite Time LQR \n')

fprintf('\n Cost Function = 1/2 Xf^T\*Qf\*Xf + 1/2 int\_0^tfinal (X^T Q X + U^T R U)dt \n');

Qf = [10000 0; 0 0];

Phist = timeRicSol(A,B,Q,R,Qf,tvec); %solving the DARE and storing solution

X0 = Xs0;

Y(1,:) = X0'; %storing states

for ct = 1:length(tvec)-2

P = reshape(Phist(ct,:), nx, nx);

rc=sin(ct);

[tv, yv] = ode45(@sysde, [tvec(ct) tvec(ct+1)], X0, [], A, B, Q, R, P, rc);

Y(ct+1,:) = yv(end,:);

X0 = yv(end,:);

end

%evaluating control history for plots

for ct = 1:length(tvec)-1

P = reshape(Phist(ct,:), nx, nx);

Xs = Y(ct,:)';

U(ct,1) = -R\(B'\*P)\*(Xs);

end

for ct=(length(tvec)-1):(length(tvec))

Y(ct,:)=-0.95892; %fixing the final state at sin(5)

Xs=-0.95892;

U(ct,1) = -R\(B'\*P)\*(Xs);

end

%plotting results

figure(1)

plot(tvec, Y(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Finite Horizon');

figure(2)

plot(tvec, Y(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Finite Horizon');

figure(3)

plot(tvec, U, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Finite Horizon');

clc; fprintf('\n Press any key to continue \n'); pause; fprintf('\n Finite vs Infinite Time LQT \n')

Pinf = care(A,B,Q,R); %CARE can also be supplied with more arguments such as R,S,E etc. MATLAB suggests using icare

X0 = Xs0;

[tv, yinf] = ode45(@sysde, tvec, X0, [], A, B, Q, R, Pinf, rc);

%Evaluating control history to generate plots

for ct = 1:length(tvec)

Xinf = yinf(ct,:)';

Uinf(ct,1) = -R\(B'\*Pinf)\*(Xinf);

end

figure(4)

plot(tvec, yinf(:,1), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_1'); title('LQR - Infinite Horizon');

figure(5)

plot(tvec, yinf(:,2), 'linewidth',2); xlabel('Time (s)'); ylabel('state X\_2'); title('LQR - Infinite Horizon');

figure(6)

plot(tvec, Uinf, 'linewidth',2), xlabel('Time (s)'); ylabel('control'); title('LQR - Infinite Horizon');

%comparison of finite vs. infinite time control

figure(7)

plot(tvec, Y(:,1), tvec, yinf(:,1), 'r-.', 'linewidth',2); title('Comparison of X1')

legend('Finite time', 'Infinite Time')

figure(8)

plot(tvec, Y(:,2), tvec, yinf(:,2), 'r-.', 'linewidth',2); title('comparison of X2')

legend('Finite time', 'Infinite Time')

figure(9)

plot(tvec, U, tvec, Uinf,'r-.', 'linewidth',2); title('Control comparison')

legend('Finite time', 'Infinite Time')

end

%local functions

function xd = sysde(t, x, A, B, Q, R, P, rc)

K = R\(B'\*P);

u = -K\*(x - [rc; 0]);

xd = A\*x + B\*u;

end

function Phist = timeRicSol(A,B,Q,R,Qf,tvec)

P0 = Qf; pvec0 = reshape(P0, numel(P0), 1); % solving the matrix differential equation starting at tf and moving backwards

[Thist, Phist] = ode45(@ricDE, tvec, pvec0, [], A, B, Q, R);

Phist = flipud(Phist);

end

function pdvec = ricDE(t, pvec, A, B, Q, R)

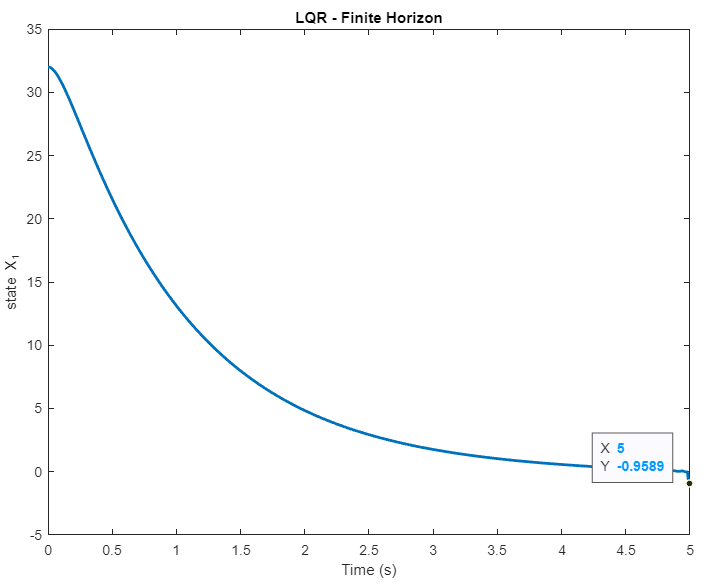
P = reshape(pvec, size(A));

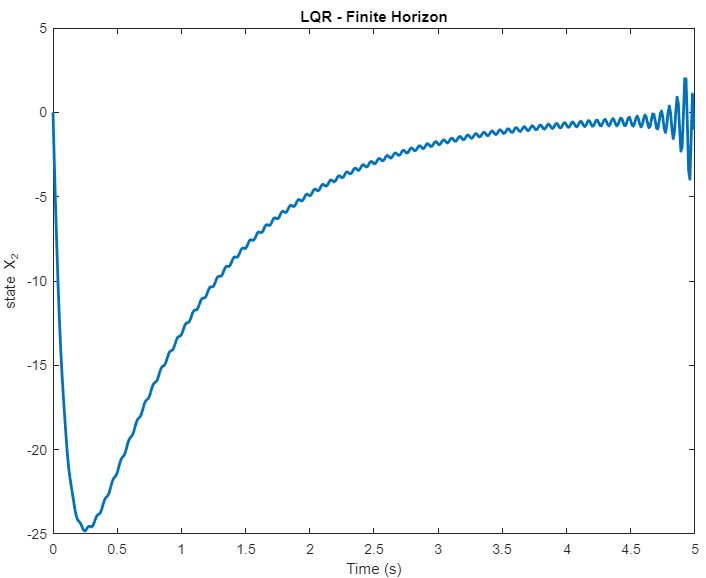
PD = P\*A + A'\*P - P\*B\*(R\(B'\*P)) + Q;

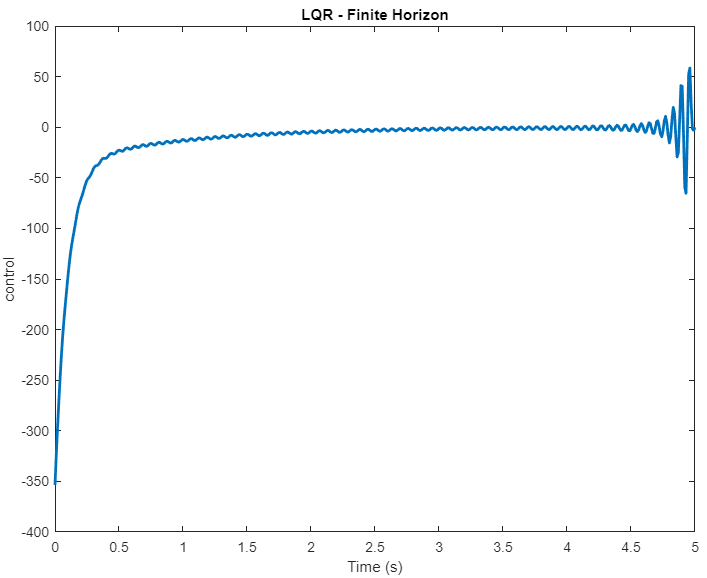
pdvec = reshape(PD, numel(PD), 1);

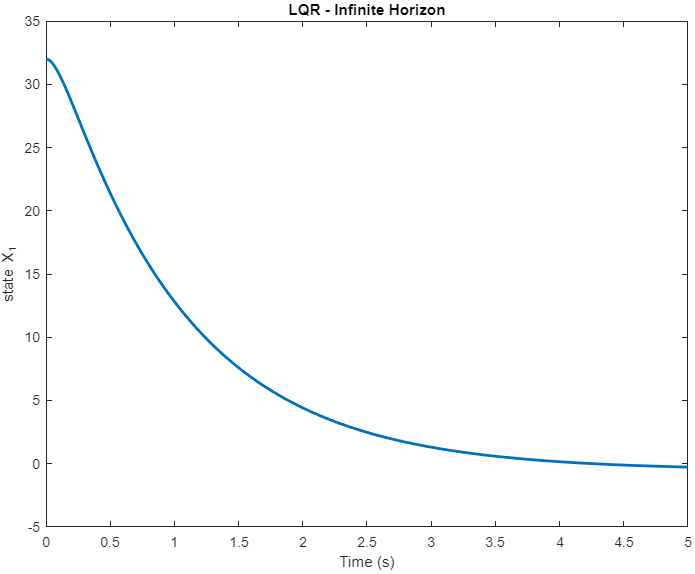
end

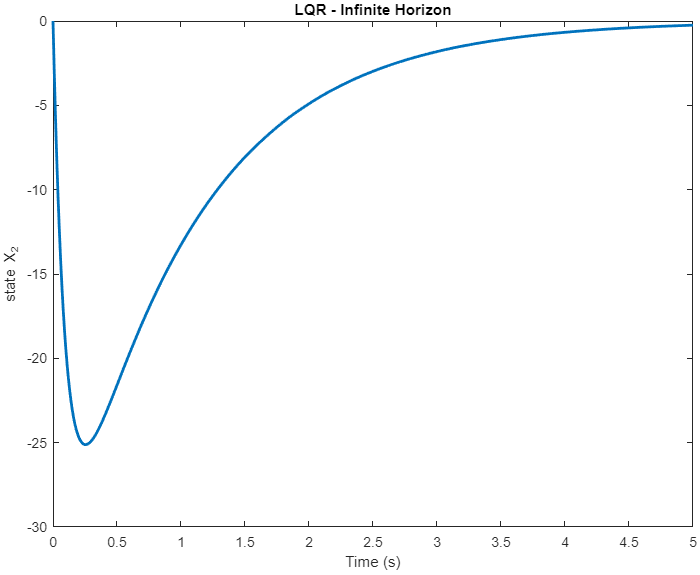
**Plots:**

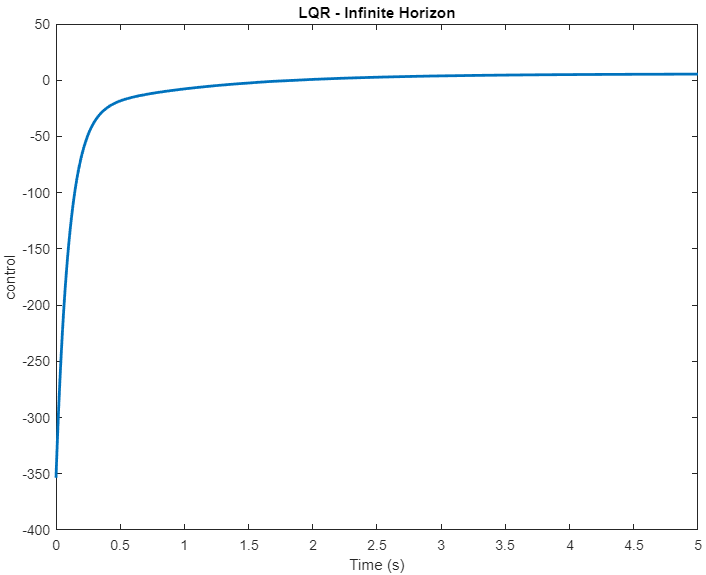
****

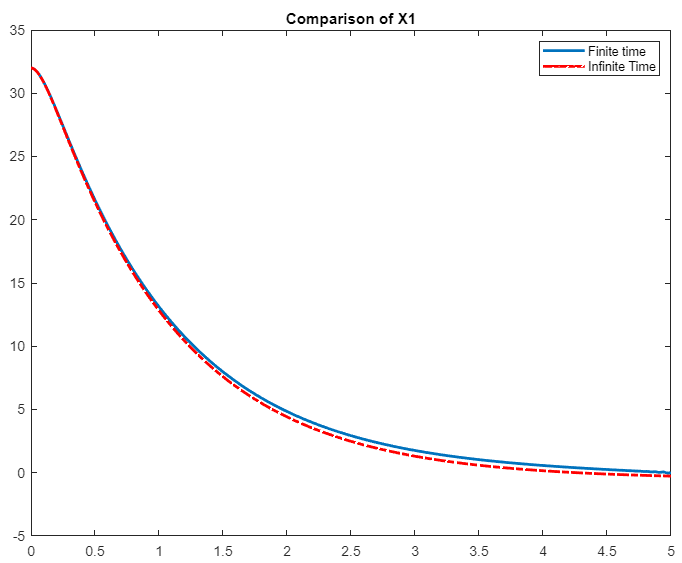
****

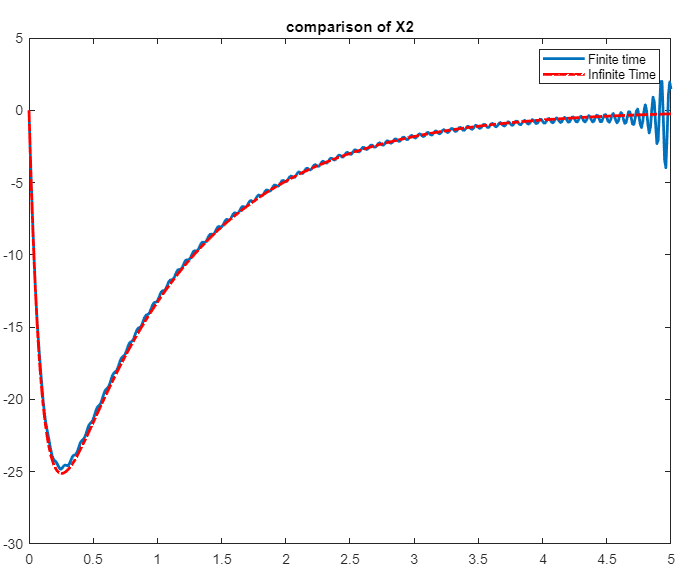
****

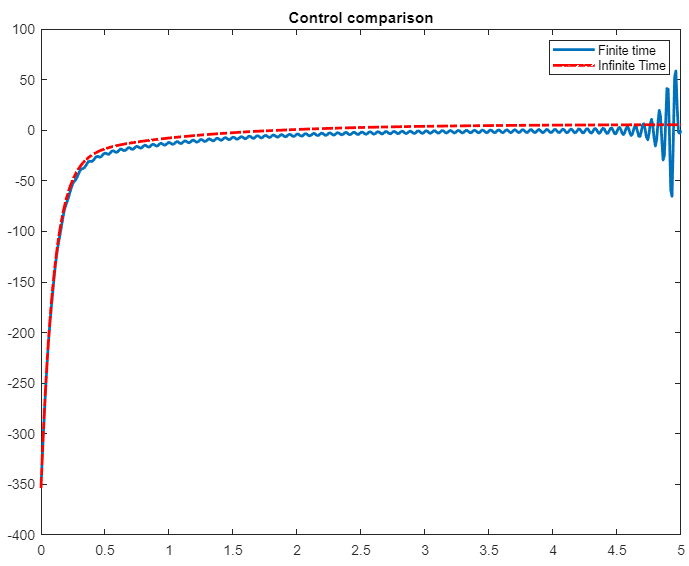
****

****

****

****

****

****