# Example 1 (from Duan & Yu : LMIs in control systems)

## problem

|  |  |
| --- | --- |
| Text, letter  Description automatically generated | Text, letter  Description automatically generated |
| Text  Description automatically generated | |

## Proving the stability using LMI-optimization

|  |  |
| --- | --- |
| Matlab code | Code output |
| %% BEFORE RUNNING THE CODE ADD "YALMIP" AND "SDPT3" libraries  %% yalmip param robust ness anlaysis  clear all,close all,clc;yalmip('clear');  A=diag([-1,-2]);  Ad=eye(2);  nx=2;  eps1=1e-5;  % P=sdpvar(nx,nx,'symmetric');  % S=sdpvar(nx,nx,'symmetric');  P=sdpvar(nx,nx,'diagonal');  S=sdpvar(nx,nx,'diagonal');  F=[];  F=[F;P>=eps1\*eye(nx)];  F=[F;[[P\*A]+A'\*P+S,[P\*Ad];Ad'\*P,-S]<=0\*eye(2\*nx)];  F=[F;0<=vec(P)<=10000];  F=[F;0<=vec(S)<=10000];  ops = sdpsettings('solver','sdpt3');  sol = optimize(F,[],ops);  sol.info  P0=value(P)  eig(P0)  S0=value(S)  eig(S0)  max(eig([[P0\*A]+[P0\*A]'+S0,[P0\*Ad];[P0\*Ad]',-S0])) |  |

## Simulating the system using dde23 Function Phase Plane

|  |  |
| --- | --- |
| Matlab code | Code output |
| %% matlab delayed-diff system simulation  clear all,close all,clc;yalmip('clear');  lags=[0.2];  t\_vec=[0:1e-4:20]';  fig1=figure(1); fig1.Color=[1,1,1];  for ii=1:1:5  sol1=dde23(@dde\_func,lags,@x\_history,t\_vec);  x\_vec = deval(sol1,t\_vec);  x1\_trajectory=x\_vec(1,:);  x2\_trajectory=x\_vec(2,:);  plot(x1\_trajectory,x2\_trajectory,...  'LineStyle','-',...  'LineWidth',[3],...  'Color','r'); hold on;  xlabel('x1'); ylabel('x2');  % plot(t\_vec,x1\_trajectory,'LineStyle','-','LineWidth',[3],'Color','r'); hold on;  % plot(t\_vec,x2\_trajectory,'LineStyle','-','LineWidth',[3],'Color','b');  % legend('x1','x2');  end  function xdot=dde\_func(t,x,x\_delayed)  x1=x(1);  x2=x(2);  xdot=zeros(2,1);  x1\_delayed=x\_delayed(1);  x2\_delayed=x\_delayed(2);  A=diag([-1,-2]);  Ad=eye(2);  xdot=A\*[x1;x2]+Ad\*[x1\_delayed;x2\_delayed];  end  function x=x\_history(t)  % x=ones(2,1);  x=0.1\*ones(2,1)+rand(2,1)\*0.3;  end |  |

## Simulating the system using dde23 Function State Signals

|  |  |
| --- | --- |
| Matlab code | Code output |
| %% matlab delayed-diff system simulation  clear all,close all,clc;yalmip('clear');  lags=[0.2];  t\_vec=[0:1e-4:20]';  fig1=figure(1); fig1.Color=[1,1,1];  for ii=1:1:5  sol1=dde23(@dde\_func,lags,@x\_history,t\_vec);  x\_vec = deval(sol1,t\_vec);  x1\_trajectory=x\_vec(1,:);  x2\_trajectory=x\_vec(2,:);  % plot(x1\_trajectory,x2\_trajectory,...  % 'LineStyle','-',...  % 'LineWidth',[3],...  % 'Color','r'); hold on; xlabel('x1'); ylabel('x2');  plot(t\_vec,x1\_trajectory,'LineStyle','-','LineWidth',[3],'Color','r'); hold on;  plot(t\_vec,x2\_trajectory,'LineStyle','-','LineWidth',[3],'Color','b');  legend('x1','x2'); xlabel('time'); ylabel('x(t)');  end  function xdot=dde\_func(t,x,x\_delayed)  x1=x(1);  x2=x(2);  xdot=zeros(2,1);  x1\_delayed=x\_delayed(1);  x2\_delayed=x\_delayed(2);  A=diag([-1,-2]);  Ad=eye(2);  xdot=A\*[x1;x2]+Ad\*[x1\_delayed;x2\_delayed];  end  function x=x\_history(t)  % x=ones(2,1);  x=0.1\*ones(2,1)+rand(2,1)\*0.3;  end |  |

# Example 2 (from Duan & Yu : LMIs in control systems)

## problem

|  |  |
| --- | --- |
| Text, letter  Description automatically generated |  |
|  | |

## Proving the stability using LMI-optimization

|  |  |
| --- | --- |
| Matlab code | Code output |
| %% BEFORE RUNNING THE CODE ADD "YALMIP" AND "SDPT3" libraries  %% yalmip analysis  % set(findall(gcf,'type','line'),'linewidth',[1]);  clear all,close all,clc;yalmip('clear');  A=[-2,0,1;0,-3,0;1,0,-2];  Ad=[-1,1,1;2,-1,1;0,0,-1];  nx=3; % state-dimension  dmax=0.3; % max delay [max value 0.3 seconds]  eps1=1e-3; % epsilon for numerical accuracy  % define the decision variables  X=sdpvar(nx,nx,'symmetric');  beta=sdpvar(1,1,'symmetric');  % enter the constraints  F=[];  F=[F;X>=eps1\*eye(nx)];  M\_11=[(A+Ad)\*X]+[(A+Ad)\*X]'+dmax\*Ad\*Ad';  M\_12=dmax\*X\*A';  M\_13=dmax\*X\*Ad';  M\_21=dmax\*A\*X;  M\_22=-dmax\*beta\*eye(nx);  M\_23=zeros(nx);  M\_31=dmax\*Ad\*X;  M\_32=zeros(nx);  M\_33=-dmax\*(1-beta)\*eye(nx);  M=[M\_11,M\_12,M\_13;M\_21,M\_22,M\_23;M\_31,M\_32,M\_33];  F=[F;M<=-eps1\*eye(3\*nx)];  F=[F;-1e2<=vec(X)<=1e2];  F=[F;eps1<=beta<=1-eps1];  % solve the opt-problem  ops = sdpsettings('solver','sdpt3');  sol = optimize(F,[],ops);  sol.info  % get the decision variables  X=value(X);P=inv(X);beta=value(beta) |  |

## Additional check

|  |  |
| --- | --- |
| Matlab code | Code output |
| %% check if there is any error  X=value(X)  P=inv(X)  beta=value(beta)  eig(P)  M\_11=[(A+Ad)\*X]+[(A+Ad)\*X]'+dmax\*Ad\*Ad';  M\_12=dmax\*X\*A';  M\_13=dmax\*X\*Ad';  M\_21=dmax\*A\*X;  M\_22=-dmax\*beta\*eye(nx);  M\_23=zeros(nx);  M\_31=dmax\*Ad\*X;  M\_32=zeros(nx);  M\_33=-dmax\*(1-beta)\*eye(nx);  M=[M\_11,M\_12,M\_13;M\_21,M\_22,M\_23;M\_31,M\_32,M\_33];  eig(P)  eig(M) | All eigenvalues of P are positive therefore it is a pos-def-matrix  All eigenvalues of M are negative therefore it is a neg-def-matrix |

## Simulating the system using dde23 Function Phase Plane

|  |  |
| --- | --- |
| Matlab code | Code output |
| %% matlab delayed-diff system simulation  clear all,close all,clc;yalmip('clear');  lags=[0.2];  t\_vec=[0:1e-4:20]';  fig1=figure(1);  fig1.Color=[1,1,1];  for ii=1:1:10  sol1=dde23(@dde\_func,lags,@x\_history,t\_vec);  x\_vec = deval(sol1,t\_vec);  x1\_trajectory=x\_vec(1,:);  x2\_trajectory=x\_vec(2,:);  x3\_trajectory=x\_vec(3,:);  plot3(x1\_trajectory,x2\_trajectory,x3\_trajectory,...  'LineStyle','-',...  'LineWidth',[1],...  'Color','r'); hold on;  % plot(t\_vec,x1\_trajectory,'LineStyle','-','LineWidth',[1],'Color','r'); hold on;  % plot(t\_vec,x2\_trajectory,'LineStyle','-','LineWidth',[1],'Color','g');  % plot(t\_vec,x3\_trajectory,'LineStyle','-','LineWidth',[1],'Color','b');  % legend('x1','x2','x3');  end  function xdot=dde\_func(t,x,x\_delayed)  x1=x(1);  x2=x(2);  x3=x(3);  xdot=zeros(3,1);  x1\_delayed=x\_delayed(1);  x2\_delayed=x\_delayed(2);  x3\_delayed=x\_delayed(3);  A=[-2,0,1;0,-3,0;1,0,-2];  Ad=[-1,1,1;2,-1,1;0,0,-1];  xdot=A\*[x1;x2;x3]+Ad\*[x1\_delayed;x2\_delayed;x3\_delayed];  end  function x=x\_history(t)  % x=ones(2,1);  x=0.1\*ones(3,1)+rand(3,1)\*0.3;  end | Notice that all the states converge to the origin regardless of the initial conditions. |

## Simulating the system using dde23 Function State Signals

|  |  |
| --- | --- |
| Matlab code | Code output |
| %% matlab delayed-diff system simulation  clear all,close all,clc;yalmip('clear');  lags=[0.2];  t\_vec=[0:1e-4:20]';  fig1=figure(1);  fig1.Color=[1,1,1];  for ii=1:1:10  sol1=dde23(@dde\_func,lags,@x\_history,t\_vec);  x\_vec = deval(sol1,t\_vec);  x1\_trajectory=x\_vec(1,:);  x2\_trajectory=x\_vec(2,:);  x3\_trajectory=x\_vec(3,:);  % plot3(x1\_trajectory,x2\_trajectory,x3\_trajectory,...  % 'LineStyle','-',...  % 'LineWidth',[1],...  % 'Color','r'); hold on;  plot(t\_vec,x1\_trajectory,'LineStyle','-','LineWidth',[1],'Color','r'); hold on;  plot(t\_vec,x2\_trajectory,'LineStyle','-','LineWidth',[1],'Color','g');  plot(t\_vec,x3\_trajectory,'LineStyle','-','LineWidth',[1],'Color','b');  legend('x1','x2','x3');xlabel('time');ylabel('x(t)');  end  function xdot=dde\_func(t,x,x\_delayed)  x1=x(1);  x2=x(2);  x3=x(3);  xdot=zeros(3,1);  x1\_delayed=x\_delayed(1);  x2\_delayed=x\_delayed(2);  x3\_delayed=x\_delayed(3);  A=[-2,0,1;0,-3,0;1,0,-2];  Ad=[-1,1,1;2,-1,1;0,0,-1];  xdot=A\*[x1;x2;x3]+Ad\*[x1\_delayed;x2\_delayed;x3\_delayed];  end  function x=x\_history(t)  % x=ones(2,1);  x=0.1\*ones(3,1)+rand(3,1)\*0.3;  end | Notice that all the states converge to the origin regardless of the initial conditions. |