SOS related problems [YALMIP-MATLAB]

Problem 1

Determine if the given polynomial is SOS?

|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 1: is this poly sos ?  clear all,close all,clc;  x1 = sdpvar(1,1);  x2 = sdpvar(1,1);  p = (1+x1)^4 + (1-x2)^2;  class(p)  %%  v = monolist([x1 x2],degree(p)/2);  % v = monolist([x1 x2],2,1);  sdisplay(v)  Q = sdpvar(length(v));  p\_sos = v'\*Q\*v;  %%  F = [coefficients(p-p\_sos,[x1 x2]) == 0, Q >= 0];  optimize(F)  Q=value(Q)  v\_sym=sym(v)  clearvars -except Q v\_sym  syms x1 x2 real  v\_sym'\*Q\*v\_sym  p = (1+x1)^4 + (1-x2)^2;  [C,T] = coeffs(p-v\_sym'\*Q\*v\_sym)  double(C)  eig(Q) |

Problem 2

Find the minimum value of the given polynomial [six hump camel]

A picture containing diagram

Description automatically generated

Text

Description automatically generated with medium confidence Text

Description automatically generated with medium confidence

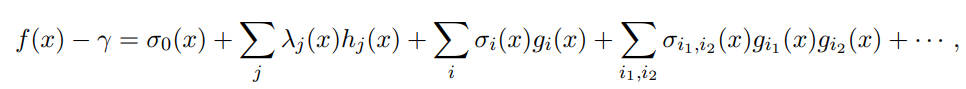
|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 2: minimize a poly [UNCONSTRAINED]  clear all,close all,clc;yalmip('clear');  x1 = sdpvar(1,1);  x2 = sdpvar(1,1);  p = (4-2.1\*x1^2+(x1^4)/3)\*x1^2+x1\*x2+(-4+4\*x2^2)\*x2^2;  %%  v = monolist([x1 x2],degree(p)/2);  Q = sdpvar(length(v));  p\_sos = v'\*Q\*v;  %%  sdpvar t  F = [coefficients((p-t)-p\_sos,[x1 x2]) == 0, Q >= 0];  optimize(F,-t)  sdisplay(p\_sos)  value(t) |

Problem 3

Finding the minimum value of the given polynomial [for the given set]

A picture containing text, clock

Description automatically generated



Problem: [sostools-400 manual pdf-page-40]

Compute the following

|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 3: minimize a poly [CONSTRAINED]  % sostools-400-manual-pdf-page-40  clear all,close all,clc;yalmip('clear');  x1 = sdpvar(1,1);  x2 = sdpvar(1,1);  f = x1+x2;  g1=x1;  g2=x2-0.5;  h1=x1^2+x2^2-1;  h2=x2-x1^2-0.5;  deg1=6;  % sigma\_0  m\_sigma\_0 = monolist([x1 x2],deg1); sdisplay(m\_sigma\_0)  Q\_sigma\_0 = sdpvar(length(m\_sigma\_0));  sigma\_0 = m\_sigma\_0'\*Q\_sigma\_0\*m\_sigma\_0; sdisplay(sigma\_0)  % sigma\_1  m\_sigma\_1 = monolist([x1 x2],deg1);  Q\_sigma\_1 = sdpvar(length(m\_sigma\_1));  sigma\_1 = m\_sigma\_1'\*Q\_sigma\_1\*m\_sigma\_1;  % sigma\_2  m\_sigma\_2 = monolist([x1 x2],deg1);  Q\_sigma\_2 = sdpvar(length(m\_sigma\_2));  sigma\_2 = m\_sigma\_2'\*Q\_sigma\_2\*m\_sigma\_2;  % sigma\_1\_2  m\_sigma\_1\_2 = monolist([x1 x2],deg1);  Q\_sigma\_1\_2 = sdpvar(length(m\_sigma\_1\_2));  sigma\_1\_2 = m\_sigma\_1\_2'\*Q\_sigma\_1\_2\*m\_sigma\_1\_2;  % lambda\_1  [lambda\_1,coeff\_1,mono\_list\_1] = polynomial([x1;x2],deg1); % sdisplay(lambda\_1)  % lambda\_2  [lambda\_2,coeff\_2,mono\_list\_2] = polynomial([x1;x2],deg1);  %  sdpvar t  F = [coefficients((f-t)-1\*[sigma\_0+lambda\_1\*h1+lambda\_2\*h2+sigma\_1\*g1+sigma\_2\*g2+sigma\_1\_2\*g1\*g2],[x1 x2]) == 0,...  Q\_sigma\_0 >= 0, Q\_sigma\_1 >= 0, Q\_sigma\_2 >= 0, Q\_sigma\_1\_2 >= 0];  options = sdpsettings('solver','bisection','bisection.solver','sdpt3');  % diagnostics = bisection(Constraint,Objective,options)  diagnostics = bisection([F],[-t],options)  value(t) |

Problem 4:

Finding a Lyapunov Function [to prove global asym stab]

Linear Programming

SOS programming

Problem 4: find a Lyapunov fcn to prove global asym stability

|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 4: find a Lyap [UNCONSTRAINED]  clear all,close all,clc;yalmip('clear');  x1 = sdpvar(1,1); x2 = sdpvar(1,1); % state-variables "indeterminates"  f1 = -x1-10\*x2^2; f2 = -2\*x2; % the dynamics  deg1=4; eps1=1e-3;  eps1\_poly=eps1\*(x1^2+x2^2); % small-sos-poly  % V  [V,coeff\_1,mono\_list\_1] = polynomial([x1;x2],deg1); sdisplay(mono\_list\_1)  % Vdot  mVdot=-jacobian(V,[x1;x2])\*[f1;f2];  options = sdpsettings('solver','sdpt3');  % constraints "V>0","-Vdot>0","V(0)=0","Vdot(0)=0"  F=[sos(V-eps1\_poly),sos(mVdot-eps1\_poly),-10<=coeff\_1<=20,coeff\_1(1)==0];  [sol,m,Q,residuals,everything] = solvesos(F,[],options,[coeff\_1]);  % positive-definiteness check  eig(value(Q{1}))  eig(value(Q{2}))  % get the matrices  Q1=Q{1}; Q1=Q1.\*(abs(Q1)>1e-3)  Q2=Q{2}; Q2=Q2.\*(abs(Q2)>1e-3)  % get the monomials as "sym"  m1=sym(m{1})  m2=sym(m{2})  clearvars -except Q1 m1 Q2 m2  syms x1 x2 real  V=m1'\*Q1\*m1  V=simplify(V)  vpa(V,4)  [C,T] = coeffs(V)  double(C)  f1 = -x1-10\*x2^2;  f2 = -2\*x2;  Vdot=jacobian(V,[x1;x2])\*[f1;f2]  Vdot=expand(Vdot)  Vdot=simplify(Vdot)  vpa(Vdot,4)  [C,T] = coeffs(Vdot)  double(C) |

Problem 5: find a Lyapunov fcn to prove global asym stability

|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 5: find a Lyap [UNCONSTRAINED]  clear all,close all,clc;yalmip('clear');  x1 = sdpvar(1,1); x2 = sdpvar(1,1); x3 = sdpvar(1,1); % state-variables "indeterminates"  f1 = -x1-10\*x2^2; f2 = -2\*x2; f3 = -2\*x3; % the dynamics  deg1=4; eps1=1e-3;  eps1\_poly=eps1\*(x1^2+x2^2+x3^2); % small-sos-poly  % V  [V,coeff\_1,mono\_list\_1] = polynomial([x1;x2;x3],deg1); sdisplay(mono\_list\_1)  % Vdot  mVdot=-jacobian(V,[x1;x2;x3])\*[f1;f2;f3];  options = sdpsettings('solver','sdpt3');  % constraints "V>0","-Vdot>0","V(0)=0","Vdot(0)=0"  F=[sos(V-eps1\_poly),sos(mVdot-eps1\_poly),-10<=coeff\_1<=20,coeff\_1(1)==0];  [sol,m,Q,residuals,everything] = solvesos(F,[],options,[coeff\_1]);  % positive-definiteness check  if ~all(real(eig(value(Q{1})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{2})))>=0) error('negativity detected->BAD!'); end  % get the matrices  Q1=Q{1}; Q1=Q1.\*(abs(Q1)>1e-3)  Q2=Q{2}; Q2=Q2.\*(abs(Q2)>1e-3)  % get the monomials as "sym"  m1=sym(m{1})  m2=sym(m{2})  clearvars -except Q1 m1 Q2 m2  syms x1 x2 x3 real  V=m1'\*Q1\*m1  V=simplify(V)  vpa(V,4) |

Problem 6: find a Lyapunov fcn to prove global asym stability

|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 6: find a Lyap [UNCONSTRAINED]  clear all,close all,clc;yalmip('clear');  x1 = sdpvar(1,1); x2 = sdpvar(1,1); x3 = sdpvar(1,1); % state-variables "indeterminates"  f1 = -x1-10\*x2^2; f2 = -2\*x2; f3 = -2\*x3; % the dynamics  deg1=4; eps1=1e-3;  eps1\_poly=eps1\*(x1^2+x2^2+x3^2); % small-sos-poly  % V  [V,coeff\_1,mono\_list\_1] = polynomial([x1;x2;x3],deg1); sdisplay(mono\_list\_1)  % Vdot  mVdot=-jacobian(V,[x1;x2;x3])\*[f1;f2;f3];  options = sdpsettings('solver','sdpt3');  % constraints "V>0","-Vdot>0","V(0)=0","Vdot(0)=0"  % F=[sos(V-eps1\_poly),sos(mVdot-eps1\_poly),-10<=coeff\_1<=20,coeff\_1(1)==0];  F=[sos(V-eps1\_poly),sos(mVdot),-10<=coeff\_1<=20,replace(V,[x1,x2,x3],[0,0,0])==0,replace(V,[x1,x2,x3],[1,0,0])==1];  [sol,m,Q,residuals,everything] = solvesos(F,[],options,[coeff\_1]); sdisplay(sosd(F))  % positive-definiteness check  if ~all(real(eig(value(Q{1})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{2})))>=0) error('negativity detected->BAD!'); end  % get the matrices  Q1=Q{1}; Q1=Q1.\*(abs(Q1)>1e-3)  Q2=Q{2}; Q2=Q2.\*(abs(Q2)>1e-3)  % get the monomials as "sym"  m1=sym(m{1})  m2=sym(m{2})  clearvars -except Q1 m1 Q2 m2  syms x1 x2 x3 real  V=m1'\*Q1\*m1  V=simplify(V)  vpa(V,4) |

Problem 7: find a Lyapunov fcn to prove global asym stability

With “replace-function”

|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 7: find a Lyap [CONSTRAINED] [local stability]  clear all,close all,clc;yalmip('clear');  x1 = sdpvar(1,1); % state-variables "indeterminates"  f1 = x1\*(x1-1)\*(x1+1); % the dynamics  deg1=4; eps1=1e-3;  eps1\_poly=eps1\*(x1^2); % small-sos-poly  % V  [V,coeff\_V,mono\_list\_1] = polynomial([x1],deg1); sdisplay(mono\_list\_1)  % sigma\_1  [sigma\_1,coeff\_sigma\_1,mono\_list\_1] = polynomial([x1],deg1);  % sigma\_2  [sigma\_2,coeff\_sigma\_2,mono\_list\_1] = polynomial([x1],deg1);  % Vdot  mVdot=-jacobian(V,[x1])\*[f1];  options = sdpsettings('solver','sdpt3');  % constraints "V>0","-Vdot>0","V(0)=0","Vdot(0)=0"  F=[sos(V-sigma\_1\*[.3^2-x1^2]-eps1\_poly),...  sos(mVdot-sigma\_2\*[.3^2-x1^2]-eps1\_poly),...  sos(sigma\_1),sos(sigma\_2),...  -10<=coeff\_V<=20,...  -10<=coeff\_sigma\_1<=20,...  -10<=coeff\_sigma\_2<=20,...  replace(V,[x1],[0])==0,...  replace(V,[x1],[1])==1];  [sol,m,Q,residuals,everything] = solvesos(F,[],options,[coeff\_V,coeff\_sigma\_1,coeff\_sigma\_2]);  % positive-definiteness check  if ~all(real(eig(value(Q{1})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{2})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{3})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{4})))>=0) error('negativity detected->BAD!'); end  % get the matrices  Q1=Q{1}; Q1=Q1.\*(abs(Q1)>1e-3)  Q2=Q{2}; Q2=Q2.\*(abs(Q2)>1e-3)  % get the monomials as "sym"  m1=sym(m{1})  m2=sym(m{2})  clearvars -except Q1 m1 Q2 m2  syms x1 real  V=m1'\*Q1\*m1  V=simplify(V)  V=expand(V)  vpa(V,4)  mVdot=m2'\*Q2\*m2  mVdot=simplify(mVdot)  mVdot=expand(mVdot)  vpa(mVdot,4)  % plot V in [-.5,.5] interval [it must be positive in this interval]  fplot(@(x1) 15.09\*x1^6 + 3.933\*x1^5 + 6.476\*x1^4 - 4.584\*x1^3 + 9.988\*x1^2,[-.5,.5])  % plot mVdot in [-.5,.5] interval [it must be positive in this interval]  fplot(@(x1) 38.6\*x1^6 + 16.25\*x1^5 - 33.12\*x1^4 - 13.01\*x1^3 + 21.15\*x1^2,[-.5,.5])  % |

Problem 8: find a Lyapunov fcn to prove LOCAL asym stability

|  |
| --- |
| Matlab code  Install [YALMIP,SDPT3] |
| %% YALMIP sos study  %% problem 8: find a Lyap [CONSTRAINED] [limit cycle]  clear all,close all,clc;yalmip('clear');  x1 = sdpvar(1,1);x2 = sdpvar(1,1); % state-variables "indeterminates"  f1=-(x1^3)-4\*x1\*(x2^2)-2\*x2; % the dynamics  f2=-2\*(x1^2)\*x2+x1-8\*(x2^3);  deg1=4; eps1=1e-3;  eps1\_poly=eps1\*(x1^2+x2^2); % small-sos-poly  % V  [V,coeff\_V,mono\_list\_1] = polynomial([x1,x2],deg1); sdisplay(mono\_list\_1)  % sigma\_1  [sigma\_1,coeff\_sigma\_1,mono\_list\_1] = polynomial([x1,x2],deg1);  % sigma\_2  [sigma\_2,coeff\_sigma\_2,mono\_list\_1] = polynomial([x1,x2],deg1);  % Vdot  mVdot=-jacobian(V,[x1;x2])\*[f1;f2];  options = sdpsettings('solver','sdpt3');  % constraints "V>0","-Vdot>0","V(0)=0","Vdot(0)=0"  F=[sos(V-sigma\_1\*[(x1^2+x2^2)-.3^2]-eps1\_poly),...  sos(mVdot-sigma\_2\*[(x1^2+x2^2)-.3^2]-eps1\_poly),...  sos(sigma\_1),sos(sigma\_2),-10<=coeff\_V<=20,...  -10<=coeff\_sigma\_1<=20,-10<=coeff\_sigma\_2<=20,...  replace(V,[x1,x2],[0,0])==0,...  replace(V,[x1,x2],[1,0])==1];  [sol,m,Q,residuals,everything] = ...  solvesos(F,[],options,[coeff\_V,coeff\_sigma\_1,coeff\_sigma\_2]);  % positive-definiteness check  if ~all(real(eig(value(Q{1})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{2})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{3})))>=0) error('negativity detected->BAD!'); end  if ~all(real(eig(value(Q{4})))>=0) error('negativity detected->BAD!'); end  % get the matrices  Q1=Q{1}; Q1=Q1.\*(abs(Q1)>1e-3)  Q2=Q{2}; Q2=Q2.\*(abs(Q2)>1e-3)  % get the monomials as "sym"  m1=sym(m{1})  m2=sym(m{2})  clearvars -except Q1 m1 Q2 m2  syms x1 x2 real  V=m1'\*Q1\*m1  V=simplify(V)  V=expand(V)  vpa(V,4) |

|  |
| --- |
| To look at some of the trajectories |
| %% let us look at some random trajectories  clear all,close all,clc;  fig1=figure(1);fig1.Color=[1,1,1];  ax1=axes('Parent',fig1);  set(0,'CurrentFigure',fig1);  set(fig1,'currentaxes',ax1);  for ii=1:1:10  tspan=[0:0.01:100]; x0=1e2\*[randi([-10,10],1,1);randi([-10,10],1,1)];  x0=1e-3\*x0;  wt=tspan;  f=randi([1,10],1,1); w=sin(2\*pi\*f\*tspan);  % w=square(tspan);  [t,x]=ode45(@(t,x) odefcn(t,x,wt,w),tspan,x0);  plot(x(:,1),x(:,2),'r-','LineWidth',[2],"Parent",ax1);  hold on;  plot(x(1,1),x(1,2),'ko','LineWidth',[2],"Parent",ax1);  hold on;  % yline(1);hold on;yline(-1);hold on;  end  axis square  function xdot=odefcn(t,x,wt,w)  w=interp1(wt,w,t);  xdot=zeros(2,1);  x1=x(1);x2=x(2);  f1=-(x1^3)-4\*x1\*(x2^2)-2\*x2;  f2=-2\*(x1^2)\*x2+x1-8\*(x2^3);  xdot(1)=f1;  xdot(2)=f2;  end |