

# Demonstration of multipoly and sosopt packages

Demonstrated during Lecture 4 of CDS 270 on Oct 13th.

By U. Topcu & A. Lamperski

## Contents

- [Basic manipulations with multipoly](#)
- [Vector representation example](#)
- [Gram matrix representation](#)
- [Is p an SOS polynomial?](#)
- [SOS feasibility using sosopt](#)
- [Montzkin polynomial](#)
- [SOS synthesis example](#)
- [Global lower bound for a polynomial](#)
- [Global Stability Analysis](#)

## Basic manipulations with multipoly

Define two polynomial objects x1 and x2

```
pvar x1 x2
```

Form the polynomial  $p = 2*x1^4 + 2*x1^3*x2 - x1^2*x2^2 + 5*x2^4$

```
p = 2*x1^4 + 2*x1^3*x2 - x1^2*x2^2 + 5*x2^4
```

```
p =  
  2*x1^4 + 2*x1^3*x2  
  - x1^2*x2^2 + 5*x2^4
```

Check the fields of p

---

```
p.coef
```

```
ans =
```

```
(1,1)      2  
(2,1)      2  
(3,1)     -1  
(4,1)      5
```

```
p.deg
```

```
ans =
```

```
(1,1)      4  
(2,1)      3  
(3,1)      2  
(2,2)      1  
(3,2)      2  
(4,2)      4
```

```
p.var
```

```
ans =
```

```
'x1'  
'x2'
```

## Vector representation example

```
x = [x1;x2];
```

Form the monomials vector

```
w = monomials(x,0:4);
```

Compute the coefficients vector

```
c = poly2basis(p,w);
```

Does the decomposition work?

```
p - c' * w
```

```
ans =  
0
```

### Gram matrix representation

Compute the monomials vector  $z$ , a particular  $Q$  for the equality  $p = z' * Q * z$ , and all homogenous solutions.

```
[z,Q,N] = gramsol(p);  
Q = full(reshape(Q,[3 3]));  
N = full(reshape(N,[3 3]));
```

Does the decomposition work?

```
p - z' * Q * z
```

```
ans =  
0
```

Check the homogenous solution.

---

```
z' * N * z
```

```
ans =  
    0
```

### Is p an SOS polynomial?

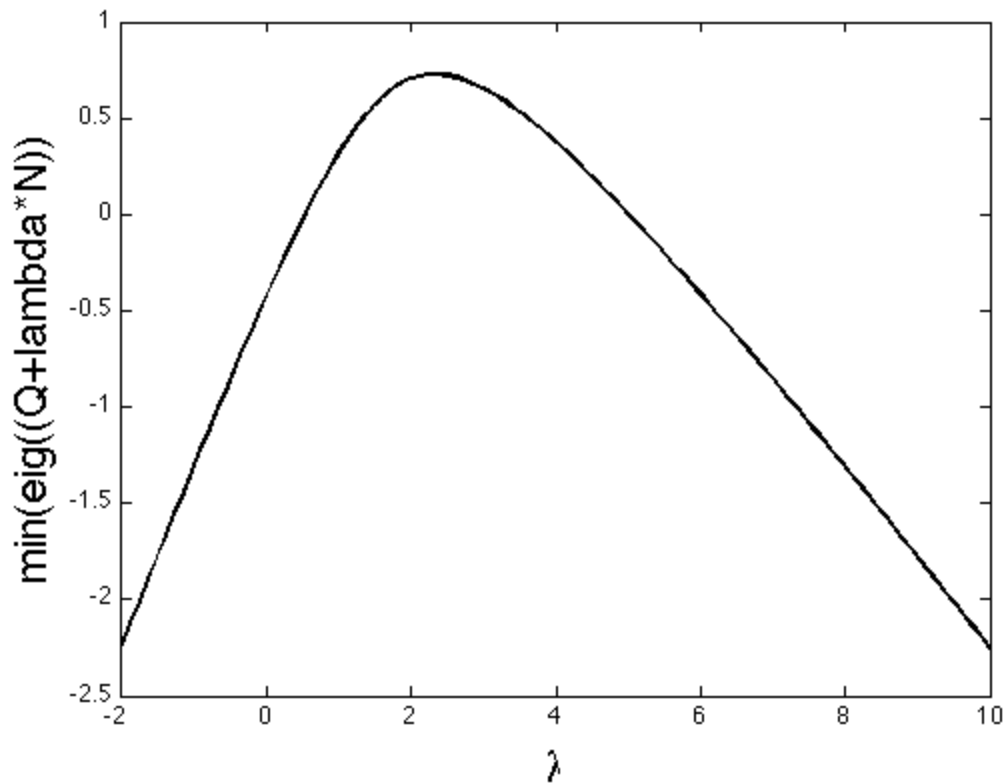
p is SOS iff there exists lambda such that  $Q + \lambda N$  is psd.

Let's take a range for lambda and check the minimum eigenvalues of  $Q + \lambda N$  for values of lambda on a grid in this range.

```
lambda = linspace(-2,10,100);  
for i1 = 1:length(lambda)  
    mineig(i1) = min(eig(Q+lambda(i1)*N));  
end
```

Plot the result

```
plot(lambda,mineig,'k','linewidth',2);  
xlabel('\lambda','fontsize',18); ylabel('min(eig((Q+lambda*N)
```



### SOS feasibility using sosopt

Check if p is a SOS polynomial using sosopt

```
[feas,z,Q] = issos(p);
```

Check if the decomposition works

```
p - z'*Q*z
```

```
ans =
-1.3145e-12*x1^4
+ 6.6191e-13*x1^3*x2
- 2.3053e-12*x1^2
*x2^2 + 1.1367e-15*x1
*x2^3 - 3.2774e-13*x2^4
```

Is the Gram matrix psd?

```
min(eig(Q))
```

```
ans =
```

```
0.7271
```

## Montzkin polynomial

What happens if we call issos for a Montzkin polynomial?

```
pMon = x1^2*x2^4 + x1^4*x2^2 + 1 - 3*x1^2*x2^2;  
[feas,z,Q] = issos(pMon);
```

Check the feasibility

```
feas
```

```
feas =
```

```
0
```

## SOS synthesis example

Given polynomials  $f_0$  and  $f_1$ , find the smallest value of  $\alpha$  such that  $f_0 + \alpha f_1$  is SOS

Problem set-up with polynomial toolbox and sosopt

```
pvar x1 x2 alpha;  
f0 = -x1^4 + 2*x1^3*x2 + 9*x1^2*x2^2 - 2*x2^4;  
f1 = x1^4 + x2^4;  
x = [x1;x2];  
obj = alpha;
```

```
[info,dopt,ssosol]=sosopt(f0+alpha*f1,x,obj);
```

s is  $f_0 + \alpha \cdot f_1$  evaluated at the minimal alpha

```
s = ssosol{1};
```

z and Q are the Gram matrix decomposition of s

```
z=ssosol{2}; Q=ssosol{3};
```

Feasibility of sosopt result

```
info.feas
```

```
ans =
```

```
1
```

Minimal value of alpha

```
dopt
```

```
dopt =
```

```
'alpha' [2.0000]
```

Verify s is  $f_0 + \alpha \cdot f_1$  evaluated at  $\alpha = 2.00$

```
s-subs(f0+alpha*f1,dopt)
```

```
ans =
```

```
0
```

Verify  $z$  and  $Q$  are the Gram matrix decomposition of  $s$

```
s-z' *Q*z
```

```
ans =  
-2.4096e-10*x1^4  
+ 4.3805e-11*x1^3*x2  
- 2.1902e-11*x1^2  
*x2^2 + 9.7467e-16*x1  
*x2^3 - 2.6285e-10*x2^4
```

Verify  $Q$  is positive semi-definite

```
min(eig(Q))
```

```
ans =  
  
1.3718e-10
```

### Global lower bound for a polynomial

Given a polynomial  $p$ ,

Problem of interest:  $\min_x p(x)$

Write the problem as

$\max_{\alpha, x} \alpha$  subject to  $p(x) - \alpha \geq 0$  for all  $x$

This is an hard problem, but if we replace psd constraint by a SOS constraint, we can compute a lower bound on the optimal value of  $\alpha$  using sosopt.

The following problem can be solved using sosopt:

$\max_{\alpha, x} \alpha$  subject to  $p(x) - \alpha \geq 0$  is SOS in  $x$

```
pvar x1 x2 x3 alpha;  
x = [x1;x2;x3];  
p = x1^4 + 4*x3^4 - 2*x1^2*x3 + 4*x1*x3^2 - 4*x2*x3^2 - 5.16*  
obj = -alpha;  
[info,dopt,soosol]=sosopt(p-alpha,x,obj);
```

Feasibility of sosopt result

```
info.feas
```

```
ans =
```

```
1
```

Optimal value of alpha

```
dopt
```

```
dopt =
```

```
'alpha' [-6.8700]
```

Then the conclusion is that  $p$  is (globally) greater than or equal to the optimal value of alpha

### Global Stability Analysis

Later in the course, we will learn why the following statement is correct and in fact many more extensions of such statement.

Let a dynamical system be governed by  $dx/dt = f(x)$  with  $f(0) = 0$ .

Then, if there exists a positive definite function  $V$  such that  $V(0) = 0$  and  $dV/dt < 0$  for all  $x$ , then the origin is a globally,

asymptotically stable equilibrium point for the system.

Now, we use sosopt to find a quadratic  $V$  that satisfies the above conditions for a specific vector field  $f$ .

Define the vector field

```
pvar x1 x2;  
x = [x1;x2];  
x1dot = -x1 - 2*x2^2;  
x2dot = -x2 - x1*x2 - 2*x2^3;  
f = [x1dot; x2dot];
```

Monomials vector to form a  $V$  using polydecvar's vector option

```
zV = monomials(x,2);
```

Candidate  $V$  is of the form  $V(x) = c^T zV(x)$

```
V = polydecvar('c',zV,'vec');
```

Constraint 1 :  $V(x) - L1 \in \text{SOS}$  – this is to make sure that  $V$  is positive definite

```
L1 = 1e-6 * ( x1^2 + x2^2 );  
sosconstr{1} = V - L1;
```

Constraint 2:  $-Vdot - L2 \in \text{SOS}$

```
L2 = 1e-6 * ( x1^2 + x2^2 );  
Vdot = jacobian(V,x)*xdot;  
sosconstr{2} = -Vdot - L2;
```

Solve with feasibility problem

```
[info,dopt,sofsol] = sosopt(sosconstr,x);
```

Is the problem feasible?

```
info.feas
```

```
ans =
```

```
1
```

What does V look like

```
Vsol = subs(V,dopt)
```

```
Vsol =
```

```
0.30089*x1^2 + 7.3212e  
-18*x1*x2 + 0.6018*x2^2
```