Polynomial Variables

This script demonstrates the creation and manipulation of polynomial variables and polynomials.

Contents

Requirements

This script requires the SOSAnalysis toolbox: http://www.aem.umn.edu/~AerospaceControl/

Create Scalar Polynomial Variables

```
% Create a scalar polynomial variables
pvar x
class(x)

% Create multiple scalar polynomial variables
pvar y z

% Concatenate polynomial variables
w = [y; z]
```

```
ans =

polynomial


w =
  [ y]
  [ z]
```

Create Vector and Matrix Polynomial Variables

```
% 4-by-1 vector polynomial variable
u = mpvar('u', 4, 1)

% 3-by-3 matrix polynomial variable
v = mpvar('v', 3, 3)
```

```
u =
  [ u_1]
  [ u_2]
  [ u_3]
  [ u_4]
```

```
v =
  [ v_1_1, v_1_2, v_1_3]
  [ v_2_1, v_2_2, v_2_3]
  [ v_3_1, v_3_2, v_3_3]
```

Create Polynomial

```
p1 = 4*x^2 + 3*y^2 + 6*x*y - 1
```

```
p1 =
  4*x^2 + 6*x*y + 3*y^2 - 1
```

Polynomial Substitution

```
% Substitute values in for independent variables
p2 = subs(p1, x, 1)

% Alternate syntax for subs
p3 = subs(p1, [y 3])

% It is also possible to substitute polynomials into other polynomials
p4 =  subs(p1, x, y + 2*z)
```

```
p2 =
  3*y^2 + 6*y + 3
```

```
p3 =
  4*x^2 + 18*x + 26
```

```
p4 =
  13*y^2 + 28*y*z + 16*z^2 - 1
```

Create Vector of Monomials

```
% Monomials of x with order 0 to 3
order = 0:3;
x_monomials = monomials(x, order)

% Monomials of x, y, and z with order 2
order = 2;
xyz_monomials = monomials([x; y; z], order)
```

```
x_monomials =
  [   1]
  [   x]
  [ x^2]
  [ x^3]
```

```
xyz_monomials =
  [ x^2]
  [ x*y]
  [ y^2]
  [ x*z]
  [ y*z]
  [ z^2]
```

Create Polynomials with many Free Parameters

Create a polynomial in x with all of its coefficients free. This is often used to create storage functions for SOS programming.

```
S = polydecvar('a', x_monomials)
```

```
S =
  a_4*x^3 + a_3*x^2 + a_2*x + a_1
```

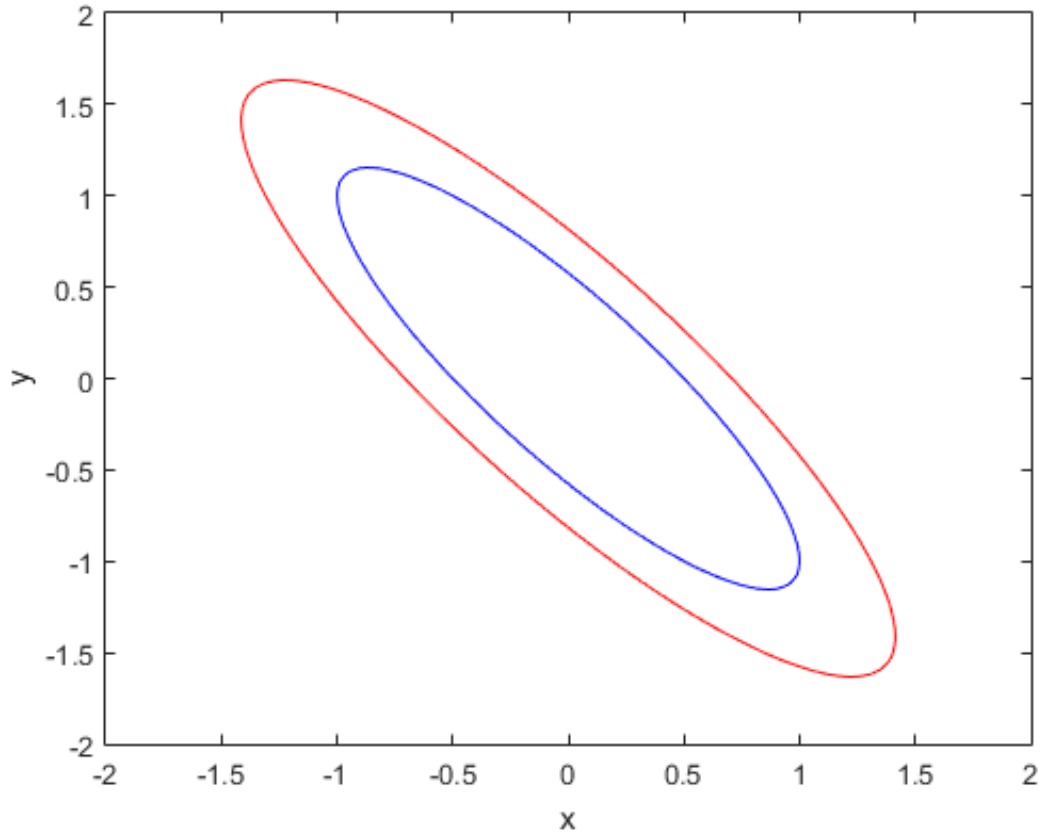
Plotting Polynomial Level Sets

Draw the contour plots of $p(x,y) = 4x + 3y^2 + xy$.

```
plotaxes = [2 -2 2 -2];

% Countour plot of p(x,y) = 0 in blue.
pcontour(p1, 0, plotaxes, 'b'); hold on

% Countour plot of p(x,y) = 1 in red.
pcontour(p1, 1, plotaxes, 'r'); hold off
```



Checking Sum-of-Squares for Polynomials

Check if the following polynomial is SOS

$$q(x,y) = x^2 - 2xy^2 + 2x^4 + 2x^3y -x^2y^2 + 6y^4$$

```
q = x^2 - 2*x*y^2 + 2*x^4 + 2*x^3*y - x^2*y^2 + 6*y^4;

% Check if q is SOS
issos(q)
```

```
ans =

     1
```

SOS Decompositions

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

```
feas =

     1
```

Gram Matrix decomposition

$q = z^T Q z$ with Q positive semidefinite

```
eig(Q)

% As a result of numerical error the SOS decompositions contain very small
% but nonzero coefficient multiplying monomials that are not in q. The
% function cleanpoly can be used to eliminate these terms. It will remove
% any term whose coefficient is less than tol.
tol = 1e-10;
cleanpoly(z'*Q*z, tol)
```

```
ans =

   5.5969e-01
   1.0681e+00
   2.9232e+00
   6.7824e+00


ans =
  1.999999999999999*x^4 + 1.999999999999996*x^3*y - 0.999999999999992*x^2
  *y^2 + 6.000000000000012*y^4 - 2.000000000000008*x*y^2
  + 1.000000000000005*x^2
```

SOS Decomposition

$q = f^T f$

```
cleanpoly(f'*f, tol)
```

```
ans =
  2*x^4 + 1.999999999999996*x^3*y - 0.999999999999992*x^2*y^2
  + 6.000000000000015*y^4 - 2.000000000000008*x*y^2 + 1.000000000000005*x^2
```

Conclusion

This example provides a brief overview of creating and manipulating polynomials with the SOSAnalysis toolbox.

Attribution

This example supplements the book "Networks of Dissipative Systems: Compositional Certification of Stability, Performance, and Safety" by Murat Arcak, Chris Meissen, and Andrew Packard.

Published with MATLAB® R2015a