EID Certification of a Nonlinear System

This example certifies equilibrium-independent passivity of a nonlinear system

$$\dot{x} = -x + 1.5x^2 - x^3 + u$$

$$y = x^3$$

This example is modified from "Equilibrium-independent passivity: A new definition and numerical certification" by George Hines, Murat Arcak, and Andy Packard, *Automatica*, vol. 47, no. 9, September 2011, pp. 1949–1956.

Contents

Requirements

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

Define Nonlinear System

Define polynomial variables

```
pvar x u xbar ubar;
```

Define system dynamics

```
f = @(x,u) -x + 1.5*x^2 - x^3 + u;
h = @(x) x^3;
```

Create storage function

Create a 4th-order storage function V with coefficients "v" serving as decision variables. V is constructed such that V(x,x) = 0.

```
Vmonomials = monomials([x; xbar], 0:2);
Q = polydecvar('v', Vmonomials);
V = (x-xbar)*Q*(x-xbar)
```

```
V =
  v_4*x^4 - 2*v_4*x^3*xbar + v_4*x^2*xbar^2 + v_5*x^3*xbar - 2*v_5*x^2
  *xbar^2 + v_5*x*xbar^3 + v_6*x^2*xbar^2 - 2*v_6*x*xbar^3 + v_6*xbar^4
  + v_2*x^3 - 2*v_2*x^2*xbar + v_2*x*xbar^2 + v_3*x^2*xbar - 2*v_3*x
  *xbar^2 + v_3*xbar^3 + v_1*x^2 - 2*v_1*x*xbar + v_1*xbar^2
```

Create a third order polynomial r, with coefficients "r" serving as decision variables.

```
Rmonomials = monomials([x; u; xbar; ubar], 3);
r = polydecvar('r' , Rmonomials)
```

```
r =
  r_1*x^3 + r_10*xbar^3 + r_11*ubar*x^2 + r_12*u*ubar*x + r_13*u^2*ubar
  + r_14*ubar*x*xbar + r_15*u*ubar*xbar + r_16*ubar*xbar^2 + r_17*ubar^2
  *x + r_18*u*ubar^2 + r_19*ubar^2*xbar + r_2*u*x^2 + r_20*ubar^3 + r_3
  *u^2*x + r_4*u^3 + r_5*x^2*xbar + r_6*u*x*xbar + r_7*u^2*xbar + r_8*x
  *xbar^2 + r_9*u*xbar^2
```

Specify SOS constraints

Create an array of expressions that are constrained to be sum-of-squares in the independent variables `(x,u,xbar,ubar)`.

```
sos_constraint = polynomial(zeros(2,1));
```

The first SOS constraint enforces the nonnegativity of V.

```
sos_constraint(1) = V;
```

The second SOS constraint enforces the nonnegativity the dissipation inequality:

$$\frac{dV}{dx} f(x,u) \leq (u-\bar{u})^T(h(x)-h(\bar{x})) - r(x,u,\bar{x},\bar{u}) f(\bar{x},\bar{u}) \quad \forall x,\bar{x} \in R^n, u,\bar{u} \in R^m$$

```
sos_constraint(2) = -jacobian(V,x)*f(x,u) + (u-ubar)'*(h(x)-h(xbar)) - r*f(xbar,ubar);
```

Solve SOS problem

Search for coefficients (in `V` and `r`) such that the SOS constraints hold. In this case, there is no objective, just a feasibility problem.

```
ind_var = [x; u; xbar; ubar];
[info,dopt] = sosopt(sos_constraint, ind_var);

if info.feas
    disp('System is Equilibrium Independent Passive')
else
    disp('Equilibrium independent passivity cannot be certified')
end
```

```
System is Equilibrium Independent Passive
```

Recover optimized `Vopt` and `ropt`

Substitute optimal coefficient values into V and r

```
Vopt = subs(V, dopt);
ropt = subs(r, dopt);
```

Verify SOS constraints

Use `cleanpoly` to eliminate terms in Dissipation inequality associated with very small monomial coefficients. Some terms in the dissipation inequality should cancel, but due to numerical roundoff, remain in the expression, with coefficients several orders of magnitude smaller than the others. The function `cleanpoly` eliminates all monomials whose coefficients are smaller (in magnitude) than a user-specified tolerance.

```
monomialTol = 1e-6;
DisInEq = cleanpoly(-jacobian(Vopt,x)*f(x,u) + (u-ubar)'*(h(x)-h(xbar)) - ropt*f(xbar,ubar), monomi
alTol);

% Check constraints
[issos(Vopt) issos(DisInEq)]
```

```
ans =

     0     1
```

Conclusions

In this example we certify that a nonlinear system is equilibrium independent dissipative. This property is important when studting interconnections of subsystems, because it allows one to analyze the entire interconnected system without specific knowledge of the systems equilibrium point.

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