Safety Certification Example

This example demonstrates safety certification for 2 intereconnected nonlinear systems.

Contents

Requirements

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

Create Polynomial Variables for each Subsystem

There are 2 subsystems, and each subsystem has 1 state and 1 input.

```
x = mpvar('x', 2,1);
u = mpvar('u', 2,1);
```

Create Polynomial Dynamics for each Subsystem

The state equations for each subsystem are

$$\dot{x}_1(t) = -x_1^3(t) + u_1(t), \quad \dot{x}_2(t) = -x_2(t) + u_2(t)$$

and the outputs are

$$y_1(t) = x_1^3(t), \quad y_2(t) = x_2(t)$$

```
f(1) = -x(1)^3 + u(1);
h(1) = x(1)^3;
f(2) = -x(2) + u(2);
h(2) = x(2);
```

Define supply-rates for each subsystem

Use supply-rates associated with output-strict passivity

```
X1 = [0 1; 1 -2];
S1 = [u(1);h(1)]'*X1*[u(1);h(1)];

X2 = [0 1; 1 -2];
S2 = [u(2);h(2)]'*X2*[u(2);h(2)];
```

Create Storage Functions

```
V1 = x(1)^4/2;
V2 = x(2)^2;
```

Verify Subsystem Dissipativity

Use sum-of-squares to verify the nonnegativity of each Dissipation inequality

```
die1 = S1 - jacobian(V1,x(1))*f(1);
die2 = S2 - jacobian(V2,x(2))*f(2);
[issos(die1) issos(die2)]

% In fact, both Dissipation inequalities are trivially 0
[die1 die2]
```

```
ans =

     1     1


ans =
  [ 0, 0]
```

Define Unsafe Set

The unsafe set is defined as

$$\mathcal{U} := \left\{x \in \mathbf{R}^2 : x_1 > 1.0 \text{ or } x_2 > 0.8\right\}$$

This is the union of two sets, defined with simple functions q_1 and q_2

$$\mathcal{U} = \{x \in \mathbf{R}^2 : \underbrace{x_1 - 1.0}_{q_1(x)} > 0\} \cup \{x \in \mathbf{R}^2 : \underbrace{x_2 - 0.8}_{q_2(x)} > 0\}$$

```
q(1) = x(1)-1;
q(2) = x(2)-0.8;
```

Create Interconnection Matrix

The interconnection is $y_1 = S_1(y_2), \quad y_2 = S_2(d - y_1)$. This characterized with an interconnection matrix of the form

```
M = [0 1 0; -1 0 1];
```

Certification

Define 2 scalar variables (later constrained to be positive), and let the to-be-determined storage function for safety verification be a linear combination of the subsystem storage functions

```
pvar p1 p2
V = p1*V1 + p2*V2;
```

Check safety for 4 different bounds on $\|d\|_2$

```
beta = [0.4 0.8 1.2 1.6];
LineColors = 'gbr';
legEntry = cell(1,1);

for j = 1:numel(beta)
   W = 1;
   X = [M; eye(size(M,2))]'*blkdiag((p1*kron(X1, diag([1 0]))+p2*kron(X2, diag([0 1]))),-W)*[M; eye
(size(M,2))];

   % X must be positive-semidefinite (by choice of p1 and p2).  Use
   % an SOS constraint, with new variables, to enforce that constraint.
   % An SOS constraint on L2con is equivalent to positive-semidefinite
   % constraint on X
   z = mpvar('z',[size(M,2) 1]);
   L2con = -z'*X*z;

   % In order to "append" the constraints that define the safe set, define
   % decision variables s1 and s2 (constrained to be positive later)
   % that are used to enforce each constraint.
   for k = 1:size(q,2)
      s(k) = polydecvar(['s' int2str(k)], 1);
   end

   % Safety constraint implying V(x) > beta on the unsafe set.
   % The variable smallE is used to enforce a strict inequality.
   smallE = 1e-6;
   sftycon(1) = V - beta(j) - smallE - s(1)*q(1);
   sftycon(2) = V - beta(j) - smallE - s(2)*q(2);

   % Vector of SOS constraints - this includes the L2 reachability
   % constraint, the nonnegativity of V, the set-containment multipliers s
   % and the safety constraint, which ensures the level sets of V do not
   % intersect the unsafe regions.
   soscon = [L2con, V, s, sftycon];

   % Solve the SOS optimization.  List all independent variables in the 2nd
```

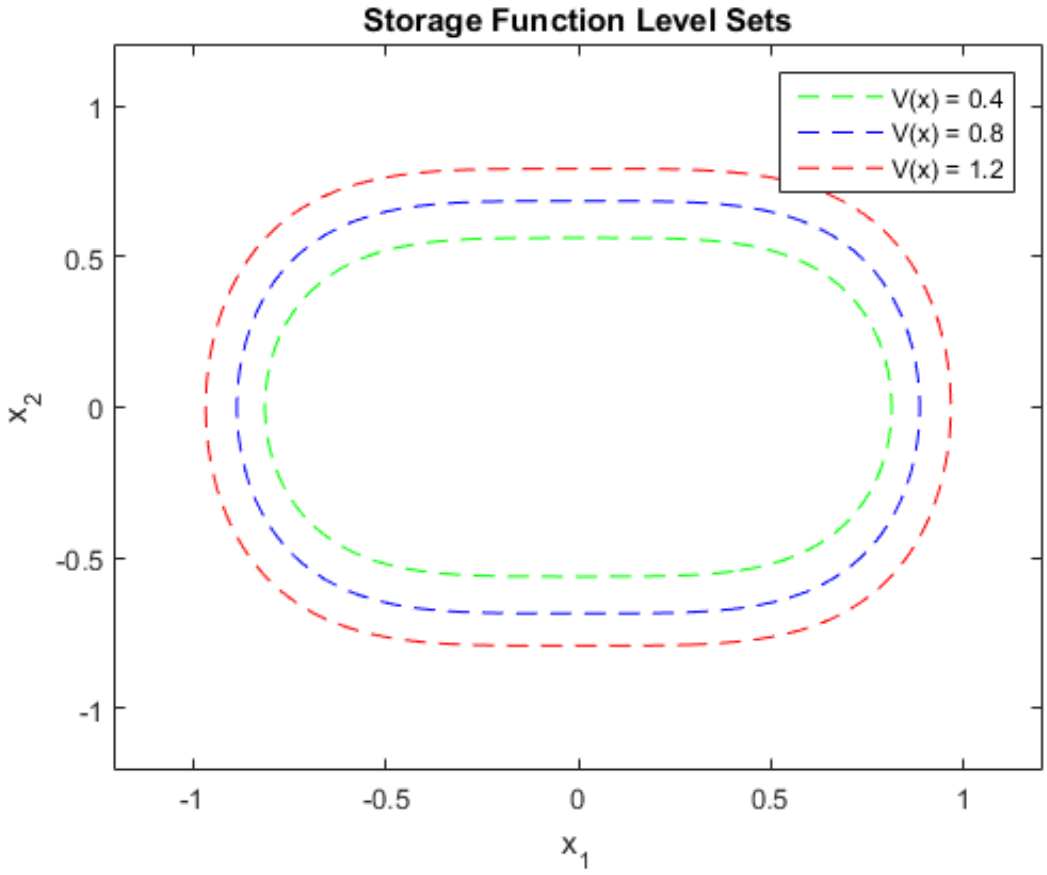
```
    % argument.  The function sosopt treats all other variables (here s and
    % p) as decision variables.  This is a feasibility problem, with no
    % objective function
    indVars = [x;z];
    [info, dopt, sossol] = sosopt(soscon, indVars, []);


    % Verify feasibility and substitute the decision variables
    % (in this case, p) into the expression for V, yielding a numerical Vs
    if info.feas
        disp(['Safety is certified for beta = ' num2str(beta(j))]);

        % Calculate storage function
        Vs = subs(V,dopt);

        % Draw contour of the safe-set for this particular value of beta
        pcontour(Vs, beta(j), 1.2*[-1 1 -1 1], [LineColors(j) '--']);
        hold on
        legEntry{j} =['V(x) = ' num2str(beta(j))];
        title('Storage Function Level Sets')
    else
        disp(['Safety cannot be certified for beta = ' num2str(beta(j))]);
    end
end
legend(legEntry);
hold off
```

```
Safety is certified for beta = 0.4
Safety is certified for beta = 0.8
Safety is certified for beta = 1.2
Safety cannot be certified for beta = 1.6
```



Conclusion

This example demonstrates compositional safety certification of a simple interconnection. SOS programming is used to find supply rates and storage functions that certify the safety and L_2 reachability of the interconnected system.

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