Platoon Safety Certification

This example demonstrates safety certification for the vehicle platoon described in Section 4.2 and 5.3.

Contents

Requirements

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

Define Interconnection of Vehicles

We assume the vehicles are in a linear formation and only measure the distance to the vehicles directly in front and back of them. The interconnection matrix M_{uw} satisfies

$$\begin{bmatrix} u \\ w \end{bmatrix} = M_{uw} \begin{bmatrix} v \\ y \end{bmatrix}$$

where v is the output of the vehicle subsystems and y is the output of the link subsystems.

```
% Define the number of vehicles N and links L
N = 3;
L = N-1;

% Form the interconnection matrix as in Equation 4.16
D = [eye(N-1); zeros(1, N-1)] - [zeros(1,N-1); eye(N-1)];
Muw = [zeros(N,N) -D; D' zeros(L,L)];
```

Add a Disturbance Input to the Interconnection

A disturbance input is appended to the interconnection matrix M such that

$$\begin{bmatrix} u \\ w \end{bmatrix} = M \begin{bmatrix} v \\ y \\ d \end{bmatrix}$$

where v is the output of the vehicle subsystems, y is the output of the link subsystems, and d is the disturbance input.

```
% Specify the vehicle to which a disturbance will be applied
% (i.e. 1 is the first vehicle and N is the last vehicle)
dist_idx = N;

% Append to Muy the disturbance input to the dist_idx vehicle
Md = zeros(N+L,1);
Md(dist_idx) = 1;
M = [Muw Md];
```

Define Vehicle Subsystems

The link dynamics are given by

$$\dot{v}(t) = -v(t) + v^0 + u(t)$$

with input $u(t)$ and output $v(t)$.

```
% Vehicle State Variables
v = mpvar('v', N, 1);
vbar = mpvar('vb', N, 1);

% Vehicle Input Variables
u = @(v,y) Muw(1:N,:)*[v; y];

% Initial vehicle velocities
v0 = [9;10;11];

% Vehicle dynamics
f_vehicle = @(v, u) -v + v0 + u;
```

Define Link Variables and Dynamics

The link dynamics are given by

$$\dot{z}(t) = w(t)$$

$$y(t) = (z(t) - z_0)^{1/3}$$

where $w(t)$ is the input and $y(t)$ is the output.

```
% Link State Variables
z = mpvar('z', L, 1);
zbar = mpvar('zb', L, 1);

% Link Input Variables
w = @(v,y) Muw((N+1:end),:)*[v; y];

% Link Dynamics
f_link = @(w) w;
```

Define Constraint for the Link Output

The link output $y(t) = (z(t) - z_0)^{1/3}$ is not a polynomial, but we can we represent it by introducing a new polynomial variable y and a polynomial constraint $y^3 = z - z_0$.

```
% Link Output Variables
y = mpvar('y', L, 1);
ybar = mpvar('yb', L, 1);

% Define the desired vehicle spacing
z0 = 20*ones(L,1);

% Link Output Constraint - In the SOS program this function will be used to
% enforce this constraint.
link_constraint = @(z, y) z - z0 - y.^3;
```

Define Storage Function and Supply Rate for Vehicle subsystems

```
% Vehicle storage functions
Sv = (v-vbar).^2;

% Supply Rate Matrix
Xv = [0 1/2; 1/2 -1];
```

Define Storage Function and Supply Rate for Link subsystems

```
% Link storage functions where y^3 = z-z0 and yb^3 = zb-z0
Rl = 3/4*y.^4 - y.^3.*ybar + 1/4*ybar.^4;

% Supply Rate Matrix
Xl = [0 1/2; 1/2 0];
```

Define Composite Storage Function

The choice of $p_i = 4$ for the storage function weights ensures that the interconnected system satisfies the L_2 reachability supply rate.

```
p = 4*ones(N+L, 1);
V = p(1:N)'*Sv + p((N+1):end)'*Rl;
```

Define Unsafe set

The unsafe set is defined as

$$\mathcal{U} = \{z \in \mathbf{R}^2 : |z_1| \leq \gamma\} \cup \{|z_2| \leq \gamma\}$$

However, since we are constraining $y^3 = (z - z_0)$, this set can also be expressed as

$$\mathcal{U} = \{y \in \mathbf{R}^2 : |z_0 + y_1^3| \leq \gamma\} \cup \{|z_0 + y_2^3| \leq \gamma\}$$

```
gamma = 5;
q = [gamma - z0 - y.^3 z0 + y.^3 + gamma];

% Number of constraints per link
nq = size(q, 2);
```

Create Multiplier Functions to enforce Equality and Safety constraints

```
% Concatenate all independent variables
ind_variables = [z; zbar; v; vbar; y; ybar];

% Create multiplier functions to enforce the EID constraints
r = polynomial(zeros(N+L,L));
for j = 1:N+L
    for k = 1:L
        r(j,k) = polydecvar(['r' int2str(j) int2str(k)], monomials(ind_variables, 0:1));
    end
end

% Create multiplier functions to enforce y^3 = z-z0 and yb^3 = zb-z0
t = polynomial(zeros(2*L,L));
for j = 1:2*L
    for k = 1:L
       t(j,k) = polydecvar(['t' int2str(j) int2str(k)], monomials([z; y; zbar; ybar], 0:1));
    end
end

% Create multiplier function to enforce the safety constraints
s = polynomial(zeros(L,nq));
for j = 1:L
    for k = 1:nq
        s(j,k) = polydecvar(['s' int2str(j) int2str(k)], monomials(ind_variables, 0:1));
    end
end
```

Create Safety Constraints

Determine the largest β such that the system is safe for any disturbance $\|d\|_2^2 \leq \beta$.

```
% Initialize polynomial variable for gain bound
pvar beta
```

```
% Define eps as a small positive number to ensure strict feasibility of the
% SOS safety constraint
eps = 1e-6;

% Create safety constraints ensuring the subsytem states do not intersect the
% unsafe set
safety_constraints = polynomial(zeros(L,1));
for k = 1:L
    safety_constraints(k) = V - beta - eps - s(:,k)'*q(k,:)' + r(:,k)'*[f_vehicle(vbar, u(vbar,ybar
)); f_link(w(vbar,ybar))] + t(:,k)'*[link_constraint(z,y); link_constraint(zbar,ybar)];
end
```

Solve SOS problem to find the maximum disturbance bound

```
% Concatenate SOS constraints
sos_constraints = [V; s(:); safety_constraints];

% Find the largest beta such that the SOS constraints holds by minimizing
% -beta subject to the SOS constraints
[info, dopt, sossol] = sosopt(sos_constraints, ind_variables, -beta);
```

Check Results

```
% Check SOS feasibility
info.feas

% Maximum value for beta
betamax = -info.obj
```

```
ans =

     1


betamax =

   5.1980e+01
```

Conclusion

In this example compositional safety certification is performed for an interonnection of subsystems that are equilibrium independent dissipative. We certify the system is safe for any disturbance $\|d\|_2^2 \leq \beta$.

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