sensorFaults

April 5, 2018

1 Model-Based Fault Diagnosis and Security of CPS

```
In [1]: display("text/javascript", """MathJax.Hub.Config({TeX: { equationNumbers: { autoNumber
```

```
In [2]: using ControlSystems
```

```
INFO: Recompiling stale cache file /Users/jleny/.julia/lib/v0.6/Polynomials.ji for module Polyn
```

```
In [3]: srand(1234);
```

1.1 Sensor Faults

```
In [4]: # Example in S. Sundaram lecture notes, ch. 4
        A = 0.39 * [1 1 0 1; 0 1 1 0; 1 0 1 1; 0 0 1 1]
        C = [1 0 0 0; 0 1 0 0; 0 1 1 0]
        sys = ss(A,zeros(4,1),C,zeros(3,1),-1)
```

```
Out[4]: StateSpace:
        A =
                x1      x2      x3      x4
          x1   0.39    0.39    0.0     0.39
          x2   0.0     0.39    0.39    0.0
          x3   0.39    0.0     0.39    0.39
          x4   0.0     0.0     0.39    0.39
        B =
                u1
          x1   0.0
          x2   0.0
          x3   0.0
          x4   0.0
        C =
                x1      x2      x3      x4
          y1   1.0     0.0     0.0     0.0
          y2   0.0     1.0     0.0     0.0
          y3   0.0     1.0     1.0     0.0
        D =
                u1
          y1   0.0
```

```
y2   0.0
y3   0.0

Sample Time: unspecified
Discrete-time state-space model
```

```
In [5]: abs.(eigvals(A))
```

```
Out[5]: 4-element Array{Float64,1}:
 0.983338
 0.347345
 0.347345
 0.39
```

```
In [6]: # Observability matrix
        C1 = Array(C[1,:]')
        rank(obsv(A,C1))
```

```
Out[6]: 4
```

```
In [7]: L = place(A',C1',[0,0,0,0])'  # place the 4 eigenvalues at 0
```

```
Out[7]: 4×1 Array{Float64,2}:
 1.56
 1.17
 1.95
 1.56
```

```
In [8]: A - L*C1
```

```
Out[8]: 4×4 Array{Float64,2}:
 -1.17  0.39  0.0   0.39
 -1.17  0.39  0.39  0.0
 -1.56  0.0   0.39  0.39
 -1.56  0.0   0.39  0.39
```

```
In [10]: nsteps = 15
         x_true = zeros(size(A,1),nsteps)
         y_meas = zeros(size(C,1),nsteps-1)
         x_true[:,1] = randn(4)
         x_est = zeros(x_true)
         y_est = zeros(y_meas-1)
         residues = zeros(size(C,1),nsteps-1)
         fault = [1;0;0]   # additive constant fault vector - choose ona at a time
         for k=1:nsteps-1
             y_meas[:,k] = C * x_true[:,k] + fault
             x_true[:,k+1] = A * x_true[:,k]
             x_est[:,k+1] = A * x_est[:,k] + L*(y_meas[1,k] - C1*x_est[:,k])
             y_est[:,k] = C * x_est[:,k]
             residues[:,k] = y_meas[:,k] - y_est[:,k]
         end
```

```
In [11]: using Plots
         plotlyjs()
         plot(residues')
```

1.2 Parity Space Method

```
In [12]: A = [1 1 0 1; 0 1 1 0; 1 0 1 1; 0 0 1 1]
         C = [1 0 0 0; 0 1 0 0; 0 1 1 0]
         Bf = [1 0; 0 1; 0 1; 1 1]
         Bd = [0; 1; 0; 1];

In [13]: O3 = obsv(A,C);

In [14]: function outputMatrices(A,B,C,D,order=size(A,1)-1)
             O = C  # Observability matrix
             M = D  # Invertibility matrix
             for i=1:order
                 M = [D zeros(size(D,1),size(M,2)); O*B M]
                 O = [C; O*A]
             end
             return(O,M)
         end
```

```
Out[14]: outputMatrices (generic function with 2 methods)
```

```
In [15]: (O,Md) = outputMatrices(A,Bd,C,zeros(size(C,1),size(Bd,2)));

In [16]: V = nullspace([O Md]')'  # parity space
```

```
Out[16]: 5×12 Array{Float64,2}:
          -0.350567    0.0134225  -0.357279     -0.376362   -0.195131    0.195131
          -0.140148   -0.0808385  -0.0997292     -0.207966    0.179485   -0.179485
          -0.155301   -0.400901    0.0451493      0.237294    0.0649487  -0.0649487
          -0.362234   -0.154066   -0.285201       0.0847226   0.346559   -0.346559
           0.0434518  -0.66265     0.374777      -0.133225   -0.0187456   0.0187456
```

```
In [17]: (O,Mf) = outputMatrices(A,Bf,C,zeros(size(C,1),size(Bf,2)));

In [18]: P = V*Mf
```

```
Out[18]: 5×8 Array{Float64,2}:
          -0.357279    0.357279   -0.506302    -0.376362    0.195131   0.0  0.0
          -0.0997292   0.0997292  -0.640607     -0.207966   -0.179485   0.0  0.0
           0.0451493  -0.0451493  -0.028196      0.237294   -0.0649487  0.0  0.0
          -0.285201    0.285201   -0.207804      0.0847226  -0.346559   0.0  0.0
           0.374777   -0.374777    0.135997     -0.133225    0.0187456  0.0  0.0
```

Because the last two columns are 0, we cannot detect any components of a fault occuring at k=3.