LMIs and Schur Formula, example: distance between ellipses

© 2024, Antonio Sala Piqueras, Universitat Politècnica de València. All rights reserved.

Objective: calculate the minimum distance (and the points that achieve it) between two ellipses (or ellipsoids, since we can generalize to n-D).

Presentation in video: https://personales.upv.es/asala/YT/V/distelliEN.html

*Let's load the necessary toolboxes (Yalmip, Sedumi in this case).

```
if ~exist("yalmiptest")
    addpath("~/bin/MATLAB/tbxmanager/")
    tbxmanager restorepath
end
```

Problem statement

Let us define ellipses as $(x-c)^T \cdot P \cdot (x-c) \leq 1$; or with the equivalent inverse form (equiv. if non degenerate): $(x-c)^T \cdot Q^{-1} \cdot (x-c) \leq 1$. Both P and Q are assumed to be positive definite matrices.

```
c{1}=[2;0];
Q{1}=[0.5 0;0 1];

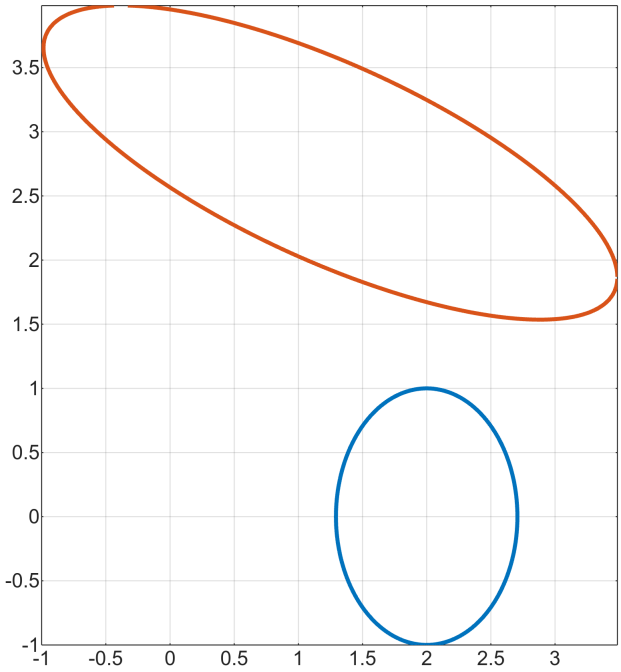
c{2}=[1.25;2.76];
Q{2}=[5 -2;-2 1.5];
```

Shape matrix Q is "proportional" to the square "size" of the ellipsoid, P es "inverse proportional".

```
for k=1:2
    P{k}=inv(Q{k});
    fimplicit(@(x,y) ([x;y]-c{k})'*P{k}*([x;y]-c{k})-1,LineWidth=2), hold
on
end
```

```
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly
vectorize your function to return an output with the same size and shape as the input arguments.
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly
vectorize your function to return an output with the same size and shape as the input arguments.
```

```
hold off, grid on, axis equal
```



Schur's formula: $A - B^T Q^{-1} B \succ 0$ iff and $Q \succ 0$ iff $\begin{pmatrix} A & B^T \\ B & Q \end{pmatrix} \succ 0.$

Identifying $B \equiv (x-c),\ A = 1$, we have that $1 - (x-c)^T Q^{-1}(x-c) > 0$ and $Q \succ 0$ iff:

$$\begin{pmatrix} 1 & x^T - c^T \\ x - c & Q \end{pmatrix} \succ 0$$

Let's introduce a search point on each ellipse as decision variable:

$$\begin{pmatrix} 1 & x_1^T - c_1^T \\ x_1 - c_1 & Q_1 \end{pmatrix} \succ 0 \qquad \begin{pmatrix} 1 & x_2^T - c_2^T \\ x_2 - c_2 & Q_2 \end{pmatrix} \succ 0$$

```
x{1}=sdpvar(2,1);
x{2}=sdpvar(2,1);
LMIs=[];
for k=1:2
    LMIs=[LMIs; ...
        [1 x{k}'-c{k}'; x{k}-c{k}  Q{k}]>=0 ];
end
```

Distance between two points being smaller than d amounts to saying $d^2 > \|x_1 - x_2\|^2 = (x_1 - x_2)^T (x_1 - x_2)$, i.e., $d^2 - (x_1 - x_2)^T (x_1 - x_2) > 0$, thus Schur formula says that it's equivalent to:

$$\begin{pmatrix} d^2 & x_1^T - x_2^T \\ x_1 - x_2 & I_{2\times 2} \end{pmatrix} \succ 0$$

We will add that to the LMIs, changing d^2 nonlinear to "dsquared= d^2" linear in dsquared.

```
dsquared=sdpvar();
LMIs=[LMIs; ...
    [dsquared x{1}'-x{2}'; x{1}-x{2} eye(2)]>=0 ];
```

If we look for the minimum distance, we have to find the minimum dsquared for which there are feasible points x{1} and x{2}.

```
sol=optimize(LMIs,dsquared)
```

```
SeDuMi 1.3 by AdvOL, 2005-2008 and Jos F. Sturm, 1998-2003.
Alg = 2: xz-corrector, theta = 0.250, beta = 0.500
eqs m = 5, order n = 10, dim = 28, blocks = 4
nnz(A) = 9 + 0, nnz(ADA) = 25, nnz(L) = 15
 it :     b*y       gap    delta  rate   t/tP*  t/tD*   feas cg cg  prec
  0 :            1.23E+01 0.000
  1 :   8.82E-02 3.19E+00 0.000 0.2594 0.9000 0.9000   1.87  1  1  1.1E+00
  2 :  -1.95E-01 5.93E-01 0.000 0.1860 0.9000 0.9000   1.64  1  1  7.7E-01
  3 :  -3.75E-01 5.30E-02 0.000 0.0893 0.9900 0.9900   1.01  1  1  8.0E-02
  4 :  -4.09E-01 3.91E-04 0.086 0.0074 0.9990 0.9990   1.02  1  1  6.2E-04
  5 :  -4.09E-01 7.63E-05 0.000 0.1952 0.9000 0.9000   1.00  1  1  1.2E-04
  6 :  -4.09E-01 1.33E-05 0.000 0.1746 0.9000 0.9000   1.00  1  1  2.1E-05
  7 :  -4.09E-01 1.55E-06 0.069 0.1165 0.9450 0.9450   1.00  1  1  2.5E-06
  8 :  -4.09E-01 7.20E-08 0.241 0.0464 0.9900 0.9900   1.00  1  1  1.1E-07
  9 :  -4.09E-01 1.52E-08 0.000 0.2118 0.9000 0.9000   1.00  1  1  2.4E-08
 10 :  -4.09E-01 8.23E-10 0.067 0.0540 0.9900 0.9900   1.00  2  2  1.3E-09
 11 :  -4.09E-01 6.66E-11 0.363 0.0809 0.9900 0.9900   1.00  2  2  1.1E-10

iter seconds digits       c*x               b*y
 11      0.1   9.7 -4.0871711521e-01 -4.0871711529e-01
|Ax-b| =   2.7e-12, [Ay-c]_+ =   1.7E-11, |x|=  2.3e+00, |y|=  3.6e+00

Detailed timing (sec)
   Pre          IPM          Post
7.594E-03    1.930E-02    8.147E-04
Max-norms: ||b||=1, ||c|| = 5.520000e+00,
Cholesky |add|=0, |skip| = 0, ||L.L|| = 2589.77.
sol = struct with fields:
    yalmipversion: '20230622'
    matlabversion: '24.1.0.2653294 (R2024a) Update 5'
       yalmiptime: 0.1351
       solvertime: 0.0284
             info: 'Successfully solved (SeDuMi)'
          problem: 0
```

```
dsquared
```

```
Linear scalar (real, 1 variable)
Current value: 0.40872
Coefficients range: 1 to 1
```

```
sqrt(value(dsquared))
```

```
ans = 0.6393
```

```
solx{1}=value(x{1});
```

```
solx{2}=value(x{2});
for k=1:2
    fimplicit(@(x,y) ([x;y]-c{k})'*P{k}*([x;y]-c{k})-1,LineWidth=2), hold
on
end
```

```
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly
vectorize your function to return an output with the same size and shape as the input arguments.
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly
vectorize your function to return an output with the same size and shape as the input arguments.
```

```
for k=1:2
    plot(solx{k}(1),solx{k}(2),'x',LineWidth=3)
end
hold off, grid on, axis equal
```

