Tutorial on semidefinite programming (SDP)
Uniandes/Externado, Spring 2006

SeDuMi exercises

1. Recall the least-squares problem

$$\min \|Pv - q\|. \tag{LS}$$

   (a) Set P, q as follows

```
> P = [1, 0 ; 1, 0.001 ; 10, -0.01] ;
> q = [1 2 0]' ;
```

   Use the matlab command

```
 > v =  P \ q
```

   to find the solution to (LS) for the above values of `P, q`.

   What is the value of `v` and the value of `norm(P*v-q)` that you found?

   (b) Now consider the matrix Q obtained by adding a small random perturbation to P

```
> Q = P + 0.05*randn(3,2)
```

   What is the value of `norm(Q*v-q)` for the solution found in (a)? Repeat this a few (two or three) times.

   (c) Now use the matlab function `rls` discussed in class to find a robust solution to (LS) for $\rho = 0.1$. That is:

```
> rho = 0.1 ;
> [At,b,c,K] = rls(P,q,rho) ;
> [x,y] = sedumi(At',b,c,K) ;
```

   what is the value of `v` and the value of `norm(P*v-q)` that you found?

   (d) Repeat part (b) for the new solution `v` that you found in (c). How different is the behavior now?

2. Let $M \in \mathbf{S}^n$ be given and consider the nearest matrix problem

$$\min \{\|M - X\| : X \succeq 0\} \tag{NM}$$

   (a) Suppose we are interested in solving (NM) for the following norm:

$$\|M - X\| = \max \{|M_{ij} - X_{ij}| : i, j = 1, \ldots, n\}.$$

   Reformulate (NM) as an LP/SOCP/SDP problem.

   (b) Write a matlab function `[At,b,c,K] = nm(M)` or `[A,b,c,K] = nm(M)` that prepares the `SeDuMi` data for the problem that you need to solve in (a). Test your code with the following matrices:

$$M = \begin{bmatrix} 1 & 1.01 \\ 1.01 & 1 \end{bmatrix}, \quad M = \begin{bmatrix} 0.3 & 0.6 & 0 \\ 0.6 & 0.8 & -0.7 \\ 0 & -0.7 & 0.6 \end{bmatrix}.$$

(c) Suppose we are interested in solving (NM) for the Frobenius norm:

$$\|M - X\| := ((M - X) \bullet (M - X))^{1/2} = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} (M_{ij} - X_{ij})^2 \right)^{1/2}.$$

Repeat parts (a) and (b) for this norm.

(d) (If there is time) Suppose we are given another matrix $E \in \mathbf{S}^n$ whose entries are zeros and ones. This defines a sparsity pattern, for example

$$E = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

corresponds to the sparsity pattern

$$\begin{bmatrix} * & * & 0 \\ * & * & * \\ 0 & * & * \end{bmatrix}$$

Modify your code in (b) so that the matrix X has the same sparsity pattern. Test your code for the matrices

$$M = \begin{bmatrix} 0.3 & 0.6 & 0 \\ 0.6 & 0.8 & -0.7 \\ 0 & -0.7 & 0.6 \end{bmatrix}, \quad E = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$