A Matlab toolbox for semidefinite programming

Franz Rendl
Technische Universität Graz
Institut für Mathematik
Kopernikusgasse 24, A-8010 Graz, Austria
e-mail: rendl@fmatbds01.tu-graz.ac.at

December 8, 1994

Abstract

Semidefinite programs have found increasing interest in the recent past, see [1, 2]. This note describes an easy to use toolbox in Matlab to solve semidefinite programs by interior point methods. The user has to provide the data of the problem and write a routine that provides an initial feasible interior point to the problem. Then she can use the preset routines to solve the problem.

1 The problem

Let C be a given symmetric $n \times n$ matrix, $a \in \Re^m$ and $\mathcal{A}$ be a linear operator, that maps symmetric matrices of size n into vectors in $\Re^m$. We provide a Matlab toolbox to solve the following pair of primal and dual semidefinite problems.

primal problem. maximize $tr CX$ such that $a - \mathcal{A}(X) = 0$, X *psd.*

dual problem. minimize $a^t y$ such that $\mathcal{A}^t(y) - C$ *psd.*

interior point assumption. It is assumed throughout, that there exists (X, y) such that X is positive definite, $\mathcal{A}(X) = a, \mathcal{A}^t(y) - C$ positive definite. We call these points *feasible interior points.*

Under this assumption strong duality holds. For the theoretical backbone of the algorithm presented, we refer to [2]. The operator $\mathcal{A}$ can be represented by m symmetric matrices A_i as follows.

$$\mathcal{A}(X) = \begin{pmatrix} tr A_1 X \\ \vdots \\ tr A_m X \end{pmatrix}$$

This implies that

$$\mathcal{A}^t(y) = \sum_i y_i A_i.$$

2 The algorithm

The method used in the Matlab routines is based on [2]. The reader is referred to this paper for any theoretical details.

It is convenient to introduce a slack matrix

$$Z = \mathcal{A}^t(y) - C$$

for the dual. The problems are solved using a predictor-corrector primal-dual interior point approach. Here is the key step. Given an interior feasible point (X, y, Z), and a value μ, a new interior feasible point is obtained by solving the following linear system, corresponding to a Newton step on the KKT conditions at (X, y, Z).

$$\begin{aligned} \delta Z &= \mathcal{A}^t(\delta y) \\ \mathcal{A}(\delta X) &= 0 \\ Z\delta X + \delta Z X &= \mu I - ZX \end{aligned}$$

The last condition comes from linearizing

$$(Z + \delta Z)(X + \delta X) = \mu I.$$

Note that the first and third of these equations can be used to express δX and δZ in terms of δy. This system is solved using a predictor-corrector approach, that provides an improved solution to the nonlinear third equation. Here are the details.

$$\begin{aligned} \delta Z^p + \delta Z^c &= \mathcal{A}^t(\delta y^p + \delta y^c) \\ \mathcal{A}(\delta X^p + \delta X^c) &= 0 \\ ZX + \delta Z^p X + \delta Z^c X + Z\delta X^p + Z\delta X^c + \delta Z^p \delta X^p &= \mu I. \end{aligned}$$

The last equation is obtained from approximating $(Z + \delta Z^p + \delta Z^c)(X + \delta X^p + \delta X^c)$ by the term on the left hand side. These equations are decoupled as follows.

$$\begin{aligned} \delta Z^p &= \mathcal{A}(\delta y^p) \\ \delta Z^c &= \mathcal{A}(\delta y^c) \\ \mathcal{A}(\delta X^p) &= 0 \\ \mathcal{A}(\delta X^c) &= 0 \\ \delta Z^p X + Z\delta X^p &= -ZX \\ \delta Z^c X + Z\delta X^c &= \mu I - \delta Z^p \delta X^p \end{aligned}$$

These systems can be solved first for the predictor $(\delta X^p, \delta y^p, \delta Z^p)$, which in turn is used to solve for the corrector $(\delta X^c, \delta y^c, \delta Z^c)$.

predictor system. After substitution one obtains:

$$\begin{aligned} \mathcal{A}(Z^{-1}\mathcal{A}^t(\delta y^p)X) &= -a \\ \delta Z^p &= \mathcal{A}^t(\delta y^p) \\ \delta X^p &= -X - Z^{-1}\delta Z^p X \end{aligned}$$

This is used to find the correctors.

corrector system.

$$\begin{aligned} \mathcal{A}(Z^{-1}\mathcal{A}^t(\delta y^c)X) &= \mu\mathcal{A}(Z^{-1}) - \mathcal{A}(Z^{-1}\delta Z^p \delta X^p) \\ \delta Z^c &= \mathcal{A}^t(\delta y^c) \\ \delta X^c &= \mu Z^{-1} - Z^{-1}\delta Z^p \delta X^p - Z^{-1}\delta Z^c X \end{aligned}$$

actual search direction. The actual search direction is now the sum of the two steps.

$$dy = \delta y^p + \delta y^c, \quad dX = \delta X^p + \delta X^c, \quad dZ = \delta Z^p + \delta Z^c.$$

To conclude one iteration one finds primal and dual step sizes α_p and α_d, so that the new point

$$X + \alpha_p dX, \quad y + \alpha_d dy, \quad Z + \alpha_d dZ$$

is again a feasible interior point. The parameter μ is updated, and one iterates, until the duality gap is sufficiently small.

3 User supplied data

The user has to provide the data defining the problem, and write a routine, that provides an interior feasible starting point. The operator $\mathcal{A}$ is defined through the m matrices A_i. These are stored in two arrays.

```
nonz(i), i = 1, ... , m
```

contains the number of nonzeros of A_i in the upper triangle, including the main diagonal. The second array stores the indices and actual entries of the nonzeros of each A_i. This array consists of 3 rows.

```
store( 3, :)
```

A typical column of this array contains

$$\begin{pmatrix} i \\ j \\ A_k(i,j) \end{pmatrix}.$$

The order of the nonzeros for a specific matrix A_k is arbitrary, but it is assumed that the first $nonz(1)$ entries correspond to A_1, the next $nonz(2)$ entries to A_2, and so on. Thus the number of columns of the array *store* is equal to the sum of the entries of the array *nonz*.

Thus an instancs of a semidefinite program is defined by the quadruple

```
C, a, store, nonz
```

where C is a symmetric matrix of size n, a and *nonz* are column vectors of size m and *store* contains the indices of the A_i.

Finally the user has to provide a Matlab function, that generates an interior feasible starting point. Here is the call to this function.

```
[X, y, Z]  = psd_strt( C, a, store, nonz)
```

Once this is done the problem can be solved by calling the matlab function *psd*.

```
[X, y, psi] = psd( C, a, store, nonz);
```

4 The Matlab functions

The toolbox consists presently of the following Matlab functions, which need not be changed by the user.

- psd.m: This is the main routine. It has the following calling sequence. On successful termination, X is primal optimal, y is dual optimal, and *psi* is the optimal value of the primal.

  ```
  [X, y, psi] = psd( C, a, store, nonz);
  ```

- op_a.m: This function evaluates $\mathcal{A}(X)$. Calling sequence:

  ```
  vector = op_a( X, store, nonz);
  ```

- op_at.m: This function evaluates $\mathcal{A}^t(y)$. Calling sequence: Note that here the primal matrix size n is a fourth input argument.

  ```
  matrix = op_at( y, store, nonz, n);
  ```

- op_biga.m: This function evaluates the system matrix corresponding to the operator $\mathcal{A}(Z^{-1}\mathcal{A}^t(.)X)$. Calling sequence:

```
matrix = op_biga( X, Zinv, store, nonz);
```

- psd_feas.m: This function checks strict feasibility of a point (X, y, Z). Calling sequence:

```
okay = psd_feas(X, y, Z, C, a, store, nonz);
```

  If $okay = 0$ then the solution is strictly feasible, within the tolerances set in the program. Otherwise a message is printed, that indicates which of the conditions is violated.

- pd_check.m: This function checks whether a given matrix is positive definite or not.

- pd_solve.m: This function solves a positive definite system, once the cholesky factor is given.

5 An example

Consider the following artificial example.

$$\max tr\, CX \quad such\ that$$

$$x_{ii} + x_{i+1,i+1} - x_{i,i+1} - x_{i+1,i} = 2, \quad i = 1, \ldots, n-1$$

$$tr(X) = n,$$

$$X \;\; \text{positive semidefinite.}$$

Here there are n constraints. The first $n-1$ constraints are zero matrices with a 2 by 2 block of the form

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

on the main diagonal starting in row and column i. The last matrix $A_n = I_n$, the identity matrix of size n.

Suppose, $n = 5$ and C is the matrix of all ones. Note that $X = I_n$ is a feasible interior primal point and $y_i = 0, \;\; i = 1, \ldots, n-1, \;\; y_n = n+1$ leads to an interior dual solution.

Here is the matlab function that defines the starting point.

```
function [ X, y, Z] = psd_strt( C, a, store, nonz);
  [n, n1] = size( C);
  X = eye(n);
  y = zeros(n,1);
  y(n)=max(sum(abs(C)))*1.1;
  Z = op_at( y, store, nonz, n) - C;
```

Note that the definition of Z is done using the supplied function *op_at*, which evaluates $\mathcal{A}^t(y)$.

The data can be set as follows.

```
n=5;
a = 2 * ones(n,1);
a(n) = n;
C = ones( n);
nonz = 3*ones(n,1);
nonz(n) =n;
for i = 1:n-1
store(:,3*i-2:3*i)= [i i i+1
                     i i+1 i+1
                     1 -1  1];
                     end;
store(:, 3*(n-1)+1: 3*(n-1)+n)=[1:n; 1:n; ones(1,n)];
```

This is contained in the m-file psd_expl.m. Here is a sample output.

```
[X,y,psi]=psd(C,a,store,nonz)
   iter     alphap     alphad    log(gap)  primal     dual      unsym
      0     1.0000     1.0000     1.3522    5.0000   27.5000
 1.0000     0.7600     1.0000     1.1093   10.1044   22.9663  -99.0000
 2.0000     1.0000     1.0000     0.5072   11.6531   14.8686  -15.3525
 3.0000     1.0000     1.0000    -0.3958   12.7525   13.1545  -14.6122
 4.0000     1.0000     1.0000    -1.2989   12.9722   13.0224  -13.2453
 5.0000     1.0000     1.0000    -2.2020   12.9963   13.0025  -12.7682
 6.0000     1.0000     1.0000    -3.1051   12.9995   13.0003  -12.0412
 7.0000     1.0000     1.0000    -4.0082   12.9999   13.0000  -10.9620
 8.0000     1.0000     1.0000    -4.9113   13.0000   13.0000   -9.9340
 9.0000     1.0000     1.0000    -5.8144   13.0000   13.0000   -9.0309
10.0000     1.0000     1.0000    -6.7175   13.0000   13.0000   -8.1278
elapsed time:
 4.9400
X =
 0.8400     0.0400     0.8400     0.0400     0.8400
 0.0400     1.2400     0.0400     1.2400     0.0400
 0.8400     0.0400     0.8400     0.0400     0.8400
 0.0400     1.2400     0.0400     1.2400     0.0400
 0.8400     0.0400     0.8400     0.0400     0.8400
y =
-2.0000
-1.0000
-1.0000
-2.0000
 5.0000
psi =
13.0000
```

6 Comments

There are several observations to make..

1. I do not take any **responsibility** or **garantuee** whatsoever on the performance of the code. It works well on the problems that I tested it, and I would encourage any feedback on the code.

2. I realize that it is limiting to have the user provide an initial strictly interior solution. This may be handled automatically in the update to this version.

3. It is possible to include linear inequalities into the model. This feature definitely will be included in the next update.

4. For specific problems, it may be advantagous to rewrite the function files describing the action of the operator $\mathcal{A}$. For instance in the max-cut case, one has

   $$\mathcal{A}(X) = \mathrm{diag}(X),$$

   and the system matrix simply is

   $$X. * Z^{-1},$$

   i.e. the elementwise product of the two matrices.

5. This version does not exploit any features of version 4 of Matlab.

References

[1] F. ALIZADEH, J.A. HAEBERLY and M.L. OVERTON. A new primal-dual interior point method for semidefinite programming. Technical Report, August 1994.

[2] C. HELMBERG, F. RENDL, R. J. VANDERBEI, and H. WOLKOWICZ. An interior-point method for semidefinite programming. Technical Report CORR 93-20, SOR 93-15, Department of Combinatorics and Optimization, Waterloo, Ont, 1993. to appear in: SIAM optimization.