SDLS: a Matlab package for solving conic least-squares problems

Didier Henrion[1,2] Jérôme Malick[3]

Version 1.1 of 26 May 2008

Abstract

This document is an introduction to the Matlab package SDLS (Semi-Definite Least-Squares) for solving least-squares problems over convex symmetric cones. The package is shortly presented through the addressed problem, a sketch of the implemented algorithm, the syntax and calling sequences, a simple numerical example and some more advanced features. The implemented method consists in solving the dual problem with a quasi-Newton algorithm. We note that SDLS is not the most competitive implementation of this algorithm: efficient, robust, commercial implementations are available (contact the authors). Our main goal with this Matlab SDLS package is to provide a simple, user-friendly software for solving and experimenting with semidefinite least-squares problems. Up to our knowledge, no such freeware exists at this date.

1 General presentation

SDLS is a Matlab freeware solving approximately convex conic-constrained least-squares problems. Geometrically, such problems amount to finding the best approximation of a point in the intersection of a convex symmetric cone with an affine subspace. In mathematical terms, those problems can be cast as follows:

$$\begin{array}{ll} \min_x & \frac{1}{2}\|x-c\|^2 \\ \text{s.t.} & Ax = b \\ & x \in K \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$ has to be found, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ are given data and K is a convex symmetric cone. The norm appearing in the objective function is $\|x\| = \sqrt{x^T x}$, the Euclidean norm associated with the standard inner product in $\mathbb{R}^n$. Note that there exists a unique solution to this optimization problem.

In practice, K must be expressed as a direct product of linear, quadratic and semidefinite cones. It is always assumed that matrix A has full row-rank, otherwise the problem can be reformulated by eliminating redundant equality constraints.

[1]LAAS-CNRS, University of Toulouse (France)
[2]Czech Technical University in Prague (Czech Republic)
[3]CNRS, Laboratoire Jean Kunztmann, University of Grenoble (France)

2 Algorithm

SDLS is an implementation of the algorithm described in [2], which consists in solving the dual to problem (1). Specifically, the dual problem is (up to the sign and an additive constant)

$$\min_y \quad f(y) = \frac{1}{2}\|p_K(c + A^T y)\|^2 - b^T y \tag{2}$$

where $p_K(z)$ denotes the orthogonal projection of vector z onto the cone K. It is easy to prove that this dual problem is smooth and convex, so it can be solved by classical optimization algorithms. Among them, quasi-Newton algorithms are known to be efficient, and SDLS uses a Matlab implementation of the BFGS algorithm available in the Matlab freeware HANSO [1]. The other key numerical component of SDLS is eigenvalue decomposition for symmetric matrices achieved by Matlab's built-in linear algebra function `eig`. We insist on the two following features:

- **Simplicity:** the SDLS package consists of only 4 Matlab interpreted m-files calling one external package and a built-in linear algebra function.
- **Easy-to-use:** as explained in the next section, the syntax follows standard patterns.

The implementation of the algorithm of [2] in SDLS is probably not the most efficient. It is not meant to outperform neither the professional or commercial implementations of this method, nor other methods solving the same problem. Our main goal is to provide a simple, user-friendly, free software for solving and experimenting with semidefinite least-squares problems. Up to our knowledge, no such freeware exists at this date.

3 Syntax

SDLS has the same syntax as the widely used freeware SeDuMi for solving linear problems over convex symmetric cones [3]. Even though SeDuMi and SDLS share the same calling syntax, note that SeDuMi is aimed at solving a different problem. Namely, SeDuMi minimizes a linear objective function $c^T x$ while SDLS minimizes a quadratic objective function as in (1).

The basic calling syntax is:

```
>> [x,y] = sdls(A,b,c,K)
```

Input arguments `A`, `b`, `c` are real-valued matrices as in (1). SDLS exploits data sparsity, so that `A`, `b` and `c` can be Matlab objects of class `sparse`. If the third input argument is empty or not specified, then `c` is set to zero.

Input argument `K` is a structure describing the components of the cone:

- `K.f` is the number of free, i.e. unrestricted components. E.g. if `K.f=2` then `x(1:2)` is unrestricted. These are always the first components in `x`
- `K.l` is the dimension of the linear cone, i.e. the number of nonnegative components. E.g. if `K.f=2` and `K.l=8` then `x(3:10)>=0`
- `K.q` lists the dimensions of the quadratic cones. E.g. if `K.l=10` and `K.q=[3 7]` then `x(11)>=norm(x(12:13))` and `x(14)>=norm(x(15:20))`. The corresponding components of `x` immediately follow the `K.l` nonnegative ones

- `K.s` lists the dimensions of the semidefinite cones. E.g. if `K.l=20` and `K.s=[4 3]`, then `reshape(x(21:36),4,4)` and `reshape(x(37:45),3,3)` are symmetric positive semidefinite matrices. These components are always the last entries in `x`

Note that rotated quadratic cones (`K.r`) and complex Hermitian components (`K.xcomplex, K.scomplex, K.ycomplex`) are currently not supported. Please refer to SeDuMi's documentation [3] for more details on the meaning of the fields in structure `K`. If the fourth input argument is empty or not specified, then `K` is the linear cone, e.g. `K.f=size(A,2)`.

Output argument `x` (same dimension as `c`) is the solution to SDLS problem (1), whereas `y` (same dimension as `b`) is the corresponding dual vector solving the smooth problem (2).

4 Installation

SDLS 1.0 is developed for Matlab version 7.2 (release 2006). It consists of a `tar.gz` or `zip` archive that can be obtained by consulting

`www.laas.fr/∼henrion/software/sdls`

Unpacking the archive creates a subdirectory `sdls` that should be activated under your Matlab environment, either with the inline command `addpath` or with the `Set Path` entry of the `File` menu.

The package HANSO, available at

`www.cs.nyu.edu/overton/software/hanso`

must also be installed and activated under your Matlab environment.

If SDLS is correctly installed, it should generate the following output:

```
>> x = sdls([1, 0],1)
x =
     1
     0
```

5 Example

As a simple illustrative example of the use of SDLS we consider the problem of calibrating correlation matrices.

Correlation matrices are symmetric positive semidefinite matrices with ones along the diagonal. They play an important role in probability and statistics but also in combinatorial optimization. Consider a symmetric matrix C obtained from a correlation matrix after modifications of some entries, for example due to measurement errors or post-processing. Those modifications may alter the positive semidefiniteness of the matrix. We want to restore basic properties (positive semidefiniteness and ones along the diagonal) by computing the correlation matrix nearest to C in the Euclidean sense. This problem can written in the form (1).

For illustration, consider the problem of computing a nearest correlation matrix of size 3. Define

```
>> A=sparse(3,3); A(1,1)=1; A(2,5)=1; A(3,9)=1;
>> b=ones(3,1); % ones along diagonal
>> C=[1 1/2 1;1/2 1 1/4;1 1/4 1]
C =
    1.0000    0.5000    1.0000
    0.5000    1.0000    0.2500
    1.0000    0.2500    1.0000
>> c=C(:);
>> K=[]; K.s=3
```

Matrix `C` was obtained by adding `1/2` to entries `(1,3)` and `(3,1)` of a correlation matrix. Modifying these 2 entries destroyed the positive semidefiniteness:

```
>> min(eig(C))
ans =
   -0.0349
```

Then we run the calibration process using SDLS:

```
>> [x,y] = sdls(A,b,c,K);
>> X = reshape(x,K.s,K.s)
X =
    1.0000    0.4910    0.9684
    0.4910    1.0000    0.2582
    0.9684    0.2582    1.0000
```

This way, we do not recover the original correlation matrix, but we get the correlation matrix nearest to `C` in the Euclidean sense. Indeed we observe that the diagonal ones remain whereas positive semidefiniteness is restored:

```
>> min(eig(X))
ans =
   3.0097e-16
```

6 Advanced use

6.1 Input arguments

Optionally, tuning parameters can be specified as a fifth input argument:

```
>> [x,y] = sdls(A,b,c,K,pars)
```

Input argument `pars` is a structure containing the following fields:

- `pars.eps`: relative accuracy (default `1e-6`)
- `pars.fid`: screen output (0 for no output, default 1)
- `pars.maxit`: maximum number of iterations for the BFGS solver (default `200`)

- `pars.reg`: regularization parameter (default `0`)
- `pars.scaling`: scaling of the linear equation (`false` for no scaling, default `true`)

The stopping rule of the algorithm is `norm(A*x-b) <= (1+norm(b))*pars.eps` where `pars.eps` is the expected relative accuracy. Since BFGS is essentially a first-order algorithm, `pars.eps` should not be set too small. Typical values are between `1e-4` and `1e-8`, depending on the problem dimensions and scaling.

The regularization parameter `pars.reg` should be used if one suspects that the qualification constraint does not hold. That is, the affine subspace may be tangent to the cone, or equivalently there may be no point satisfying the affine constraint in the interior of the cone. A typical value is `pars.reg=1e-6`.

When `pars.scaling=true`, the scaling consists simply in dividing input data `A` and `b` by `max(1,norm(b))`. Otherwise, the input data are left unchanged.

6.2 Output arguments

An additional third output argument:

```
>> [x,y,info] = sdls(A,b,c,K)
```

can be retrieved. Argument `info` contains information on the objective function of dual problem (2), as provided by the BFGS solver:

- `info.f`: function value $f(y)$ at the optimum
- `info.g`: gradient of $f(y)$ at the optimum
- `info.H`: approximation of inverse Hessian of $f(y)$ at the optimum
- `info.time`: CPU time

The norm of the residual `norm(A*x-b)` is equal to the norm of the gradient `norm(info.g)`. It should be less than the absolute accuracy `(1+norm(b))*pars.eps` (see above) and it can be used as an estimate of the quality of the solution produced by SDLS.

References

[1] J.V. Burke, A.S. Lewis and M.L. Overton. HANSO (hybrid algorithm for non-smooth optimization): a Matlab package based on BFGS, bundle and gradient sampling methods. Version 1.0 released in 2006. See `www.cs.nyu.edu/faculty/overton/software/hanso`

[2] J. Malick. A dual approach to semidefinite least-squares problems. SIAM J. Matrix Anal. Appl. Vol. 26, No. 1, pp. 272-284, 2004.

[3] J.F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. Optimization Methods and Software. Vol. 11-12, pp. 625–653, 1999. See `sedumi.mcmaster.ca`