

# SDPLR 1.03-beta User's Guide (short version)

August 12, 2009

## 1 Summary

SDPLR is a C software package for solving large-scale semidefinite programming problems. Source code, binaries, and a Matlab interface for SDPLR can be downloaded from

<http://dollar.biz.uiowa.edu/~sburer/software/SDPLR/>.

SDPLR accepts the standard sparse SDPA format as well as a special format that handles low-rank data matrices. People involved with the development of SDPLR are

- Samuel Burer (samuel-burer@uiowa.edu)
- Renato D.C. Monteiro (monteiro@isye.gatech.edu)
- Changhui Choi (changhui.choi@ucdenver.edu)

## 2 Installation from Source

It is assumed that the user has downloaded and unpacked the SDPLR source files.

### 2.1 Matlab

Within Matlab, run *mexinstall.m* from the main SDPLR directory. Then type *help sdplr* for instructions. Add the SDPLR directory to your Matlab path.

**Note to Windows users.** The top two lines of *mexinstall.m* may need to be adjusted for the location of the Matlab's LAPACK and BLAS libraries on your system.

### 2.2 Command line (UNIX, Linux, etc.)

#### Step 1

Make sure BLAS and LAPACK are installed on your system and determine how to link them with other programs. The linking information will be required for the LIB and LIB\_DIRS options in *Makefile.inc* (see Step 2).

## Step 2

Tailor the file *Makefile.inc* to your system; *Makefile.inc.linux* is provided as an example. There are three sections:

- Specify the location and switch for the LAPACK library using `LAPACK_LIB_DIR` and `LAPACK_LIB`.
- Specify the location and switch for the BLAS library using `BLAS_LIB_DIR` and `BLAS_LIB`.
- Specify the compiling options:
  - `CC`: Specify the compiler, e.g., `gcc` or `cc`.
  - `CFLAGS`: Specify any flags to pass to the compiler and linker, e.g., optimization flags such as `-O3`.
  - `LIB_DIRS`: If necessary, using the `-L` switch, specify any directory locations for additional libraries that need to be linked in. Leave items involving LAPACK and BLAS unchanged. It is not necessary to include the SDPLR *lib* directory since it is automatically specified.
  - `LIBS`: Specify the flags for the libraries that will be linked in. This should definitely include `-lgsl`, which is included with and required by SDPLR, and the flags for the BLAS and LAPACK libraries from Step 1. Remember that linking order is important! For example, on an Ubuntu 9.04 Linux system, the line might read

```
LIBS = -lgsl -llapack -lblas -lgfortran -lm
```

Here, `-lgfortran` indicates a library that is required by LAPACK and BLAS, and `-lm` is the standard math library.

## Step 3

Type `make` to compile the program. Then type `sdplr vibra1.dat-s` to test the installation. Your output should be similar to the file *vibra1.dat-s.out* provided in the installation directory. Typing `make clean` will remove all object files, and typing `make cleanall` will remove the SDPLR executable and GSL library as well.

## 2.3 Windows (via MinGW)

MinGW is a GNU compiling environment for Windows. To compile SDPLR, you will need

- The MinGW base system (including the *make* utility). Add MinGW's *bin* directory to your Windows path.
- MinGW's *gcc* tools. It is suggested to install the latest *gcc* release with additional packages such as *gfortran*.
- Working LAPACK and BLAS libraries within MinGW. The following guide is suggested: <http://gcc.gnu.org/wiki/LAPACK%20on%20Windows>.

Then tailor *Makefile.inc* as described above; *Makefile.inc.mingw* is provided as an example. Finally, type *mingw32-make mingw*.

## 3 Usage

### 3.1 Matlab

Make sure that the Matlab binary for SDPLR and the file *sdplr.m* are in your Matlab path. Then type *help sdplr* for more information.

### 3.2 Command line

The syntax for SDPLR is

```
Usage #1: sdplr <input_file> [params_file] [soln_in] [soln_out]
Usage #2: sdplr gen_params
```

Regarding usage #1:

- If `params_file` is not specified, then a set of default parameters is used.
- Both files `soln_in` and `soln_out` refer to a particular file format generated and used by SDPLR.
- If `soln_in` is specified, then `params_file` must be specified.
- If `soln_out` is specified, then both `params_file` and `soln_in` must be specified. However, a dummy filename for `soln_in` may be used, allowing one to save an out-file without an in-file.

Regarding usage #2: This can be used to automatically generate a valid `params_file` for SDPLR. The user is prompted to enter each parameter value, and detailed explanations of the parameters can be gotten by just typing 'i'.

## 4 Input Formats

By default, SDPLR accepts the sparse SDPA format, which is explained at

<http://www.nmt.edu/~sdplib/FORMAT>

However, some SDPs are not sparse but still have a great deal of structure. The SDPLR format, which SDPLR is also capable of handling, is more or less an extension of sparse SDPA format which also allows low-rank data matrices to be specified easily.

The structure of the SDPLR format is as follows:

- The first line contains  $m$ , the number of constraint matrices.

- The second line contains  $k$ , the number of blocks in the SDP.
- The next  $k$  lines contain the sizes  $n_1, \dots, n_k$  of the blocks, where a negative  $n_j$  indicates a diagonal block.
- The next line contains  $b$ , the right-hand side vector, all on one line.
- The next line is currently ignored, but should contain a number. (This line is related to a feature of SDPLR which is in development. Basically, for +/-1 combinatorial optimization SDPs, this line will contain the trace of the primal matrix (a constant), which can be used to provide dual bounds during the execution of SDPLR.)
- The remaining portion of the file is divided into  $(m + 1)k$  sections, giving the  $k$  blocks of the  $m + 1$  objective and constraint data matrices. The matrices should be listed in order, i.e., the objective matrix first and then the constraint matrices in order, and within each matrix, the blocks should be listed in order.
- Two types of blocks can be specified for the data matrices: sparse and low-rank.
- If the  $j$ -th block of  $A_i$  is a sparse data matrix, then the section is as follows:
  - The first line contains the matrix number  $i$  ( $i = 0$  for objective), the block number  $j$ , the character **s** to indicate ‘sparse,’ and the number **nnz**, which is the number nonzeros in the upper triangular part of  $A_i$ .
  - The next **nnz** lines contain the entries in “ $i \ j \ \text{entry}$ ” format, where  $i \leq j$ .
- If the  $j$ -th block of  $A_i$  is a low-rank data matrix of rank  $r$ , then a factorization  $BDB^T$  of this block must be known such that  $B \in \Re^{n_j \times r}$  and  $D \in \Re^{r \times r}$  is diagonal. Then the section specifying the block is as follows:
  - The first line contains the matrix number (0 for objective), the block number  $j$ , the character **l** to indicate ‘low-rank,’ and the number  $r$ , which is the rank of the  $j$ -th block of  $A_i$ .
  - The next  $r$  lines contain the diagonal entries of  $D$  in order.
  - The next  $n_j r$  lines contain the entries of  $B$  in column-major format. Note that  $B$  is specified as dense.

A brief example of the SDPLR format is the following, which encodes the Lovász theta SDP for the 5-cycle (more examples available on the SDPLR website):

```

6
1
5
0.0 0.0 0.0 0.0 0.0 1.0
-1.0
0 1 1 1
-1.0

```

1.0  
1.0  
1.0  
1.0  
1.0  
1 1 s 1  
1 2 1.0  
2 1 s 1  
2 3 1.0  
3 1 s 1  
3 4 1.0  
4 1 s 1  
4 5 1.0  
5 1 s 1  
1 5 1.0  
6 1 s 5  
1 1 1.0  
2 2 1.0  
3 3 1.0  
4 4 1.0  
5 5 1.0