

# SLSPack: an Interface Library for Sparse Linear Solver Packages

VERSION 1.0

Hui Liu  
2018

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview . . . . .	1
1.2	License . . . . .	1
1.3	Citation . . . . .	1
1.4	Website . . . . .	2
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Configuration . . . . .	3
2.2	Options . . . . .	3
2.3	Compilation . . . . .	7
2.4	Installation . . . . .	7
2.5	Optional Packages . . . . .	7
2.5.1	BLAS . . . . .	7
2.5.2	LAPACK . . . . .	8
2.5.3	LASPack . . . . .	8
2.5.4	SuiteSparse . . . . .	9
2.5.5	MUMPS . . . . .	9
2.5.6	PETSC . . . . .	10
2.5.7	LIS . . . . .	10
2.5.8	FASP . . . . .	11
2.5.9	SUPERLU . . . . .	11
2.5.10	PARDISO . . . . .	12
2.5.11	HSL MI20 (AMG) . . . . .	12
2.5.12	SXAMG . . . . .	12
<b>3</b>	<b>Matrix and Vector</b>	<b>15</b>
3.1	Matrix Definition . . . . .	15
3.2	Matrix Management . . . . .	15
3.2.1	Initialize . . . . .	15
3.2.2	Create . . . . .	16
3.2.3	Destroy . . . . .	16
3.3	Vector Definition . . . . .	16
3.4	Vector Management . . . . .	16
3.4.1	Create . . . . .	16

3.4.2	Destroy . . . . .	16
3.4.3	Set Value . . . . .	17
3.4.4	Get Value . . . . .	17
3.4.5	Copy . . . . .	17
<b>4</b>	<b>Linear Solvers</b>	<b>19</b>
4.1	Solver Types . . . . .	19
4.2	Solver Management . . . . .	20
4.2.1	Create . . . . .	20
4.2.2	Assemble . . . . .	22
4.2.3	Solve . . . . .	22
4.2.4	Destroy . . . . .	22
4.3	Solver Settings . . . . .	22
4.3.1	General Settings . . . . .	22
4.3.2	AMG Solver Setting . . . . .	23
4.3.3	FASP Solver Setting . . . . .	23
4.3.4	LIS Solver Setting . . . . .	23
4.3.5	SXAMG Setting . . . . .	24
4.3.6	PETSc Setting . . . . .	25
<b>5</b>	<b>Utilities</b>	<b>27</b>
5.1	Print . . . . .	27
5.2	Memory . . . . .	27
5.3	Performance . . . . .	28

# Chapter 1

## Introduction

### 1.1 Overview

SLSPack is an interface library for famous linear solver packages. The library is designed for Linux, Unix and Mac systems. It is also possible to compile under Windows. The code is written by C, and it is serial.

SLSPack has interfaces to many popular packages, such as PETSc, MUMPS, FASP, UMFPACK (SUITESPARSE), KLU (SUITESPARSE), LSPACK, LIS, PARDISO, SUPERLU, and HSL MI20.

### 1.2 License

The package uses GPL license. If you have any issue, please contact: [hui.sc.liu@gmail.com](mailto:hui.sc.liu@gmail.com)

### 1.3 Citation

If you like our SLSPack library, you may cite it like this,

```
@misc{slspack-library,  
  author="Hui Liu",  
  title="SLSPack: an Interface Library for Sparse Linear Solver Packages",  
  year="2018",  
  note={\url{https://github.com/huiscliu/slspack/}}  
}
```

## 1.4 Website

The official website for SLSPack is <https://github.com/huiscliu/slspack/>.

# Chapter 2

## Installation

SLSPack has interfaces to some external packages, such as PETSc, MUMPS, FASP, UMFPACK (SuiteSparse), KLU (SuiteSparse), LAPACK, LIS, PARDISO, SUPERLU, and HSL MI20. All these packages are optional. Most packages are enabled by default. However, they will be disabled if not found by configuration script.

SLSPack uses `autoconf` and `make` to detect these packages and system parameters, to build and to install.

### 2.1 Configuration

The simplest way to configure is to run command:

```
./configure
```

This command will try to find optional packages from certain directories. Searching details can be read from `configure.in` and some are explained below. It also sets system parameters.

### 2.2 Options

The script `configure` has many options, if user would like to check, run command:

```
./configure --help
```

Output will be like this,

```
'configure' configures this package to adapt to many kinds of systems.
```

```
Usage: ./configure [OPTION]... [VAR=VALUE]...
```

To assign environment variables (e.g., CC, CFLAGS...), specify them as VAR=VALUE. See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

#### Configuration:

-h, --help	display this help and exit
--help=short	display options specific to this package
--help=recursive	display the short help of all the included packages
-V, --version	display version information and exit
-q, --quiet, --silent	do not print 'checking ...' messages
--cache-file=FILE	cache test results in FILE [disabled]
-C, --config-cache	alias for '--cache-file=config.cache'
-n, --no-create	do not create output files
--srcdir=DIR	find the sources in DIR [configure dir or '..']

#### Installation directories:

--prefix=PREFIX	install architecture-independent files in PREFIX [/usr/local/slspack]
--exec-prefix=EPREFIX	install architecture-dependent files in EPREFIX [PREFIX]

By default, 'make install' will install all the files in '/usr/local/slspack/bin', '/usr/local/slspack/lib' etc. You can specify an installation prefix other than '/usr/local/slspack' using '--prefix', for instance '--prefix=HOME'.

For better control, use the options below.

#### Fine tuning of the installation directories:

--bindir=DIR	user executables [EPREFIX/bin]
--sbindir=DIR	system admin executables [EPREFIX/sbin]
--libexecdir=DIR	program executables [EPREFIX/libexec]
--sysconfdir=DIR	read-only single-machine data [PREFIX/etc]
--sharedstatedir=DIR	modifiable architecture-independent data [PREFIX/com]
--localstatedir=DIR	modifiable single-machine data [PREFIX/var]
--libdir=DIR	object code libraries [EPREFIX/lib]
--includedir=DIR	C header files [PREFIX/include]
--oldincludedir=DIR	C header files for non-gcc [/usr/include]
--datarootdir=DIR	read-only arch.-independent data root [PREFIX/share]
--datadir=DIR	read-only architecture-independent data [DATAROOTDIR]
--infodir=DIR	info documentation [DATAROOTDIR/info]
--localedir=DIR	locale-dependent data [DATAROOTDIR/locale]
--mandir=DIR	man documentation [DATAROOTDIR/man]
--docdir=DIR	documentation root [DATAROOTDIR/doc/PACKAGE]
--htmldir=DIR	html documentation [DOCDIR]



## 2.2. Options

---

```
--dvidir=DIR          dvi documentation [DOCDIR]
--pdfdir=DIR          pdf documentation [DOCDIR]
--psdir=DIR           ps documentation [DOCDIR]
```

### System types:

```
--build=BUILD        configure for building on BUILD [guessed]
--host=HOST           cross-compile to build programs to run on HOST [BUILD]
```

### Optional Features:

```
--disable-option-checking  ignore unrecognized --enable/--with options
--disable-FEATURE          do not include FEATURE (same as --enable-FEATURE=no)
--enable-FEATURE[=ARG]    include FEATURE [ARG=yes]
--enable-rpath             enable use of rpath (default)
--disable-rpath            disable use of rpath
--with-rpath-flag=FLAG    compiler flag for rpath (e.g., "-Wl,-rpath,")
--disable-assert          turn off assertions
--enable-blas              enable BLAS support (default)
--disable-blas             disable BLAS support
--with-blas=blas BLAS lib
--enable-lapack            enable LAPACK support (default)
--disable-lapack           disable LAPACK support
--with-lapack=lapack LAPACK lib
--enable-laspack           enable LASPACK support (default)
--disable-laspack          disable LASPACK support
--with-laspack-libdir=DIR  path for LASPACK library
--with-laspack-incdir=DIR  path for LASPACK header file
--enable-ssparsc           enable SSPARSE support (default)
--disable-ssparsc          disable SSPARSE support
--with-ssparsc-libdir=DIR  path for SSPARSE library
--with-ssparsc-incdir=DIR  path for SSPARSE header file
--enable-mumps             enable MUMPS solver (default)
--disable-mumps            disable MUMPS solver
--with-mumps-incdir=DIR    MUMPS header files directory
--with-mumps-libdir=DIR    MUMPS libraries directory
--enable-petsc             enable PETSC solver (default)
--disable-petsc            disable PETSC solver
--with-petsc-incdir=DIR    PETSC header files directory
--with-petsc-libdir=DIR    PETSC libraries directory
--enable-lis               enable LIS support (default)
--disable-lis              disable LIS support
--with-lis-libdir=DIR      path for LIS library
--with-lis-incdir=DIR      path for LIS header file
--enable-fasp              enable FASP support (default)
--disable-fasp             disable FASP support
--with-fasp-libdir=DIR     path for FASP library
--with-fasp-incdir=DIR     path for FASP header file
--enable-superlu           enable SUPERLU support (default)
```

```

--disable-superlu      disable SUPERLU support
--with-superlu-libdir=DIR path for SUPERLU library
--with-superlu-incdir=DIR path for SUPERLU header file
--enable-pardiso       enable PARDISO support (default)
--disable-pardiso      disable PARDISO support
--with-pardiso-libdir=DIR path for PARDISO library
--with-pardiso-incdir=DIR path for PARDISO header file
--enable-hslmi20       enable HSL_MI20 support (default)
--disable-hslmi20      disable HSL_MI20 support
--with-hslmi20-libdir=DIR path for HSL_MI20 library
--with-hslmi20-incdir=DIR path for HSL_MI20 header file

```

Some influential environment variables:

CC	C compiler command
CFLAGS	C compiler flags
LDFLAGS	linker flags, e.g. -L<lib dir> if you have libraries in a nonstandard directory <lib dir>
LIBS	libraries to pass to the linker, e.g. -l<library>
CPPFLAGS	(Objective) C/C++ preprocessor flags, e.g. -I<include dir> if you have headers in a nonstandard directory <include dir>
CXX	C++ compiler command
CXXFLAGS	C++ compiler flags
FC	Fortran compiler command
FCFLAGS	Fortran compiler flags
CPP	C preprocessor

Use these variables to override the choices made by ‘configure’ or to help it to find libraries and programs with nonstandard names/locations.

The options follow the same convention,

- `--enable-pack`, to enable package `pack`, such as `--enable-itsol`;
- `--disable-pack`, to disable package `pack`, such as `--disable-itsol`;
- `--with-pack-libdir=DIR`, to set `DIR` as the library path of package `pack`, such as `--with-itsol-libdir=/usr/local/itsol/lib/`;
- `--with-pack-incdir=DIR`, to set `DIR` as the include path of package `pack`, such as `--with-itsol-incdir=/usr/local/itsol/include/`;

The configuration script tries to find package from `/usr/local/`, and `/opt/`, such as `/usr/local/itsol/`, and it tries to set correct include path, library path, and specific libraries. However, if configure cannot find correct information, users can help configure by using options.

## 2.3 Compilation

After configuration, `Makefile` and related scripts will be set correctly. A simple `make` command can compile the package,

```
make
```

## 2.4 Installation

Run command:

```
make install
```

The package will be installed to a directory. The default is `/usr/local/slspack/`. A different directory can be set by `--prefix=DIR`.

## 2.5 Optional Packages

### 2.5.1 BLAS

The BLAS (Basic Linear Algebra Subprograms) are routines that provide standard building blocks for performing basic vector and matrix operations. The Level 1 BLAS perform scalar, vector and vector-vector operations, the Level 2 BLAS perform matrix-vector operations, and the Level 3 BLAS perform matrix-matrix operations. Because the BLAS are efficient, portable, and widely available, they are commonly used in the development of high quality linear algebra software, LAPACK for example.

Official website: <http://www.netlib.org/blas/>

BLAS search directories:

```
/usr/local/lib
/usr/local/lib64
/usr/local/blas/lib
/usr/local/blas*/lib
/usr/local/blas/
/usr/local/blas*/
/usr/lib
/usr/lib64
/opt/blas/lib
/opt/blas*/lib
```

### 2.5.2 LAPACK

LAPACK is written in Fortran 90 and provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

Official website: <http://www.netlib.org/lapack/>

LAPACK search directories:

```
/usr/local/lib
/usr/local/lib64
/usr/local/lapack/lib
/usr/local/lapack*/lib
/usr/local/lapack/
/usr/local/lapack*/
/usr/lib
/usr/lib64
/opt/lapack/lib
/opt/lapack*/lib
```

### 2.5.3 LASPack

LASPack is a package for solving large sparse systems of linear equations like those which arise from discretization of partial differential equations. It contains classical as well as selected state-of-the-art algorithms which are commonly used for large sparse systems such as CG-like methods for non-symmetric systems (CGN, GMRES, BiCG, QMR, CGS, and BiCGStab) and multilevel methods such as multigrid and conjugate gradient method preconditioned by multigrid and BPX preconditioners. LASPack is written in ANSI C and is thus largely portable.

Official website: [http://www.netlib.org/utk/misc/sw\\_survey/urc/html/LASPack.1.html](http://www.netlib.org/utk/misc/sw_survey/urc/html/LASPack.1.html)

LASPack search directories (include and lib):

```
/usr/local/laspack/
/usr/local/laspack*/
/usr/local/
/usr/
/opt/laspack/
/opt/laspack*
```

The include and lib directories are sub-directories of above directories, such as /usr/local/las-

## 2.5. Optional Packages

---

pack/include and /usr/local/laspack/lib. Users can also set customized directories with options: `-with-laspack-libdir=DIR` and `-with-laspack-incdir=DIR`.

### 2.5.4 SuiteSparse

A Suite of Sparse matrix software, including UMFPACK, CHOLMOD, SPQR, KLU, BTF, and ordering methods (AMD, CAMD, COLAMD, and CCOLAMD).

Official website: <https://github.com/jluttine/suitesparse>

Official website: <http://www.suitesparse.com>

SuiteSparse search directories:

```
/usr/local/SuiteSparse
/usr/local/SuiteSparse*
/usr/local
/usr
/opt/SuiteSparse
/opt/SuiteSparse*
```

The include and lib directories are sub-directories of above directories. Users can also set customized directories with options.

### 2.5.5 MUMPS

MUMPS (MULTifrontal Massively Parallel sparse direct Solver) is a software application for the solution of large sparse systems of linear algebraic equations on distributed memory parallel computers. It was developed in European project PARASOL (1996–1999) by CERFACS, IRIT-ENSEEIH and RAL. The software implements the multifrontal method, which is a version of Gaussian elimination for large sparse systems of equations, especially those arising from the finite element method. It is written in Fortran 90 with parallelism by MPI and it uses BLAS and ScaLAPACK kernels for dense matrix computations. Since 1999, MUMPS has been supported by CERFACS, IRIT-ENSEEIH, and INRIA.

Official website: <http://mumps.enseeiht.fr/>

MUMPS search directories:

```
/usr/local/mumps-seq/
/usr/local/mumps*-seq/
/usr/local/
/usr/
/opt/mumps-seq/
/opt/mumps*-seq/
```

The include and lib directories are sub-directories of above directories. Users can also set customized directories with options.

### 2.5.6 PETSC

PETSc, pronounced PET-see (the S is silent), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It supports MPI, and GPUs through CUDA or OpenCL, as well as hybrid MPI-GPU parallelism. PETSc is intended for use in large-scale application projects, many ongoing computational science projects are built around the PETSc libraries. PETSc is easy to use for beginners. Moreover, its careful design allows advanced users to have detailed control over the solution process. PETSc includes a large suite of parallel linear, nonlinear equation solvers and ODE integrators that are easily used in application codes written in C, C++, Fortran and now Python. PETSc provides many of the mechanisms needed within parallel application codes, such as simple parallel matrix and vector assembly routines that allow the overlap of communication and computation. In addition, PETSc includes support for parallel distributed arrays useful for finite difference methods.

Official website: <https://www.mcs.anl.gov/petsc/>

PETSC search directories:

```
/usr/local/petsc-seq
/usr/local/petsc*-seq/
/opt/petsc-seq/
/opt/petsc*-seq/
```

The include and lib directories are sub-directories of above directories. Users can also set customized directories with options.

### 2.5.7 LIS

Lis (Library of Iterative Solvers for linear systems, pronounced [lis]) is a parallel software library for solving linear equations and eigenvalue problems that arise in the numerical solution of partial differential equations using iterative methods.

Official website: <http://www.ssisc.org/lis/>

LIS search directories:

```
/usr/local/lis
/usr/local/lis*
/usr/local
/usr
/opt/lis
```

## 2.5. Optional Packages

---

```
/opt/lis*
```

The include and lib directories are sub-directories of above directories. Users can also set customized directories with options.

### 2.5.8 FASP

FASP team plans to construct a pool of discrete problems arising from partial differential equations (PDEs) or PDE systems and efficient linear solvers for these problems. They mainly utilize the methodology of Auxiliary Space Preconditioning (ASP) to construct efficient linear solvers. A set of Krylov solvers and AMG solvers have been implemented.

Official website: <http://fasp.sourceforge.net/>

FASP search directories:

```
/usr/local/fasp  
/usr/local/fasp*  
/usr/local  
/usr  
/opt/fasp  
/opt/fasp*
```

The include and lib directories are sub-directories of above directories. Users can also set customized directories with options.

### 2.5.9 SUPERLU

SuperLU is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library is written in C and is callable from either C or Fortran. The library routines will perform an LU decomposition with partial pivoting and triangular system solves through forward and back substitution. The LU factorization routines can handle non-square matrices but the triangular solves are performed only for square matrices.

Official website: <http://crd-legacy.lbl.gov/~xiaoye/SuperLU/>

SUPERLU search directories:

```
/usr/local/superlu  
/usr/local/superlu*  
/usr/local  
/usr  
/opt/superlu  
/opt/superlu*
```

The include and lib directories are sub-directories of above directories. Users can also set customized directories with options.

### 2.5.10 PARDISO

The package PARDISO is a thread-safe, high-performance, robust, memory efficient and easy to use software for solving large sparse symmetric and unsymmetric linear systems of equations on shared-memory and distributed-memory multiprocessors.

Official website: <http://www.pardiso-project.org/>

PARDISO search directories:

```
/usr/local/intel
/opt/intel

/usr/local/pardiso
/usr/local/pardiso*
/usr/local/lib
/usr/local/lib64
/usr/lib
/usr/lib64
/opt/pardiso
/opt/pardiso*
```

### 2.5.11 HSL MI20 (AMG)

An AMG package using classical method.

Official website: [http://www.hsl.rl.ac.uk/catalogue/hsl\\_mi20.html](http://www.hsl.rl.ac.uk/catalogue/hsl_mi20.html)

HSL MI20 search directories:

```
/usr/local/hsl_mi20
/usr/local/hsl_mi20*
/opt/hsl_mi20
/opt/hsl_mi20*
```

### 2.5.12 SXAMG

An AMG package using classical method.

Official website: <https://github.com/huiscliu/sxamg/>



## 2.5. Optional Packages

---

SXAMG search directories:

```
/usr/local/sxamg  
/usr/local/sxamg*  
/opt/sxamg  
/opt/sxamg*
```

The include and lib directories are sub-directories of above directories. Users can also set customized directories using options.



# Chapter 3

## Matrix and Vector

### 3.1 Matrix Definition

SLSPack uses `int` for integer and `double` for floating-point number. In this library, matrix indices and array indices follow C style, which start from 0.

```
typedef struct slspack_mat_csr_  
{  
    int num_rows;  
    int num_cols;  
    int num_nnz;  
    int *Ap;  
    int *Aj;  
    double *Ax;  
}  
slspack_mat_csr;
```

The definition of `slspack_mat_csr` is the same as standard definition.

### 3.2 Matrix Management

#### 3.2.1 Initialize

`slspack_mat_init` initializes a matrix, which sets row, column and non-zero to zero and set arrays to NULL.

```
void slspack_mat_init(slspack_mat_csr *A);
```

### 3.2.2 Create

`slspack_mat_create` creates a CSR matrix using user input.

```
slspack_mat_csr slspack_mat_create(int nrows, int ncols, int *Ap, int *Aj, double *Ax);
```

### 3.2.3 Destroy

`slspack_mat_destroy` destroys a matrix object and releases memory.

```
void slspack_mat_destroy(slspack_mat_csr *csr);
```

## 3.3 Vector Definition

```
typedef struct slspack_vec_  
{  
    int n;  
    double *d;  
}  
slspack_vec;
```

`slspack_vec` has two members, which are vector length (`n`) and data (memory, `d`).

## 3.4 Vector Management

### 3.4.1 Create

`slspack_vec_create` creates a length `n` floating-point vector.

```
slspack_vec slspack_vec_create(int n);
```

### 3.4.2 Destroy

`slspack_vec_destroy` destroys a vector.

```
void slspack_vec_destroy(slspack_vec *v);
```

### 3.4.3 Set Value

`slspack_vec_set_value`, `slspack_vec_set_value_by_array` and `slspack_vec_set_value_by_index` set vector values.

`slspack_vec_set_value` sets the vector to the same value.

```
void slspack_vec_set_value(slspack_vec x, double val);
```

`slspack_vec_set_value_by_array` sets vector value by a buffer, which has the same length as vector.

```
void slspack_vec_set_value_by_array(slspack_vec x, double *val);
```

`slspack_vec_set_value_by_index` sets value to the  $i$ -th component,  $x[i] = val$ .

```
void slspack_vec_set_value_by_index(slspack_vec x, int i, double val);
```

### 3.4.4 Get Value

`slspack_vec_get_value` copies vector's values to a buffer, which should have the same length as the vector.

```
void slspack_vec_get_value(double *val, slspack_vec x);
```

`slspack_vec_get_value_by_index` gets the value of the  $i$ -th component.

```
double slspack_vec_get_value_by_index(slspack_vec x, int i);
```

### 3.4.5 Copy

`slspack_vec_copy` copies data from source to destination.

```
void slspack_vec_copy(slspack_vec des, const slspack_vec src);
```



# Chapter 4

## Linear Solvers

### 4.1 Solver Types

The solver type is defined as `SLSPACK_SOLVER_TYPE`,

```
/* solver type */
typedef enum SLSPACK_SOLVER_TYPE_
{
#ifdef USE_LASPACK
    SLSPACK_SOLVER_LASPACK,    /* laspack */
#endif

#ifdef USE_SSPARSE
    SLSPACK_SOLVER_UMFPACK,    /* ssparse, umf */
    SLSPACK_SOLVER_KLU,        /* ssparse, klu */
#endif

#ifdef USE_MUMPS
    SLSPACK_SOLVER_MUMPS,      /* mumps */
#endif

#ifdef USE_PETSC
    SLSPACK_SOLVER_PETSC,      /* petsc */
#endif

#ifdef USE_LIS
    SLSPACK_SOLVER_LIS,        /* lis */
#endif

#ifdef USE_FASP
    SLSPACK_SOLVER_FASP,       /* FASP */
    SLSPACK_SOLVER_AMG,        /* amg from FASP */

```

```

    SLSPACK_SOLVER_FMG,      /* fmg from FASP */
#endif

#if USE_SUPERLU
    SLSPACK_SOLVER_SUPERLU, /* superlu */
#endif

#if USE_PARDISO
    SLSPACK_SOLVER_PARDISO, /* pardiso */
#endif

#if USE_HSL_MI20
    SLSPACK_SOLVER_MI20AMG, /* MI20 AMG */
#endif

#if USE_SX_AMG
    SLSPACK_SOLVER_SXAMG,   /* SXAMG */
#endif

} SLSPACK_SOLVER_TYPE;

```

SLSPack implements interfaces to other famous linear solver packages, such as LAPACK, SSPARSE, MUMPS, PETSC, LIS, FASP, SUPERLU, PARDISO, HSL\_MI20 and SXAMG. SLSPack also has two internal solvers, GMRES(m) and BICGSTAB.

## 4.2 Solver Management

Figure 4.1 shows solution process, which includes the following steps:

1. create solver;
2. change default solver parameters, which is optional;
3. assemble solver;
4. solve the linear system;
5. destroy solver, matrix and vectors;

### 4.2.1 Create

`slspack_solver_create` creates solver object using solver type.

```
void slspack_solver_create(SLSPACK_SOLVER *s, SLSPACK_SOLVER_TYPE s_type);
```



## 4.2. Solver Management

---

```
{
    slspack_mat_csr A;
    SLSPACK_SOLVER solver;
    slspack_vec x;
    slspack_vec b;

    /* setup A, x, b */
    x = slspack_vec_create(n);
    b = slspack_vec_create(n);

    A = slspack_mat_create(nrows, ncols, Ap, Aj, Ax);
    for (i = 0; i < n; i++) {
        slspack_vec_set_value_by_index(x, i, v1);
        slspack_vec_set_value_by_index(b, i, v2);
    }

    /* 1: create solver: any detected solver works */
    slspack_solver_create(solver, SLSPACK_SOLVER_AMG);

    /* 2: change default settings (optional) */
    slspack_solver_set_restart(solver, m);
    slspack_solver_set_maxit(solver, itr_max);

    /* 3: assemble solver */
    slspack_solver_assemble(solver, A, x, b);

    /* 4: solve */
    slspack_solver_solve(solver);

    /* 5: destroy solver */
    slspack_solver_destroy(solver);

    /* get value */
    for (i = 0; i < n; i++) {
        value = slspack_vec_get_value_by_index(x, i);
    }

    slspack_mat_destroy(A);
    slspack_vec_destroy(x);
    slspack_vec_destroy(b);
}
```

Figure 4.1: Solution process

### 4.2.2 Assemble

`slspack_solver_assemble` assembles solver object.

```
void slspack_solver_assemble(SLSPACK_SOLVER *s, slspack_mat_csr Ax, slspack_vec x,
    slspack_vec b);
```

### 4.2.3 Solve

`slspack_solver_solve` solves linear system.

```
int slspack_solver_solve(SLSPACK_SOLVER *solver);
```

### 4.2.4 Destroy

`slspack_solver_destroy` destroys solver object and releases internal memory.

```
void slspack_solver_destroy(SLSPACK_SOLVER *s);
```

## 4.3 Solver Settings

These setting functions should be applied after the solver object is created and before it is assembled.

### 4.3.1 General Settings

`slspack_solver_set_rtol` sets relative tolerance.

```
void slspack_solver_set_rtol(SLSPACK_SOLVER *s, double tol);
```

`slspack_solver_set_atol` sets absolute tolerance.

```
void slspack_solver_set_atol(SLSPACK_SOLVER *s, double tol);
```

`slspack_solver_set_rbtol` sets relative b norm tolerance.

```
void slspack_solver_set_rbtol(SLSPACK_SOLVER *s, double tol);
```

```
/* set maximal number of iteration */
```

```
void slspack_solver_set_maxit(SLSPACK_SOLVER *s, int maxit);
```

### 4.3. Solver Settings

---

[slspack\\_solver\\_set\\_restart](#) set the number of restart.

```
void slspack_solver_set_restart(SLSPACK_SOLVER *s, int m);
```

[slspack\\_solver\\_set\\_verbosity](#) sets verbosity of a solver.

```
void slspack_solver_set_verbosity(SLSPACK_SOLVER *s, int v);
```

[slspack\\_solver\\_get\\_residual](#) gets residual.

```
double slspack_solver_get_residual(SLSPACK_SOLVER s);
```

[slspack\\_solver\\_get\\_nits](#) gets number of iteration.

```
int slspack_solver_get_nits(SLSPACK_SOLVER s);
```

#### 4.3.2 AMG Solver Setting

[slspack\\_solver\\_amg\\_set\\_pars](#) sets new parameters to AMG solver.

```
void slspack_solver_amg_set_pars(SLSPACK_SOLVER *s, AMG_param par);
```

#### 4.3.3 FASP Solver Setting

[slspack\\_fasp\\_set\\_pars](#) sets parameters. If parameter pointer is not [NULL](#), default parameters will be overridden.

```
void slspack_fasp_set_pars(SLSPACK_SOLVER *solver, input_param *inparam,  
    ITS_param *itsparam, AMG_param *amgparam, ILU_param *iluparam,  
    SWZ_param *schparam);
```

#### 4.3.4 LIS Solver Setting

[slspack\\_solver\\_lis\\_set\\_pars](#) sets solver id and preconditioner id.

```
static const char * lis_solver[] = {  
    "-i bicgstab",  
    "-i bicgstabl",  
    "-i cg",  
    "-i cgs",  
    "-i bicg",  
    "-i bicgsafe",
```

```

    "-i bicr",
    "-i cr",
    "-i bicrstab",
    "-i bicrsafe",
    "-i idrs",
    "-i crs",
    "-i gpbicr",
    "-i gpbicg",
    "-i tfqmr",
    "-i orthomin",
    "-i gmres",
    "-i fgmres",
    "-i minres",
};

static const char * lis_pc[] = {
    "-p none",
    "-p ilut",
    "-p ilu -ilu_fill 1",
    "-p is",
    "-p sainv",
    "-p saamg -saamg_unsym -saamg_theta 0.5",
    "-p hybrid",
    "-p iluc",
    "-p ssor",
    "-p jacobi",
};

void slspack_solver_lis_set_pars(SLSPACK_SOLVER *solver, unsigned int solver_id,
    unsigned int pc_id);

```

[slspack\\_solver\\_lis\\_set\\_option](#) sets any legal LIS options.

```
void slspack_solver_lis_set_option(SLSPACK_SOLVER *solver, char *o);
```

### 4.3.5 SXAMG Setting

[slspack\\_solver\\_sxamg\\_set\\_pars](#) sets parameters.

```
void slspack_solver_sxamg_set_pars(SLSPACK_SOLVER *solver, SX_AMG_PARS *pars);
```

### 4.3.6 PETSc Setting

[slspack\\_solver\\_petsc\\_setting](#) sets PETSc parameters by a function.

```
typedef void (*solver_petsc_setting)(void *ksp, void *pc);  
  
void slspack_solver_petsc_setting(solver_petsc_setting func);
```



# Chapter 5

## Utilities

### 5.1 Print

`slspack_printf` outputs to stdout.

```
int slspack_printf(const char *fmt, ...);
```

`slspack_error` prints output error message and quits with error code.

```
void slspack_error(int code, const char *fmt, ...);
```

`slspack_warning` print warning info.

```
void slspack_warning(const char *fmt, ...);
```

### 5.2 Memory

The following functions provide memory allocation, calloc, reallocation, freeing and copying.

```
void * slspack_malloc(size_t n);  
void * slspack_calloc(size_t n);  
void slspack_free(void *p);  
  
void slspack_memcpy_on(void *dst, const void *src, size_t n);  
void * slspack_copy_on(const void *src, size_t n);
```

## 5.3 Performance

`slspack_get_time` gets current time point.

```
double slspack_get_time();
```



# Bibliography

- [1] C. L. Lawson, R. J. Hanson, D. Kincaid, and F. T. Krogh, Basic Linear Algebra Subprograms for FORTRAN usage, *ACM Trans. Math. Soft.*, 5 (1979), pp. 308—323.
- [2] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, An extended set of FORTRAN Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.*, 14 (1988), pp. 1—17.
- [3] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, Algorithm 656: An extended set of FORTRAN Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.*, 14 (1988), pp. 18—32.
- [4] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, A set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.*, 16 (1990), pp. 1—17.
- [5] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms, *ACM Trans. Math. Soft.*, 16 (1990), pp. 18—28.
- [6] Anderson, E. and Bai, Z. and Bischof, C. and Blackford, S. and Demmel, J. and Dongarra, J. and Du Croz, J. and Greenbaum, A. and Hammarling, S. and McKenney, A. and Sorensen, D., *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, 1999.
- [7] Algorithm 907: KLU, A Direct Sparse Solver for Circuit Simulation Problems, Timothy A. Davis, Ekanathan Palamadai Natarajan, *ACM Transactions on Mathematical Software*, Vol 37, Issue 6, 2010, pp 36:1 - 36:17.
- [8] Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization, Timothy A. Davis, *ACM Transactions on Mathematical Software*, Vol 38, Issue 1, 2011, pp 8:1 - 8:22.
- [9] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, PETSc Web page, <http://www.mcs.anl.gov/petsc>, year = 2018.
- [10] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik,

- Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang,, PETSc Users Manual, Argonne National Laboratory, ANL-95/11 - Revision 3.9, 2018.
- [11] Satish Balay, William D. Gropp, Lois Curfman McInnes, Barry F. Smith, Efficient Management of Parallelism in Object Oriented Numerical Software Libraries, *Modern Software Tools in Scientific Computing*, Birkhäuser Press, 1997, 163–202.
- [12] P. R. Amestoy, A. Buttari, J.-Y. L’Excellent and T. Mary, On the complexity of the Block Low-Rank multifrontal factorization, *SIAM Journal on Scientific Computing*, volume 39, number 4, pages A1710-A1740.
- [13] Xiaoye S. Li, An Overview of SuperLU: Algorithms, Implementation, and User Interface, *toms*, 31(3), 2005, 302-325.
- [14] X.S. Li and J.W. Demmel and J.R. Gilbert and iL. Grigori and M. Shao and I. Yamazaki, SuperLU Users’ Guide, Lawrence Berkeley National Laboratory, LBNL-44289, September, 1999.
- [15] James W. Demmel and Stanley C. Eisenstat and John R. Gilbert and Xiaoye S. Li and Joseph W. H. Liu, A supernodal approach to sparse partial pivoting, *SIAM J. Matrix Analysis and Applications*, 20(3), 1999, 720-755.
- [16] Hui Liu, SXAMG: a Serial Algebraic Multigrid Solver Library, 2018.
- [17] Xiaozhe Hu, Wei Liu, Guan Qin, Jinchao Xu, Zhensong Zhang, Development of A Fast Auxiliary Subspace Pre-conditioner for Numerical Reservoir Simulators, SPE-148388-MS, SPE Reservoir Characterisation and Simulation Conference and Exhibition, 9-11 October, Abu Dhabi, UAE.