Gen_SuperLU package (version 1.0)
by Ahmet Duran and David Saunders
Computer & Information Science Department
University of Delaware
August 2002

Gen_SuperLU contains a set of subroutines to solve a sparse linear system A*X=B over any field. It uses Gaussian elimination with partial pivoting (GEPP). The columns of A may be preordered before factorization; the preordering for sparsity is completely separate from the factorization.

Gen_SuperLU was used for the rank computations at
"A. Duran, B. D. Saunders and Z. Wan. Hybrid Algorithms for Rank of Sparse Matrices. Proceeding of the SIAM International Conference on Applied Linear Algebra. July 2003."
and
"A. Duran, B. D. Saunders and Z. Wan. Rank of Sparse {0, 1} Matrices, ECCAD'03".

Gen_SuperLU became a part of LinBox package.

Gen_SuperLU is adapted from SuperLU, version 2.0, from UC Berkeley (Demmel, et al). Our version references field arithmetic operations in the LinBox style (the field object is an explicit parameter to the operation as well as the field elements involved). This allows Gen_SuperLU to be used with arbitrary fields including finite field representations from linbox and light wrappers on traditional floating point types (float, double. complex).

Gen_SuperLU is implemented in C++. It requires the GNU C++ compiler (gcc-2.95.1 or newer) or any compiler supporting advanced template features. It provides functionality for real and integer matrices on the fields such as

```
LinBox::UnparametricField<double> // light wrapper for double
LinBox::UnparametricField<float>  // light wrapper for float
LinBox::Modular<LinBox::uint32>   // arithmetic mod p, for p < 2^32.
LinBox::GivaroZpz<Std32>          // arithmetic mod p, using zech log tables,
                                     p < 10^6 generally.
```

The code uses C++ template parameters for the field. This allows generic code to work for single and double precision floating point arithmetic and for finite field arithmetic. Also, template specialization may be used to optimize special cases.

Gen_SuperLU contains the following directory structure:

```
gen_superlu/README      instructions on installation
gen_superlu/example     example program genlinsol.C and
readtriple.h
Makefile

gen_superlu/src         header files
gen_superlu/make.gen    definitions for compiler choice, compile flags,
LINBOXLIB and
INCLUDESLINBOX.         (You may need to edit it to include some
                         libraries and directories from the LinBox package.
```

Before using the package, please examine the three things dependent on your system setup:

1. Edit the make.gen include file.

This make include file is referenced inside each of the Makefiles in the various subdirectories. As a result, there is no need to edit the Makefiles in the subdirectories. All information that is machine specific has been defined in this include file.

Example machine-specific SuperLU/make.inc include files are provided in the top-level SuperLU/ directory for several systems, such as IBM RS/6000, DEC Alpha, SunOS 4.x, SunOS 5.x (Solaris), HP-PA and SGI Iris 4.x. When you have selected the machine to which you wish to install SuperLU, copy the appropriate sample include file (if one is present) into SuperLU/make.inc. For example, if you wish to run SuperLU on an IBM RS/6000, you can do

```
cp make.rs6k make.inc
```

For the systems other than listed above, slight modifications to the make.inc file will need to be made.

We used SuperLU (Version 2.0)
Thanks to
Univ. of California Berkeley, Xerox Palo Alto Research Center,
and Lawrence Berkeley National Lab.

Their README follows.

SuperLU (Version 2.0)

SuperLU contains a set of subroutines to solve a sparse linear system A*X=B. It uses Gaussian elimination with partial pivoting (GEPP). The columns of A may be preordered before factorization; the preordering for sparsity is completely separate from the factorization.

SuperLU is implemented in ANSI C, and must be compiled with standard ANSI C compilers. It provides functionality for both real and complex matrices, in both single and double precision. The file names for the single-precision real version start with letter "s" (such as sgstrf.c); the file names for the double-precision real version start with letter "d" (such as dgstrf.c); the file names for the single-precision complex version start with letter "c" (such as cgstrf.c); the file names for the double-precision complex version start with letter "z" (such as zgstrf.c).

SuperLU contains the following directory structure:

SuperLU/README	instructions on installation
SuperLU/CBLAS/	needed BLAS routines in C, not necessarily fast
SuperLU/EXAMPLE/	example programs
SuperLU/INSTALL/	test machine dependent parameters; the Users' Guide.
SuperLU/MATLAB/	Matlab mex-file interface
SuperLU/SRC/	C source code, to be compiled into the superlu.a library

```
    SuperLU/TESTING/  driver routines to test correctness
    SuperLU/Makefile  top level Makefile that does installation and testing
    SuperLU/make.inc  compiler, compile flags, library definitions and C
                      preprocessor definitions, included in all Makefiles.
                      (You may need to edit it to be suitable for your system
                       before compiling the whole package.)
```

Before installing the package, please examine the three things dependent on your system setup:

1. Edit the make.inc include file.
   This make include file is referenced inside each of the Makefiles in the various subdirectories. As a result, there is no need to edit the Makefiles in the subdirectories. All information that is machine specific has been defined in this include file.

   Example machine-specific SuperLU/make.inc include files are provided in the top-level SuperLU/ directory for several systems, such as IBM RS/6000, DEC Alpha, SunOS 4.x, SunOS 5.x (Solaris), HP-PA and SGI Iris 4.x. When you have selected the machine to which you wish to install SuperLU, copy the appropriate sample include file (if one is present) into SuperLU/make.inc. For example, if you wish to run SuperLU on an IBM RS/6000, you can do

   ```
       cp make.rs6k make.inc
   ```

   For the systems other than listed above, slight modifications to the make.inc file will need to be made.

2. The BLAS library.
   If there is BLAS library available on your machine, you may define the following in the file SuperLU/make.inc:

   ```
        BLASDEF = -DUSE_VENDOR_BLAS
        BLASLIB = <BLAS library you wish to link with>
   ```

   The CBLAS/ subdirectory contains the part of the C BLAS needed by SuperLU package. However, these codes are intended for use only if there is no faster implementation of the BLAS already available on your machine. In this case, you should do the following:

    1) In SuperLU/make.inc, undefine (comment out) BLASDEF, and define:
       ```
       BLASLIB = ../blas$(PLAT).a
       ```

    2) Go to the SuperLU/ directory, type:
       ```
       make blaslib
       ```
       to make the BLAS library from the routines in the CBLAS/ subdirectory.

3. C preprocessor definition CDEFS.
   In the header file SRC/Cnames.h, we use macros to determine how C routines should be named so that they are callable by Fortran. (Some vendor-supplied BLAS libraries do not have C interface. So the re-naming is needed in order for the SuperLU BLAS calls (in C) to interface with the Fortran-style BLAS.)
   The possible options for CDEFS are:

    o -DAdd_: Fortran expects a C routine to have an underscore postfixed to the name;

- o -DNoChange: Fortran expects a C routine name to be identical to that compiled by C;
- o -DUpCase: Fortran expects a C routine name to be all uppercase.

4. The Matlab MEX-file interface.

   The MATLAB/ subdirectory includes Matlab C MEX-files, so that our factor and solve routines can be called as alternatives to those built into Matlab. In the file SuperLU/make.inc, define MATLAB to be the directory in which Matlab is installed on your system, for example:

   ```
   MATLAB = /usr/local/matlab
   ```

   At the SuperLU/ directory, type "make matlabmex" to build the MEX-file interface. After you have built the interface, you may go to the MATLAB/ directory to test the correctness by typing (in Matlab):

   ```
   trysuperlu
   trylusolve
   ```

A Makefile is provided in each subdirectory. The installation can be done completely automatically by simply typing "make" at the top level. The test results are in the files below:

```
INSTALL/install.out
TESTING/stest.out
TESTING/dtest.out
TESTING/ctest.out
TESTING/ztest.out
```