SuperLU Users' Guide

Xiaoye S. Li[1] **James W. Demmel**[2] **John R. Gilbert**[3] **Laura Grigori**[4]

Contents

2.9.1 Driver routines 29
2.9.2 Computational routines 29
2.9.3 Utility routines 30
2.10 Matlab interface 31
2.11 Installation 33
2.11.1 File structure 33

Chapter 1

	Sequential SuperLU	SuperLU_MT	SuperLU_DIST
Platform	**serial**		

1.3.2 Tuning Parameters for BLAS

1.3.6 Iterative Refinement

Step 6 of the expert driver algorithm, iterative refinement,

Here $\|x\|_\infty \equiv \max_i |x_i|$. Thus, if FERR = 10^{-6} then each component of x has an error bounded by about 10^{-6}

1.4 How the three libraries differ

Chapter 2

Sequential SuperLU (Version 4.1)

```
    /* De-allocate storage */
    SUPERLU_FREE (rhs);
    SUPERLU_FREE (perm_r);
    SUPERLU_FREE (perm_c);
    Destroy_CompCol_Matrix(&A);
    Destroy_SuperMatrix_Store(&B);
    Destroy_SuperNode_Matrix(&L);
    Destroy_CompCol_Matrix(&U);
    StatFree(&stat);
}
```

2.3 Matrix data structures

```
typedef struct {
    Stype_t Stype; /* Storage type: indicates the storage format of *Store. */
    Dtype_t Dtype; /* Data type. */
    Mtype_t Mtype; /* Mathematical type */
    int  nrow;     /* number of rows */
    int  ncol;     /* number of columns */
```

```
                              which column j belongs */
int   *sup_to_col;   /* sup_to_col[s] points to the starting column
```

- ```
  A = { Stype = SLU_NC; Dtype = SLU_D; Mtype = SLU_GE; nrow = 5; ncol = 5;
        *Store = { nnz = 12;
                   nzval = [ 19.00, 12.00, 12.00, 21.00, 12.00, 12.00, 21.00,
  ```
  ```

- Equil **{** YES **|** NO **}**
  Specifies whether to equilibrate the system (scale A's rows and columns to have unit norm).
- ColPerm
  Specifies how to permute the columns of the matrix for sparsity preservation.
  - NATURAL**: natural ordering.**

- DROP_PROWS:

2.5 Permutations

Furthermore, we incorporated several heuristics for adapt

2.9.1 Driver routines

We provide two types of driver routines for solving systems o

- `dgscon()`

```
/* Convert the compressed row fromat to the compressed column format. */
dCompRow_to_CompCol(int m, int n, int nnz,
```

```
[L,U,prow,pcol] = superlu(A)  preorders the columns of A by min degree,
```

using a supernodal LU factorization that is faster than Matlab’s

$$\mathbf{FERR} = \frac{\| \, |A^{-1}| \, f \, \|_\infty}{\|x\|_\infty} .$$

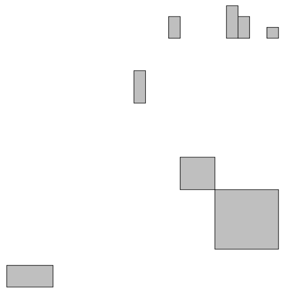
Here, f

```
printf("No of nonzeros in L+U = %d\n", Lstore->nnz + Ustore->nnz);
dQuerySpace(L, U, &mem_usage);
```

```
      enddo
*
```

```
typedef struct {
    int  nnz;          /* number of nonzeros in the matrix */
    int  nsuper;       /* number of supernodes */
    void *nzval;       /* pointer to array of nonzero values,
                          packed by column */
    int *nzval_colbeg; /* nzval_colbeg[j] points to beginning of column j
                          in nzval[] */
    int *nzval_colend; /* nzval_colend[j] points to one past the last
                          element of column j in nzval[] */
    int *rowind;       /* pointer to array of compressed row indices of
                          the supernodes */
    int *rowind_colbeg;/* rowind_colbeg[j] points to beginning of column j
                          in rowind[] */
    int *rowind_colend;/* rowind_colend[j] points to one past the last
                          element of column j in rowind[] */
    int *col_to_sup;   /* col_to_sup[j] is the supernode number to which
                          column j belongs */
    int *sup_to_colbeg;/* sup_to_colbeg[s] points to the first column
                          of the s-th supernode /
    int *sup_to_colend;/* sup_to_colend[s] points to one past the last
                          column of the s-th supernode */
} SCPformat;
```

Chapter 4



This process grid will use the first `nprow`

NOTE: All processes in the base group, including those not in

any other ordering in place of the ones provided in the librar

- `RefineInitialized` **{** YES **|** NO **}**

```
*   4. Call pdgssvx
*   5. Release the process grid and terminate the MPI environment
*
* On the Cray T3E, the program may bee
```

```
PStatInit(&stat);

/* Call the linear equation solver. */
pdgssvx(&options, &A, &ScalePermstruct, b, ldb, nrhs, &grid,
        &LUstruct, &SOLVEstruct, berr, &stat, &info);


/* Check the accuracy of the solution. */
pdinf_norm_error(iam, ((NRformat_loc *)A.Store)->m_loc,
                 nrhs, b, ldb, xtrue, ldx, &grid);

PStatPrint(&options, &stat, &grid);        /* Print the statistics. */

/* ------------------------------------------------------------
```

`superlu_gridinit()`. **In this example, the communication domain for SuperLU is built upon the MPI default communicator** `MPI_COMM_WORLD`**. In general, it can be built upon any MPI communicator. Section 4.4 contains the details about this step.**

```
ScalePermstructFree(ScalePermstruct_t *ScalePermstruct);

/* Allocate storage in LUstruct. */
LUstructInit(const int_t m, const int_t n, LUstruct_t *LUstruct);

/* Deallocate the distributed L & U factors in LUstruct. */
Destroy_LU(int_t n, gridinfo_t *grid, LUstruct_t *LUstruct);

/* Deallocate LUstruct. */
LUstructFree(LUstruct_t *LUstruct);

/* Initialize the statistics variable. */
```

```
      parameter ( maxn = 10000, maxnz = 100000, maxnrhs = 10 )
      integer rowind(maxnz), colptr(maxn)
      real*8  values(maxnz), b(maxn), berr(maxnrhs)
      integer n, m, nnz, nrhs, ldb, i, ierr, info, iam
      integer nprow, npcol
      integer init

      integer(superlu_ptr) :: grid
      integer(superlu_ptr) :: options
      integer(superlu_ptr) :: ScalePermstruct
      integer(superlu_ptr) :: LUstruct
      integer(superlu_ptr) :: SOLVEstruct
      integer(superlu_ptr) :: A
      integer(superlu_ptr) :: stat


! Create Fortran handles for the C structures used in SuperLU_DIST
      call f_create_gridinfo(grid)
      call f_create_options(options)
      call f_create_ScalePermstruct(ScalePermstruct)
      call f_create_LUstruct(LUstruct)
      call f_create_SOLVEstruct(SOLVEstruct)
      call f_create_SuperMatrix(A)
      call f_create_SuperLUStat(stat)

! Initialize MPI environment
      call mpif47 [(c)-3(a)-3(l)-3-519.993(e)-3(e)-2.)cactacl(l)-((d)-2.99886()519.993a)-3(
```

```
subroutine set_SuperMatrix(A, nrow, ncol)
  integer(superlu_ptr) :: A
  integer, optional :: nrow, ncol
```

```
fptr *SOLVEstruct, double *berr, fptr *stat, int *info)
```

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J

[27] Message Passing Interface (MPI) forum. http://www.mp