# SKYPACK User's Guide

Osni Marques[*]

May 27, 2009

**Abstract**

This document describes how to use SKYPACK (SKYline Package), which is a set of subprograms intended for linear algebra computations involving real symmetric matrices stored in skyline or profile form. This document gives an overview of the package and describes the storage strategy employed, out-of-core features supported, and the computational routines and drivers available in the package.

[*]National Energy Research Scientific Computing Center (NERSC), Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA.

## Contents

**1 Overview** **1**

**2 Storage strategy** **2**

2.1 NBLOCK=1 ($A$ is stored in-core) . . . . . . . . . . . . . . . . . . . . . . . . . . 2

2.2 NBLOCK>1 ($A$ is stored out-of-core) . . . . . . . . . . . . . . . . . . . . . . . . 3

**3 Computational routines and drivers** **4**

SKYSF2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

SKYPF2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

SKYSS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

MTRXYS . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

SKYSCF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

SKYPCF . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

MTRXY . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

**4 Out-of-core storage** **9**

**5 General information** **9**

**Bibliography** **10**

## 1 Overview

SKYPACK (for Skyline Package) is a set of subprograms written in standard Fortran 77 intended for linear algebra computations involving real symmetric matrices stored in skyline or profile form. For further details about the skyline storage the reader should consult [4, 6, 7, 11].

Let $A$ be a real symmetric matrix of dimension $n$, $B$ a real diagonal matrix or a real symmetric matrix with the same pattern of $A$, $L$ a lower triangular matrix, $D$ a diagonal matrix, $x$ and $y$ vectors, and $\sigma$ a real scalar. SKYPACK requires the upper triangle of $A$ (and $B$) stored in skyline form and supports the following computations:

- the factorization $A = LDL^T$, standard implementation (Level 1 BLAS based);
- the factorization $A + \sigma B = LDL^T$, standard implementation (Level 1 BLAS based);
- the factorization $A = LDL^T$, partitioned implementation (Level 2 and 3 BLAS based);
- the factorization $A + \sigma B = LDL^T$, partitioned implementation (Level 2 and 3 BLAS based);
- the solution of $Ax = b$, standard implementation (Level 1 BLAS based), after the factorization of $A = LDL^T$ has been performed;
- the product $y = Ax$.

SKYPACK also provides out-of-core implementations of the computations listed above. The partitioned implementation allows for significant computational performance improvements on machines with cache memory, for further details the reader should consult [9].

It should be noted that many applications benefit from more general sparse storage schemes for $A$, such as those implemented in MA27 and MA47 [1], MUMPS [2], SPOOLES [3], SuperLU [5], PSPASES [8] and MFACT [10]. In particular, sparse storage schemes usually allow for the implementation of sophisticated pivoting strategies to preserve the numerical stability of factorizations of indefinite matrices. However, with the skyline storage the amount of memory required in the computations is known beforehand (providing no pivoting is used), in contrast to other sparse storage schemes, and this information may be useful for prototyping codes. Also, the skyline storage may be appropriate for the solution of 2D finite element problems.

No pivoting technique is implemented in SKYPACK (which would change the skyline pattern), therefore the factorization of an indefinite matrix may require some attention from the user.

In the next section, we show how SKYPACK implements the skyline storage. Then, the computational routines and drivers available are described, followed by details about the out-of-core storage features and more general information about the package.

## 2 Storage strategy

In SKYPACK, a matrix $A$ is represented by means of three arrays:

- `A`, which holds the entries of $A$ (upper triangle only, including the diagonal);
- `DGPNT`, which holds the addresses of the diagonal entries of $A$ in the array `A`;
- `BLOCK`, which holds information about the splitting or blocking of $A$.

A matrix $A$ can be split into NBLOCK blocks of different sizes and all the information needed to define the splitting of $A$ is provided in the array BLOCK. Depending on the amount of memory available to store the entries of $A$, there are two cases to be considered, NBLOCK=1 and NBLOCK>1. These two cases are examined below.

### 2.1 NBLOCK=1 ($A$ is stored in-core)

This is the ideal situation in terms of computational performance. Let us consider the upper triangle of a small symmetric matrix with $n = 8$,

$$A = \begin{bmatrix} x & x & x & & & & & \\ & x & 0 & x & & x & x & \\ & & x & 0 & x & 0 & 0 & \\ & & & x & 0 & 0 & 0 & \\ & & & & x & x & x & \\ & & & & & x & x & x \\ & & & & & & x & x \\ & & & & & & & x \end{bmatrix},$$

where $x$ indicates a non zero entry. In SKYPACK, $A$ is represented as:

$$\begin{aligned} \texttt{A} \quad &= \big[\; a_{1,1} \;\; a_{1,2} \;\; a_{2,2} \;\; a_{1,3} \;\; a_{2,3} \;\; a_{3,3} \;\; a_{2,4} \;\; a_{3,4} \;\; a_{4,4} \;\; a_{3,5} \;\; a_{4,5} \;\; a_{5,5} \;\; a_{2,6} \\ &\qquad a_{3,6} \;\; a_{4,6} \;\; a_{5,6} \;\; a_{6,6} \;\; a_{2,7} \;\; a_{3,7} \;\; a_{4,7} \;\; a_{5,7} \;\; a_{6,7} \;\; a_{7,7} \;\; a_{6,8} \;\; a_{7,8} \;\; a_{8,8} \;\big] \\ \texttt{DGPNT} \quad &= \begin{bmatrix} 1 & 3 & 6 & 9 & 12 & 17 & 23 & 26 \end{bmatrix} \\ \texttt{BLOCK} \quad &= \begin{bmatrix} 1 \\ 8 \\ 1 \end{bmatrix} \begin{array}{l} \text{(JMIN, first column of the block)} \\ \text{(JMAX, last column of the block)} \\ \text{(minimum row index in the block)} \end{array} \end{aligned}$$

Therefore, `A` holds all entries from the first non zero entry to the diagonal entry in each column of $A$. This scheme is known as "profile-in" (another possibility is the "diagonal-out" which stores from the diagonal to the first non null entry in the column, if needed both strategies can be easily modified to store $A$ by rows). Note that $a_{2,3} = a_{3,4} = a_{4,5} = a_{3,6} = a_{4,6} = a_{3,7} = a_{4,7} = 0$ also need to be stored. The i-*th* entry of `DGPNT` points to the entry of `A` holding $a_{i,i}$. In this case, `BLOCK` has only one column, such as JMIN=1 and JMAX=$n$.

## 2.2 NBLOCK>1 ($A$ is stored out-of-core)

This situation happens when there is not enough memory to store all entries of $A$ at once. In such a case, $A$ can be stored by blocks in a binary file which is accessed sequentially by SKYPACK. Let us consider the upper triangle of a small symmetric matrix with $n = 15$,

$$A = \left[ \begin{array}{ccccccccccccccc}
x & x & x & & & & & & & & & & & & \\
 & x & 0 & x & & x & x & & & & & & & & \\
 & & x & 0 & x & 0 & 0 & & & & & & & & \\
 & & & x & 0 & 0 & 0 & & & & & & & & \\
 & & & & x & x & x & & & & & & & & \\
 & & & & & x & x & x & & x & & & & & \\
 & & & & & & x & x & x & 0 & & & & x & \\
 & & & & & & & x & 0 & x & & & & 0 & \\
 & & & & & & & & x & x & x & x & & x & \\
 & & & & & & & & & x & x & x & & 0 & \\
 & & & & & & & & & & x & 0 & x & 0 & \\
 & & & & & & & & & & & x & 0 & 0 & x \\
 & & & & & & & & & & & & x & 0 & 0 \\
 & & & & & & & & & & & & & x & x \\
 & & & & & & & & & & & & & & x
\end{array} \right],$$

where $x$ indicates a non zero entry. Let us also assume that only 20 entries of $A$ can be stored in `A` at the same time. Then, NBLOCK=3, and $A$ is represented in SKYPACK as:

First block, 17 entries:

$$\texttt{A} = \left[ \begin{array}{cccccccc} a_{1,1} & a_{1,2} & a_{2,2} & a_{1,3} & a_{2,3} & a_{3,3} & \cdots & a_{6,6} \end{array} \right]$$

Second block, 20 entries:

$$\texttt{A} = \left[ \begin{array}{ccccccc} a_{2,7} & a_{3,7} & \cdots & a_{10,10} & a_{9,11} & a_{10,11} & a_{11,11} \end{array} \right]$$

Third block, 19 entries:

$$\texttt{A} = \left[ \begin{array}{ccccc} a_{9,12} & a_{10,12} & a_{11,12} & \cdots & a_{15,15} \end{array} \right]$$

Also,

$$\begin{array}{lcl}
\texttt{DGPNT} & = & \left[ \begin{array}{ccccccccccccccc} 1 & 3 & 6 & 9 & 12 & 17 & 6 & 9 & 12 & 17 & 20 & 4 & 7 & 15 & 19 \end{array} \right] \\
\texttt{BLOCK} & = & \left[ \begin{array}{ccc} 1 & 7 & 12 \\ 6 & 11 & 15 \\ 1 & 2 & 7 \end{array} \right] \begin{array}{l} \text{(JMIN, first column of the block)} \\ \text{(JMAX, last column of the block)} \\ \text{(minimum row index in the block)} \end{array}
\end{array}$$

In this case, SKYPACK deals with one block of $A$ at a time. Again, some zero entries need to be stored ($a_{2,3}$ for instance) and the i-*th* entry of `DGPNT` points to the entry of `A` holding $a_{i,i}$. Now `BLOCK` has three columns, defining how $A$ is represented in `A` and `DGPNT`.

## 3 Computational routines and drivers

This section describes the computational routines and drivers available in SKYPACK. An argument used as input is marked with an 'i', output with an 'o', and workspace with a 'w'. Details of the out-of-core features are given in Section 4.

**Computational routines:**

SKYSF2 computes the factorization $A = LDL^T$, standard implementation [4, 7, 11],

```
CALL SKYSF2 (A,D,DGPNT,JMIN,JMAX)
```

| | | |
|---|---|---|
| A | : | (io) real (double precision) array of dimension at least max(DGPNT($k$)), $k = 1, 2 \ldots n$. On input A holds the entries of $A$, on output A holds the entries of $L$ and the reciprocals of the entries of $D$. See Section 4 for information about out-of-core storage. |
| D | : | (o) real (double precision) array of dimension at least $n$ which holds the reciprocals of the entries of $D$. |
| DGPNT | : | (i) integer array of dimension at least $n$ which holds the addresses of the diagonal entries of $A$. |
| JMIN | : | (i) integer, index of the first column of $A$. |
| JMAX | : | (i) integer, index of the last column of $A$. |

SKYPF2 computes the factorization $A = LDL^T$, partitioned implementation [9],

```
CALL SKYPF2 (INDEX,JMAX,JMIN,MCOLS,MROWS,DGPNT,A,RW,TW)
```

| | | |
|---|---|---|
| INDEX | : | (w) integer array of dimension at least max(MCOLS,MROWS). |
| JMAX | : | (i) integer, index of the last column of $A$. |
| JMIN | : | (i) integer, index of the first column of $A$. |
| MCOLS | : | (i) integer, maximum number of columns stored in RW. |
| MROWS | : | (i) integer, maximum number of rows stored in RW. |
| DGPNT | : | (i) integer array of dimension at least $n$ which holds the addresses of the diagonal entries of $A$. |
| A | : | (io) real (double precision) array of dimension at least max(DGPNT($k$)), $k = 1, 2, \ldots n$. On input A holds the entries of $A$, on output A holds the entries of $L$ and $D$. See Section 4 for information about out-of-core storage. |
| RW | : | (w) real (double precision) work array of dimension MROWS×MCOLS. |
| TW | : | (w) real (double precision) work array of dimension MROWS×MROWS. |

SKYSS solves $AX = Y$ after $A$ has been factorized as $A = LDL^T$,

CALL SKYSS (LN,N,NX,A,D,X,DGPNT,BLOCK,NBLOCK,FNAME,FUNIT, INFO)

| | |
|---|---|
| LN | : (i) integer, leading dimension of $X$. |
| N | : (i) integer, dimension of $A$. |
| NX | : (i) integer, number of columns in $X$ and $Y$ (number of right-hand sides). |
| A | : (i) real (double precision) array of dimension at least max(DGPNT(k)), $k = 1, 2 \ldots n$, which holds the entries of $L$ and the reciprocals of the entries of $D$. See Section 4 for information about out-of-core storage. |
| D | : (i) real (double precision) array of dimension at least $n$ which holds the reciprocals of the entries of $D$. |
| X | : (io) real (double precision) array of dimension LN×NX which holds the right-hand side $X$ on input and the solution $Y$ on output. |
| DGPNT | : (i) integer array of dimension at least $n$ which holds the addresses of the diagonal entries of $A$. |
| BLOCK | : (i) integer array of dimension 3×NBLOCK which holds information about the blocking of $A$. |
| NBLOCK | : (i) integer, number of blocks. |
| FNAME | : (i) string, file name associated with $A$. |
| FUNIT | : (i) integer, file unit associated with $A$. FUNIT is ignored if FNAME is set to "INCORE" or "incore" . |
| INFO | : (o) integer, exit flag. INFO=0, normal exit, INFO =1, IO error, INFO=2, the file specified by FNAME was not found. |

MTRXYS computes the product $Y = AX$,

CALL MTRXYS (LN,N,NY,A,X,Y,DGPNT,BLOCK,NBLOCK,FNAME,FUNIT, INFO)

| | |
|---|---|
| LN | : (i) integer, leading dimension of $X$ and $Y$. |
| N | : (i) integer, dimension of $A$. |
| NY | : (i) integer, number of columns in $X$ and $Y$ (number of vectors to be multiplied by $A$). |
| A | : (i) real (double precision) array of dimension at least max(DGPNT($k$)), $k = 1, 2, \ldots n$, which holds the entries of $A$. See Section 4 for information about out-of-core storage. |
| X | : (i) real (double precision) array of dimension LN×NY which holds the input vectors $X$. |
| Y | : (o) real (double precision) array of dimension LN×NY which holds the output vectors $Y$. |
| DGPNT | : (i) integer array of dimension at least $n$ which holds the addresses of the diagonal entries of $A$. |

| | |
|---|---|
| BLOCK | : (i) integer array of dimension 3×NBLOCK which holds information about the blocking of $A$. |
| NBLOCK | : (i) integer, number of blocks. |
| FNAME | : (i) string, file name associated with $A$. |
| FUNIT | : (i) integer, file unit associated with FNAME. FUNIT is ignored if FNAME is set to "INCORE" or "incore". |
| INFO | : (o) integer, exit flag. INFO=0, normal exit, INFO=1, IO error. |

**Computational drivers:**

SKYSCF computes the factorization $A + \sigma B = LDL^T$, standard implementation,

```
CALL SKYSCF (N,DGPNT,BLOCK,NBLOCK,FANAME,FAUNIT,FFNAME,
             FFUNIT,FBNAME,FBUNIT,MBTYPE,A,B,
             D,SIGMA,INRTIA,INFO)
```

| | |
|---|---|
| N | : (i) integer, dimension of $A$. |
| DGPNT | : (i) integer array of dimension at least $n$ which holds the addresses of the diagonal entries of $A$. |
| BLOCK | : (i) integer array of dimension 3×NBLOCK which holds information about the blocking of $A$. |
| NBLOCK | : (i) integer, number of blocks. |
| FANAME | : (i) string, file name associated with $A$. |
| FAUNIT | : (i) integer, file unit associated with $A$. FAUNIT is ignored if FANAME is set to "INCORE" or "incore". |
| FFNAME | : (i) string, file name associated with $D$ and $L$. |
| FFUNIT | : (i) integer, file unit associated with $D$ and $L$. FFUNIT is ignored if FFNAME is set to "INCORE" or "incore". |
| FBNAME | : (i) string, file name associated with $B$. |
| FBUNIT | : (i) integer, file unit associated with $B$. FBUNIT is ignored if FBNAME is set to "INCORE" or "incore". |
| MBTYPE | : (i) integer, type of $B$. MTYPE=1, $B$ is diagonal, MTYPE=2, $B$ is skyline. |
| A | : (io) real (double precision) array of dimension at least max(DGPNT(k)), $k = 1, 2, \ldots n$. On input A holds the entries of $A$, on output A holds the entries of $L$ and the reciprocals of the entries of $D$. See Section 4 for information about out-of-core storage. |
| B | : (i) real (double precision) array which holds the entries of $B$. If MTYPE=1 the dimension of B must be at least $n$, otherwise max(DGPNT(k)), $k = 1, 2, \ldots n$. See Section 4 for information about out-of-core storage. |

| | | |
|---|---|---|
| D | : | (i) real (double precision) array of dimension at least $n$ which holds the reciprocals of the entries of $D$. |
| SIGMA | : | (i) real (double precision) scalar that multiplies $B$. |
| INRTIA | : | (o) integer array of dimension 3, on output it holds the inertia of $A + \sigma B$, INRTIA(1) = number of eigenvalues less than 0, INRTIA(2) = number of eigenvalues equal to 0, INRTIA(3) = number of eigenvalues greater than 0. |
| INFO | : | (o) integer, exit flag. INFO=0, normal exit, INFO=1, IO error, INFO=2, the file specified by FANAME was not found, INFO=3, the file specified by FBNAME was not found. |

SKYPCF computes the factorization $A + \sigma B = LDL^T$, partitioned implementation,

```
CALL SKYPCF (N,DGPNT,BLOCK,NBLOCK,FANAME,FAUNIT,FFNAME,
             FFUNIT,FBNAME,FBUNIT,MBTYPE,A,B,D,SIGMA,
             RW,TW,MCOLS,MROWS,INDEX,INRTIA,INFO)
```

| | | |
|---|---|---|
| N | : | (i) integer, dimension of $A$. |
| DGPNT | : | (i) integer array of dimension at least $n$ which holds the addresses of the diagonal entries of $A$. |
| BLOCK | : | (i) integer array of dimension 3×NBLOCK which holds information about the blocking of $A$. |
| NBLOCK | : | (i) integer, number of blocks. |
| FANAME | : | (i) string, file name associated with $A$. |
| FAUNIT | : | (i) integer, file unit associated with $A$. FAUNIT is ignored if FANAME is set to "INCORE" or "incore". |
| FFNAME | : | (i) string, file name associated with $D$ and $L$. |
| FFUNIT | : | (i) integer, file unit associated with $D$ and $L$. FFUNIT is ignored if FFNAME is set to "INCORE" or "incore". |
| FBNAME | : | (i) string, file name associated with $B$. |
| FBUNIT | : | (i) integer, file unit associated with $B$. FBUNIT is ignored if FBNAME is set to "INCORE" or "incore". |
| MBTYPE | : | (i) integer, type of $B$. MBTYPE=1, $B$ is diagonal, MBTYPE=2, $B$ is skyline. |
| A | : | (io) real (double precision) array of dimension at least max(DGPNT($k$)), $k = 1, 2, \dots n$. On input A holds the entries of $A$, on output A holds the entries of $L$ and the reciprocals of the entries of $D$. See Section 4 for information about out-of-core storage. |
| B | : | (i) real (double precision) array which holds the entries of $B$. If MTYPE=1 the dimension of B must be at least $n$, otherwise max(DGPNT(k)), $k = 1, 2, \dots n$. See Section 4 for information about out-of-core storage. |

| | |
|---|---|
| D | : (i) real (double precision) array of dimension at least $n$ which holds the reciprocals of the entries of $D$. |
| SIGMA | : (i) real (double precision) scalar that multiplies $B$. |
| RW | : (i) real (double precision) work array of dimension MROWS×MCOLS. |
| TW | : (i) real (double precision) work array of dimension MROWS×MROWS. |
| MCOLS | : (i) integer, maximum number of columns in RW. |
| MROWS | : (i) integer, maximum number of rows in RW. |
| INDEX | : (i) integer work array with dimension max(MCOLS,MROWS). |
| INRTIA | : (o) integer array of dimension 3, on output it holds the inertia of $A + \sigma B$, INRTIA(1) = number of eigenvalues less than 0, INRTIA(2) = number of eigenvalues equal to 0, INRTIA(3) = number of eigenvalues greater than 0. |
| INFO | : (o) integer, exit flag. INFO=0, normal exit, INFO=1, IO error, INFO=2, the file specified by FANAME was not found, INFO=3, the file specified by FBNAME was not found. |

MTRXY computes the product $Y = AX$,

```
CALL MTRXY (LN,N,NY,A,X,Y,DGPNT,BLOCK,NBLOCK,MTYPE,FNAME,
            FUNIT,INFO)
```

| | |
|---|---|
| LN | : (i) integer, leading dimension of X and Y |
| N | : (i) integer, dimension of $A$ |
| NY | : (i) integer, number of vectors to be multiplied by $A$ |
| A | : (i) real (double precision) array of dimension at least max(DGPNT(k)), $k = 1, 2, \ldots n$, which holds the entries of $A$. See Section 4 for information about out-of-core storage. |
| X | : (i) real (double precision) array of dimension LN×NY which holds the input vectors $X$. |
| Y | : (o) real (double precision) array of dimension LN×NY which holds the output vectors $Y$. |
| DGPNT | : (i) integer array of dimension at least $n$ which holds the addresses of the diagonal entries of $A$. DGPNT is ignored if MTYPE=1. |
| BLOCK | : (i) integer array of dimension 3×NBLOCK, which holds information about the blocking of $A$. BLOCK is ignored if MTYPE=1. |
| NBLOCK | : (i) integer, number of blocks. NBLOCK is ignored if MTYPE=1. |
| MTYPE | : (i) integer, matrix type. MTYPE=1, $A$ is diagonal, MTYPE=2, $A$ is skyline. |
| FNAME | : (i) string, file name associated with $A$ |
| FUNIT | : (i) integer, file unit associated with $A$. FUNIT is ignored if FNAME is set to "INCORE" or "incore". |

| | | |
|---|---|---|
| INFO | : | (o) integer, exit flag. INFO=0, normal exit, INFO=1, IO error, INFO=2, the file specified by FNAME was not found. |

## 4 Out-of-core storage

**The matrix $A$.** $A$ can be partitioned into NBLOCK blocks, as described in Section 2. If these NBLOCK blocks are stored sequentially in a binary file labeled FANAME, they can be read by SKYSCF (or SKYPCF). Then, SKYSCF (or SKYPCF) computes the factorization of $A$ one block at a time and writes the result ($L$ and $D$) sequentially in a binary file labeled FFUNIT. This file can be then accessed by SKYSS at the solution phase. Similarly, MTRXYS (or MTRXY) can also read $A$ by blocks from a binary file. Therefore, in all these cases the array `A` is used as workspace.

**The matrix $B$.** There are two cases. If MTYPE=1, $B$ can be read from a sequential binary file labeled FBNAME, containing one record with $n$ words; if MTYPE=2, then the rules of the previous paragraph hold.

## 5 General information

**SKYSCF and SKYPCF.** The subroutine SKYSCF or SKYPCF can perform the factorization of blocked matrices stored in a file (disk) as described in Sections 2 and 4. The auxiliar subroutine SKYSF1 or SKYPF1 (not described in this document but included in the SKYPACK distribution) deals with blocks already factorized and stored in a file, while the subroutine SKYSF2 or SKYPF2 completes the factorization of a block. However, the subroutine SKYSF2 or SKYPF2 can be used in stand alone mode when NBLOCK=1. In this case, SKYSF2 or SKYPF2 has to be called with JMIN=1 and JMAX=$n$.

**Computation performance of SKYPF1 and SKYPF2.** The subroutines SKYPF1 and SKYPF2 are likely to perform well on machines with cache memory. However, their performance may vary depending on the cache size. If no information is available on the cache size, reasonable values for MCOLS and MROWS on an IBM Risc 6000, for example, are 128 and 64, respectively.

**BLAS kernels.** The following BLAS kernels (or their double precision versions) are called by SKYPACK: SSCAL, SAXPY, SDOT, SSYR, SGEMV and STRSM.

**SKYSS.** The subroutine SKYSS requires the reciprocals of $D$. On output, the drivers SKYSCF and SKYPCF provide such reciprocals in `A`. However, SKYPF2 itself does not, in contrast with SKYSF2.

## References

[1] AEA Technology, Didcot, England. *Harwell Subroutine Library.* Release 11 (July 1993).

[2] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, J. Koster, and M. Tuma. *Multifrontal Massively Parallel Solver (MUMPS Version 4.0).* June, 1999.

[3] C. C. Ashcraft, R. G. Grimes, D. J. Pierce, and D. K. Wah. *The User Manual for SPOOLES: Release 1.0: An Object Oriented Software Library for Solving Sparse Linear Sustems of Equations.* Boeing Shared Services Group, Seattle, USA. December, 1997.

[4] K.-J. Bathe. *Finite Element Procedures in Engineering Analysis.* Prentice-Hall, Englewood Cliffs, USA, 1982.

[5] J. W. Demmel, J. R. Gilbert, and X. S. Li. *SuperLU User's Guide.* September, 1999.

[6] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Clarendon Press, Oxford, England, 1986.

[7] T. J. R. Hughes. *The Finite Element Method.* Prentice Hall International Editions, 1987.

[8] M. Joshi, G. Karypis, V. Kumar, A. Gupta, and F. Gustavson. *PSPASES: An Efficient and Scalable Parallel Sparse Direct Solver*, 1999.

[9] O. A. Marques. A Partitioned Skyline $LDL^T$ Factorization. Technical Report TR/PA/93/53, CERFACS, Toulouse, France, 1993.

[10] Padma Raghavan. *MFACT: A Multifrontal Sparse Direct Solver.* University of Tennessee, Knoxville, USA.

[11] O. C. Zienkiewikz and R. L. Taylor. *The Finite Element Method*, volume 1 (Basic Formulation and Linear Problems), 2 (Solid and Fluid Mechanics, Dynamics and Non-Linearity). McGraw Hill International Editions, fourth edition, 1989.