



Rectangular Full Packed Format for Cholesky's Algorithm Factorization, Solution and Inversion

Gustavson, Fred G.; Wasniewski, Jerzy; Dongarra, Jack J.

Publication date:
2008

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Gustavson, F. G., Wasniewski, J., & Dongarra, J. J. (2008). *Rectangular Full Packed Format for Cholesky's Algorithm: Factorization, Solution and Inversion*. Technical University of Denmark, DTU Informatics, Building 321. D T U Compute. Technical Report No. 2008-03

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Rectangular Full Packed Format for Cholesky's Algorithm: Factorization, Solution and Inversion

Fred G. Gustavson

IBM T.J. Watson Research Center

and

Jerzy Waśniewski

Technical University of Denmark

and

Jack J. Dongarra

University of Tennessee, Oak Ridge National Laboratory and University of Manchester

We describe a new data format for storing triangular and symmetric matrices called RFPF (Rectangular Full Packed Format). The standard two dimensional arrays of Fortran and C (also known as full format) that are used to represent triangular and symmetric matrices waste nearly half of the storage space but provide high performance via the use of Level 3 BLAS. Standard packed format arrays fully utilize storage (array space) but provide low performance as there are no Level 3 packed BLAS. We combine the good features of packed and full storage using RFPF to obtain high performance via using Level 3 BLAS as RFPF is a standard full format representation. Also, RFPF requires exactly the same minimal storage as packed format. Each LAPACK full and/or packed symmetric, triangular, and Hermitian routine becomes a single new RFPF routine based on eight possible data layouts of RFPF. This new RFPF routine usually consists of two calls to the corresponding LAPACK full format routine and two calls to Level 3 BLAS routines. This means *no* new software is required. As examples, we present LAPACK routines for Cholesky factorization, solution and inverse computation in RFPF to illustrate this new work and to describe its performance on several commonly used computer platforms. Performance of LAPACK full routines using RFPF versus LAPACK full routines using standard format for both serial and SMP parallel processing is about the same while using half the storage. Performance gains are roughly one to a factor of 33 for serial and one to a factor of 100 for SMP parallel times faster using LAPACK full routines with RFPF than with using LAPACK packed routines. Existing LAPACK routines and vendor LAPACK routines were used in the serial and the SMP parallel study respectively. In the full and packed studies vendor Level 3 BLAS were used.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra – Linear Systems (symmetric and Hermitian); G.4 [**Mathematics of Computing**]: Mathematical Software

General Terms: Algorithms, BLAS, Performance

Additional Key Words and Phrases: real symmetric matrices, complex Hermitian matrices, positive definite matrices, Cholesky factorization and solution, recursive algorithms, novel packed matrix data structures.

Authors' addresses: F.G. Gustavson, IBM T.J. Watson Research Center, Yorktown Heights, NY-10598, USA, email: fg2@us.ibm.com; J. Waśniewski, Department of Informatics and Mathematical Modelling, Technical University of Denmark, Richard Petersens Plads, Building 321, DK-2800 Kongens Lyngby, Denmark, email: jw@imm.dtu.dk; J.J. Dongarra, Electrical Engineering and Computer Science Department, University of Tennessee, 1122 Volunteer Blvd, Knoxville, TN 37996-3450, USA, email: dongarra@cs.utk.edu.

1. INTRODUCTION

A very important class of linear algebra problems deals with a coefficient matrix A that is symmetric and positive definite [Dongarra et al. 1998; Demmel 1997; Golub and Van Loan 1996; Trefethen and Bau 1997]. Because of symmetry it is only necessary to store either the upper or lower triangular part of the matrix A .

Fig. 1. The **full** format array layout of an order N symmetric matrix required by LAPACK. LAPACK requires $LDA \geq N$. Here we set $LDA=N=7$.

| Lower triangular case | Upper triangular case |
|---|--|
| $\begin{pmatrix} 1 & & & & & & \\ 2 & 9 & & & & & \\ 3 & 10 & 17 & & & & \\ 4 & 11 & 18 & 25 & & & \\ 5 & 12 & 19 & 26 & 33 & & \\ 6 & 13 & 20 & 27 & 34 & 41 & \\ 7 & 14 & 21 & 28 & 35 & 42 & 49 \end{pmatrix}$ | $\begin{pmatrix} 1 & 8 & 15 & 22 & 29 & 36 & 43 \\ & 9 & 16 & 23 & 30 & 37 & 44 \\ & & 17 & 24 & 31 & 38 & 45 \\ & & & 25 & 32 & 39 & 46 \\ & & & & 33 & 40 & 47 \\ & & & & & 41 & 48 \\ & & & & & & 49 \end{pmatrix}$ |

Fig. 2. The **packed** format array layout of an order 7 symmetric matrix required by LAPACK.

| Lower triangular case | Upper triangular case |
|--|--|
| $\begin{pmatrix} 1 & & & & & & \\ 2 & 8 & & & & & \\ 3 & 9 & 14 & & & & \\ 4 & 10 & 15 & 19 & & & \\ 5 & 11 & 16 & 20 & 23 & & \\ 6 & 12 & 17 & 21 & 24 & 26 & \\ 7 & 13 & 18 & 22 & 25 & 27 & 28 \end{pmatrix}$ | $\begin{pmatrix} 1 & 2 & 4 & 7 & 11 & 16 & 22 \\ & 3 & 5 & 8 & 12 & 17 & 23 \\ & & 6 & 9 & 13 & 18 & 24 \\ & & & 10 & 14 & 19 & 25 \\ & & & & 15 & 20 & 26 \\ & & & & & 21 & 27 \\ & & & & & & 28 \end{pmatrix}$ |

1.1 LAPACK full and packed storage formats

The LAPACK library [Anderson et al. 1999] offers two different kinds of subroutines to solve the same problem: POTRF¹ and PPTRF both factorize symmetric, positive definite matrices by means of the Cholesky algorithm. A major difference in these two routines is the way they access the array holding the triangular matrix (see figures 1 and 2).

In the POTRF case, the matrix is stored in one of the lower left or upper right triangles of a full square matrix ([Anderson et al. 1999, pages 139 and 140] and

¹Four names SPOTRF, DPOTRF, CPOTRF and ZPOTRF are used in LAPACK for real symmetric and complex Hermitian matrices [Anderson et al. 1999], where the first character indicates the precision and arithmetic versions: S – single precision, D – double precision, C – complex and Z – double complex. LAPACK95 uses one name LA_POTRF for all versions [Barker et al. 2001]. In this paper, POTRF and/or PPTRF express, any precision, any arithmetic and any language version of the PO and/or PP matrix factorization algorithms.

[IBM 1997, page 64])², the other triangle is wasted (see figure 1). Because of the uniform storage scheme, blocked LAPACK and Level 3 BLAS [Dongarra et al. 1990b; Dongarra et al. 1990a] subroutines can be employed, resulting in a fast solution.

In the PPTRF case, the matrix is kept in *packed* storage ([Anderson et al. 1999, pages 140 and 141], [Agarwal et al. 1994] and [IBM 1997, pages 74 and 75]), which means that the columns of the lower or upper triangle are stored consecutively in a one dimensional array (see figure 2). Now the triangular matrix occupies the strictly necessary storage space but the nonuniform storage scheme means that use of full storage BLAS is impossible and only the Level 2 BLAS [Lawson et al. 1979; Dongarra et al. 1988] packed subroutines can be employed, resulting in a slow solution.

To summarize: LAPACK offers a choice between high speed with double the waste of memory space versus low speed and no waste of memory space.

1.2 Packed Minimal Storage Data Formats related to RPPF

Recently many new data formats for matrices have been introduced for improving the performance of Dense Linear Algebra (DLA) algorithms. The survey article [Elmroth et al. 2004] gives an excellent overview.

Recursive Packed Format: [Andersen et al. 2001; Andersen et al. 2002] A new compact way to store a symmetric or triangular matrix called Recursive Packed Format (RPF) is described in [Andersen et al. 2001] as are novel ways to transform RPF to and from standard packed format. New algorithms, called Recursive Packed Cholesky (RPC) [Andersen et al. 2001; Andersen et al. 2002] that operate on the RPF format are presented here. RPF format operates almost entirely by calling Level 3 BLAS GEMM [Dongarra et al. 1990b; Dongarra et al. 1990a] but requires variants of algorithms TRSM and SYRK [Dongarra et al. 1990b; Dongarra et al. 1990a] that are designed to work on RPF. We call these algorithms RPTRSM and RPSYRK [Andersen et al. 2001] and find that they do most of their FLOPS by calling GEMM [Dongarra et al. 1990b; Dongarra et al. 1990a]. It follows that almost all of execution time of the RPC algorithm is done in calls to GEMM.

There are three advantages of this storage scheme compared to traditional packed and full storage. First, the RPF storage format uses the minimum amount of storage required for symmetric, triangular, or Hermitian matrices. Second, the RPC algorithm is a Level 3 implementation of Cholesky factorization. Finally, RPF requires no block size tuning parameter. A disadvantage of the RPC algorithm was that it had a high recursive calling overhead. The paper [Gustavson and Jonsson 2000] removed this overhead and added other novel features to the RPC algorithm.

Square Block Packed Format (SBPF) [Gustavson 2003]: SBPF is described in Section 4 of [Gustavson 2003]. A strong point of SBPF is that it requires minimum block storage and all its blocks are contiguous and of equal size. If one uses SBPF with kernel routines then data copying is mostly eliminated during Cholesky factorization.

Block Packed Hybrid Format (BPHF) [Andersen et al. 2005; Gustavson et al. 2007]: We consider an efficient implementation of the Cholesky solution of symmetric positive-definite full linear systems of equations using packed storage. We

²In Fortran column major, in C row major.

take the same starting point as that of LINPACK [Dongarra et al. 1979] and LAPACK [Anderson et al. 1999], with the upper (or lower) triangular part of the matrix being stored by columns. Following LINPACK [Dongarra et al. 1979] and LAPACK [Anderson et al. 1999], we overwrite the given matrix by its Cholesky factor. The paper [Andersen et al. 2005] uses the BPHF where blocks of the matrix are held contiguously. The paper compares BPHF versus conventional full format storage, packed format and the RPF for the algorithms its consider in its timing studies. BPF is a variant of SBPF in which the diagonal blocks are stored in packed format and so its storage requirement is equal to that of packed storage.

We mention that for packed matrices SBPF and BPHF have become the format of choice for multi-core processors when one stores the blocks in register block format [Gustavson et al. 2006]. Recently, there have been many papers published on new algorithms for multi-core processors. This literature is extensive. So, we only mention two projects, PLASMA [Buttari et al. 2007] and FLAME [Chan et al. 2007], and refer the interested reader to the literature for additional references.

In regard to other references on NDS, the survey article [Elmroth et al. 2004] gives an excellent overview. However, since 2005 at least two new data formats for Cholesky type factorizations have emerged, [Herrero 2006] and the subject matter of this paper, RFPF [Gustavson and Waśniewski 2006]. In the next sub-section we highlight the main features of the RFPF.

1.3 A novel way of representing symmetric, triangular and Hermitian matrices in LAPACK

LAPACK has two types of subroutines for symmetric, triangular, and Hermitian matrices called packed and full format routines. LAPACK has about 300 these kind of subroutines. So, in either format, a variety of problems can be solved by these LAPACK subroutines. The RFPF can replace both these LAPACK data formats. Furthermore, and this is important, using RFPF does not require any new LAPACK subroutines to be written. Using RFPF in LAPACK only requires the use of already existing LAPACK and BLAS routines.

1.4 Overview of the Paper

First we introduce the RFPF in general, see section 2. Secondly we show how to use RFPF on symmetric and Hermitian positive definite matrices; e.g., for the factorization (section 3), solution (section 4), and inversion (section 5) of these matrices. Section 6 describes LAPACK subroutines for the factorization, solution, and inversion of symmetric and Hermitian positive definite matrices using RFPF. Section 7 indicates that the stability results of using RFPF is unaffected by this format choice as RFPF uses existing LAPACK algorithms which are already known to be stable. Section 8 describes a variety of performance results on commonly used platforms both for serial and parallel SMP execution. These results show that performance of LAPACK full routines using RFPF versus LAPACK full routines using standard format for both serial and SMP parallel processing is about the same while using half the storage. Also, performance gains are roughly one to a factor of 33 for serial and one to a factor of 100 for SMP parallel times faster using LAPACK full routines with RFPF than with using LAPACK packed routines. Existing LAPACK routines and vendor LAPACK routines were used in the serial

and the SMP parallel study respectively. In the full and packed studies vendor Level 3 BLAS were used. Section 9 gives a short summary and brief conclusions.

LAPACK software using the RFPP is already written and well tested. In particular, it has passed all the LAPACK test for Cholesky and positive definite Hermitian routines. It will be separately announced by [Gustavson et al. 2007].

2. DESCRIPTION OF RECTANGULAR FULL PACKED FORMAT

We describe Rectangular Full Packed Format (RFPP). It transforms a standard Packed Array AP of size $NT = N(N + 1)/2$ to a full 2D array. This means that performance of LAPACK's [Anderson et al. 1999] packed format routines becomes equal to or better than their full array counterparts. RFPP is a variant of Hybrid Full Packed (HFP) format [Gunnels and Gustavson 2004]. RFPP is a rearrangement of a Standard full format rectangular Array SA of size $LDA * N$ where $LDA \geq N$. Array SA holds a triangular part of a symmetric, triangular, or Hermitian matrix A of order N . The rearrangement of array SA is equal to compact full format Rectangular Array AR of size $LDA1 * N1 = NT$ and hence array AR like array AP uses minimal storage. Array AR will hold a full rectangular matrix A_R obtained from a triangle of matrix A . Note also that the transpose of the rectangular matrix A_R^T resides in the transpose of array AR and hence also represents A . Therefore, Level 3 BLAS [Dongarra et al. 1990b; Dongarra et al. 1990a] can be used on array AR or its transpose. In fact, with the equivalent LAPACK algorithm which uses the array AR or its transpose, the performance is slightly better than standard LAPACK algorithm which uses the array SA or its transpose. Therefore, this offers the possibility to replace all packed or full LAPACK routines with equivalent LAPACK routines that work on array AR or its transpose. For examples of transformations of a matrix A to a matrix A_R see the figures in Section 6.

RFPP is closely related to HFP format, see [Gunnels and Gustavson 2004], which represents A as the concatenation of two standard full arrays whose total size is also NT . A basic simple idea leads to both formats. Let A be an order N symmetric matrix. Break A into a block 2×2 form

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} \text{ or } A = \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} \quad (1)$$

where A_{11} and A_{22} are symmetric. Clearly, we need only store the lower triangles of A_{11} and A_{22} as well as the full matrix $A_{21} = A_{12}^T$ when we are interested in a lower triangular formulation.

When $N = 2k$ is even, the lower triangle of A_{11} and the upper triangle of A_{22}^T can be concatenated together along their main diagonals into a $(k + 1) \times k$ dense matrix (see the figures where N is even in Section 6). This last operation is the crux of the basic simple idea. The off-diagonal block A_{21} is $k \times k$, and so it can be appended below the $(k + 1) \times k$ dense matrix. Thus, the lower triangle of A can be stored as a single $(n + 1) \times k$ dense matrix A_R . In effect, each block matrix A_{11} , A_{21} and A_{22} is now stored in 'full format'. This means all entries of matrix A_R in array AR of size $LDA1 = n + 1$ by $N1 = k$ can be accessed with constant row and column strides. So, the full power of LAPACK's block Level 3 codes are now available for RFPP which uses the minimum amount of storage. Finally, matrix A_R^T which has

size $k \times (N + 1)$ is represented in the transpose of array `AR` and hence has the same desirable properties. There are eight representations of RFPF. The matrix A can have either odd or even order N , or it can be represented either in standard lower or upper format or it can be represented by either matrix A_R or its transpose A_R^T giving $2^3 = 8$ representations in all.

All eight cases or representations are presented in Section 6. The RFPF matrices are in the upper right part of the figures. We have introduced colors and horizontal lines to try to visually delineate triangles T_1, T_2 representing lower, upper triangles of symmetric matrices A_{11}, A_{22}^T respectively and square or near square S_1 representing matrices A_{21} . For an upper triangle of A , T_1, T_2 represents lower, upper triangles of symmetric matrices A_{11}^T, A_{22} respectively and square or near square S_1 representing matrices A_{12} . For both lower and upper triangles of A we have, after each $a_{i,j}$, added its position location in the arrays holding matrices A and A_R .

We now consider performance aspects of using RFPF in the context of using LAPACK routines on triangular matrices stored in RFPF. Let X be a Level 3 LAPACK routine that operates either on standard packed or full format. X has a full Level 3 LAPACK block 2×2 algorithm, call it FX . We write a simple related partition algorithm (SRPA) with partition sizes $n1$ and $n2$ where $n1 + n2 = N$. Apply the new SRPA using the new RFPF. The new SRPA almost always has four major steps consisting entirely of calls to existing full format LAPACK routines in two steps and calls to Level 3 BLAS in the remaining two steps, see fig. 3.

Fig. 3. Simple related partition algorithm (SRPA) of RFPF

| | |
|--|--|
| call FX('L',n1,T1,ldt) ! step 1 | call L3BLAS(n1,n2,S,lds,'U',T2,ldt) ! step 3 |
| call L3BLAS(n1,n2,'L',T1,ldt,S,lds) ! step 2 | call FX('U',n2,T2,ldt) ! step 4 |

Section 6 shows FX algorithms equal to factorization, solution and inversion algorithms on symmetric positive definite or Hermitian matrices.

3. CHOLESKY FACTORIZATION USING RECTANGULAR FULL PACKED FORMAT

The Cholesky factorization of a symmetric and positive definite matrix A can be expressed as

$$\begin{aligned} A &= LL^T \text{ or } A = U^T U \text{ (in the symmetric case)} \\ A &= LL^H \text{ or } A = U^H U \text{ (in the Hermitian case)} \end{aligned} \quad (2)$$

where L and U are lower triangular and upper triangular matrices.

Break the matrices L and U into 2×2 block form in the same way as was done for the matrix A in equation (1):

$$L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \text{ and } U = \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \quad (3)$$

We now have

$$\begin{aligned}
 & \text{the symmetric case:} \\
 LL^T &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix} \text{ and } U^T U = \begin{bmatrix} U_{11}^T & 0 \\ U_{12}^T & U_{22}^T \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \\
 & \text{and the Hermitian case:} \\
 LL^H &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^H & L_{21}^H \\ 0 & L_{22}^H \end{bmatrix} \text{ and } U^H U = \begin{bmatrix} U_{11}^H & 0 \\ U_{12}^H & U_{22}^H \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}
 \end{aligned} \tag{4}$$

where L_{11} , L_{22} , U_{11} , and U_{22} are lower and upper triangular submatrices, and L_{21} and U_{12} are square or almost square submatrices. The block values of L or U are stored over the of A .

Using equations (2) and equating the blocks of equations (1) and equations (4) gives us the basis of a 2×2 block algorithm for Cholesky factorization using RFPF. We can now express each of these four block equalities by calls to existing LAPACK and Level 3 BLAS routines. An example, see Section 6, of this is the three block equations is $L_{11}L_{11}^T = A_{11}$, $L_{21}L_{11}^T = A_{21}$ and $L_{21}L_{21}^T + L_{22}L_{22} = A_{22}$. The first and second of these block equations are handled by calling LAPACK's POTRF routine $L_{11} \leftarrow A_{11}$ and by calling Level 3 BLAS TRSM via $L_{21} \leftarrow L_{21}L_{11}^{-T}$. In both these block equations the Fortran equality of replacement (\leftarrow) is being used so that the lower triangle of A_{11} is being replaced L_{11} and the nearly square matrix A_{21} is being replaced by L_{21} . The third block equation breaks into two parts: $A_{22} \leftarrow L_{21}L_{21}^T$ and $L_{22} \leftarrow A_{22}$ which are handled by calling Level 3 BLAS SYRK or HERK and by calling LAPACK's POTRF routine. At this point we can use the flexibility of the LAPACK library. In RFPF A_{22} is in upper format (upper triangle) while in standard format A_{22} is in lower format (lower triangle). Due to symmetry, both formats of A_{22} contain equal values. This flexibility allows LAPACK to accommodate both formats. Hence, in the calls to SYRK or HERK and POTRF we set uplo = 'U' even though the rectangular matrix of SYRK and HERK comes from a lower triangular formulation.

New LAPACK like routine PFTRF [Gustavson et al. 2007] performs these four computations. PF stands for Positive Full, and was chosen to fit with LAPACK's use of PO and PP. The PF routine covers the Cholesky Factorization algorithm for the eight cases of the RFPF. Section 6 has a figure 4 with four subfigures. Here we are interested here in the first and second subfigure. The first subfigure contains the layouts of matrices A and A_R . The second subfigure has the Cholesky factorization algorithm obtained by simple algebraic manipulations of the three block equalities obtained above.

4. SOLUTION

In Section 3 we obtained the 2 by 2 Cholesky factorization (3) of matrix A . Now, we can solve the equation $AX = B$:

- If A has lower triangular format then

$$\begin{aligned}
 LY &= B \text{ and } L^T X = Y \text{ (in the symmetric case)} \\
 LY &= B \text{ and } L^H X = Y \text{ (in the Hermitian case)}
 \end{aligned} \tag{5}$$

- If A has an upper triangular format then

$$\begin{aligned} U^T Y &= B \text{ and } UX = Y \text{ (in the symmetric case)} \\ U^H Y &= B \text{ and } UX = Y \text{ (in the Hermitian case)} \end{aligned} \quad (6)$$

B , X and Y are either vectors or rectangular matrices. B contains the RHS values. X and Y contain the solution values. B , X and Y are vectors when there is one RHS and matrices when there are many RHS. The values of X and Y are stored over the values of B .

Expanding (5) and (6) using (3) gives the forward substitution equations

$$\begin{aligned} &\text{in the symmetric case:} \\ \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} &= \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \text{ and } \begin{bmatrix} U_{11}^T & 0 \\ U_{12}^T & U_{22}^T \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \end{aligned}, \quad (7)$$

$$\begin{aligned} &\text{and in the Hermitian case:} \\ \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} &= \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \text{ and } \begin{bmatrix} U_{11}^H & 0 \\ U_{12}^H & U_{22}^H \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \end{aligned}$$

and the back substitution equations

$$\begin{aligned} &\text{in the symmetric case:} \\ \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} &= \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \text{ and } \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \end{aligned}. \quad (8)$$

$$\begin{aligned} &\text{and in the Hermitian case:} \\ \begin{bmatrix} L_{11}^H & L_{21}^H \\ 0 & L_{22}^H \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} &= \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \text{ and } \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}. \end{aligned}$$

The equations (7) and (8) gives the basis of a 2×2 block algorithm for Cholesky solution using RFPF format. We can now express these two sets of two block equalities by using existing Level 3 BLAS routines. An example, see Section 6, of the first set of these two block equalities is $L_{11}Y_1 = B_1$ and $L_{21}Y_1 + L_{22}Y_2 = B_2$. The first block equality is handled by Level 3 BLAS TRSM: $Y_1 \leftarrow L_{11}^{-1}B_1$. The second block equality is handled by Level 3 BLAS GEMM and TRSM: $B_2 \leftarrow B_2 - L_{21}Y_1$ and $Y_2 \leftarrow L_{22}^{-1}Y_2$. The backsolution routines are similarly derived. One gets $X_2 \leftarrow L_{22}^{-T}Y_2$, $Y_1 \leftarrow Y_1 - L_{21}^T X_2$ and $X_1 \leftarrow L_{11}^{-T}Y_1$.

New LAPACK like routine PFTRS performs these two solution computations for the eight cases of RFPF. PFTRS calls a new Level 3 BLAS TFSSM in the same way that POTRS calls TRSM. The third subfigure in Section 6 gives the Cholesky solution algorithm using RFPF obtained by simple algebraic manipulation of the block equations (7) and (8).

5. INVERSION

Following LAPACK we consider the following three stage procedure:

- (1) Factorize the matrix A and overwrite A with either L or U by calling PFTRF; see section 3.
- (2) Compute the inverse of either L or U . Call these matrices W or V and overwrite either L or U with them. This is done by calling new routine new LAPACK like TFTRF or HFTRF.

- (3) Calculate either the product $W^T W$ or $V V^T$ and overwrite either W or V with them.

As in Sections 3 and 4 we examine 2 by 2 block algorithms for the steps two and three above. In Section 3 we obtain either matrices L or U in RFPF. Like LAPACK inversion algorithms for POTRI and PPTRI, this is our starting point for our LAPACK inversion algorithm using RFPF. The LAPACK inversion algorithms for POTRI and PPTRI also follow from steps two and three above by first calling in the full case LAPACK TRTRI and then calling LAPACK LAUUM.

Take the inverse of equation (2) and obtain

$$\begin{aligned} A^{-1} &= W^T W \text{ or } A^{-1} = V V^T \text{ (in the symmetric case)} \\ A^{-1} &= W^H W \text{ or } A^{-1} = V V^H \text{ (in the Hermitian case)} \end{aligned} \quad (9)$$

where W and V are lower and upper triangular matrices.

Using the 2 by 2 blocking for either L or U in equation (3) we obtain the following 2 by 2 blocking for W and V :

$$W = \begin{bmatrix} W_{11} & 0 \\ W_{21} & W_{22} \end{bmatrix} \text{ and } V = \begin{bmatrix} V_{11} & V_{12} \\ 0 & V_{22} \end{bmatrix} \quad (10)$$

From the identities $WL = LW = I$ and $VU = UV = I$ and the 2 by 2 block layouts of equations (3) and (3) we obtain three block equations for W and V which can be solved using LAPACK routines for TRTRI and Level 3 BLAS TRMM. An example, see Figure 4, of these three block equations is $L_{11}W_{11} = I$, $L_{21}W_{11} + L_{22}W_{21} = 0$ and $L_{22}W_{22} = I$. The first and third of these block equations are handled by LAPACK TRTRI routines as $W_{11} \leftarrow L_{11}^{-1}$ and $V_{22} \leftarrow U_{22}^{-1}$. In the second inverse computation we use the fact that L_{22} is equally represented by its transpose L_{22}^T which is U_{22} in RFPF. The second block equation leads to two calls to Level 3 BLAS TRMM via $L_{21} \leftarrow -L_{21}W_{11}$ and $W_{21} = W_{22}L_{21}$. In the last two block equations the Fortran equality of replacement (\leftarrow) is being used so that $W_{21} = -W_{22}L_{21}W_{11}$ is replacing L_{21} .

Now we turn to part three of the three stage LAPACK procedure above. For this we use the 2 by 2 blocks layouts of equation (10) and the matrix multiplications indicated by following block equations (11) giving

$$\begin{aligned} &\text{the symmetric case:} \\ W^T W &= \begin{bmatrix} W_{11}^T & W_{21}^T \\ 0 & W_{22}^T \end{bmatrix} \begin{bmatrix} W_{11} & 0 \\ W_{21} & W_{22} \end{bmatrix} \text{ and } V V^T = \begin{bmatrix} V_{11} & V_{12} \\ 0 & V_{22} \end{bmatrix} \begin{bmatrix} V_{11}^T & 0 \\ V_{12}^T & V_{22}^T \end{bmatrix} \end{aligned} \quad (11)$$

$$\begin{aligned} &\text{and the Hermitian case:} \\ W^H W &= \begin{bmatrix} W_{11}^H & W_{21}^H \\ 0 & W_{22}^H \end{bmatrix} \begin{bmatrix} W_{11} & 0 \\ W_{21} & W_{22} \end{bmatrix} \text{ and } V V^H = \begin{bmatrix} V_{11} & V_{12} \\ 0 & V_{22} \end{bmatrix} \begin{bmatrix} V_{11}^H & 0 \\ V_{12}^H & V_{22}^H \end{bmatrix} \end{aligned}$$

where W_{11} , W_{22} , V_{11} , and V_{22} are lower and upper triangular submatrices, and W_{21} and V_{12} are square or almost square submatrices. The values of the indicated block multiplications of W or V in equation (11) are stored over the block values of W or V .

Performing the indicated 2 by 2 block multiplications of equation (11) leads to three block matrix computations. An example, see Section 6, of these three block

computations is $W_{11}^T W_{11} + W_{21}^T W_{21}$, $W_{22}^T W_{21}$ and $W_{22}^T W_{22}$. Additionally, we want to overwrite the values of these block multiplications on their original block operands. Block operand W_{11} only occurs in the 11 block operand computation and hence can be overwritten by a call to LAPACK LAUUM, $W_{11} \leftarrow W_{11}^T W_{11}$, followed by a call to Level 3 BLAS SYRK or HERK, $W_{11} \leftarrow W_{11} + W_{21}^T W_{21}$. Block operand W_{21} now only occurs in the 21 block computation and hence can be overwritten by a call to Level 3 BLAS TRMM, $W_{21} \leftarrow W_{22}^T W_{21}$. Finally, block operand W_{22} can be overwritten by a call to LAPACK LAUUM, $W_{22} \leftarrow W_{22}^T W_{22}$.

The fourth subfigure in Section 6 has the Cholesky inversion algorithms using RFPF based on the results of this Section. New LAPACK routine, PFTRI, performs this computation for the eight cases of RFPF.

6. RFP DATA FORMATS AND ALGORITHMS

This section contains three figures.

- (1) The first figure describes the RFPF (Rectangular Full Packed Format) and gives algorithms for Cholesky factorization, solution and inversion of symmetric positive definite matrices, where N is odd, uplo = 'lower', and 'trans' = 'no transpose'. This figure has four subfigures.
 - (a) The first subfigure depicts the lower triangle of a symmetric positive definite matrix A in **standard full** and its representation by the matrix A_R in **RFPF**.
 - (b) The second subfigure gives the RFPF factorization algorithm and its calling sequences of the LAPACK and BLAS subroutines, see Section 3.
 - (c) The third subfigure gives the RFPF solution algorithm and its calling sequences to the LAPACK and BLAS subroutines, see Section 4.
 - (d) The fourth subfigure in each figure gives the RFPF inversion algorithm and its calling sequences to the LAPACK and BLAS subroutines, see Section 5.
- (2) The second figure shows the transformation from full to RFPF of all "no transform" cases.
- (3) The third figure depicts all eight cases in RFPF.

The data format for A has LDA = N . Matrix A_R has LDAR = N if N is odd and LDAR = $N + 1$ if N is even and $n1$ columns where $n1 = \lceil N/2 \rceil$. Hence, matrix A_R always has LDAR rows and $n1$ columns. Matrix A_R^T always has $n1$ rows and LDAR columns and its leading dimension is equal to $n1$. Matrix A_R always has LDAR \times $n1 = NT = N(N + 1)/2$ elements as does matrix A_R^T .

The order N of matrix A in the first figure is seven and six or seven in the remaining two figures.

7. STABILITY OF THE RFPF ALGORITHM

The algebraic manipulation, how we developed the RFPF factorization (Section 3), solution (Section 4), and inversion (Section 5) algorithms are equivalent to the traditional algorithms in the books [Dongarra et al. 1998; Demmel 1997; Golub and Van Loan 1996; Trefethen and Bau 1997]. The whole theory of the traditional Cholesky factorization, solution, inversion and BLAS algorithms carries over to this three Cholesky and BLAS algorithms described in Sections 3, 4, and 5. The

Fig. 4. The Cholesky factorization algorithm using the Rectangular Full Packed Format (RFPF) if N is odd, uplo='lower', and trans='no transpose'.

| A of LAPACK full data format LDA=N = 7, memory needed LDA × N = 49 | A_R of Rectangular full packed LDAR=N = 7, memory needed LDAR × n1 = 28 |
|--|--|
| $\begin{pmatrix} a_{1,1} & & & & & & \\ a_{2,12} & a_{2,29} & & & & & \\ a_{3,13} & a_{3,210} & a_{3,317} & & & & \\ a_{4,14} & a_{4,211} & a_{4,318} & a_{4,425} & & & \\ a_{5,15} & a_{5,212} & a_{5,319} & a_{5,426} & a_{5,533} & & \\ a_{6,16} & a_{6,213} & a_{6,320} & a_{6,427} & a_{6,534} & a_{6,641} & \\ a_{7,17} & a_{7,214} & a_{7,321} & a_{7,428} & a_{7,535} & a_{7,642} & a_{7,749} \end{pmatrix}$ | $\begin{pmatrix} a_{1,11} & a_{5,58} & a_{6,515} & a_{7,522} \\ a_{2,12} & a_{2,29} & a_{6,616} & a_{7,623} \\ a_{3,13} & a_{3,210} & a_{3,317} & a_{7,724} \\ a_{4,14} & a_{4,211} & a_{4,318} & a_{4,425} \\ a_{5,15} & a_{5,212} & a_{5,319} & a_{5,426} \\ a_{6,16} & a_{6,213} & a_{6,320} & a_{6,427} \\ a_{7,17} & a_{7,214} & a_{7,321} & a_{7,428} \end{pmatrix}$ |
| Matrix A | Matrix A_R |

Factorization Algorithm ($n1 = \lceil N/2 \rceil, n2 = N - n1$):

- | | |
|--|--|
| <ol style="list-style-type: none"> 1) factor $L_{11}L_{11}^T = A_{11}$; call POTRF('L', n1, AR, N, & info); 2) solve $L_{21}L_{11}^T = A_{21}$; call TRSM('R', 'L', 'T', 'N', n2, & n1, one, AR, N, AR(n1 + 1, 1), N); | <ol style="list-style-type: none"> 3) update $A_{22} := A_{22} - L_{21}L_{21}^T$; call SYRK/HERK('U', 'N', n2, n1, & -one, AR(n1 + 1, 1), N, one, AR(1, 2), N); 4) factor $U_{22}^TU_{22} = A_{22}$; call POTRF('U', n2, AR(1, 2), N, & info); |
|--|--|

Solution Algorithm,

where $B(LDB, nr)$ and $LDB \geq N$ (here $LDB = N$):

$$LY = B \qquad L'X = Y$$

- | | |
|--|--|
| <ol style="list-style-type: none"> 1) solve $L_{11}Y_1 = B_1$; call TRSM('L', 'L', 'N', 'N', n1, & nr, one, AR, N, B, N); 2) Multiply $B_2 = B_2 - L_{21}Y_1$; call GEMM('N', 'N', n2, nr, n1, -one, & AR(n1 + 1, 1), N, B, N, one, & B(n1 + 1, 1), N); 3) solve $L_{22}Y_2 = B_2$; call TRSM('L', 'U', 'T', 'N', n2, & nr, one, AR(1, 2), N, B(n1 + 1, 1), N); | <ol style="list-style-type: none"> 1) solve $L_{22}^TX_2 = Y_2$; call TRSM('L', 'U', 'N', 'N', n2, & nr, one, AR(1, 2), N, B(n1 + 1, 1), N); 2) Multiply $Y_1 = Y_1 - L_{21}^TX_2$; call GEMM('T', 'N', n1, nr, n2, -one, & AR(n1 + 1, 1), N, B(n1 + 1, 1), & N, one, B, N); 3) solve $L_{22}^TX_1 = Y_1$; call TRSM('L', 'L', 'T', 'N', n1, & nr, one, AR, N, B, N); |
|--|--|

The Inversion Algorithm :

Inversion

- 1) invert $W_{11} = L_{11}^{-1}$;
call **TRTRI**('L', 'N', n1, AR, N, info);
- 2) Multiply $L_{21} = -L_{21}W_{11}$;
call **TRMM**('R', 'L', 'N', 'N', n2, & n1, -one, AR, N, AR(n1 + 1, 1), N);
- 3) invert $V_{22} = U_{22}^{-1}$;
call **TRTRI**('U', 'N', n2, AR(1, 2), & N, info);
- 4) invert $V_{22} = U_{22}^{-1}$;
call **TRMM**('L', 'U', 'T', 'N', n2, & n1, one, AR(1, 2), N, AR(n1 + 1, 1), N);

Triangular matrix multiplication

- 1) **Triang. Mult.** $W_{11} = W_{11}^TW_{11}$;
call **LAUUM**('L', n1, AR, N, info);
- 2) **update** $W_{11} = W_{11} + W_{21}^TW_{21}$;
call **SYRK/HERK**('L', 'T', n1, n2, & one, AR(n1 + 1, 1), N, one, AR, N);
- 3) **Multiply** $W_{21} = V_{22}W_{21}$;
call **TRMM**('L', 'U', 'N', 'N', n2, & n1, one, AR(1, 2), N, A(n1 + 1, 1), N);
- 4) **Triang. Mult.** $V_{11} = V_{11}V_{11}^T$;
call **LAUUM**('U', n2, AR(1, 2), N, info);

Fig. 5. Eight two-dimensional arrays for storing the matrices A and A_R that are needed by the LAPACK subroutine POTRF (full format) and PFTRF RFPF respectively. The leading dimension LDA is N for LAPACK, and LDAR for RFPF. $LDAR = N$ for N odd, and $N + 1$ for N even. Here N is 7 or 6. The memory needed is $LDA \times N$ for full format and $LDAR \times n1 = (N + 1)N/2$ for RFPF Here 49 and 36 for full format and 28 and 21 for RFPF. The column size of RFPF is $n1 = \lceil N/2 \rceil$, here 4 and 3.

5.1 The matrices A of order N and A_R of size $LDAR$ by $n1$, here $N = 7$.

| | | |
|---|--|---|
| 5.1.1 Full Format | | 5.1.2 RFPF |
| $\left[\begin{array}{cccc ccc} a_{1,1} & \diamond & \diamond & \diamond & \diamond & \diamond & \diamond \\ a_{2,1} & a_{2,2} & \diamond & \diamond & \diamond & \diamond & \diamond \\ a_{3,1} & a_{3,2} & a_{3,3} & \diamond & \diamond & \diamond & \diamond \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & \diamond & \diamond & \diamond \\ \hline a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & \diamond & \diamond \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} & \diamond \\ a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & a_{7,5} & a_{7,6} & a_{7,7} \end{array} \right],$ | | $\left[\begin{array}{ccc ccc} a_{1,1} & a_{5,5} & a_{6,5} & a_{7,5} \\ a_{2,1} & a_{2,2} & a_{6,6} & a_{7,6} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{7,7} \\ \hline a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \\ \hline a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} \\ a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} \end{array} \right]$ |

5.2 The matrices A of order N and A_R of size $LDAR$ by $n1$, here $N = 6$.

| | | |
|---|--|---|
| 5.2.1 Full format | | 5.2.2 RFPF |
| $\left[\begin{array}{ccc ccc} a_{1,1} & \diamond & \diamond & \diamond & \diamond & \diamond \\ a_{2,1} & a_{2,2} & \diamond & \diamond & \diamond & \diamond \\ a_{3,1} & a_{3,2} & a_{3,3} & \diamond & \diamond & \diamond \\ \hline a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & \diamond & \diamond \\ a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & a_{5,5} & \diamond \\ a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & a_{6,5} & a_{6,6} \end{array} \right],$ | | $\left[\begin{array}{ccc ccc} a_{4,4} & a_{5,4} & a_{6,4} \\ a_{1,1} & a_{5,5} & a_{6,5} \\ a_{2,1} & a_{2,2} & a_{6,6} \\ \hline a_{3,1} & a_{3,2} & a_{3,3} \\ \hline a_{4,1} & a_{4,2} & a_{4,3} \\ a_{5,1} & a_{5,2} & a_{5,3} \\ a_{6,1} & a_{6,2} & a_{6,3} \end{array} \right]$ |

5.3 The matrices A of order N and A_R of size $LDAR$ by $n1$, here $N = 7$.

| | | |
|---|--|--|
| 5.3.1 Full format | | 5.3.2 RFPF |
| $\left[\begin{array}{ccc cccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} \\ \diamond & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} \\ \diamond & \diamond & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} \\ \hline \diamond & \diamond & \diamond & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} \\ \diamond & \diamond & \diamond & \diamond & a_{5,5} & a_{5,6} & a_{5,7} \\ \diamond & \diamond & \diamond & \diamond & \diamond & a_{6,6} & a_{6,7} \\ \diamond & \diamond & \diamond & \diamond & \diamond & \diamond & a_{7,7} \end{array} \right],$ | | $\left[\begin{array}{cccc ccc} a_{1,4} & a_{1,5} & a_{1,6} & a_{1,7} \\ a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} \\ a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} \\ \hline a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} \\ \hline a_{1,1} & a_{5,5} & a_{5,6} & a_{5,7} \\ a_{1,2} & a_{2,2} & a_{6,6} & a_{6,7} \\ a_{1,3} & a_{2,3} & a_{3,3} & a_{7,7} \end{array} \right]$ |

5.4 The matrices A of order N and A_R of size $LDAR$ by $n1$, here $N = 6$.

| | | |
|---|--|---|
| 5.4.1 Full format | | 5.4.2 RFPF |
| $\left[\begin{array}{ccc ccc} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} \\ \diamond & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} \\ \diamond & \diamond & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} \\ \hline \diamond & \diamond & \diamond & a_{4,4} & a_{4,5} & a_{4,6} \\ \diamond & \diamond & \diamond & \diamond & a_{5,5} & a_{5,6} \\ \diamond & \diamond & \diamond & \diamond & \diamond & a_{6,6} \end{array} \right],$ | | $\left[\begin{array}{ccc ccc} a_{1,4} & a_{1,5} & a_{1,6} \\ a_{2,4} & a_{2,5} & a_{2,6} \\ a_{3,4} & a_{3,5} & a_{3,6} \\ \hline a_{4,4} & a_{4,5} & a_{4,6} \\ \hline a_{1,1} & a_{5,5} & a_{5,6} \\ a_{1,2} & a_{2,2} & a_{6,6} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{array} \right]$ |

Fig. 6. Eight two-dimensional arrays for storing the matrices A_R and A_R^T in RFPF. The leading dimension $LDAR$ of A_R is N when N is odd and $N + 1$ when N is even. For the matrix A_R^T it is $n1 = \lceil N/2 \rceil$. The memory needed for both A_R and A_R^T is $(N + 1)/2 \times N$. This amount is **28** for $N = 7$ and **21** for $N = 6$.

6.1 RFPF for the matrices of rank odd, here $N = 7$ and $n1 = 4$

Lower triangular

$$\begin{array}{c}
 LDAR = N \\
 \left[\begin{array}{cccc|cccc}
 a_{1,1} & a_{5,5} & a_{6,5} & a_{7,5} & & & & \\
 a_{2,1} & a_{2,2} & a_{6,6} & a_{7,6} & & & & \\
 a_{3,1} & a_{3,2} & a_{3,3} & a_{7,7} & & & & \\
 a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} & & & & \\
 \hline
 a_{5,1} & a_{5,2} & a_{5,3} & a_{5,4} & & & & \\
 a_{6,1} & a_{6,2} & a_{6,3} & a_{6,4} & & & & \\
 a_{7,1} & a_{7,2} & a_{7,3} & a_{7,4} & & & &
 \end{array} \right]
 \end{array}$$

Upper triangular

$$\begin{array}{c}
 LDAR = N \\
 \left[\begin{array}{cccc|cccc}
 a_{1,1} & a_{2,1} & a_{3,1} & a_{4,1} & a_{5,1} & a_{6,1} & a_{7,1} & \\
 a_{5,5} & a_{2,2} & a_{3,2} & a_{4,2} & a_{5,2} & a_{6,2} & a_{7,2} & \\
 a_{6,5} & a_{6,6} & a_{3,3} & a_{4,3} & a_{5,3} & a_{6,3} & a_{7,3} & \\
 a_{7,5} & a_{7,6} & a_{7,7} & a_{4,4} & a_{5,4} & a_{6,4} & a_{7,4} &
 \end{array} \right]
 \end{array}$$

6.2 RFPF for the matrices of rank even, here $N = 6$ and $n1 = 3$.

Lower triangular

$$\begin{array}{c}
 LDAR = N + 1 \\
 \left[\begin{array}{ccc|ccc}
 a_{4,4} & a_{5,4} & a_{6,4,15} & & & \\
 a_{1,1} & a_{5,5} & a_{6,5,16} & & & \\
 a_{2,1} & a_{2,2} & a_{6,6} & & & \\
 a_{3,1} & a_{3,2} & a_{3,3} & & & \\
 \hline
 a_{4,1} & a_{4,2} & a_{4,3} & & & \\
 a_{5,1} & a_{5,2} & a_{5,3} & & & \\
 a_{6,1} & a_{6,2} & a_{6,3} & & &
 \end{array} \right]
 \end{array}$$

Upper triangular

$$\begin{array}{c}
 LDAR = N + 1 \\
 \left[\begin{array}{ccc|ccc}
 a_{1,4} & a_{1,5} & a_{1,6} & a_{4,4} & a_{1,1} & a_{1,2} & a_{1,3} \\
 a_{2,4} & a_{2,5} & a_{2,6} & a_{4,5} & a_{5,5} & a_{2,2} & a_{2,3} \\
 a_{3,4} & a_{3,5} & a_{3,6} & a_{4,6} & a_{5,6} & a_{6,6} & a_{3,3} \\
 \hline
 a_{4,4} & a_{4,5,11} & a_{4,6} & & & & \\
 a_{1,1} & a_{5,5} & a_{5,6,19} & & & & \\
 a_{1,2} & a_{2,2} & a_{6,6} & & & & \\
 a_{1,3} & a_{2,3} & a_{3,3} & & & &
 \end{array} \right]
 \end{array}$$

| n | RFPF | | | | LAPACK | | | |
|------|----------|------|-------|------|--------|------|-------|-----|
| | NO TRANS | | TRANS | | POTRF | | PPTRF | |
| | U | L | U | L | U | L | U | L |
| 50 | 456 | 520 | 516 | 482 | 460 | 464 | 291 | 294 |
| 100 | 753 | 813 | 829 | 768 | 612 | 827 | 399 | 369 |
| 200 | 946 | 979 | 997 | 955 | 933 | 1150 | 455 | 370 |
| 400 | 1208 | 1231 | 1158 | 1183 | 1081 | 1244 | 483 | 339 |
| 500 | 1173 | 1227 | 1138 | 1186 | 1121 | 1340 | 511 | 343 |
| 800 | 1316 | 1318 | 1189 | 1269 | 1256 | 1310 | 522 | 324 |
| 1000 | 1275 | 1318 | 1281 | 1303 | 1313 | 1406 | 530 | 288 |
| 1600 | 1350 | 1387 | 1358 | 1312 | 1405 | 1234 | 502 | 223 |
| 2000 | 1264 | 1367 | 1403 | 1323 | 1360 | 1491 | 394 | 163 |
| 4000 | 1287 | 1450 | 1537 | 1263 | 1392 | 1565 | 300 | 153 |

Table 1. Performance in Mflop/s of Cholesky Factorization on SUN UltraSPARC-III computer

| n | RFPF | | | | LAPACK | | | |
|------|----------|------|-------|------|--------|------|-------|-----|
| | NO TRANS | | TRANS | | POTRI | | PPTRI | |
| | U | L | U | L | U | L | U | L |
| 50 | 379 | 379 | 380 | 381 | 330 | 328 | 318 | 338 |
| 100 | 698 | 696 | 699 | 700 | 698 | 707 | 412 | 446 |
| 200 | 1012 | 989 | 1008 | 997 | 1052 | 1030 | 467 | 558 |
| 400 | 1290 | 1223 | 1229 | 1263 | 1229 | 1212 | 452 | 606 |
| 500 | 1290 | 1276 | 1238 | 1330 | 1285 | 1276 | 448 | 595 |
| 800 | 1446 | 1445 | 1330 | 1356 | 1343 | 1325 | 408 | 566 |
| 1000 | 1428 | 1337 | 1442 | 1436 | 1378 | 1318 | 404 | 531 |
| 1600 | 1418 | 1372 | 1369 | 1396 | 1142 | 1317 | 262 | 450 |
| 2000 | 1387 | 1333 | 1370 | 1536 | 1400 | 1366 | 242 | 394 |
| 4000 | 1460 | 1408 | 1389 | 1453 | 1421 | 1395 | 201 | 288 |

Table 2. Performance in Mflop/s of Cholesky Inversion on SUN UltraSPARC-III computer

error analysis and stability of these algorithms is very well described in the book of [Higham 1996]. The difference between LAPACK algorithms PO, PP and RFPF³ is how inner products are accumulated. In each case a different order is used. They are all mathematically equivalent, and, stability analysis shows that any summation order is stable.

8. A PERFORMANCE STUDY USING RFP FORMAT

There are 11 tables giving performance results of LAPACK and RFPF routines. The LAPACK Library routines POTRF, PPTRF, POTRI, TRTRI, LAUUM, PPTRI, POTRS and PPTRS are compared with the RFPF like PFTRF, PFTRI, TFTRI and PFTRS for Cholesky factorization, inverse and solution respectively. In all cases real long precision arithmetic (double precision) is used. Results were

³full, packed and rectangular full packed.

| r h s | n | RFPF | | | | LAPACK | | | |
|-------------|------|----------|------|-------|------|--------|------|-------|-----|
| | | NO TRANS | | TRANS | | POTRS | | PPTRS | |
| | | U | L | U | L | U | L | U | L |
| 100 | 50 | 1132 | 1123 | 1153 | 1135 | 1103 | 1069 | 353 | 353 |
| 100 | 100 | 1193 | 1163 | 1237 | 1211 | 1262 | 1262 | 478 | 478 |
| 100 | 200 | 1477 | 1478 | 1500 | 1490 | 1280 | 1235 | 557 | 554 |
| 100 | 400 | 1494 | 1505 | 1514 | 1534 | 1150 | 1149 | 582 | 582 |
| 100 | 500 | 1466 | 1443 | 1436 | 1445 | 1217 | 1229 | 560 | 569 |
| 100 | 800 | 1503 | 1505 | 1535 | 1469 | 1151 | 1096 | 528 | 526 |
| 100 | 1000 | 1553 | 1524 | 1499 | 1576 | 1089 | 1125 | 513 | 513 |
| 160 | 1600 | 1595 | 1564 | 1603 | 1577 | 1155 | 1121 | 421 | 403 |
| 200 | 2000 | 1600 | 1636 | 1610 | 1615 | 1105 | 1087 | 347 | 338 |
| 400 | 4000 | 1666 | 1668 | 1696 | 1665 | 1080 | 1084 | 292 | 290 |

Table 3. Performance in Mflop/s of Cholesky Solution on SUN UltraSPARC-III computer

| n | RFPF | | | | LAPACK | | | |
|------|----------|------|-------|------|--------|------|-------|------|
| | NO TRANS | | TRANS | | POTRF | | PPTRF | |
| | U | L | U | L | U | L | U | L |
| 50 | 781 | 771 | 784 | 771 | 1107 | 739 | 495 | 533 |
| 100 | 1843 | 1788 | 1848 | 1812 | 1874 | 1725 | 879 | 825 |
| 200 | 3178 | 2869 | 2963 | 3064 | 2967 | 2871 | 1323 | 1100 |
| 400 | 3931 | 3709 | 3756 | 3823 | 3870 | 3740 | 1121 | 1236 |
| 500 | 4008 | 3808 | 3883 | 3914 | 4043 | 3911 | 1032 | 1257 |
| 800 | 4198 | 4097 | 4145 | 4126 | 3900 | 4009 | 612 | 1127 |
| 1000 | 4115 | 4038 | 4015 | 3649 | 3769 | 3983 | 305 | 697 |
| 1600 | 3851 | 3652 | 3967 | 3971 | 3640 | 3987 | 147 | 437 |
| 2000 | 3899 | 3716 | 3660 | 3660 | 3865 | 3835 | 108 | 358 |
| 4000 | 3966 | 3791 | 3927 | 4011 | 3869 | 4052 | 119 | 398 |

Table 4. Performance in Mflop/s of Cholesky Factorization on ia64 Itanium computer

obtained on several different computers using everywhere the vendor Level 3 and Level 2 BLAS.

Due to space limitations, we cannot present all of our timing results. We noticed a few anomalies in the performance runs for POTRS on SUN in Table 3, the PP runs on NEC (Tables 7, 8 and 9) and the PP and PO runs on SUN SMP parallel, Table 11. We have re-run these cases and have obtained the same results. At this time we do *not* have a rational explanation for these anomalies. Finally, our timings do *not* include the cost of sorting any LAPACK data formats to RFPF data formats and vice versa.

The tables from 1 to 9 show the performance comparison in Mflop/s of factorization, inversion and solution on SUN UltraSPARC-III (clock rate: 1200 MHz; L1 cache: 64 kB 4-way data, 32 kB 4-way instruction, and 2 kB Write, 2 kB Prefetch; L2 cache: 8 MB; TLB: 1040 entries), ia64 Itanium (CPU: Intel Itanium2: 1.3 GHz,

| n | RFPF | | | | LAPACK | | | |
|------|----------|------|-------|------|--------|------|-------|------|
| | NO TRANS | | TRANS | | POTRI | | PPTRI | |
| | u | l | u | l | u | l | u | l |
| 50 | 633 | 659 | 648 | 640 | 777 | 870 | 508 | 460 |
| 100 | 1252 | 1323 | 1300 | 1272 | 1573 | 1760 | 815 | 810 |
| 200 | 2305 | 2442 | 2431 | 2314 | 2357 | 2639 | 1118 | 1211 |
| 400 | 3084 | 3199 | 3188 | 3094 | 3152 | 3445 | 1234 | 1363 |
| 500 | 3204 | 3316 | 3329 | 3218 | 3400 | 3611 | 1239 | 1382 |
| 800 | 3617 | 3741 | 3720 | 3640 | 3468 | 3786 | 1182 | 1268 |
| 1000 | 3611 | 3716 | 3637 | 3590 | 3456 | 3790 | 767 | 946 |
| 1600 | 3721 | 3802 | 3795 | 3714 | 3589 | 3713 | 500 | 609 |
| 2000 | 3784 | 3812 | 3745 | 3704 | 3636 | 3798 | 473 | 596 |
| 4000 | 3822 | 3762 | 3956 | 3851 | 3760 | 3750 | 467 | 614 |

Table 5. Performance in Mflop/s of Cholesky Inversion on ia64 Itanium computer

| r h s | n | RFPF | | | | LAPACK | | | |
|-------------|------|----------|------|-------|------|--------|------|-------|------|
| | | NO TRANS | | TRANS | | POTRS | | PPTRS | |
| | | u | l | u | l | u | l | u | l |
| 100 | 50 | 2409 | 2412 | 2414 | 2422 | 3044 | 3018 | 725 | 714 |
| 100 | 100 | 3305 | 3301 | 3303 | 3303 | 3889 | 3855 | 1126 | 1109 |
| 100 | 200 | 4149 | 4154 | 4127 | 4146 | 4143 | 4127 | 1526 | 1512 |
| 100 | 400 | 4398 | 4403 | 4416 | 4444 | 4469 | 4451 | 1097 | 1088 |
| 100 | 500 | 4313 | 4155 | 4374 | 4394 | 4203 | 4093 | 1054 | 1045 |
| 100 | 800 | 3979 | 3919 | 4040 | 4051 | 3969 | 4011 | 692 | 720 |
| 100 | 1000 | 3716 | 3608 | 3498 | 3477 | 3630 | 3645 | 376 | 372 |
| 160 | 1600 | 3892 | 3874 | 4020 | 3994 | 4001 | 4011 | 188 | 182 |
| 200 | 2000 | 4052 | 4073 | 4040 | 4020 | 4231 | 4203 | 119 | 119 |
| 400 | 4000 | 4245 | 4225 | 4275 | 4287 | 4330 | 4320 | 115 | 144 |

Table 6. Performance in Mflop/s of Cholesky Solution on ia64 Itanium computer

cache: 3 MB on-chip L3 cache), and NEC SX-6 computer (8 CPU's, per CPU peak : 8 Gflops, per node peak : 64 Gflops, vector register length: 256). The tables 10 and 11 show the SMP parallelism of these subroutines on the IBM Power4 (clock rate: 1300 MHz; two CPUs per chip; L1 cache: 128 KB (64 KB per CPU) instruction, 64 KB 2-way (32 KB per CPU) data; L2 cache: 1.5 MB 8-way shared between the two CPUs; L3 cache: 32 MB 8-way shared (off-chip); TLB: 1024 entries) and SUN UltraSPARC-IV (the same hardware parameters as SUN UltraSPARC-III except the clock rate: 1350 MHz) computers respectively. They compare SMP times of PFTRF, POTRF and PPTRF. The tables 10 and 11 also show the times of the four operations (POTRF, TRSM, SYRK and again POTRF) inside the new algorithm PFTRF.

| n | RFPF | | | | LAPACK | | | |
|------|----------|------|-------|------|--------|------|-------|------|
| | NO TRANS | | TRANS | | POTRF | | PPTRF | |
| | u | l | u | l | u | l | u | l |
| 50 | 206 | 200 | 225 | 225 | 365 | 353 | 57 | 238 |
| 100 | 721 | 728 | 789 | 788 | 1055 | 989 | 120 | 591 |
| 200 | 2028 | 2025 | 2005 | 2015 | 1380 | 1639 | 246 | 1250 |
| 400 | 3868 | 3915 | 3078 | 3073 | 1763 | 3311 | 479 | 1975 |
| 500 | 4483 | 4470 | 4636 | 4636 | 4103 | 4241 | 585 | 2149 |
| 800 | 5154 | 5168 | 4331 | 4261 | 3253 | 4469 | 870 | 2399 |
| 1000 | 5666 | 5654 | 5725 | 5703 | 5144 | 5689 | 1035 | 2474 |
| 1600 | 6224 | 6145 | 5644 | 5272 | 5375 | 5895 | 1441 | 2572 |
| 2000 | 6762 | 6788 | 6642 | 6610 | 6088 | 6732 | 1654 | 2598 |
| 4000 | 7321 | 7325 | 7236 | 7125 | 6994 | 7311 | 2339 | 2641 |

Table 7. Performance in Mflop/s of Cholesky Factorization on SX-6 NEC computer with Vector Option

| n | RFPF | | | | LAPACK | | | |
|------|----------|------|-------|------|--------|------|-------|------|
| | NO TRANS | | TRANS | | POTRI | | PPTRI | |
| | u | l | u | l | u | l | u | l |
| 50 | 152 | 152 | 150 | 152 | 148 | 145 | 91 | 61 |
| 100 | 430 | 432 | 428 | 432 | 313 | 310 | 194 | 126 |
| 200 | 950 | 956 | 940 | 941 | 636 | 627 | 404 | 249 |
| 400 | 1850 | 1852 | 1804 | 1806 | 1734 | 1624 | 722 | 470 |
| 500 | 2227 | 2228 | 2174 | 2181 | 2180 | 2029 | 856 | 572 |
| 800 | 3775 | 3775 | 3668 | 3686 | 3405 | 3052 | 1186 | 842 |
| 1000 | 4346 | 4346 | 4254 | 4263 | 4273 | 3638 | 1342 | 985 |
| 1600 | 5313 | 5294 | 5137 | 5308 | 5438 | 4511 | 1690 | 1361 |
| 2000 | 6006 | 6006 | 5930 | 5931 | 5997 | 4832 | 1854 | 1536 |
| 4000 | 6953 | 6953 | 6836 | 6888 | 7041 | 4814 | 1921 | 2122 |

Table 8. Performance in Mflop/s of Cholesky Inversion on SX-6 NEC computer with Vector Option

| r h s | n | RFPF | | | | LAPACK | | | |
|-------------|------|----------|------|-------|------|--------|------|-------|------|
| | | NO TRANS | | TRANS | | POTRS | | PPTRS | |
| | | U | L | U | L | U | L | U | L |
| 100 | 50 | 873 | 870 | 889 | 886 | 1933 | 1941 | 88 | 88 |
| 100 | 100 | 2173 | 2171 | 2200 | 2189 | 3216 | 3236 | 181 | 179 |
| 100 | 200 | 4236 | 4230 | 4253 | 4245 | 4165 | 4166 | 352 | 347 |
| 100 | 400 | 5431 | 5431 | 5410 | 5408 | 5302 | 5303 | 648 | 644 |
| 100 | 500 | 5563 | 5562 | 5568 | 5567 | 5629 | 5632 | 783 | 779 |
| 100 | 800 | 6407 | 6407 | 6240 | 6240 | 5569 | 5593 | 1132 | 1128 |
| 100 | 1000 | 6578 | 6578 | 6559 | 6558 | 6554 | 6566 | 1325 | 1320 |
| 160 | 1600 | 6781 | 6805 | 6430 | 6430 | 6799 | 6809 | 1732 | 1727 |
| 200 | 2000 | 7568 | 7569 | 7519 | 7519 | 7406 | 7407 | 1920 | 1914 |
| 400 | 4000 | 7858 | 7858 | 7761 | 7761 | 7626 | 7627 | 2414 | 2410 |

Table 9. Performance in Mflop/s of Cholesky Solution on SX-6 NEC computer with Vector Option

| n | n proc | Mflop/s | | Times | | | | | |
|------|-----------|---------|-------|----------|------|------|-------|--------|--------|
| | | PFTRF | | in PFTRF | | | | LAPACK | |
| | | | | POTRF | TRSM | SYRK | POTRF | POTRF | PPTRF |
| 1000 | 1 | 2695 | 0.12 | 0.02 | 0.05 | 0.04 | 0.02 | 0.12 | 0.94 |
| | 5 | 7570 | 0.04 | 0.01 | 0.02 | 0.01 | 0.01 | 0.03 | 0.32 |
| | 10 | 10699 | 0.03 | 0.01 | 0.01 | 0.01 | 0.00 | 0.02 | 0.16 |
| | 15 | 18354 | 0.02 | 0.00 | 0.01 | 0.00 | 0.00 | 0.01 | 0.11 |
| 2000 | 1 | 2618 | 1.02 | 0.13 | 0.38 | 0.38 | 0.13 | 0.97 | 8.74 |
| | 5 | 10127 | 0.26 | 0.04 | 0.10 | 0.09 | 0.04 | 0.24 | 3.42 |
| | 10 | 17579 | 0.15 | 0.02 | 0.06 | 0.05 | 0.03 | 0.12 | 1.65 |
| | 15 | 23798 | 0.11 | 0.02 | 0.04 | 0.04 | 0.01 | 0.13 | 1.11 |
| 3000 | 1 | 2577 | 3.49 | 0.45 | 1.33 | 1.28 | 0.44 | 3.40 | 30.42 |
| | 5 | 11369 | 0.79 | 0.11 | 0.28 | 0.30 | 0.11 | 0.71 | 11.76 |
| | 10 | 19706 | 0.46 | 0.06 | 0.19 | 0.16 | 0.05 | 0.38 | 6.16 |
| | 15 | 29280 | 0.31 | 0.05 | 0.12 | 0.10 | 0.04 | 0.26 | 4.28 |
| 4000 | 1 | 2664 | 8.01 | 1.01 | 2.90 | 3.09 | 1.01 | 7.55 | 75.72 |
| | 5 | 11221 | 1.90 | 0.26 | 0.68 | 0.72 | 0.24 | 1.65 | 25.73 |
| | 10 | 21275 | 1.00 | 0.13 | 0.39 | 0.36 | 0.12 | 0.86 | 13.95 |
| | 15 | 31024 | 0.69 | 0.09 | 0.28 | 0.24 | 0.08 | 0.59 | 10.46 |
| 5000 | 1 | 2551 | 16.34 | 2.04 | 6.16 | 6.10 | 2.04 | 15.79 | 154.74 |
| | 5 | 11372 | 3.66 | 0.45 | 1.37 | 1.44 | 0.40 | 3.27 | 47.76 |
| | 10 | 22326 | 1.87 | 0.25 | 0.78 | 0.62 | 0.22 | 1.73 | 28.13 |
| | 15 | 32265 | 1.29 | 0.17 | 0.53 | 0.45 | 0.14 | 1.16 | 20.95 |

Table 10. Performance Times and Mflop/s of Cholesky Factorization on an IBM Power 4 computer using SMP parallelism on 1, 5, 10 and 15 processors. Here vendor codes for Level 2 and 3 BLAS and POTRF are used, ESSL library version 3.3. PPTRF is LAPACK code. UPLO = 'L'.

| n | n pr oc | Mflop/s | | Times | | | | | |
|------|---------------|---------|-------|----------|------|------|-------|--------|--------|
| | | PFTRF | | in PFTRF | | | | LAPACK | |
| | | | | POTRF | TRSM | SYRK | POTRF | POTRF | PPTRF |
| 1000 | 1 | 1587 | 0.21 | 0.03 | 0.09 | 0.07 | 0.03 | 0.19 | 1.06 |
| | 5 | 4762 | 0.07 | 0.02 | 0.02 | 0.02 | 0.02 | 0.07 | 1.13 |
| | 10 | 5557 | 0.06 | 0.01 | 0.01 | 0.02 | 0.02 | 0.06 | 1.12 |
| | 15 | 5557 | 0.06 | 0.02 | 0.01 | 0.01 | 0.02 | 0.06 | 1.11 |
| 2000 | 1 | 1668 | 1.58 | 0.22 | 0.63 | 0.52 | 0.22 | 1.45 | 11.20 |
| | 5 | 6667 | 0.40 | 0.07 | 0.13 | 0.13 | 0.07 | 0.38 | 11.95 |
| | 10 | 8602 | 0.31 | 0.06 | 0.07 | 0.11 | 0.07 | 0.25 | 11.24 |
| | 15 | 9524 | 0.28 | 0.06 | 0.06 | 0.08 | 0.08 | 0.23 | 11.66 |
| 3000 | 1 | 1819 | 4.95 | 0.62 | 1.98 | 1.72 | 0.63 | 4.86 | 45.48 |
| | 5 | 6872 | 1.31 | 0.20 | 0.42 | 0.48 | 0.20 | 1.38 | 55.77 |
| | 10 | 12162 | 0.74 | 0.14 | 0.22 | 0.21 | 0.16 | 0.76 | 46.99 |
| | 15 | 12676 | 0.71 | 0.14 | 0.16 | 0.30 | 0.16 | 0.61 | 45.71 |
| 4000 | 1 | 1823 | 11.70 | 1.52 | 4.62 | 4.01 | 1.55 | 11.86 | 112.52 |
| | 5 | 7960 | 2.68 | 0.40 | 0.94 | 0.92 | 0.42 | 2.74 | 112.77 |
| | 10 | 14035 | 1.52 | 0.26 | 0.47 | 0.49 | 0.30 | 1.61 | 112.53 |
| | 15 | 17067 | 1.25 | 0.24 | 0.37 | 0.35 | 0.29 | 1.29 | 111.67 |
| 5000 | 1 | 1843 | 22.61 | 2.92 | 8.76 | 8.00 | 2.93 | 23.60 | 218.94 |
| | 5 | 8139 | 5.12 | 0.77 | 1.81 | 1.80 | 0.74 | 5.45 | 221.58 |
| | 10 | 14318 | 2.91 | 0.50 | 0.97 | 0.93 | 0.51 | 3.11 | 214.54 |
| | 15 | 17960 | 2.32 | 0.45 | 0.72 | 0.68 | 0.47 | 2.40 | 225.08 |

Table 11. Performance in Times and Mflop/s of Cholesky Factorization on SUN UltraSPARC-IV computer with a different number of Processors, testing the SMP Parallelism. PPTRF does not show any SMP parallelism. UPLO = 'L'.

9. SUMMARY AND CONCLUSIONS

This paper describes RFPF as a standard minimal full format for representing both symmetric and triangular matrices. Hence, these matrix layouts are a replacement for both the standard formats of DLA, namely full and packed storage. These new layouts possess three good features: they are supported by Level 3 BLAS and LAPACK full format routines, and they require minimal storage.

10. ACKNOWLEDGMENTS

The results in this paper were obtained on five computers, an IBM, two SUN, Itanium and NEC computers. The IBM machine belongs to the Center for Scientific Computing at Aarhus, the SUN machines to the Danish Technical University, and the Itanium and NEC machines to the Danish Meteorological Institute. We would like to thank Bernd Dammann for consulting on the SUN systems; Niels Carl W. Hansen for consulting on the IBM system; and Bjarne Stig Andersen for obtaining the results on the Itanium and NEC computers. We thank IBMers John Gunnels who worked earlier on the RFPF format and JP Fasano who was instrumental in getting the source code released by the IBM Open Source Committee. We thank Julien Langou for helping to explain the LAPACK testing code for the LAPACK PO, PP routines as well as agreeing to do the final packaging of the delivered codes. We thank Allan Backer for discussions about a new version of this paper.

REFERENCES

- AGARWAL, R., GUSTAVSON, F. G., AND ZUBAIR, M. 1994. Exploiting functional parallelism on power2 to design high-performance numerical algorithms. *IBM Journal of Research and Development* 38, 5 (September), 563–576.
- ANDERSEN, B. S., GUNNELS, J. A., GUSTAVSON, F. G., REID, J. K., AND WAŚNIEWSKI, J. 2005. A fully portable high performance minimal storage hybrid format Cholesky algorithm. *ACM Transactions on Mathematical Software* 31, 201–227.
- ANDERSEN, B. S., GUNNELS, J. A., GUSTAVSON, F. G., AND WAŚNIEWSKI, J. 2002. A Recursive Formulation of the Inversion of symmetric positive definite Matrices in Packed Storage Data Format. In J. FAGERHOLM, J. HAATAJA, J. JÄRVINEN, M. LYLY, AND P. R. V. SAVOLAINEN Eds., *Proceedings of the 6th International Conference, PARA 2002, Applied Parallel Computing*, Number 2367 in Lecture Notes in Computer Science (Espoo, Finland, June 2002), pp. 287–296. Springer.
- ANDERSEN, B. S., GUSTAVSON, F. G., AND WAŚNIEWSKI, J. 2001. A Recursive Formulation of Cholesky Factorization of a Matrix in Packed Storage. *ACM Transactions on Mathematical Software* 27, 2 (Jun), 214–244.
- ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, L., DEMMEL, J., DONGARRA, J., DUCROZ, J., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. 1999. *LAPACK Users' Guide* (Third ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA.
- BARKER, V. A., BLACKFORD, L. S., DONGARRA, J. J., CROZ, J. D., HAMMARLING, S., MARIANOVA, M., WAŚNIEWSKI, J., AND YALAMOV, P. 2001. *LAPACK95 Users' Guide* (first ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA.
- BUTTARI, A., LANGOU, J., KURZAK, J., AND DONGARRA, J. 2007. A class of parallel tiled linear algebra algorithms for multi-core architectures. *Tech Rep. UT-CS-07-0600*.
- CHAN, E., QUINTANA-ORTI, E., QUINTANA-ORTI, G., AND VAN DE GEIJN, R. 2007. Supermatrix out-of-order scheduling of matrix operations for smp and multi-core architectures. *SPAA 07, Proceedings of the 19th ACM Symposium on Parallelism in Algorithms and Architecture*, 116–125.
- DEMMEL, J. W. 1997. *Applied Numerical Linear Algebra*. SIAM, Philadelphia.
- DONGARRA, J., DU CROZ, J., DUFF, I., AND HAMMARLING, S. 1990a. Algorithm 679: A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.* 16, 1 (March), 18–28.
- DONGARRA, J., DU CROZ, J., DUFF, I., AND HAMMARLING, S. 1990b. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.* 16, 1 (March), 1–17.
- DONGARRA, J., DUCROZ, J., HAMMARLING, S., AND HANSON, R. J. 1988. An Extended Set of Fortran Basic Linear Algebra Subroutines. *ACM Trans. Math. Soft.* 14, 1 (March), 1–17.
- DONGARRA, J. J., BUNCH, J., MOLER, C., AND STEWART, G. 1979. *Linpack Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
- DONGARRA, J. J., DUFF, I., SORENSEN, D., AND VAN DER VORST, H. 1998. *Numerical Linear Algebra for High Performance Computers*. SIAM, Society for Industrial and Applied Mathematics, Philadelphia.
- ELMROTH, E., GUSTAVSON, F. G., KAGSTROM, B., AND JONSSON, I. 2004. Recursive blocked algorithms and hybrid data structures for dense matrix library software. *SIAM Review* 46, 1 (March), 3–45.
- GOLUB, G. AND VAN LOAN, C. 1996. *Matrix Computations* (Third ed.). Johns Hopkins University Press, Baltimore, MD.
- GUNNELS, J. A. AND GUSTAVSON, F. G. 2004. A new array format for symmetric and triangular matrices. In J. W. JACK J. DONGARRA, K. MADSEN Ed., *Applied Parallel Computing, State of the Art in Scientific Computing, PARA 2004*, Volume LNCS 3732 (Springer-Verlag, Berlin Heidelberg, 2004), pp. 247–255. Springer.
- GUSTAVSON, F. G. 2003. High performance linear algebra algorithms using new generalized data structures for matrices. *IBM Journal of Research and Development* 47, 1 (January), 823–849.

- GUSTAVSON, F. G., GUNNELS, J. A., AND SEXTON, J. C. 2006. Minimal data copy for dense linear algebra factorization. In *Applied Parallel Computing, State of the Art in Scientific Computing, PARA 2006*, Volume LNCS 4699 (Springer-Verlag, Berlin Heidelberg, 2006), pp. 540–549. Springer.
- GUSTAVSON, F. G. AND JONSSON, I. 2000. Minimal storage high performance cholesky via blocking and recursion. *IBM Journal of Research and Development* 44, 6 (Nov), 823–849.
- GUSTAVSON, F. G., REID, J. K., AND WAŚNIEWSKI, J. 2007. Algorithm 865: Fortran 95 subroutines for cholesky factorization in blocked hybrid format. *ACM Transactions on Mathematical Software* 33, 1 (March), 5.
- GUSTAVSON, F. G. AND WAŚNIEWSKI, J. 2006. Rectangular full packed format for lapack algorithms timings on several computers. In *Applied Parallel Computing, State of the Art in Scientific Computing, PARA 2006*, Volume LNCS 4699 (Springer-Verlag, Berlin Heidelberg, 2006), pp. 570–579. Springer.
- GUSTAVSON, F. G., WAŚNIEWSKI, J., AND DONGARRA, J. J. 2007. Fortran 77/95 Subroutines for the Cholesky Algorithm in Rectangular Full Packed data format. For submission to the ACM Transactions on Mathematical Software.
- HERRERO, J. 2006. *A Framework for Efficient Execution of Matrix Computations*. Ph. D. thesis, Universitat Politècnica de Catalunya.
- HIGHAM, N. 1996. *Accuracy and Stability of Numerical Algorithms*. SIAM.
- IBM. 1997. *Engineering and Scientific Subroutine Library for AIX* (Version 3, Volume 1 ed.). IBM. Pub. number SA22-7272-0.
- LAWSON, C., HANSON, R., KINCAID, D., AND KROGH, F. 1979. Basic Linear Algebra Subprograms for Fortran Usage. *ACM Trans. Math. Soft.* 5, 308–323.
- TREFETHEN, L. N. AND BAU, D. 1997. *Numerical Linear Algebra*. SIAM, Philadelphia.