LAPACK Cholesky Routines in Rectangular Full Packed Format

Fred G. Gustavson[1] and Jerzy Waśniewski[2]

[1] T. J. Watson Research Center, Yorktown Heights, NY 10598, USA
fg2@us.ibm.com
[2] Institute of Mathematical Modeling, Technical University of Denmark, DK-2800 Lyngby, Denmark
jw@imm.dtu.dk

Abstract. We describe a new data format for storing triangular and symmetric matrices called RFP (Rectangular Full Packed). The standard two dimensional arrays of Fortran and C (also known as full format) that are used to store triangular and symmetric matrices waste half the storage space but provide high performance via the use of level 3 BLAS. 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 RFP format to obtain high performance using L3 (Level 3) BLAS as RFP is totally full format. Also, RFP format requires exactly the same minimal storage as packed storage. Each full and/or packed symmetric/triangular routine becomes a single new RFP routine. We present LAPACK routines for Cholesky factorization and inverse computation in RFP format to illustrate this new work and to describe its performance on the Intel, IBM, Itanium, SGI, and Sun platforms. Performance of RPF verses LAPACK full routines is about the same while using half the storage. Performance is roughly one to twenty times faster for LAPACK packed routines while using the same storage. In the performance study only existing LAPACK routines and level 3 BLAS were used. We describe codes to directly input RFP format arrays. These codes are similar to codes for inputting full format arrays.

1 Description of Rectangular Full Packed Format

We describe RFP format. It represents a standard packed array as a full 2D array. This means that performance of LAPACK's [2] packed format routines becomes equal to or better than their full array counterparts. RFP format is a variant of hybrid full packed (HFP) format [1]. RFP format is a rearrangement of a standard full rectangular array `SA` holding a symmetric / triangular matrix A into a compact full storage rectangular array `AR` that uses minimal storage `NT=N(N+1)/2`. Note also that the transpose of the matrix in array `AR` also represents A. Therefore, Level 3 BLAS can be used on `AR` or its transpose. In fact, with the equivalent LAPACK algorithm, using array `AR` on its transpose instead

of array SA, gives slightly better performance. 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.

2 Cholesky Factorization using Rectangular Full Packed Format.

RFP format is a standard full array of size NT=n(n+1)/2 that holds a symmetric / triangular matrix A of order n. It is closely related to HFP format, see [1], 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

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} \tag{1}$$

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}. 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 an $(k+1) \times k$ dense matrix. 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 AR. In effect, each block matrix A_{11}, A_{21} and A_{22} is now stored in "full format", This means all entries of AR can be accessed with constant row and column strides. So, the full power of LAPACK's block Level 3 BLAS are now available for symmetric and triangular computations. Additionally, one is using the minimal amount of storage. Finally, AR^T which is $k \times (n+1)$ has these same two desirable properties. In Figure 1 with $n = 10$ or $n = 9$ we have added vertical |'s to try to visually delineate triangles T1,T2 representing lower, upper triangles of A_{11} , A_{22}^T respectively and square or near square S1 representing matrix A_{21}. The elements of A are represented using 0 indexing. Now A has a block 2×2 form Cholesky factorization

$$LL^T = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix} \tag{2}$$

where L_{11} and L_{22} are lower triangular. Equation 2 is the basis of a *simple related partition algorithm* (SRPA) on RFP. We now illustrate this by using existing LAPACK routines and Level 3 BLAS. The SRPA with partition sizes k and k and $n = 2k$ is: (see equations 1, 2 and Figure 1).

1. factor $L_{11}L_{11}^T = A_{11}$
   ```
   call dpotrf('L',k,AR(1,0),n+1,info)
   ```
2. solve $L_{21}L_{11}^T = A_{21}$
   ```
   call dtrsm('R','L','T','N',k,k,
   one,AR(1,0),n+1,AR(k+1,0),n+1)
   ```

```
   LRFP AR          LRFP AR
00|55 65 75 85   55 65 75 85 95
10 11|66 76 86   00|66 76 86 96
20 21 22|77 87   10 11|77 87 97
30 31 32 33|88   20 21 22|88 98
40 41 42 43 44   30 31 32 33|99
50 51 52 53 54   40 41 42 43 44
60 61 62 63 64   50 51 52 53 54
70 71 72 73 74   60 61 62 63 64
80 81 82 83 84   70 71 72 73 74
                 80 81 82 83 84
                 90 91 92 93 94
```

Fig. 1. Lower Rectangular Full Packed formats when $n = 9, 10$, `LDAR = n,n+1`

3. update $A_{22}^T \leftarrow A_{22}^T - L_{21}L_{11}^T$
   `call dsyrk('U','N',k,k,-one,`
   `AR(k+1,0),n+1,one,AR(0,0),n+1)`
4. factor $U_{22}^T U_{22} = A_{22}^T$
   `call dpotrf('U',k,AR(0,0),n+1,info)`

This covers RFP format when `uplo` = 'L' and n is even. A similar result holds for n odd. Also, for `uplo` = 'U' and n even or odd similar results hold.

We now consider performance aspects of using RFP format in the context of using LAPACK routines on triangular matrices stored in RPF format. The above SRPA should perform about the same as the corresponding full format LAPACK routine. This is because both the SRPA code and the corresponding LAPACK code are nearly the same and both data formats are full format. Therefore, the SRPA code should outperform the corresponding LAPACK packed code by about the same margin as does the corresponding LAPACK full code. In [1] performance results for HFP format on the IBM Power 4 Processor is given. Those results are very similar to what we obtained for RFP format. The gain of full code over packed code is anywhere from roughly a factor of one to a factor of seven.

3 A Preliminary Performance Study using RFP Format

The final paper will gives performance results for the four LAPACK routines versus RFP format namely DPOTRF/DPPTRF(uplo='L' and uplo = 'U') verses RFP format for DPOTRF/DPPTRF and for DPOTRI/DPPTRI(uplo='L' and uplo = 'U') versus RFP format for DPOTRI/DPPTRI. Results will be run on several platforms using either vendor Level 3 BLAS or ATLAS. In all cases, only LAPACK code will be called so only standard LAPACK on different standard full and packed data formats is being compared versus various RFP formats. The preliminary results on Intel, IBM Power 4 and Sun show that RFP variants are about the same as LAPACK full routines and are roughly one to twenty times faster than LAPACK packed routines.

4 Summary and Conclusions

This extended abstract describes RFP format as a standard *minimal* full storage array for representing both symmetric and triangular matrices. Hence, for these types of matrices it is a replacement for both the standard formats of DLA, namely full and packed storage. It possesses three good features: it is supported by Level 3 BLAS and LAPACK full format routines and it requires minimal storage.

References

1. J. A. Gunnels, F. G. Gustavson. A New Array Format for Symmetric and Triangular Matrices. *Computational Science - Para 2004*, J. J. Dongarra, K. Madsen, J. Waśniewski, eds., Lecture Notes in Computer Science 3732. Springer-Verlag, pp. 247-255, 2004.
2. E. Anderson, et. al.. *LAPACK Users' Guide Release 3.0*, SIAM, Philadelphia, 1999, (http://www.netlib.org/ lapack/lug/lapack_lug.html).