Recursive Formulation for Some of the Dense Linear Algebra Algorithms

Jerzy Waśniewski
The Danish Computing Center for
Research and Education (UNI•C)
DTU, Bldg. 304
DK - 2800 Lyngby, Denmark
e-mail: jerzy.wasniewski@uni-c.dk
http://www.cs.utk.edu/~jerzy/wasniewski/

With collaborations:

- **Bjarne Stig Andersen**
  UNI•C
- **Fred Gustavson**
  IBM T.J. Watson Research Center, Yorktown Heights, NY
- **Plamen Yalamov**
  Computer Science, Rousse University, Bulgaria

Recursive Formulation

IBM/UNI-C Project

IBM to UNI-C

- **High Performance Switch**
- **Two SMP nodes**

UNI-C to IBM

- **Research on recursive dense linear algebra algorithms**
- **Three typical users' problems on SMP**

Recursive Formulation

Recursion

Fred Gustavson and S. Toledo (PhD student)

- F.G. Gustavson
- Recursive Leads to Automatic Variable Blocking for Dense Linear-Algebra Algorithms
- IBM Journal of Research and Development, Volume 41, Number 6, November 1997

- S. Toledo
- Locality of Reference in *LU* Decomposition with Partial Pivoting
- SIAM Journal on Matrix Analysis and Applications, Vol. 18, No. 4, 1997

Recursive Formulation

Recursion

We have already published:

- **B.S. Andersen, F. Gustavson and J. Waśniewski**
- **Recursive Formulation of Cholesky Algorithm in Fortran 90**
- **PARA'98 Proceedings, Umeå, Sweden, LNCS, Springer, 1998**

We have submitted to the SIAM'99 conference:

1. - **B.S. Andersen, J. Chen, F. Gustavson, J. Waśniewski and P.Y. Yalamov**
   - **Recursive Formulation of Dense Linear Algebra Algorithms**
2. - **B.S. Andersen, F. Gustavson and J. Waśniewski**
   - **A Recursive Formulation of the Cholesky Factorization Operating on a Matrix in Packed Storage Form**

Recursive Formulation

C H O L E S K Y Decomposition

$$AX = B$$

- A **is a real symmetric and positive definite matrix**
- X **and** B **are rectangular matrices or vectors**

- $A = U^T U$, if upper triangular part of A is given
- $A = L\,L^T$, if lower triangular part of A is given
- U **is an upper triangular matrix**
- L **is a lower triangular matrix**
- $L = U^T$ **and** $U = L^T$
- **The factored form of** A **is then used to** $A\,X = B$

Recursive Formulation

C H O L E S K Y Decomposition (contd)

Lower Triangular Case

The Cholesky decomposition algorithm can be written:

$$
\begin{array}{l}
\textbf{For } j = 1,\, 2,\, \ldots\, n\textbf{;} \\
\quad l_{jj} = (a_{jj} - \Sigma_{k=1}^{j-1}\, l_{jk}^2)^{1/2} \\
\quad \textbf{For } i = j+1,\, \ldots\, n\textbf{;} \\
\quad\quad l_{ij} = (a_{ij} - \Sigma_{k=1}^{j-1}\, l_{ik} l_{jk}) / l_{jj}
\end{array}
$$

- a_{ij} **and** l_{ij} **for** $i = 1,\, 2,\, \ldots\, n$ **and** $j = 1,\, 2,\, \ldots\, i$
- a_{ij} **and** l_{ij} **are elements of the matrices** A **and** L **respectively**
- $n^3/3$ **flop operations required**

Recursive Formulation

C H O L E S K Y Decomposition
Lower Triangular Case (contd)

A **and** L **can be partitioned to smaller submatrices:**

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \textsf{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

$$A = \left(\begin{array}{ccc|cccc} a_{11} & & & & & & \\ a_{21} & a_{22} & & & & & \\ a_{31} & a_{32} & a_{33} & & & & \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & & & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & & \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} \end{array}\right)$$

Recursive Formulation

C H O L E S K Y Decomposition
Lower Triangular Case (contd)

The above formulae can be modified so that the computations can be done on small matrices:

1. **For** A_{11}**,** $i = 1, 2, \ldots p$ **and** $j = 1, 2, \ldots i$**.**
2. **For** A_{21}**,** $i = p + 1, \ldots n$ **and** $i = 1, 2, \ldots p$**.**
3. **For** A_{22}**,** $i = p + 1, \ldots n$ **and** $j = p + 1, \ldots i$**.**

where $p = n/2$

$$A = \begin{pmatrix} A_{11} & \\ A_{21} & A_{22} \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix}$$

Recursive Formulation

C H O L E S K Y Decomposition
Lower Triangular Case (contd)

Computation on submatrix A_{11}:

For $j = 1,\ 2,\ \ldots\ p$**;**
$\quad l_{jj} = (a_{jj} - \Sigma_{k=1}^{j-1} l_{jk}^2)^{1/2}$
$\quad$**For** $i = j+1,\ \ldots\ p$**;**
$\quad\quad l_{ij} = (a_{ij} - \Sigma_{k=1}^{j-1} l_{ik} l_{jk})/l_{jj}$

- $p = n/2$

- **Cholesky of submatrix A_{11} of size $p \times p$**

Recursive Formulation

C H O L E S K Y Decomposition
Lower Triangular Case (contd)

Computation on submatrix A_{21}

$$\begin{array}{l} \textbf{For } j = 1, 2, \ldots p \\ \quad \textbf{For } i = p+1, \ldots n; \\ \qquad l_{ij} = (a_{ij} - \Sigma_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj} \end{array}$$

Multiplying by l_{jj} ($l_{jj} \neq 0$) and simplifying:

$$\sum_{k=1}^{j} l_{ik} l_{jk} = a_{ij}$$

for $i = p+1, \ldots n$ **and** $j = 1, 2, \ldots p$

A linear triangular system of equations with multiple RHS:

$$L_{21} L_{11}^T = A_{21},$$

- L_{21} **unknown**
- L_{11}^T **coefficient matrix and**
- A_{21} **RHS matrix**
- **BLAS subroutine TRSM**

Recursive Formulation

C H O L E S K Y Decomposition
Lower Triangular Case (contd)

Computation on submatrix A_{22}

$$
\begin{array}{l}
\textbf{for } j = p+1, \ldots n\textbf{:} \\
\quad l_{jj} = (a_{jj} - \Sigma_{k=1}^{j-1} l_{jk}^2)^{1/2} \\
\quad \textbf{for } i = j+1, \ldots n\textbf{:} \\
\quad\quad l_{ij} = (a_{ij} - \Sigma_{k=1}^{j-1} l_{ik} l_{jk})/l_{jj}
\end{array}
$$

By simple modification:

$$
\begin{array}{l}
\textbf{for } j = p+1, \ldots n\textbf{:} \\
\quad l_{jj} = (\hat{a}_{jj} - \Sigma_{k=p+1}^{j-1} l_{jk}^2)^{1/2} \\
\quad \textbf{for } i = j+1, \ldots n\textbf{:} \\
\quad\quad l_{ij} = (\hat{a}_{ij} - \Sigma_{k=p+1}^{j-1} l_{ik} l_{jk})/l_{jj}
\end{array}
$$

Recursive Formulation

C H O L E S K Y Decomposition
Lower Triangular Case (contd)

$$\hat{a}_{jj} = a_{jj} - \sum_{k=1}^{p} l_{jk}^2$$

$$\hat{a}_{ij} = a_{ij} - \sum_{k=1}^{p} l_{ik} l_{jk}$$

and:

$$\hat{A}_{22} := A_{22} - L_{21} L_{21}^T$$

can be calculated by:

BLAS subroutine SYRK

- **Cholesky of submatrix** $\hat{A}_{22}$ **of size** $(n-p) \times (n-p)$

Recursive Formulation

C H O L E S K Y Decomposition
Lower Triangular Case (contd)

Cholesky recursive algorithm
if lower triangular part of A is given:

- **if** $n > 1$ **then**
  - L_{11} := **Cholesky of** A_{11}
  - $L_{21}L_{11}^T = A_{21}$ **(TRSM)**
  - $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T$ **(SYRK)**
  - L_{22} := **Cholesky of** $\hat{A}_{22}$
- **otherwise** • $L := \sqrt{A}$

Recursive Formulation

C H O L E S K Y Decomposition (contd)
Upper Triangular Case

The matrices A and U are in this case:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ & A_{22} \end{pmatrix} \quad \textsf{and} \quad L = \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix}$$

$$A = \left(\begin{array}{ccc|cccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ & & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \\ \hline & & & a_{44} & a_{45} & a_{46} & a_{47} \\ & & & & a_{55} & a_{56} & a_{57} \\ & & & & & a_{66} & a_{67} \\ & & & & & & a_{77} \end{array}\right)$$

Recursive Formulation

C H O L E S K Y Decomposition
Upper Triangular Case (contd)

Cholesky Decomposition of matrix A.

For $j = 1, 2, \ldots n$**;**
$\quad u_{jj} = (a_{jj} - \Sigma_{k=1}^{j-1} u_{jk}^2)^{1/2}$
$\quad$**For** $i = j+1, \ldots n$**;**
$\quad\quad u_{ij} = (a_{ij} - \Sigma_{k=1}^{j-1} u_{ik} u_{jk}) / u_{jj}$

Cholesky recursive algorithm
if upper triangular part of A is given:

- **if** $n > 1$ **then**
  - $U_{11} :=$ **Cholesky of** A_{11}
  - $U_{11}^T U_{12} = A_{12}$ **(TRSM)**
  - $\hat{A}_{22} := A_{22} - U_{12}^T U_{12}$ **(SYRK)**
  - $U_{22} :=$ **Cholesky of** $\hat{A}_{22}$
- **otherwise** • $U := \sqrt{A}$

Recursive Formulation

C H O L E S K Y Decomposition (contd)
Subroutine of Cholesky Recursive Algorithm

```
RECURSIVE SUBROUTINE RPOTRF( A, UPLO, INFO )
   USE LA_PRECISION, ONLY: WP => DP
   USE LA_AUXMOD, ONLY: ERINFO, LSAME
   USE F90_RCF, ONLY: RCF => RPOTRF, RTRSM, RSYRK
   IMPLICIT NONE
   CHARACTER(LEN=1), OPTIONAL, INTENT(IN) :: UPLO
   INTEGER, OPTIONAL, INTENT(OUT) :: INFO
   REAL(WP), INTENT(INOUT) :: A(:,:)

   CHARACTER(LEN=*), PARAMETER :: SRNAME = 'RPOTRF'
   REAL(WP), PARAMETER :: ONE = 1.0_WP
   CHARACTER(LEN=1) :: LUPLO
   INTEGER :: N, P, LINFO

   N = SIZE(A,1); LINFO = 0
   IF( PRESENT(UPLO) )THEN; LUPLO = UPLO; ELSE; &
                             LUPLO = 'U'; ENDIF

   IF( N < 0 .OR. N /= SIZE(A,2) )THEN; LINFO = -1
   ELSE IF( .NOT. (LSAME(LUPLO,'U').OR.LSAME(LUPLO,'L')) ) &
                                          THEN; LINFO = -2
```

Recursive Formulation

C H O L E S K Y Decomposition

Subroutine of Cholesky Recursive Algorithm (contd)

```
   ELSE IF (N == 1) THEN; A(1,1) = SQRT(A(1,1))
   ELSE IF( N > 0 )THEN; P=N/2
     IF( LSAME(LUPLO,'L') )THEN
       CALL RCF( A(1:P,1:P), LUPLO, LINFO )
       CALL RTRSM( A(1:P,1:P), A(P+1:N,1:P), UPLO=LUPLO, &
                               SIDE='R', TRANSA='T' )
       CALL RSYRK( A(P+1:N,1:P), A(P+1:N,P+1:N), ALPHA=-ONE, &
                                            UPLOC=LUPLO )
       CALL RCF( A(P+1:N,P+1:N), LUPLO, LINFO )
     ELSE
       CALL RCF( A(1:P,1:P), LUPLO, LINFO )
       CALL RTRSM( A(1:P,1:P), A(1:P,P+1:N), TRANSA='T' )
       CALL RSYRK( A(1:P,P+1:N), A(P+1:N,P+1:N), ALPHA=-ONE, &
                                             TRANSA='T' )
       CALL RCF( A(P+1:N,P+1:N), LUPLO, LINFO )
     ENDIF
   ENDIF

   CALL ERINFO( LINFO, SRNAME, INFO )
END SUBROUTINE RPOTRF
```

Recursive Formulation

C H O L E S K Y Decomposition (contd)
TRSM and SYRK BLA Subroutines

TRSM solves one of the matrix equations:

- $op(A) \times X = \alpha \times B$, **or** $X \times op(A) = \alpha \times B$
- α **is a scalar**
- X **and** B **are rectangular matrices**
- A **is a unit, or non-unit, upper or lower triangular matrix**
- $op(A)$ **is one of** $op(A) = A$ **or** $op(A) = A^T$
- **The matrix** X **is overwritten on** B

SYRK performs one of the symmetric rank k operations:

- $C := \alpha \times A \times A^T + \beta \times C$, **or**
- $C := \alpha \times A^T \times A + \beta \times C$,
- α **and** β **are scalars**
- C **is a square symmetric matrix**
- A **is a rectangular matrix**

Recursive Formulation

C H O L E S K Y Decomposition (contd)
GEMM BLA Subroutines

GEMM does the following operation:

- $C := \alpha \times op(A) \times op(B) + \beta \times C$
- α **and** β **are scalars**
- A, B **and** C **are rectangular matrices**
- $op(X)$ **is one of** $op(X) = X$, $op(X) = X^T$ **or** $op(X) = X^T$
- **where** X **is** A **or** B

Recursive Formulation (contd)

Recursive BLA Subroutines
RTRSM BLA Subroutines

A **is an upper triangular matrix**

- **UPLO = 'U', TRANS = 'N', SIDE = 'L'** $\Rightarrow \quad A\,X = \alpha B$

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \alpha \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \Rightarrow \begin{array}{l} A_{11}X_1 = \alpha B_1 - A_{12}X_2 \\ A_{22}X_2 = \alpha B_2 \end{array}$$

- **UPLO = 'U', TRANS = 'N', SIDE = 'R'** $\Rightarrow \quad X\,A = \alpha B$

$$\begin{pmatrix} X_1 & X_2 \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} = \alpha \begin{pmatrix} B_1 & B_2 \end{pmatrix} \Rightarrow \begin{array}{l} X_1A_{11} = \alpha B_1 \\ X_2A_{22} = \alpha B_2 - X_1A_{12} \end{array}$$

- **UPLO = 'U', TRANS = 'T', SIDE = 'L'** $\Rightarrow \quad A^T\,X = \alpha B$

$$\begin{pmatrix} A_{11}^T & 0 \\ A_{12}^T & A_{22}^T \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \alpha \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \Rightarrow \begin{array}{l} A_{11}^TX_1 = \alpha B_1 \\ A_{22}^TX_2 = \alpha B_2 - A_{12}^TX_1 \end{array}$$

- **UPLO = 'U', TRANS = 'T', SIDE = 'R'** $\Rightarrow \quad X\,A^T = \alpha B$

$$\begin{pmatrix} X_1 & X_2 \end{pmatrix} \begin{pmatrix} A_{11}^T & 0 \\ A_{12}^T & A_{22} \end{pmatrix} = \alpha \begin{pmatrix} B_1 & B_2 \end{pmatrix} \Rightarrow \begin{array}{l} X_1A_{11}^T = \alpha B_1 - X_2A_{12}^T \\ X_2A_{22}^T = \alpha B_2 \end{array}$$

Recursive Formulation

Recursive BLA Subroutines
RTRSM BLA Subroutines (contd)

A **is a lower triangular matrix**

- **UPLO = 'L', TRANS = 'N', SIDE = 'L'** $\Rightarrow \quad A\,X = \alpha B$

$$\begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \alpha \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \Rightarrow \begin{array}{l} A_{11}X_1 = \alpha B_1 \\ A_{22}X_2 = \alpha B_2 - A_{21}X_1 \end{array}$$

- **UPLO = 'L', TRANS = 'N', SIDE = 'R'** $\Rightarrow \quad X\,A = \alpha B$

$$\begin{pmatrix} X_1 & X_2 \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix} = \alpha \begin{pmatrix} B_1 & B_2 \end{pmatrix} \Rightarrow \begin{array}{l} X_1A_{11} = \alpha B_1 - X_2A_{21} \\ X_2A_{22} = \alpha B_2 \end{array}$$

- **UPLO = 'L', TRANS = 'T', SIDE = 'L'** $\Rightarrow \quad A^T\,X = \alpha B$

$$\begin{pmatrix} A_{11}^T & A_{21}^T \\ 0 & A_{22}^T \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \alpha \begin{pmatrix} B_1 \\ B_2 \end{pmatrix} \Rightarrow \begin{array}{l} A_{11}^TX_1 = \alpha B_1 - A_{21}^TX_1 \\ A_{22}^TX_2 = \alpha B_2 \end{array}$$

- **UPLO = 'L', TRANS = 'T', SIDE = 'R'** $\Rightarrow \quad X\,A^T = \alpha B$

$$\begin{pmatrix} X_1 & X_2 \end{pmatrix} \begin{pmatrix} A_{11}^T & A_{21}^T \\ 0 & A_{22} \end{pmatrix} = \alpha \begin{pmatrix} B_1 & B_2 \end{pmatrix} \Rightarrow \begin{array}{l} X_1A_{11}^T = \alpha B_1 \\ X_2A_{22}^T = \alpha B_2 - X_1A_{21}^T \end{array}$$

Recursive Formulation

C H O L E S K Y Decomposition (contd)

Packed Storage

The matrix A is stored in one dimensional array AP:

$A_{i,j}$	i, j	**UPLO**
$\mathbf{AP}_{i+(j-1)j/2}$	$1 \leq j \leq n$ $1 \leq i \leq j$	$'\mathsf{U}'$
$\mathbf{AP}_{i+(j-1)(2n-j)/2}$	$1 \leq j \leq n$ $j \leq i \leq n$	$'\mathsf{L}'$

Recursive Formulation
C H O L E S K Y Decomposition
Packed Storage (contd)

$n = 4$, **UPLO = 'U':**

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

$a_{ij} = \mathsf{conjg}(a_{ji})$ for $i, j = 1, \cdots, 4.$

AP on entry

a_{11}	a_{12}	a_{22}	a_{13}	a_{23}	a_{33}	a_{14}	a_{24}	a_{34}	a_{44}

AP on exit

u_{11}	u_{12}	u_{22}	u_{13}	u_{23}	u_{33}	u_{14}	u_{24}	u_{34}	u_{44}

$size(\mathbf{AP}) = n(n+1)/2 = 10$
If $n = 1000$ **then** $size(\mathbf{AP}) = n(n+1)/2 = 500500$**, saving** 499500

Recursive Formulation

C H O L E S K Y Decomposition
Packed Storage (contd)

Matrix size	LAPACK L	LAPACK U	ESSL L	ESSL U	Recursive L	Recursive U
500		.16		.18		.15
1000		1.3		1.5		1.3
1500		4		4.5		3.9
2000		11		11		10
2500		19		21		17
3000		39		36		34

Run time in seconds of the Cholesky factorization algorithm on the IBM SMP four processor node.

Matrix size	LAPACK L	LAPACK U	ESSL L	ESSL U	Recursive L	Recursive U
500	1	.9	.24		.2	.2
1000	11	7	1.8		1.6	2
1500	50	35	6		5	5
2000	121	80	15		12	12
2500	236	153	29		23	22
3000	411	269	50		39	38

Run time in seconds of the packed Cholesky factorization algorithm on the IBM SMP four processor node.

Recursive Formulation

C H O L E S K Y Decomposition
Packed Storage (contd)

Matrix size	LAPACK		SSL		Recursive	
	L	U	L	U	L	U
500	xx	xx	xx	xx	xx	xx
1000	xx	xx	xx	xx	xx	xx
1500	xx	xx	xx	xx	xx	xx
2000	xx	xx	xx	xx	xx	xx
2500	xx	xx	xx	xx	xx	xx
3000	xx	xx	xx	xx	xx	xx

Run time in seconds of the Cholesky factorization algorithm on the SGI Origin 2000.

Matrix size	LAPACK		SSL		Recursive	
	L	U	L	U	L	U
500	.4	.4	.4	.5	.2	.2
1000	4	2	4	2	1	1
1500	15	15	15	15	4	4
2000	45	38	46	39	10	9
2500	107	85	104	85	21	17

Run time in seconds of the packed Cholesky factorization algorithm on the SGI Origin 2000.

Recursive Formulation
C H O L E S K Y Decomposition
Packed Storage (contd)

Two things are important

- **BLA Subprograms**
- **Data storage**

Why our algorithm is faster?

- **LAPACK uses BLAS Level2,**
  - **we use BLAS Level 3**
- **We use different data storage**

Recursive Formulation
C H O L E S K Y Decomposition
Packed Storage (contd)

RPPTRF – Cholesky factorization in packed storage

- Data reordering
- Recursive factorization subroutine
  - Testing
  - Recursive factorization subroutine
  - Recursive triangular solver subroutine
    - Testing
    - Recursive triangular solver subroutine
    - Fast GEMM
    - Recursive triangular solver subroutine
  - Recursive symmetric rank k subroutine
    - Testing
    - Recursive symmetric rank k subroutine
    - Fast GEMM
    - Recursive symmetric rank k subroutine
  - Recursive factorization subroutine
- Data reordering (back to FORTRAN data storage)

Recursive Formulation

Data Storage

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \\ A_{51} & A_{52} & A_{53} & A_{54} \end{bmatrix}$$

$$\begin{array}{c} FORTRAN \\ \begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 7 & 12 & 17 \\ 3 & 8 & 13 & 18 \\ 4 & 9 & 14 & 19 \\ 5 & 10 & 15 & 20 \end{bmatrix} \end{array} \qquad \begin{array}{c} C(Algol) \\ \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \\ 17 & 18 & 19 & 20 \end{bmatrix} \end{array}$$

$$\begin{array}{c} RBR \\ \left[\begin{array}{cc|cc} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \\ \hline\hline 9 & 10 & 15 & 16 \\ \hline 11 & 12 & 17 & 18 \\ 13 & 14 & 19 & 20 \end{array}\right] \end{array} \qquad \begin{array}{c} RBC \\ \left[\begin{array}{cc|cc} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \\ \hline\hline 9 & 10 & 15 & 16 \\ \hline 11 & 13 & 17 & 19 \\ 12 & 14 & 18 & 20 \end{array}\right] \end{array}$$

Recursive Formulation

Data Storage (contd)

$$A = \left[\begin{array}{cccc} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \\ A_{51} & A_{52} & A_{53} & A_{54} \end{array}\right]$$

$$\begin{array}{c} FORTRAN \\ (lda = 7) \\ \left[\begin{array}{cccc} 1 & 8 & 15 & 22 \\ 2 & 9 & 16 & 23 \\ 3 & 10 & 17 & 24 \\ 4 & 11 & 18 & 25 \\ 5 & 12 & 19 & 26 \\ \hline 6 & 13 & 20 & 27 \\ 7 & 14 & 21 & 28 \end{array}\right] \end{array} \quad \begin{array}{c} RBR \\ \left[\begin{array}{cc|cc} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \\ \hline\hline 9 & 10 & 15 & 16 \\ \hline 11 & 12 & 17 & 18 \\ 13 & 14 & 19 & 20 \end{array}\right] \end{array} \quad \begin{array}{c} RBC \\ \left[\begin{array}{cc|cc} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \\ \hline\hline 9 & 10 & 15 & 16 \\ \hline 11 & 13 & 17 & 19 \\ 12 & 14 & 18 & 20 \end{array}\right] \end{array}$$

Recursive Formulation

Data Storage (contd)

Summary

- **A matrix A is of size $M \times N$**
- **Block sizes of $A_{i,j}$ are $mb \times nb$,
  where $1 \le mb \le M$ and $1 \le nb \le N$**
- **A block-partitioned A consists of $NB = m \times n$
  blocks $A_{i,j}$ of size $mb \times nb$,
  where $m = \lceil M/mb \rceil$ and $n = \lceil N/nb \rceil$**
- **RBR recursive block row,
  always divides the row dimension**
- **RBC recursive block column,
  always divides the column dimension**

Recursive Formulation

Gauss elimination with partial pivoting

$$\mathbf{PA = LU}$$

For $j = 1,\ 2,\ \ldots\ n-1$**;**
  Determine μ **with** $k \le \mu \le n$ **so that**
    $|A(\mu, k)| = \|A(k:n, k\|_\infty$
  $A(k, k:n) \leftrightarrow A(\mu, k:n)$
  $p(k) = \mu$
  If $A(k, k) \neq 0$
    For $rows = k+1,\ k+2,\ \ldots\ n$**;**
      $A(rows, k) = A(rows, k)/A(k, k)$
      $A(rows, rows) = A(rows, rows) -$
        $A(rows, k) \times A(k, rows)$
    end of For of rows
  end of If
end of For of j

Recursive Formulation

Gauss elimination with partial pivoting (contd)

Main loop of LAPACK routine _GETRF.

```
do J=1,MIN(M,N),NB
  JB=MIN(MIN(M,N)-J+1,NB)
  call _GETF2(M-J+1,JB,A(J,J),LDA,IPIV(J),INFO)
  do I=J,MIN(M,J+JB-1)
    IPIV(I)=J-1+IPIV(I)
  end do
  CALL _LASWP(J-1,A,LDA,J,J+JB-1,IPIV,1)
  IF(J+JB.LE.N)THEN
    CALL _LASWP(N-J-JB+1,A(1,J+JB),LDA,J,
$        J+JB-1,IPIV,1)
    CALL _TRSM('L','L','N','U',JB,N-J-JB+1,ONE,
$        A(J,J),LDA,A(J,J+JB),LDA)
    IF(J+JB.LE.M)THEN
       CALL _GEMM('N','N',M-J-JB+1,N-J-JB+1,JB,
$          -ONE,A(J+JB,J),LDA,A(J,J+JB),
$          LDA,ONE,A(J+JB,J+JB),LDA)
    end if
  end if
end do
```

Recursive Formulation

Gauss elimination with partial pivoting (contd)

Recursive LU algorithm

If $(N = 1)$ **then**
  compute $PA = LU$
    i.e find pivot, interchange it and scale
else
  $P := n/2$
  LU **factor**
    $P_1 A(1 : M, 1 : P) = LU(1 : M, 1 : P)$ **(recursive call)**
  forward pivot $A(1 : M, P + 1 : N) =$
    $P_1 A(1 : M, P + 1 : M)$ **(_LASWP)**
  solve $L(1 : P, 1 : P)\, X = A(1 : P, P + 1 : N)$ **(_TRSM)**
  update $A(P + 1 : M, P + 1 : N) = A(P + 1 : M, P + 1 : N) -$
    $L(P + 1 : M, 1 : P)\, U(1 : P, P + 1 : N)$ **(_GEMM)**
  LU **factor** $P_2 A(P + 1 : M, P + 1 : P) =$
    $LU(P + 1 : M, P + 1 : N)$ **(recursive call)**
  back pivot $A(P + 1 : M, 1 : P) =$
    $P_2 A(P + 1 : M, 1 : P)$ **(_LASWP)**
end if

Recursive Formulation

Gauss elimination with partial pivoting (contd)
Subroutine of Recursive LU Algorithm

```
RECURSIVE SUBROUTINE RLUF(M,N,A,IPIV)
!  BSA
   IMPLICIT NONE
   INTEGER, INTENT(IN) :: M,N
   REAL, DIMENSION(1:M,1:N), INTENT(INOUT) :: A
   INTEGER, DIMENSION(1:M), INTENT(OUT) :: IPIV
   INTEGER :: P,IPIV,ISAMAX,I
   REAL :: STMP
   INTEGER, DIMENSION(1:M) :: IPIV2
   DO I=1,M
      IPIV(I)=I
      IPIV2(I)=0
   ENDDO
   IF (N .EQ. 1) THEN
      IPIV=I_AMAX(M,A(1:M,1:1),1)
      IPIV(1)=IPIV(IPIV)
      STMP=A(1,1)
      A(1,1)=A(IPIV,1)
      A(IPIV,1)=STMP
      IF(M.GT.1) THEN
         CALL _SCAL(M-1,1.0/A(1,1),A(2,1),1)
      ENDIF
```

Recursive Formulation

Gauss elimination with partial pivoting
Subroutine of Recursive LU Algorithm (contd)

```
   ELSE
      P=N/2
      CALL RLUF(M,P,A(1:M,1:P),IPIV(1:M))
      CALL _LASWP(N-P,A(1:M,P+1:N),M,1,M,IPIV(1:M),1)
      CALL _TRSM('L','L','N','U',P,N-P,1.0,A(1:P,1:P), &
                 P,A(1:P,P+1:N),P)
      CALL _GEMM('N','N',M-P,N-P,P,-1.E0,A(P+1:M,1:P), &
           M-P,A(1:P,P+1:N),P,1.E0,A(P+1:M,P+1:N),M-P)
      CALL RLUF(M-P,N-P,A(P+1:M,P+1:N),IPIV2(P+1:M))
      CALL _LASWP(P,A(P+1:M,1:P),M-P,1,M-P,IPIV2(P+1:M),1)
      DO I=1,M-P
         IF (IPIV2(P+I).NE.I) IPIV(P+I)=IPIV(P+IPIV2(P+I))
      ENDDO
   ENDIF
END
```

Recursive Formulation

Symmetric indefinite Matrix
Bunch–Kaufman Algorithm

$$\overset{A}{\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ & a_{22} & a_{23} & a_{24} & a_{25} \\ & & a_{33} & a_{34} & a_{35} \\ & & & a_{44} & a_{45} \\ & & & & a_{55} \end{pmatrix}} \quad \text{or} \quad \overset{A}{\begin{pmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ a_{41} & a_{42} & a_{43} & a_{44} & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix}}$$

A diagonal pivoting method is used to factor A as

$$A = P^T\, U\, D\, U^T\, P, \quad \text{if } \mathbf{UPLO} = \text{'U'}$$

or

$$A = P^T\, L\, D\, L^T\, P, \quad \text{if } \mathbf{UPLO} = \text{'L'}$$

- **P is a permutation matrix**
- **U and L are unit upper and lower triangular matrices, respectively**
- **D is a symmetric block diagonal with 1×1 and 2×2 diagonal blocks.**

Recursive Formulation

Symmetric indefinite Matrix
Bunch–Kaufman Algorithm (contd)

The algorithm for LDL^T decomposition is (LAPACK _SYTRF):

$L = I$ **(identity matrix);** $k = 1$;
while $(k < n)$
  Apply pivoting: choose a 1×1 $(s = 0)$, **or**
  2×2 $(s = 1)$ **pivot block,**
  and exchange the corresponding rows and columns;
  $E = A(k : k + s, k : k + s)$;
  $C = A(k + s + 1 : n, k : k + s)$;
  $B = A(k + s + 1 : n, k + s + 1 : n)$;
  $L(k + s + 1 : n, k : k + s) = CE^{-1}$;
  $D(k : k + s, k : k + s) = E$;
  $A(k + s + 1 : n, k + s + 1 : n) = B - CE^{-1}C^T$;
  k = k+s+1;
end

The pivoting strategy is well described in the book:

- **Nicholas J. Higham**
- **Accuracy and Stability of Numerical Algorithms**
- **SIAM 1996**
- **pages 218–223**

Recursive Formulation

Symmetric indefinite Matrix (contd)
Recursive Algorithms

The work is not finished yet
We have several versions

$$\begin{pmatrix} A_{11} & A_{12} \\ & A_{22} \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} A_{11} & \\ A21 & A22 \end{pmatrix}$$

$$\begin{array}{c} A \\ \left(\begin{array}{cc|ccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ & a_{22} & a_{23} & a_{24} & a_{25} \\ \hline & & a_{33} & a_{34} & a_{35} \\ & & & a_{44} & a_{45} \\ & & & & a_{55} \end{array}\right) \end{array} \quad \text{or} \quad \begin{array}{c} A \\ \left(\begin{array}{cc|ccc} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ \hline a_{31} & a_{32} & a_{33} & & \\ a_{41} & a_{42} & a_{43} & a_{44} & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{array}\right) \end{array}$$

Problem is very difficult, several people are involved
Plamen Yalamov, Fred Gustavson, John Reid and
Jerzy Waśniewski

Recursive Formulation

Symmetric indefinite Matrix (contd)
Recursive Algorithms

Other way of doing it:

$$\begin{pmatrix} A_{11} & A_{12} \\ \mathsf{A}_{12} & A_{22} \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} A_{11} & \mathsf{A}_{21} \\ A21 & A22 \end{pmatrix}$$

$$\overset{A}{\left(\begin{array}{cc|ccc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ & a_{22} & a_{23} & a_{24} & a_{25} \\ & & a_{33} & a_{34} & a_{35} \\ & & & a_{44} & a_{45} \\ & & & & a_{55} \end{array}\right)} \quad \text{or} \quad \overset{A}{\left(\begin{array}{cc|ccc} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ a_{41} & a_{42} & a_{43} & a_{44} & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{array}\right)}$$

Problem is very difficult, several people are involved Plamen Yalamov, Fred Gustavson, John Reid and Jerzy Waśniewski

Recursive Formulation

Symmetric indefinite Matrix (contd)
Recursive Algorithms

We need new BLAS

$$C = \alpha C + \beta A\, D\, A^T$$

- **α and β are constants**
- **C symmetric**
- **A general rectangular**
- **D block diagonal**

Recursive Formulation

Serial and Parallel Recursive QR Factorization

Erik Elmroth and Fred Gustavson
Umeå University and IBM Watson Research Center

_GEQRF, RGEQRF and PRGETRF

Recursive Formulation

Conclusion and Further Development