

November 27, 2003

New Data Storage Formats for Dense Matrices Lead to Variety of High-Performance Algorithms

The slides can be obtained from:

**<http://www.imm.dtu.dk/~jw/lectures/031202b.ps> or
<http://www.imm.dtu.dk/~jw/lectures/031202b.pdf>**

Jerzy Waśniewski

Emeritus Senior Research Professor

Department of Informatics & Mathematical Modeling

Technical University of Denmark

DTU, Bldg. 305, DK - 2800 Lyngby, Denmark

e-mail: jw@imm.dtu.dk

<http://www.imm.dtu.dk/~jw/JerzyWasniewski/>

November 27, 2003

Outline of my Talk:

1. Introduction

2. First Step

- Cholesky and Gauss

3. Symm. Pos. Def. Matrices

(a) LAPACK storages

- Full storage
- Packed storage
- Performance results

(b) New data formats

- Recursive storage
 - Algorithm

- Recursive BLAS

- Performance results

- Hybrid storage

- Performance results

4. Symm. Indef. Matrices

(a) Perturbation Approach

(b) Packed Block Storage

- Data format
- Performance results

5. More results

November 27, 2003

Introduction

Starting references:

Fred 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.

Sivan Toledo: “Locality of Reference in LU Decomposition with Partial Pivoting”, SIAM Journal on Matrix Analysis and Applications, Vol. 18, No. 4, 1997.

Recursion: Examples

- **Gamma function**

$$\Gamma(z + 1) = z\Gamma(z) = z! = z(z - 1)!$$

$z! = \Gamma(z + 1)$ – **factorial if z is integer**

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \text{ – Euler's Integral}$$

- **Bessel functions**

$$\mathfrak{C}_{\nu-1}(z) + \mathfrak{C}_{\nu+1}(z) = \frac{2\nu}{z} \mathfrak{C}_\nu$$

$$\mathfrak{C}_{\nu-1}(z) - \mathfrak{C}_{\nu+1}(z) = 2\mathfrak{C}'_\nu$$

\mathfrak{C} denotes $J, Y, H^{(1)},$ or $H^{(2)}$ **Bessel functions**

Cholesky and Gauss Factorizations

$$LL^T = U^T U \text{ and } LU$$

Full storage data format

First step of our work

$LL^T = U^T U$ Cholesky Decomposition

$$AX = B$$

- A – a symmetric or Hermitian, positive definite
- X and B – 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 – an upper triangular matrix
- ★ L – a lower triangular matrix
- ★ $L = U^T$ and $U = L^T$
- ★ The factored form of A is then used to $AX = B$

LU Gauss Decomposition

$$AX = B$$

- ***A* is a general real or complex matrix**
- ***X* and *B* are rectangular matrices or vectors**
- ★ **$A = P^T L U$**
- ★ ***P* is a permutation matrix**
- ★ ***U* is an upper triangular matrix**
- ★ ***L* is a lower triangular matrix**
- ★ **The factored form of *A* is then used to solve**
 $A X = B$

$LL^T = U^T U$ **Cholesky**
Factorization
Full storage data format
The first step of our work

$LL^T = U^T U$ Cholesky Decomposition

$$AX = B$$

- A – a symmetric or Hermitian, positive definite
- X and B – 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 – an upper triangular matrix
- ★ L – a lower triangular matrix
- ★ $L = U^T$ and $U = L^T$
- ★ The factored form of A is then used to $AX = B$

Cholesky: $A = LL^T$

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

$$A = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \times \begin{pmatrix} L_{11}^T & L_{21}^T \\ & L_{22}^T \end{pmatrix}$$

$$A = \begin{pmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11}L_{11}^T & L_{11}L_{21}^T \\ L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{pmatrix}$$

$$A_{11} = L_{11}L_{11}^T, \quad L_{21}L_{11}^T = A_{21} \quad \text{and} \quad \hat{A}_{22} = L_{22}L_{22}^T$$

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

Cholesky: $A = LL^T$

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

Do recursion

• **if $n > 1$ then**

- $L_{11} :=$ **rcholesky** of A_{11}
- $L_{21}L_{11}^T = A_{21} \rightarrow$ **RTRSM**
- $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T \rightarrow$ **RSYRK**
- $L_{22} :=$ **rcholesky** of \hat{A}_{22}

• **otherwise**

- $L := \sqrt{A_{11}}$

End recursion

Cholesky: $A = U^T U$

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

Do recursion

• **if $n > 1$ then**

- $U_{11} :=$ **rcholesky** of A_{11}
- $U_{11}^T U_{12} = A_{12} \rightarrow$ **RTRSM**
- $\hat{A}_{22} := A_{22} - U_{12}^T U_{12} \rightarrow$ **RSYRK**
- $U_{22} :=$ **rcholesky** of \hat{A}_{22}

• **otherwise**

- $U := \sqrt{A_{11}}$

End recursion

Cholesky

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

Cholesky

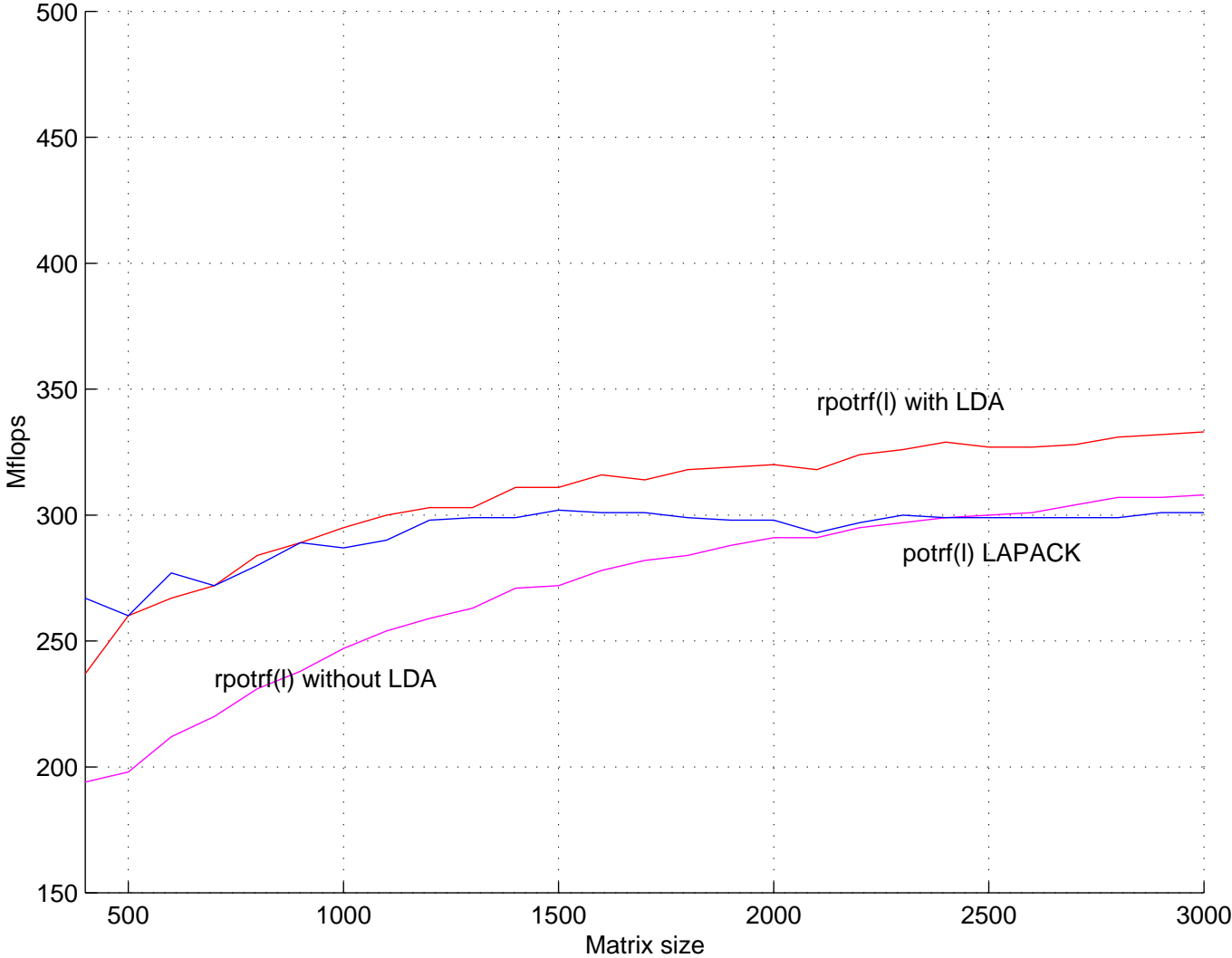
```
RECURSIVE SUBROUTINE RPOTRF( A, UPLO, INFO)
  USE LA_PRECISION, ONLY: WP => DP
  USE LA_AUXMOD, ONLY: ERINFO, LSAME
  USE F90_RCF, ONLY: RCF => RPOTRF, &
    RTRSM, RSYRK
  IMPLICIT NONE
  CHARACTER(LEN=1), OPTIONAL, INTENT(IN) :: UPLO
  INTEGER, OPTIONAL, INTENT(OUT) :: INFO
  REAL(WP), INTENT(INOUT) :: A(:, :)
  ... .. Locals declaration ... ..
  ... .. Testing arguments ... ..
  ... .. Recursive block ... ..
  CALL ERINFO(LINFO, SRNAME, INFO)
END SUBROUTINE RPOTRF
```

Cholesky

```
RECURSIVE SUBROUTINE RPOTRF( A, LDA, 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
  INTEGER, INTENT(IN) :: LDA
  REAL(WP), INTENT(INOUT) :: A(:,LDA)
  ... .. Locals declaration ... ..
  ... .. Testing arguments ... ..
  ... .. Recursive block ... ..
  CALL ERINFO(LINFO, SRNAME, INFO)
END SUBROUTINE RPOTRF
```

Cholesky

Cholesky factorization, UPLO=L, Intel Pentium III, @ 500 MHz, Atlas



Reference

- **J. Waśniewski, B.S. Andersen and F. Gustavson.**
“Recursive Formulation of Cholesky Algorithm in Fortran 90”. Proceedings of the Workshop on Applied Parallel Computing, Large Scale Scientific and Industrial Problems, PARA’98, 1998, Umeå, Sweden, Lecture Notes in Computer Science Number 1541, Springer, June, pages 574–578.

LU Gauss Factorization

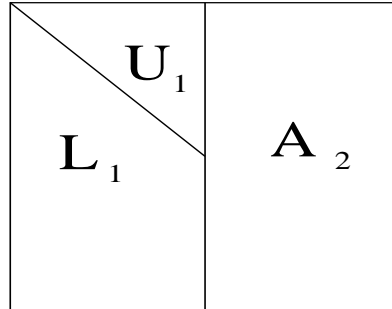
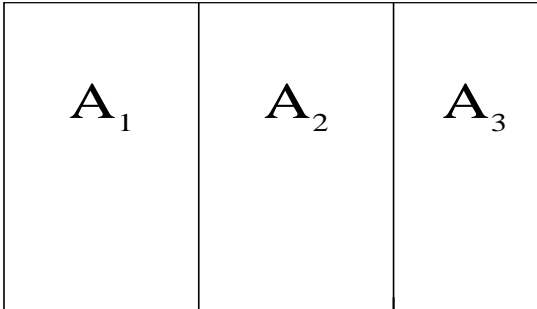
LU Gauss Decomposition

$$AX = B$$

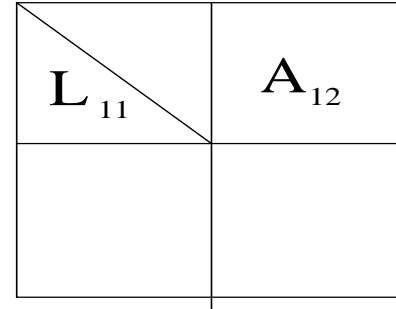
- ***A* is a general real or complex matrix**
- ***X* and *B* are rectangular matrices or vectors**
- ★ **$A = P^T L U$**
- ★ ***P* is a permutation matrix**
- ★ ***U* is an upper triangular matrix**
- ★ ***L* is a lower triangular matrix**
- ★ **The factored form of *A* is then used to solve**
 $A X = B$

LU Gauss Decomposition

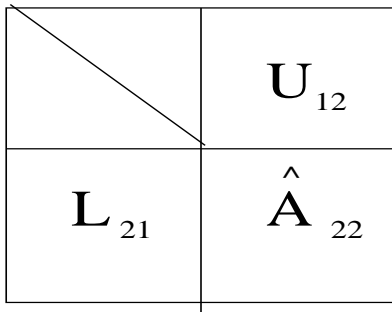
A



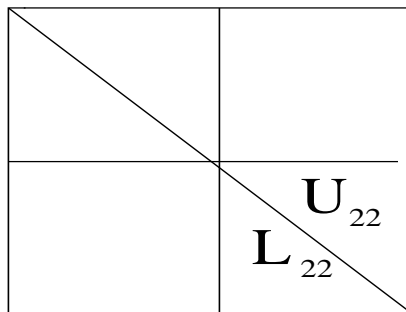
$(L_1, U_1) = LU(A_1)$



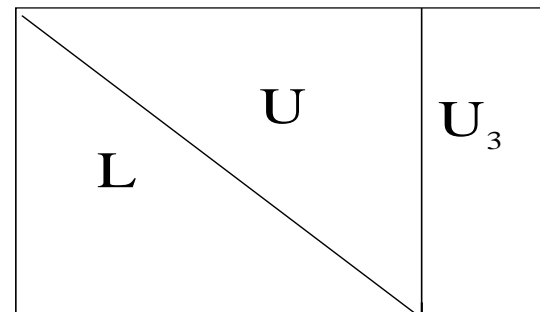
$L_{11} U_{12} = A_{12}$



$\hat{A}_{22} := A_{22} - L_{21} U_{12}$



$(L_{22}, U_{22}) := LU(\hat{A}_{22})$



$L U_3 = A_3$

The Recursive Algorithm without pivoting

LU Gauss Decomposition

Do recursion

- **if** $n > 1$ **then**
 - $(L_1, U_1) := \mathbf{rgausslu}(A_1)$
 - $L_{11}U_{12} = A_{12} \rightarrow \mathbf{RTRSM}$
 - $\hat{A}_{22} := A_{22} - L_{21}U_{12} \rightarrow \mathbf{GEMM}$
 - $(L_{22}, U_{22}) := \mathbf{rgausslu}(\hat{A}_{22})$
- **otherwise**
 - $L_1 := A_1/a_{11}$

End recursion

- **if** $n > m$ **then**
 - $LU_3 = A_3 \rightarrow \mathbf{RTRSM}$

The Recursive Algorithm without pivoting

LU Gauss Decomposition

The Recursive Algorithm with partial pivoting:

Do recursion

- **if $n > 1$ then**
 - $(P_1, L_1, U_1) = \mathbf{rgausslu}(A_1)$
 - **Forward pivot A_2 by $P_1 \rightarrow \mathbf{LASWP}$**
 - $L_{11}U_{12} = A_{12} \rightarrow \mathbf{R TRSM}$
 - $\hat{A}_{22} := A_{22} - L_{21}U_{12} \rightarrow \mathbf{GEMM}$
 - $(P_2, L_{22}, U_{22}) = \mathbf{rgausslu}(\hat{A}_{22})$
 - **Back pivot A_1 by $P_2 \rightarrow \mathbf{LASWP}$, $P = P_2P_1$**
- **otherwise**
 - **pivot A_1 , $L_1 := A_1/a_{11}$ and $U_1 = a_{11}$**

End recursion

- **if $n > m$ then**
 - **Forward pivot A_3 by $P \rightarrow \mathbf{LASWP}$**
 - $LU_3 = A_3 \rightarrow \mathbf{RTRSM}$

where $-P_1$ and P_2 are permutation matrices.

References

- **B.S. Andersen, F. Gustavson, A. Karaivanov, J. Waśniewski, and P.Y. Yalamov. “LAWRA – Linear Algebra with Recursive Algorithms”. Proceedings of the Conference on Parallel Processing and Applied Mathematics, PPAM’99, September, 1999, Kazimierz Dolny, Poland. Published by the Technical University of Częstochowa, pages 63–76.**
- **Presented at the SIAM Conference on Parallel Processing and Scientific Computing, 1999, San Antonio, USA.**
- **Presented at the PARA2000 Workshop on Applied Parallel Computing, 2000, Bergen, Norway.**

$$LL^T = U^T U$$

Cholesky Factorization

November 27, 2003

LAPACK Algorithms and Data Formats

Symmetric/Hermitian matrix

$n = 7, \quad lda = 9, \quad \text{memory needed} = lda \times n = 63$

$a_{1,1_1}$	\diamond	\diamond	\diamond	\diamond	\diamond	\diamond
$a_{2,1_2}$	$a_{2,2_{11}}$	\diamond	\diamond	\diamond	\diamond	\diamond
$a_{3,1_3}$	$a_{3,2_{12}}$	$a_{3,3_{21}}$	\diamond	\diamond	\diamond	\diamond
$a_{4,1_4}$	$a_{4,2_{13}}$	$a_{4,3_{22}}$	$a_{4,4_{31}}$	\diamond	\diamond	\diamond
$a_{5,1_5}$	$a_{5,2_{14}}$	$a_{5,3_{23}}$	$a_{5,4_{32}}$	$a_{5,5_{41}}$	\diamond	\diamond
$a_{6,1_6}$	$a_{6,2_{15}}$	$a_{6,3_{24}}$	$a_{6,4_{33}}$	$a_{6,5_{42}}$	$a_{6,6_{51}}$	\diamond
$a_{7,1_7}$	$a_{7,2_{16}}$	$a_{7,3_{25}}$	$a_{7,4_{34}}$	$a_{7,5_{43}}$	$a_{7,6_{52}}$	$a_{7,7_{61}}$
\circ	\circ	\circ	\circ	\circ	\circ	\circ
\circ	\circ	\circ	\circ	\circ	\circ	\circ

The mapping of 7×7 real symmetric or complex Hermitian matrix for the LAPACK algorithm using the full storage. Lower triangular case.

Symmetric/Hermitian matrix

$n = 7$, $lda = 9$, memory needed = $lda \times n = 63$

$$\begin{pmatrix}
 a_{1,1_1} & a_{1,2_{10}} & a_{1,3_{19}} & a_{1,4_{28}} & a_{1,5_{37}} & a_{1,6_{46}} & a_{1,7_{55}} \\
 \diamond & a_{2,2_{11}} & a_{2,3_{20}} & a_{2,4_{29}} & a_{2,5_{38}} & a_{2,6_{47}} & a_{2,7_{56}} \\
 \diamond & \diamond & a_{3,3_{21}} & a_{3,4_{30}} & a_{3,5_{39}} & a_{3,6_{48}} & a_{3,7_{57}} \\
 \diamond & \diamond & \diamond & a_{4,4_{31}} & a_{4,5_{40}} & a_{4,6_{49}} & a_{4,7_{58}} \\
 \diamond & \diamond & \diamond & \diamond & a_{5,5_{41}} & a_{5,6_{50}} & a_{5,7_{59}} \\
 \diamond & \diamond & \diamond & \diamond & \diamond & a_{6,6_{51}} & a_{6,7_{60}} \\
 \diamond & \diamond & \diamond & \diamond & \diamond & \diamond & a_{7,7_{61}} \\
 \circledast & \circledast & \circledast & \circledast & \circledast & \circledast & \circledast \\
 \circledast & \circledast & \circledast & \circledast & \circledast & \circledast & \circledast
 \end{pmatrix}$$

The mapping of 7×7 real symmetric or complex Hermitian matrix for the LAPACK algorithm using the full storage. Upper triangular case.

Symmetric/Hermitian matrix

$$n = 7, \quad \text{memory needed} = n \times (n + 1) / 2 = 28$$

$$\left(\begin{array}{ccccccc} a_{1,1_1} & & & & & & \\ a_{2,1_2} & a_{2,2_8} & & & & & \\ a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\ a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\ a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\ a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\ a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}} \end{array} \right)$$

The mapping of 7×7 real symmetric or complex Hermitian matrix for the LAPACK algorithm using the packed storage. Lower triangular case.

Example; packed data storage

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

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

UPLO = 'U'

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

UPLO = 'L'

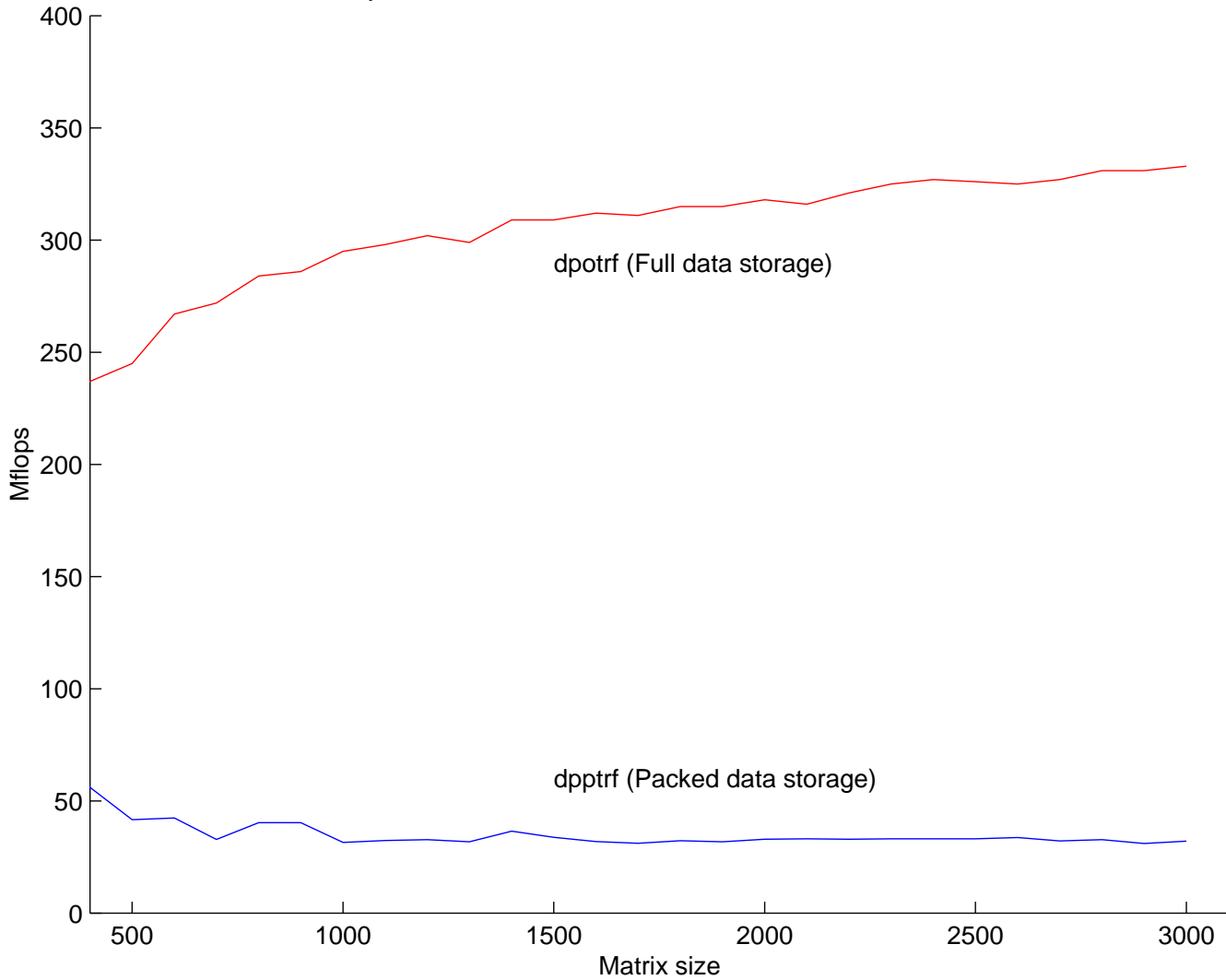
$a_{11} \ a_{21} \ a_{31} \ a_{41} \ a_{22} \ a_{32} \ a_{42} \ a_{33} \ a_{43} \ a_{44}$

$\text{size}(\mathbf{AP}) = n(n+1)/2 = 10$

If $n = 1000$ then $\text{size}(\mathbf{AP}) = 500500$, saving 499500

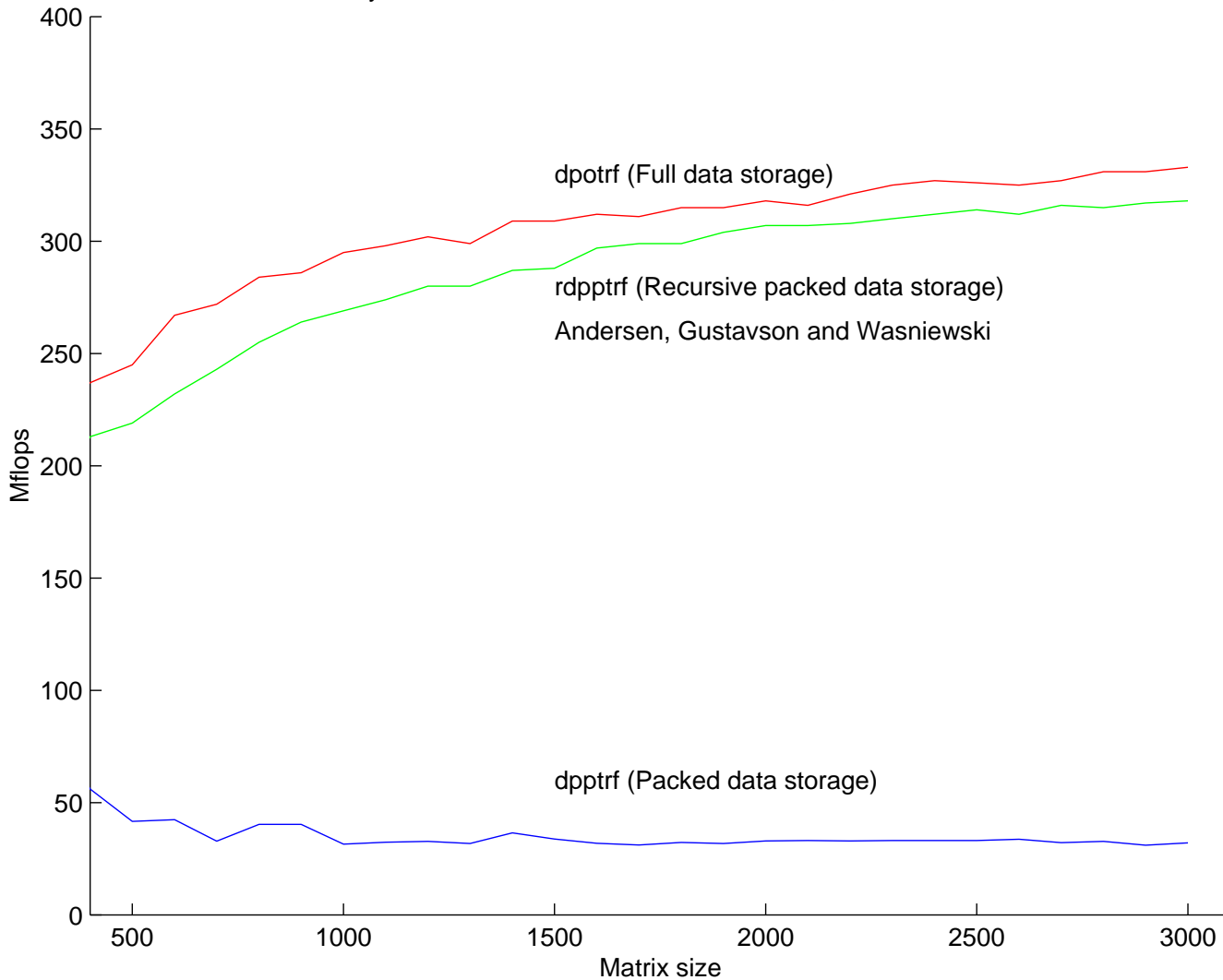
Cholesky, a packed and full data storage

Cholesky factorization, UPLO=L, Intel Pentium III, @ 500 MHz, Atlas



Cholesky, a packed and full data storage

Cholesky factorization, UPLO=L, Intel Pentium III, @ 500 MHz, Atlas



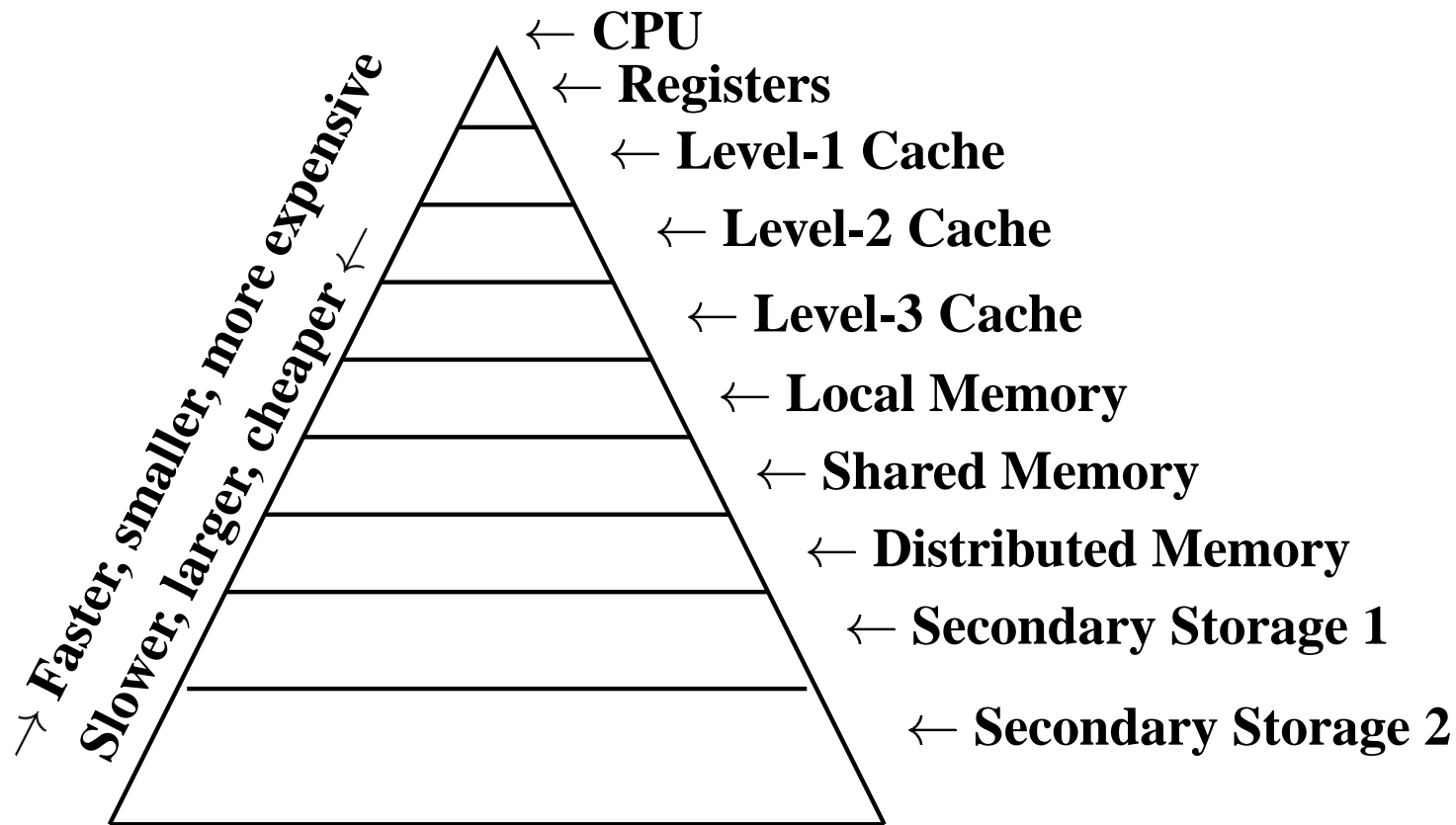
November 27, 2003

New Algorithms

Collaboration:

- **Bjarne S. Andersen**
UNI•C, Danish IT Center, Denmark
- **John A. Gunnels and Fred Gustavson**
IBM Research Center, Yorktown Heights, NY
- **J.K. Reid, Atlas Centre,**
Rutherford Appleton Laboratory, UK
- **A. Karaivanov, M. Marinova, and**
P. Yalamov, Rouse University, Bulgaria
- **Jack Dongarra (discussions)**
University of Tennessee, Knoxville, TN, USA

A computer memory hierarchy



Symmetric/Hermitian Positive Definite Matrices

Recursive Packed Storage Data Format

Cholesky: $A \approx LL^T$

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

$$A = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \times \begin{pmatrix} L_{11}^T & L_{21}^T \\ & L_{22}^T \end{pmatrix}$$

$$A = \begin{pmatrix} A_{11} & A_{21} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11}L_{11}^T & L_{11}L_{21}^T \\ L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{pmatrix}$$

$$A_{11} = L_{11}L_{11}^T, \quad L_{21}L_{11}^T = A_{21} \quad \text{and} \quad \hat{A}_{22} = L_{22}L_{22}^T$$

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

Cholesky: $A \approx LL^T$

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

Do recursion

• **if $n > 1$ then**

- $L_{11} :=$ **rcholesky** of A_{11}
- $L_{21}L_{11}^T = A_{21} \rightarrow$ **RTRSM**
- $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T \rightarrow$ **RSYRK**
- $L_{22} :=$ **rcholesky** of \hat{A}_{22}

• **otherwise**

- $L := \sqrt{A_{11}}$

End recursion

Cholesky, Packed Storage

$$n = 7, \quad \text{memory needed} = n \times (n + 1)/2 = 28$$

$$\left(\begin{array}{ccccccc} a_{1,1_1} & & & & & & \\ a_{2,1_2} & a_{2,2_8} & & & & & \\ a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\ a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\ a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\ a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\ a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}} \end{array} \right)$$

The mapping to array-subscript order of a 7×7 matrix for LAPACK Cholesky Algorithm using packed storage. Lower triangular case.

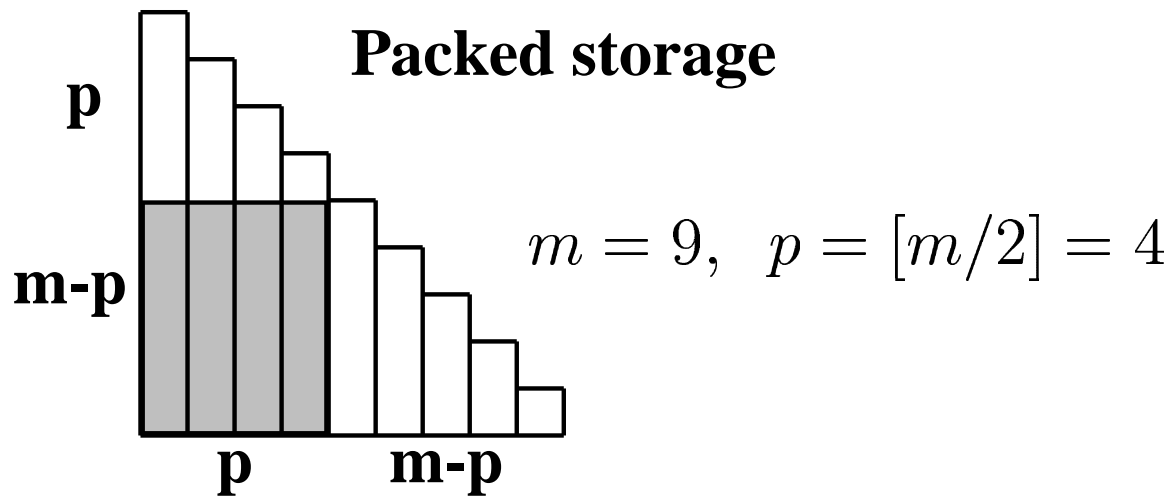
Cholesky, Recursive Packed Storage

$$n = 7, \quad \text{memory needed} = n \times (n + 1) / 2 = 28$$

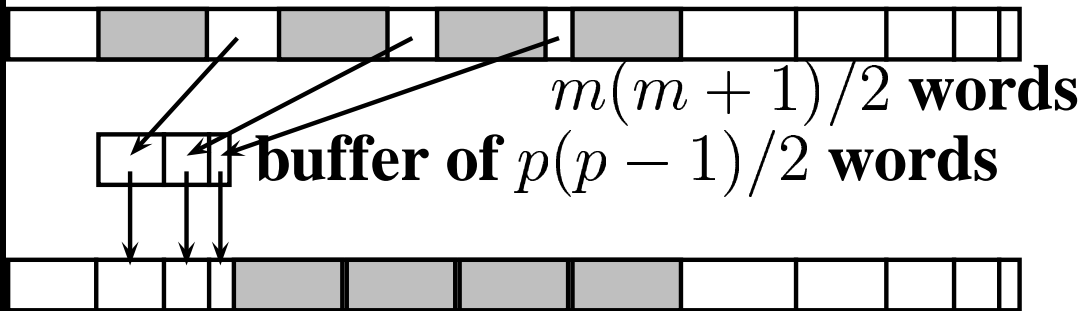
$a_{1,11}$						
$a_{2,12}$	$a_{2,24}$					
$a_{3,13}$	$a_{3,25}$	$a_{3,36}$				
$a_{4,17}$	$a_{4,211}$	$a_{4,315}$	$a_{4,419}$			
$a_{5,18}$	$a_{5,212}$	$a_{5,316}$	$a_{5,420}$	$a_{5,521}$		
$a_{6,19}$	$a_{6,213}$	$a_{6,317}$	$a_{6,422}$	$a_{6,524}$	$a_{6,626}$	
$a_{7,110}$	$a_{7,214}$	$a_{7,318}$	$a_{7,423}$	$a_{7,525}$	$a_{7,627}$	$a_{7,728}$

The mapping to array-subscript order of a 7×7 matrix for the Cholesky Algorithm using the recursive packed storage. The recursive block division is illustrated. Lower triangular case.

Cholesky, Recursive Packed Storage



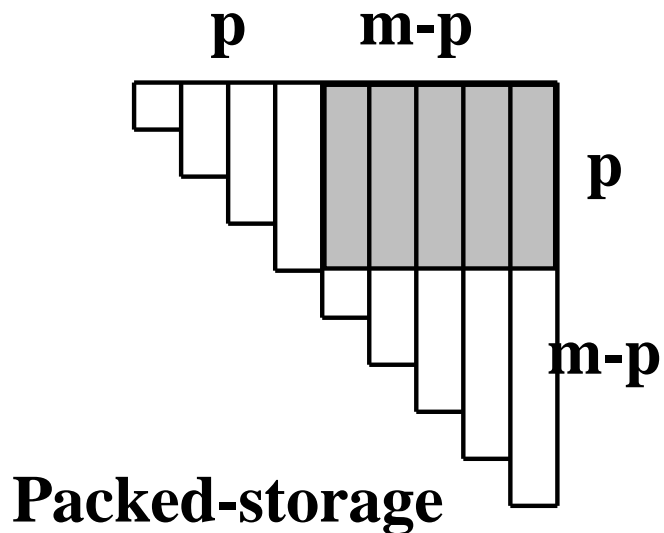
LAPACK packed storage memory map



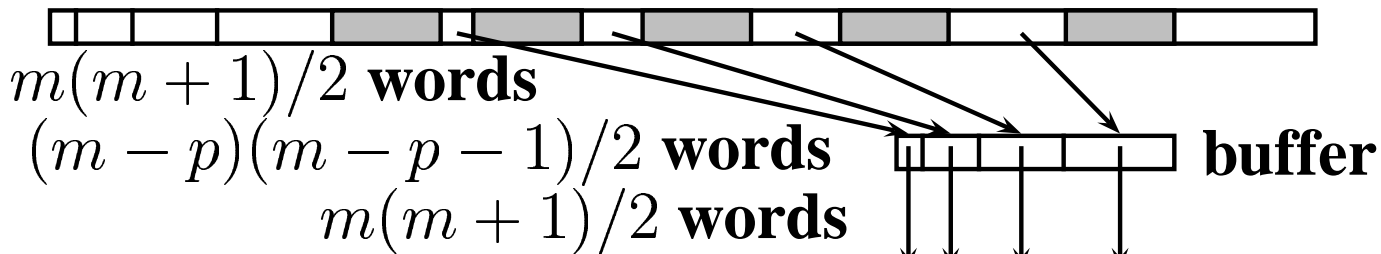
Recursive packed storage memory map

$m(m+1)/2$ words

Cholesky, Recursive Packed Storage

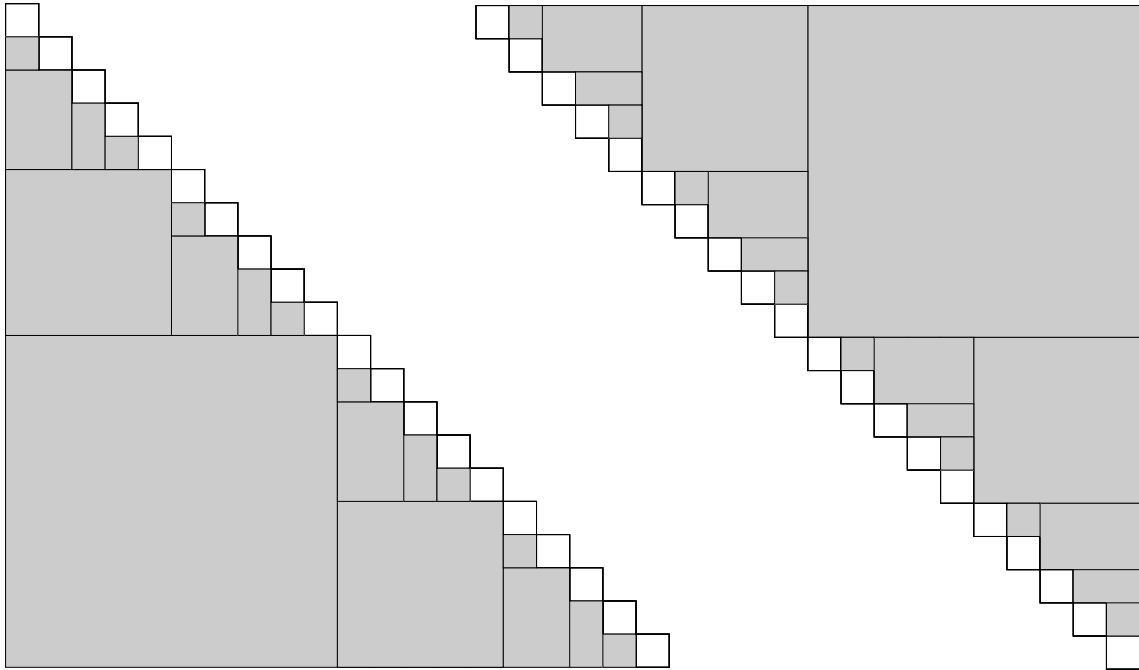


LAPACK packed-storage memory map



Recursive packed-storage memory map

Cholesky, Recursive Packed Storage



Cholesky: $A \approx LL^T$

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

Do recursion

- **if $n > 1$ then**
 - $L_{11} :=$ **rcholesky** of A_{11}
 - $L_{21}L_{11}^T = A_{21} \rightarrow$ **RTRSM**
 - $\hat{A}_{22} := A_{22} - L_{21}L_{21}^T \rightarrow$ **RSYRK**
 - $L_{22} :=$ **rcholesky** of \hat{A}_{22}
- **otherwise**
 - $L := \sqrt{A_{11}}$

End recursion

The Recursive RTRSM and RSYRK

By simple algebraic manipulations

RTRSM:

$$X_{11}A_{11}^T = \alpha B_{11} \quad \text{RTRSM}$$

$$\hat{B}_{12} = B_{12} - \alpha^{-1}X_{11}A_{21}^T \quad \text{GEMM}$$

$$X_{12}A_{22}^T = \alpha\hat{B}_{12} \quad \text{RTRSM}$$

RSYRK:

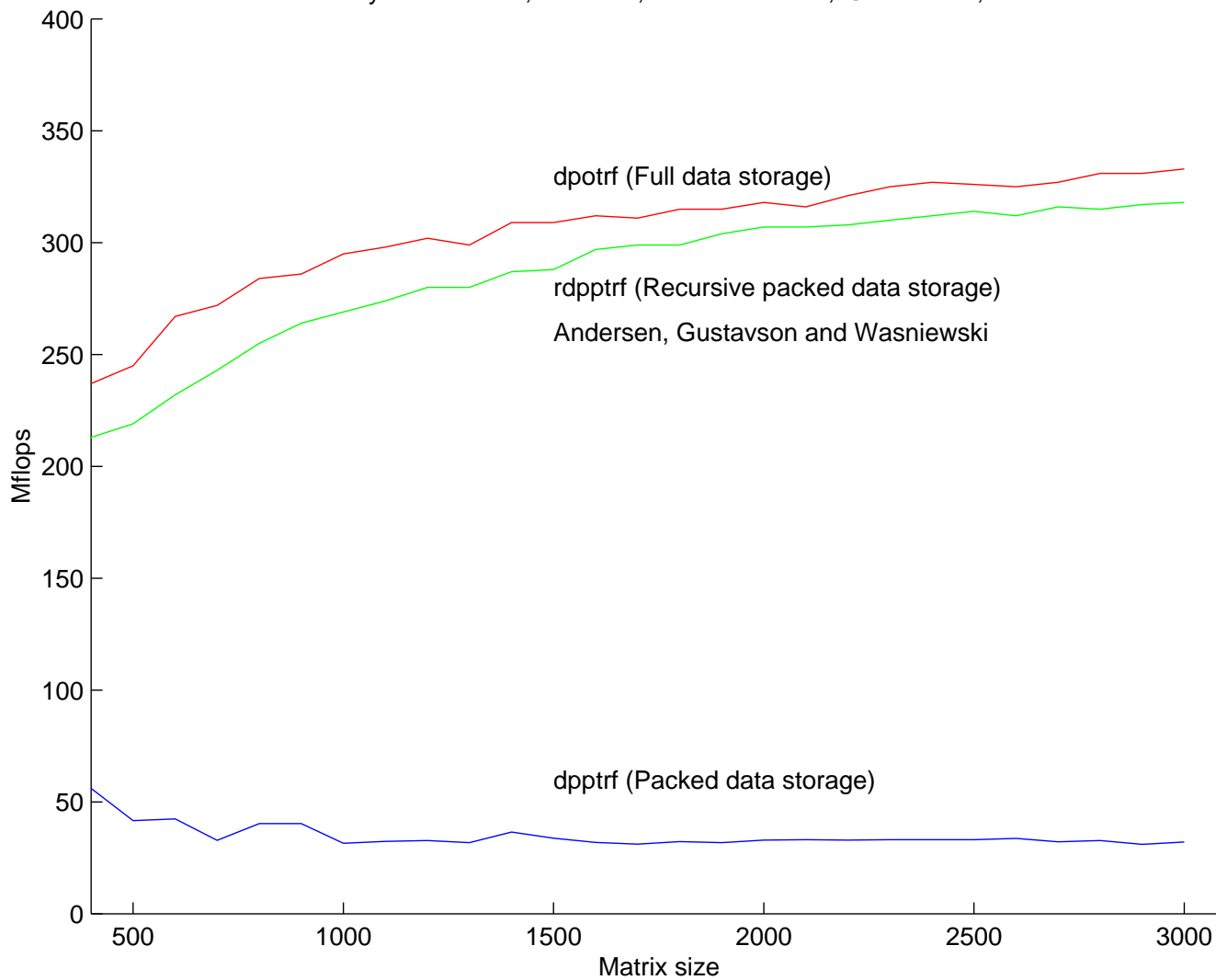
$$C_{11} = \beta C_{11} + \alpha A_{11}A_{11}^T \quad \text{RSYRK}$$

$$C_{21} = \beta C_{21} + \alpha A_{21}A_{11}^T \quad \text{GEMM}$$

$$C_{22} = \beta C_{22} + \alpha A_{21}A_{21}^T \quad \text{RSYRK}$$

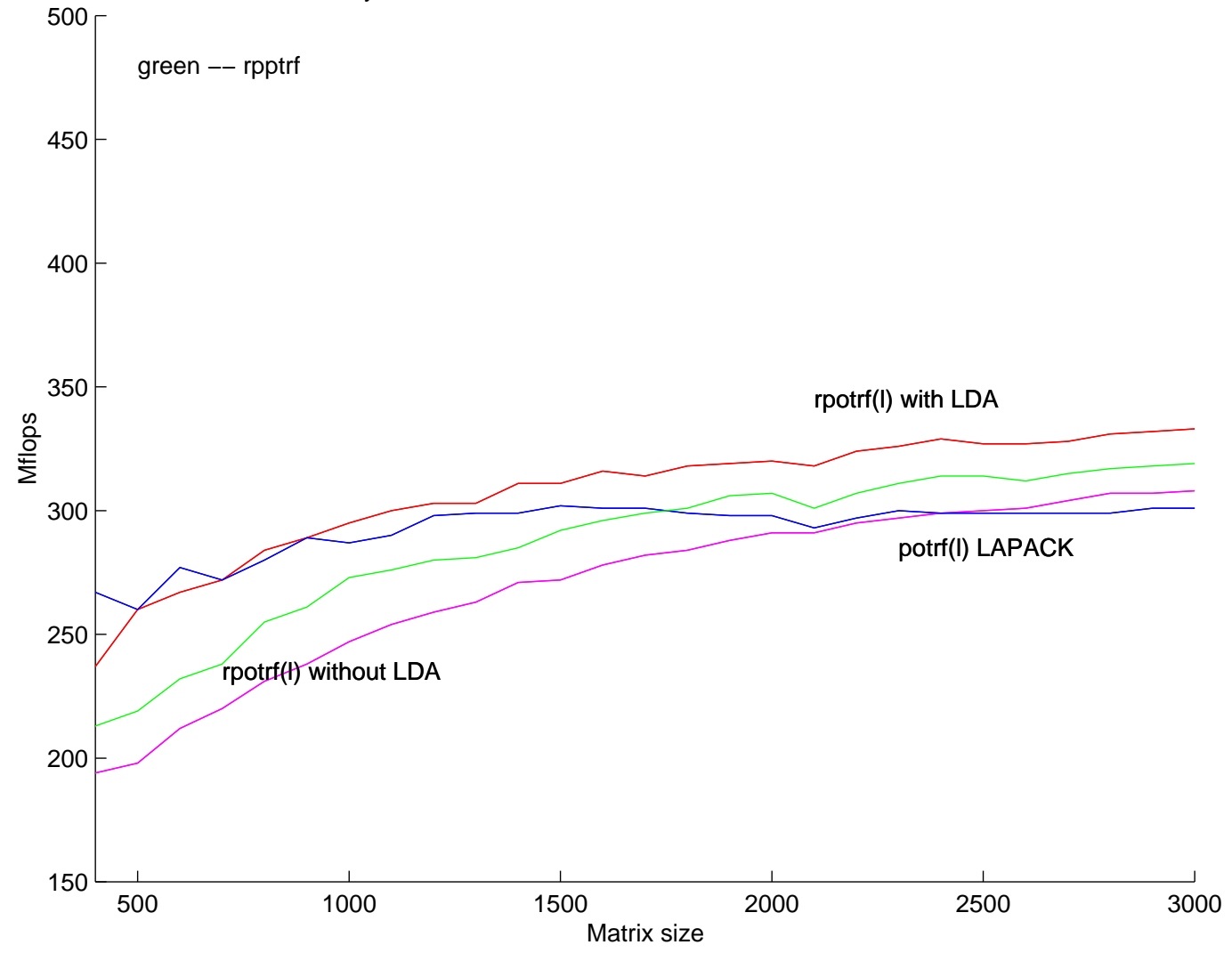
Cholesky: packed vs. full data storage

Cholesky factorization, UPLO=L, Intel Pentium III, @ 500 MHz, Atlas



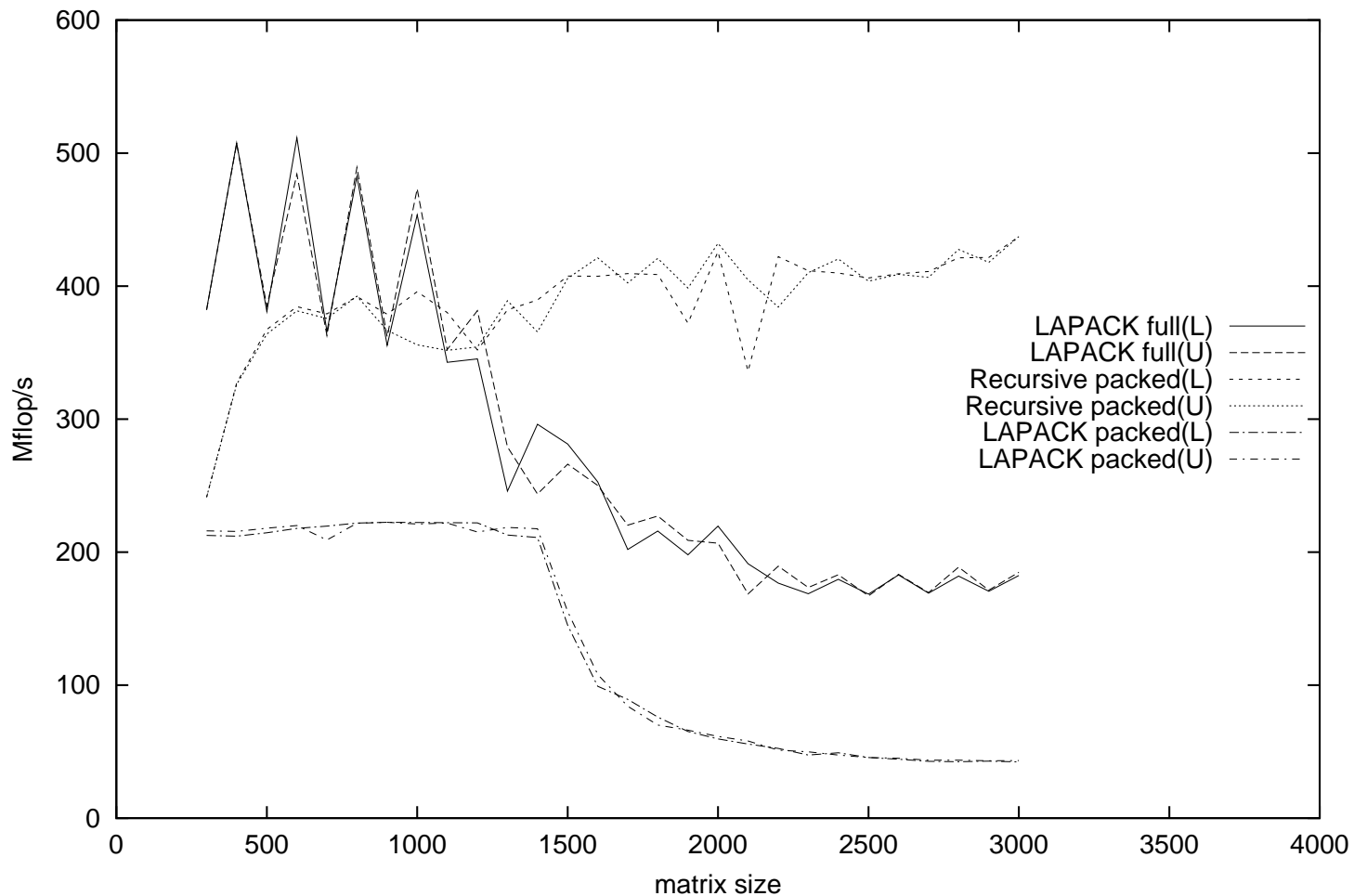
Cholesky

Cholesky factorization, UPLO=L, Intel Pentium III, @ 500 MHz, Atlas



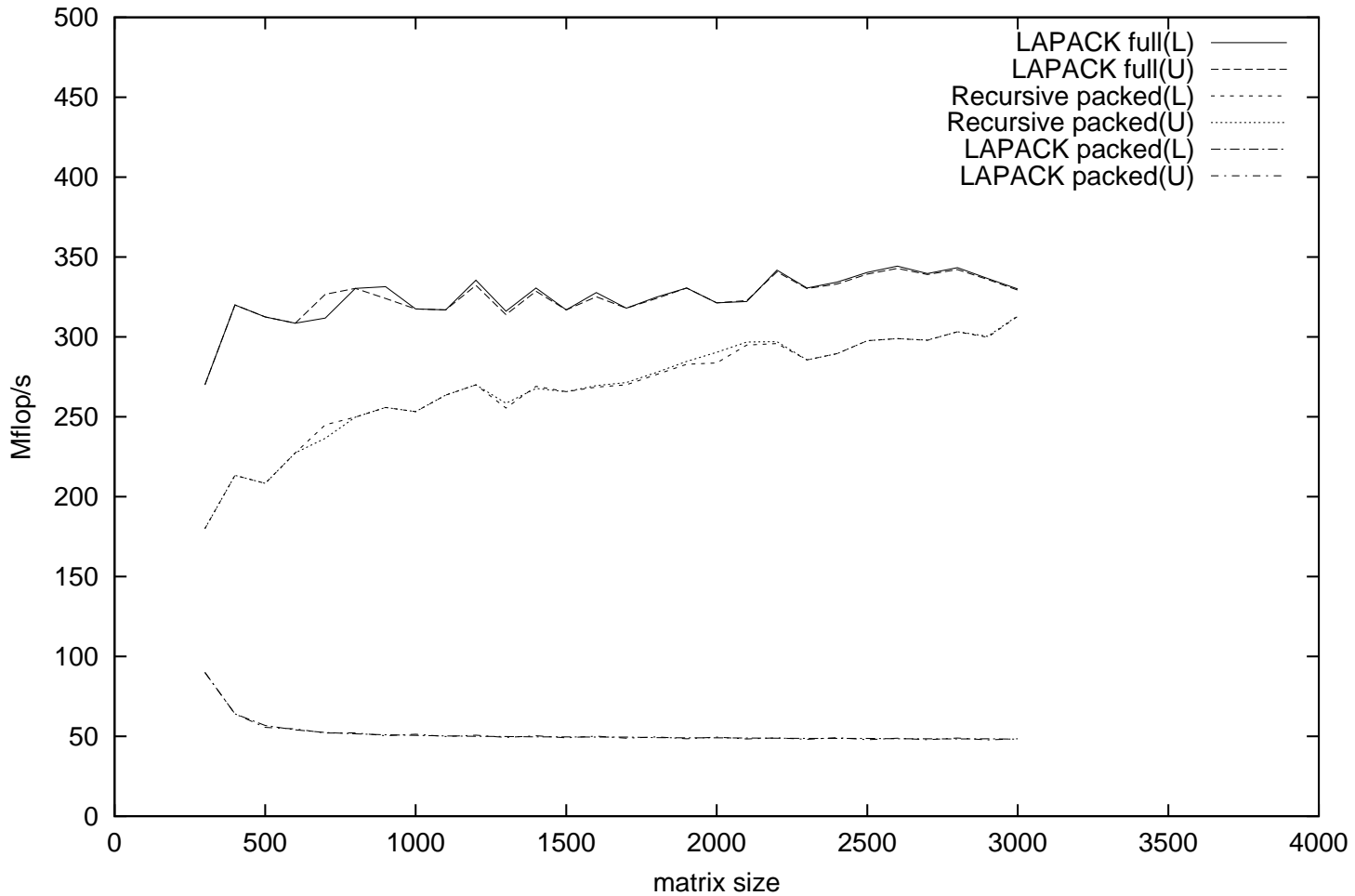
Cholesky, Recursive Packed Storage

Solution performance on SUN UltraSparc II 400 MHz, NRHS=N/10



Cholesky, Recursive Packed Storage

Solution performance on IBM 1 proc. PowerPC 604e 332 MHz, NRHS=N/10



References

- **F.G. Gustavson. “Recursion Leads to Automatic Variable Blocking for Dense Linear-Algebra Algorithms”. IBM Journal of Research and Development, 41(6), November 1997.**
- **Bjarne S. Andersen, Fred G. Gustavson, and Jerzy Waśniewski. “A recursive formulation of Cholesky factorization of a matrix in packed storage”. ACM Transactions on Mathematical Software, 27(2):214–244, June 2001**

Symmetric/Hermitian Positive Definite Matrices

Block Packed Hybrid Storage Data Format

Cholesky, Packed Storage

$$n = 7, \quad \text{memory needed} = n \times (n + 1)/2 = 28$$

$$\left(\begin{array}{ccccccc} a_{1,1_1} & & & & & & \\ a_{2,1_2} & a_{2,2_8} & & & & & \\ a_{3,1_3} & a_{3,2_9} & a_{3,3_{14}} & & & & \\ a_{4,1_4} & a_{4,2_{10}} & a_{4,3_{15}} & a_{4,4_{19}} & & & \\ a_{5,1_5} & a_{5,2_{11}} & a_{5,3_{16}} & a_{5,4_{20}} & a_{5,5_{23}} & & \\ a_{6,1_6} & a_{6,2_{12}} & a_{6,3_{17}} & a_{6,4_{21}} & a_{6,5_{24}} & a_{6,6_{26}} & \\ a_{7,1_7} & a_{7,2_{13}} & a_{7,3_{18}} & a_{7,4_{22}} & a_{7,5_{25}} & a_{7,6_{27}} & a_{7,7_{28}} \end{array} \right)$$

The mapping to array-subscript order of a 7×7 matrix for LAPACK Cholesky Algorithm using packed storage. Lower triangular case.

Cholesky, Block Packed Hybrid Storage

$$n = 7, \quad \text{memory needed} = n \times (n + 1)/2 = 28$$

$a_{1,1_1}$							
$a_{2,1_2}$	$a_{3,1_3}$						
$a_{4,1_4}$	$a_{5,1_5}$	$a_{6,1_6}$					
$a_{7,1_7}$	$a_{2,2_8}$	$a_{3,2_9}$	$a_{4,4_{19}}$				
$a_{2,4_{10}}$	$a_{2,5_{11}}$	$a_{2,6_{12}}$	$a_{5,4_{20}}$	$a_{6,4_{21}}$			
$a_{2,7_{13}}$	$a_{3,3_{14}}$	$a_{3,4_{15}}$	$a_{7,4_{22}}$	$a_{5,5_{23}}$	$a_{5,6_{24}}$		
$a_{3,5_{16}}$	$a_{3,6_{17}}$	$a_{3,7_{18}}$	$a_{7,5_{25}}$	$a_{6,6_{26}}$	$a_{7,6_{27}}$	$a_{7,7_{28}}$	

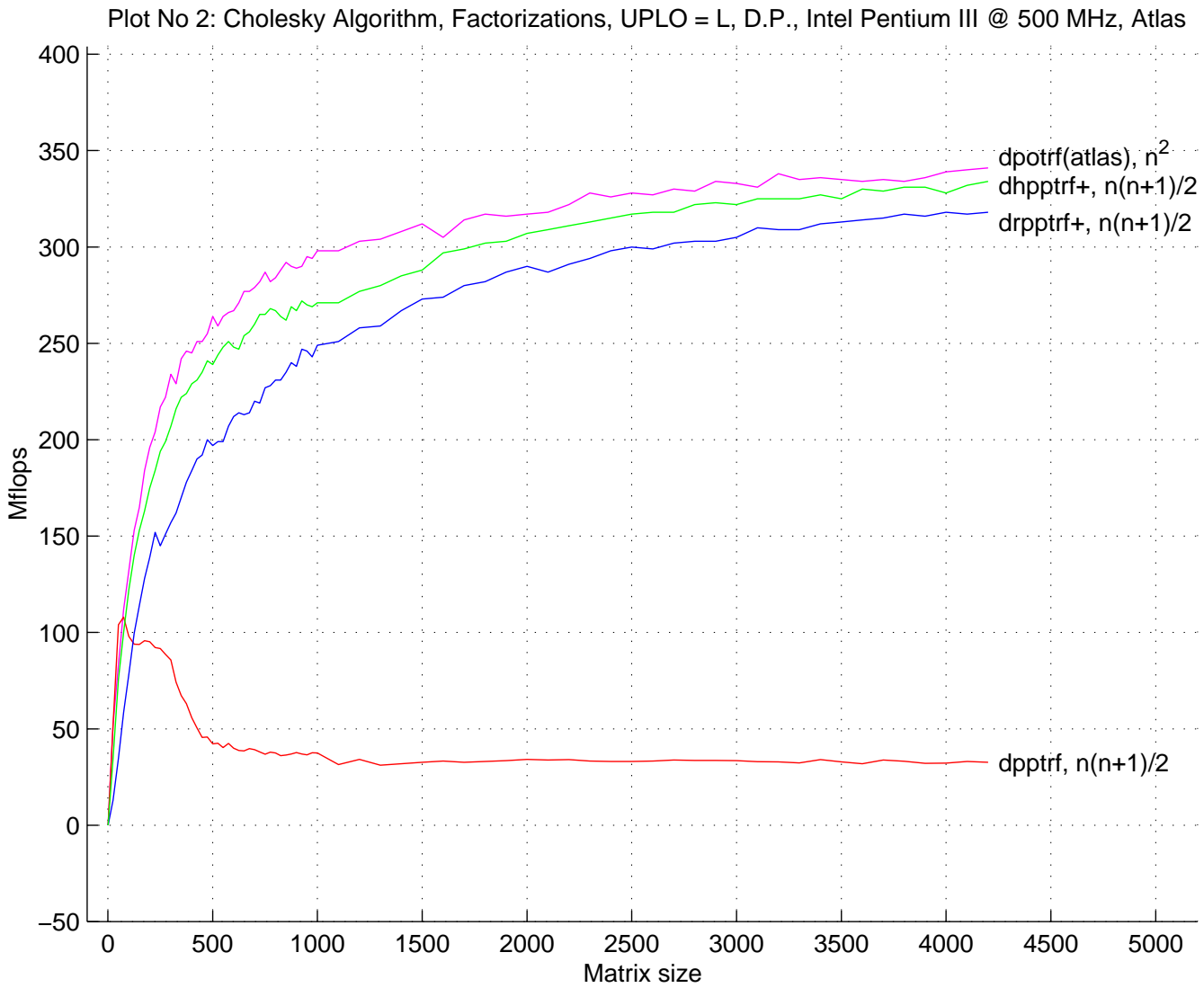
The mapping to array-subscript order of a 7×7 matrix for LAPACK Cholesky Algorithm using block packed hybrid storage. Lower triangular case.

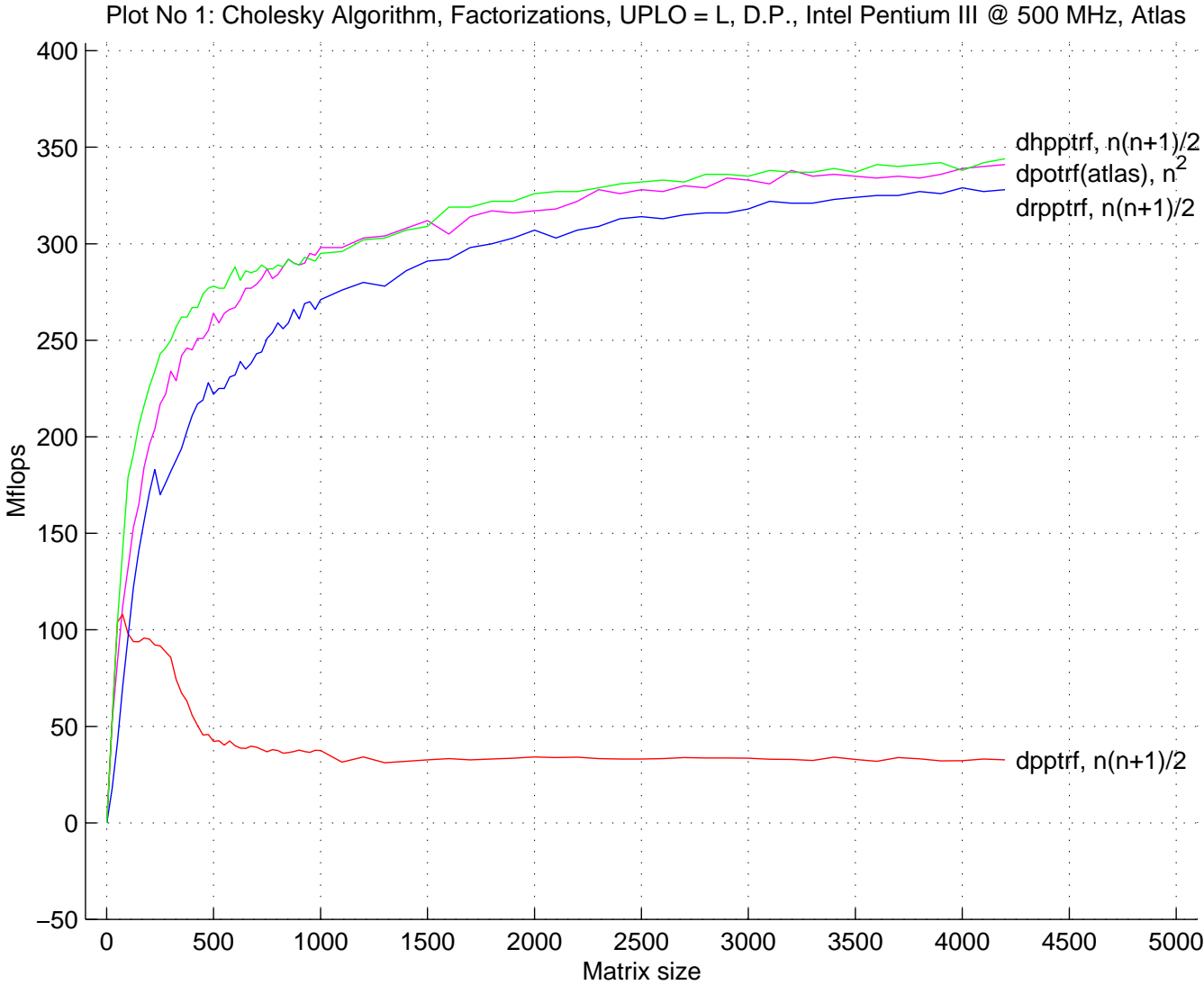
LL^T Implementation for Lower Blocked Hybrid Format

```

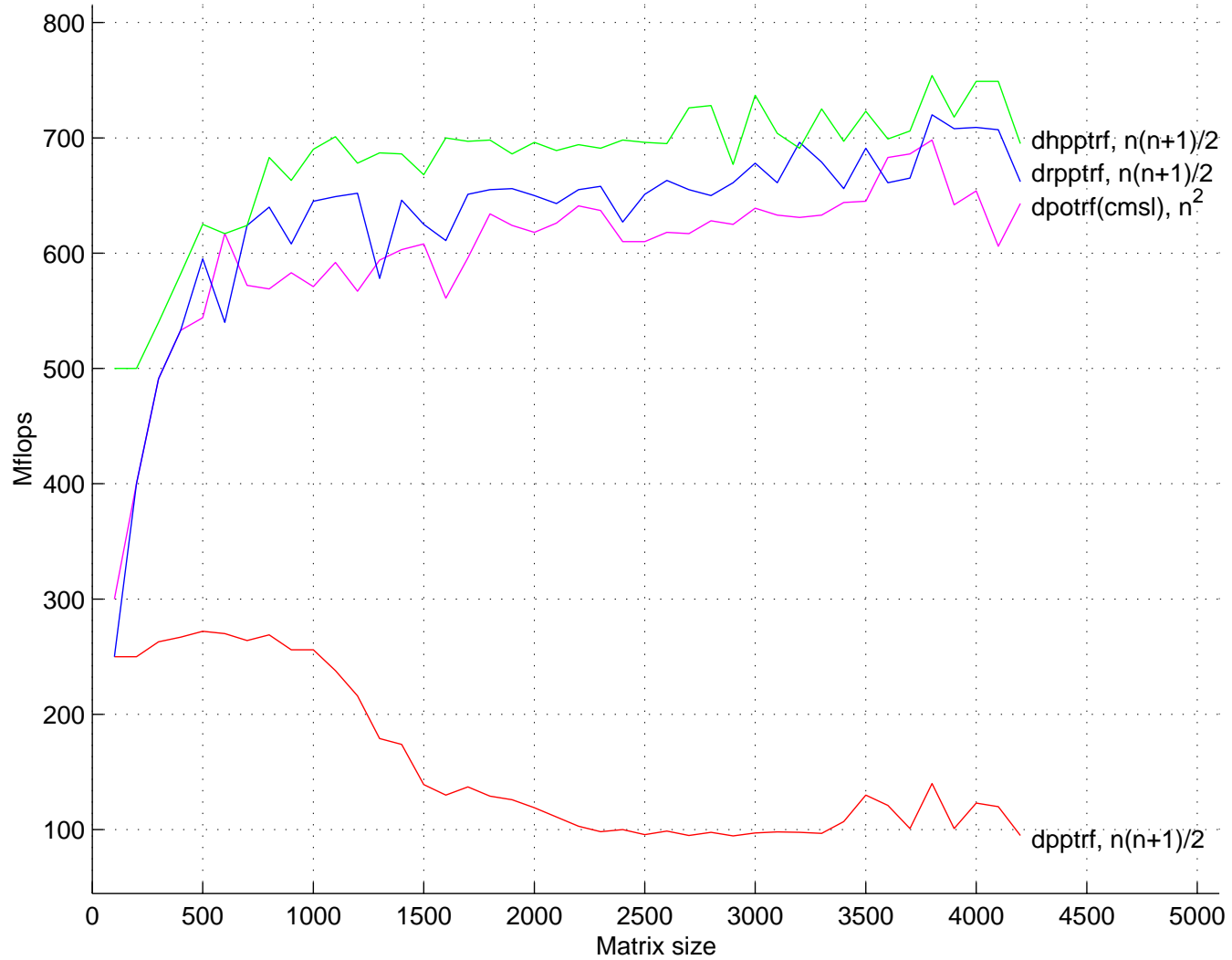
do  $j = 1, l$                                 !  $l = \lceil n/nb \rceil$ 
  do  $k = 1, j - 1$ 
     $A_{jj} = A_{jj} - L_{jk}L_{jk}^T$              ! Call of Level-3 BLAS _SYRK
  do  $i = j + 1, l$ 
     $A_{ij} = A_{ij} - L_{ik}L_{jk}^T$            ! Call of Level-3 BLAS _GEMM
  end do
end do
   $L_{jj}L_{jj}^T = A_{jj}$                        ! Call of Cholesky Kernel subroutine
  do  $i = j + 1, l$ 
     $L_{ij}L_{jj}^T = A_{ij}$                    ! Call of Level-3 BLAS _TRSM
  end do
end do

```





Plot No 1: Cholesky Algorithm, Factorizations, UPLO = L, D.P., Compaq Alpha EV6 @ 500 MHz, CMSL



Reference

- **B.S. Andersen, J.A. Gunnels, F. Gustavson, J.K. Reid, and J. Waśniewski. “A Fully Portable High Performance Minimal Storage Hybrid Format Cholesky Algorithm”. It will appear in *TOMS*.**

Symmetric/Hermitian Indefinite Matrices

Symmetric Indefinite Matrix

$$\begin{array}{ccc}
 A & & A \\
 \left(\begin{array}{cccc}
 a_{11} & a_{12} & a_{13} & a_{14} \\
 & a_{22} & a_{23} & a_{24} \\
 & & a_{33} & a_{34} \\
 & & & a_{44}
 \end{array} \right) & \text{or} & \left(\begin{array}{cccc}
 a_{11} & & & \\
 a_{21} & a_{22} & & \\
 a_{31} & a_{32} & a_{33} & \\
 a_{41} & a_{42} & a_{43} & a_{44}
 \end{array} \right)
 \end{array}$$

A diagonal pivoting method is used to factor A as

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

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

- 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.

Bunch-Kaufman Pivoting Strategy

$\alpha = (1 + \sqrt{17})/8$; $\lambda = |a_{r1}| = \max\{|a_{21}|, \dots, |a_{n1}|\}$

if $\lambda > 0$

if $|a_{11}| \geq \alpha\lambda$ **then** $s = 1$; $P_1 = I$

else

$\sigma = |a_{pr}| = \max\{|a_{1r}, \dots, |a_{r-1,r}, |a_{r+1,r}, \dots, |a_{nr}|\}$

if $\sigma|a_{11}| \geq \alpha\lambda^2$ **then** $s = 1$, $P_1 = I$

else if $|a_{rr}| \geq \alpha\sigma$

$s = 1$ **and choose** P_1 **so** $(P_1^T A P_1)_{11} = a_{rr}$

else

$s = 2$ **and choose** P_1 **so** $(P_1^T A P_1)_{21} = a_{r1}$

end

end

end

Perturbation Approach

Perturbation Approach

- **Perturbation approach with iterative refinement.**
- **Perturbation approach with the Sherman-Morrison-Woodbury formula.**

$$(A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1},$$

where A is n -by- n , U and V are n -by- k matrices (k must be small).

- **Mixed approach.**

Perturbation Approach**Recursive Perturbation-Based Algorithm**

We add a small number $\delta > 0$ to each divisor $|a| < \delta$:

$$a = a + \text{Sgn}(a)\delta$$

where

$$\text{Sgn}(a) = \begin{cases} \text{sign}(a) & \mathbf{if } a \neq 0 \\ 1 & \mathbf{if } a = 0 \end{cases}$$

Symmetric Indefinite Matrix

Numerical Experiments

We ran two kind of experiments:

- Experiments (denoted by Series 1), the random matrices are generated by the LAPACK subroutine DLAGGE.
- Experiments (denoted by Series 2), the matrices A are of the following type^a:

$$A = \begin{pmatrix} \Delta & C \\ C^T & I \end{pmatrix},$$

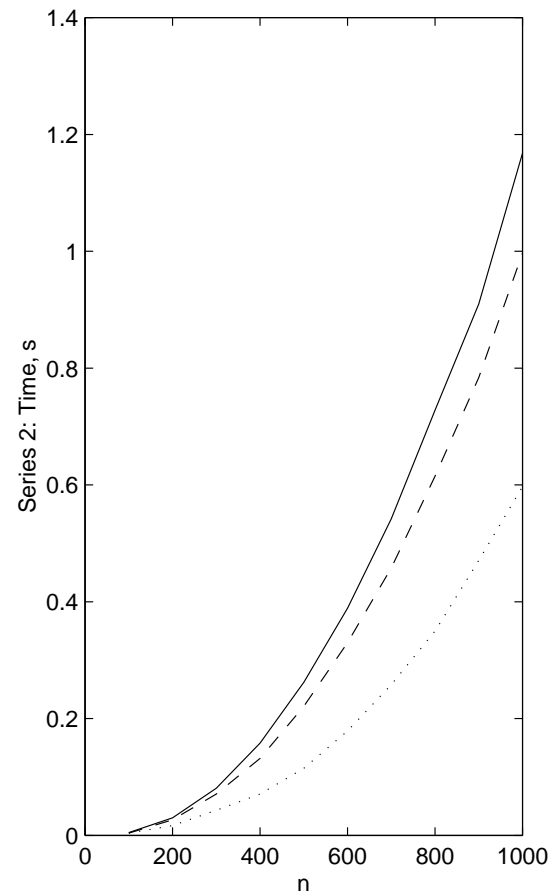
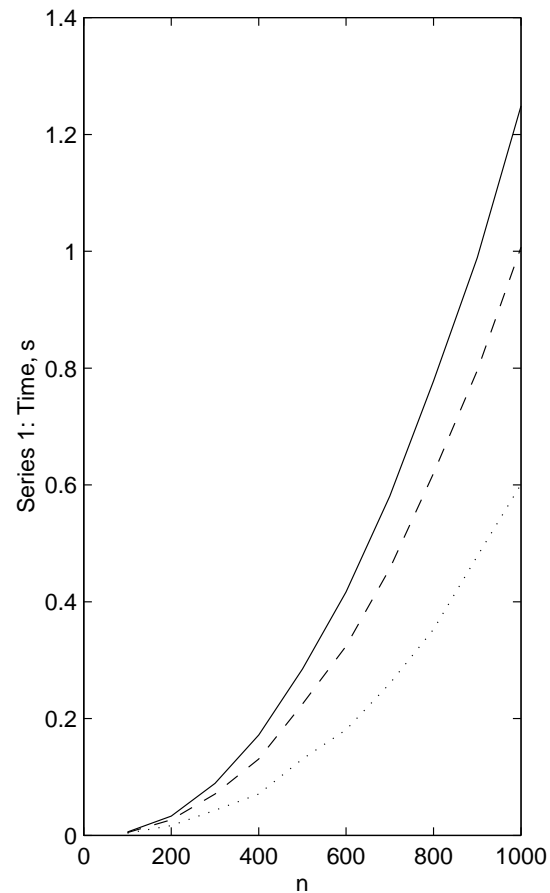
where Δ is a diagonal matrix with small entries, C is a random matrix with large entries, and I is the identity. The entries of Δ are chosen to be less than δ .

^aSuggested to us by John Reid.

Symmetric Indefinite Matrix

Perturbation Approach

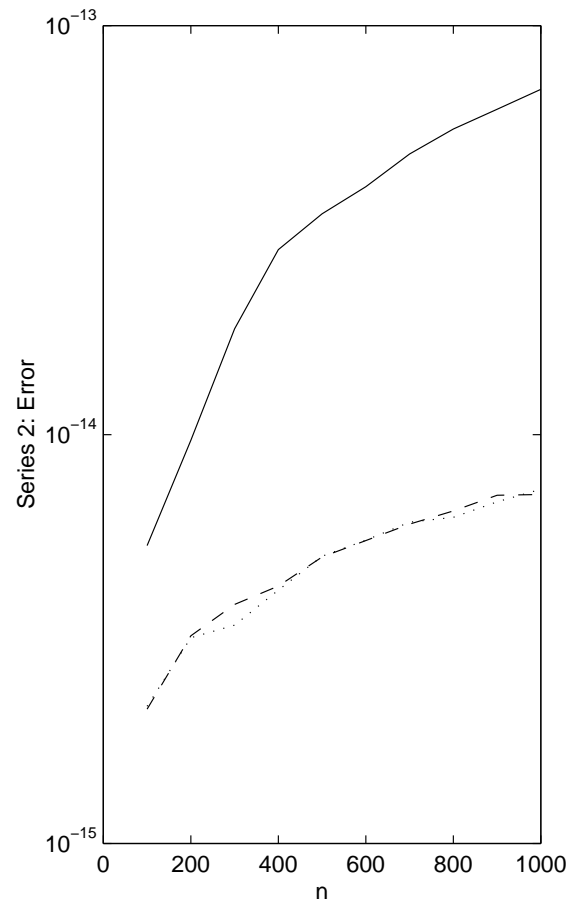
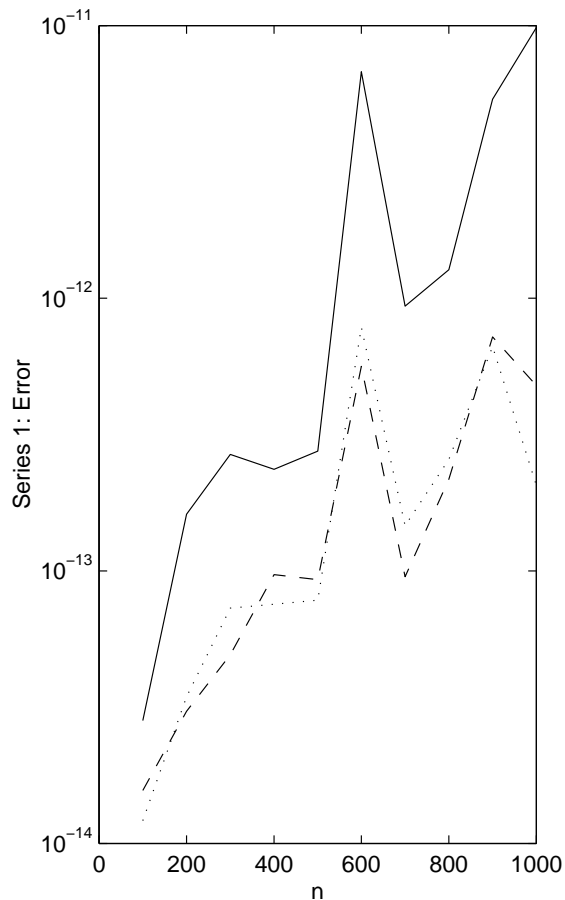
CPU times on the IBM SMP (4xPPC604e/332Mhz) of
DSYSV (—), P_DSYSV (- - -) and DPRSIV (. . .)



Symmetric Indefinite Matrix

Perturbation Approach

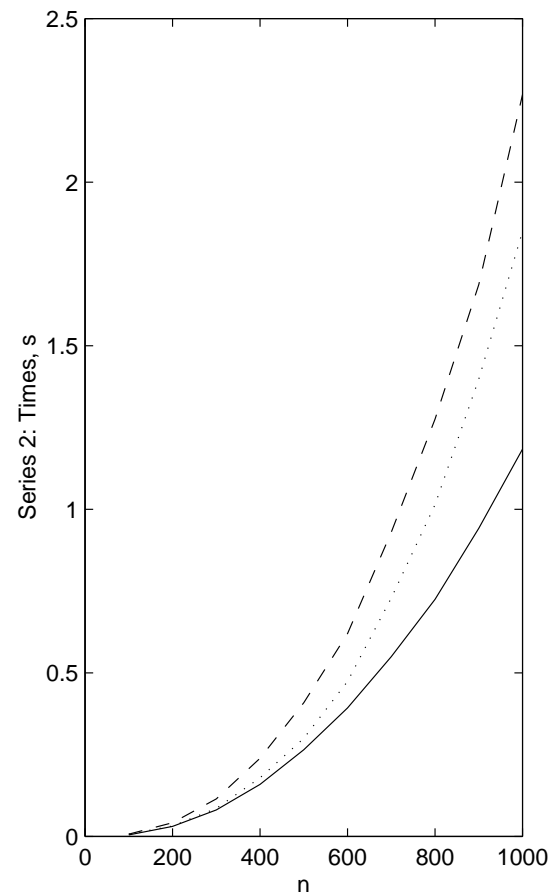
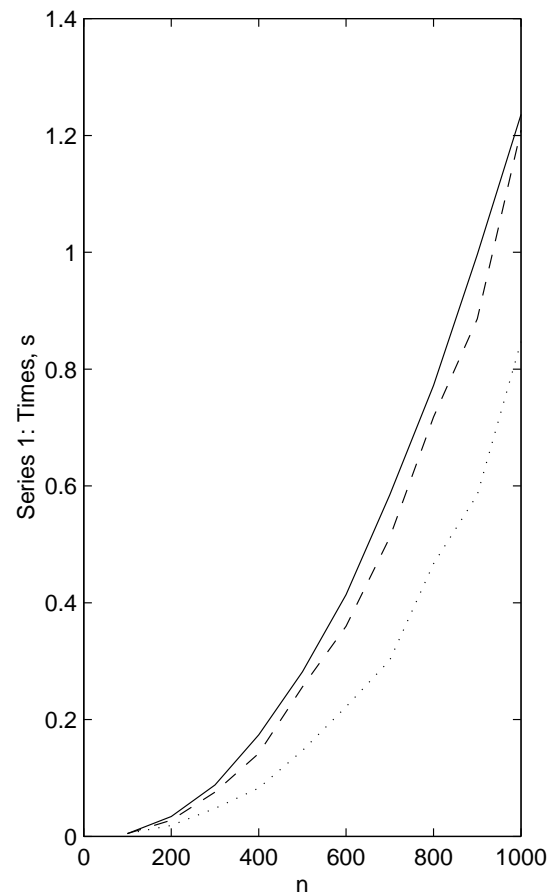
Errors on the IBM SMP (4xPPC604e/332Mhz) of
 DSYSV (—), P_DSYSV (- - -) and DPRSIV (. . .)



Symmetric Indefinite Matrix

Perturbation SMW Approach

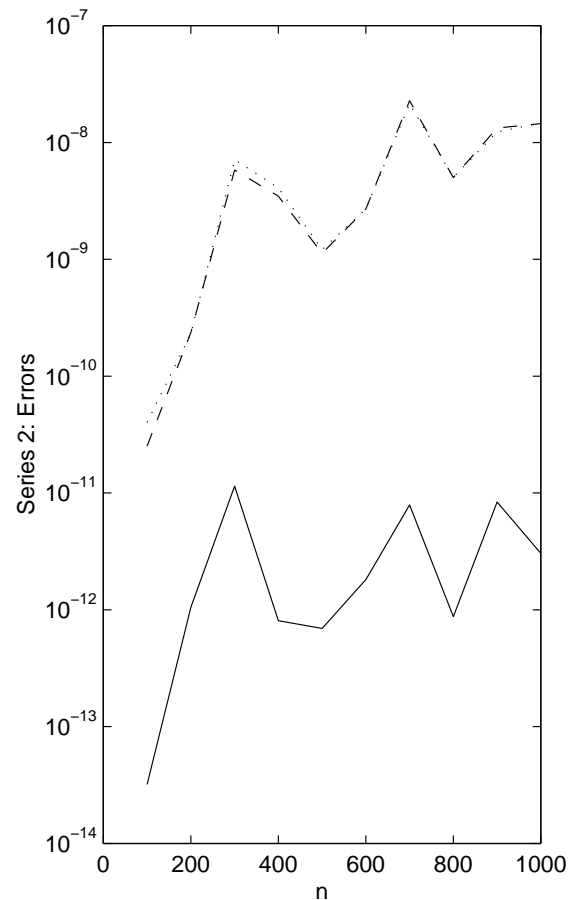
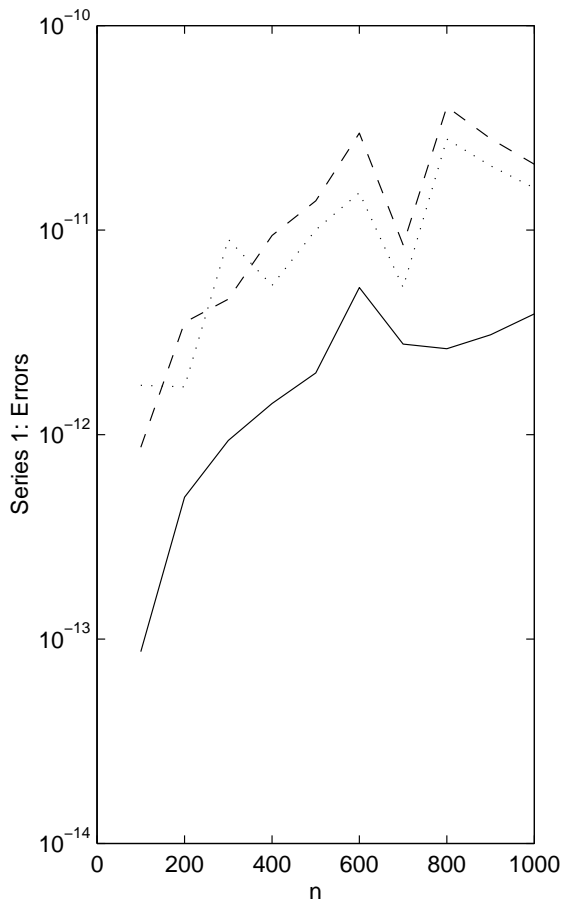
CPU times on the IBM SMP (4xPPC604e/332Mhz) of
DSYSV (—), P_DSYSV (- - -) and DPRSIV (. . .)



Symmetric Indefinite Matrix

Perturbation SMW Approach

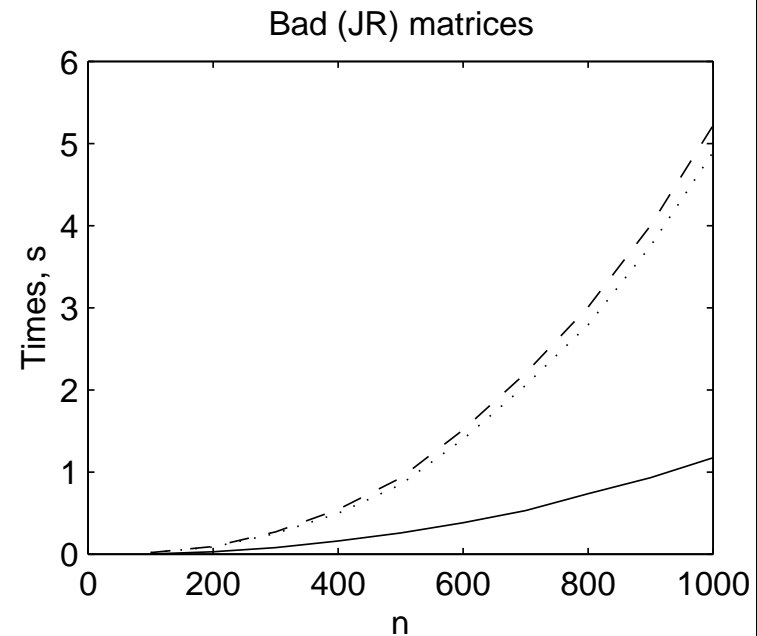
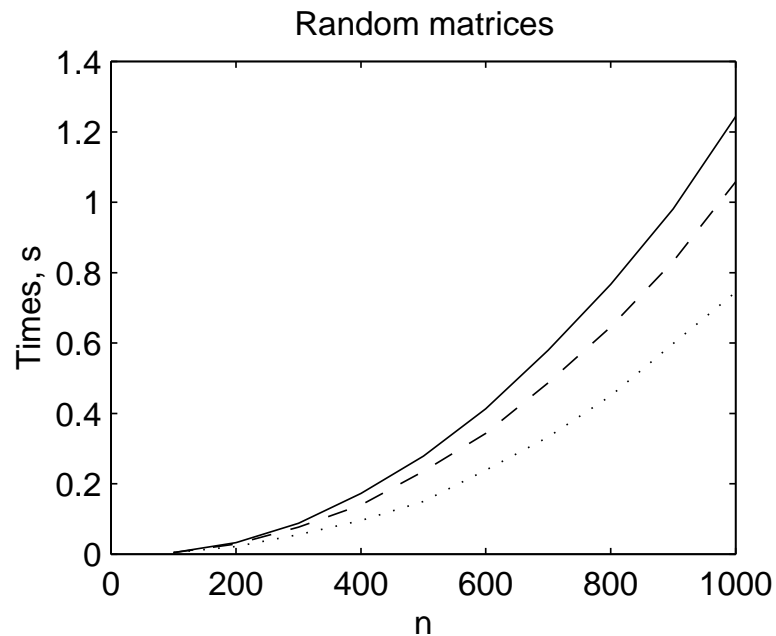
Errors on the IBM SMP (4xPPC604e/332Mhz) of
 DSYSV (—), P_DSYSV (- - -) and DPRSIV (. . .)



Symmetric Indefinite Matrix

Mixed Approach

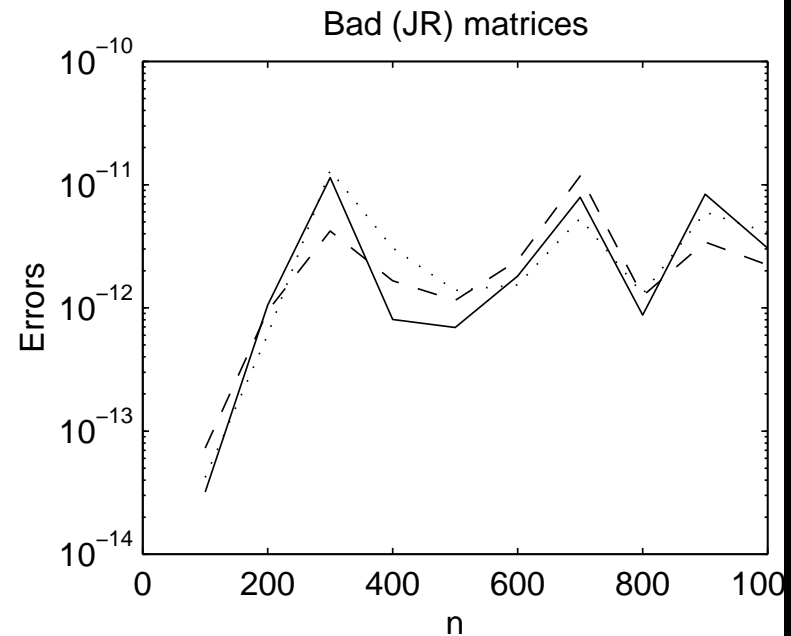
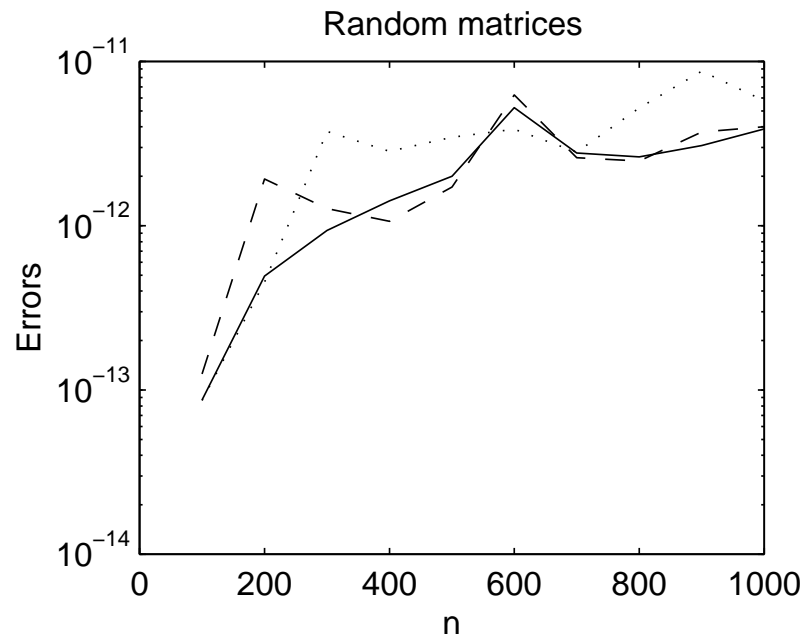
CPU times on the IBM SMP (4xPPC604e/332Mhz) of
DSYSV (—), P_DSYSV (- - -) and DPRSIV (. . .)



Symmetric Indefinite Matrix

Mixed Approach

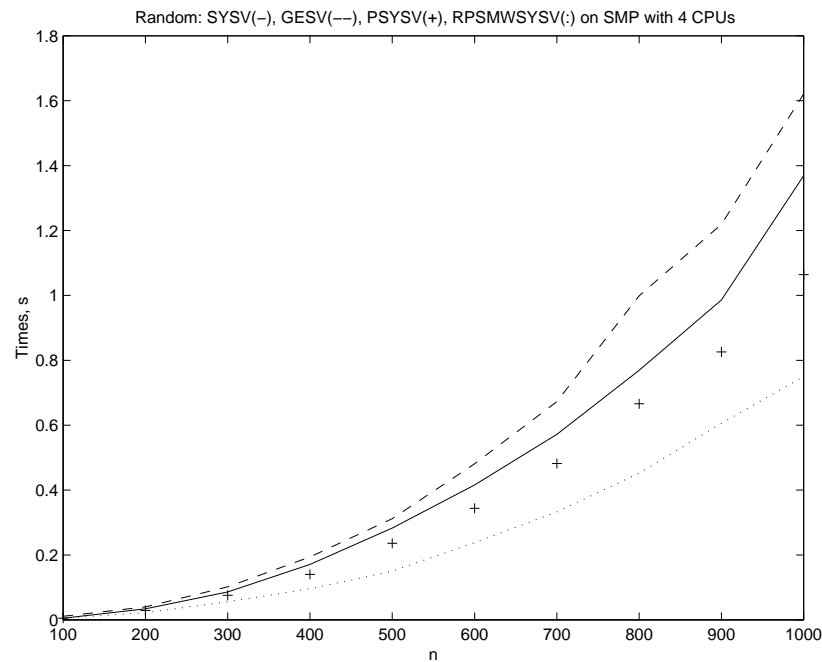
Errors on the IBM SMP (4xPPC604e/332Mhz) of
DSYSV (—), P_DSYSV (- - -) and DPRSIV (. . .)



Perturbation Approach, Numerical Results

LU and LDL^T

CPU times on the IBM SMP (4xPPC604e/332Mhz) of
DSYSV (—), DGESV (- - -) DPSYSV (+) and
RPSMWSYSV (. . .)



Reference

- **F. Gustavson, A. Karaivanov, J. Waśniewski, and P. Yalamov. “A Recursive Formulation of Algorithms for Symmetric Indefinite Linear Systems”. UNI•C report, 15 pages, Number UNIC-99-03, 1999. Submitted to SIAM J. Matrix Anal. Appl.**

Symmetric/Hermitian Indefinite Matrices

Block Packed Algorithm

Symmetric Indefinite Matrix

$$\begin{array}{c} A \\ \left(\begin{array}{cccc} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{array} \right) \end{array} \quad \text{or} \quad \begin{array}{c} A \\ \left(\begin{array}{cccc} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array} \right) \end{array}$$

A diagonal pivoting method is used to factor A as

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

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

- 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.

Bunch-Kaufman Pivoting Strategy

$$\alpha = (1 + \sqrt{17})/8; \lambda = |a_{r1}| = \max\{|a_{21}|, \dots, |a_{n1}|\}$$

if $\lambda > 0$

if $|a_{11}| \geq \alpha\lambda$ **then** $s = 1$; $P_1 = I$

else

$$\sigma = |a_{pr}| = \max\{|a_{1r}, \dots, |a_{r-1,r}, |a_{r+1,r}, \dots, |a_{nr}|\}$$

if $\sigma|a_{11}| \geq \alpha\lambda^2$ **then** $s = 1$, $P_1 = I$

else if $|a_{rr}| \geq \alpha\sigma$

$s = 1$ **and choose** P_1 **so** $(P_1^T A P_1)_{11} = a_{rr}$

else

$s = 2$ **and choose** P_1 **so** $(P_1^T A P_1)_{21} = a_{r1}$

end

end

end

Symmetric Indefinite Matrix

$$\mathbf{n} = 7, \quad \text{memory needed} = \mathbf{n} \times (\mathbf{n} + 1)/2 = 28$$

$$\left(\begin{array}{ccccccc} a_{1,11} & & & & & & \\ a_{2,12} & a_{2,28} & & & & & \\ a_{3,13} & a_{3,29} & a_{3,314} & & & & \\ a_{4,14} & a_{4,210} & a_{4,315} & a_{4,419} & & & \\ a_{5,15} & a_{5,211} & a_{5,316} & a_{5,420} & a_{5,523} & & \\ a_{6,16} & a_{6,212} & a_{6,317} & a_{6,421} & a_{6,524} & a_{6,626} & \\ a_{7,17} & a_{7,213} & a_{7,318} & a_{7,422} & a_{7,525} & a_{7,627} & a_{7,728} \end{array} \right)$$

The mapping of 7×7 real symmetric, complex symmetric or complex Hermitian matrix for the LAPACK algorithm using the packed storage. Lower triangular case.

Symmetric Indefinite Matrix

$n = 7$, $lda = 9$, $nb = 2$, memory needed = 51

$$\left(\begin{array}{ccccccc}
 a_{1,11} & \otimes & & & & & \odot \\
 a_{2,12} & a_{2,211} & & & & & \odot \\
 a_{3,13} & a_{3,212} & a_{3,321} & & & & \otimes & \odot \\
 a_{4,14} & a_{4,213} & a_{4,322} & a_{4,429} & & & & \odot \\
 a_{5,15} & a_{5,214} & a_{5,323} & a_{5,430} & a_{5,537} & & \otimes & \odot \\
 a_{6,16} & a_{6,215} & a_{6,324} & a_{6,431} & a_{6,538} & a_{6,643} & & \odot \\
 a_{7,17} & a_{7,216} & a_{7,325} & a_{7,432} & a_{7,539} & a_{7,644} & & a_{7,749} \\
 \otimes & \otimes & \otimes & \otimes & \otimes & \otimes & & \otimes \\
 \otimes & \otimes & \otimes & \otimes & \otimes & \otimes & & \otimes
 \end{array} \right)$$

The mapping of 7×7 real symmetric, complex symmetric or complex Hermitian matrix for the algorithm using the block packed overlapped storage.

Lower triangular case.

Symmetric Indefinite Matrix

$n = 7$, $lda = 9$, $nb = 3$, **memory needed = 54**

$$\left(\begin{array}{ccc|ccc|c}
 a_{1,11} & \otimes & \otimes & \odot & & & \\
 a_{2,12} & a_{2,211} & \otimes & \odot & & & \\
 a_{3,13} & a_{3,212} & a_{3,321} & \odot & & & \\
 a_{4,14} & a_{4,213} & a_{4,322} & a_{4,431} & \otimes & \otimes & \odot \\
 a_{5,15} & a_{5,214} & a_{5,323} & a_{5,432} & a_{5,538} & \otimes & \odot \\
 a_{6,16} & a_{6,215} & a_{6,324} & a_{6,433} & a_{6,539} & a_{6,645} & \odot \\
 a_{7,17} & a_{7,216} & a_{7,325} & a_{7,434} & a_{7,540} & a_{7,646} & a_{7,752} \\
 \otimes & \otimes & \otimes & \otimes & \otimes & \otimes & \otimes \\
 \otimes & \otimes & \otimes & \otimes & \otimes & \otimes & \otimes_{54}
 \end{array} \right)$$

The mapping of 7×7 real symmetric, complex symmetric or complex Hermitian matrix for the algorithm using the block packed overlapped storage.

Lower triangular case.

Symmetric Indefinite Matrix

```
SUBROUTINE SSPTRF( UPLO, N, AP, &  
                  IPIV, INFO)
```

```
!
```

```
CHARACTER UPLO
```

```
INTEGER INFO, N
```

```
!
```

```
INTEGER IPIV( * )
```

```
REAL AP( * )
```

Symmetric Indefinite Matrix

uplo = 'l', n = 7, lda = 9, nb = 2, length of AP = 28

$$\mathbf{AP} = \left(\begin{array}{cc|c|c} a_{1,11} & \otimes & \odot & \diamond \\ a_{2,12} & a_{2,211} & \odot & \\ a_{3,13} & a_{3,212} & a_{3,321} & \\ a_{4,14} & a_{4,213} & a_{4,322} & \\ a_{5,15} & a_{5,214} & a_{5,323} & \\ a_{6,16} & a_{6,215} & a_{6,324} & \\ a_{7,17} & a_{7,216} & a_{7,325} & \\ \hline \otimes & \otimes & \otimes & \\ \otimes & \otimes & \otimes & \end{array} \right)$$

length of BUFF = 31

$$\mathbf{BUFF} = \left(\begin{array}{cc|cc|c} a_{3,31} & \otimes & \odot & & \\ a_{4,32} & a_{4,49} & \odot & & \\ a_{5,33} & a_{5,410} & a_{5,517} & \otimes & \odot \\ a_{6,34} & a_{6,411} & a_{6,518} & a_{6,623} & \odot \\ a_{7,35} & a_{7,412} & a_{7,519} & a_{7,624} & a_{7,729} \\ \hline \otimes & \otimes & \otimes & \otimes & \otimes \\ \otimes & \otimes & \otimes & \otimes & \otimes \end{array} \right)$$

Block packed overlapped storage. Lower triangular case.

Symmetric Indefinite Matrix

$n = 7$, $lda = 9$, $nb = 3$, length of **BUFF** = 36

$$\mathbf{BUFF} = \left(\begin{array}{ccc|c} a_{1,11} & \otimes & \otimes & \odot \\ a_{2,12} & a_{2,211} & \otimes & \odot \\ a_{3,13} & a_{3,212} & a_{3,321} & \odot \\ a_{4,14} & a_{4,213} & a_{4,322} & a_{4,431} \\ a_{5,15} & a_{5,214} & a_{5,323} & a_{5,432} \\ a_{6,16} & a_{6,215} & a_{6,324} & a_{6,433} \\ a_{7,17} & a_{7,216} & a_{7,325} & a_{7,434} \\ \hline \otimes & \otimes & \otimes & \otimes \\ \otimes & \otimes & \otimes & \otimes \end{array} \right)$$

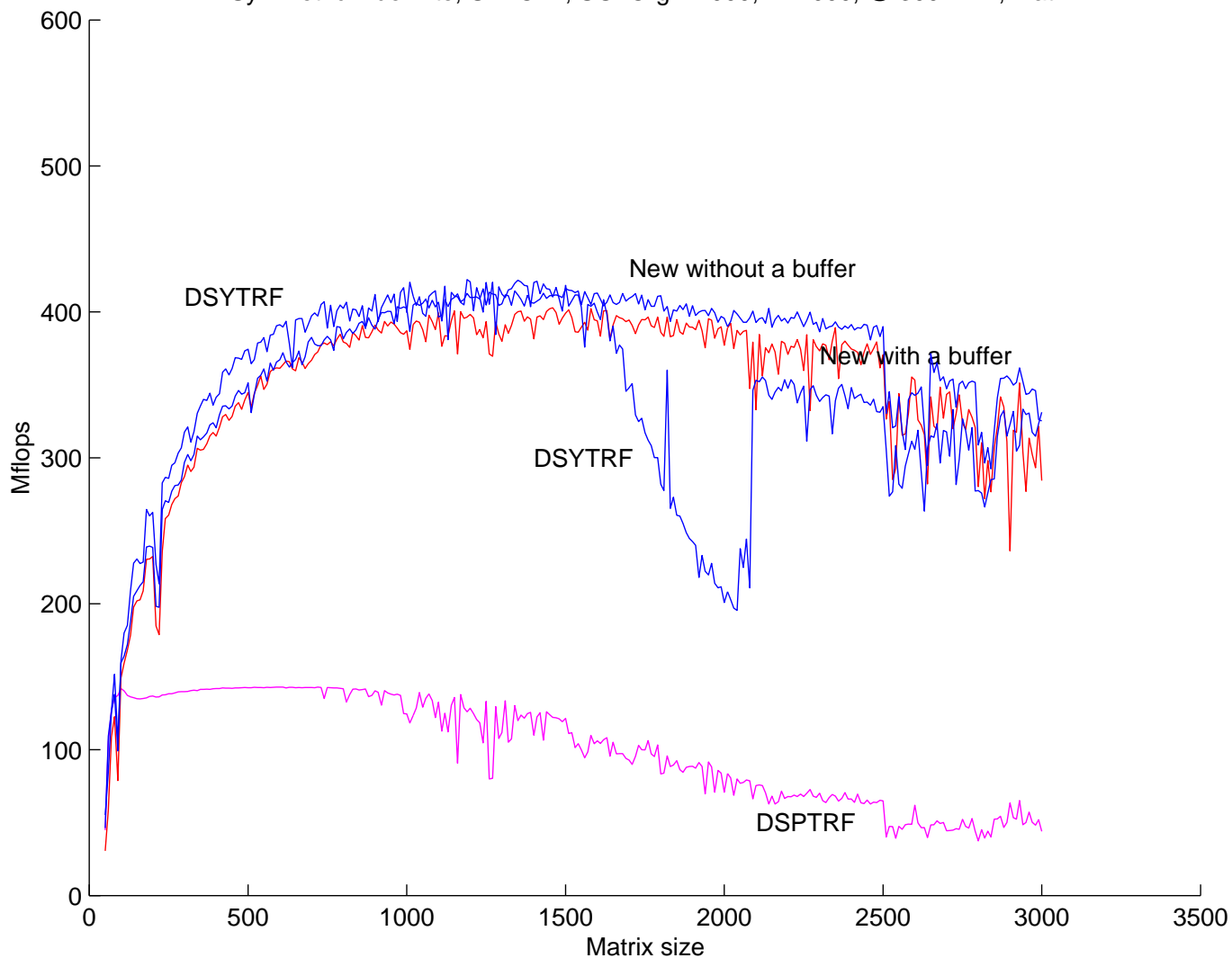
length of **AP** = 28

$$\mathbf{AP} = \left(\begin{array}{ccc|cc} a_{4,41} & \otimes & \otimes & \odot & \diamond \\ a_{5,42} & a_{5,58} & \otimes & \odot & \diamond \\ a_{6,43} & a_{6,59} & a_{6,615} & \odot & \diamond \\ a_{7,44} & a_{7,510} & a_{7,616} & a_{7,722} & \diamond \\ \hline \otimes & \otimes & \otimes & \otimes & \\ \otimes & \otimes & \otimes & \otimes & \end{array} \right)$$

Block packed overlapped storage. Lower triangular case.

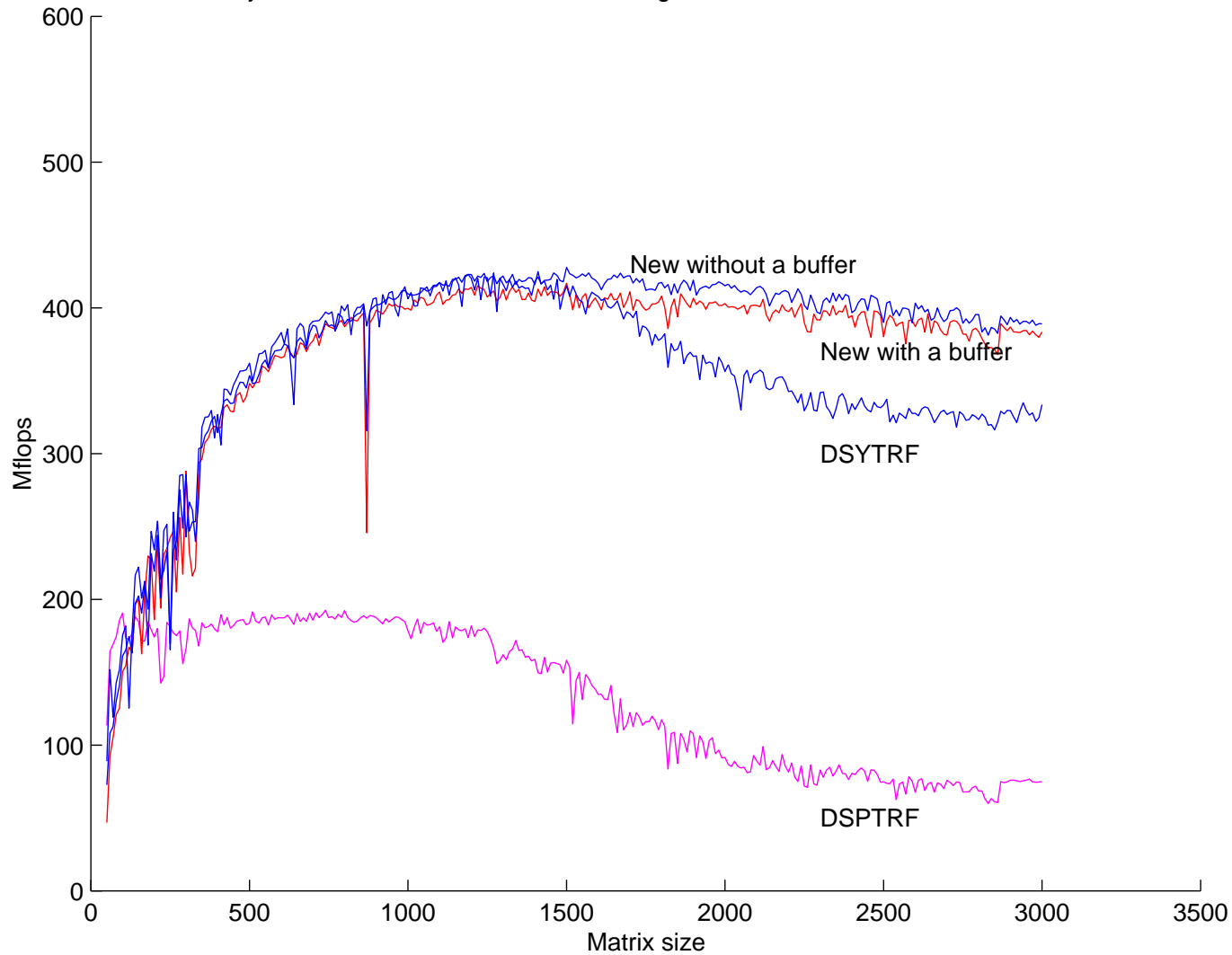
LDL^T , Four Factorization Algorithms, SGI Lib

Symmetric Indefinite, UPLO=L, SGI Origin 2000, R12000, @ 300 MHz, Math



LDL^T , Four Factorization Algorithms, Atlas Lib

Symmetric Indefinite, UPLO=L, SGI Origin 2000, R12000, @ 300 MHz, Atlas



Reference

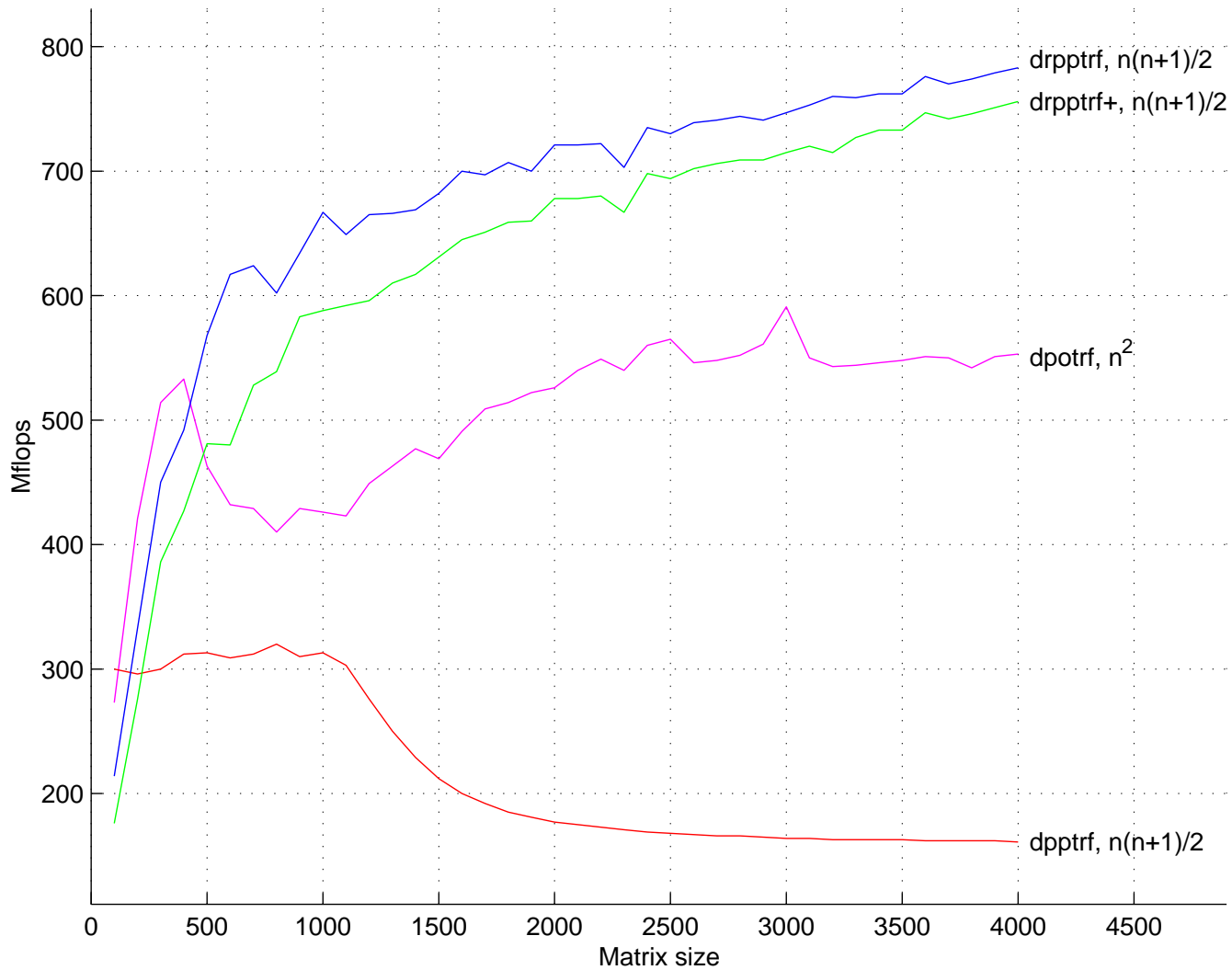
- **F. Gustavson, A. Karaivanov, M. Marinova, J. Wasniewski, and P. Yalamov. “A Fast Minimal Storage Symmetric Indefinite Solver”. In Para’2000 Conference Proceedings, Bergen, Norway, 2000.**

Symmetric Matrices

More Results

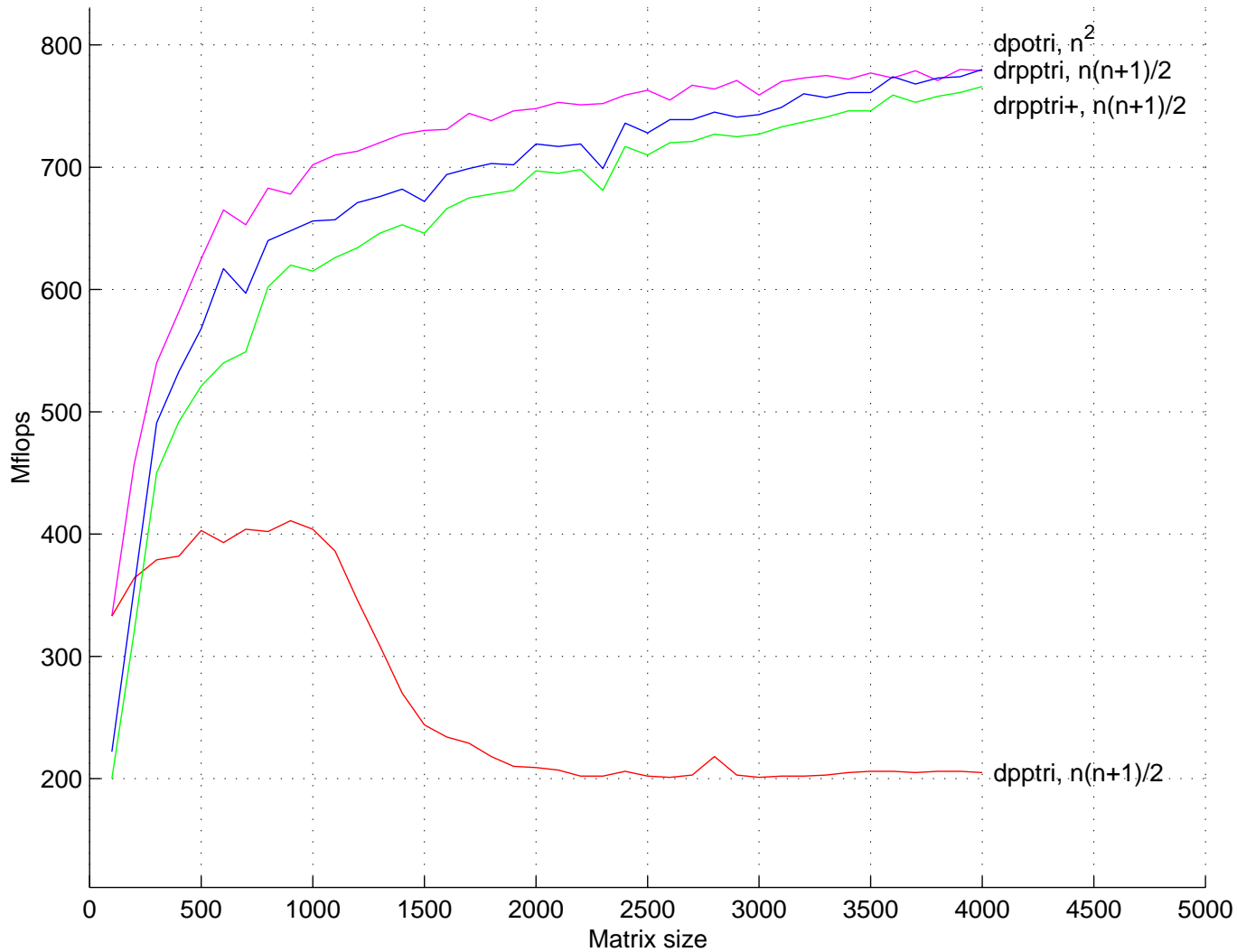
Factorization

Cholesky Algorithm, Factorizations, UPLO = L, COMPAQ Alpha EV6 @ 1000 MHz, Cxml



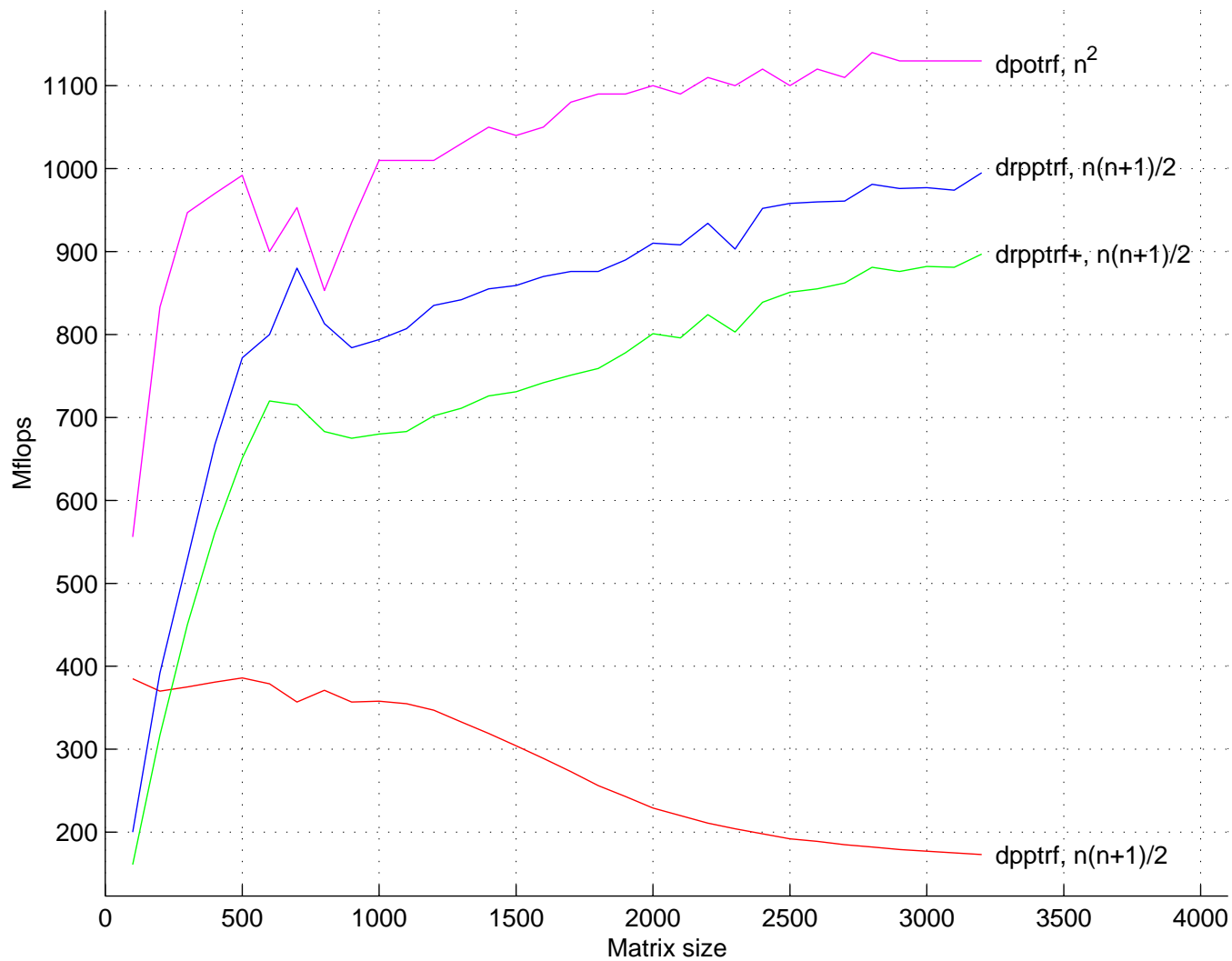
Inversion

Cholesky Algorithm, Inverse, UPLO = L, COMPAQ Alpha EV6 @ 1000 MHz, Cxmi



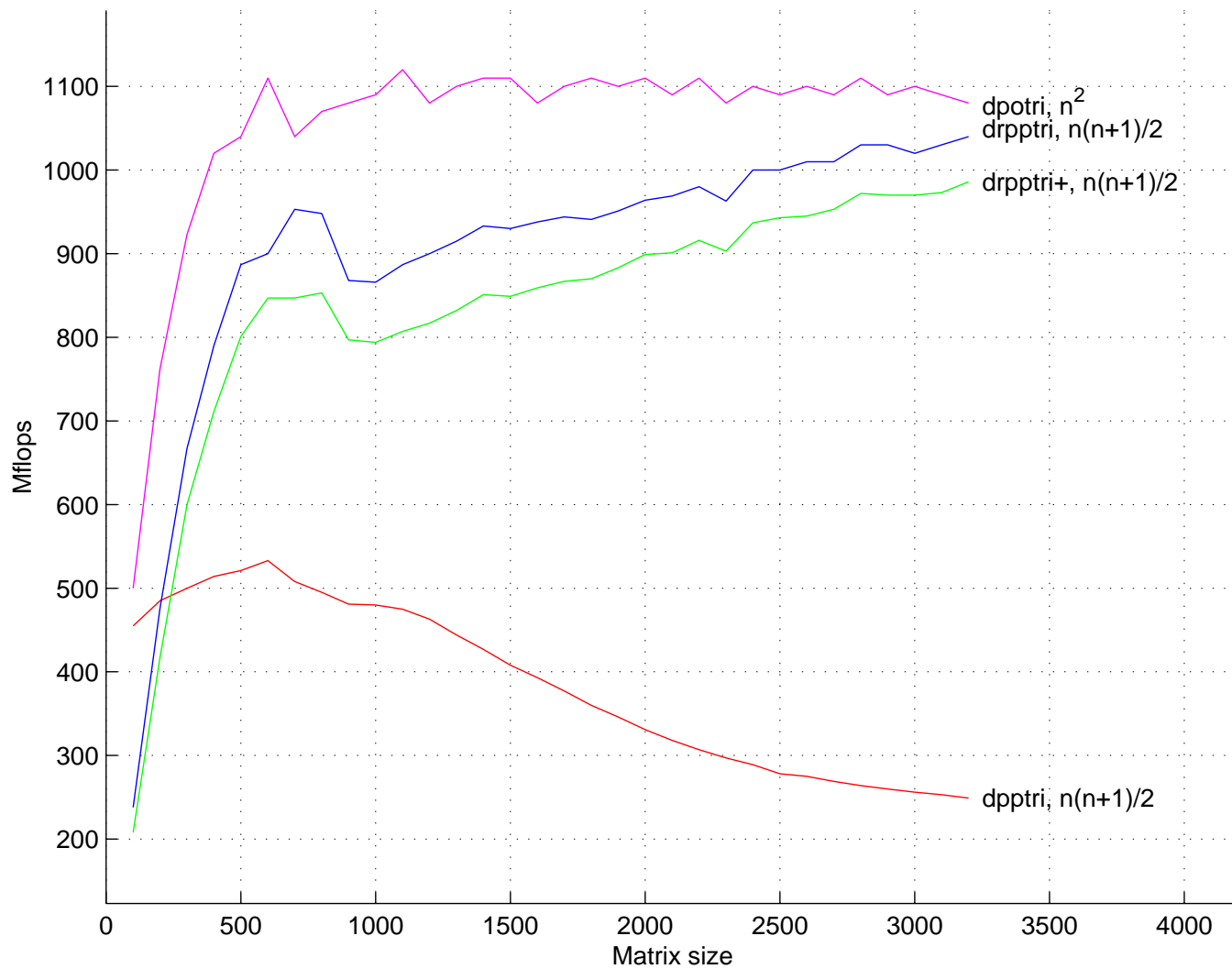
Factorization

Cholesky Algorithm, Factorizations, UPLO = L, IBM Power3 NH2 @ 375 MHz, ESSL



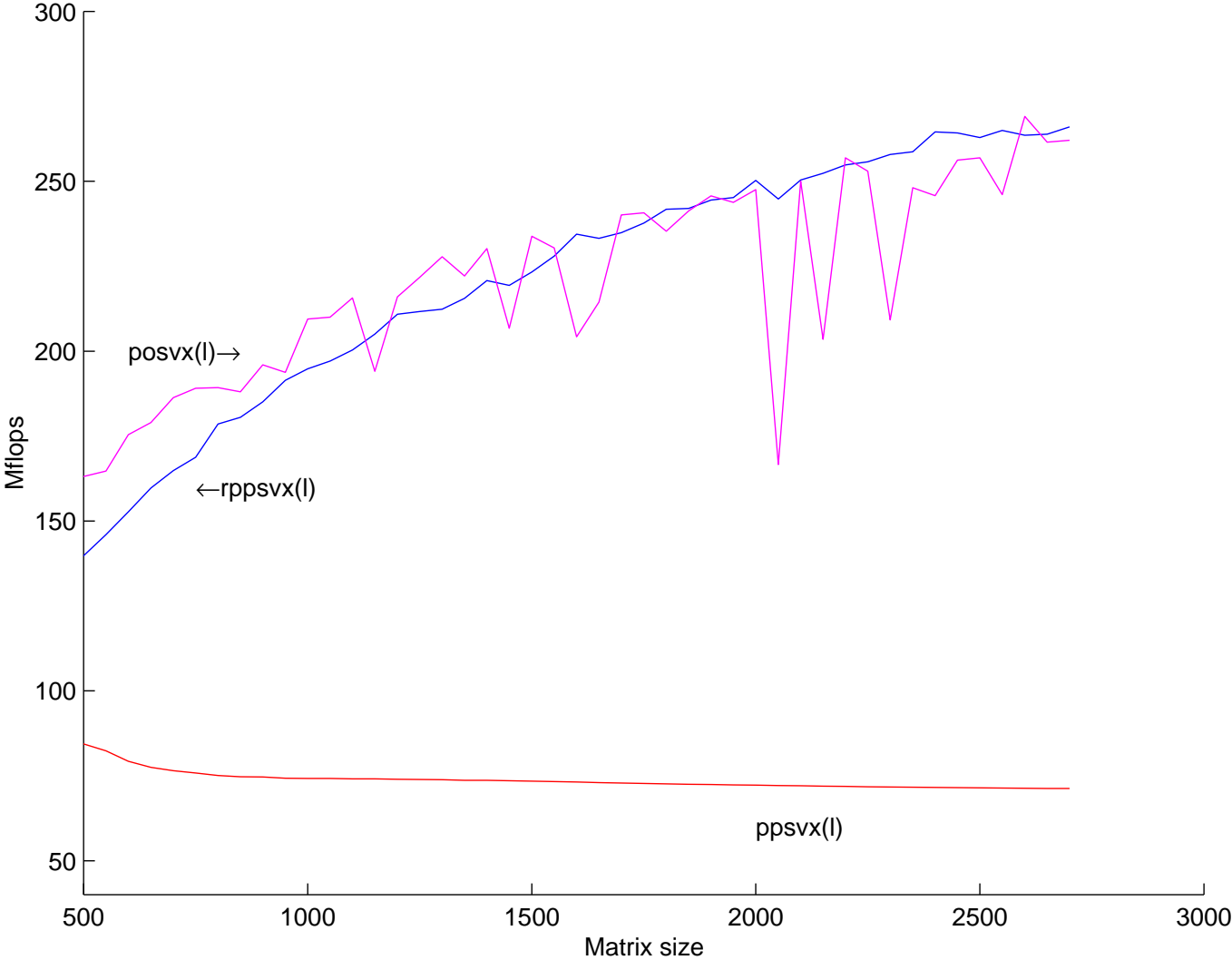
Inversion

Cholesky Algorithm, Inversion, UPLO = L, IBM Power3 NH2 @ 375 MHz, ESSL



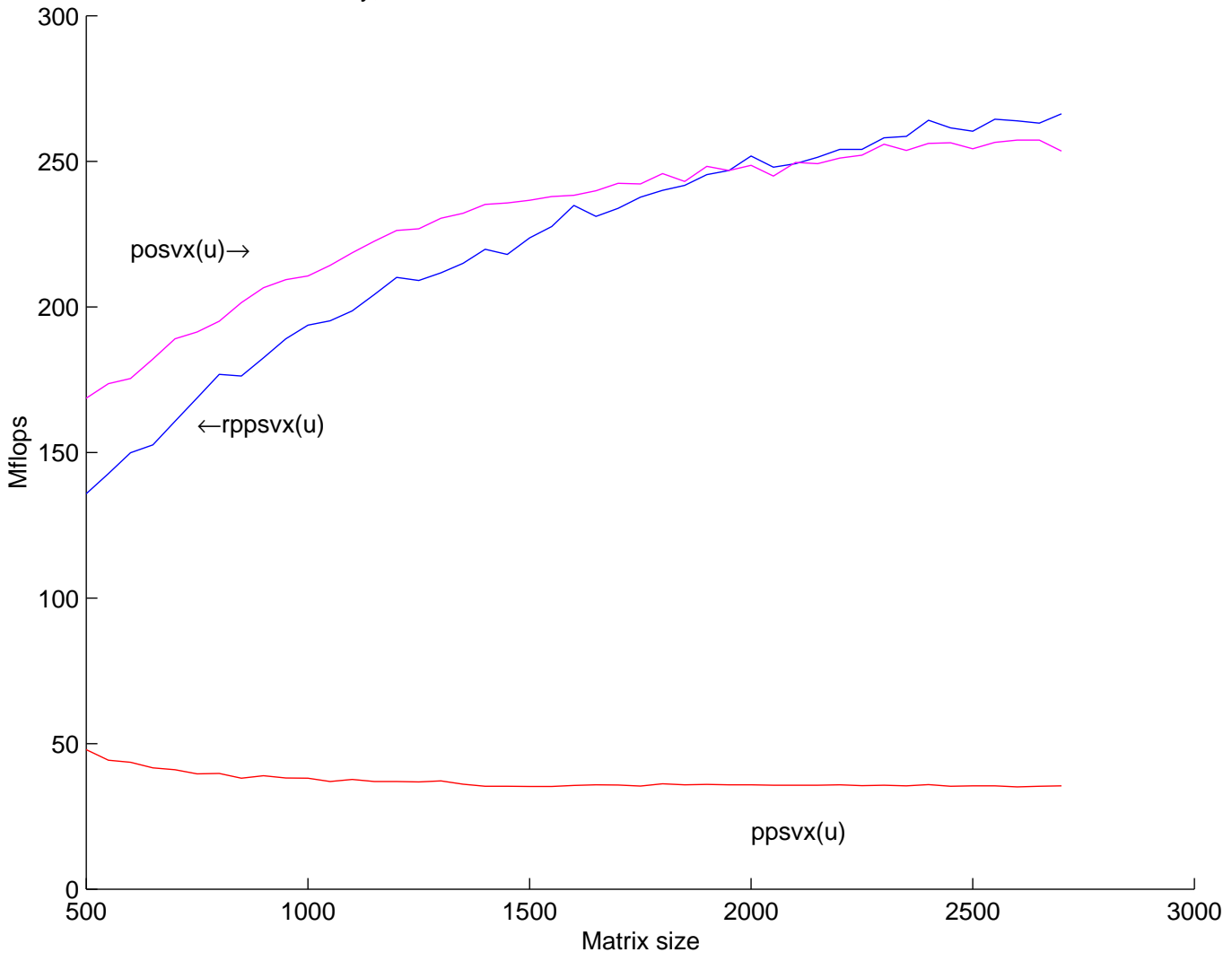
Cholesky

Cholesky, RPPSVX, UPLO = L, Intel Pentium III, @ 500 MHz, Atlas



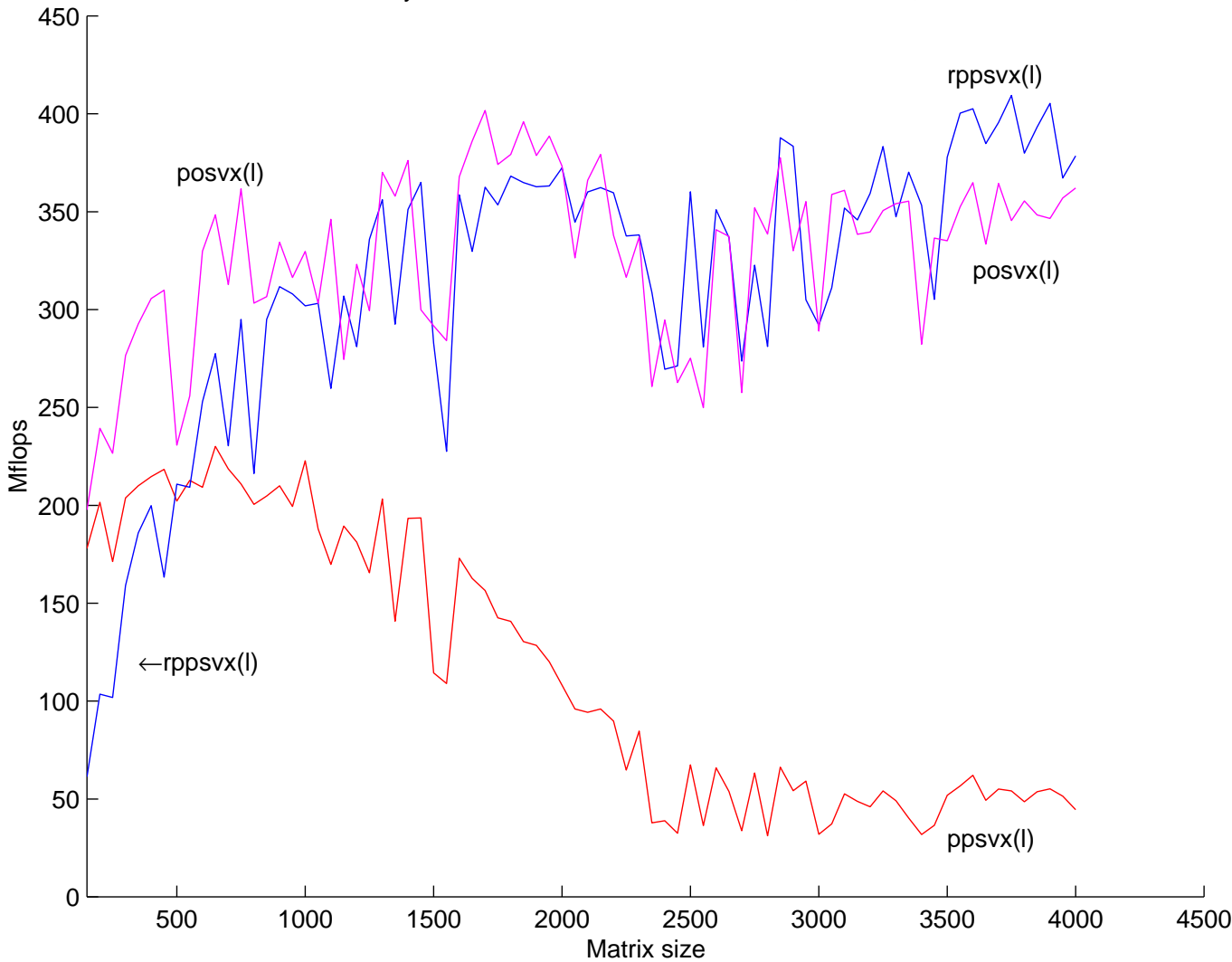
Cholesky

Cholesky, RPPSVX, UPLO = U, Intel Pentium III, @ 500 MHz, Atlas



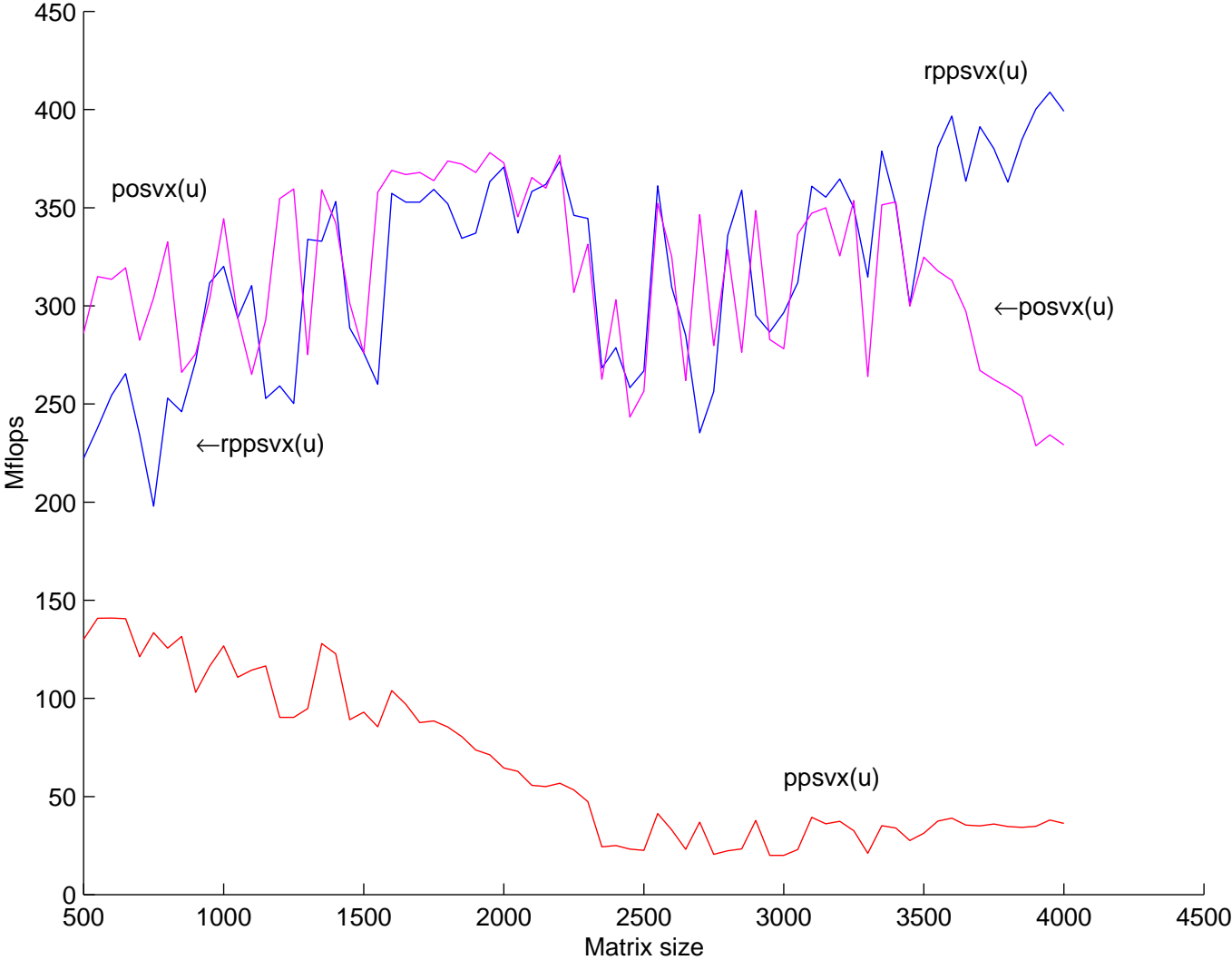
Cholesky

Cholesky, RPPSVX, UPLO = L, SGI R1200, @ ? MHz, Atlas



Cholesky

Cholesky, RPPSVX, UPLO = U, SGI R12000, @ ? MHz, Atlas



**New Data Storage Formats for Dense Matrices Lead to
Variety of High-Performance Algorithms**

The slides can be obtained from:

**<http://www.imm.dtu.dk/~jw/lectures/031202b.ps> or
<http://www.imm.dtu.dk/~jw/lectures/031202b.pdf>**

Jerzy Waśniewski

Emeritus Senior Research Professor

Department of Informatics & Mathematical Modeling

Technical University of Denmark

DTU, Bldg. 305, DK - 2800 Lyngby, Denmark

e-mail: jw@imm.dtu.dk

<http://www.imm.dtu.dk/~jw/JerzyWasniewski/>

November 27, 2003