Accelerated Methods for Performing the LDLT Decomposition

or:

Riding the edge of numerical stability

Peter Strazdins

Australian National University

e-mail: peter@cs.anu.edu.au

September 21, 1999

1 Introduction

- $N \times N$ symmetric indefinite ($x^T Ax \not> 0, ||x|| > 0$) systems of linear equations arise in:
  - incompressible flow computations, linear and non-linear optimization

  and also in electro-magnetic field analysis, eg. the AccuField application (Fujitsu):
  - dense, complex; up to order 50000; often weakly indefinite (= mostly large diagonals)
- stable algorithms exist that exploit symmetry to require only $\frac{N^3}{3} + O(N^2)$ FLOPS
  - LAPACK version of the Bunch-Kaufman algorithm is very competitive!
  - no previous Distributed Memory implementations yet known
- the Fujitsu AP3000 is an UltraSPARC-based DM multicomputer
  - 16 KB level-1 cache, 1 MB level-2 cache, 64-entry TLB
  - relatively high communication costs; a cluster-like machine
- wish to develop such a factorization routine for AccuField, primarily for the AP3000
  - the DBLAS (Distributed BLAS) Library is an essential component

2 Choice of Algorithms

- in the 1970's, 2 efficient and stable algorithms were proposed & refined:
  - tridiagonal reduction method (Aasen)
    - $A = LTL^T$, T is tridiagonal, L is N–$1 \times N$–1 lower triangular (unit diagonal)
    - exploits properties of Hessenberg matrices ($H = TL^T$)
    - for stability, need pivoting over the *sub-diagonal* portion of each column
  - diagonal pivoting method (Bunch-Kaufman)
    - $A = LDL^T$, D is diagonal with 1×1 or 2×2 sub-blocks, L is $N \times N$
    - in eliminating column j, 4 cases arise:

    D1: $|A_{j,j}| \geq \alpha |A_{i,j}|$, where $|A_{i,j}| = \max_{k=j+1}^{N-1} |A_{k,j}|$, $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.64$
    ie. a 1×1 pivot from $A_{j,j}$ is stable ((+ve) definite case; no interchange)

    D2: neither D3 nor D4 stable; effect as for D1

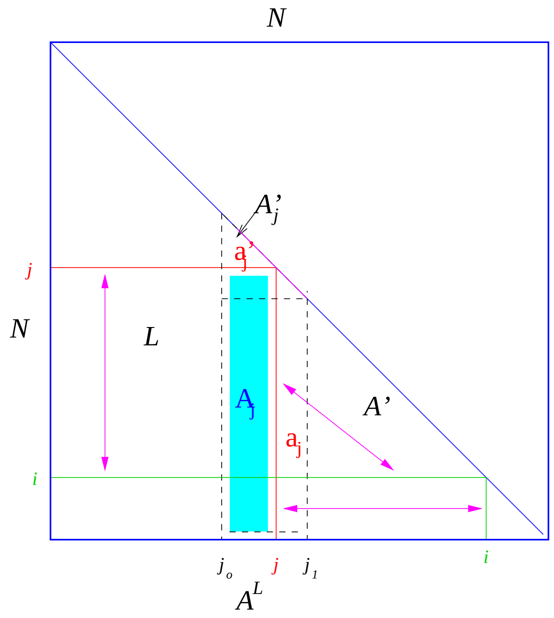
    D3: a 1×1 pivot from $A_{i,i}$ is stable (semi-definite case, interchange j with i)

    D4: a 2×2 pivot using $A_{j,j}$ & $A_{i,i}$ is stable (indefinite case, interchange $j + 1$ with i)

- the choice between them is primarily one of performance (on the target machine / matrices)!

3 Parallel Symmetric Interchanges

- symmetric $=$ *both* rows *and* columns i and j must be interchanged
  - ie. $A_{i,j}$ is not moved; $A_{j,j} \leftrightarrow A_{i,i}$
- store A in lower triangular half; overwrite with L and D:



  - relative communication volume is $2\times$ that of LU
  - greater startup costs:
    - requires 3 separate operations, one a transposition (a $P \times P$ grid is best)
    - value of i must be *immediately* broadcast to *all* cells!
    - row interchanges must be applied $2\times$ in the solve stage
- tridiagonal methods require an interchange on most columns (also $T^{-1}X$ must be serialized); B-K achieves case D1 $\approx 50\%$ (random matrices), $\geq 90\%$ (weakly indefinite)

4 The LAPACK Diagonal Pivoting Algorithm

create new matrix W aligned with A^L

for each j in current panel

$w_j \leftarrow (w'_j)^T A_j + a_j$

determine the pos. i of the max. in w_j

if D1 does not apply:

(partial) interchange $j+1$ with i

$w_{j+1} \leftarrow (w'_{j+1})^T A_j + a_{j+1}$; find max. of w_{j+1}

if D3 or D4 applies, complete interchange

if D4 applies: (2 × 2 pivot)

$$(a_j, a_{j+1}) \leftarrow (w_j, w_{j+1}) D_j^{-1} \quad , \; D_j = \begin{pmatrix} A_{j,j} & A_{j+1,j} \\ A_{j+1,j} & A_{j+1,j+1} \end{pmatrix}$$

skip column $j+1$

else

$a_j \leftarrow w_j / A_{j,j}$

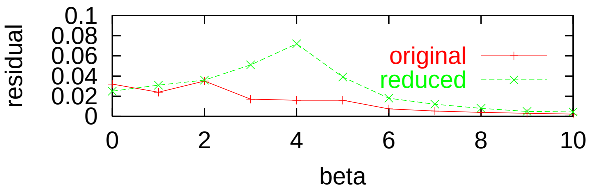
$A' \mathrel{-}= A^L W^T$ (matrix-matrix multiply, A' is triangular)

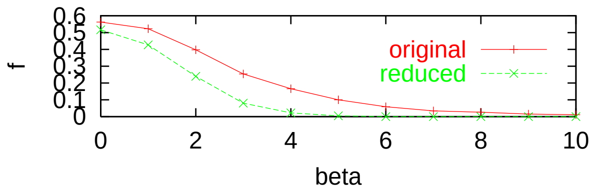
undo row swaps in A^L ($\Rightarrow$ L in 'standard' form)

5 Minimizing Symmetric Interchanges

- has potentially large gains || performance (also helps preserve band matrix structure)
- idea of Patrick/Jones (1991):
  - for a block of width p eliminated by D1, can use overall growth considerations: $\mu_k = \Pi_{i=0}^{k-1}(1 + \frac{|W_{i,i}|}{\lambda_i}) \leq (1 + \frac{1}{\alpha})^p$
- can be applied to (different) LAPACK version of B-K algorithm, but:
  - not using (the more conservative) *a priori* growth estimate
  - *always* achieves target blocking factor ω (performance, storage blocking)
  - the pivoting behavior (eg. fraction of cases D2–D4, f) is independent of ω

  by considering overall growth of last $p_{\max} > \omega$ columns
- av. residual and f for simulated 500×500 matrices (of form $A' + \beta I$, $A' \in \mathrm{random}[-1, 1]$)





6 Using Lookahead to Achieve 'True' Cholesky Speed

- LLT has several inherent performance advantages over LDLT
  - as matrices approach definiteness (eg. $f \to 0$), can we approach LLT speed?
- if we knew *in advance* columns $0 : j-1$ of A^L could be eliminated by D1, can eliminate via $A_j \leftarrow A_j (W'_j)^{-T}$ (fast, if $\omega^{\mathrm{p}} = j \approx \omega$)
- *a priori* estimate (Patrick/Jones 1991):

$$\mu_k^{\mathrm{p}} = \mu_{k-1}^{\mathrm{p}} \left(1 + \frac{\lambda_k^{\mathrm{p}}}{|W_{k,k}|}\right) \qquad , \lambda_k^{\mathrm{p}} = \lambda'_k + \Sigma_{i=0}^{k-1} |A_{k,i}| \lambda_i^{\mathrm{p}}$$

proved too conservative for large $\omega = 44$ (complex matrices, $N = 528$):

β	4	6	8	10	15	20	25	30
$\bar{\omega}^{\mathrm{p}}$	2	2	2	4	6	16	32	37
$\bar{\omega}^{\mathrm{p}\prime}$	10	13	18	27	37	37	40	40

- improved block size ($\bar{\omega}^{\mathrm{p}\prime}$) by recomputing actual growth by a 2nd maximum search (after elimination)

- solution: heuristic prediction (use λ'_k instead of λ_k^{p})
  - backtrack if later find an unstable elimination was made (rare, if f small)

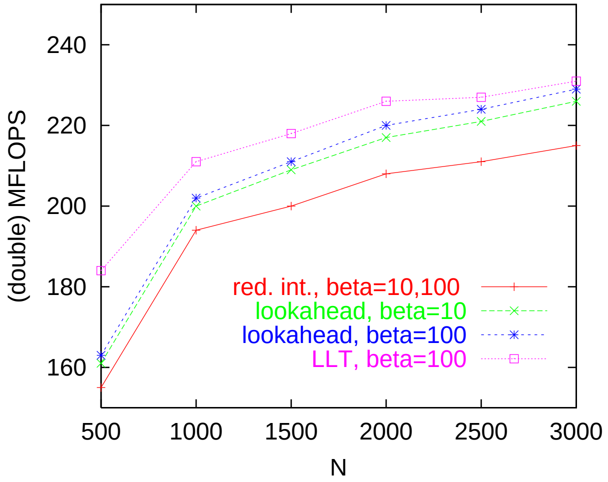
7 Serial Performance

- speed in (complex) MFLOPs on 300 MHz UltraSPARC II, AccuField matrices:

	zsytrf()	DZSYTRF()		
N		original	reduced interchange	lookahead
161	80	73	74	76
1601	96	108	109	111

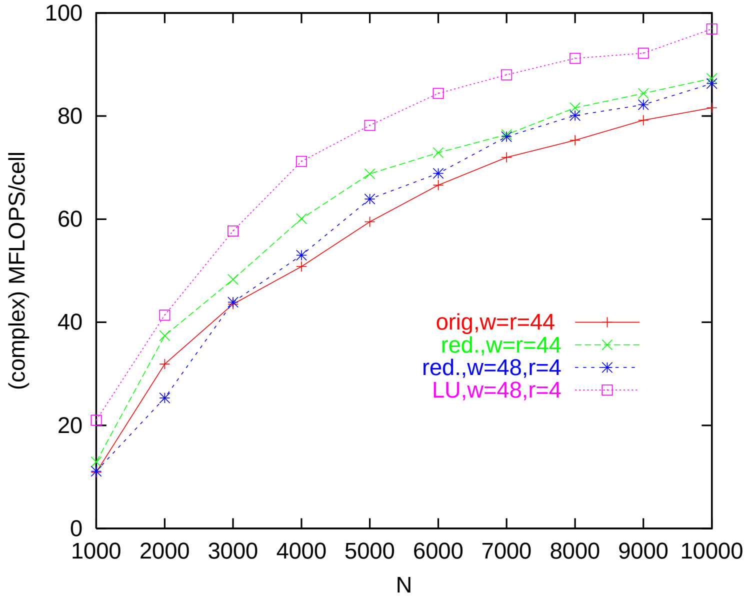
  - $N = 1601$ represents the asymptotic speed
  - an efficient (DBLAS) triangular matrix multiply made the most difference

- a clearer improvement for lookahead for double precision (larger $\omega = 64$, speed difference of $A_j(W_j')^{-T}$ & $(w_j')^T A_j$ larger):



8 Parallel Implementation and Performance

- reduced communication startups in many places, by coalescing various sub-computations
- routine can use an block-cyclic distibution block size r and any ω
- for simulated matrices with $\beta = 7$ on a 4×4 300 MHz AP3000:



- storage blocking ($\omega = r$) has unusual advantages:
  - larger messages $\Rightarrow$ higher bandwidth on the AP3000
  - algorithmic blocking ($\omega < r$) cant 'recycle' horiz. communication of A^L
  - actual panel width can be $\omega + 1 \Rightarrow$ makes harder to implement!

- LU speed probably represents an 'upper bound' for LDLT speed

9 Conclusions

- factoring symmetric indefinite matrices has a tradeoff in accuracy and performance
  - algorithms far more complex & harder to efficiently ||ize than for other matrix factorizations
  - on AP3000, blocking methods used had little performance impact
- reduction of symmetric interchanges is an important consideration for || implementations
  - resulted in some increase in residuals, although same growth bound achieved
- lookahead method complex to implement
  - required heuristic growth estimate, to be re-calculated after elimination
  - worthwhile serial performance gains, at least for double precision
  - parallel reswults will be interesting ...
  - still to be resolved: to decide when trying lookahead will be worthwhile