An Efficient And Stable Method for Parallel Factorization of Dense Symmetric Indefinite Matrices

Peter Strazdins* and John G. Lewis†

* Department of Computer Science, Australian National University

† Boeing Corporation

`http://cs.anu.edu.au/~Peter.Strazdins (/seminars)`

5th Int'l Conference on High-Performance Computing in the Asia-Pacific Region
25 September 2001

1 Talk Outline

- introduction to general dense symmetric systems and their factorization
- Diagonal Pivoting methods, illustrated by the Bunch-Kaufman algorithm
  - stability problems with the Bunch-Kaufman algorithm
  - alternatives:
    - the Bounded Bunch-Kaufman algorithm
      - Ashcraft, Grimes & Lewis, 1999
    - the exhaustive block search strategy
      - Ashcraft, Grimes & Lewis, 1999
      - modified from Duff & Reid, 1976 (sparse matrices)
    - outline of how to parallelize these
- evaluation of the pivoting behaviour of the three algorithms
- serial & parallel performance
- conclusions

2 Introduction to Symmetric Indefinite Systems Solution

- $N \times N$ general (ie. indefinite: $x^T Ax \not> 0, ||x|| > 0$) symmetric systems of linear equations, eg. $Ax = b$, arise in:
  - incompressible flow computations, linear and non-linear optimization, electromagnetic scattering & field analysis & data mining (HPCAsia'01 DM session)
- a direct solution for x is the most general and accurate method
  - require $O(N^3)$ FLOPS, dominated by the factorization of A, ie. $A \rightarrow PLDL^T P^T$
  - iterative solutions require only $O(N^2)$ FLOPS, but dense systems may require a direct solution as a 'preconditioner'
- stable algorithms exist that exploit symmetry to require only $\frac{N^3}{3} + O(N^2)$ FLOPS in the factorization:
  - LAPACK version of the Bunch-Kaufman algorithm is very competitive!
  - variants shown to have an especially efficient parallelization (Strazdins, 1999)
  - stability issues of B-K recently challenged ('blew up' on a Boeing application, mid 90's) (Ashcraft, Grimes & Lewis, 1999)

Diagonal Pivoting Methods For Matrix Factorization

- the diagonal pivoting method use symmetric (row & column) interchanges based on 1×1 or 2×2 'pivots'
  - nb. interchange $i \leftrightarrow j$: $A_{i,j}$ is not moved; $A_{j,j} \leftrightarrow A_{i,i}$
- eg. the Bunch-Kaufman algorithm (1975):
  - $A = PLDL^T P^T$, P is a permutation matrix, D is a diagonal matrix with 1×1 or 2×2 sub-blocks, L is an $N \times N$ lower triangular matrix with $L_{i,i} = 1$
  - in eliminating column j, 4 cases arise:

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

  D2: neither D3 nor D4 stable; effect as for D1 (no interchange)

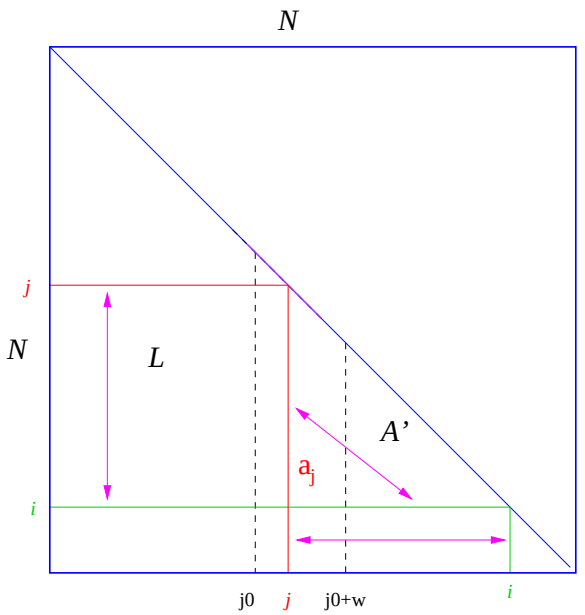
  D3: a 1×1 pivot from $A_{i,i}$ is stable ($j \leftrightarrow i$)

  D4: $|A_{j,j}|\gamma_i < \alpha \gamma_j^2$: a 2×2 pivot using $A_{j,j}$, $A_{i,j}$ & $A_{i,i}$ is stable ($j+1 \leftrightarrow i$)

- symmetric interchanges have high || overheads; B-K minimizes these
  - also finds a pivot using at most two column searches!

4 Bunch-Kaufman Algorithm & Stability Issues

- store A in lower triangular half; overwrite with L and D
- unblocked algorithm at step j:



find γ_j, decide D1–D4 (may also require γ_i)
if D4 does not apply
  if D3, interchange $j \leftrightarrow i$
  $w_j \leftarrow a_j; \quad a_j \leftarrow a_j / A_{j,j}$
  $A' \leftarrow A' - a_j w_j^T$
else
  interchange $(j+1) \leftrightarrow i$
  $w_j \leftarrow a_{j:j+1}; \quad a_{j:j+1} \leftarrow a_{j:j+1} D_j^{-1}$
  $A' \leftarrow A' - a_{j:j+1} w_j^T$

- stability requirement: at any step, size of (largest) matrix element should grow by only a constant amount (ideally < 2)
  - original proof only showed this applied to A', *not* to a_j:
    - this can grow by a unbounded factor of $\frac{\gamma_i}{\gamma_j}$ in cases D2 & D4

Alternative: the Bounded Bunch-Kaufman Algorithm

- idea: don't use case D2; only apply D4 if $\frac{\gamma_i}{\gamma_j} = 1$
  - ie. $A_{i,j}$ is *both* a row and column maximum
- if $\gamma_i \neq \gamma_j$, and neither D1 or D3 applies,
  interchange $i \leftrightarrow j$ and then repeat search at position j
  - this will terminate within $N - j$ steps, because γ_i must monotonically increase!
  - in practice, terminates in ≈ 2.5 steps on average
- there can be $O(N)$ interchanges at position j: problem in representing P
  - solution: while searching, perform only 'half-interchanges' ($j \leftarrow i$); record only final interchanges $j \leftrightarrow i'$ and $j + 1 \leftrightarrow i$
- extra searches and interchanges $\Rightarrow$ extra (esp. ||) overheads in B-B-K
  - partially alleviated by half-interchanges

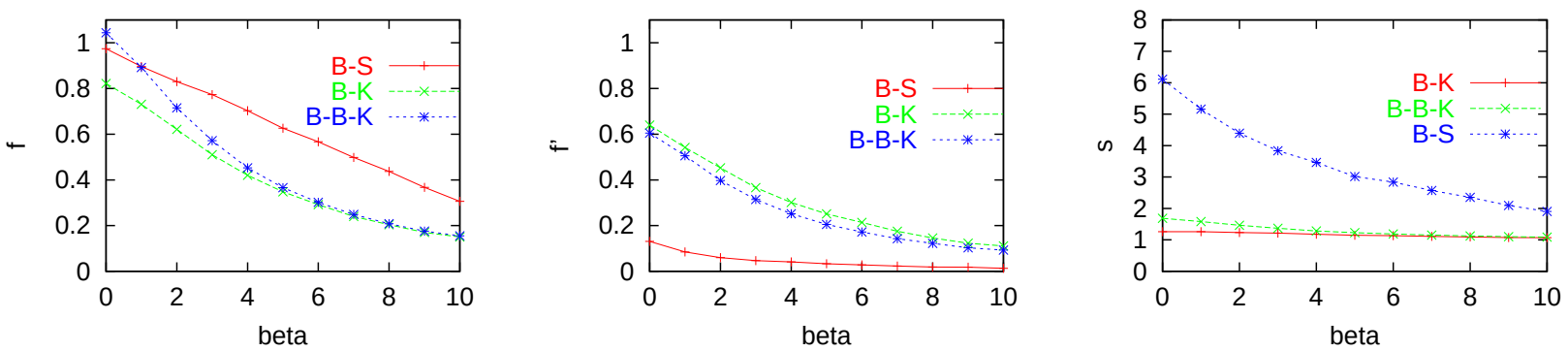
  but otherwise as fast as original B-K

6 Alternative: the Exhaustive Block Search Strategy

- generally, factor via a blocked algorithm, eliminating a block of $\omega > 1$ columns at each major step (for performance)
  - must bring 'up-to-date' columns j, i & i' to calc. γ_j, γ_i & γ'_i (avg. $\frac{N\omega}{2}$ FLOPS each)
  - advantages in finding current pivots i and i' within block (incl. avoid wasted FLOPS)
  - sparse: using multifrontal methods, preserves sparsity: a big payoff!
  - || dense: if $\omega =$ the storage block size, no || overheads in the interchanges!
- case D4 is stable if $A_{j,j}$, $A_{i,j}$ & $A_{i,j}$ are relatively large compared with γ_i & γ_i
- issues for adaptation to the || dense case:
  - a stable pivot in remainder of block may not exist; use B-B-K as a fall-back
  - must avoid 'cascades' of case D3, as $i \leftrightarrow j$ is now 'short-range'
  - finding many γ_i's is expensive:
    - can amortize communication overheads by finding $\gamma_{i:i+b-1}$ in one go
    - can reduce costs by limiting searches to first $\omega_s \leq \omega/2$ columns
      - ω_s columns can be updated as a 'block': significantly faster

Pivoting Behaviors of the three algorithms

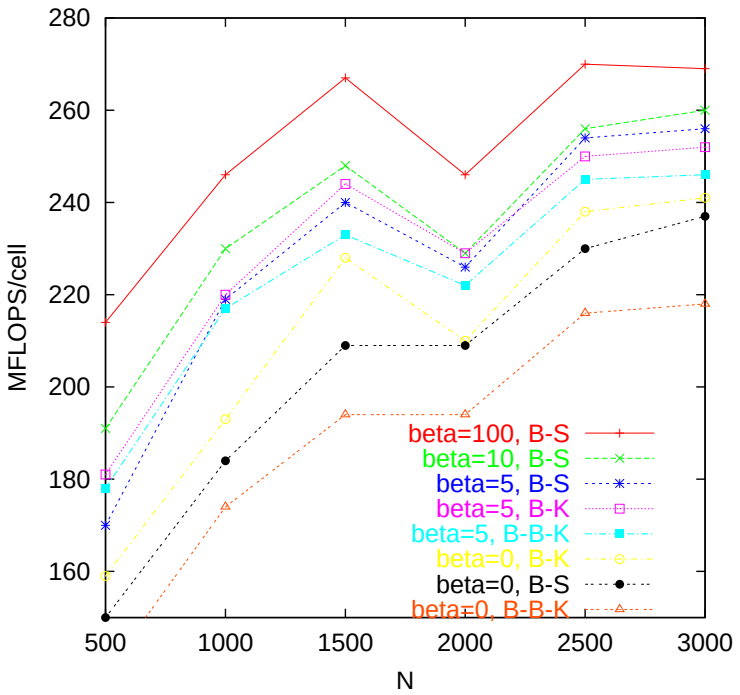
- use matrices of form $A' + \beta I$, $A' \in \text{random}[-1, 1]$ ($\beta \geq 0$ determines degree of 'definiteness')
- useful to define 3 quantities related to pivot choice and overhead:
  - f: number of interchanges (+ half-interchanges $\times 0.5$) $(/N)$ (corresp. cases D3 & D4)
  - f': like f but only for interchanges outside the current storage block (|| overhead)
  - s: number of γ_i's & γ_j's computed $(/N)$
- plots for $N = 1000$ and $\omega = 64$ (B-S is block search with $\omega_s = 16$)



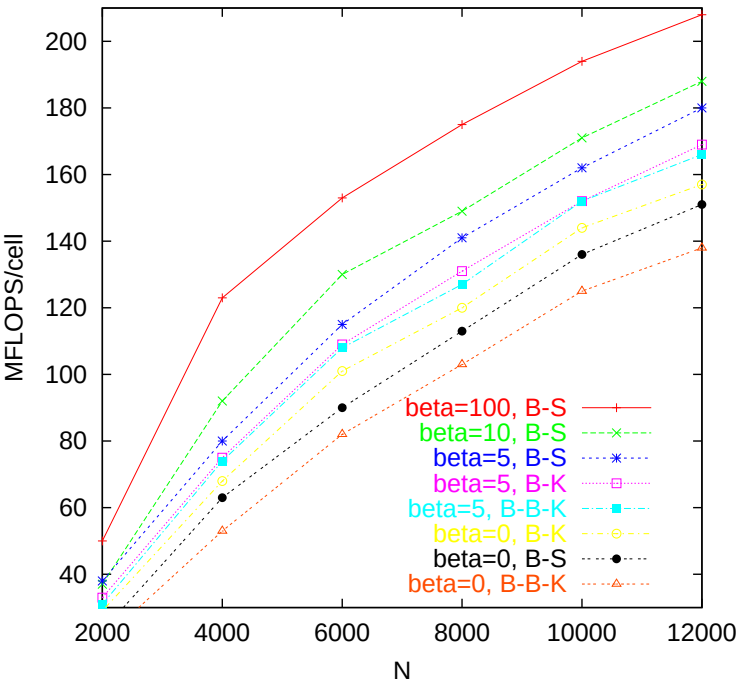
- for B-S: low $f' \Rightarrow$ block search normally succeeds; large s is a concern: are the overheads too high? why f is higher than for B-B-K is unclear...

8 Performance

- on an AP3000 with 200MHz UltraSPARC IIs (double precision, $\omega = 64$, $\omega_s = 16$)



(a) 1×1 AP3000



(b) 4×4 AP3000

- $\beta = 0$: B-K $\approx 10\%$ faster than B-S (in turn $\approx 10\%$ faster than B-B-K)
- $\beta = 5$: now B-S $\approx 6\%$ faster than B-K, in turn faster than B-B-K
- $\beta \geq 10$: all 3 have the same performance (and faster than for $\beta \leq 5$)

9 Conclusions

- solving general symmetric systems has an accuracy–performance tradeoff
  - more accurate B-B-K is slightly slower than B-K
  - but both ||ize *reasonably* efficiently
- reduction of overhead of || symmetric interchanges is important
- augmenting B-B-K with a exhaustive block search reduced these successfully
  - can also avoid $O(N^2\omega)$ redundant FLOPS from failed column searches
  - required many optimizations, implementation is very complex
  - has still considerable overheads:
    - more memory movements (higher f), large number of searches (high s)
  - future work: can reduce latter by:
    - predict which columns i very likely or unlikely to form stable pivots

    locality of pivoting especially suitable for banded and out-of-core solvers
- high serial and parallel performance can be achieved for sufficiently large general linear systems, at an acceptable level of accuracy