Recursive Blocked Algorithms and Hybrid Data Structures for Dense Matrix Computations

Bo Kågström
Dept of Computing Science & HPC2N
Umeå University, Sweden

4th China-Sweden Workshop on Computational Mathematics, June 6-9, 2004, Beijing, P.R. China

HPC2N - “HPC to North”

National center for Scientific and Parallel Computing

New super cluster (installed 2004-06-07):
- 392 proc (64 bit, AMD Opteron)
- 1.5 TB memory
- Myrinet
- ~ 1.3 Tflops/s HP-Linpack
- Most powerful computer in Sweden
- Funded by the Wallenberg Foundation (KAW)

Funded by the Swedish Research Council and its meta-center SNIC
Matrix Computations

- Fundamental and ubiquitous in computational science and its vast application areas
- Library software - optimized building blocks for fundamental operations
  - BLAS, (Sca)LAPACK, SLICOT (see also NETLIB)
  - ESSL and other vendors
  - Portability and efficiency
- Continuing demand for new and improved algorithms and software along with the computer evolution

“Data transport” in memory hierarchies

- of today’s computer systems
- PC - cluster - supercomputer

Small, Fast, Expensive

Large, Slow, less Expensive
Management of deep memory hierarchies

- **Architecture evolution**: HPC systems with multiple SMP nodes, several levels of caches, more functional units per CPU
- **Key to performance**: understand the algorithm and architecture interaction
- **Hierarchical blocking**

The fundamental AHC triangle
Outline

- Hierarchical blocking: motivation and implications
- Recursive blocked templates and algorithms
- Recursive blocked data structures
- Case studies:
  - General matrix multiply and add (GEMM)
  - Packed Cholesky factorization
  - QR factorization and linear systems
  - Triangular matrix equations and condition estimation
- Some related and complementary work
- Work in progress: periodic matrix equations
- Concluding remarks

Blocking for a memory hierarchy

Explicit multi-level blocking
Recursion leads to automatic variable blocking

- Fits low level in memory hierarchy
- Fits high level in memory hierarchy

Recursive blocking

Stopping criteria controlled by parameter (blksz).

Splittings defining independent and dependent tasks

Critical path of subtasks: (1), (2), (3)
TRSM Operation: $AX = C$, $A$ mxm upper triangular, $C/X$ mxn

$A \begin{bmatrix} X_1 & X_2 \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$

$AX_1 = C_1,$
$AX_2 = C_2.$

Case Study 1

General matrix multiply and add (GEMM)
Recursive splittings for GEMM:

\[ C \leftarrow \beta \text{op}(C) + \alpha \text{op}(A) \text{op}(B) \]

### Splitting by breadth or by depth

**Split**

- **m, n, k**
  \[ \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \]

- **m**
  \[ \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \]

- **n**
  \[ \begin{bmatrix} C_{11} \\ C_{21} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{12} \end{bmatrix} = \]

- **k**
  \[ \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{21} & B_{22} \end{bmatrix} = \]

### Recursive splitting - by breadth or by depth
GEMM recursive blocked template - splitting by depth

- \( C = \text{rgemm}( A, B, C, \text{blksz}) \)
  - If \( m, n, \) and \( k \leq \text{blksz} \)
    - \( C = \text{opt}_\text{gemm}( A, B, C) \) % optimized GEMM kernel!

elseif \( m = \max(m, n, k) \) % split \( m: m2 = m/2 \)
  - \( C(1:m2, :) = \text{rgemm}( A(1:m2, :) , B, C(1:m2, :) , \text{blksz}) \)
  - \( C(m2+1:m, :) = \text{rgemm}( A(m2+1:m, :) , B, C(m2+1:m, :) , \text{blksz}) \)

elseif \( n = \max(n,k) \) % split \( n: n2 = n/2 \), \( k \)
  - \( C(:,1:n2) = \text{rgemm}( A, B(:,1:n2), C(:,1:n2), \text{blksz}) \)
  - \( C(:,n2+1:n) = \text{rgemm}( A, B(:,n2+1:n), C(:,n2+1:n), \text{blksz}) \)

else % split \( k: k2 = k/2 \),
  - \( C = \text{rgemm}(A(:,1:n2), B(1:m2, :) , C, \text{blksz}) \)
  - \( C = \text{rgemm}(A(:,n2+1:n), B(m2+1:m, :) , C, \text{blksz}) \)

When to end the recursive splitting?

Locality of reference

- Recursive blocked algorithms mainly improve on the temporal locality
- Further performance improvements by matching the data structure with the algorithm (and vice versa)
- Recursive blocked data structures improve on the spatial locality
**Blocked data formats**

Blocks $A_{ij}$ of size $mb \times nb$ can be ordered in $(pq)!$ different ways.

**Recursive blocked row format**

Recursive ordering: a 1-dim tour through a 2-dim object (Hilbert space filling heuristics)

RBR $\leftrightarrow$ Z-Morton ordering
Recursive blocked column format

\[
\begin{bmatrix}
A_{11} & A_{12} & \cdots & A_{18} \\
A_{21} & \cdots & \cdots & A_{28} \\
\vdots & \vdots & \ddots & \vdots \\
A_{q1} & A_{q2} & \cdots & A_{q8}
\end{bmatrix}
\]

\(q = 8\)

\(p = 8\)

RBC <-> reflected-N-Morton space filling ordering

Triangular recursive data format

\[
\begin{bmatrix}
A_{11} \\
A_{21} \\
\vdots \\
A_{41} & A_{42} & \cdots & A_{44}
\end{bmatrix}
\]

\[
\begin{bmatrix}
A_{11} & A_{21} & \cdots & A_{41} \\
A_{21} & \cdots & \cdots & A_{42} \\
\vdots & \vdots & \ddots & \vdots \\
A_{41} & A_{42} & \cdots & A_{44}
\end{bmatrix}
\]

Bo Kågström 2004
Recursive GEMM: multi-level vs. recursive blocking

IBM PPC604, 112 MHz

Recursive blocked GEMM and SMP parallelism via threads

IBM PPC604, 4 proc
Recursion template for one-sided matrix factorization

1. Partition
2. Factor left hand side
3. Update right hand side
4. Factor right hand side

Case Study 2

Cholesky factorization for matrices in packed format
Packed Cholesky factorization

\[ A \equiv \begin{bmatrix} A_{11} & A_{12}^T \\ A_{21} & A_{22} \end{bmatrix} = LL^T \equiv \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \]

Standard approach (typified by LAPACK):
- Packed storage -> cannot use standard level 3 BLAS (e.g., DGEMM)
- Possible to produce packed level 3 BLAS routines at a great programming cost
- Run at level 2 performance, i.e., much below full storage routines.
- Minimum storage: \( 1/2n(n+1) \) elements

Packed recursive blocked data

- Divide into two isosceles triangles \( T_1, T_2 \) and rectangle \( R \)
- Divide triangles recursively down to element level
- Store in order: \( T_1, R, T_2 \)
- Rectangles stored in full format

Possible to use full storage level 3 BLAS

---

Fig. 3.1. Memory indices for \( 7 \times 7 \) upper triangular matrix stored in traditional packed format and recursive packed format.
**Cholesky recursive blocked template**

\[
A = \begin{pmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{pmatrix} = LL^T = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22} \end{pmatrix}
\]  

(1)

Factor: \( A_{11} = L_{11}L_{11}^T \).  

(2)

TRSM: \( L_{21}L_{11}^T = A_{21} \).  

(3)

SYRK: \( \tilde{A}_{22} = A_{22} - L_{21}L_{21}^T \)  

(4)

Factor: \( \tilde{A}_{22} = L_{22}L_{22}^T \)  

(5)

---

**TRSM recursive blocked template**

\[
X \begin{pmatrix} A_{11} & A_{21}^T \\ 0 & A_{22} \end{pmatrix} = B \quad \text{or} \quad m \left\{ \begin{pmatrix} \tilde{X}_1 & \tilde{X}_2 \end{pmatrix} \begin{pmatrix} A_{11}^T & A_{21}^T \\ 0 & A_{22} \end{pmatrix} = m \left\{ \begin{pmatrix} \tilde{B}_1 & \tilde{B}_2 \end{pmatrix} = B \right. 
\]

(6)

If we break Equation (6) into its component pieces we get

TRSM: \( X_1A_{11}^T = B_1 \)  

(7)

GEMM: \( \tilde{B}_2 = B_2 - X_1A_{21}^T \)  

(8)

TRSM: \( X_2A_{22}^T = \tilde{B}_2 \)  

(9)
Packed recursive blocked Cholesky highlights

- Recursive blocked algorithm + recursive packed data layout \Rightarrow can make use of high performance level 3 BLAS routines (e.g., DGEMM)
- Use minimal storage for matrix $A$
- Temporary workspace $= 1/8n^2$ elements ($\sim 25\%$)
- Leaf problems ($< \text{blksz}$) are solved using superscalar kernels (Cholesky, TRSM, SYRK)

Recursive blocked Cholesky vs. LAPACK - (rec.) packed format

Runs at level 3 performance - at least!
Case Study 3

QR factorization and linear systems

1. Divide $A_{mxn}$ in two parts (left & right)

2. Factorize left hand side by a recursive call

3. Update right hand side

4. Factorize by a recursive call

Stopping criteria: if $n < 4$ use standard algorithm
Aggregating $Q = I - YTY^T$

Given $Q_1 = I - \tau_1v_1v_1^T$ and $Q_2 = I - \tau_2v_2v_2^T$, then
\[ T = \begin{pmatrix} \tau_1 & -\tau_1v_1^Tv_2 \\ 0 & \tau_2 \end{pmatrix} \quad \text{and} \quad Y = (v_1 \quad v_2) \]

Two elementary transformations

Given $Q_1 = I - Y_1T_1Y_1^T$ and $Q_2 = I - \tau_2v_2v_2^T$, then
\[ T = \begin{pmatrix} T_1 & -T_1Y_1^Tv_2 \\ 0 & \tau_2 \end{pmatrix} \quad \text{and} \quad Y = (Y_1 \quad v_2) \]

One block and one elementary transformation

Column by column using Level 2 operations

Given $Q_1 = I - Y_1T_1Y_1^T$ and $Q_2 = I - Y_2T_2Y_2^T$, then
\[ T = \begin{pmatrix} T_1 & -T_1Y_1^TY_2 \\ 0 & T_2 \end{pmatrix} \quad \text{and} \quad Y = (Y_1 \quad Y_2) \]

Two block transformations

Recursively, block by block using Level 3 operations

Recursive blocked QR highlights

- Recursive splitting controlled by $nb$ (splitting point = $\min(nb, n/2)$, $nb = 32 - 64$)
- Level 3 algorithm for generating $Q = I - YTY^T$ (compact WY) within the recursive blocked algorithm ($T$ triangular of size $\leq nb$)
- Replaces LAPACK level 2 and 3 algorithms
Recursive QR vs. LAPACK

Fig. 4.1 Performance results in Mflops/s for square matrices (left) and performance ratio for tall, thin matrices (right) for the recursive algorithm RGEQRF and DGEQRF of LAPACK on the 200 MHz IBM Power3.

Least squares recursive algorithm

\[ X = \text{RGELS}(A, B, nb) \]

If \( n \leq nb \)

1. Factor \( A = Q \left[ \begin{array}{c} R_1 \\ \end{array} \right] ; \quad \hat{B} \leftarrow Q^T B; \quad \text{solve} \quad RX = \hat{B}(1: n,:) \)

else

2. Set \( A = \left[ \begin{array}{cc} A_1 & A_2 \end{array} \right] ; \quad B = \left[ \begin{array}{c} B_1 \\ B_2 \end{array} \right] \) with \( nb \) cols in \( A_1 \), \( nb \) rows in \( B_1 \)

3. Factor \( A_1 = Q_1 \left[ \begin{array}{c} R_{11} \\ 0 \end{array} \right] \)

4. Set \( \left[ \begin{array}{cc} R_{12} & B_1 \\ A_{22} & B_2 \end{array} \right] \leftarrow Q_1^T \left[ \begin{array}{cc} A_2 & B \end{array} \right] \)

5. \( \hat{X}_2 = \text{RGELS}(A_{22}, \hat{B}_2, nb) \)

6. Solve \( R_{11} X_1 = \hat{B}_1 - R_{12} \hat{X}_2 \); return \( X = \left[ \begin{array}{c} X_1 \\ X_2 \end{array} \right] \)

Fig. 4.2 Recursive least squares RGELS algorithm for computing the solution to \( AX = B \), where \( A \) is \( m \times n \) (\( m \geq n \)).
Recursive linear systems solvers

Solve $\text{op}(A)X = B$, $A_{m \times n}$ - full row (or column) rank (compare LAPACK DGELS):

1. linear least squares solution to $\min ||AX - B||_F (m \geq n)$;
2. linear least squares solution to $\min ||A^T X - B||_F (m < n)$;
3. minimum norm solution to $\min ||A^T X - B||_F (m \geq n)$;
4. minimum norm solution to $\min ||AX - B||_F (m < n)$.

- RGELS solves P1
- P2 solved as P1 after explicit transposition
- RGELS-like algorithm solves P3
- P4 solved as P3 after explicit transposition

Case Study 4

Triangular matrix equations and condition estimation
Matrix equations

<table>
<thead>
<tr>
<th>Name</th>
<th>Matrix equation</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Sylvester (CT)</td>
<td>$AX - XB = C$</td>
<td>SYCT</td>
</tr>
<tr>
<td>Standard Lyapunov (CT)</td>
<td>$AX + XA^T = C$</td>
<td>LYCT</td>
</tr>
<tr>
<td>Generalized coupled Sylvester</td>
<td>$(AX - YB, DX - YE) = (C, F)$</td>
<td>GCSY</td>
</tr>
<tr>
<td>Standard Sylvester (DT)</td>
<td>$AXB^T - X = C$</td>
<td>SYDT</td>
</tr>
<tr>
<td>Standard Lyapunov (DT)</td>
<td>$AXA^T - X = C$</td>
<td>LYDT</td>
</tr>
<tr>
<td>Generalized Sylvester</td>
<td>$AXB^T - CXD^T = E$</td>
<td>GSYL</td>
</tr>
<tr>
<td>Generalized Lyapunov (CT)</td>
<td>$AXE^T + EXA^T = C$</td>
<td>GLYCT</td>
</tr>
<tr>
<td>Generalized Lyapunov (DT)</td>
<td>$AXA^T - EXE^T = C$</td>
<td>GLYDT</td>
</tr>
</tbody>
</table>

One-sided (top) and two-sided (bottom)

Block diagonalization and spectral projectors

$S$ block-diagonalized by similarity:

$$
\begin{bmatrix}
I_m & -R \\
0 & I_n
\end{bmatrix}
S
\begin{bmatrix}
I_m & R \\
0 & I_n
\end{bmatrix} =
\begin{bmatrix}
A & 0 \\
0 & B
\end{bmatrix}
S =
\begin{bmatrix}
A & -C \\
0 & B
\end{bmatrix}
$$

where $R$ satisfies $AR - RB = C$

Spectral projector associated with $(1,1)$-block:

$$
P =
\begin{bmatrix}
I_m & R \\
0 & 0
\end{bmatrix}
$$

$$
\|P\|_2 = (1 + \|R\|_2^2)^{1/2}
$$

Computed estimate:

$$
s = \frac{1}{\|P\|_F}
$$
Separation of two matrices

\[
\text{Sep}[A, B] = \inf_{\|X\|_F=1} \|AX - XB\|_F = \sigma_{\min}(Z),
\]

where \( Z = I_n \otimes A - B^T \otimes I_m. \)

Computing \( \text{Sep}[A,B] \) costs \( O(m^3n^3) \) - impractical!

Reliable \( \text{Sep-estimates of cost } O(m^2n + mn^2): \)

\[
\frac{\|x\|_2}{\|y\|_2} \leq \frac{\|X\|_F}{\|C\|_F} \leq \|Z^{-1}\|_2 = \frac{1}{\sigma_{\min}(Z)} = \text{Sep}^{-1},
\]

\[
(mn)^{-1/2}\|Z^{-1}\|_1 \leq \|Z^{-1}\|_2 \leq \sqrt{mn}\|Z^{-1}\|_1.
\]

Matrix equation Sep-functions

<table>
<thead>
<tr>
<th>( Z )-matrix</th>
<th>Sep-function = ( \sigma_{\min}(Z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Z_{\text{SYCT}} = I_n \otimes A - B^T \otimes I_m )</td>
<td>( \inf_{|X|_F=1} |AX - XB|_F )</td>
</tr>
<tr>
<td>( Z_{\text{LYCT}} = I_n \otimes A + A \otimes I_n )</td>
<td>( \inf_{|X|_F=1} |AX - X(A^T)|_F )</td>
</tr>
<tr>
<td>( Z_{\text{GCSY}} = \begin{bmatrix} I_n \otimes A &amp; -B^T \otimes I_m \ I_n \otimes D &amp; -E^T \otimes I_m \end{bmatrix} )</td>
<td>( \inf_{|X,Y|_F=1} |(AX - YB, DX - YE)|_F )</td>
</tr>
</tbody>
</table>

\( Z \times = b, Z \) is a Kronecker product representation

Sep-function = smallest singular value of \( Z \)
Recursive blocked SYCT template

**Case 1:** $1 \leq n \leq m/2$

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{22}
\end{bmatrix}
\begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
- \begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{22}
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

**Case 2:** $1 \leq m \leq n/2$

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{22}
\end{bmatrix}
\begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
- \begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{22}
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

**Case 3:** $n/2 < m < 2n$

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{22}
\end{bmatrix}
\begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
- \begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{22}
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

---

Recursive SYCT - Case 3

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{22}
\end{bmatrix}
\begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
- \begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
\begin{bmatrix}
B_{11} & B_{12} \\
B_{22}
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]

\[
A_{11}X_{11} - X_{11}B_{11} = C_{11} - A_{12}X_{21}
\]
\[
A_{11}X_{12} - X_{12}B_{22} = C_{12} - A_{12}X_{22} + X_{11}B_{12}
\]
\[
A_{22}X_{21} - X_{21}B_{11} = C_{21}
\]
\[
A_{22}X_{22} - X_{22}B_{22} = C_{22} + X_{21}B_{12}
\]
Recursive SYCT - Case 3

\[
\begin{align*}
A_{11}X_{11} - X_{11}B_{11} &= C_{11} - A_{12}X_{21} \\
A_{11}X_{12} - X_{12}B_{22} &= C_{12} - A_{12}X_{22} + X_{11}B_{12} \\
A_{22}X_{21} - X_{21}B_{11} &= C_{21} \\
A_{22}X_{22} - X_{22}B_{22} &= C_{22} + X_{21}B_{12}
\end{align*}
\]

2a, 2b can be executed in parallel as well as 3a, 3b

1. SYLV(’N’, ’N’, A_{22}, B_{11}, C_{21})
2a. GEMM(’N’, ’N’, \alpha = +1, C_{21}, B_{12}, C_{22})
2b. GEMM(’N’, ’N’, \alpha = -1, A_{12}, C_{21}, C_{11})
3a. SYLV(’N’, ’N’, A_{22}, B_{22}, C_{22})
3b. SYLV(’N’, ’N’, A_{11}, B_{11}, C_{11})
4. GEMM(’N’, ’N’, \alpha = -1, A_{12}, C_{22}, C_{12})
5. GEMM(’N’, ’N’, \alpha = +1, C_{11}, B_{12}, C_{12})
6. SYLV(’N’, ’N’, A_{11}, B_{22}, C_{12})

SYCT and matrix functions

- \(A\) triangular => \(F := f(A)\) triangular
- \(f\) analytic => exists series expansion => \(A F - F A = 0\)
- recursive template:

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{22}
\end{bmatrix}
\begin{bmatrix}
F_{11} & F_{12} \\
F_{21} & F_{22}
\end{bmatrix}
- \begin{bmatrix}
F_{11} & F_{12} \\
F_{21} & F_{22}
\end{bmatrix}
\begin{bmatrix}
A_{11} & A_{12} \\
A_{22}
\end{bmatrix}
= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}
\]

\[
\Rightarrow
A_{11}F_{11} - F_{11}A_{11} = 0
A_{11}F_{12} - F_{12}A_{22} = F_{11}A_{12} - A_{12}F_{22},
A_{22}F_{22} - F_{22}A_{22} = 0
\]
Triangular generalized coupled
Sylvester equation - GCSY

\[ AX - YB = C \]
\[ DX - YE = F \]

\((A, D)\) and \((B, E)\) in

generalized Schur

form

Solution \((X, Y)\) over-

writes r.h.s. \((C, F)\)

Two-sided matrix equation: GLYDT

\[ AXA^T - EXE^T = C \]

\[ C = C^T \text{nxn; } (A, E) \text{n x n in gen. Schur form} \]

\[ \text{Unique sol'n } X = X^T \iff \lambda_i \text{ of } A - \lambda E \text{ satisfy } \lambda_i \lambda_j \neq 1 \]

Recursive splitting:

\[
\begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
\begin{bmatrix}
A_{11}^T & A_{12}^T \\
A_{21}^T & A_{22}^T
\end{bmatrix}
\]

\[
\begin{bmatrix}
E_{11} & E_{12} \\
E_{21} & E_{22}
\end{bmatrix}
\begin{bmatrix}
X_{11} & X_{12} \\
X_{21} & X_{22}
\end{bmatrix}
\begin{bmatrix}
E_{11}^T & E_{12}^T \\
E_{21}^T & E_{22}^T
\end{bmatrix}
= \begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
\]
**GLYDT recursion template**

\[ X_{21} = X_{21}^T \Rightarrow \text{three GLYDT subequations:} \]

\[
A_{11}X_{11}A_{11}^T - E_{11}X_{11}E_{11}^T = C_{11} - A_{12}X_{12}A_{12}^T - (A_{11}X_{12} + A_{12}X_{22})A_{12}^T \\
+ E_{12}X_{12}E_{11}^T + (E_{11}X_{12} + E_{12}X_{22})E_{12}^T,
\]

\[
A_{11}X_{12}A_{22}^T - E_{11}X_{12}E_{22}^T = C_{12} - A_{12}X_{22}A_{22}^T + E_{12}X_{22}E_{22}^T,
\]

\[
A_{22}X_{22}A_{22}^T - E_{22}X_{22}E_{22}^T = C_{22}.
\]

**Four two-sided updates of** \( C_{11} \) **as two SYR2K ops:**

\[
C_{11} = C_{11} - (A_{11}X_{12})A_{12}^T - A_{12}(A_{11}X_{12})^T \\
= C_{11} + (E_{11}X_{12})E_{12}^T + E_{12}(E_{11}X_{12})^T.
\]

where \( A_{11}X_{12} \) and \( E_{11}X_{12} \) are TRMM operations.

---

**Two-sided matrix product**

\[ C = \beta C + \alpha \text{op}(A) \text{op}(B)^T \]

- \( A \) and/or \( B \) can be dense or triangular
- One or several of \( A, B \) and \( C \) can be symmetric
- Extra workspace - size of r.h.s.

**Make use of symmetry, e.g., in GLYDT:**

\[ C_{11} = C_{11} - A_{12}X_{22}A_{12}^T \quad \text{and} \quad C_{11} = C_{11} + E_{12}X_{22}E_{12}^T \]
GLYDT performance with optional condition estimation

Table 5.3  Timings for solving unreduced two-sided matrix equations (GLYDT) with optional condition estimation. (Job = X, compute solution only; Job = X + Sep, compute solution and Sep-estimation.) Results from 375 MHz IBM Power3.

<table>
<thead>
<tr>
<th>n</th>
<th>SG03AD using SG03AX</th>
<th></th>
<th>SG03AD using GLYDT</th>
<th></th>
<th>Speedup</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total time</td>
<td>Solver part</td>
<td>Total time</td>
<td>Solver part</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.0277</td>
<td>49.9 %</td>
<td>0.0185</td>
<td>20.1 %</td>
<td>1.50</td>
<td>X</td>
</tr>
<tr>
<td>100</td>
<td>0.180</td>
<td>51.2 %</td>
<td>0.0967</td>
<td>9.0 %</td>
<td>1.86</td>
<td>X</td>
</tr>
<tr>
<td>250</td>
<td>2.89</td>
<td>46.8 %</td>
<td>1.62</td>
<td>4.7 %</td>
<td>1.79</td>
<td>X</td>
</tr>
<tr>
<td>500</td>
<td>59.0</td>
<td>42.3 %</td>
<td>34.5</td>
<td>15. %</td>
<td>1.71</td>
<td>X</td>
</tr>
<tr>
<td>750</td>
<td>303.4</td>
<td>42.0 %</td>
<td>177.5</td>
<td>9.9 %</td>
<td>1.71</td>
<td>X</td>
</tr>
<tr>
<td>1000</td>
<td>646.6</td>
<td>44.6 %</td>
<td>361.8</td>
<td>1.9 %</td>
<td>1.79</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>SG03AD using SG03AX</th>
<th></th>
<th>SG03AD using GLYDT</th>
<th></th>
<th>Speedup</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total time</td>
<td>Solver part</td>
<td>Total time</td>
<td>Solver part</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.117</td>
<td>87.6 %</td>
<td>0.0263</td>
<td>45.6 %</td>
<td>4.44</td>
<td>X+Sep</td>
</tr>
<tr>
<td>100</td>
<td>0.709</td>
<td>87.3 %</td>
<td>0.152</td>
<td>40.6 %</td>
<td>4.68</td>
<td>X+Sep</td>
</tr>
<tr>
<td>250</td>
<td>9.98</td>
<td>84.5 %</td>
<td>2.08</td>
<td>25.4 %</td>
<td>4.81</td>
<td>X+Sep</td>
</tr>
<tr>
<td>500</td>
<td>178.6</td>
<td>80.9 %</td>
<td>37.8</td>
<td>9.4 %</td>
<td>4.73</td>
<td>X+Sep</td>
</tr>
<tr>
<td>750</td>
<td>924.1</td>
<td>80.9 %</td>
<td>184.4</td>
<td>4.5 %</td>
<td>5.01</td>
<td>X+Sep</td>
</tr>
<tr>
<td>1000</td>
<td>2076.6</td>
<td>82.7 %</td>
<td>391.8</td>
<td>8.4 %</td>
<td>5.30</td>
<td>X+Sep</td>
</tr>
</tbody>
</table>

RECSY library

- Recursive blocked algorithms for solving reduced matrix equations
- Recursion implemented in F90
- SMP versions using OpenMP
- F77 wrappers for LAPACK and SLICOT routines
- [www.cs.umu.se/research/parallel/recsy/](http://www.cs.umu.se/research/parallel/recsy/)
ScalAPACK-style library

- The methods presented here can be applied to several similar problems.
- Our aim is to construct a ScalAPACK-style software package of matrix equation solvers for distributed memory machines.
- The triangular solvers will be used in implementing parallel condition estimators for each matrix equation.
- Robert Granat, PhD student

<table>
<thead>
<tr>
<th>Equation</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{op}(A)X \pm \text{Xop}(B) = C$</td>
<td>SYCT</td>
</tr>
<tr>
<td>$\text{op}(A)X + \text{Xop}(A^T) = C$</td>
<td>LYCT</td>
</tr>
<tr>
<td>$\text{op}(A)X\text{op}(B) \pm X = C$</td>
<td>SYDT</td>
</tr>
<tr>
<td>$\text{op}(A)X\text{op}(A^T) - X = C$</td>
<td>LYDT</td>
</tr>
<tr>
<td>$\text{op}(A)X \pm \text{Yop}(B) = C$</td>
<td>GCSY</td>
</tr>
<tr>
<td>$\text{op}(D)X \pm \text{Yop}(E) = F$</td>
<td>GSYL</td>
</tr>
<tr>
<td>$\text{op}(A)\text{Xop}(B) \pm \text{op}(D)\text{Xop}(E) = C$</td>
<td>GLYCT</td>
</tr>
<tr>
<td>$\text{op}(A)\text{Xop}(A^T) - \text{op}(E)\text{Xop}(E^T) = C$</td>
<td>GLYDT</td>
</tr>
</tbody>
</table>

Recursive blocking...

- creates new algorithms for linear algebra software
- expresses dense linear algebra algorithms entirely in terms of level~3 BLAS like matrix-matrix operations
- introduces an automatic variable blocking that targets every level of a deep memory hierarchy
- can also be used to define hybrid data formats for storing block-partitioned matrices (general, triangular, symmetric, packed)
High-performance software

- implementations are based on data locality and superscalar optimization techniques
- recursive blocked algorithms improve on the temporal data locality
- hybrid data formats improve on the spatial data locality
- portable and generic superscalar kernels ensure that all functional units on the processor(s) are used efficiently

Some related and complementary work

- Recursive algorithms and hybrid data structures
  - Winograd-Strassen'69: Douglas et al.'94, ESSL, Demmel-Higham'92 (stability)
  - Quad- and octtrees: Samet'84, Salman-Warner'94 (N-body, Barnes-Hut'84)
  - Cache oblivious algorithms: Leiserson et al.'99 (sorting, FFT, $A^T$)
  - GEMM: Chatterjee et al.'02, Valsalam and Skjellum'02, ATLAS-project
  - LU: Toledo'97 (dense), Dongarra, Eijkhout Luszczek'01 (sparse)
  - QR: Rabani and Toledo'01 (out-of-core), Frens and Wise'03 (Givens-based)
Some related and complementary work

- Automated generation of library software and compiler technology
  - Empirical optimization:
    - PHiPAC - Bilmes, Demmel et al.'97,
    - ATLAS - Whaley, Petitet and Dongarra'00,
    - Sparse kernels - Vuduc, Demmel et al.'03
  - FLAME: Gunnels, Goto, Van de Geijn et al.'01, '02
  - Compiler blockability: Wolf and Lam'91 (loop transformations), Carr and Lehoucq'97
  - Automatic generation of recursive codes:
    - Ahmed and Pingali'00 (iterative algorithms -> recursive), Yi, Adve and Kennedy'00 (convert loop nests into recursive form)

- Thanks for your attention!