

Recursive Blocked Algorithms for Solving Triangular Sylvester-Type Matrix Equations

Erik Elmroth¹, Fred Gustavson², Isak Jonsson¹ and Bo Kågström¹

¹ Department of Computing Science and HPC2N
Umeå University, SE-901 87 Umeå, SWEDEN
{elmroth, isak, bokg}@cs.umu.se

² IBM T. J. Watson Research Center
Box 218, Yorktown Heights, NY 10598, U.S.A.
{gustav@cs.umu.se}

Outline

- Blocked algorithms and memory hierarchies – matching an algorithm and its data structure.
- Recursive blocked algorithms (GEMM) and data formats.
- Sylvester-type matrix equations and the triangular counterparts with applications.
- Recursive blocked solvers for triangular Sylvester-type matrix equations.
- Implementation issues (serial, parallel).
- Some performance results and comparisons.
- Some references.

Blocked Algorithms and Memory Hierarchies

–Matching an Algorithm and its Data Structure

GOAL: Maintain 2-dim data locality at every level of the 1-dim tiered memory structure.

1. EXPLICIT MULTI-LEVEL BLOCKING

- Each loop set matches a specific level of the memory hierarchy.
- Deep knowledge of architecture characteristics needed.
- Needs a blocking parameter for each level.
- Two-level blocked matrix multiply (tuned for L1 and L2 cache).

2. AUTOMATIC BLOCKING VIA RECURSION

- RECURSION: key concept for matching an algorithm and its data structure.
- Recursive algorithms – divide and conquer style.
- Automatic HIERARCHICAL BLOCKING – variable and “squarish”.
- Only tuning parameter is L1 cache.

Recursive Blocked Algorithms

$$\text{GEMM: } C \leftarrow \beta C + \alpha A \cdot B$$

We can split the blocked matrix multiply and add in any of the three matrix dimensions: M , N , and K .

$$\begin{aligned} & \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \\ = & \begin{bmatrix} \begin{bmatrix} C_{11} & C_{12} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \\ \begin{bmatrix} C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \end{bmatrix} \\ = & \begin{bmatrix} \begin{bmatrix} C_{11} \\ C_{21} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix}, \begin{bmatrix} C_{12} \\ C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{12} \\ B_{22} \end{bmatrix} \end{bmatrix} \\ = & \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} + \begin{bmatrix} A_{11} \\ A_{21} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \end{bmatrix} + \begin{bmatrix} A_{12} \\ A_{22} \end{bmatrix} \begin{bmatrix} B_{21} & B_{22} \end{bmatrix} \end{aligned}$$

Always split the largest dimension: problem is kept “squarish”.

Ratio between #flops made on subblocks and #subblocks is maintained as high as possible.

Recursive Blocked GEMM

If $M \leq \Gamma$ and $N \leq \Gamma$ and $K \leq \Gamma$

solve problem using optimized GEMM kernel.

Otherwise,

If $M = \max(M, N, K)$,

two recursive calls: $M' = M/2, N' = N, K' = K$.

else, if $N = \max(N, K)$,

two recursive calls: $M' = M, N' = N/2, K' = K$.

else,

two recursive calls: $M' = M, N' = N, K' = K/2$.

Why $\Gamma > 1$? For very small problems, the overhead for the recursion becomes too expensive – use OPTIMIZED KERNEL.

The split ordering is not so crucial when 2 or 3 of the dimensions are equal. Less than optimal ordering will only impair performance in one or two levels of the recursion.

“Conquer” part is trivial: addition of the results are made implicitly by the leaf kernels.

The recursive technique *automatically blocks* for every level of cache.

This gives temporal locality!

Recursive Blocked Data Formats

OBJECTIVE: Match the recursive algorithm with a recursively blocked data format.

- A new set of data formats for storing block-partitioned matrices.
- Hybrid of two addressing techniques.
- At **BLOCK LEVEL** each submatrix is stored in standard column major (or row-major) order.
- Block size constrained so that a few of them fit in L1 cache.
- Blocks are stored in two **RECURSIVE MATRIX FORMATS**: Rectangle and Isosceles triangle.

RECURSIVE BLOCKED FORMATS allow for maintaining the 2-dim data locality at every level of the 1-dim layered memory structure. CLAIM!

BLOCK ROW and **BLOCK COLUMN ORDERINGS** only maintain data locality at a submatrix level.

Rectangular Recursive Blocked Data Formats

- Always split the largest dimension of rectangular submatrix.
- If the dimensions are equal, we have two choices: split rows (**RBR**), or split columns (**RBC**).
- Odd number of rows: middle row assigned to block at the bottom.
- Odd number of columns: middle column assigned to block to the right.

Blocks to the right or at the bottom may contain submatrices that are not entirely filled. Strategy “minimizes” the difference in number of used elements between the two blocks after the splitting.

- The mutual ordering of the blocks are assigned recursively.

EXAMPLE: A of size 496×380 and $mb = nb = 100$, giving $m = 5$ and $n = 4$.

Assign the numbers 0–19 (contiguous blocks in memory) to the blocks of A .

Rectangular RBR and RBC Formats

RBR: First split is horizontal ($m = 5 > n = 4$). Assign 0–7 to upper part and 8–19 to lower part.

Since #block-rows in upper part ($2 < n = 4$), second split is vertical, assigning 0–3 to the left hand side. This submatrix is square so next split is by row. Etc.

A_{11}	A_{12}	A_{13}	A_{14}	0	5	10	15	0	1	4	5		
A_{21}	A_{22}	A_{23}	A_{24}	1	6	11	16	2	3	6	7		
A_{31}	A_{32}	A_{33}	A_{34}	2	7	12	17	8	9	14	15		
A_{41}	A_{42}	A_{43}	A_{44}	~	3	8	13	~	10	11	16	17	~
A_{51}	A_{52}	A_{53}	A_{54}		4	9	14	19	12	13	18	19	
	Block matrix				Block column				Recursive block		split row (RBR)		
					(BC)								

	0	2	4	6
	1	3	5	7
	8	9	14	15
~	10	12	16	18
	11	13	17	19
	Recursive block			
	split column (RBC)			

This gives spatial locality!

Triangular Recursive Blocked Data Formats

All matrix factorizations can be expressed in terms of rectangular and isosceles triangular matrices.

⇒ Enough to consider the **ISOSCELES CASE**.

For an isosceles right triangle of order N , the splitting procedure resembles the rectangular case:

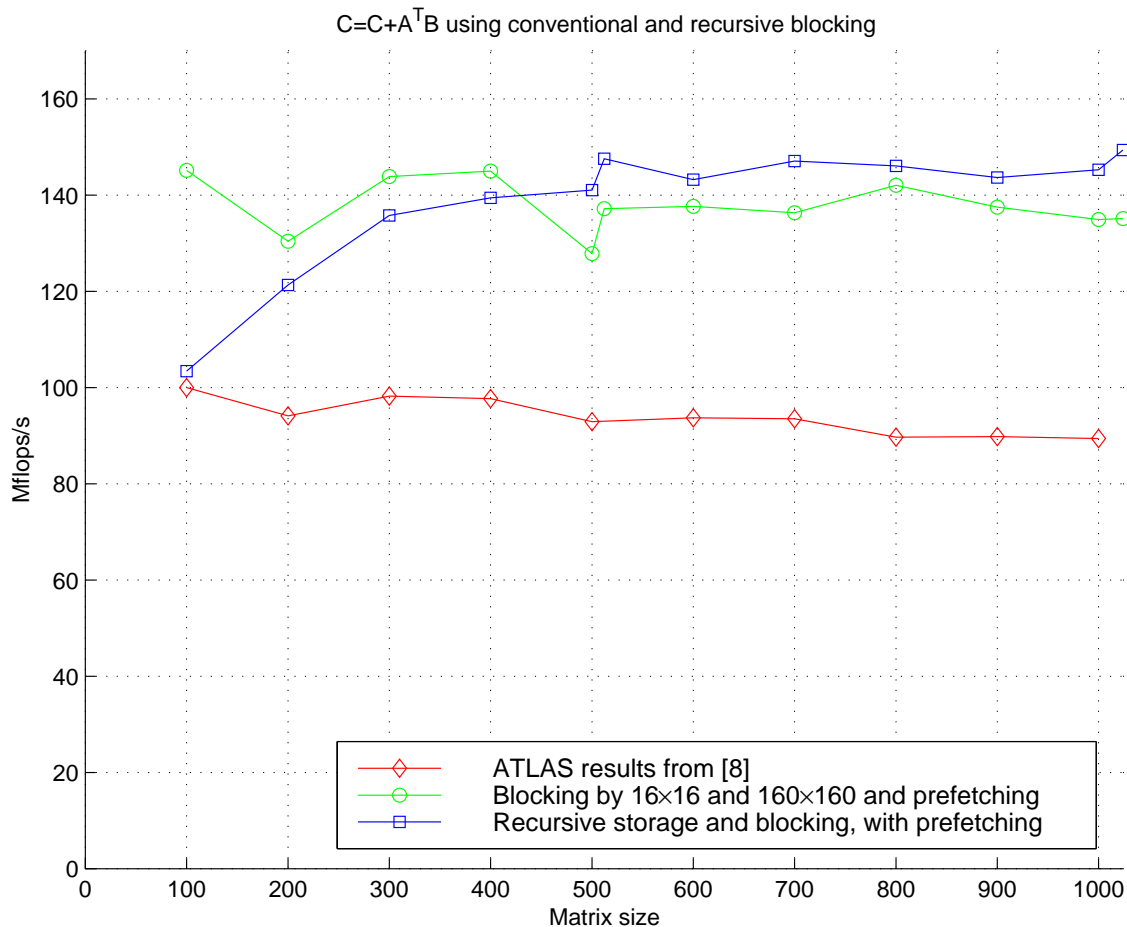
- Split the triangle in two sub-triangles and one rectangle at $N/2$.
- Interior ordering of the blocks in the triangles are determined by applying the algorithm recursively.
- For the rectangle, we have the choices of RBR or RBC.
- Blocks in last block row or column are possibly padded.

Partitioning of a triangular matrix of size $450 \cdot 450$,
block size is $100 \cdot 100$:

A_{11}	0		0		
$A_{21} \ A_{22}$	1 2		1 2		
$A_{31} \ A_{32} \ A_{33}$	3 4	9	3 4	9	
$A_{41} \ A_{42} \ A_{43} \ A_{44}$	5 6	10 12	5 7	10 12	
$A_{51} \ A_{52} \ A_{53} \ A_{54} \ A_{55}$	7 8	11 13 14	6 8	11 13 14	
Block matrix		(RBR)		(RBC)	

Multi-level vs. Recursive Blocking

Uniprocessor Performance—IBM PowerPC 604, 112 MHz



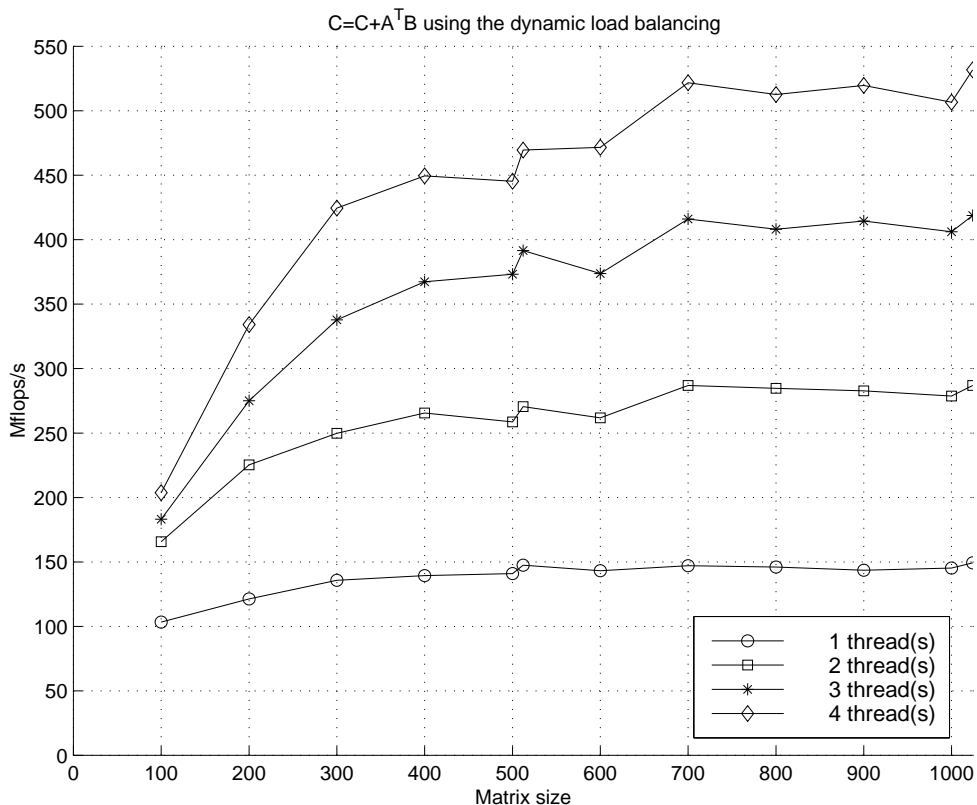
Optimized GEMM Kernel with Level 3 Prefetching:

Technique which enables data to be brought to registers and cache ahead of its use, so when it is needed, it is immediately available. Embedded in the level 3 kernel; during a subblock GEMM computation, the next set of subblocks are prefetched.

SMP Parallelization Using Threads

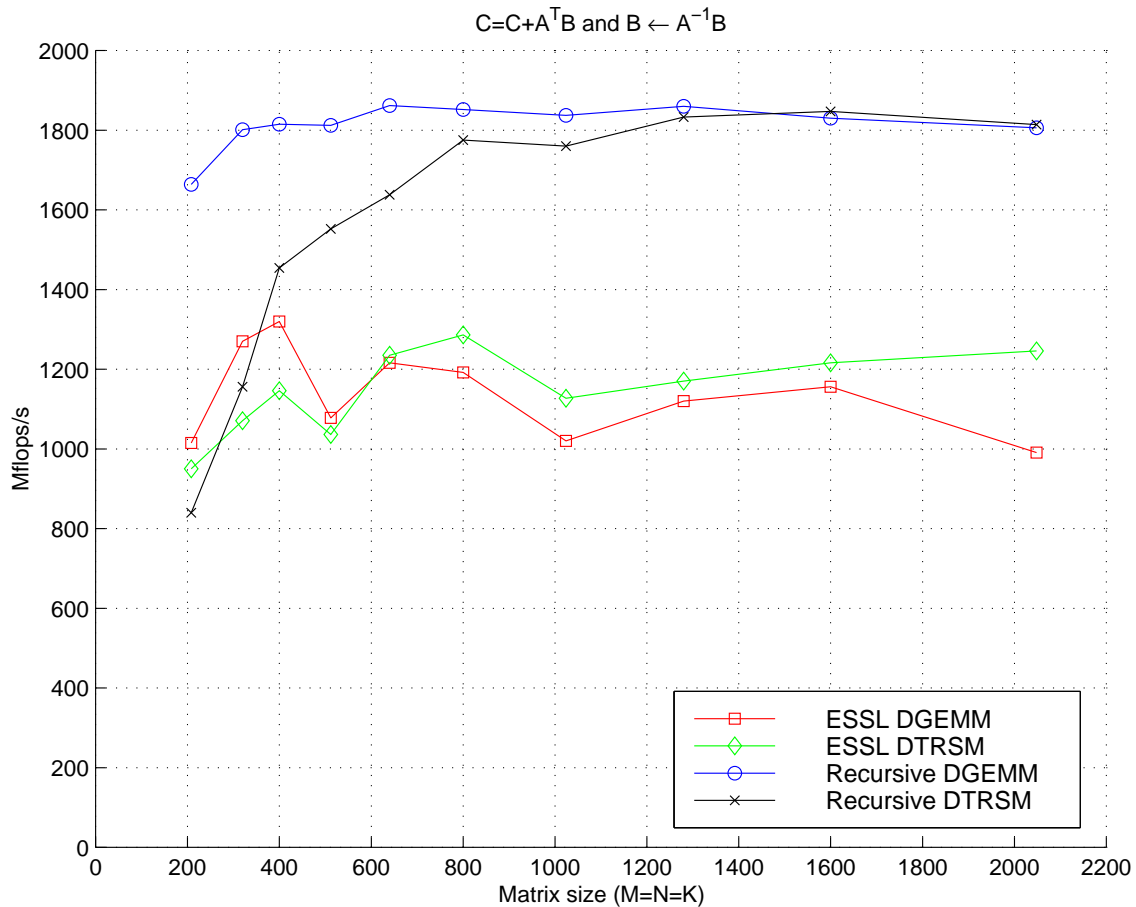
We use a **dynamic distribution** of tasks, which is well-suited for a non-dedicated SMP. A virtual recursion tree is maintained throughout the execution, which is divided into subtrees. Different processes or threads execute on different subtrees.

Performance of Recursive GEMM



DGEMM on a non-dedicated 4-processor PowerPC 604 node, 112 MHz.

Performance of Recursive GEMM and TRSM



DGEMM and DTRSM performance on a non-dedicated PowerPC 604e node, 332 MHz. Peak rate is 2656 Mflops/s. Time for data restructuring is not taken into account (add 5–10 % overhead).

Sylvester-Type Matrix Equations

Matrix equations appear in different control theory applications. Examples include:

- Sylvester: $AX - XB = C$, A , B and C general.
- Lyapunov: $AX - XA^T = C$.
- Stein (or discrete Lyapunov): $AXA^T - X = C$, A general, $C = C^T$ (semi)definite.
- Generalized (coupled) Sylvester:
 $(AX - YB, DX - YE) = (C, F)$.
- Generalized Sylvester: $AXB - CXD = F$.

Typically, the second major step in their solution is to solve a triangular matrix equation. Our blocked recursive technique works for all cases. Here, we illustrate with the [TRIANGULAR STANDARD AND GENERALIZED SYLVESTER EQUATIONS](#).

They also appear naturally in estimating the condition numbers of matrix equations and different eigenspace problems [9, 7, 8]. We illustrate with the standard eigenvalue problem.

Standard Sylvester Eq.–Application 1: Block diagonalization and Spectral projectors

$$S = \begin{bmatrix} A & -C \\ 0 & B \end{bmatrix}, \quad \text{in Schur form.}$$

- S block diagonalized by **similarity** transformation:

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

where R satisfies $AR - RB = C$.

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

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

Important quantity in error bounds for invariant subspaces and clusters of eigenvalues.

- Large $\|P\|_2 = (1 + \|R\|_2^2)^{1/2}$, signals ILL-CONDITIONING.
- Computed estimate: $s = 1/\|P\|_F$

Standard Sylvester Eq.–Application 2: Separation of two matrices

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

$$\text{where } Z = I_n \otimes A - B^T \otimes I_m.$$

- $\text{Sep}[A, B] = 0$ if and only if A and B have a **common** eigenvalue.
- $\text{Sep}[A, B]$ is **small** if there is small perturbation of A or B that makes them have a common eigenvalue.
- General case: Sep may be much smaller than min. distance between the eigenvalues of A and B .
- COMPUTING $\sigma_{\min}(Z)$: $O(m^3n^3)$ operation. Impractical!
- Reliable Sep **estimates** of cost $O(mn^2 + m^2n)$:

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

and

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

Recursive Triangular Sylvester Solvers

$op(A) \cdot X \pm X \cdot op(B) = \beta \cdot C$, $C \leftarrow X (M \times N)$, where $A(M \times M)$ and $B(N \times N)$ upper quasi-triangular.

$transA = 'N'$, $transB = 'N'$, $sign = -$, $\beta = 1$:

Case 1: $M \gg N$: Split A and C (by rows)

$$\begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} - \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} B = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

$$A_{11}X_1 - X_1B = C_1 - A_{12}X_2$$

$$A_{22}X_2 - X_2B = C_2$$

1. SYLV('N', 'N', A_{22} , B , C_2)
2. GEMM('N', 'N', $\alpha = -1$, A_{12} , C_2 , C_1)
3. SYLV('N', 'N', A_{11} , B , C_1)

Case 2: $M \ll N$: Split B and C (by columns)

$$A \begin{bmatrix} X_1 & X_2 \end{bmatrix} - \begin{bmatrix} X_1 & X_2 \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$$

$$AX_1 - X_1B_{11} = C_1$$

$$AX_2 - X_2B_{22} = C_2 + X_1B_{12}$$

1. SYLV('N', 'N', A , B_{11} , C_1)
2. GEMM('N', 'N', $\alpha = +1$, C_1 , B_{12} , C_2)
3. SYLV('N', 'N', A , B_{22} , C_2)

Case 3: $N/2 \leq M \leq 2N$: Split A , B and C

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

$$A_{11}X_{11} - X_{11}B_{11} = C_{11} - A_{12}X_{21}$$

$$A_{11}X_{12} - X_{12}B_{22} = C_{12} - A_{12}X_{22} + X_{11}B_{12}$$

$$A_{22}X_{21} - X_{21}B_{11} = C_{21}$$

$$A_{22}X_{22} - X_{22}B_{22} = C_{22} + X_{21}B_{12}$$

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})

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

Implementation issues

Two alternatives for doing the recursive splits:

1. Always split the largest dimension in two (Cases 1 and 2).
2. Split both dimensions simultaneously (Case 3) when the dimensions are within a factor 2 from each other.

First alternative gives less code but a longer and more narrow recursion tree. When we split all matrices we get a shorter but wider recursion tree.

POWER3: Performance is almost the same, with alternative 1 being about 2 % faster for problem sizes 500–1000.

In the performance graph shown below, alternative 1 is used.

For the [recursive subroutine using recursive GEMM](#), the largest dimensions is split as close as possible at a multiple of the recursive data format block size (= 60).

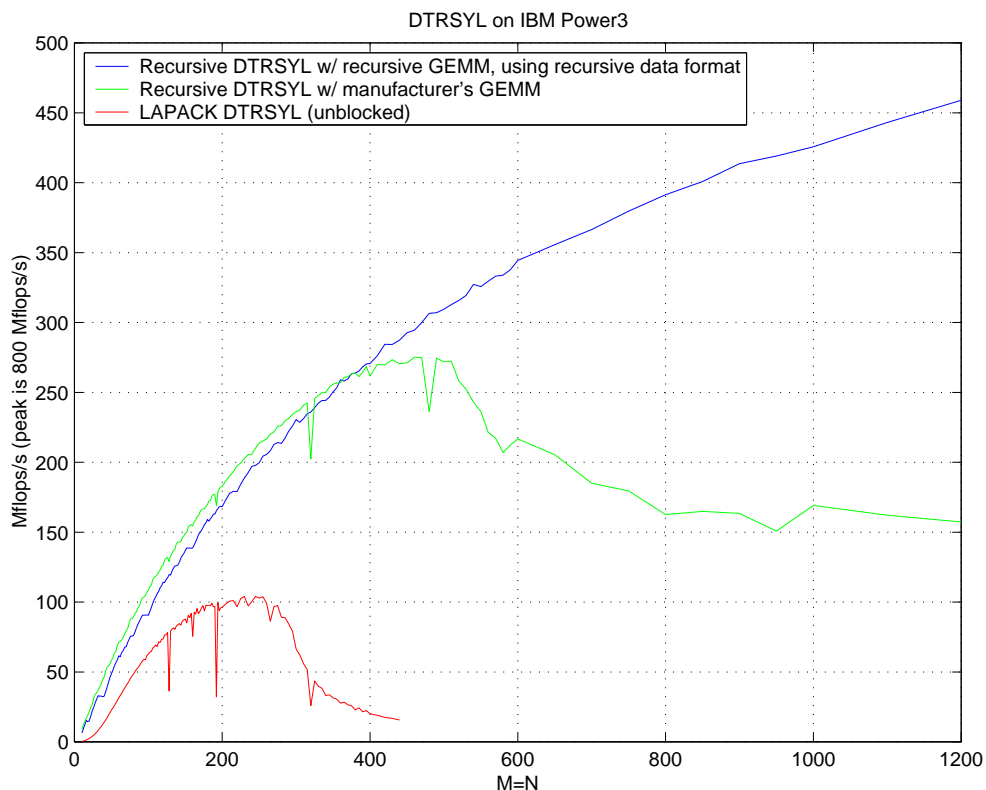
For problems smaller than the block size, the dimension is split in two until the problem is of size $2 \times 2 - 4 \times 4$, when subsystems are solved using the [Kronecker product representation](#):

$$(I_n \otimes A - B^T \otimes I_m) \text{vec}(X) = \text{vec}(C)$$

The system is permuted in order to make the problem more upper triangular and solved (LU with row pivoting).

For the recursive subroutine using the **manufacturer's GEMM**, the same procedure is done, except that there is no attempt to make the problem dimensions a multiple of any block size (can not be controlled explicitly).

Performance Standard Sylvester



Results for DTRSYL-variants on IBM Power3, 200 MHz.

Preliminary results!

Recursive Generalized Sylvester Solvers

$$\begin{aligned} AX - YB &= \beta C, & C &\leftarrow X(M \times N) \\ DX - YE &= \beta F, & F &\leftarrow Y(M \times N) \end{aligned}$$

(A, D) and (B, E) in **generalized Schur form** with A, B quasi-triangular and D, E triangular.

Case 1: $M > N$: Split (A, D) and (C, F) (by rows)

$$\begin{aligned} \begin{bmatrix} A_{11} & A_{12} \\ & A_{22} \end{bmatrix} \begin{bmatrix} R_1 \\ R_2 \end{bmatrix} - \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} B &= \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \\ \begin{bmatrix} D_{11} & D_{12} \\ & D_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} - \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} E &= \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A_{11}X_1 - Y_1B &= C_1 - A_{12}X_2 \\ A_{22}X_2 - Y_2B &= C_2 \\ D_{11}X_1 - Y_1E &= F_1 - D_{12}X_2 \\ D_{22}X_2 - Y_2E &= F_2 \end{aligned}$$

1. TGSYL('N', A_{22} , B , C_2 , D_{22} , E , F_2)
2. GEMM('N', 'N', $\alpha = -1$, A_{12} , C_2 , C_1)
3. GEMM('N', 'N', $\alpha = -1$, D_{12} , C_2 , F_1)
4. TGSYL('N', A_{11} , B , C_1 , D_{11} , E , F_1)

Recursive Generalized Sylvester Solvers

$$AX - YB = \beta C, \quad C \leftarrow X(M \times N)$$

$$DX - YE = \beta F, \quad F \leftarrow Y(M \times N)$$

Case 2: $M < N$: Split (B, E) and (C, F) (by columns)

$$A \begin{bmatrix} X_1 & X_2 \end{bmatrix} - \begin{bmatrix} Y_1 & Y_2 \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ & B_{22} \end{bmatrix} = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$$

$$D \begin{bmatrix} X_1 & X_2 \end{bmatrix} - \begin{bmatrix} Y_1 & Y_2 \end{bmatrix} \begin{bmatrix} E_{11} & E_{12} \\ & E_{22} \end{bmatrix} = \begin{bmatrix} F_1 & F_2 \end{bmatrix}$$

$$AX_1 - Y_1 B_{11} = C_1$$

$$AX_2 - Y_2 B_{22} = C_2 + Y_1 B_{12}$$

$$DX_1 - Y_1 E_{11} = F_1$$

$$DX_2 - Y_2 E_{22} = F_2 + Y_1 E_{12}$$

1. TGSYL('N', A, B₁₁, C₁, D, E₁₁, F₁)
2. GEMM('N', 'N', $\alpha = 1$, F₁, B₁₂, C₂)
3. GEMM('N', 'N', $\alpha = 1$, F₁, D₁₂, F₂)
4. TGSYL('N', A, B₂₂, C₂, D, E₂₂, F₂)

Implementation issues

As for the standard case it is also possible to split both dimensions simultaneously (Case 3).

We always split the largest dimension in the performance graphs presented below (Cases 1 and 2).

Recursion is done down to $2 \times 2 - 4 \times 4$ blocks.

Recursive 1: Small generalized Sylvester equations are solved by LU and row pivoting on the Kronecker product representation [9].

Recursive 2: Small problems are solved using the LAPACK kernel DTGSY2 [8].

Using Recursive 1 we **gain much speed** due to a less robust and reliable solver. The LAPACK kernel DTGSY2 makes use of both scaling and complete pivoting.

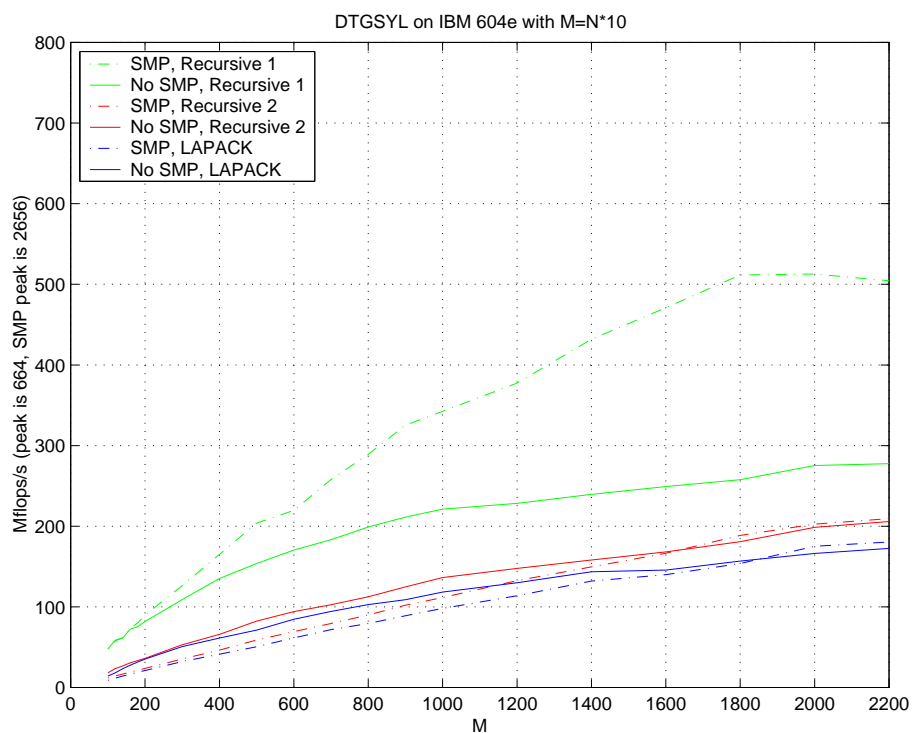
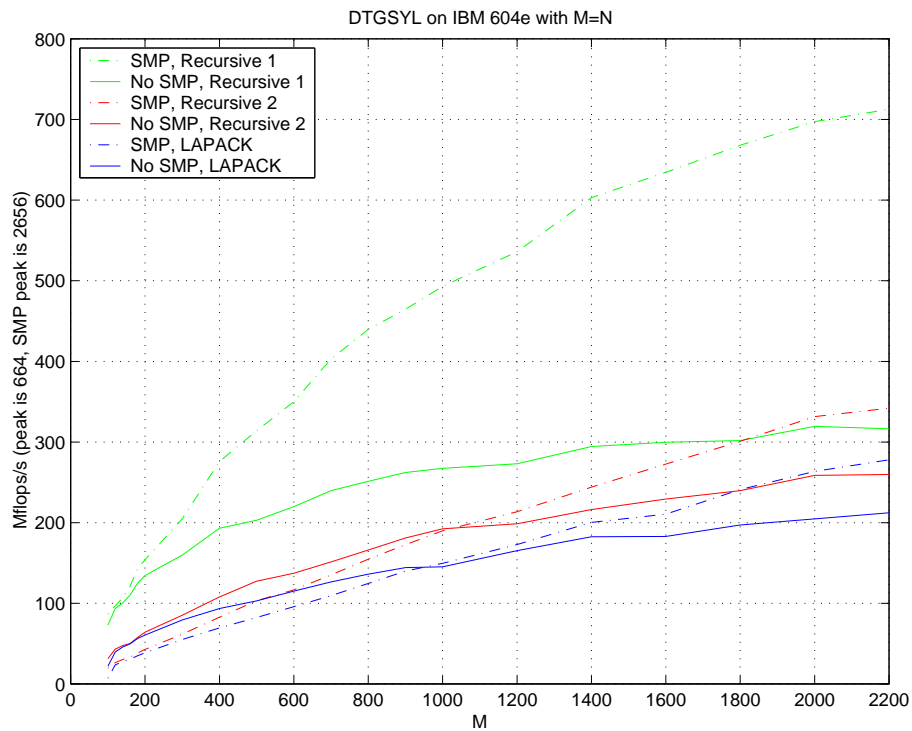
Recursive 2 outperforms the LAPACK DTGSY subroutine [8] for the problem sizes tested.

DTGSYL implements a blocked algorithm, and it is necessary to carefully tune the algorithm with the correct blocking parameters (as with all LAPACK routines).

This is not necessary with the recursive algorithms!

For larger problems, when $M = N \geq 1400$, there is not enough physical memory at the Power 3 machine that the tests runs were performed on, it is necessary to use paging and virtual memory. It is our belief that the recursive blocked data format will use this virtual memory in an optimal manner and thus minimize paging. However, we have not seen this behaviour in our first implementations.

Performance gen. Sylvester-IBM 604E



Acknowledgements

This research was conducted using the resources of the High Performance Computing Center North (HPC2N), Sweden, and UNI-C, Denmark.

New paper appears soon! – Some other REFERENCES

- [1] E. Elmroth and F. Gustavson. New Serial and Parallel Recursive QR Factorization Algorithms for SMP Systems. In Kågström et al. (eds), *Applied Parallel Computing, PARA'98*, Lecture Notes in Computer Science, Vol. 1541, pp 120–128, Springer-Verlag, 1998.
- [2] F. Gustavson. Recursion leads to automatic variable blocking for dense linear algebra. *IBM J. Res. Develop.*, 41(6):737–755, November 1997.
- [3] F. Gustavson, A. Henriksson, I. Jonsson, B. Kågström and P. Ling. Recursive Blocked Data Formats and BLAS's for Dense Linear Algebra Algorithms. In Kågström et al. (eds), *Applied Parallel Computing, PARA'98*, Lecture Notes in Computer Science, Vol. 1541, pp 195–206, Springer-Verlag, 1998.
- [4] F. Gustavson, A. Henriksson, I. Jonsson, B. Kågström and P. Ling. Superscalar GEMM-based Level 3 BLAS – The On-going Evolution of a Portable and High-Performance Library. In Kågström et al. (eds), *Applied Parallel Computing, PARA'98*, Lecture Notes in Computer Science, Vol. 1541, pp 207–215, Springer-Verlag, 1998.
- [5] B. Kågström, P. Ling, and C. Van Loan. GEMM-based level 3 BLAS: High-performance model implementations and performance evaluation benchmark. *ACM Trans. Math. Software*, 24(3):268–302, 1998.
- [6] B. Kågström, P. Ling, and C. Van Loan. GEMM-based level 3 BLAS: Portability and optimization issues. *ACM Trans. Math. Software*, 24(3):303–316, 1998.
- [7] B. Kågström and P. Poromaa. Distributed and Shared Memory Block Algorithms for the Triangular Sylvester Equation with sep^{-1} Estimator. *SIAM Journal on Matrix Analysis and Application*, 13(1):90–101, January 1992.
- [8] B. Kågström and P. Poromaa. LAPACK–Style Algorithms and Software for Solving the Generalized Sylvester Equation and Estimating the Separation between Regular Matrix Pairs. *ACM Trans. Math. Software*, 22(1):78–103, March 1996.
- [9] B. Kågström and L. Westin. Generalized Schur methods with condition estimators for solving the generalized Sylvester equation. *IEEE Trans. Autom. Contr.*, 34(4):745–751, 1989.
- [10] P. Poromaa. Parallel Algorithms for Triangular Sylvester Equations: Design, Scheduling and Scalability Issues. In Kågström et al. (eds), *Applied Parallel Computing, PARA'98*, Lecture Notes in Computer Science, Vol. 1541, pp 438–446, Springer-Verlag, 1998.