Computers Math. Applic. Vol. 31, No. 4/5, pp. 55–61, 1996

Using Strassen's Matrix Multiplication in High Performance Solution of Linear Systems

M. PAPRZYCKI AND C. CYPHERS
Department of Mathematics and Computer Science
University of Texas of the Permian Basin
Odessa, TX 79762-0001, U.S.A.

Abstract—Performance characteristics of dense and structured blocked linear system solvers are studied when Strassen's matrix multiplication is used in the update step. Results of experiments on a multiprocessor Cray Y-MP are presented and discussed.

Keywords—Strassen's algorithm, Dense and structured linear solvers, Performance characteristics.

1. INTRODUCTION

In 1969, Strassen published a paper in which a new recursive method of multiplying matrices was introduced [1]. This idea was quickly dismissed as the new algorithm has been considered unstable. Only nowadays when a large number of algorithms have been redesigned in a blocked form rich in matrix multiplication [2,3] a renewed interest in Strassen's Algorithm can be observed [4,5]. The aim of this paper is to discuss the performance characteristics of linear algebraic solvers when the Strassen's algorithm is applied in their block update step. In Section 2, Strassesn's algorithm is introduced, its stability discussed and its performance compared to that of standard matrix multiplication. Section 3 contains examples of applying Strassen's algorithm to dense linear solvers. Section 4 presents the results of applying Strassen's algorithm to the solution of Almost Block Diagonal systems.

2. STRASSEN'S ALGORITHM

The following description is based on the original paper by Strassen [1]. Assume that $A, B, C \in \mathbf{R}^{n\times n}$ and, for simplicity, assume that $n = 2^k$. Consider the following division of A, B and C into square blocks:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \qquad AB = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix},$$

and calculate:

This work was supported by Cray time from the Center for High Performance Computing in Austin. Their help is kindly acknowledged.

$$
\begin{aligned}
\mathrm{I} &= (A_{11} + A_{22})(B_{11} + B_{22}),\\
\mathrm{II} &= (A_{21} + A_{22})B_{11},\\
\mathrm{III} &= A_{11}(B_{12} - B_{22}),\\
\mathrm{IV} &= A_{22}(-B_{11} + B_{21}),\\
\mathrm{V} &= (A_{11} + A_{12})B_{22},\\
\mathrm{VI} &= (-A_{11} + A_{12})(B_{11} + B_{12}),\\
\mathrm{VII} &= (A_{12} - A_{22})(B_{21} + B_{22}).
\end{aligned}
$$

To obtain the final answer, compute

$$
\begin{aligned}
C_{11} &= \mathrm{I} + \mathrm{IV} - \mathrm{V} + \mathrm{VII}, & C_{12} &= \mathrm{III} + \mathrm{V},\\
C_{21} &= \mathrm{II} + \mathrm{IV}, & C_{22} &= \mathrm{I} + \mathrm{III} - \mathrm{II} + \mathrm{VI}.
\end{aligned}
$$

This process is repeated recursively. The arithmetical complexity of this algorithm (for any n) is bounded by $4.7n^{\log 7}$ arithmetical operations (multiplications, additions and subtractions). As $\log 7 \approx 2.8$, for large n, the new algorithm should be considerably faster than the standard one.

The stability of Strassen's algorithm has been extensively studied by Higham [6]. He showed that for square matrices (where $n = 2^k$), if $\hat{C}$ is the computed approximation to $C(\hat{C} = AB + \Delta C)$, then

$$\|\Delta C\| \le c(n,n,n)u\|A\|\,\|B\| + O(u^2),$$

where u is the unit roundoff, and

$$c(n,n,n) = \left(\frac{n}{2^k}\right)^{\log_2 12}\left(\left(2^k\right)^2 + 5 * 2^k\right) - 5n.$$

(For nonsquare matrices the function defining c becomes substantially more complicated, but the overall result remains the same.) Observe that $nu\|A\|\,\|B\| + O(u^2)$ is the upper bound for the standard matrix multiplication. Thus, even though Strassen's matrix multiplication is less stable, the instability predicted by the error analysis should not (in most cases) be catastrophic.

To study the possible gains from Strassen's algorithm, we have performed a series of experiments on an 8-processor Cray Y-MP. Cray Research Inc., as a part of its scientific library, provides an optimized implementation of Strassen's algorithm (routine SGEMMS). It is implemented using calls to level 3 BLAS routines and requires $2.34n^2$ element workspace. One-processor timings were performed using the *perftrace* utility. In a multiprocessor environment, where *perftrace* does not work correctly, the system's *timef* function was used on an empty machine. Each result presented in this paper is an average of multiple runs.

We have compared the performance of the Strassen's algorithms with the best standard matrix multiplication subroutine (SGEMM) from the Cray's Scientific Library. It has been found that in the uniprocessor mode Strassen's algorithm outperforms the standard one for matrices of size larger than $n = 150$. For $n = 1500$ the time gain was already 30% (see [7] for more details). The situation changes slightly for the multiprocessor case. It was found that as the number of processors increases the matrix size for which any gain can be reported also increases. For the 8-processor system for $n = 1900$, Strassen's algorithm was about 12% faster [8].

3. DENSE GAUSSIAN ELIMINATION

Since the introduction of the BLAS standard, most of the linear algebraic algorithms have been expressed in a blocked form [2,3]. Such representations are rich in matrix multiplications, and at the same time as the matrix size increases, the percent of time spent in block updates increases [9]. We have experimented with three (SAXPY, DOT and GAXPY) column-oriented versions of Gaussian elimination.

The SAXPY version is a standard Gaussian elimination represented in a blocked form (and its name originates from a frequent use of *axpy* operations: $x = x + \alpha y$). The DOT version is a variant of Crout's method (and its name originates from a frequent calculation of ***dot-products***). Finally, the GAXPY version is based on a generalized *axpy* operation (e.g., matrix-vector multiplication). For more detailed descriptions of these variants see [3]. The results of selected experiments on a one-processor system are summarized in Table 1.

Table 1. Dense Gaussian elimination; performance comparison; time in seconds; (S) indicates the code using Strassen-based update.

n	SAXPY	GAXPY	DOT	SAXPY(S)	GAXPY(S)	DOT(S)
300	7.03E−2	6.97E−2	7.07E−2	7.06E−2	7.05E−2	7.08E−2
800	1.28E+0	1.28E+0	1.27E+0	1.21E+0	1.23E+0	1.22E+0
1300	4.86E+0	4.84E+0	4.85E+0	4.41E+0	4.56E+0	4.37E+0
1800	1.31E+1	1.28E+1	1.28E+1	1.08E+0	1.09E+0	1.10E+0

Figure 1 summarizes the results for the full 8-processor system for $n = 600, \ldots, 2600$ (the optimal performance blocksize 256 was used). The performance is represented using scaling similar to one used by [4] (where the number of arithmetical operations of a standard Gaussian elimination was divided by the execution time).

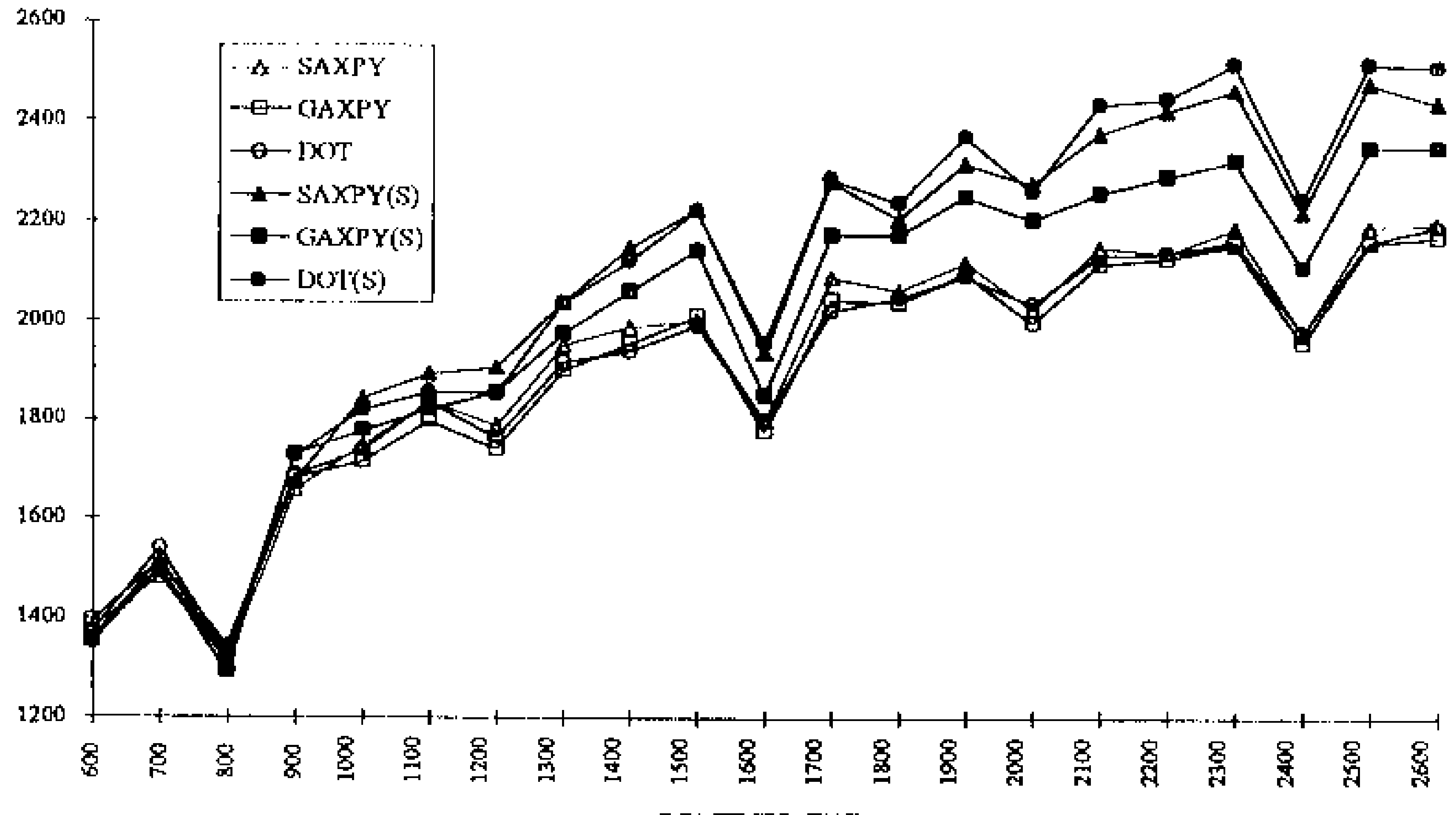

Figure 1. Performance comparison; 8-processor system; results in scaled MFlops.

The gain from using Strassen's algorithm on a one-processor system is observable from approximately $n = 400$ onward; for $n = 1800$ it reaches about 12%. The standard GAXPY and DOT versions outperform the SAXPY version; among the codes utilizing Strassen-based update the SAXPY version is the fastest since larger matrices are multiplied in its update step (see [5] for more details). For the multiprocessor system, as the number of processors increases the size of the matrix for which any gain occurs increases (this is similar to what was reported for matrix multiplication). For the 8-processor system that minimal matrix size is $n = 1000$. For matrices of size $n = 2500$ the gain is approximately 14%. The DOT and SAXPY versions outperform the GAXPY version for both standard and Strassen-based codes. It should be added that the 8-processor performance decrease for $n = 800$, 1600 and 2400 has been also observed for the one-processor system and can be attributed to the memory section conflicts (this is a well-known problem with Cray's hardware, see also [5,7,8] for more details).

4. ALMOST BLOCK DIAGONAL SYSTEMS

The final set of experiments was devoted to the application of Strassen's matrix multiplication in structured linear systems solvers. We have selected Almost Block Diagonal (ABD) systems arising from the discretization of boundary value ordinary differential equations and spectral decomposition applied to the fluid flow in a re-entrant tube. One of the most efficient ABD solvers is a level 3 BLAS based extension of the work of Diaz *et al.* [10] proposed in [11]. We will describe the algorithm very briefly since a more detailed description can be found in [11–13]. For our purposes we will represent the ABD system as

$$\begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} & & & & & \\ A_{2,1} & A_{2,2} & A_{2,3} & & & & & \\ & A_{3,2} & A_{3,3} & A_{3,4} & A_{3,5} & & & \\ & A_{4,2} & A_{4,3} & A_{4,4} & A_{4,5} & & & \\ & & & \ddots & & & & \\ & & & & A_{n-1,n-3} & A_{n-1,n-2} & A_{n-1,n-1} & A_{n-1,n} \\ & & & & A_{n,n-3} & A_{n,n-2} & A_{n,n-1} & A_{n,n} \end{pmatrix},$$

where $A_{i,i}$ are square and $A_{i,j}$ are rectangular blocks of varying sizes. The i^{th} step of the algorithm consists of two phases. In Phase I, the rectangular block

$$\begin{pmatrix} A_{2i-1,2i-1} \\ A_{2i,2i-1} \end{pmatrix},$$

is decomposed using Gaussian elimination with partial pivoting and row interchanges into

$$P \begin{pmatrix} L_{2i-1,2i-1} \\ L_{2i,2i-1} \end{pmatrix} U_{2i-1,2i-1},$$

where P is the permutation matrix. After this factorization, block

$$\begin{pmatrix} A_{2i-1,2i} & A_{2i-1,2i+1} \\ A_{2i,2i} & A_{2i,2i+1} \end{pmatrix}$$

will be updated by the inverse of

$$\begin{pmatrix} L_{2i-1,2i-1} & 0 \\ L_{2i,2i-1} & I \end{pmatrix}.$$

In Phase II, block

$$\begin{pmatrix} A_{2i,2i} & A_{2i,2i+1} \end{pmatrix}$$

is decomposed using Gaussian elimination with partial pivoting and column interchanges, and updated as in Phase I. The decomposition will be performed in a blocked fashion using LAPACK [2] routine SGETRF (which uses appropriate level 1 and 2 BLAS kernels) or its column interchange based version. The update steps consist of calls to the level 3 BLAS routines STRSM (triangular system solver for multiple right-hand sides) and SGEMM. The general performance characteristics of this algorithm have been discussed in [12,13]. It has been shown that it outperforms other versions of alternate row and column elimination.

First, let us consider applying this algorithm to a fairly regular ABD system arising from a finite difference discretization of a system of ordinary differential equations. Table 2 summarizes the results for the ABD systems representing systems of 200 and 400 first order differential equations with separated boundary conditions discretized on a mesh of 50 points. ITBC represents the

Table 2. Performance comparison; regular ABD system; time in seconds; DGE—non-Strassen version; STR—Strassen version.

ITBC	200 equations		400 equations	
	DGE	STR	DGE	STR
1	2.67	2.68	19.6	19.6
50	2.80	2.81	20.1	20.1
100	2.88	2.82	20.8	20.4
150			21.3	20.7
200			21.7	20.4

number of left boundary conditions. The performance optimal blocksize of 256 was used in the blocked decomposition step.

Only for relatively large blocks can any gain be observed. When ITBC is increasing, the size of the blocks in the overlap is also increasing, which leads to additional gains from using Strassen's algorithm (which replaces all calls to the matrix multiplication regardless of the matrix size).

Let us now consider Chebyshev spectral collocation method applied to the flow of an incompressible fluid through a re-entrant tube. Only a short description of the problem and the method is presented here (for more details, see [14,15]). The flow is assumed to be steady and laminar. It is governed by the Navier-Stokes equations which, in the stream function formulation, become

$$\Delta^4\psi - \mathrm{Re}\left[\frac{\partial\psi}{\partial y}\frac{\partial}{\partial x}(\Delta^2\psi) - \frac{\partial\psi}{\partial x}\frac{\partial}{\partial y}(\Delta^2\psi)\right] = 0,$$

where Re is the Reynolds number. The problem is linearized using a Newton-type method and the flow region is divided into four elements. To find the solution of the problem a linear system resulting from the discretization needs to be solved in each step. This linear system is of the form presented in Figure 2.

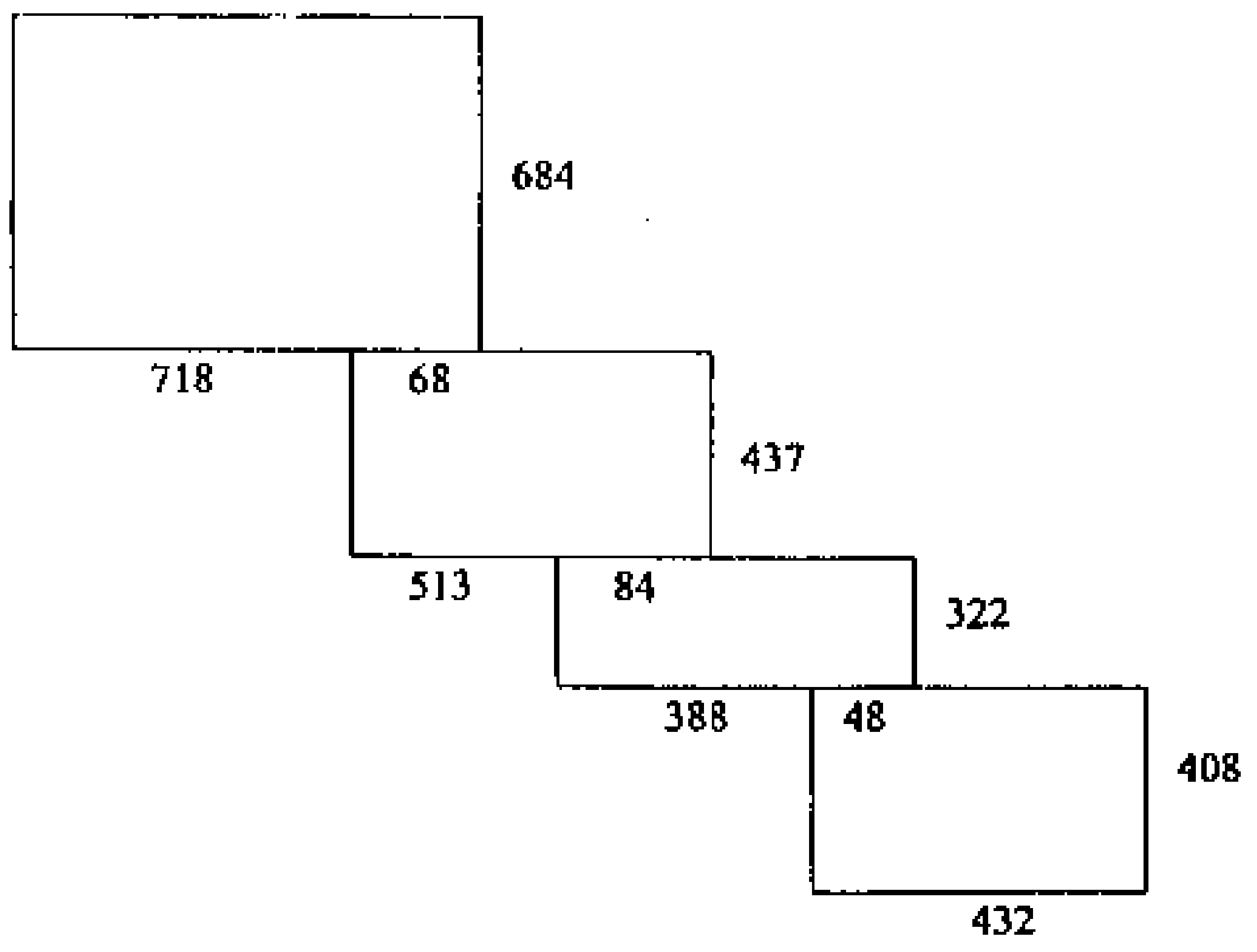

Figure 2. Example of an almost block diagonal system to be solved in each step of the algorithm.

Table 3. Solution of the linear system; time in seconds; DGE—non-Strassen version; STR—Strassen version.

Problem size	Decomposition	
	DGE	STR
1547	0.644	0.634
1642	0.771	0.758
1813	1.66	1.14
1851	1.28	1.24

Depending on the problem parameters the size of the first block changes (from 380×414, 475×509, 646×680 to 684×718) leading to linear systems of sizes from $n = 1547$ to $n = 1851$. The first series of experiments was performed with the solution of a linear system with a given size. Table 3 summarizes the results (time in seconds).

Second, we have experimented with a one processor solution of the whole problem. Table 4 presents the results for the largest system ($n = 1851$) for various values of Re (increase in Re forces the use of a continuation-type method to obtain the solution and increases the number of linear systems solved).

Table 4. Solution of the whole problem; one processor; results in seconds; DGE—non-Strassen version; STR—Strassen version.

Re	# of iterations	DGE	STR
100	8	14.74	14.72
200	12	18.95	18.91
300	17	26.89	26.82
400	22	34.74	34.61

In the case of the linear system itself, any real gain can be reported only for the largest system. It translates into a minimal gain in the solution of the whole system. These facts can be explained first by a very small overlap between the blocks, which hinders the performance of Strassen's algorithm. Second, when the whole problem is solved, additional factors related to Cray's architecture seem to influence the performance (for similar results, cf. [16]).

Finally, observe that there are two places in which Strassen's algorithm can be used in the ABD solver: in the update step or inside the blocked Gaussian elimination. When Strassen's update was used in both places (for the above problem) the iterations did not converge. This seems to suggest that although Strassen's algorithm can be usually applied, there is a limit on its applicability due to its stability properties.

5. CONCLUSION

We have presented a number of examples where Cray's implementation of Strassen's matrix multiplication was applied to the solution of dense and structured linear systems. This algorithm definitely leads to performance improvements in a one-processor environment as well as for large dense linear systems. The gains for structured linear systems are much less spectacular. In a multiprocessor environment, Cray's implementation is not very successful. It can be hoped that new research into the parallelization of Strassen's algorithm [17,18] will lead to more promising results for the other matrix-multiplication oriented algorithms.

REFERENCES

1. V. Strassen, Gaussian elimination is not optimal, *Numerical Mathematics* **13**, 354–356 (1969).
2. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK User's Guide*, SIAM, Philadelphia, (1993).
3. M.J. Dayde and I.S. Duff, Level 3 BLAS in LU factorization on Cray-2, ETA-10P and IBM 3090-200/VF, *The International Journal of Supercomputer Applications* **3**, 40–70 (1989).
4. D.H. Bailey, K. Lee and H.D. Simon, Using Strassen's algorithm to accelerate the solution of linear systems, *The Journal of Supercomputing* **4**, 357–371 (1990).
5. M. Paprzycki, Comparison of Gaussian elimination algorithms on a Cray Y-MP, *Lin. Alg. and Appl.* **172**, 57–69 (1992).
6. N.J. Higham, Exploiting fast matrix multiplication within the level 3 BLAS, *ACM Trans. on Math. Soft.* **16**, 352–368 (1990).
7. M. Paprzycki and C. Cyphers, Multiplying matrices on the Cray—Practical considerations, *CHPC Newsletter* **6**, 77–82 (1991).
8. M. Paprzycki, Parallel matrix multiplication—Can we learn anything new?, *CHPC Newsletter* **7**, 55–59 (1992).

9. R.A. van de Geijn, *LINPACK Benchmark on the Intel Touchstone GAMMA and DELTA Machines*, Preliminary Report, Department of Computer Science, University of Texas, (1991).
10. J.C. Diaz, G. Fairweather and P. Keast, FORTRAN packages for solving certain almost block diagonal linear systems by modified alternative row and column elimination, *ACM Trans. on Math. Software* **9**, 359–375 (1983).
11. C. Cyphers, M. Paprzycki and I. Gladwell, A level 3 BLAS based solver for almost block diagonal systems, SMU Software Report 92-3, Southern Methodist University, (1992).
12. M. Paprzycki and I. Gladwell, Solving almost block diagonal systems using level 3 BLAS, In *Proceedings of The Fifth SIAM Conference on Parallel Processing for Scientific Computing*, pp. 52–62, SIAM, Philadelphia, (1992).
13. I. Gladwell and M. Paprzycki, Parallel solution of almost block diagonal systems using level 3 BLAS, *J. of Comp. and Appl. Math.* **45**, 181–189 (1993).
14. A. Karageorghis, The numerical solution of laminar flow in a re-entrant tube geometry by a Chebyshev spectral element collocation method, *Comp. Meth. in Appl. Mech. and Eng.* **100**, 339–358 (1992).
15. C. Cyphers, M. Paprzycki and A. Karageorghis, High performance solution of partial differential equations discretized using a Chebyshev spectral collocation method, *J. Comp. Apl. Math.* (to appear).
16. M. Paprzycki, N. Irwin and P.E. Hodges, Using BLAS in the estimation of nonlinear, Non-Gaussian state-space models, In *Proceedings of the Ninth Annual Conference on Applied Mathematics*, pp. 332–342, University of Central Oklahoma, (February 1993).
17. J.A. Brown, A transform approach to fast matrix multiplication, *Workshop on Parallel Scientific Computing*, Technical University of Denmark, Copenhagen, (June 1994).
18. C.C. Chou, Y. Deng, G. Li and Y. Wang, Parallelizing Strassen's method for matrix multiplication on distributed-memory MIMD Computers, The University at Stony Brook Technical Report, SUNYSB-AMS-93-17 (1994).