TR/PA/94/10

The CERFACS experience

M. J. Daydé [1,3] and I. S. Duff [2,3]

ABSTRACT

We consider various aspects of the European Centre, CERFACS, located in Toulouse, SW France. We discuss its history and structure which we believe to be unique among European or indeed world scientific centres. We then describe some of the current research activities of the Parallel Algorithm Group at CERFACS. These include work on computational kernels for linear algebra, the solution of sparse systems by both direct and iterative methods, and the porting of industrial codes to parallel computers.

Keywords: CERFACS, parallel algorithms, numerical linear algebra, BLAS kernels, sparse matrices, direct methods, iterative methods, industrial codes, porting.

AMS(MOS) subject classifications: 65F05, 65F10, 65F15, 65F50.

---

[1] Also at: ENSEEIHT-IRIT, 2 rue Camichel, 31071 Toulouse Cedex, France. dayde@enseeiht.fr

[2] Also at: Central Computing Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England. isd@letterbox.rl.ac.uk

[3] Current reports available by anonymous ftp from orion.cerfacs.fr (internet 138.63.200.33) in the directory "pub/algo/pubs".

Parallel Algorithm Project
CERFACS
42 Ave G Coriolis
31057 Toulouse Cedex
France

August 3, 1994.

Contents

The CERFACS experience

M. J. Daydé and I. S. Duff

ABSTRACT

We consider various aspects of the European Centre, CERFACS, located in Toulouse, SW France. We discuss its history and structure which we believe to be unique among European or indeed world scientific centres. We then describe some of the current research activities of the Parallel Algorithm Group at CERFACS. These include work on computational kernels for linear algebra, the solution of sparse systems by both direct and iterative methods, and the porting of industrial codes to parallel computers.

1 Introduction

This short paper serves two functions. One is to describe the origins and structure of CERFACS, a European Centre of Scientific Computation located in Toulouse, SW France. This is discussed in Section 2.

One of the projects within CERFACS is the Parallel Algorithm Project and the second purpose of this paper is to outline a few examples of the principal research themes explored by this group. These include the solution of sparse equations using direct and iterative techniques and the porting of industrial codes to parallel computers. These topics are discussed in Sections 3 to 5 respectively. In our concluding remarks in Section 6, we comment briefly on likely future developments.

2 What is CERFACS?

CERFACS, Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, was first proposed in 1982 and a Scientific Committee consisting of several distinguished researchers in the area of scientific computation spent some years discussing possible ways of organizing and launching such a centre. It is important to stress that the European nature of the Centre was the principal theme from the outset and remains so until this day.

It was soon clear that the European Commission was not interested in adding to the number of Laboratories that it ran so, when CERFACS was launched in October 1987, it was supported by a consortium of industry, government, and academic institutions, as it is today. Indeed it is this amalgamation that comprises the governing body of CERFACS (the partners) that, together with its European outlook, makes CERFACS unique. The collaboration between the three sectors is woven into the fabric of CERFACS and gives a motivation and direction that it often lacking in more academic research centres.

The initial President of the Administrative Council was Jacques Louis Lions who passed the baton in 1991 to Jean Pierre Contzen, a senior official of the European Commission who was responsible for the JRC Laboratories. Our current Director is Roland Glowinski and Robert Dautray is the President of the Scientific Council. Pierre-Henri Cros, who was

one of the main instigators of the CERFACS concept, is Deputy Director and heads the administration.

The scientific life of CERFACS is based on a Project structure. CERFACS began with three projects in 1987. The Parallel Algorithm Project (led by Iain Duff) and projects in Computational Aerodynamics (led by Art Rizzi and Hieu Ha Minh) and Turbulence (led by Maurice Meneguzzi). In 1991, these two latter groups were combined into a CFD Project, led by Thierry Poinsot. Currently about one third of the researchers at CERFACS work in each of these two projects. The other main projects are in Climate Modelling (led by Olivier Thual) and Post-Processing and Visualization (led by Jacques David). A new project in electromagnetism (led by Patrick Joly) started last year. Within the Parallel Algorithm Project, there are research groups in qualitative computation (led by Françoise Chatelin), shape optimization (led by Mohamed Masmoudi), and computational chemistry (led by Feng Shou Zhang).

The current partners of CERFACS are Aérospatiale, CNES, Conseil Régional Midi-Pyrénées, INPT, INRIA, Matra-Marconi Space, Météo France, and UPS. We have had a long and close relationship with the local science and engineering school ENSEEIHT (part of INPT) and some of the work that we discuss in the following has been done in collaboration with scientists from there. Because of lack of space, we concentrate on a few areas of research but strongly suggest that the interested reader obtains a copy of our most recent report to the Scientific Council for further information on the work of the Parallel Algorithm Project and the other projects at CERFACS [6].

3 Computational kernels

The building blocks for much of our work, both in the solution of sparse as well as full systems and in more complicated areas of scientific computation, are the computational kernels known as the BLAS. For reasons of efficiency, we are interested in the higher level BLAS, in particular the Level 3 BLAS [11] which include kernels like the matrix-matrix multiply routine _GEMM. Indeed, in [10] and [16], it is shown how all the Level 3 BLAS routines can be designed for high performance using the _GEMM kernel so it is reasonable to only consider the performance of that routine here.

In conjunction with ENSEEIHT, we have developed highly tuned _GEMM routines for a wide range of computers. In Table 3.1, we show the performance of the matrix-matrix product DGEMM from the Level 3 BLAS on the workstations IBM RS/6000-950, HP 730 and SUN SPARC 10/41, obtained using a block version of the BLAS designed for efficiency on RISC processors. We also show the results from a standard Fortran implementation of DGEMM.

This blocked implementation of the BLAS, in this case DGEMM, gives a gain in performance of greater than a factor of two compared with the standard Fortran coded version. Furthermore, we have observed that the performance is even better if the matrices are already held in the cache (which was not the case in these experiments). The performance for parallel versions of _GEMM on the BBN TC2000 and the KSR1 are given in Table 3.2 on matrices of order 512. We show results for both single (32-bit) and double (64-bit) precision on the BBN and for normal (64-bit) precision on the KSR1. We see that the matrices are too small to get maximum performance. Indeed, on matrices of order 1536, SGEMM can reach 150 Mflop/s on 24 processors on the BBN TC2000, and it is also possible to get 1320 Mflop/s with 72 processors on the KSR1 for matrices of order 768.

This parallel version of the BLAS has been used to exploit, in a transparent manner, parallelism in codes from the LAPACK Library. We show in Table 3.3 the performance

Workstation	Kernel	Order of matrices			
		32	64	96	128
HP 730	DGEMM standard	13.5	11.8	11.5	11.9
	DGEMM optimized	13.5	21.3	22.3	23.5
IBM RS/6000-950	DGEMM standard	21.8	21.8	22.1	20.6
	DGEMM optimized	65.5	65.5	68.1	59.9
SUN SPARC 10/41	DGEMM standard	13.5	9.7	8.9	9.2
	DGEMM optimized	13.5	17.7	18.8	19.2

Table 3.1: Performance in Mflop/s of a blocked implementation of DGEMM (64-bit precision) on RISC workstations.

Computer	Precision	Uniproc.	Number of processors					
			1	2	4	8	16	24
BBN TC2000	32 bits	7.8	6.6	13.4	26.2	52.1	98.8	124.4
	64 bits	2.7	2.5	4.9	9.7	19.2	37.2	47.0
KSR1	64 bits	27.5	25.4	42.9	81.9	165.4	305.4	418.3

Table 3.2: Performance in Mflop/s of GEMM with matrices of order 512 on a BBN TC2000 and a KSR1.

of LAPACK codes corresponding to blocked **LU** (GETRF) and Cholesky (POTRF) factorizations on the BBN TC2000. Only the parallelism internal to the BLAS has been exploited. The matrices are declared "shared interleaved" on the BBN TC2000 (that is, they are logically shared but physically distributed over the memory of the processors).

	Precision	Number of processors					
		1	2	4	8	16	24
LU	single	5.4	10.1	18.1	31.1	49.1	60.6
	double	2.0	3.7	6.7	11.3	18.8	24.1
Cholesky	single	6.1	11.6	21.4	38.6	67.6	91.3
	double	2.3	4.4	8.3	14.8	25.9	35.9

Table 3.3: Performance in Mflop/s on the BBN TC2000 of LU and Cholesky factorizations on matrices of order 2000.

These results show that most of the possible parallelism in these LAPACK routines has been exploited. We observe that there are better speedups for the Cholesky factorization because it has a greater percentage of Level 3 BLAS operations than the **LU** factorization.

4 Solution of sparse equations

4.1 Direct solution

We first consider the sparse **LU** and **QR** factorizations of square and overdetermined matrices on shared memory multiprocessors. We use a multifrontal approach to design our factorization algorithms. Further background on this approach can be obtained from

the original papers by Duff and Reid ([14], [15]). As is common in sparse elimination, the factorization is split into a symbolic phase, which performs an analysis using only the sparsity pattern of the matrix, and a numerical factorization phase.

In a multifrontal method, the sparse factorization proceeds by a sequence of factorizations on small dense matrices, called frontal matrices. The ordering for the sequence of computations and the frontal matrices are determined by a computational tree where each node represents a full matrix factorization and each edge the transfer of data from child to parent node. We say that the data from the children is *assembled* at the parent node. This tree is determined from the sparsity pattern of the matrix and from a reordering that aims at minimizing the fill-in during the numerical factorization (such as the minimum degree ordering that we use here). During the numerical factorization, eliminations at any node can proceed as soon as those at the child nodes have completed. This will be referred to as *tree parallelism*. Note that the factorization at each node is done using full linear algebra and direct addressing so that we can use the BLAS kernels. All the indirect addressing is confined to the assembly process.

4.1.1 Multifrontal LU factorization

We have developed a parallel multifrontal code for the solution of symmetrically structured unsymmetric equations. The results in this section are from runs with this code, called MUPS, and experimental versions of this code ([1], [2]).

During the **LU** factorization, if we only exploit the tree parallelism, the speedup is very disappointing. The actual speedup depends on the problem but is typically only 2 to 3 irrespective of the number of processors. This poor performance is caused by the fact that the tree parallelism decreases while going towards the root of the tree. Moreover, Amestoy and Duff [2] have observed that typically 75% of the work is performed in the top three levels of the computational tree. It is thus necessary to obtain further parallelism within the large nodes near the root of the tree (so-called *node parallelism*) by using parallel versions of the BLAS in the factorizations within the nodes. When combining both tree and node parallelism the situation becomes much more encouraging and we show typical speedups for a range of computers in Table 4.1. A medium size sparse matrix, BCSSTK15 from the Harwell-Boeing set ([13]), is used to illustrate our discussion. This is a structural analysis matrix of order 3948 with 117816 nonzeros. A minimum degree ordering is used in the analysis and the number of floating-point operations for the factorization is 443 million.

		(1)		(2)	
Computer	nprocs	Mflop/s	(speedup)	Mflop/s	(speedup)
ALLIANT FX/80	8	15	(1.9)	34	(4.3)
IBM 3090E/3VF	3	83	(1.9)	105	(2.4)
IBM 3090J/6VF	6	126	(2.1)	227	(3.8)
CRAY-2	4	316	(1.8)	404	(2.3)
CRAY Y-MP	6	529	(2.3)	1119	(4.8)
CRAY C90	4	1124	(2.2)	1486	(2.9)

Table 4.1: Performance summary of the multifrontal **LU** factorization on matrix BCSSTK15. In column (1) we exploit only parallelism from the tree; in column (2) we combine the two levels of parallelism.

4.1.2 Multifrontal QR factorization

The computational tree for the multifrontal **QR** factorization of a matrix **A** is built from the symbolic factorization of the matrix $\mathbf{A}^{\mathbf{T}}\mathbf{A}$. A detailed description of the multifrontal approach for the **QR** factorization can be found in [3] and [17]. The choice of an adequate strategy is crucial for the memory requirements, the degree of parallelism, and the computational time. We have studied several strategies for the factorization of the frontal matrices at each node. Additionally, techniques similar to those for sparse **LU** factorization have been implemented to improve the performance. In addition to a node parallelism similar to that used for the **LU** factorization, the performance is also increased by relaxing the sparse structure in the original matrix and in the frontal matrices (this causes a reduction in the number of indirect operations and an increase in the block size of the frontal matrices). The performance of a version of the multifrontal **QR** factorization incorporating these features is reported in Table 4.2.

		Number proc.		
Matrix	$(m \times n)$	1	8	Speedup
medium2	(18794×12238)	140.7	28.8	4.88
large2	(56508×34528)	1365.0	220.0	6.21
Nfac90	(31684×8100)	77.0	13.3	5.78
Nfac100	(39204×10000)	111.9	19.7	5.69

Table 4.2: Factorization time (in seconds) of the multifrontal **QR** factorization on the ALLIANT FX/80. Householder transformations are used within the **QR** factorization.

4.1.3 Iterative solution

Quite recently we have started an HCM project on iterative methods in which one of our rôles is to develop and study some novel ideas for preconditioning. However, the Parallel Algorithm Project has a long history of work in iterative methods and one of our earliest investigations, concerning the projection method termed Block Cimmino [4], is now being developed for efficient implementation on distributed memory machines.

Our investigations [5] showed that the use of conjugate gradients as an acceleration technique for the Block Cimmino algorithm could perform very badly when the system was ill-conditioned due to clustering of eigenvalues in the relevant matrix. We have thus studied variants of block conjugate gradients to alleviate this problem and, as a first step to the parallelization of the Block Cimmino code, have developed parallel versions of a stabilized block conjugate gradient algorithm. We show in Table 4.3 some early results of this work. A fuller discussion is given in [12].

Matrix Order	Laplace 4096		LANPRO3 960	
Number of PE's	Time	Speedup	Time	Speedup
1	0.279	-	0.065	-
2	0.143	1.95	0.034	1.89
4	0.071	3.90	0.019	3.37
8	0.038	7.28	0.012	5.36
12	0.028	9.79	0.010	6.29
16	0.024	11.63	0.010	6.24

Table 4.3: Factorization time (in seconds) and speedup for stabilized block conjugate gradient algorithm on two matrices, 5-point discretization of Laplace's equation on a 64×64 grid and matrix LANPRO3 from the Harwell-Boeing Collection.

The overhead of the one processor run above a serial version is negligible and the LANPRO3 problem is clearly not large enough for the larger number of processors. Otherwise the speedups are very encouraging.

5 Porting industrial codes

As we mentioned earlier, some of the partners of CERFACS are industrial companies. We are involved in training exercises and much casual consultancy with these partners and, from time to time, have explicit contracts to investigate numerical and computing aspects of some of their industrial codes. Two recent exercises [7], [9] are discussed and summarized in [8]. As an illustrative example, we show (see Table 5.1) the results obtained on a flutter calculation code (for more details see [8]). This application has been parallelized on a wide range of parallel shared memory machines (ALLIANT FX/80, CONVEX C220, and CRAY-2), virtual shared memory computers (BBN TC2000), and, using PVM, on networks of workstations (IBM RS/6000-320 connected by Ethernet and IBM RS/6000 950, 550, and 530 connected using SOCC).

		Initial version	Modified version	
Computer	No procs	Sequential	Sequential	Parallel
ALLIANT FX/80	8	5100	1717	326 (5.3)
BBN TC2000	19	>30000	2994	202 (14.8)
CONVEX C220	2	1919	216	119 (1.8)
CRAY-2	4	624	45	15 (3.0)
IBM RS/6000-320	8	>6000	622	93 (6.7)
Cluster RS/6000	3	>2000	235	89 (2.6)

Table 5.1: Execution time (in seconds) and speedups obtained on a flutter calculation.

6 Concluding comments

It has been our intention to give a flavour for some of the work being done at CERFACS but, of course, we have only just been able to scratch the surface and interested readers are strongly advised to consult a report of CERFACS to the Scientific Council [6].

This last year has seen a very marked growth in the involvement of CERFACS in European HPCN projects. We now have nearly twenty separate grants, principally in the ESPRIT (HPCN) and HCM programmes and, through these interactions with many laboratories, companies, and universities throughout Europe, we have established research collaborations with most countries in Europe. Most of these projects run for two or three years but we are already applying for more funding to strengthen even further the European nature of CERFACS.

Within the Parallel Algorithm Project, two of our grants are directly concerned with research on sparse solvers so clearly this work will continue although perhaps with more emphasis on iterative methods and preconditioning. We hope to work more on nonlinear problems and on closer interaction between the work on linear algebra and the study of conditioning and reliability in the qualitative computing group. We have also proposals to investigate the use of our solution techniques and others in various application areas including computational fluid dynamics to which end some proposals for funding are in collaboration with the CFD Project.

7 Acknowledgments

We are very grateful to our many colleagues at CERFACS and ENSEEIHT for the collaborations that produced the work discussed in this paper. Their contributions are evident by consulting the referenced papers.

References

[1] P. R. Amestoy and I. S. Duff. Vectorization of a multiprocessor multifrontal code. *Int. J. of Supercomputer Applics.*, 3:41–59, 1989.

[2] P. R. Amestoy and I. S. Duff. Memory allocation issues in sparse multiprocessor multifrontal methods. *Int. J. of Supercomputer Applics.*, 7:64–82, 1993.

[3] P. R. Amestoy, I. S. Duff, and C. Puglisi. Multifrontal **QR** factorization in a multiprocessor environment. Technical Report To appear, CERFACS, Toulouse, France, 1994.

[4] M. Arioli, I. S. Duff, J. Noailles, and D. Ruiz. A block projection method for sparse equations. *SIAM Journal on Scientific and Statistical Computing*, 13:47–70, 1990.

[5] M. Arioli, I. S. Duff, D. Ruiz, and M. Sadkane. Block Lanczos techniques for accelerating the Block Cimmino method. Technical Report TR/PA/92/70, CERFACS, Toulouse, France, 1992.

[6] CERFACS. Cerfacs Scientific Report 1992 - 1993. Technical report, CERFACS, Toulouse, France, 1993.

[7] J. L. Charles, M. J. Daydé, A. Petitet, L. Prévost, and E. Simmonet. Evaluation de calculateurs multiprocesseurs pour les logiciels et bibliothèques scientifiques du CNES: Rapport final. Technical report, CERFACS, Toulouse, France, 1993.

[8] M. J. Daydé and I. S. Duff. Porting industrial codes and developing sparse linear solvers on parallel computers. Technical Report TR/PA/94/01, CERFACS, Toulouse, France, 1994.

[9] M. J. Daydé, I. S. Duff, J. Y. L'Excellent, and L. Giraud. Evaluation d'ordinateurs vectoriels et parallèles sur un jeu de programmes représentatifs des calculs à la division avions de l'Aérospatiale : Rapport final. Technical Report FR/PA/93/19, CERFACS, Toulouse, France, 1993.

[10] M. J. Daydé, I. S. Duff, and A. Petitet. A parallel block implementation of Level 3 BLAS kernels for MIMD vector processors. *ACM Transactions on Mathematical Software*, To appear:??–??, 1994.

[11] J. J. Dongarra, J. J. Du Croz, I. S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 16:1–17, 1990.

[12] L. A. Drummond, I S. Duff, and D. Ruiz. A parallel distributed implementation of the block conjugate gradient algorithm. Technical Report (to appear), CERFACS, Toulouse, France, 1994.

[13] I. S. Duff, R. G. Grimes, and J. G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection (Release I). Technical Report RAL 92-086, Rutherford Appleton Laboratory, 1992.

[14] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Transactions on Mathematical Software*, 9:302–325, 1983.

[15] I. S. Duff and J. K. Reid. The multifrontal solution of unsymmetric sets of linear systems. *SIAM Journal on Scientific and Statistical Computing*, 5:633–641, 1984.

[16] B. Kågström, P. Ling, and C. Van Loan. Portable high performance GEMM-based Level-3 BLAS. In *To appear Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 1993.

[17] C. Puglisi. **QR** *factorization of large sparse overdetermined and square matrices using the multifrontal method in a multiprocessor environment.* PhD thesis, Report TH/PA/93/33, CERFACS, Toulouse, France, 1993.