RAL-TR-98-028

Numerical Analysis Group Progress Report January 1996 - December 1997

Iain S. Duff (Editor)

ABSTRACT

We discuss the research activities of the Numerical Analysis Group in the Department for Computation and Information at the Rutherford Appleton Laboratory for the period January 1996 to December 1997.

Keywords: sparse matrices, numerical linear algebra, large-scale optimization, Fortran, Harwell Subroutine Library, HSL

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

Current reports available by anonymous ftp from matisa.cc.rl.ac.uk in the directory "pub/reports". This report is in file duRAL98028.ps.gz.

Department for Computation and Information
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

March 26, 1998

Contents

Personnel in Numerical Analysis Group

Staff

Iain Duff.
Group Leader. Sparse matrices and vector and parallel computers and computing.

Nick Gould.
Optimization and nonlinear equations particularly for large systems.

Petr Plecháč (from November 1st 1997).
PARASOL Project.

John Reid.
Sparse matrices and development of the Fortran programming language.

Jennifer Scott.
Sparse linear systems and sparse eigenvalue problems. Seminar organization.

Visitors and Attached Staff

Jerome Décamps (ENSEEIHT, Toulouse) Matrix stretching and optimization.

Jean-Yves L'Excellent (CERFACS) PARASOL.

Mike Hopper (Consultant) Support for Harwell Subroutine Library and for TSSD.

John Lewis (Boeing) Sparse matrices and software.

Karl Meerbergen (University of Utrecht) Preconditioning techniques for sparse eigenproblems.

Jorge Moré (Argonne National Laboratory) Optimization.

Jorge Nocedal (Northwestern University, Illinois) Optimization.

Massimo Roma (Rome) Optimization.

Philippe Toint (University of Namur) Optimization.

1 Introduction (I. S. Duff)

This report covers the period from January 1996 to December 1997 and describes work performed by the Numerical Analysis Group within the Department for Computing and Information at the Rutherford Appleton Laboratory.

The details of our activities are documented in the following pages. These words of introduction are intended merely to provide an introduction and additional information on activities that are not appropriate for the detailed reports.

This past period has seen substantial organizational changes. Not only did the Department change its name again but the merger of Daresbury Laboratory with the Rutherford Appleton Laboratory to create CCLRC (Central Laboratory for the Research Councils) brought us into closer contact with the Theory Division at Daresbury Laboratory, led by Professor Paul Durham. Additionally, it was felt by an SLA (Service Level Agreement) review panel that we were too research oriented to be entirely funded through the SLA route so we were asked to apply to EPSRC for a research grant to cover 85% of our funding. I am delighted to report that our application was successful, and indeed that the referees' reports were very supportive of our work and continued existence.

The support and development of the Harwell Subroutine Library (HSL) continues to be one of our major activities. There have, however, been no releases of either HSL or the NAG-marketed Harwell Sparse Matrix Library during the period of this report and so Section 7.2 is rather sparse. As normal between releases, there are only a few new routines in final form. We do not explicitly describe partly-completed routines although their functionality may sometimes be deduced from reading our research reports, many of which result in software for HSL or LANCELOT. The HSL marketing effort from AEA Technology has been much more stable than in earlier years and Scott Roberts and Richard Lee have been ably assisted by Virginia Ifould and then Maria Woodbridge. As well as facilitating our contact with AEA Technology, the stability has helped boost HSL sales. Although we had no releases of HSL, we still benefited greatly from the consultancy of Mike Hopper who helped us both in typesetting and the ongoing commitment to higher software standards. Our contacts with Ian Jones and his CFDS group at Harwell continue on a very informal basis, and we have continued our collaboration with Andrew Cliffe on the solution of finite-element equations by frontal methods (Section 2.4).

We continue to maintain close links with the academic community in Britain and abroad. Iain and John are Visiting Professors at Strathclyde University and RMCS Shrivenham, respectively, Nick is a visiting Fellow at Shrivenham, and John and Jennifer have lectured in M.Sc. courses at Reading University. All members of the Group gave contributed talks at the Dundee Numerical Analysis meeting in 1997, Iain is co-hosting a visitor from China (Zhong-Zhi Bai) with Andy Wathen from Oxford on an EPSRC grant,

and Nick and Iain are supervising a CASE student (Carsten Keller), again with Andy Wathen. Nick and Iain were on the jury for the habilitation defence of Michel Daydé in Toulouse, Iain was on the jury for the thesis exams of Jacko Koster and Luiz Carvalho, and Nick was on the jury for the thesis of Jerome Décamps, all in Toulouse.

Most of our visitors stayed only for a short time although the interaction with them has been quite intense. We continue our close association with Oxford University through the Joint Computational Mathematics and Applications Seminar series and have hosted several talks at RAL through that programme (see Section 8).

John has continued to combine his interests in Fortran and sparse matrices giving several talks on these topics during the last two years. He has been very involved through ISO WG5 in influencing the development of Fortran 2000 and has been their main architect of a proposal for the inclusion of exception handling. He discusses this and other related Fortran activities in Section 5. John has attended Fortran meetings in Las Vegas and Vienna, and has given courses on Fortran 90 at Reading University, RAL, and RMCS Shrivenham. His talk on Fortran 90 at RAL was used by the Graphics Group to test their ideas on videoing meetings for the World Wide Web. He has continued with his collaboration on automatic differentiation with groups at Shrivenham and the University of Hertfordshire. John has remained an active member of IFIP Working Group 2.5 and has attended and spoken at their meetings in Oxford and Albuquerque. John gave an invited talk on his Linear Programming work at a meeting in Copenhagen, and he attended conferences in Coeur d'Alene (Idaho) and York.

Nick's collaboration with Conn and Toint continues to expand the theory and practice of large-scale optimization, and they are currently writing a massive tome on trust-region methods. Much of their work is embodied in the LANCELOT package which continues to be in great demand and has had one commercial sale. He still has joint research activities with contacts made during his visit to CERFACS in 1993 and has had an Alliance grant from the British Council to support this activity. He was a co-supervisor of Jerome Décamps, who completed his thesis at ENSEEIHT-IRIT in Toulouse in October 1997. He obtained a British Council-MURST/CRUI grant 1996-1997 to collaborate with the Optimization group at the Università di Roma "La Sapienza". Nick was also involved in a commercial project with ICI paint division in Slough during 1996. Nick was an invited speaker at conferences at Ischia, Lausanne, and York and gave a seminar at Oxford University.

Jennifer has developed several international collaborative projects over the two years, primarily in the computation of sparse eigenvalues. Although she has continued her short-hours working, she remains so productive that it is easy to forget this fact. In addition to her work on eigensystems, she has continued to work on frontal solvers and has been involved in a collaboration (see Section 2.9) that combines both of these areas. Jennifer continues to coordinate our joint seminar series with Oxford University. She was invited to

talk at a summer school in Hamburg and gave an invited talk at a workshop at Argonne, Illinois. She has presented seminars at Argonne, Durham, Liverpool, and Strathclyde and has attended meetings at York and Coeur d'Alene (Idaho).

Petr Plecháč joined the Group at the beginning of November, 1997 to work on the PARASOL Project (see Section 6.2). He had previously been a postdoc with John Ball at Oxford. He has a strong background in analysis but also has considerable computing skills. Petr has been working on the integration of graph partitioning with sparse ordering techniques. He has implemented the PARASOL interface library for the MUMPS solver and has written the test drivers so that the MUMPS code (see Section 2.2) can be used by the PARASOL Library interface (Section 6.3).

Iain still leads a project at the European Centre for Research and Advanced Training in Scientific Computation (CERFACS) at Toulouse in France (see Section 6.1). Iain is an Editor of the IMA Journal of Numerical Analysis, an Honorary Secretary of the IMA, editor of the IMANA Newsletter, chairman of the IMA Programme Committee, chairman of the Adjudicating Committee for the Fox Prize, IMA representative on the CCIAM International Committee that overseas the triennial international conferences on applied mathematics, a member of the International Scientific Programme Committee for ICIAM '99, and is on the Mathematics College of the EPSRC. In high performance computing, he has given tutorials at Supercomputing '97 (San Jose CA) and the SIAM 1997 National Meeting (Stanford CA). Iain has been on the Programme and Organizing Committee for several international meetings and has given frequent expository and survey talks to mathematicians and other scientists. He has given invited talks at PARA'96 (Copenhagen), EUROPAR'96 (Lyon), VECPAR'96 (Porto), AspenWorld'97 (Boston) and invited talks at meetings in Lille, Miskolc (Hungary), Santorini (Greece), Trieste and York. He has presented contributed talks and seminars in Coeur d'Alene (Idaho), MIT, Pontresina (Switzerland), and Strathclyde.

We have tried to subdivide our activities to facilitate the reading of this report. This is to some extent an arbitrary subdivision since much of our work spans these subdivisions. Our main research areas and interests lie in sparse matrix research, nonlinear algebra and optimization, and numerical linear algebra. Work pertaining to these areas is discussed in Sections 2 to 4, respectively. The work by John on Fortran is discussed in Section 5, and we group some miscellaneous topics in Section 6. Much of our research and development results in high quality advanced mathematical software for the Harwell Subroutine Library, and we report on our computer infrastructure and HSL developments in Section 7. Lists of seminars (in the joint series with Oxford), technical reports, and publications are given in Sections 8, 9, and 10, respectively. Current information on the activities of the Group and on Group members can be found through page `http://www.dci.clrc.ac.uk/Group.asp?DCICSENAG` of the World Wide Web.

2 Sparse matrices

2.1 Rutherford-Boeing Sparse Matrix Collection (I. S. Duff, R. G. Grimes, and J. G. Lewis)

The change in the name of this activity from the Harwell-Boeing Sparse Matrix Collection reflects the fact that Iain has been away from Harwell for nearly eight years and also serves to signify rather major changes to the organization and design of the Collection. Additionally, it should help to avoid confusion between this project and the Harwell Subroutine Library.

If one is defining a standard, albeit a de facto one, then it is very important to get the design correct and to allow for possible future extensions. It is concerns of this kind that has preoccupied us in the recent development of the Rutherford-Boeing Sparse Matrix Collection (RBSMC).

The principal change from the Harwell-Boeing Sparse Matrix Collection has been that the format has been extended so that the matrix is held in one file but other data such as starting guesses, exact solutions, eigenvalues and vectors, singular values and vectors, permutations, partitions, and geometric data are held in separate files. The formats are also more restricted so that all the data files can be easily read from C programs.

Additional features that have been added more recently include the ability to handle rectangular element matrices, Laplacian vectors, Schur bases, and an auxiliary matrix values file. Documentation on the RBSMC has been completed (Duff, Grimes and Lewis, 1997) and discussions have continued with NIST (`http://math.nist.gov/MatrixMarket/`) on the integration of these structures within the Matrix Market database.

The RBSMC has also been adopted by the PARASOL Project (see Section 6.2) as the platform for data exchange (Supalov, 1998) and all the test examples from the PARASOL end users are held in this format.

References

I. S. Duff, R. G. Grimes, and J. G. Lewis. The Rutherford-Boeing sparse matrix collection. Technical Report RAL-TR-97-031, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

A. Supalov. The Rutherford-Boeing File Formats and the PARASOL Test Driver. Version 2.1. January 9, 1998.

2.2 MUMPS - a distributed memory multifrontal solver (P. R. Amestoy, I. S. Duff and J.-Y. L'Excellent)

We are involved in the development of direct solvers within the EU LTR Project PARASOL (Sections 6.2 and 6.3). This work is the result of a collaboration with ENSEEIHT and CERFACS in Toulouse, France. It is based on the shared memory parallel multifrontal code `MA41` developed by Amestoy and Duff (1993) that was extended by Espirat (1996) to a message passing version based on PVM.

First, an MPI/F90 version of this code was developed (known as MUMPS Version 1.0), and was released in May, 1997. This version is for unsymmetric matrices and allows pivoting, but only exploits parallelism of the elimination tree. It can be used with limited functionality within the PARASOL Interface (Supalov, 1998).

Most of the work since Version 1.0 has consisted in developing a code to allow for better parallelism, to solve larger problems, and to improve the robustness, error handling, and memory management. Specific changes include:

- Introduction of parallelism inside the nodes, and use of ScaLAPACK at the root node, with a fully parallel distributed assembly processes,
- Development of a module to manage asynchronous messages of various types,
- Improvement of the mapping to balance both flops and memory, and
- Distribution of the arrowheads (the reordered original matrix).

More details on technical aspects of this work can be found in Amestoy, Duff and L'Excellent (1998). The version of the code described in that report, called MUMPS Version 2.0, will be released at the end of January 1998, and has much better scalability than Version 1.0.

We show results of a run on one of the PARASOL test problems in Table 2.1. This problem came from MSC and is from a finite-element model of a crankshaft. The super-linear speedup is quite a common effect and reflects the fact that the distribution of data over more processors allows better memory management and less penalties from paging.

Future work will include an interface to graph partitioning packages, a more efficient code for symmetric positive-definite matrices, an interface to handle unassembled finite-element problems directly, and facilities for holding factors out-of-core.

Matrix	CRANKSEG1
Order	52,804
Entries	10,614,210
Entries in factors	80,148,400
Analysis time	36
Factorization time	
Number of Processors	
16	626
24	320
33	234

Table 2.1: Execution times in seconds on an IBM SP-2

References

P. R. Amestoy and I. S. Duff. Memory management issues in sparse multifrontal methods on multiprocessors. *Int. J. Supercomputer Applics*, **7**, 64–82, 1993.

P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. MUMPS MUltifrontal Massively Parallel Solver, Version 2.0. Technical Report TR/PA/98/02, CERFACS, 1998.

V. Espirat. Développement d'une approche multifrontale pour machines à mémoire distribuée et réseau hétérogène de stations de travail. Technical Report Rapport de stage de 3ieme année, ENSEEIHT-IRIT, Toulouse, France, 1996.

A. Supalov (editor). PARASOL Interface Specification. Version 2.1. January 9, 1998.

2.3 Further developments of an unsymmetric multifrontal method (T. A. Davis and I. S. Duff)

Partly because of some referees' comments on our earlier work, we have conducted more extensive experimentation and testing of our unifrontal/multifrontal method for the direct solution of large sparse linear equations when the coefficient matrix does not have a symmetric structure. In particular, we have compared our code with several other sparse codes on very large unsymmetric matrices collected by Tim Davis (available at `http://www.cise.ufl.edu/~davis/sparse/`).

Our findings have been very much "horses for courses". The advantages of incorporating the unifrontal element into our sparse unsymmetric multifrontal code are clear and

the resulting code is very competitive with other direct codes if the matrices are very unsymmetric in pattern. For matrices which are less unsymmetric in pattern `MA41` (Amestoy and Duff, 1989) is usually better, while for unsymmetric matrices that do not fill-in much during factorization, `MA48` (Duff and Reid, 1996) can be best. A description of our algorithm and our experiments are discussed in the report by Davis and Duff (1997) which will shortly appear in the ACM Transactions of Mathematical Software.

References

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

T. A. Davis and I. S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. Technical Report RAL-TR-97-046, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

I. S. Duff and J. K. Reid. The design of MA48, a code for the direct solution of sparse unsymmetric linear systems of equations. *ACM Transactions on Mathematical Software*, **22**(2), 187–226, 1996.

2.4 Performance issues for frontal schemes on a cache-based high performance computer (K. A. Cliffe, I. S. Duff, and J. A. Scott)

The frontal solution scheme is a technique for the direct solution of the linear systems of equations

$$\boldsymbol{AX} = \boldsymbol{B}, \tag{2.1}$$

where the $n \times n$ matrix $\boldsymbol{A}$ is large and sparse. $\boldsymbol{B}$ is an $n \times nrhs$ ($nrhs \geq 1$) matrix of right-hand sides and $\boldsymbol{X}$ is the $n \times nrhs$ solution matrix. The method is a variant of Gaussian elimination and involves the factorization of a permutation of $\boldsymbol{A}$ which can be written as

$$\boldsymbol{A} = \boldsymbol{PLUQ}, \tag{2.2}$$

where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are permutation matrices, and $\boldsymbol{L}$ and $\boldsymbol{U}$ are lower and upper triangular matrices, respectively. The code `MA42` developed by Duff and Scott (1996a) for the Harwell Subroutine Library uses a frontal scheme for solving systems of the form (2.1) with $\boldsymbol{A}$ unsymmetric. `MA42` includes an option which allows the assembled matrix $\boldsymbol{A}$ to be input by rows. However, as illustrated by Duff and Scott (1996b), the power of the frontal

scheme is more apparent when the matrix $\boldsymbol{A}$ comprises contributions from the elements of a finite-element discretization. That is, we can express $\boldsymbol{A}$ as the sum of elemental matrices

$$\boldsymbol{A} = \sum_{l=1}^{m} \boldsymbol{A}^{(l)}, \tag{2.3}$$

where $\boldsymbol{A}^{(l)}$ is nonzero only in those rows and columns that correspond to variables in the l-th element.

The aim of this study was to look at the performance of a frontal solver on a machine where data must be in the cache before arithmetic operations can be performed on it. An example of such a machine is a Silicon Graphics Power Challenge machine. One way of achieving efficiency in the solution of linear equations is through the use of the Basic Linear Algebra Subprograms (BLAS). The BLAS are subdivided into three levels. In each succeeding level (from 1 to 3) more operations are performed for each data movement. Thus the best performance is obtained by the Level 3 BLAS and for efficiency on modern computers, maximum use should be made of Level 3 BLAS. In this study, we first showed how the computation in `MA42` is organized to exploit `_GEMM`, the Level 3 BLAS kernel that implements **dense** matrix-matrix multiplication. We then looked at how the frontal algorithm may be modified to obtain a factorization which requires a larger number of floating-point operations but which is richer in Level 3 BLAS. This was achieved by introducing a parameter r_{min} to control the minimum number of pivots that are eliminated at once. Using a range of practical problems, we found that, on the Silicon Graphics Power Challenge, using $r_{min} \geq 1$ leads to good performance in terms of Mflops.

We also performed experiments on an IBM RS/6000 and on a single processor of a CRAY J932. Our experience was that the implementation of the frontal method which uses only pivot blocks of size 1 does reasonably well on vector machines but performs poorly on cache-based machines. For problems in elemental form, we found that the most significant improvement in performance comes from not forcing all blocks to be of size 1, but for some assembled problems, in which there is normally only one pivot available after each assembly, better results are obtained if $r_{min} > 1$.

Clearly, it is important that we implement our algorithms for solving large sparse systems to make effective use of machines which have a hierarchical memory structure. The techniques which we used in this study for making better reuse of data in the cache are applicable to other direct solvers.

References

I.S. Duff and J.A. Scott. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Transactions on Mathematical Software*, **22**(1), 30–45, 1996a.

I. S. Duff and J. A. Scott, A comparison of frontal software with other sparse direct solvers, Technical Report RAL-TR-96-102 (Revised), Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1996b.

K. A. Cliffe, I. S. Duff and J. A. Scott, Performance issues for frontal schemes on a cache-based high performance computer, Technical Report RAL-TR-97-001, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997. To appear in *Int J. Numerical Methods in Engineering.*

2.5 Exploiting zeros in frontal solvers (J. A. Scott)

We are interested in using the frontal method to solve systems of linear equations

$$\boldsymbol{AX} = \boldsymbol{B}, \tag{2.4}$$

where the $n \times n$ matrix $\boldsymbol{A}$ is large and sparse. $\boldsymbol{B}$ is an $n \times nrhs$ ($nrhs \geq 1$) matrix of right-hand sides and $\boldsymbol{X}$ is the $n \times nrhs$ solution matrix. The frontal method is a variation of Gaussian elimination that interleaves an assembly phase and an elimination phase, thereby allowing operations to be confined to a relatively small matrix, termed the *frontal matrix*, that is usually regarded as dense. The efficiency of the method comes form the use of high-level Basic Linear Algebra Subprograms (BLAS). In the Harwell Subroutine Library (HSL), the code `MA42` (Duff and Scott, 1996a) implements the frontal method for unsymmetric systems. `MA42` allows the matrix $\boldsymbol{A}$ to be input by elements or equations but is principally structured for finite-element problems.

Experiments by Duff and Scott (1996b) found that, for some problems, `MA42` can require substantially more operations and storage for the factors than other HSL codes for solving large sparse systems. In particular, the equation entry to `MA42` was found to be inefficient if the equations were poorly ordered. Closer examination revealed that, in this case, a large number of zeros are held explicitly within the factors. The aim of this work was to reduce the number of zeros within the factors, while not compromising the efficiency of the code when applied to well-ordered problems. Reducing the zeros reduces the operation count and the storage required for the matrix factors as well as the time needed for using the factors to solve for different right-hand sides $\boldsymbol{B}$.

Our study has led to two proposed future modifications to our frontal solver. The first involves exploiting zeros within the frontal matrix and the second concerns the more efficient treatment of static condensation variables (variables that are internal to an element or belong to a single equation). We found that we are able to exploit zeros without altering the internal data structures of `MA42`. The overheads are the search for zeros and, once found, additional row and column permutations. For element entry, efficient static condensation requires an additional array of length the number of variables in the incoming element

since, if off-diagonal pivoting is used (the default), it is necessary to hold both row and column indices. We anticipate that, in a future release of the Harwell Subroutine Library, `MA42` will be superseded by a new frontal code which will use an additional array for static condensations.

Numerical experiments using a range of practical problems have shown that the proposed modifications can substantially improve the performance of the frontal solver. This is illustrated in Table 2.2. The first three test problems are unassembled element problems and the second three use the equation entry to the frontal solver.

Identifier	Zeros exploited	Real storage (Kwords)		Integer storage (Kwords)	Factorize time (seconds)	Solve time (seconds)
TRDHEIM	N	6686	(1642)	535	51.9	3.8
	Y	5048	(4)	352	50.3	2.8
AEAC6398	N	1474	(589)	290	6.3	0.9
	Y	906	(21)	276	5.7	0.5
AEA87000	N	32114	(11608)	4432	192.9	22.2
	Y	20702	(195)	2553	185.2	14.0
WEST2021	N	545	(512)	229	1.1	0.44
	Y	60	(27)	20	0.5	0.07
WANG3	N	38480	(12792)	37504	971.4	48.1
	Y	25687	(0)	25068	956.8	34.1
ONETONE2	N	24246	(21019)	13102	100.4	24.8
	Y	4356	(700)	1311	46.8	3.3

Table 2.2: The effect of exploiting zeros in the front (Sun Ultra 1). The figures in parentheses are the number of zeros (in thousands) which are held explicitly in the factors.

References

I.S. Duff and J.A. Scott. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Transactions on Mathematical Software*, **22**(1), 30–45, 1996a.

I. S. Duff and J. A. Scott, A comparison of frontal software with other sparse direct solvers, Technical Report RAL-TR-96-102 (Revised), Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1996b.

J. A. Scott, Exploiting zeros in frontal solvers, Technical Report RAL-TR-97-041, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

2.6 MA62: A frontal code for positive-definite symmetric systems arising from finite-element applications (I. S. Duff and J. A. Scott)

The code MA42 (Duff and Scott, 1996) is a general frontal code for solving sparse unsymmetric linear systems of equations. Since its inclusion in the Harwell Subroutine Library, MA42 has been used to solve a wide range of practical problems, including finite-element problems for which the system matrix is symmetric and positive definite. However, apart from offering an option of restricting pivoting to the diagonal, MA42 does not exploit symmetry or positive definiteness and, as a result, the code is more expensive in terms of both storage requirements and operation counts than it need be for this class of problems. Our goal was to design and develop a frontal code specifically for the efficient solution of sparse positive-definite symmetric systems from finite-element applications with $\boldsymbol{A}$ in element form $\boldsymbol{A} = \sum_{k=1}^{m} \boldsymbol{A}^{(l)}$. The new code is called MA62.

MA62 has three distinct phases.

1. An analysis phase that comprises a prepass and a symbolic factorization. The prepass determines in which element each variable appears for the last time and thus when a variable is fully summed and can be eliminated. The symbolic factorization uses the information from the prepass to determine the amount of real and integer storage required for the factorization.

2. A factorization phase in which the information from the analysis phase is used in the factorization of the matrix. In addition, if element right-hand sides $\boldsymbol{B}^{(l)}$ are specified, the equations $\boldsymbol{AX} = \boldsymbol{B}$ with right-hand side(s) $\boldsymbol{B} = \sum_{k=1}^{m} \boldsymbol{B}^{(l)}$ are solved.

3. An optional solution phase which uses the factors produced by the factorization phase to solve rapidly for further right-hand sides $\boldsymbol{B}$.

The interface to the user is through reverse communication with control returned to the calling program each time an element needs to be input. This provides the user with flexibility when choosing how to hold the element matrices. In particular, the user may choose to generate an element only when it is required. This is useful when solving problems which are too large to allow all the elements to be stored at once.

The user may hold the matrix factors in direct access files. MA62 uses two direct access files, one for the reals in the factors and one for the indices of the variables in the factors. Use of direct access files is unnecessary if there is sufficient in-core space for the factors.

For efficiency, Level 3 BLAS are used in performing the numerical factorization and in the solution phase. To make better use of BLAS, albeit at the cost of additional operations, eliminations are delayed until a minimum number of pivots are available (see

also Section 2.4). Also for efficiency, MA62 attempts to exploit zeros within the frontal matrix (see also Section 2.5).

Extensive numerical experiments have been performed with the new code. Some results are given in Tables 2.3 and 2.4. Our experience has been that, as well as needing

Identifier	Code	Factor Storage (Kwords)		Factor ops ($*10^6$)
		Real	Integer	
TRDHEIM	MA42	6663	533	866.5
	MA62	2154	110	502.1
CRPLAT2	MA42	12915	2116	4980.1
	MA62	6490	370	2674.6
OPT1	MA42	16674	1194	11047.1
	MA62	7984	341	5710.9
TSYL201	MA42	20905	1020	10723.5
	MA62	10405	483	5542.2
RAMAGE02	MA42	41792	3495	55870.1
	MA62	20996	851	28523.2

Table 2.3: A comparison of the operation count and storage requirements for MA42 and MA62 on symmetric positive-definite unassembled finite-element systems.

approximately half the real storage for the matrix factors as the general frontal code MA42, the new code can be more than twice as fast as MA42. Compared with other HSL codes for symmetric positive-definite systems, we have seen that the frontal method can provide a very powerful approach for the solution of large sparse systems (see Section 2.7), although our conclusions may have to be modified when multifrontal and variable band codes for positive-definite systems that exploit the Level 3 BLAS become available.

References

I.S. Duff and J.A. Scott. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Transactions on Mathematical Software*, **22**(1), 30–45, 1996.

I. S. Duff and J. A. Scott, MA62 – a new frontal code for sparse positive-definite symmetric systems from finite-element applications, Technical Report RAL-TR-97-012, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

Identifier	Code	Time (seconds)			
		Analyse	Factorize	Solve	
				nrhs = 1	nrhs = 10
TRDHEIM	MA42	0.6	8.8	0.49	2.00
	MA62	0.6	5.7	0.33	1.42
CRPLAT2	MA42	1.2	45.3	1.13	4.78
	MA62	1.2	20.3	0.62	2.60
OPT1	MA42	0.8	76.0	0.91	3.65
	MA62	0.8	37.0	0.70	2.80
TSYL201	MA42	0.7	71.8	1.00	3.88
	MA62	0.7	38.8	0.90	3.60
RAMAGE02	MA42	1.4	389.6	2.29	9.32
	MA62	1.4	175.7	1.66	6.58

Table 2.4: A comparison of MA42 and MA62 on symmetric positive-definite unassembled finite-element systems (single processor CRAY J932).

2.7 A comparison of frontal software with other HSL direct solvers (I. S. Duff and J. A. Scott)

In recent years, a number of packages for the direct solution of sparse linear systems have been included within the Harwell Subroutine Library (HSL). The aim of this study was to compare the performance of the sparse HSL frontal codes MA42 and MA62 against other HSL sparse direct solvers. We looked at the case of assembled and unassembled systems for both symmetric and unsymmetric matrices and used a wide range of problems for real engineering and industrial problems in our tests. To try and draw general conclusions, we performed experiments using a single processor of a CRAY J932, an IBM RS/6000 550 and a DEC 7000.

For symmetric positive-definite problems, the frontal solver MA62 was compared with the multifrontal code MA27 and the variable-band solver VBAN (the prototype for a new HSL code MA55). Results for a subset of our test problems are given in Table 2.5. The first three problems are elemental problems and the last problem is assembled. Our numerical experiments showed that the frontal code performs well on unassembled finite-element problems, particularly in the analysis and numerical factorization phases. The benefit of holding the factors out-of-core in order to reduce the maximum amount of in-core storage is evident. However, storage for the factors is usually significantly greater for the frontal method than the multifrontal method and this is reflected in the much poorer times for subsequent solution, although MA62 is more efficient if multiple right-hand sides are being solved at the same time.

Identifier	Code	Time (seconds)			Factor ops ($*10^6$)	Storage (Kwords)	
		Analyse	Factorize	Solve		In-core	Factors
TRDHEIM	MA27	10.8	17.7	0.27	211.0	2893	2002
	VBAN	15.3	19.6	0.39	459.0	798	2958
	MA62	0.6	5.6	0.33	502.1	130	2262
OPT1	MA27	11.9	77.1	0.32	3648.9	7741	5975
	VBAN	20.7	74.2	0.86	4116.5	3315	7215
	MA62	0.8	37.0	0.70	5523.8	990	8325
CRPLAT2	MA27	5.3	40.2	0.30	1623.8	4554	3815
	VBAN	9.4	54.9	0.77	2475.8	2276	6406
	MA62	1.2	20.3	0.62	2624.2	302	6527
BCSSTK15	MA27	2.7	7.1	0.07	219.4	951	788
	VBAN	1.0	7.0	0.08	230.7	524	904
	MA62	2.9	3.4	0.11	267.3	154	1013

Table 2.5: A comparison of MA27, VBAN and MA62 on symmetric positive-definite systems (single processor CRAY J932).

For unsymmetric matrices in assembled form, we compared the equation entry of MA42 with the HSL codes MA38, MA41, and MA48. MA38 is based on a combined unifrontal/multifrontal algorithm, MA41 is a multifrontal code, and MA48 is a general sparse code using Gaussian elimination. We found that no one code is clearly better than the others. The choice of code is dependent on the problem being solved. For problems with a nearly symmetric structure, MA41 generally has the fastest factorize time and, by using MC40 to order the rows, MA42 factorizes the matrix more rapidly than both MA38 and MA48. However, the frontal code has the disadvantage of generally producing many more entries in the factors than the other codes. For problems which are far from symmetric in structure, MA38 or MA48 generally perform well, with MA48 being particularly suitable for very sparse problems.

For elemental problems, we compared MA42 directly with the multifrontal code MA46, which uses element input. We also assembled the elements and solved the resulting assembled system using MA38, MA41, and MA48. Our results showed that for these problems it is advantageous to use a code that accepts input by elements and, in general, MA46 had the fastest factorization time, while MA42 had the fastest analysis time and required the least in-core storage.

Our experiments in other environments suggested that it is important to take account of sparsity within the Level 3 BLAS implementation, that the performance of the out-of-core frontal schemes are significantly affected by the efficiency of the i/o, and that it is

important to exploit machine characteristics, such as cache, for efficient implementation.

Our overall conclusions were that frontal codes can be a very powerful approach for the solution of unassembled finite-element problems when a good ordering can be found. In this case, although other approaches may result in much less fill-in, the frontal code is often better in terms of the analysis time and, if the factors are held in direct access files, is far superior in terms of main memory. Indeed, in some cases, this reduction in main memory requirement meant that it was feasible to solve a problem with our frontal schemes which could not be solved by other methods. This has indicated to us the desirability of developing out-of-core versions of some of the HSL multifrontal codes. For most assembled problems, the use of approaches other than the frontal method might be better and, if a good ordering is not available, frontal methods can perform badly.

Finally, we found that the performance of some HSL codes can be very dependent on the machine and on some of their parameters, including the cache size parameter for `MA46` and the parameter controlling the switch to full code in `MA48`. Further tests on the sensitivity of the codes to such parameters are being performed, and we warn against making too sweeping a conclusion on the merits of a code without considering the fine tuning further.

References

I. S. Duff and J. A. Scott, A comparison of frontal software with other sparse direct solvers, Technical Report RAL-TR-96-102 (Revised), Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1996.

2.8 Ordering symmetric sparse matrices for small profile and wavefront (J. K. Reid and J. A. Scott)

The ordering of large sparse symmetric matrices for small profile and wavefront or for small bandwidth is important for the efficiency of frontal and variable-band solvers. In this study, we were primarily concerned with positive-definite matrices, so we worked only with the pattern of the matrix and did not take into account any permutations needed for numerical stability. The work is useful also for a matrix that is non-definite or is symmetric only in the pattern of its entries, but in these cases the actual factorization may be more expensive and require more storage. For finite-element applications, we assumed that the matrix had been assembled.

In recent years, much attention has been paid to the problem of ordering symmetric sparse systems. One method which has been widely used for profile reduction is that of Sloan (1986). Sloan exploits the close relationship between a symmetric matrix $\boldsymbol{A} = \{a_{ij}\}$ of order n and its undirected graph with n nodes. Two nodes i and j are neighbours in the graph if and only if a_{ij} is nonzero. Sloan's algorithm has two distinct phases. In the

first, a start node and an end node are chosen. In the second phase, the chosen start node is numbered first and a list of nodes that are eligible to be numbered next is formed. At each stage of the numbering, the list of eligible nodes comprises the neighbours of nodes which have already been numbered and their neighbours. The next node to be numbered is selected from the list of eligible nodes by means of a priority function. A node has a high priority if it causes either no increase or only a small increase to the current front size and is at a large distance from the end node.

The Harwell Subroutine Library code `MC40` implements the ordering algorithm of Sloan and has been in satisfactory use for a decade. One reason for deciding a revision was now needed was that Kumfert and Pothen (1997) found that, for the large problems that are currently handled, there is a considerable efficiency gain from using a binary heap to manage the list of eligible nodes in the second phase of Sloan's algorithm. Our investigations supported this conclusion and, to make our code efficient both on the small problems used by Sloan and the much larger problems which are common today, we commence with code that performs a simple search, but switch to code that uses a binary heap if the number of eligible nodes exceeds a threshold. Our experience was that the performance is not very sensitive to this threshold. Based on our numerical experiments, we use a threshold of 100 in our code.

Another reason for revising the HSL code was to economize by working with supervariables (sets of variables for which the corresponding matrix columns have identical patterns) when the number of supervariables is significantly less than the number of variables. In our new code, the use of supervariables is optional.

We have added an option that permits users to provide a global priority vector because Kumfert and Pothen (1997) report that the final ordering can be significantly better if a hybrid algorithm that combines a spectral ordering with the Sloan algorithm is used. Our numerical experiments showed that, for large problems, the hybrid method generally gives smaller profiles than the Sloan algorithm, but for the small test problems, there seems to be little advantage in using the hybrid algorithm. Some examples illustrating this are given in Table 2.6.

A further option we have added is that of supplying the weights in Sloan's priority function. Experiments showed that, for some problems, the choice of (2,1) used by Sloan was far from optimal.

In our study, we also compared the effectiveness of using an algorithm based on level-set structures with using the spectral method for the computation of pseudoperipheral nodes as the basis of the Reverse Cuthill-McKee algorithm for bandwidth reduction. We found that using the end of the pseudodiameter with the narrower level-set structure as the starting node was important for both algorithms and that neither algorithm was consistently better than the other. As part of our revision of the code, we took the opportunity to include an

Identifier	Order	Sloan	Hybrid
bcsstk30	28,924	16.15	7.88
copter2	55,476	37.96	32.78
finance256	37,376	6.35	6.44
ford2	100,196	41.05	35.97
onera_dual	85,567	87.75	46.47
tandem_dual	94,069	66.21	42.22
DWT1005	1,005	0.035	0.031
DWT1242	1,242	0.036	0.040
DWT2680	2,680	0.090	0.091

Table 2.6: Profiles ($*10^6$) for the Sloan and hybrid algorithms

option for performing the Reverse Cuthill-McKee algorithm.

Our new code is called `MC60`, and we also provide a simple driver called `MC61`. Both codes will be included in the next release of HSL.

References

J. K. Reid and J. A. Scott, Ordering symmetric sparse matrices for small profile and wavefront, Technical Report RAL-TR-98-016, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1998.

G. Kumfert and A. Pothen. Two improved algorithms for envelope and wavefront reduction. *BIT*, **18**, 559–590, 1997.

S.W. Sloan. An algorithm for profile and wavefront reduction of sparse matrices. *Int J. Numerical Methods in Engineering*, **23**, 1315–1324, 1986.

2.9 Computing eigenvalues of the discretized Navier Stokes equations (R. B. Lehoucq and J. A. Scott)

Mixed finite-element discretizations of time-dependent equations modelling incompressible fluid flow problems typically produce nonlinear finite dimensional systems of the form

$$\begin{aligned} \boldsymbol{M}\dot{\boldsymbol{u}} + \boldsymbol{H}(\boldsymbol{u})\boldsymbol{u} + \boldsymbol{L}\boldsymbol{u} + \boldsymbol{C}\boldsymbol{p} &= \boldsymbol{b}, \\ \boldsymbol{C}^{\mathrm{T}}\boldsymbol{u} &= \boldsymbol{c}, \end{aligned} \tag{2.5}$$

where $\boldsymbol{u} \in \mathcal{R}^n$, $\boldsymbol{p} \in \mathcal{R}^m$ with $n > m$. $\boldsymbol{M}$ and $\boldsymbol{L}$ are symmetric positive-definite $n \times n$ matrices, $\boldsymbol{H}(\boldsymbol{u})$ is a nonsymmetric $n \times n$ matrix, and $\boldsymbol{C}$ is an $n \times m$ matrix of full

rank. Linearized stability analysis leads to the problem of finding a few eigenvalues of the generalized nonsymmetric eigenvalue problem

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x} \tag{2.6}$$

where

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{K} & \boldsymbol{C} \\ \boldsymbol{C}^{\mathrm{T}} & \boldsymbol{0} \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{M} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}, \quad \text{and} \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{p} \end{pmatrix}, \tag{2.7}$$

with the matrix $\boldsymbol{K}$ being a linearization of $\boldsymbol{H}(\boldsymbol{u}) + \boldsymbol{L}$. In the case of the so-called primitive variable formulation of the discretized Navier Stokes equations for incompressible flow, $\boldsymbol{u}$ and $\boldsymbol{p}$ denote the velocity and pressure degrees of freedom, respectively.

For stability analysis, the interest lies in computing the eigenvalues of smallest real part (the left-most eigenvalues). Of special interest is the case when the eigenvalues of smallest real part are complex, because algorithms for the detection of Hopf bifurcations in parameter dependent systems can be developed from knowledge of these eigenvalues. A complication that arises is that the eigenvalue problem can have infinite eigenvalues, corresponding to eigenvectors of the form $(\boldsymbol{0} \quad \boldsymbol{p})^{\mathrm{T}}$.

To be able to apply an iterative eigensolver, it is first necessary to transform the generalized eigenvalue problem $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}$ into a standard eigenvalue problem of the form $\boldsymbol{T}\boldsymbol{x} = \theta \boldsymbol{x}$. Because the solution of some linear system involving $\boldsymbol{A}$, $\boldsymbol{B}$ and/or a linear combination of $\boldsymbol{A}$ and $\boldsymbol{B}$ is needed anyway, rational transformations are an obvious choice. In this study, we considered using shift-invert and Cayley transformations. Traditionally, subspace iteration has been the method of choice for the transformed problem because Arnoldi's method was perceived as being less reliable. Experiments reported by Garratt (1991) found that Arnoldi's method sometimes missed the sought-after left-most eigenvalue and, because reliability is more important than efficiency in linear stability analysis, Garratt favoured subspace iteration. However, the recent work of Meerbergen and Spence (1997) demonstrated, in theory, that the implicitly restarted Arnoldi method (IRAM) combined with shift-invert transformations, can be successfully employed to compute the left-most eigenvalues of generalized eigenvalue problems with the block structure (2.7). In this work, we looked at using the package ARPACK of Lehoucq, Sorensen and Yang (1998), which implements the IRAM, combined with the frontal solver `MA42` of Duff and Scott (1996), to compute the left-most eigenvalues of the discretized Navier Stokes equations.

The algorithm we have adopted combines using the shift-invert transformation with a zero pole to obtain an initial approximation to the spectrum and then using a generalized Cayley transformation. We have looked at using the standard and the $\boldsymbol{B}$ semi-inner product within Arnoldi's method and compared using zero shifts with exact shifts in the IRAM. We have also examined the use of purification techniques to improve the quality

of the computed eigenvectors. Our results suggest that, although using the $\boldsymbol{B}$ semi-inner product is more expensive than the standard inner product, it does offer advantages in terms of reliability and, in general, converges in fewer iterations. We also found that, because of the connection between IRAM and subspace iteration (see, for example, Lehoucq et al., 1998), it is more reliable to use zero shifts rather than exact shifts. In addition, in our numerical experiments we found that the accuracy of the eigenvectors computed using the generalized Cayley transformation can be reduced, sometimes very significantly, by purification with a shift-invert operator. We conclude that, with careful implementation, implicitly restarted Arnoldi methods can be used reliably for linear stability analysis.

Based on our findings so far, the plan is to incorporate the ARPACK software package within the AEA Technology finite-element package ENTWIFE (Cliffe, 1996).

References

K.A. Cliffe. ENTWIFE (Release 6.3) Reference Manual. Technical Report AEAT-0823, AEA Technology, Harwell Laboratory, Oxfordshire, England, 1996.

I.S. Duff and J.A. Scott. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Transactions on Mathematical Software*, **22**(1), 30–45, 1996.

T.J. Garratt. *The numerical detection of Hopf bifurcations in large systems arising in fluid mechanics*. Phd Thesis, University of Bath, 1991.

R. B. Lehoucq and J. A. Scott, Implicitly restarted Arnoldi methods and eigenvalues of the discretized Navier Stokes equations. Technical Report RAL-TR-97-057, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

R.B. Lehoucq, D.C. Sorensen, and C. Yang. *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, PA, 1998. To appear.

K. Meerbergen and A. Spence. Implicitly restarted Arnoldi with purification for the shift–invert transformation. *Mathematics of Computation*, **218**, 667–689, 1997.

2.10 Iterative methods for finite-element problems (M. J. Daydé, J. P. Décamps, and N. I. M. Gould)

We are interested in the solution of systems of symmetric linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{2.8}$$

where

$$\boldsymbol{A} = \sum_{i=1}^{e} \boldsymbol{E}_i,$$

and where the matrices $\boldsymbol{E}_i$ are symmetric and of low rank. Systems of this form arise in finite-element methods for solving partially differential equations, and as subproblems within partially separable optimization calculations. In order to solve very large equations of this form, we may have to resort to iterative methods, and thus it is important to be able to precondition the systems to accelerate the convergence of such methods. In previous work (Daydé, L'Excellent and Gould, 1997), we considered so-called element-by-element (EBE) methods, in which factors of modifications of each of the individual matrices $\boldsymbol{E}_i$ are combined to produce an overall preconditioner. We found it particularly advantageous to amalgamate or assemble groups of the $\boldsymbol{E}_i$ before applying the preconditioner.

More recently, we have considered two extensions to this work. The first is appropriate when each $\boldsymbol{E}_i$ only has nonzeros in a small number of rows and columns, and for which the resulting dense "element" is nonsingular. Then, if we build the block diagonal matrix $\boldsymbol{B}$ whose diagonal blocks are these elements, it is easy to show that (2.8) is equivalent to the augmented system

$$\begin{pmatrix} \boldsymbol{B} & \boldsymbol{C} \\ \boldsymbol{C}^{\mathrm{T}} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{pmatrix} \tag{2.9}$$

where $\boldsymbol{C}$ is very sparse and has entries ± 1. This technique is known as matrix *stretching*. Since inverting $\boldsymbol{B}$ is trivial, the obvious approach is to use factors of $\boldsymbol{B}$ and the Schur complement $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{B}^{-1}\boldsymbol{C}$ to solve (2.9). However, the Schur complement may be dense, and it is usually preferable to solve the Schur complement system using a preconditioned iterative method. We compared a variety of preconditioners in Daydé, Décamps and Gould (1997) on a number of challenging large examples. The EBE method often provides a good preconditioner, particularly when the original system is ill-conditioned. We also noted that the Schur complement is frequently better conditioned than the whole system. Finally, we reported on some preliminary parallel experiments that indicate how stretching frequently improves the parallelization of the linear solver.

Another topic of interest is the solution of (2.8) when $\boldsymbol{A}$ has the form

$$\boldsymbol{A} = \sum_{i=1}^{e} \boldsymbol{A}_i \boldsymbol{A}_i^{\mathrm{T}},$$

and where $\boldsymbol{A}_i$ is an n by n_i ($n_i \ll n$) real matrix. Systems of this form arise naturally in a number of ways, such as in normal equation approaches to least-squares problems, Schur complement methods following partial elimination in augmented systems, and in the Newton equations for partially separable optimization of unary and more general partially separable functions. We suppose that n is sufficiently large that the structure of the system must be exploited, but we do *not* assume that all the $\boldsymbol{A}_i$ are sparse. In Daydé, Décamps

and Gould (1998), we derive suitable EBE-like preconditioners for such systems, which take advantage of the null-spaces of the $\boldsymbol{A}_i$, and demonstrate their effectiveness on a number of test examples. We also consider combining these methods with existing techniques to cope with the commonly-occurring case where the coefficient matrix is the linear sum of elements, some of which are of very low rank.

References

M. J. Daydé, J. Décamps, and N. I. M. Gould. Solution of unassembled linear systems using block stretching: Preliminary experiments. Technical Report RT/APO/97/3, ENSEEIHT-IRIT, Toulouse, France, 1997.

M. J. Daydé, J. P. Décamps, and N. I. M. Gould. Subspace-by-Subspace preconditioners for structured linear systems. Technical Report RAL-TR-98-005, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1998.

M. J. Daydé, J.-Y. L'Excellent, and N. I. M. Gould. Element-by-element preconditioners for large partially separable optimization problems. *SIAM Journal on Scientific Computing*, **18**(6), 1767–1787, 1997.

2.11 Permuting large entries to the diagonal (I. S. Duff and J. Koster)

We have conducted further investigation into permutations that put large entries onto the diagonal of a sparse matrix. In particular, we have developed a sparse variant of a permutation and scaling algorithm of Olschowka and Neumaier (1996) that first minimizes the absolute value of the product of entries on the diagonal and then scales the matrix so that these entries are one and all other entries are no greater than one. We call the version without scaling the maximum product algorithm (MPD) and that with scaling the MPS algorithm. We have implemented these strategies using algorithms based on bipartite weighted matchings (BWM) that do not rely on repeated applications of the depth first search transversal algorithm `MC21`. We also used a BWM approach to implement the bottleneck transversal algorithm that maximizes the minimum entry (in modulus) on the diagonal.

We performed a series of experiments using these transversal algorithms and comparing them with unweighted transversal selection (that is, `MC21`) on a range of matrices. We studied the effect of these reorderings on a multifrontal direct code, on ILU-type preconditionings, and on the Block Cimmino algorithm.

We find that these new codes (MPD and MPS) can surprisingly be sometimes faster than the original `MC21` code although normally algorithm `MC21` is faster than BT, and this

is faster than MPD. Although the gains from using such preorderings are evident in all three solution schemes and sometimes dramatically so, the effect is not uniform and no single ordering is consistently best, except in the case of Block Cimmino where the scaling in MPS has a significant effect.

The findings of this work are reported in Duff and Koster (1997) and in more detail in the thesis of Jacko Koster (Koster, 1997). A further report is in preparation.

References

M. Olschowka and A. Neumaier. A new pivoting strategy for Gaussian elimination. *Linear Algebra and its Applications*, **240**, 131–151, 1996.

I. S. Duff and J. Koster. The design and use of algorithms for permuting large entries to the diagonal. Technical Report RAL-TR-97-059, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

J. Koster. On the parallel solution and the reordering of unsymmetric sparse linear systems. INPT PhD Thesis TH/PA/97/51, CERFACS, Toulouse, France, 1997.

2.12 Least-squares problems from kinematics (J. Cardenal, I. S. Duff, and J. Jimenez)

Least-squares problems from kinematics have certain characteristics that can be exploited in their solution. First, they are nearly square (*quasi-square*) so that null-space methods can be used to great effect. We have studied such an approach in Cardenal, Duff and Jiménez (1997). Additionally, the rectangular coefficient matrix can be permuted to a generalized block triangular form through a Dulmage-Mendelsohn decomposition performed, for example, by an algorithm like that developed by Pothen and Fan (1990). If the rectangular matrix $\boldsymbol{A}$ can be ordered to the form $\begin{pmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 \\ \mathbf{0} & \boldsymbol{A}_4 \end{pmatrix}$ where $\boldsymbol{A}_2$ and $\boldsymbol{A}_4$ are rectangular, then the augmented system $\begin{pmatrix} \boldsymbol{I} & \boldsymbol{A} \\ \boldsymbol{A}^{\mathrm{T}} & \mathbf{0} \end{pmatrix}$ can be permuted to block triangular form. Although this destroys symmetry, the efficiency from using the block triangular form may outweigh this so we have the interesting situation where an unsymmetric sparse code that takes advantage of the block triangular form (for example, `MA48`) may outperform a symmetric code (for example, `MA27` or `MA47`). We show this is the case in Table 2.7.

Number rows	117	170	181	385	574
Number columns	105	161	180	361	526
Number entries	615	1069	1200	1661	3365
`MA27`	7.3	15.6	13.1	43.4	120.7
`MA48`	6.6	10.1	7.1	17.4	58.1

Table 2.7: Times (in seconds) for runs on a SGI Onyx

References

J. Cardenal, I. S. Duff, and J. M. Jiménez. Solution of sparse quasi-square rectangular systems by Gaussian elimination. Technical Report RAL-TR-96-013 (Revised), Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997. To appear in *IMA J. Numerical Analysis*.

A. Pothen and C. Fan. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software*, **16**(4), 303–324, 1990.

2.13 An improvement to the general sparse-matrix solver MA48 (J. K. Reid)

Following a request from AspenTech, we modified `MA50A/AD`, which is called from `MA48` (Duff and Reid, 1996) to perform an analysis of an unsymmetric sparse matrix prior to its LU factorization. If it now exits prematurely for lack of storage, it returns the amount of storage it would have needed to get to the same point in `MA50B/BD` (actual LU factorization). This has proved very helpful to the user in deciding how much storage to allocate before re-entry.

References

I. S. Duff and J. K. Reid. The design of MA48, a code for the direct solution of sparse unsymmetric linear systems of equations. *ACM Transactions on Mathematical Software*, **22**(2), 187–226, 1996.

2.14 Sparse BLAS (I. S. Duff)

The Basic Linear Algebra Subprograms (BLAS) have had a profound influence on the development of algorithms and software for the solution of systems of linear equations with full coefficient matrices. The BLAS Technical Forum has been meeting 4-5 times

each year in the United States in an attempt to determine a consensus and establish a standard for extensions to the BLAS. The main aspects that the Forum is considering are: functional extensions to the existing dense BLAS; mixed precision BLAS; C, C++ and F90 bindings; and sparse BLAS.

Sparse BLAS for computations of the form $\boldsymbol{AX}$ where $\boldsymbol{A}$ is sparse and $\boldsymbol{X}$ is dense are particularly important for the iterative solution of sets of sparse linear equations. The sparse BLAS are organized at two levels, a User Level that has a simple and standardized interface (Duff, Marrone, Radicati and Vittoli, 1997), and a Toolkit level that implements the kernels for a specific data structure (Carney, Heroux, Li and Wu, 1994).

The functionality of the sparse BLAS consists of a sparse matrix-by-matrix multiplication and a sparse triangular solution routine. We have designed routines for these cases as Level 3 BLAS so that the Level 2 equivalents are available as a particular case.

In the sparse case, there is the problem of how the matrix is stored, and indeed storage schemes differ very widely and are usually applications dependent. We have therefore also designed routines for data conversion with a standard interface to include the main data structures that we are familiar with. Finally, we have also included two permutation routines to facilitate the efficient use of the kernels in iterative solvers.

In the User Level BLAS, we have deliberately shielded the user from the details of the representations and have offered a transparent interface to allow vendors to choose the best representation without the need for the user to be concerned with this choice. We have also illustrated a Fortran 90 binding for this User Level approach.

References

Sandra Carney, Michael A. Heroux, Guangye Li, and Kesheng Wu. A revised proposal for a sparse BLAS toolkit. Technical Report 94-034, Army High Performance Computing Research Center, June 1994. Updated version at Web address `http://www.cray.com/products/applications/support/scal/spblastk.ps`.

I. S. Duff, Michele Marrone, Giuseppe Radicati, and Carlo Vittoli. Level 3 Basic Linear Algebra Subprograms for sparse matrices: a user level interface. Technical Report RAL-TR-95-049 (Revised), Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997. To appear in *ACM Trans Math Softw* 23(3).

2.15 Survey articles on sparse matrix and optimization research (I. S. Duff and N. I. M. Gould)

An important part of our activity lies in developing survey and tutorial material and disseminating this through publications and talks. We are concerned both to assist young numerical analysts and also scientists from application areas who wish to use modern numerical analysis techniques.

As part of this activity, we have written surveys of research in direct methods for sparse matrices and in methods for constrained optimization, that were presented at a State of the Art in Numerical Analysis meeting. This was the fourth meeting in a decennial series and was held in York in April 1996.

The survey on sparse matrices covered high performance sparse factorization techniques using Level 2 and Level 3 BLAS, new techniques for symmetric orderings, new approaches to the solution of unsymmetric systems, the solution of indefinite systems, least-squares problems, parallel computing, preconditioning, and a discussion of recent trends towards a sparse problem solving environment. The survey on constrained optimization concentrated on advances in SQP, penalty, augmented-Lagrangian, and optimality-condition based methods for handing general nonlinear constraints, as well as special methods for dealing with linear and convex constraints, and a discussion on available software. Both surveys are available as RAL reports (Duff, 1996, and Conn, Gould and Toint, 1996, respectively) and are included in the State of the Art Proceedings (Duff, 1997, and Conn, Gould and Toint, 1997, respectively), which was edited by Iain Duff and Alistair Watson (Duff and Watson, 1997).

References

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Methods for nonlinear constraints in optimization calculations. Technical Report RAL-TR-96-042, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1996.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Methods for nonlinear constraints in optimization calculations. *in* I. S. Duff and G. A. Watson, eds, 'The State of the Art in Numerical Analysis', pp. 363–390, Oxford, 1997. Oxford University Press.

I. S. Duff. Sparse numerical linear algebra: direct methods and preconditioning. Technical Report RAL-TR-96-047, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1996.

I. S. Duff. Sparse numerical linear algebra: direct methods and preconditioning. *in* I. S. Duff and G. A. Watson, eds, 'The State of the Art in Numerical Analysis', pp. 27–62, Oxford, 1997. Oxford University Press.

I. S. Duff and A. Watson, editors. *The State of the Art in Numerical Analysis*, Oxford University Press, Oxford, 1997.

3 Optimization

3.1 LANCELOT (A. R. Conn, N. I. M. Gould and Ph. L. Toint)

Since its release in 1992, the large-scale nonlinear optimization package LANCELOT A (Conn, Gould and Toint, 1992) has been installed at over 250 sites throughout the world. Over the past two years, an interface has been provided by others to the popular language modelling language AMPL, and the package has become one of the most-used solvers under NEOS, the public-domain Network-Enabled Optimization System, at Argonne National Laboratory.

Despite its considerable successes, work has continued on its successor, particularly as it has long been recognized that LANCELOT is deficient in a number of ways:

(i) Inequality constraints are converted to equations by the addition of "slack" variables. Thus a problem involving 20 variables and 5000 inequality constraints was formulated by LANCELOT A as a problem in 5020 variables.

(ii) The package was designed to handle nonlinear constraints, while little attention was given to the possibility that many of these might actually be linear. Thus the method is less efficient for linearly constrained problems than other software packages.

(iii) The "projected-gradient" mechanism used to identify which subset of constraints are "binding" at the solution proved to be inefficient for (nearly) degenerate problems.

(iv) The general linear equation solvers we provided were not especially suited for the types of structures routinely encountered in many applications.

(v) The simple "trust region" used was not particularly appropriate when the problem is structured.

Much work has been performed at RAL and elsewhere over the past few years with the aim of correcting such deficiencies. In this and the following sections, we describe some of our efforts.

A long-time aim has been to compare LANCELOT with other state-of-the-art codes. We have been particularly interested in MINOS 5.5, which is probably the best-known code of all. The authors of both packages agreed to collaborate on such a comparison, and the results of significant numerical testing (see Bongartz, Conn, Gould, Saunders and Toint, 1997*b*) were summarized by Bongartz, Conn, Gould, Saunders and Toint (1997*a*). We observed that LANCELOT is usually more efficient in terms of the number of function and derivative evaluations. If the latter are inexpensive, MINOS may require less CPU time unless there are many degrees of freedom. LANCELOT proves to be less reliable than MINOS on linear programming problems, but somewhat more reliable on problems

involving nonlinear constraints. Although none of these conclusions were particularly surprising given the different design intentions of the authors, it was comforting to have our prejudices confirmed.

Since our current means of determining the correct set of active inequality constraints at the solution to a nonlinear programming problem has proved to be rather inefficient, and since interior-point—and particularly primal-dual—methods for linear programming have purported to avoid such inefficiencies, it is natural to consider interior-point methods for nonlinear optimization. We have considered two such approaches. In Conn, Gould and Toint (1996), we proposed a new primal-dual algorithm for the minimization of a non-convex objective function subject to simple bounds and linear equality constraints. The method alternates between a classical primal-dual step and a Newton-like step in order to ensure descent on a suitable merit function. We show convergence of a well-defined subsequence of iterates to a first-order critical point. Preliminary numerical results are most encouraging although there are still difficulties primarily with an effective means of coping with negative curvature.

As trust-region methods are often designed to cope with difficulties caused by non-convexity, our second approach is of the trust-region variety. This approach resembles more closely a classical sequential barrier minimization, but the challenges which arise when a trust-region method is used to solve the barrier subproblem are rather more subtle. Once again, global convergence to a first-order critical point may be demonstrated (see, Conn, Gould and Toint, 1998). Of particular interest are the choice of trust-region norm, which is made so as to capture the geometry of the barrier function, and the potentially unbounded preconditioners which are recommended. It has yet to be seen which of the two approaches turns out to be more effective in practice, but careful implementations are underway.

On a theoretical note, it has long been known that barrier algorithms for constrained optimization can produce a sequence of iterates converging to a critical point satisfying weak second-order necessary optimality conditions, while their inner iterations ensure that second-order necessary conditions hold at each barrier minimizer. We have shown that, despite this, strong second-order necessary conditions may fail to be attained at the limit, even if the barrier minimizers satisfy second-order sufficient optimality conditions (see Gould and Toint, 1997 for details).

References

I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and Ph. L. Toint. A numerical comparison between the LANCELOT and MINOS packages for large-scale nonlinear optimization. Technical Report RAL-TR-97-054, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997*a*.

I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and Ph. L. Toint. A numerical comparison between the LANCELOT and MINOS packages for large-scale nonlinear optimization: the complete results. Technical Report (97/14), Department of Mathematics, FUNDP, Namur, Belgium, 1997*b*.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A)*. Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. A primal-dual algorithm for minimizing a non-convex function subject to bound and linear equality constraints. Technical Report RAL-TR-96-096, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1996.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. A primal-dual trust-region algorithm for minimizing a non-convex function subject to bound and linear equality constraints. Technical Report (to appear), Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1998.

N. I. M. Gould and Ph. L. Toint. A note on the second-order convergence of optimization algorithms using barrier functions. Technical Report RAL-TR-97-055, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

3.2 Exploiting negative curvature in large-scale nonlinear programming (N. I. M. Gould, S. Lucidi, M. Roma and Ph. L. Toint)

This research was directed towards methods for solving highly nonlinear unconstrained minimization problems. Such problems typically exhibit negative curvature away from the solution, and are such that function and derivative values may change dramatically for small changes in the minimization variables. The whole program focused on three separate but related topics,—the third of which is described in the next section—and was supported by a British Council-MURST research grant.

The first piece of work (Gould, Lucidi, Roma and Toint, 1997) aimed to define new efficient linesearch algorithms for solving large-scale unconstrained optimization problems, and which exploit the local non-convexity of the objective function. Existing algorithms of this type compute two search directions at each iteration: a Newton-type direction that ensures global and fast local convergence, and a negative curvature direction that enables the iterates to escape from the region of local non-convexity. A new point is then generated by performing a search along a curve obtained by combining these two directions. However,

the relative scaling of the two directions is typically ignored, and this often led to one of the directions dominating the other. Our way around this difficulty is simply to select the more promising of the two directions, and then to perform a step along this direction. The selection criterion is based on a test on the rate of decrease of the quadratic model of the objective function. We have established global convergence to second-order critical points for the new algorithm and reported some encouraging but preliminary numerical results.

The second research topic considers algorithms for unconstrained nonlinear optimization for which the model used by the algorithm to represent the objective function explicitly includes "memory" of past iterations. This is intended to make the algorithm less myopic in the sense that its behaviour is not completely dominated by the local nature of the objective function, but rather by a more global view. We proposed a non-monotone linesearch algorithm that had this feature, and established its global convergence (Gould, Lucidi, Roma and Toint, 1998). Numerical evidence indicating the effectiveness of the approach was reported. There are still a number of outstanding issues here, and future work will concentrate on improving the basic approach.

References

N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint. Exploiting negative curvature directions in linesearch methods for unconstrained optimization. Technical Report RAL-TR-97-064, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint. A linesearch algorithm with memory for unconstrained optimization. Technical Report RAL-TR-98-003, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1998.

3.3 Trust-region methods (A. R. Conn, N. I. M. Gould, S. Lucidi, J. Nocedal, M. Roma and Ph. L. Toint)

Trust-region methods replace a nonlinear optimization problem by a sequence of simpler problems, whose solutions are constrained to lie within a neighbourhood of the current estimate of the solution. Typically, the simple "trust-region" problem which must be (approximately) solved at each iteration involves the minimization of a quadratic function within the intersection of a trust region—a constraint on the allowable step $\boldsymbol{s}$ of the form

$$\|\boldsymbol{s}\| \leq \Delta$$

for some appropriate norm and positive radius Δ—and, possibly, some other simplified constraints. Such methods possess extremely powerful convergence properties under weak assumptions, and perform very well in practice.

We have considered the approximate minimization of a quadratic function within an ellipsoidal trust region. This is an important subproblem for many nonlinear programming methods. When the number of variables is large, the most widely-used strategy is to trace the path of conjugate gradient iterates either to convergence or until it reaches the trust-region boundary. In Gould, Lucidi, Roma and Toint (1997), we investigated ways of continuing the process once the boundary has been encountered. The key was to observe that the trust-region problem within the currently generated Krylov subspace has a very special structure which enables it to be solved very efficiently. We compared the new strategy with existing methods. The resulting software package has been made available as a Fortran 90 module, `HSL_VF05`, within the Harwell Subroutine Library. This work is now being extended to cope with linear constraints on the variables.

Since the actual norm which defines the trust region is, to a large extent, arbitrary, we might consider whether there are certain norms for which the solution of the subproblem is easy. In Gould and Nocedal (1997), we proposed a trust-region method for unconstrained minimization, using a trust-region norm based upon a modified absolute-value factorization of the model Hessian. We showed that the resulting trust-region subproblem may be solved using a single factorization. In the convex case, the method reduces to a backtracking Newton linesearch procedure. The resulting software package `HSL_VF06` is also available within the Harwell Subroutine Library, and numerical evidence shows that the approach is effective in the non-convex case.

These and many other topics are covered in our forthcoming book (Conn, Gould and Toint, 1999) on trust-region methods. The book should be published (subject to satisfactory final reviews) by SIAM at the end of 1999, and is intended to be a definitive account of the trust-region framework in all its different guises. Topics to be covered will include unconstrained and constrained differentiable optimization, non-differentiable optimization and nonlinear systems of equations, and it is intended that the book be both comprehensive and essentially self-contained. The current manuscript is roughly six hundred pages long and will ultimately be about eight hundred pages.

References

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-region methods.* In preparation, 1999.

N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint. Solving the trust-region subproblem using the Lanczos method. Technical Report RAL-TR-97-028, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

N. I. M. Gould and J. Nocedal. The modified absolute-value factorization norm for trust-region minimization. Technical Report RAL-TR-97-071, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1997.

3.4 Steepest-edge simplex code LA04 for linear programming (J. K. Reid)

In response to requests from two potential clients, we have resumed work on the steepest-edge (Goldfarb and Reid, 1977) simplex code `LA04` for linear programming. Although most interest in recent years has been in interior-point methods, it has become apparent that simplex methods have a useful role to play in conjunction with them or for problems that are not very large. `LA04` aims for robustness using the principle that it seeks a solution that is exact for a nearby problem. It has run successfully on all but one of the test examples in the netlib test set. This test example is called QAP15 and is an extremely large and extremely degenerate problem and `LA04` 'stalls', that is, does a huge number of iterations without progress. The problem is recognized as too difficult for most codes.

References

D. Goldfarb and J. K. Reid. A practical steepest-edge simplex algorithm. *Mathematical Programming*, **12**, 361–371, 1977.

4 Numerical Linear Algebra

4.1 BLAS kernels (M. J. Daydé and I. S. Duff)

The Basic Linear Algebra Subprograms (BLAS) are the most important building blocks for linear algebra codes. In particular, the Level 3 BLAS, which include kernels for matrix-matrix multiplication, and the solution of sets of triangular equations, are a particularly powerful tool for the construction of efficient algorithms and codes on workstations, vector supercomputers, and high-performance computers that utilize RISC processors. We have been designing and implementing Level 3 BLAS kernels over the past several years in collaboration with our colleagues at CERFACS and ENSEEIHT in Toulouse. In this past year, we have continued our work on the development of Level 3 BLAS kernels by extending the techniques of blocking and loop unrolling to optimize performance on RISC-based computers and workstations. This has included the design and tuning of a version for the SGI Power Challenge and one for the CRAY T3D. Our portable code is competitive with the vendor supplied kernels on the SGI and significantly outperforms that provided on the T3D on many of the kernels. The kernels have also been tested on Sun and HP workstations and our software performs far better than a standard implementation. The current version is described in the report by Daydé and Duff (1997*a*).

We are currently developing versions of the BLAS for complex matrices and show some results from our kerensl (called RISC B.) in Table 4.1. On every machine except the IBM SP2 we outperform the vendor's code and the performance of the RISC BLAS is everywhere substantially better than that of standard code.

We have also demonstrated the use of these building blocks in the construction of dense and sparse linear equation solvers in the report by Daydé and Duff (1997*b*).

References

M. J. Daydé and I. S. Duff. A blocked implementation of Level 3 BLAS for RISC processors (revised version). Technical Report RT/APO/97/2, ENSEEIHT-IRIT, Toulouse, France, 1997*a*.

M. J. Daydé and I. S. Duff. The use of computational kernels in full and sparse linear solvers, efficient code design on high-performance RISC processors. *in* J. M. L. M. Palma and J. Dongarra, eds, 'Vector and Parallel Processing - VECPAR'96', Lecture Notes in Computer Science **1215**, pp. 108–139, Berlin, 1997*b*. Springer.

Comp.	Kernel	op($\boldsymbol{A}$) op($\boldsymbol{B}$)	'N' 'N'	'C' 'N'	'T' 'N'	'N' 'C'	'N' 'T'	'C' 'C'	'C' 'T'	'T' 'C'	'T' 'T'
		Version									
CRAY	CGEMM	Standard	104	112	147	100	101	72	72	70	73
T3E		SCILIB	217	207	213	157	158	219	220	226	227
		RISC B.	222	234	234	200	201	250	250	217	217
HP	CGEMM	Standard	195	169	174	194	197	162	167	169	174
C160		RISC B.	331	323	333	350	341	367	378	352	341
	ZGEMM	Standard	197	177	193	201	200	163	160	159	167
		RISC B.	299	284	294	295	295	350	363	361	333
IBM	CGEMM	Standard	94	106	142	92	92	94	110	113	122
SP2		ESSL	398	417	401	388	390	434	396	396	403
P2SC		RISC B.	361	385	399	360	393	360	377	377	389
	ZGEMM	Standard	110	140	143	106	105	80	99	97	94
		ESSL	382	431	388	362	362	362	373	364	372
		RISC B.	348	355	361	304	321	328	324	342	333
SGI	CGEMM	Standard	204	295	251	200	199	193	185	182	183
POWER		library	304	286	286	291	291	275	275	275	276
10000		RISC B.	317	318	318	315	315	319	319	320	320
	ZGEMM	Standard	200	264	232	193	192	141	153	153	209
		library	279	255	254	252	252	236	236	236	236
		RISC B.	275	276	276	279	279	276	277	276	276
SUN	CGEMM	Standard	50	35	35	46	47	32	32	31	32
Ultra1		RISC B.	139	143	138	128	138	110	109	117	121
143MHz	ZGEMM	Standard	46	49	55	47	48	40	43	47	46
		RISC B.	73	71	71	68	64	49	48	54	53

Table 4.1: Average performance in Mflop/s of the blocked implementation of the complex GEMM on range of RISC processors (using square matrices of order 32, 64, 96, and 128).

4.2 Implicit scaling of linear least-squares problems (J. K. Reid)

We have been considering the weighted least-squares problem of minimizing

$$(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x})^{\mathrm{T}} \boldsymbol{W}^2 (\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}),$$

where $\boldsymbol{W}$ is a diagonal matrix. This can be expressed as the unweighted least-squares solution of the system

$$\boldsymbol{W}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{W}\boldsymbol{b}.$$

Powell and Reid (1969) formed this set of equations explicitly and used QR factorization of $\boldsymbol{A}$ (Golub, 1965). They performed a backward error analysis that showed the solution obtained to be exact for a perturbed system where the perturbations in each row were small compared with the largest element in the row.

We have shown that the algorithm can be extended to the constrained case by use of implicit scaling, that is, without ever forming $\boldsymbol{W}\boldsymbol{A}$ or $\boldsymbol{W}\boldsymbol{b}$ explicitly. Corresponding to the QR factorization of $\boldsymbol{A}$, we obtain the factorization

$$\boldsymbol{W}\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{W}\boldsymbol{R}$$

and the backward error analysis of Powell and Reid is applicable. Furthermore, it is applicable to the constrained case by using infinite weights. By storing the inverse weights $\boldsymbol{W}^{-1}$, we can include the case of infinite weights without the need to represent ∞.

We also show that iterative refinement is applicable with implicit and possibly infinite weights.

For further details of this work, see Reid (1998).

References

G. H. Golub. Numerical methods for solving linear least squares problems. *Numerische Mathematik*, **7**, 206–216, 1965.

M. J. D. Powell and J. K. Reid. On applying householder transformations to linear least squares problems. *in* A. J. H. Morrell, ed., 'Information Processing 68', pp. 122–126. North-Holland, 1969.

J. K. Reid. Implicit scaling of linear least squares problems. Technical Report RAL-TR-98-027, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1998.

4.3 Numerical linear algebra for high performance computers (J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst)

The SIAM book on "Solving Linear Systems on Vector and Shared Memory Computers" (Dongarra, Duff, Sorensen and van der Vorst, 1991) is still a best seller, but it has ceased to be at the leading-edge because of recent developments both in computer hardware and also in the design and implementation of algorithms in numerical linear algebra.

We have therefore been planning and writing a substantial revision of this text over the past two years and have recently completed a complete first draft. It is hoped that this will be published by SIAM in 1998 or early in 1999.

We have decided not to change a winning formula too much but have felt it necessary to add some chapters to cover new developments and areas where we felt the "first edition" to be deficient. Our discussion of computer hardware and the study of the design and performance of kernels now includes distributed memory parallel computers and, in addition to a much revised chapter on the solution of dense linear systems (now including copious reference to ScaLAPACK), there is a full chapter on eigensolvers. The chapters on direct and on iterative methods for sparse systems have been completely rewritten and extra chapters have been added on preconditioning, generalized eigenproblems, and on Krylov methods to avoid duplication between the iterative solution and eigensystem chapters.

References

Jack J. Dongarra, I. S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM Press, Philadelphia, 1991. Second edition in preparation.

5 Fortran

5.1 Introduction (J. K. Reid)

John Reid has been very active in the area of Fortran standardization and associated activities. He contributed a chapter on the array features (Reid, 1996) in a special issue of Computer Standards and Interfaces devoted to Fortran 90. He revised his book with Mike Metcalf (Metcalf and Reid, 1996a) to keep it up-to-date with formal interpretations of the standard and to include a chapter on Fortran 95. John and Mike also wrote a version of their book for the subset language F (Metcalf and Reid, 1996b) and we say more about this in Section 5.2. John has been striving for several years to see exception handling included in Fortran and this work has now reached a conclusion with the publication of a 'Type-2 Technical Report', which he edited (see Section 5.3).

John also acted as editor for the corrigenda to Fortran 90 and the repository of requirements for Fortran 2000. The third and final corrigendum was formally accepted in 1997, with very few changes needed following a formal ballot. Significant activity on the repository was needed during the year, but the content of Fortran 2000 has now been chosen and that activity is now dormant.

The Group is involved with the ESPRIT IV PARASOL Project for the parallel solution of sparse linear equations (see Section 6.2), which has decided to rely on the use of MPI from Fortran. MPI is essentially a library of C routines, some of which rely on argument associations that are not valid for Fortran. This led John to an active email collaboration with the binding subgroup of the MPI committee, as a result of which the problems are better explained in the MPI-2 document and some strategies that may be used to ameliorate the problem are explained.

References

M. Metcalf and J. K. Reid. *Fortran 90/95 explained.* Oxford University Press, Oxford, 1996a.

M. Metcalf and J. K. Reid. *The F programming language.* Oxford University Press, Oxford, 1996b.

J. K. Reid. The array features. *Computer Standards and Interfaces*, **18**, 323–331, 1996.

5.2 The F programming language (J. K. Reid)

F is a subset of Fortran 90, chosen by Walt Brainerd, David Epstein, and Dick Hendrickson of Imagine Inc. to be easy to learn, easy to implement, easy to understand and safe. It is

fully defined by the book of Metcalf and Reid (1996) and its advantages are summarized by Reid (1996). Of particular note are the following:

No duplications. Where Fortran 90 has alternative syntax for expressing the same thing, F picks a good one and discards the rest. For example, all type declaration statements must have a double colon and must specify all the attributes of the entities.

Source form restrictions make it possible to catch all occurrences of a name with a case-sensitive editor.

No obsolescent features, since they are redundant.

No labels, since even the most disciplined use of labels leaves the reader uncertain over how each statement may be reached.

No use of explicit kind values, since they are not portable.

Reserved words. All the language keywords, such as program, do, true, etc. are reserved for use as such.

The baroque rules for use association are drastically simplified.

Procedures all have explicit interfaces, and so can be checked at compile time.

Functions are required to be free of side effects.

Storage association is fundamentally unsafe particularly when COMMON and EQUIVALENCE are used to share storage between data of different types, and is not permitted.

References

M. Metcalf and J. K. Reid. *The F programming language*. Oxford University Press, Oxford, 1996.

J. K. Reid. An appreciation of F. *Fortran Journal*, **8**(6), 3–9, 1996.

5.3 Exception handling in Fortran 90 (J. K. Reid)

The ISO Fortran Standardization Committee, WG5, decided in 1995 that exception handling was too important to leave until Fortran 2000, and so established a development body to create a 'Type 2 Technical Report'. This requires a less lengthy approval process than a standard and the committee promised that it will be part of Fortran 2000 unless

serious snags are uncovered in the field, which will permit vendors to implement it as an extension of Fortran 95. It is thus a kind of beta-test facility for a new language feature.

John Reid led the development body and considered two approaches, the enable construct or a collection of procedures. The merits of the two were discussed by Reid (1997). The enable construct provides a mechanism for specifying the scope within which extra checks are required. The collection of procedures provides a mechanism for inspecting and altering the values of the exception flags of the IEEE floating-point standard.

The enable construct has been under consideration for many years and is seen by most people as the best long-term solution. However, there have always been difficulties with the details, mainly because people have varying perceptions of what should happen after the detection of an exception, particularly when the only possible handler is much higher up the call chain and some of the intermediate calls are in procedures without exception handling. For these reasons, and because the procedures approach provides the useful by-product of other support for the IEEE floating-point standard, the enable approach was discarded.

The procedures approach was not without its difficulties since the aim was to accommodate vendors that do not implement the IEEE standard or provide options for much faster execution at the expense of partial compliance. A particular difficulty arose from some vendors requiring different machine instructions to be generated according to whether exception handling is in operation. The solution adopted was to provide the feature in an intrinsic module and allow the compiler to treat the USE statement as a directive that controls the code generated. An unfortunate consequence of this is that the feature cannot be implemented without changes to the compiler itself.

The Technical Report passed its first formal ISO ballot in March 1997 with no opposition, but some requests for minor changes. These changes have been made and the document is in its final stages of formal adoption, with no indication of any objections.

References

J. K. Reid. Two approaches to exception handling in Fortran 90. *in* R. F. Boisvert, ed., *'The Quality of Numerical Software: Assessment and Enhancement'*, pp. 210–223. Chapman and Hall, London, 1997.

5.4 F^{--}, a simple parallel extension to Fortran 90 (R. W. Numrich and J. K. Reid)

John Reid has been collaborating with Robert Numrich of Cray Research in the detailed design of an extension to Fortran 90 for parallel programming called F^{--} (Numrich and Steidel, 1997). The '–' signifies that it is a small extension.

An F^{--} program is interpreted as if it were replicated a number of times and that all copies were executed asynchronously. Each copy has its own set of data objects and is termed an 'image'. A data object is accessible only within its own image unless it is specified with additional dimensions in square brackets. Such an object has the same shape and address on all images and may be accessed from another image or from a set of images with the help of trailing subscripts enclosed in square brackets. The additional subscripts are mapped to images in the same way as Fortran array syntax maps sets of subscripts to memory locations. Such a 'co-array' may be used in expressions and assignments as if it were an ordinary Fortran array.

References without square brackets are to the local array (or scalar), so that code that can run independently is uncluttered. Only where there are square brackets or there is a procedure call is communication between images involved.

Array pointer components of co-arrays provide a mechanism for cases that require arrays to have different sizes on different images. They are carefully limited so that each references data only on its own image. However, the target may be addressed from elsewhere:

```
A(1:N) = B[P]%C(1:N)
```

There are intrinsic functions that return the number of images and the index of the current image. There is an intrinsic subroutine for synchronizing all images or a specified set of images.

The use of array notation to address data on other images provides a very flexible and clear mechanism for parallel programming. We illustrate this with a redistribution that is needed for 3-dimensional Fourier transforms. Given the declarations

```
INTEGER :: I,K,LX,LY,LZ
REAL :: A(LX,LY)[LZ], B(LY,LZ)[LX]
```

the code

```
K = THIS_IMAGE( ) + 1 ! Image indices commence at zero.
IF (K<=LZ) THEN
   DO I = 1, LX
      A(I,:) = B(:,K)[I]
   END DO
END IF
```

redistributes the co-array B in the co-array A.

The difficulties that we have faced have been associated with how the F^{--} extensions are integrated with the dynamic storage, pointer, and generic procedure mechanisms of

Fortran 90. We currently allow allocatable co-arrays, but require that each allocation occurs on every image and involves implicit synchronization of all images. We allow co-array pointers, but with strict rules to avoid unexpected consequences. We have decided not to allow co-array sections as actual arguments to avoid any change to the (complicated) generic procedure rules or unexpected consequences from copy-in copy-out across images.

The work is ongoing. Cray Research already have a subset implementation and it is hoped that a full implementation will be made during 1998.

References

R. W. Numrich and J. L. Steidel. F^{--}: A simple parallel extension to Fortran 90. *SIAM News*, **30**(7), 1–8, 1997.

5.5 Conditional compilation in Fortran (D. Epstein and J. K. Reid)

Conditional compilation (coco) in Fortran has been controversial for many years. Recently, the principle disagreement has been between those who see the existing C pre-processor `cpp` as adequate for Fortran and those who think Fortran should have its own 'Fortran-like' pre-processor. A majority of the ISO Committee WG5 has consistently been in favour of the Fortran-like approach.

Another source of disagreement has been over the power of the pre-processor and whether it should be used to avoid additions to the underlying language. David Epstein of Imagine Inc. has consistently argued in recent years in favour of a minimalist approach. He wants a very simple facility that may itself be written in Fortran (though it would be better integrated with the compiler) and which will be completely transparent to all users.

John Reid has collaborated with David in refining his proposal to one that is in its final stages of acceptance as an auxiliary standard that will constitute Part 3 of the Fortran standard.

The document defines a 'coco program' that will normally reside in a file and which will produce 'coco output' when executed. Execution is controlled by directives in a very short 'SET' file that will vary according to the version required and directives in the coco program itself. All directives have '??' in character positions 1 and 2. Directives are either omitted from the coco output or are changed to Fortran comments. Other lines (noncoco lines) are either copied unchanged to the output, omitted, or changed to Fortran comments.

A simple example is the coco program

```
?? LOGICAL :: USE_SECTIONS
```

```
   :
?? IF (USE_SECTIONS) THEN
   :
?? IF (USE_SECTIONS) THEN
     A(1:10,1:10) = B(1:10,1:10) + C(1:10,1:10)
?? ELSE
     DO I = 1, 10
       DO J = 1, 10
         A(I,J) = B(I,J) + C(I,J)
       ENDDO
     ENDDO
?? ENDIF
```

which permits a segment of code to be adapted according to whether the compiler is more efficient with array section syntax or with loop syntax. If run with the SET file

```
?? ALTER: DELETE  ! Delete directives
?? LOGICAL :: USE_SECTIONS = .TRUE.
```

its output is

```
   :
     A(1:10,1:10) = B(1:10,1:10) + C(1:10,1:10)
```

6 Miscellaneous Activities

6.1 CERFACS (I. S. Duff)

Iain has continued to lead a project at CERFACS on Parallel Algorithms and several of the contributions to this report reflect interactions with that team. A major recent activity at CERFACS has been the International Linear Algebra Year. This was held from September 1995 until September 1996 and included four workshops and a visitor's programme. Iain co-chaired the international scientific committee with Gene Golub from Stanford and is participating in all the workshops. The two final Workshops of the International Linear Algebra Year were held in 1996 and were each attended by about eighty researchers of whom half were from outside France. Nick helped to organize the optimization workshop that took place in Albi in April 1996, and the final Workshop on July was on iterative methods. Selected papers from all four workshops have appeared as special issues of BIT. Iain and Nick were two of the guest editors for the second issue (Volume **37**, Issue 3 of BIT) that contained papers from the Direct Methods, Iterative Methods, and Optimization workshops.

The main areas of research in the Parallel Algorithms Group are the development and tuning of kernels for numerical linear algebra, the solution of sparse systems using direct methods or iterative methods or a combination of the two, heterogeneous computing including the use of PVM and MPI and the design of schedulers, large eigensystem calculations, optimization, and the reliability of computations. Other activities of the Group include advanced training by both courses and research and the porting of industrial codes. Fuller details on the activities of the Parallel Algorithms Team can be found in the report CERFACS (1998).

During the reporting period, three students completed their PhDs at CERFACS. Iain was co-supervisor for Jacko Koster who finished his thesis in November 1996 (Koster, 1997). Jacko will come to RAL in February 1998 to work on the PARASOL Project.

Nick continued to visit both CERFACS and ENSEEIHT-IRIT with the support of a British Council grant to enable him to develop and extend some of the work commenced when he spent a year at CERFACS in 1994.

The main projects at CERFACS still include Parallel Algorithms and Computational Fluid Dynamics although recent emphasis on environmental modelling has led to a significant increase in the size of the Climate Modelling Group. There are smaller groups in electromagnetics, and signal processing.

The home page for CERFACS is `http://www.cerfacs.fr` and current information on the Parallel Algorithms Group can be found on page `http://www.cerfacs.fr/algor/` of the World Wide Web.

References

CERFACS. Activity report of the parallel algorithms project at CERFACS. January 1997 - December 1997. Technical Report SR/PA/98/06, CERFACS, 1998.

J. Koster. On the parallel solution and the reordering of unsymmetric sparse linear systems. INPT PhD Thesis TH/PA/97/51, CERFACS, Toulouse, France, 1997.

6.2 European project PARASOL, an integrated programming environment for PARAllel sparse matrix SOLvers (I. S. Duff)

PARASOL is a long term research (LTR) ESPRIT IV Project for "An Integrated Environment for Parallel Sparse Matrix Solvers". This Project started on January 1st, 1996, and its aim is to develop a parallel scalable library of sparse matrix solvers using Fortran 90 and MPI. At the end of the Project, the codes will be made available in the public domain.

The PARASOL Consortium is managed by PALLAS in Germany and consists of

- leading European research organizations with a well-known experience and track-record in the development of parallel solvers (CERFACS, GMD-SCAI, ONERA, RAL, Univ. of Bergen);
- industrial code developers who define the requirements for PARASOL and will use its results (Apex Technologies, Det Norske Veritas (DNV), INPRO, MacNeal-Schwendler, Polyflow);
- two leading European HPC software companies who will exploit the project results and will provide programming development tools (GENIAS, PALLAS).

The codes in the Library will include direct methods, domain decomposition techniques, and multigrid approaches. Within this project, RAL is involved in the development of direct solvers and is working in this context in close collaboration with CERFACS and with ENSEEIHT (Toulouse, France). More details on this work can be found in Section 2.2.

We had problems recruiting at the beginning of the Project both at RAL and CERFACS, but now have two people working full time, Jean-Yves L'Excellent joined the Project at CERFACS in October 1996 and is working on the MUMPS code (see Section 2.2). Petr Plecháč started at RAL at the beginning of November 1997 and has been working on combining graph partitioning with sparse matrix orderings and on the interface between MUMPS and PARASOL. Jacko Koster will join the PARASOL Team at RAL in February 1998.

There have been a number of meetings of the PARASOL Project over the past two years. After a kick-off meeting at MSC in Munich in January 1996, we had a one-day meeting in Paris in May and a two-day one at INPRO in Berlin in September. A one-day meeting to finalize the interface was held in December at Frankfurt airport. In the past year, there was a successfully negotiated Project Review meeting in Brussels in January, and a Project Meeting coupled with an open Workshop was held at CERFACS in September. We hosted a two-day meeting of the PARASOL Project partners at RAL in November, with participants staying in Cosener's House and business sessions in the Atlas Centre. At that meeting we decided to request the Commission that the Project be extended until June 30, 1999.

6.3 PARASOL interface to new parallel solvers for industrial applications (J. K. Reid and A. Supalov)

In connection with the PARASOL Project, John Reid collaborated very actively with Alexander Supalov of GMD on the design of a standard interface and the writing of the document that specifies it.

The aim of the interface is to provided a unified framework for users to specify problems to the various PARASOL solvers, some of which are direct and some of which are iterative. It needs to accommodate three modes of parallel execution:

- Host-node - one process inputs the problem definition and the others exchange data with it and compute the solution.
- Hybrid-host - one process inputs the problem definition and exchanges data with the others, but then helps the others to compute the solution.
- Hybrid-node - all processes input the problem definition and compute the solution.

The interface also needs to allow for several problems being in the process of solution at the same time, including the possibility of one solver calling another for a subproblem.

To achieve these goals, the PARASOL interface differs in several respects from the more conventional approaches of other parallel solver packages. The main characteristic of the PARASOL interface is that the data are passed to and from the solvers by data exchange routines instead of actual arguments to the solvers. This permits users to supply their data in manageable pieces, request an intermediate return of control, and request further solutions with changed right-hand side vectors or changed matrices. It also permits the solvers to use the data structures most suited to them and to have some control over the order in which data are provided.

Having flexibility on both sides cannot work without an agreement between the parties. Therefore, the computation is broken into a sequence of data exchange series. Each series starts with a negotiation in which the user specifies what data are desired to be sent and what answers are desired to be received. The package either accepts this contract or refuses it. In the case of refusal, the user has to seek an alternative way to obtain the desired result. The hints returned by the PARASOL package can be used to help. If the contract is accepted, the user then has to make a series of calls that accord with the contract and the data exchange protocol. The package will check that the calls are done in an appropriate order.

For further details, see Reid, Supalov and Thole (1998).

References

J. K. Reid, A. Supalov, and C-A Thole. PARASOL interface to new parallel solvers for industrial applications. *in* 'Proceedings ParCo '97'. Elsevier, 1998.

6.4 IFIP Working Group 2.5 (J. K. Reid)

There were two meetings of IFIP Working Group 2.5 (mathematical software) during the reporting period and John Reid played an active role in both. In July 1996, the Group organized a working conference on 'The Quality of Numerical Software: Assessment and Enhancement' and he gave a talk (Reid, 1997) on 'Two Approaches to Exception Handling in Fortran 90'. In October 1997, the Group organized a workshop on 'Numerical and Programming Environment Aspects of DOE's ASCI (Advanced Scientific Computing Initiative) Modelling and Simulation Projects' and (in place of a speaker who was unable to attend) he gave a talk on the F^{--} language, which was well received and promoted a lively discussion.

John's main activity for the Working Group was in connection with the Group's wish to see the addition of exception handling to Fortran, which has reached a successful conclusion (see Section 5.3).

References

J. K. Reid. Two approaches to exception handling in Fortran 90. *in* R. F. Boisvert, ed., 'The quality of numerical software: assessment and enhancement', pp. 210–223. Chapman and Hall, London, 1997.

7 Computing and the Harwell Subroutine Library

7.1 The computing environment within the Group

Our policy of upgrading the Group's workstations has continued over the past two years. The main change has been that the Group now runs 3 SUN Ultra Sparc 1s as replacements for Iain, John and Jennifer's SUN Sparc 10/30s. Petr has inherited one of the older machines, while the others have replaced our previous generation of SUN Sparc 1s, all of which have finally been retired. Nick had intended to replace his IBM RISC Systems/6000 3BT with the next generation of IBM machine, but recent announcements of significantly better machines from IBM and others has meant that we have delayed this upgrade until 1998. The Group's IBM Thinkpad 701 portable computer has proved to be less reliable than had been hoped, but it has still proved useful for Group members on their frequent travels.

The responsibility for SUN software support has been delegated to other parts of the Department, although Group members have still found it more convenient to get their hands dirty for simple tasks. The Group continues to support a series of World Wide Web pages describing its activities. This has resulted in much wider publicity for the Group, and made it easier for external users to access our technical reports and publicly-available software. In addition, we maintain and update pages of links to other relevant numerical analysis information.

We continue to benefit from other DCI machines, in particular the DEC Alpha 3000 and HP-9000 farms, the CRAY J932 and the newer DEC 8400 multiprocessor system. Most of the Group's files now reside on a central UNIX data store, which is backed up on a daily basis by the Department. We now have access to eight Fortran 90 compilers, some on our own machines and some on other DCI machines. This has enabled our gradual transformation from Fortran 77 to 90 to proceed, and in some cases we have found these compilers uncovered previously hidden errors in our existing Fortran 77 codes. We have also installed MPI and associated parallel language support systems on some of our machines, as our development of parallel algorithms continues.

7.2 The Harwell Subroutine Library

The following new packages were added to the Library during the reporting period.

`MA62` (I. S. Duff and J. A. Scott)

This package solves one or more sets of sparse symmetric linear unassembled finite-element equations, $\boldsymbol{AX} = \boldsymbol{B}$, by the frontal method, optionally holding the matrix factor

out-of-core in direct access files. The package is primarily designed for positive-definite matrices since numerical pivoting is not performed. Use is made of high-level BLAS kernels. The coefficient matrix $\boldsymbol{A}$ must of the form

$$\boldsymbol{A} = \sum_{k=1}^{m} \boldsymbol{A}^{(k)},$$

with $\boldsymbol{A}^{(k)}$ nonzero only in those rows and columns that correspond to variables in the k-th element.

The frontal method is a variant of Gaussian elimination and involves the factorization

$$\boldsymbol{A} = \boldsymbol{PLD}(\boldsymbol{PL})^{\mathrm{T}},$$

where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{D}$ is a diagonal matrix, and $\boldsymbol{L}$ is a unit lower triangular matrix. `MA62` stores the reals of the factors and their indices separately. A principal feature of `MA62` is that, by holding the factors out-of-core, large problems can be solved using a predetermined and relatively small amount of in-core memory. At an intermediate stage of the solution, l say, the 'front' contains those variables associated with one or more of $\boldsymbol{A}^{(k)}$, $k = 1, 2, \ldots, l$, which are also present in one or more of $\boldsymbol{A}^{(k)}$, $k = 1, 2, \ldots, m$. For efficiency, the user should order the $\boldsymbol{A}^{(k)}$ so that the number of variables in the front (the 'front size') is small. For example, a very rectangular grid should be ordered pagewise parallel to the short side of the rectangle. The elements may be preordered using the Harwell Subroutine Library routine `MC43`.

`MC60` (J. K. Reid and J. A. Scott)

This subroutine uses a variant of Sloan's method to calculate a symmetric permutation that aims to reduce the profile and wavefront of a sparse matrix $\boldsymbol{A}$ with a symmetric sparsity pattern. Alternatively, the Reverse Cuthill-McKee Method may be requested to reduce the bandwidth. There are optional facilities for looking for sets of columns with identical patterns and taking advantage of them. There is also an option for computing a row order that would be appropriate for use with a row-by-row frontal solver (for example, the equation entry to `MA42`). These optional facilities may also be used independently.

`MC61` (J. K. Reid and J. A. Scott)

Let $\boldsymbol{A}$ be an $n \times n$ sparse matrix with a symmetric sparsity pattern. Given the sparsity pattern of $\boldsymbol{A}$, this subroutine uses a variant of Sloan's method to calculate a symmetric permutation that aims to reduce the profile and wavefront of $\boldsymbol{A}$. Alternatively, the Reverse Cuthill-McKee (RCM) method may be requested to reduce the bandwidth, or the user may request an ordering for the rows of $\boldsymbol{A}$ that is efficient when used with a row-by-row frontal solver (for example, equation entry to `MA42`).

MC61 provides the user with a straightforward interface to the MC60 package when detailed control of the steps in constructing a symmetric permutation or row ordering is not required.

HSL_MI02 (N. I. M. Gould)

This routine uses the SYMMBK method to solve the $n \times n$ symmetric but possibly indefinite linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, optionally using preconditioning. If $\boldsymbol{P}\boldsymbol{P}^{\mathrm{T}}$ is the preconditioning matrix, the routine actually solves the preconditioned system

$$\bar{\boldsymbol{A}}\bar{\boldsymbol{x}} = \bar{\boldsymbol{b}},$$

with $\bar{\boldsymbol{A}} = \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^{\mathrm{T}}$ and $\bar{\boldsymbol{b}} = \boldsymbol{P}\boldsymbol{b}$ and recovers the solution $\boldsymbol{x} = \boldsymbol{P}^{\mathrm{T}}\bar{\boldsymbol{x}}$. Reverse communication is used for preconditioning operations and matrix-vector products of the form $\boldsymbol{A}\boldsymbol{z}$.

HSL_VF05 (N. I. M. Gould)

Given real $n \times n$ symmetric matrices $\boldsymbol{H}$ and $\boldsymbol{M}$ (with $\boldsymbol{M}$ positive definite), a real n vector c and a positive scalar Δ, this package finds an approximate minimizer of the quadratic objective function

$$\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x},$$

where the vector $\boldsymbol{x}$ is required to satisfy the constraint $\| \boldsymbol{x} \|_M \leq \Delta$, and where the M-norm of $\boldsymbol{x}$ is $\| \boldsymbol{x} \|_M = \sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}}$. This problem commonly occurs as a trust-region subproblem in nonlinear optimization calculations. The method may be suitable for large n as no factorization of $\boldsymbol{H}$ is required. Reverse communication is used to obtain matrix-vector products of the form $\boldsymbol{H}\boldsymbol{z}$ and $\boldsymbol{M}^{-1}\boldsymbol{z}$.

8 Seminars

11 January 1996	Dr T Braconnier (Manchester) Computing the fields of values and pseudospectra using the Lanczos method with continuation.
16 January 1996	Dr J Reid (Rutherford Appleton Laboratory) The current status of Fortran 90 and Fortran 95.
1 February 1996	Dr K Meerbergen (Leuven, Belgium) The calculation of eigenvalues of Navier Stokes problems.
9 May 1996	Dr I Smith (Liverpool) A comparative study of solution strategies for domain decomposition reordered systems.
31 October 1996	Dr L Hemmingsson (Uppsala, Sweden) Domain decomposition methods and fast solvers for first-order PDEs.
30 January 1997	Professor John Pryce (Shrivenham) Solving systems of differential algebraic equations by Taylor series.
6 March 1997	Dr Stephen Zitney (AspenTech) Sparse matrix methods for dynamic chemical process simulation.
18 March 1997	Drs F. Malucelli and L. Tarricone (Perugia, Italy) Bandwidth minimization algorithms.
11 April 1997	Dr Martin van Gijzen (Utrecht, The Netherlands) GMRES-like methods on distributed memory computers.
8 May 1997	Istvan Maros (Imperial College) On numerically exact implementation of the simplex method.
15 May 1997	Dr Mike Osborne (Canberra, Australia) Wrap-around partitioning for block bidiagonal linear systems.
19 June 1997	Dr George Goodsell (Cambridge) A new iterative method for thin plate spline interpolation to scattered data.
27 November 1997	Dr Sven Leyffer (Dundee) An integrated approach to integer and nonlinear optimization.

9 Reports issued in 1996-1997

We give a full listing of Rutherford Technical Reports issued during the period of this Progress Report. The other report listings, from organizations with which we collaborate, only include reports not already included as RAL reports. All of our current technical reports are publicly accessible via the internet from

"http://www.rl.ac.uk/departments/ccd/numerical/reports/reports.html".

Rutherford Reports

RAL-TR-96-010 MA46, a Fortran code for direct solution of sparse unsymmetric linear systems of equations from finite-element applications. A. C. Damhaug and J. K. Reid.

RAL-TR-96-013 A projection method for the solution of rectangular systems. J. Cardenal, I. S. Duff, and J. M. Jimenez.

RAL-TR-96-014 A blocked implementation of Level 3 BLAS for RISC processors. M. J. Daydé and I. S. Duff.

RAL-TR-96-015 Numerical Analysis Group - Progress report. January 1994 - December 1995. I. S. Duff (Editor).

RAL-TR-96-022 An evaluation of subspace iteration software for sparse nonsymmetric eigenproblems. R. B. Lehoucq and J. A. Scott.

RAL-TR-96-023 An evaluation of Arnoldi based software for sparse nonsymmetric eigenproblems. R. B. Lehoucq and J. A. Scott.

RAL-TR-96-042 Methods for nonlinear constraints in optimization calculations. A. R. Conn, N. I. M. Gould, and Ph. L. Toint.

RAL-TR-96-047 Sparse numerical linear algebra: direct methods and preconditioning. I. S. Duff.

RAL-TR-96-096 A primal-dual algorithm for minimizing a non-convex function subject to bound and linear equality constraints. A. R. Conn, N. I. M. Gould, and Ph. L. Toint.

RAL-TR-96-102 A comparison of frontal software with other sparse direct solvers. I. S. Duff and J. A. Scott.

RAL-TR-97-001 Performance issues for frontal schemes on a cache-based high performance computer. K. A. Cliffe, I. S. Duff, and J. A. Scott.

RAL-TR-97-012 MA62 - a frontal code for sparse positive-definite symmetric systems from finite-element applications. I. S. Duff and J. A. Scott.

RAL-TR-97-028 Solving the trust-region subproblem using the Lanczos method. N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint.

RAL-TR-97-031 The Rutherford-Boeing sparse matrix collection. I. S. Duff, R. G. Grimes, and J. G. Lewis.

RAL-TR-97-041 Exploiting zeros in frontal solvers. J. A. Scott.

RAL-TR-97-046 A combined unifrontal/multifrontal method for unsymmetric sparse matrices. T. A. Davis and I. S. Duff.

RAL-TR-97-054 A numerical comparison between the LANCELOT and MINOS packages for large-scale nonlinear optimization. I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and Ph. L. Toint.

RAL-TR-97-055 A note on the second-order convergence of optimization algorithms using barrier functions. N. I. M. Gould and Ph. L. Toint.

RAL-TR-97-058 Implicitly restarted Arnoldi methods and eigenvalues of the discretized Navier Stokes equations. R. B. Lehoucq and J. A. Scott.

RAL-TR-97-059 The design and use of algorithms for permuting large entries to the diagonal. I. S. Duff and J. Koster.

RAL-TR-97-064 Exploiting negative curvature directions in linesearch methods for unconstrained optimization. N. I. M. Gould, S. Lucidi, M. Roma, and Ph. L. Toint.

RAL-TR-97-071 The modified absolute-value factorization norm for trust-region minimization. N. I. M. Gould and J. Nocedal.

CERFACS Reports

TR/PA/96/47 Use of computational kernels in full and sparse linear solvers, efficient code design on high-performance RISC processors. M. J. Daydé and I. S. Duff.

FUNDP Reports

97/14 A numerical comparison between the LANCELOT and MINOS packages for large-scale nonlinear optimization: the complete results. I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders, and Ph. L. Toint.

ENSEEIHT-IRIT Reports

RT/APO/97/2 A blocked implementation of Level 3 BLAS for RISC processors (revised version). M. J. Daydé and I. S. Duff.

RT/APO/97/3 Solution of unassembled linear systems using block stretching: Preliminary experiments. M. J. Daydé, J. Décamps, and N. I. M. Gould.

Argonne National Laboratory Reports

MCS-P547-1195 An evaluation of software for computing eigenvalues of nonsymmetric matrices. R. B. Lehoucq and J. A. Scott.

10 External Publications in 1996-1997

P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, **17**(4), 886–905, 1996.

P. R. Amestoy, I. S. Duff, and C. Puglisi. Multifrontal QR factorization in a multiprocessor environment. *Numerical Linear Algebra with Applications*, **3**(4), 275–300, 1996.

L. Carvalho, I. S. Duff, and L. Giraud. Linear algebra kernels for parallel domain decomposition methods. *In* M. Papadrakakis and G. Bugeda, eds, 'Advanced Computational Methods in Structural Mechanics', pp. 1–17. International Centre for Numerical Methods in Engineering (CIMNE), Barcelona, Spain, 1996.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization. *Mathematical Programming*, **73**(1), 73–110, 1996.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Methods for nonlinear constraints in optimization calculations. *in* I. S. Duff and G. A. Watson, eds, 'The State of the Art in Numerical Analysis', pp. 363–390, Oxford, 1997. Oxford University Press.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. A globally convergent Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. *Mathematics of Computation*, **66**, 261–288 and S1–S11, 1997.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. On the number of inner iterations per outer iteration of a globally convergent algorithm for optimization with general nonlinear inequality constraints and simple bounds. *Computational Optimization and Applications*, **7**(1), 41–69, 1997.

A. R. Conn, N. I. M. Gould, A. Sartenaer, and Ph. L. Toint. Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints. *SIAM Journal on Optimization*, **6**(3), 674–703, 1996.

A. R. Conn, N. I. M. Gould, A. Sartenaer, and Ph. L. Toint. Convergence properties of minimization algorithms for convex constraints using a structured trust region. *SIAM Journal on Optimization*, **6**(4), 1059–1086, 1996.

A. R. Conn, N. I. M. Gould, A. Sartenaer, and Ph. L. Toint. On iterated-subspace minimization methods for nonlinear optimization. *In* L. Adams and L. Nazareth, eds, 'Proceedings on Linear and Nonlinear Conjugate Gradient-Related Methods', pp. 50–78, SIAM, Philadelphia, 1996.

T. A. Davis and I. S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM Journal on Matrix Analysis and Applications*, **18**(1), 140–158, 1997.

M. J. Daydé and I. S. Duff. The use of computational kernels in full and sparse linear solvers, efficient code design on high-performance RISC processors. *In* J. M. L. M. Palma and J. Dongarra, eds, 'Vector and Parallel Processing - VECPAR'96', Lecture Notes in Computer Science **1215**, pp. 108–139, Springer, Berlin, 1997.

M. J. Daydé, J. P. Décamps, J.-Y. L'Excellent, and N. I. M. Gould. Solution of large scale partially separable unconstrained optimization problems using element-by-element preconditioners. *In* 'Proceedings of NAFEMS World Congress 97', Vol. 2, pp. 942–953, 1997.

M. J. Daydé, J.-Y. L'Excellent, and N. I. M. Gould. Preprocessing of sparse unassembled linear systems for efficient solution using element-by-element preconditioners. *In* A. M. L. Bourgé, P. Fragniaud and Y. Roberts, eds, 'Proceedings of Euro-Par 96, Lyons', number 1124 *in* 'Lecture Notes in Computer Science', pp. 34–43, Springer Verlag, Heidelberg, Berlin, New York, 1996.

M. J. Daydé, J.-Y. L'Excellent, and N. I. M. Gould. Element-by-element preconditioners for large partially separable optimization problems. *SIAM Journal on Scientific Computing*, **18**(6), 1767–1787, 1997.

I. S. Duff. A review of frontal methods for solving linear systems. *Computer Physics Communications*, **97**(1&2), 45–52, 1996.

I. S. Duff. Sparse numerical linear algebra: direct methods and preconditioning. *in* I. S. Duff and G. A. Watson, eds, 'The State of the Art in Numerical Analysis', pp. 27–62, Oxford, 1997. Oxford University Press.

I. S. Duff and J. K. Reid. The design of MA48, a code for the direct solution of sparse unsymmetric linear systems of equations. *ACM Transactions on Mathematical Software*, **22**(2), 187–226, 1996.

I. S. Duff and J. K. Reid. Exploiting zeros on the diagonal in the direct solution of indefinite sparse symmetric linear systems. *ACM Transactions on Mathematical Software*, **22**(2), 227–257, 1996.

I. S. Duff and J. A. Scott. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Transactions on Mathematical Software*, **22**, 30–45, 1996.

I. S. Duff and J. A. Scott. Frontal software for the solution of sparse linear equations. *In* J. Wasniewski, J. Dongarra, K. Madsen and D. Olesen, eds, 'Applied Parallel Computing. Industrial Computation and Optimization. Proceedings of the Third International Workshop, PARA '96', Lecture Notes in Computer Science 1184, pp. 227–238, Springer Verlag, Heidelberg, Berlin, New York, 1996.

I. S. Duff and A. Watson, editors. *The State of the Art in Numerical Analysis*, Oxford University Press, Oxford, 1997.

S. Jones, B. Jumarhon, S. McKee, and J. A. Scott. A mathematical model of a biosensor. *Journal of Engineering Mathematics*, **30**, 321–337, 1996.

M. Metcalf and J. K. Reid. *The F programming language*. Oxford University Press, Oxford, 1996.

M. Metcalf and J. K. Reid. *Fortran 90/95 explained*. Oxford University Press, Oxford, 1996.

J. K. Reid. An appreciation of F. *Fortran Journal*, **8**(6), 3–9, 1996.

J. K. Reid. The array features. *Computer Standards and Interfaces*, **18**, 323–331, 1996.

J. K. Reid. Remark on fast floating-point processing in common lisp. *ACM Transactions on Mathematical Software*, **22**, 496–497, 1996.

J. K. Reid. Two approaches to exception handling in Fortran 90. *In* R. F. Boisvert, ed., 'The Quality of Numerical Software: Assessment and Enhancement', pp. 210–223. Chapman and Hall, London, 1997.

J. A. Scott. Element resequencing for use with a multiple front solver. *International Journal of Numerical Methods in Engineering*, **39**, 3999–4020, 1996.