COMMUNICATIONS IN APPLIED NUMERICAL METHODS, Vol. 3, 23–29 (1987)

THE PERFORMANCE OF ITPACK ON VECTOR COMPUTERS FOR SOLVING LARGE SPARSE LINEAR SYSTEMS ARISING IN SAMPLE OIL RESERVOIR SIMULATION PROBLEMS†

THOMAS C. OPPE AND DAVID R. KINCAID

Center for Numerical Analysis, The University of Texas at Austin, Austin, Texas 78712–1067, U.S.A.

SUMMARY

Results are presented of the performance of two different versions of the ITPACK software package on large sparse linear systems arising in the solution of sample oil reservoir simulation problems using three different large vector computers (Cyber 205, Cray X-MP48 and Amdahl VP1200). The main focus of this paper is on the necessary changes in the package that were made for efficiency when going from a memory-to-memory computer to a register-to-register computer. We feel that these are typical changes that must be considered when moving software of this type between vector computers. The execution times presented should not be considered to be representative of the overall performance of these computers.

INTRODUCTION

In this paper, a comparison is made between two different versions of the ITPACK software package for solving the sparse systems of linear algebraic equations arising from typical oil reservoir modelling problems. Various iterative algorithms are compared on the basis of numbers of iterations, required workspace, and timings on three different vector computers. ITPACK 2C is the version of the package originally written for use on scalar computers and ITPACKV 2C is the modified package which runs much more efficiently on vector computers. The programming changes necessary to achieve this latter version of the package are discussed.

The tests were carried out on a set of linear systems available from J. S. Nolen & Associates.[1] The linear systems were taken from actual simulation runs on different types of simulators and they appear to be relatively difficult to solve by commonly used methods. The sample systems are representative of a wide spectrum of reservoir modelling situations, including the effects of faults, implicit bottom-hole pressure and steam injection.

ITPACK was designed as a research tool for the study of selected iterative methods. In particular, it was not designed for optimal efficiency for the structured sparse problems that often result from reservoir modelling problems. The matrix representation schemes used in the two versions of ITPACK discussed in this paper make no assumptions about the pattern of non-zeros in the matrix. We understand that any researcher who wants to solve a difficult physical reservoir simulation problem would develop a special purpose linear equation solver as part of the total simulator which was carefully tailored to the matrix structure and to the vector computer being used. Nevertheless, we feel that ITPACK provides a good experimental tool to explore the convergence rates of various iterative algorithms, of their range of applications, and of characteristics of their software implementation. Based on these experimental results using sample problems, it may be possible to select an iterative algorithm which suits the particular application and computer architecture at hand.

† This work was supported in part by the National Science Foundation under Grant MCS-8214731 and by the Department of Energy under Grant DE-AS05-81ER10954 with The University of Texas at Austin. Thomas C. Oppe's participation was supported by the Control Data Corporation through its PACER Fellowship Program.

0748–8025/87/010023–07$05.00 *Received May 1986*

Table I

Problem	1	2	3	4	5
NX	10	6	35	16	16
NY	10	6	11	23	23
NZ	10	5	13	3	3
NB	1	6	1	1	3
N	1000	1080	5005	1104	3312

PROBLEMS AND PACKAGES

The five sample oil reservoir simulation problems all exhibit a three-dimensional structure similar to those that would be obtained by the use of a 7-point finite difference stencil over a three-dimensional box domain with possible multiple unknowns at each grid point. Boundary values problems of this type are described throughout the literature (see, for example, Reference 2). In Table I, *NX*, *NY* and *NZ* are the number of *X*, *Y* and *Z* grid lines, respectively, *NB* is the number of unknowns per grid point, and $N = NX*NY*NZ*NB$ is the total number of unknowns.

Problem 1 is symmetric and positive definite (SPD), problems 3 and 4 are slightly non-symmetric, and problems 2 and 5 are strongly non-symmetric. Also, problems 1, 3 and 4 have point Property A, while problems 2 and 5 have block Property A. Problems 1, 3 and 4 have a point seven-diagonal structure, while problems 2 and 5 have a block seven-diagonal structure. In each problem the diagonals are located as follows: Problem 1: 0, ±1, ±10, ±100; problem 2: 0, ±1, . . . , ±11, ±31, . . . , ±41, ±211, . . . , ±221; problem 3: 0, ±1, ±35, ±385; problem 4: 0, ±1, ±16, ±368; problem 5: 0, ±1, . . . , ±5, ±46, . . . , ±50, +1102, +1103, ±1104, . . . , ±1106.

In this paper, timings on three vector computers are given for various iterative solution methods contained in two iterative software packages. The computers used were the CDC Cyber 205 at Colorado State University (2-pipe), the Cray X-MP48 at Mendota Heights, Minnesota (used in uniprocessor mode), and the Amdahl VP1200 at Sunnyvale, California. The two packages used were ITPACK 2C and ITPACKV 2C. A short description of each package and the changes made to the packages for these computers is given below. (For more information on ITPACK or ITPACKV, see References 3 and 4.

ITPACK 2C contains the Jacobi (J), Symmetric Successive Over-Relaxation (SSOR), and Reduced System (RS) basic iterative methods accelerated by either conjugate gradient (CG) or Chebyshev (SI, for semi-iterative) acceleration. Successive Over-Relaxation (SOR) with no acceleration is also included.

ITPACK stores the matrix in the A-JA-IA sparse data structure. With this structure, the non-zeros of the matrix are stored by rows in the linear real array A. The linear integer array JA of the same length is used to store the corresponding column numbers of the non-zeros. Then a linear integer array IA contains pointers to the starting positions of new rows in the previous two arrays. (Details on the A-JA-IA storage format is available from many sources. For example, see References 3 and 5.) With this data structure, a row-oriented inner product algorithm is used for the matrix-vector product, Ax, and other computational kernels. Since the condensed rows of the matrix are typically short, vectorization was not profitable for these kernels.

For the computer runs given in this paper, the only changes made to the ITPACK code were to re-roll DO loops that had been unrolled for efficiency on scalar computers, but which prevented vectorization on vector computers. The routines affected were BLAS (Basic Linear Algebra Subprograms) or BLAS-like routines (see Reference 6 for information on the BLAS). No attempt was made to bring short vectorizable routines such as these in-line to save subroutine linkage times. Also, no attempt was made to inhibit vectorization when the length of the loop made vectorization inadvisable. The same code was run on all three vector machines.

ITPACKV 2C contains the same algorithms as the version of the package for scalar computers ITPACK 2C, but utilizes a different sparse matrix storage scheme called the COEF–JCOEF data structure. With this format, the non-zeros of the matrix are stored in the array COEF and their corresponding column numbers are stored in the array JCOEF. These are both two-dimensional arrays of size *N* by MAXNZ, where *N* is the number of unknowns and MAXNZ is the maximum number of non-zeros per row of the matrix. (For details on the COEF–JCOEF storage format

see References 7 and 8). Thus a seven-diagonal matrix, such as three of the five problems considered in this paper, could be stored in two N by 7 arrays with COEF containing a diagonal of the matrix in each column and JCOEF containing the column numbers. For a diagonally structured problem, JCOEF is clearly unnecessary, but vector ITPACKV does not assume that the columns of COEF represent diagonals. The operation Ax is computed using gather operations on vector x according to the index lists contained in the columns of JCOEF. Thus, MAXNZ gather operations of length N are needed to compute the product.

ITPACKV 2C was extensively modified for these experiments, and slightly different versions were used on the three computers. First, all BLAS-like routines were placed in-line as vectorizable loops. In the Cyber 205 version, calls to the gather and scatter routines (Q8VGATHR and Q8VSCATR, respectively) were also placed in-line where needed. This was also the only time when the Cyber 205 vector syntax was employed. The Fortran compiler with options O=BOUV was used. The virtual memory was mapped to large pages with GRLPALL parameter. On the Cray X-MP, the gather operations were done implicitly in the vector registers, as shown in the example below. On this machine, the NEXTSCALAR and IVDEP compiler directives were used to prevent vectorization of very short loops and to force vectorization of potentially recursive loops, respectively. On the Amdahl VP1200, the Cray version of ITPACKV was run with the corresponding compiler directives invoked.

The matrix-vector product kernel, Ax, is important since it is the most time-consuming operation for many iterative methods. This includes the Jacobi method for natural ordering and all methods in ITPACKV for red-black ordering. The change to the COEF-JCOEF data structure in ITPACKV was made primarily to make this operation vectorizable. The code for accomplishing $y = Ax$ in ITPACKV on the Cyber 205 is given below:

```
      DO 10 I = 1,N
10    Y(I) = 0.0
      DO 20 J = 1, MAXNZ
        WORK(1;N) = Q8VGATHR (X(1;N), JCOEF (1,J;N);WORK(1;N))
        DO 15 I = 1,N
15      Y(I) = Y(I) + COEF(I,J)*WORK(I)
20    CONTINUE
```

Loops 10 and 15 vectorize on the Cyber 205 and Q8VGATHR is a hardware vector instruction.

The code for the same operation on the Cray X-MP went through several revisions. First, the particular X-MP used had a hardware gather instruction so the Q8VGATHR could be replaced with a loop employing indirect addressing which vectorized:

```
      DO 10 I = 1,N
10    Y(I) = 0.0
      DO 25 J = 1, MAXNZ
        DO 15 I = 1,N
15      WORK(I) = X(JCOEF(I,J))
        DO 20 I = 1,N
20      Y(I) = Y(I) + COEF(I,J)*WORK(I)
25    CONTINUE
```

Loops 10, 15 and 20 vectorized on the Cray X-MP. Then, it was decided to combine loops 15 and 20 into a single loop, thereby eliminating the workspace array WORK:

```
      DO 10 I = 1,N
10    Y(I) = 0.0
      DO 25 J = 1, MAXNZ
        DO 20 I = 1,N
20      Y(I) = Y(I) + COEF(I,J)*X(JCOEF(I,J))
25    CONTINUE
```

Loops 10 and 20 vectorized. This modification had the advantage of reducing the workspace requirements for all methods in ITPACKV (except SOR) by N, since it was no longer necessary to explicitly gather the X unknowns into a separate vector. For the next revision, it was decided to unroll the 25 loop to a depth of four. This was done to reduce memory traffic between main memory and the vector registers (see Reference 9). The Y array is an accumulation vector and

so it is advantageous to force it to remain in the vector registers as long as possible. The code above would force the Y array to travel back and forth between the memory and vector registers. The unrolled version of the code above will not be given since it is somewhat tedious. The final pass at an improvement was the coding of this kernel in CAL, or Cray Assembly Language. Using CAL, it is possible to effectively unroll the 25 loop to an infinite depth, but since MAXNZ is typically small, little improvement was seen. For the test runs given below, the unrolled Fortran version was used.

NOTES ON TIMINGS

Several remarks should be made about the timings given in Tables II–VII.

For each routine, the number of iterations required until convergence is given using ZETA = $1{\cdot}0E{-}6$ for the stopping test. The stopping test used was:

$$\frac{1}{1 - M(G)} \sqrt{\left[\frac{(\delta^{(n)}, \delta^{(n)})}{(u^{(n)}, u^{(n)})}\right]}$$

where $u^{(n)}$ is the current approximate solution, $\delta^{(n)}$ is the current pseudo-residual, and $M(G)$ is an estimate of the maximum eigenvalue of the iteration matrix G corresponding to the preconditioner.

Also, two timings are given in seconds, Time1 and Time2. Time1 includes just the iteration time. Time2 includes this time as well as the time to scale and unscale the system, and, if red-black ordering is used, the time to permute and unpermute the linear system. In ITPACKV, these operations vectorize but they do not in ITPACK.

In addition to the timings, the amount of workspace required for each routine is given. This figure does not include the workspace necessary for storage of the solution vector, the right-hand-side vector, the coefficient matrix, or any integer vectors needed for permutation of the matrix. The workspace requirements for the Cyber 205 version of ITPACKV is N greater for all methods except SOR, which requires the same amount of workspace. The workspace requirements listed are for the Cray X-MP and Amdahl versions of ITPACKV.

NOTES ON METHODS

None of the methods in either version of ITPACK converged for the two badly non-symmetric problems. Thus, timings are given only for problems 1, 3 and 4. In addition, some adaptation routines for adaptive parameter determination failed in both versions of ITPACK. For both natural and red-black orderings, the adaption scheme for ω in the SSORCG method failed. It was decided to use the following plan:

- For the SSORCG method with natural ordering, use non-adaptively the final parameters computed from the SSORSI run, which was successful.
- For both the SSORCG and SSORSI methods with red-black ordering, use $\omega = 1$ non-adaptively. It can be shown that this value of ω is optimal for SSOR in red-black ordering.

Another discovery in using the ITPACK software involved the eigenvalue estimation procedure used with conjugate gradient in the non-symmetric case. In the non-symmetric case, this procedure is much more time-consuming than the procedure in the symmetric case. It can sometimes dominate the time needed to do the iterations themselves. The eigenvalue estimation procedure also has not been vectorized. This explains why in some of the timings reported in the tables for the non-symmetric problems, the SI routines perform almost as well or better than the corresponding CG routines, even though more iterations are being done.

CONCLUSIONS

For ITPACK, the results presented in Tables II–IV show that RSCG was the best routine for all problems on all three computers. SSORCG often ran a close second. Since ITPACK is predominantly not vectorizable, the timings are roughly proportional to the number of operations done. Hence, RSCG and SSORCG do well since they require the fewest number of iterations.

Table II. ITPACK on problem 1

Method	Iterations	Workspace	Cyber 205 Time 1	Cyber 205 Time 2	Cray X-MP Time 1	Cray X-MP Time 2	Amdahl VP1200 Time 1	Amdahl VP1200 Time 2
			Natural ordering					
JCG	268	4536	0·733	0·752	0·595	0·608	0·591	0·612
JSI	698	2000	1·739	1·758	1·475	1·488	1·523	1·541
SOR	380	1000	1·623	1·642	1·433	1·446	1·367	1·385
SSORCG	114	6228	1·090	1·110	0·921	0·935	0·950	0·969
SSORSI	270	5000	2·176	2·194	1·797	1·810	1·728	1·747
			Red-black ordering					
JCG	268	4536	0·700	0·833	0·525	0·619	0·481	0·601
JSI	698	2000	1·652	1·785	1·281	1·375	1·222	1·343
SOR	393	1000	1·626	1·759	1·147	1·240	1·181	1·301
SSORCG	131	6262	1·215	1·348	0·832	0·925	0·941	1·061
SSORSI	354	5000	2·733	2·866	1·734	1·827	1·800	1·922
RSCG	134	2306	0·450	0·585	0·370	0·465	0·379	0·501
RSSI	369	1346	1·185	1·320	0·996	1·091	1·028	1·150

Table III. ITPACK on problem 3

Method	Iterations	Workspace	Cyber 205 Time 1	Cyber 205 Time 2	Cray X-MP Time 1	Cray X-MP Time 2	Amdahl VP1200 Time 1	Amdahl VP1200 Time 2
			Natural ordering					
JCG	493	21992	10·112	10·246	6·148	6·228	6·548	6·675
JSI	1220	10010	21·851	21·981	13·766	13·841	15·030	15·153
SOR	687	5005	14·118	14·247	5·279	8·353	10·948	11·070
SSORCG	123	30522	5·502	5·632	3·121	3·196	4·627	4·749
SSORSI	355	25025	16·587	16·716	9·065	9·140	12·217	12·340
			Red-black ordering					
JCG	493	21992	10·112	11·244	6·113	6·930	6·313	7·477
JSI	1220	10010	21·850	22·982	13·679	14·496	14·504	15·674
SOR	726	5005	14·921	16·053	8·696	9·513	11·767	12·937
SSORCG	237	30978	10·383	11·516	5·901	6·718	8·733	9·905
SSORSI	584	25025	26·359	27·491	14·199	15·015	19·372	20·554
RSCG	244	10322	4·350	5·490	2·788	3·610	3·480	4·656
RSSI	604	6452	9·971	11·111	6·623	7·445	8·356	9·537

Table IV. ITPACK on problem 4

Method	Iterations	Workspace	Cyber 205 Time 1	Cyber 205 Time 2	Cray X-MP Time 1	Cray X-MP Time 2	Amdahl VP1200 Time 1	Amdahl VP1200 Time 2
			Natural ordering					
JCG	255	5436	2·919	2·946	1·160	1·176	1·468	1·494
JSI	309	2208	1·092	1·117	0·720	0·735	0·768	0·793
SOR	174	1104	0·703	0·728	0·436	0·452	0·512	0·537
SSORCG	63	6876	0·907	0·932	0·426	0·442	0·627	0·652
SSORSI	98	5520	0·937	0·963	0·535	0·551	0·652	0·676
			Red-black ordering					
JCG	255	5436	2·919	3·116	1·159	1·301	1·448	1·636
JSI	309	2208	1·091	1·289	0·720	0·862	0·751	0·936
SOR	174	1104	0·703	0·899	0·436	0·579	0·517	0·705
SSORCG	83	6956	0·730	0·927	0·432	0·574	0·610	0·794
SSORSI	166	5520	1·493	1·690	0·841	0·984	0·976	1·163
RSCG	95	2303	0·331	0·530	0·218	0·362	0·252	0·439
RSSI	147	1377	0·484	0·682	0·328	0·472	0·379	0·566

Table V. ITPACKV on problem 1

Method	Iterations	Workspace	Cyber 205 Time 1	Cyber 205 Time 2	Cray X-MP Time 1	Cray X-MP Time 2	Amdahl VP1200 Time 1	Amdahl VP1200 Time 2
			Natural ordering					
JCG	268	4536	0·208	0·224	0·100	0·104	0·050†	0·056†
JSI	698	2000	0·403	0·414	0·196	0·199	0·135	0·137
SOR	380	2000	1·432	1·458	1·216	1·228	0·969	0·976
SSORCG	114	6228	0·842	0·869	0·761	0·773	0·686†	0·693†
SSORSI	270	5000	1·971	1·997	1·775	1·787	1·618	1·625
			Red-black ordering					
JCG	268	4536	0·208	0·230	0·100	0·113	0·050†	0·062†
JSI	698	2000	0·400	0·422	0·198	0·210	0·130	0·143
SOR	393	2000	0·307	0·329	0·101	0·114	0·073	0·086
SSORCG	131	6262	0·164	0·186	0·095	0·107	0·056†	0·068†
SSORSI	354	5000	0·353	0·375	0·201	0·213	0·139	0·151
RSCG	134	2306	0·088	0·110	0·042	0·055	0·026	0·038
RSSI	369	1346	0·175	0·197	0·087	0·100	0·071	0·083

† Estimated value.

Table VI. ITPACKV on problem 3

Method	Iterations	Workspace	Cyber 205 Time 1	Cyber 205 Time 2	Cray X-MP Time 1	Cray X-MP Time 2	Amdahl VP1200 Time 1	Amdahl VP1200 Time 2
			Natural ordering					
JCG	493	21992	2·418	2·470	1·038	1·051	0·876	0·889
JSI	1220	10010	2·894	2·947	1·457	1·469	0·867	0·879
SOR	687	10010	12·758	12·891	10·845	10·904	9·723	9·756
SSORCG	123	30522	4·633	4·767	4·155	4·214	3·761	3·794
SSORSI	355	25025	12·797	12·930	11·630	11·689	10·505	10·538
			Red-black ordering					
JCG	493	21992	2·404	2·513	1·045	1·107	0·856	0·918
JSI	1220	10010	2·859	2·966	1·465	1·528	0·814	0·875
SOR	726	10010	2·560	2·667	0·777	0·839	0·466	0·528
SSORCG	237	30978	1·483	1·590	0·839	0·902	0·522	0·584
SSORSI	584	25025	2·558	2·665	1·441	1·504	0·809	0·870
RSCG	244	10322	0·797	0·905	0·352	0·414	0·287	0·348
RSSI	604	6452	1·092	1·202	0·547	0·610	0·355	0·417

Table VII. ITPACKV on problem 4

Method	Iterations	Workspace	Cyber 205 Time 1	Cyber 205 Time 2	Cray X-MP Time 1	Cray X-MP Time 2	Amdahl VP1200 Time 1	Amdahl VP1200 Time 2
			Natural ordering					
JCG	255	5436	2·175	2·187	0·668	0·670	0·880	0·883
JSI	309	2208	0·211	0·222	0·095	0·098	0·066	0·069
SOR	174	2208	0·726	0·757	0·616	0·629	0·503	0·509
SSORCG	63	6876	0·878	0·907	0·579	0·592	0·588	0·596
SSORSI	98	5520	0·811	0·842	0·727	0·740	0·657	0·664
			Red-black ordering					
JCG	255	5436	2·175	2·200	0·669	0·683	0·877	0·891
JSI	309	2208	0·209	0·232	0·096	0·110	0·065	0·078
SOR	174	2208	0·149	0·174	0·048	0·061	0·035	0·049
SSORCG	83	6956	0·128	0·151	0·073	0·086	0·053	0·067
SSORSI	166	5520	0·183	0·207	0·102	0·115	0·070	0·085
RSCG	95	2303	0·070	0·095	0·033	0·046	0·029	0·043
RSSI	147	1377	0·075	0·100	0·036	0·050	0·029	0·043

For ITPACKV, examining Tables V–VII indicate that RSCG and RSSI were the best routines for all problems on all three computers. Since many methods in this package vectorize, the timings are not so closely related to the number of operations done. A vectorized routine such as JSI may do better than a non-vectorized routine such as SSORCG, in spite of the greater number of iterations. It was also noticed that JSI does much better than JCG for problem 4 on all three computers. This results from the time required for the eigenvalue estimate in the conjugate gradient routines when applied to non-symmetric problems.

Recently, similar studies of this type have been carried out by Mamun, Pope and Sepehrnoori[10] and by Rood.[11]

In this paper, we are comparing different versions of the ITPACK software package and have indicated what programming changes needed to be made to take advantage of the speed available in three different vector computers. We did not attempt to directly compare these vector computers, since their performance can vary depending on careful recoding using optimization techniques that take advantage of the problem structure and machine architecture.

ACKNOWLEDGEMENTS

We wish to thank Robert E. Lynch for helpful discussions on iterative methods and his suggestions on increasing the efficiency of our software contained in Reference 12. As always, David M. Young has been most generous with help and guidance throughout the development of ITPACK 2C and ITPACKV 2C. We wish to thank Ray Lester and Chani Pangali of the Amdahl Corp. for arranging for the runs on the VP1200.

REFERENCES

1. R. P. Kendall and A. H. Sherman, Letter and test matrices entitled 'Linear algebra for reservoir simulation—comparison study of numerical algorithms', J. S. Nolen & Associates, Inc., 16225 Park 10 Place, Suite 560, Houston, TX. 77084 (1984).
2. A. Datta Gupta, B. Mossberg, G. Pope and K. Sepehrnoori, 'Application of vector processors to chemical enhanced oil recovery simulation', *Commun. appl. numer. methods*, **2**, 297–303 (1986).
3. D. R. Kincaid, J. R. Respess, D. M. Young, and R. G. Grimes, 'ITPACK 2C—a Fortran package for solving large sparse linear systems by adaptive accelerated iterative methods', *ACM Trans. Math. Software*, **8**(3), 302–322 (1982).
4. D. R. Kincaid, T. C. Oppe, J. R. Respess and D. M. Young, 'ITPACKV 2C User's Guide', *Report CNA-191*, Center for Numerical Analysis, Univ. Texas, Austin, TX (1984).
5. S. C. Eisenstat, M. C. Gursky, H. H. Schultz and A. H. Sherman, 'Yale sparse matrix package II, the nonsymmetric codes', *Report 114*, Dept. Computer Science, Yale Univ., New Haven, CT (1977).
6. C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T. Krogh, 'Basic linear algebra subprograms for Fortran usage', *ACM Trans. Math. Software*, **5**(3), 308–323 (1979).
7. J. R. Rice and R. F. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, NY, 1985.
8. D. R. Kincaid, T. C. Oppe and D. M. Young, 'Adapting ITPACK routines for use on a vector computer', *Report CNA-177*, Center for Numerical Analysis, Univ. Texas, Austin, TX (1982).
9. J. J. Dongarra and S. C. Eisenstat, 'Squeezing the most out of an algorithm in CRAY FORTRAN', *ACM Trans. Math. Software*, **10**(3), 219–230 (1984).
10. C. K. Mamun, G. A. Pope and K. Sepehrnoori, 'A comparison of iterative methods for reservoir simulation problems on vector processors', *Report No. 86–2*, Center for Enhanced Oil and Gas Recovery Research, Univ. Texas, Austin, TX (1986).
11. J. D. Rood, 'A comparison of iterative methods for reservoir simulation problems on vector computers, *M.S. thesis*, Univ. Texas, Austin, TX (1986).
12. R. E. Lynch 'Increasing the efficiency of vector ITPACK', unpublished manuscript (1984).
13. CDC Cyber 200 Fortran Version 2 Reference Manual, *Publication No. 60485000*, Control Data Corp., St. Paul, MN, 1983.
14. Cray X-MP Series, Model 48, Mainframe Reference Manual, *Publication No. HR-0097*, Cray Research, Inc., Mendota Heights, MN, 1984.
15. L. A. Hageman and D. M. Young, *Applied Iterative Methods*, Academic Press, New York, 1981.
16. D. R. Kincaid and D. M. Young, 'Survey of iterative methods', in *Encyclopedia of Computer Sciences and Technology*, Vol. 13 (Eds. J. Belzer, A. Holzman and A. Kent), Marcel Dekker, New York, pp. 354–391, 1979.
17. D. R. Kincaid and T. C. Oppe, 'ITPACK on supercomputers', in *Numerical Methods* (Eds. A. Dold and B. Echmann), *Lecture Notes in Mathematics 1005*, Springer-Verlag, New York, NY, pp. 151–161, 1983.
18. D. R. Kincaid, T. C. Oppe and D. M. Young, 'Vector computations for sparse linear systems', *SIAM J. Algebraic Discrete Methods*, **7**(1), 99–112 (1986).
19. P. J. Sylow 'Optimization guide', *Cray Computer Systems Technical Note, SN-0220*, Cray Research, Inc., Mendota Heights, MN (1983).
20. D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.
21. D. M. Young, T. C. Oppe, D. R. Kincaid and L. J. Hayes, 'On the use of vector computers for solving large sparse linear systems', *Report CNA-199*, Center for Numerical Analysis, Univ. Texas, Austin, TX (1985).