ORTHOGONAL s-STEP METHODS FOR NONSYMMETRIC LINEAR SYSTEMS OF EQUATIONS

Charles D. Swanson
Cray Research, Inc., 655F Lone Oak Drive, Eagan, MN 55121

Anthony T. Chronopoulos
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455

Abstract. Conjugate Gradient-like methods such as Orthomin(k) have been developed to obtain a good numerical approximation to the solution of $Ax = f$ when the matrix A is large, sparse, and nonsymmetric. In s-step variations of these iterative methods, s consecutive steps of the one-step methods are performed simultaneously. The number of inner products required is reduced and the resulting algorithms are more suitable for parallel computations. However, lack of orthogonality between the s direction vectors at each iteration leads to instability unless s is small ($s \leq 5$). In this project, the effect of orthogonalizing the s direction vectors at each iteration was studied. The A^TA-Orthogonal s-Step Orthomin(k) and p-Orthogonal s-Step Orthomin(k) algorithms were developed and shown to be stable for large values of s (up to s=20). The performance of these algorithms on a multiple processor CRAY Y-MP8 computer was analyzed.

1. Introduction.

If A is a nonsingular matrix of order n, an approximation to the solution of the linear system $Ax = f$ can be obtained by the conjugate gradient (CG) method if A is symmetric and positive definite. An approximate solution x_i which minimizes the error functional

$$E(x_i) = (x - x_i)^T A(x - x_i)$$

is obtained at each iteration. The conjugate residual (CR) method is similar to CG but minimizes the residual

$$E(x_i) = \|f - Ax_i\|_2^2.$$

In [1], Chronopoulos surveys variations of the CR method and develops s-step versions of the Generalized Conjugate Residual (GCR) and Orthomin(k) methods for nonsymmetric matrices with symmetric part $M = (A + A^T)/2$ positive definite or indefinite. In s-step methods, s consecutive steps of the one-step methods are performed simultaneously.

The generalized CR method applied to nonsymmetric matrices minimizes $\|r_{i+1}\|_2^2$ along the direction p_i in order to determine the steplength in $x_{i+1} = x_i + a_i p_i$. p_i is made A^TA-orthogonal to p_{i-1} but is not automatically A^TA-orthogonal to all of the previous p_j's as in the symmetric case; orthogonalization must be explicitly implemented. Positive definiteness assures that $a_i = (r_i,Ap_i)/(Ap_i,Ap_i)$ is positive so the solution is improved at every step. Orthogonality and the norm reducing property of CR guarantee convergence in at most n iterations in the symmetric and positive definite case.

(i) **Orthomin(k).** For the GCR method, p_i is A^TA-orthogonalized to all of the proceeding directions. In Orthomin(k), orthogonalization is to only the preceding k directions:

Algorithm 1.1. Orthomin(k)

```
1.    $x_0$, $p_0 = r_0 = f - Ax_0$
For $i = 0$ Until Convergence Do
2.    $a_i = \dfrac{(r_i, Ap_i)}{(Ap_i, Ap_i)}$
3.    $x_{i+1} = x_i + a_i p_i$
4.    $r_{i+1} = r_i - a_i Ap_i$
5.    $b_j^i = \dfrac{-(Ar_{i+1}, Ap_j)}{(Ap_j, Ap_j)}$
6.    $p_{i+1} = r_{i+1} + \sum_{j=i-k+1}^{i} b_j^i p_j$
7.    Compute $Ap_{i+1}$
EndFor.
```

(ii) s-step Orthomin(k). In the s-step Orthomin(k) method, the s directions $\{r_i,\dots,A^{s-1}r_i\}$ are formed and A^TA-orthogonalized to k of the preceding directions $\{p_j^1,\dots,p_j^s\}$, $j=i-k+1,i$ The norm of the residual $\|r_{i+1}\|$ is minimized simultaneously in all s new directions in order to obtain x_{i+1}. Following the detailed discussion in Section 4 of [1], let

$W_i = [(Ap_i^j, Ap_i^l)]$, where $1 \le j,l \le s$

$\underline{a}_i = [a_i^1,\cdots,a_i^s]$ (the steplengths in updating x_i)

$\underline{m}_i = [(r_i,Ap_i^1),\cdots,(r_j,Ap_i^s)]^T$

$\underline{c}_j^l = [(A^l r_{i+1},Ap_j^1),\cdots,(A^l r_{i+1},Ap_j^s)]^T$

$\underline{b}_j^l = \{b_j^{(l,m)}\}_{m=1}^s$ for $j = i-k+1,\cdots,i$ and $l = 1,\cdots,s$

(the coefficients to A^TA – orthogonalize to the previous directions)

$P_i = [p_i^1,\cdots,p_i^s]$ (the direction vectors)

$R_i = [r_i,Ar_i,\cdots,A^{s-1}r_i]$ (the residuals)

Then a description of the s-step Orthomin(k) method can be given as follows:

Algorithm 1.2. s - step Orthomin(k)

```
1.    $x_0,\ P = \left[r_0 = f - Ax_0, Ar_0,\cdots,A^{s-1}r_0\right]$
For $i = 0$ Until Convergence Do
2.    Compute $\underline{m}_i, W_i$
3.    Call Scalar1
4.    $x_{i+1} = x_i + P_i\underline{a}_i$
5.    $r_{i+1} = r_i - AP_i\underline{a}_i$
6.    Compute $\underline{c}_j^i,\quad j = i-k+1,\cdots,i$
7.    Call Scalar2
8.    Compute $R_i = \left[r_i,Ar_i,\cdots,A^{s-1}r_i\right]$
9.    $P_{i+l} = R_{i+l} + \sum_{j=i-k+l}^{i} P_j\left[\underline{b}_j^l\right]_{l=l}^s$
10.   Compute $AP_{i+1}$ or,
11.   $AP_{i+1} = AR_{i+1} + \sum_{j=i-k+1}^{i} AP_j\left[\underline{b}_j^l\right]_{l=1}^s$
EndFor.
```

Scalar1: Decomposes $W_i = [(Ap_i^j,Ap_i^l)]$, where $1 \le j,l \le s$.

It then solves $W_i\underline{a}_i = \underline{m}_i$, where $\underline{a}_i = [a_i^1,\cdots,a_i^s]^T$ and

$\underline{m}_i = [(r_i,Ap_i^1),\cdots,(r_j,Ap_i^s)]^T$.

Scalar2: Solves $W_j\underline{b}_j^l = -\underline{c}_j^l$ for $j = i-k+1,\cdots,i$ and

$l = 1,\cdots,s$ where $\underline{c}_j^l = [(A^l r_{i+1},Ap_j^1),\cdots,(A^l r_{i+1},Ap_j^s)]^T$

and $\underline{b}_j^l = \{b_j^{(l,m)}\}_{m=1}^s$ for $j = 1 - k + 1,\cdots,i$ and $l = 1,\cdots,s$.

The solution of the Scalar1 and Scalar2 linear systems may cause a quick loss of orthogonality of the s-dimensional direction subspaces P_i. Non-orthogonality of the s vectors can lead to instability in the algorithm as s gets larger than 5 [1,3]. The purpose of this project is to determine if orthogonalizing the s direction vectors within each subspace P_i can lead to a stable algorithm for larger values of s thereby reducing the number of iterations required. The increased work in each iteration can be executed in parallel.

(iii) Numerical Experiments. The numerical experiments carried out utilized two test problems. The first problem was derived from a five point discretization of the partial differential equation

$$-(bu_x)_x - (cu_x)_x + du_x + (du_x)_x + eu_y + (eu_y)_y + fu = g$$

on the unit square, where

$$b(x,y) = e^{-xy},\quad c(x,y) = e^{xy},\quad d(x,y) = b(x+y)$$

$$e(x,y) = (x+y) \text{ and } f(x,y) = 1./(1+x+y)$$

subject to Dirichlet boundary conditions $u=0$. The right hand side g was chosen so that the solution was known to be $e^{xy}sin(\pi x)sin(\pi x)$. The parameters were chosen to produce a nonsymmetric matrix. This problem will be referred to as the "PDE problem."

The second test problem is from Walker [4] and is used to evaluate the numerical stability of the various algorithms.

The matrix A is

$$A = \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & 0 & \alpha \\ 0 & 2 & \cdot & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot & n-1 & 0 \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & n \end{bmatrix}$$

while the right hand side $b = (1,...,1)^T$. The problem was solved for $n = 100$ with $a = 2 \times 10^6$. This problem will be referred to as the "ill-conditioned problem." The matrix condition number (in the infinity norm) is greater or equal to α^2.

2. A^TA-Orthogonal s-Step Orthomin(k). In this method the direction vectors within each subspace P_i are A^TA-orthogonalized using the Modified Gram-Schmidt method. The Scalar1 and Scalar2 linear systems in Algorithm 1.2 need not be solved in the A^TA-Orthogonal s-Step Orthomin(k) method since the W_i matrix is the identity matrix if P_i is perfectly A^TA-orthogonalized. The algorithm is as follows:

Algorithm 2.1. A^TA - Orthogonal s - step Orthomin(k)

```
1.      $x_0, P = [r_0 = f - Ax_0, Ar_0, \cdots, A^{s-1}r_0]$
For i = 0 Until Convergence Do
    2.      Compute $\underline{a}_i$
    3.      $x_{i+1} = x_i + P_i\underline{a}_i$
    4.      $r_{i+1} = r_i - AP_i\underline{a}_i$
    5.      Compute $\underline{b}_j^i$
    6.      Compute $R_i = [r_i, Ar_i, \cdots, A^{s-1}r_i]$
    7.      $P_{i+1} = R_{i+1} + \sum_{j=i-k+1}^{i} P_j[\underline{b}_j^i]_{l=1}^s$
    8.      Compute $AP_{i+1}$ or,
    9.      $AP_{i+1} = AR_{i+1} + \sum_{j=i-k+1}^{i} AP_j[\underline{b}_j^i]_{l=1}^s$
    10.     $A^TA$ – orthogonalize $AP_{i+1}$ using the Modified
            Gram - Schmidt method to obtain $\underline{AP}_{i+1}$ and $\underline{P}_{i+1}$
EndFor.
```

Here $\underline{AP}_{i+1}$ is orthogonalized.

Results for the PDE problem are shown in Table 2.1 for the s-Step method and the A^TA-Orthogonal s-Step method. The initial values x(i)=0.05*mod(i,50) were used. The stopping criterion was $\|r_i\|_2 \leq 10^{-6}$ with the maximun number of iterations allowed set to 700. The number of iterations to convergence, maximum error, and CPU time are listed. The maximum error is the maximum value of abs[x(i)-sol(i)] where x(i) is the calculated value and sol(i) is the known solution. The s=1 case is standard Orthomin(k). For k=1 and s>1, the maximum error for the s-Step method shows a dramatic increase at s=12 while the Orthogonal s-step method maintains the initial maximum error through s=16. At s=20 the error increases very quickly. For k=2 and k=4, the s-Step method fails at s=12. The Orthogonal method is stable through s=16 for k=2 and again shows an increase in the maximum error at s=20. For k=4, the Orthogonal method shows an increase in the number of iterations at s=16 and an increase in the maximum error at s=20.

To test the effect of removing the $s \times s$ linear system solvers, a set of runs was made with A^TA-orthogonalization of P_i, but with the linear systems in place. The results in Table 2.1 show minimal effect for the PDE problem.

Results for the ill-conditioned problem which severely tests the numerical stability of the algorithms are shown in Table 2.2. The advantage of the Orthogonal s-Step method versus the s-Step method is apparent at s=8, k=1 where the iteration count and maximum error for the s-Step method begin to increase. The Orthogonal method remains stable through s=12. For tests with k=4, the iteration count is twice as high for the s-Step method at s=4.

Again, runs with the linear system solvers in place were made for the ill-conditioned problem. There were some improvements at s=16 for both k=1 and k=4 but the maximum error was still high (Table 2.2).

To test the effect of the stopping criterion, the parameter "eps" was tightened from 10^{-10} to 10^{-12}. The results are shown in Table 2.3. At s=8, the maximum error for the Orthogonal method has improved by a factor of 65 but at s=12 it is roughly the same for the two cases.

3. p-Orthogonal s-Step Orthomin(k). An alternative to A^TA-orthogonalizing the direction vectors is to simply orthogonalize the direction vectors within the subspace P_i directly. This is simple to implement, but does not diagonalize the matrices in the linear systems of Algorithm 1.2 so they still have to be solved. This is called the p-Orthogonal s-step Orthomin(k) algorithm:

Algorithm 3.1. p - Orthogonal s - step Orthomin(k)

```
1.   $x_0, P = [r_0 = f - Ax_0, Ar_0, \cdots, A^{s-1}r_0]$
For $i$ = 0 Until Convergence Do
2.       Compute $\underline{m}_i, W_i$
3.       Call Scalar1
4.       $x_{i+1} = x_i + P_i\underline{a}_i$
5.       $r_{i+1} = r_i - AP_i\underline{a}_i$
6.       Compute $\underline{c}^i_j, \quad j = i-k+1, \cdots, i$
7.       Call Scalar2
8.       Compute $R_i = [r_i, Ar_i, \cdots, A^{s-1}r_i]$
9.       $P_{i+1} = R_{i+1} + \sum_{j=i-k+1}^{i} P_j[\underline{b}^l_j]^s_{l=1}$
10.      Orthogonalize $P_{i+1}$ using a Modified
         Gram - Schmidt or Householder
         method to obtain $\underline{P}_{i+1}$
11.      Compute $A\underline{P}_{i+1}$
EndFor.
```

Orthogonalization of the direction vectors was done with the Modified Gram-Schmidt method and also with the Householder technique. The Modified Gram-Schmidt is generally more efficient but the Householder technique is more accurate for ill-conditioned matrices [5, p. 219]. However, the Householder method is also more expensive.

Table 3.1 shows results for the PDE problem using s-Step Orthomin(k) and p-Orthogonal s-Step Orthomin(k) with Modified Gram-Schmidt and Householder orthogonalization. The s-Step algorithm begins to show instability at s=12 (for k=1). The Orthogonal algorithm is stable through s=20, with the maximum error having the same order of magnitude throughout for both Modified Gram-Schmidt and Householder orthogonalization.

For k>1, the p-Orthogonal s-Step method shows a higher iteration count than the s-Step method at s=4, especially for k=4 . The direction vectors in P_i which are A^TA-orthogonalized to the k previous dirction vectors in the other orthomin(k) methods discussed have been p-orthogonalized here. This lack of A^TA-orthogonalization to the previous k vectors means that the p-Orthogonal s-Step Orthomin(k) algorithm is not effective for k>1 (see ill-conditioned problem results at k=2 in Table 3.2).

Table 3.2 compares both orthogonalization methods applied to the ill-conditioned problem with n=100 and $\alpha = 2 \times 10^6$ and eps $=1 \times 10^{-10}$. For k=1 both orthogonal methods are stable through s=20 (the s-Step method failed at s=8). However, for large values of s (16 and 20) the Householder orthogonalization produced maximum errors an order of magnitude smaller than the Modified Gram-Schmidt method which can be expected.

For k=4, the p-Orthogonal method shows a much higher iteration count at k=2, due to the lack of A^TA-orthogonalization to the previous k vectors.

4. Performance. Numerical experiments were executed on a CRAY Y-MP8 at Cray Research, Inc. in Eagan, MN. This computer has eight processors, a 6 nanosecond clock period, and 128 million 64-bit words of shared memory. Performance analysis tools available with the UNICOS 6.0 operating system were used.

(i) Single Processor Performance.

The PERFTRACE performance tool uses the hardware performance monitors on the CRAY Y-MP to determine the amount of vectorization in the code and measure the performance in Megaflops for each routine. The statistics in Table 4.1 for the A^TA-Orthogonal s-Step Orthomin(k) code indicate that it is highly vectorized, since the peak performance on a single CRAY Y-MP processor is 330 Megaflops.

The CPU seconds for the PDE problem with N=130 (where N^2 is the order of matrix A) are included in Tables 2.1 and 3.1. The standard Orthomin(k) method is obtained by setting s=1 in the s-Step code. The best single processor timing for Orthomin(k) was obtained for k=2. The best s-Step Orthomin(k) timing was obtained with s=2 and k=1. For both Orthogonal s-Step algorithms the best single processor timings were also obtained with s=2 and k=1. These four codes were then executed for problems with N ranging from 32 to 256 with the results shown in Table 4.2.

The Orthogonal methods take more time than the s-Step method because of the added work to orthogonalize the P_i direction vectors. Since the linear systems are not solved in the A^TA-Orthogonal method, it is faster than the p-Orthogonal method.

(ii) Multiple Processor Performance

The codes were executed in parallel on the CRAY Y-MP using the Autotasking feature of the Fortran compiling system which provides automatic multitasking over the eight processors. The parallel processing performance was studied and measured using the "Atexpert" tool, an expert system that predicts the expected performance on a dedicated system from runs made on a nondedicated system. The programmer can use the information provided by Atexpert along with compiler directives explicitly inserted in the code to iteratively enhance the parallel performance of the program.

For the A^TA-Orthogonal method, Atexpert showed that the initial version of the program was 60% parallel. Amdahl's Law gives a maximum speedup of 1.4 on an 8 processor system for this level of parallelism. The Atexpert tool helps the programmer identify the parallel and serial regions in the program. Once the troublesome serial regions have been identified, compiler directives can then be inserted to provide enhanced parallel execution. A simple example is to tell the compiler to execute an inner loop on multiple processors, when ordinarily it would not. Since the inner loops for the largest problem are 65,536 elements long (N=256) this was effective. Directives can also be inserted to tell the compiler when potential dependencies inhibiting parallel code generation can safely be ignored. Using these techniques, the serial regions of the A^TA-Orthogonal code were attacked in order to provide a highly parallel program.

The value of orthogonalizing the P_i direction vectors is that it allows s to become larger. This means the number of iterations is reduced with more work being done in each iteration, a situation that should make more efficient use of multiple processors. The additional work from the orthogonalization should be compensated by the enhanced parallel performance to obtain a faster algorithm.

Multiple processor runs were made on a dedicated CRAY Y-MP8 for the Orthomin(k), s-step Orthomin(k), and A^TA-Orthogonal s-step Orthomin(k) algorithms, using the large N=256 PDE problem. The results are given in Table 4.3 and illustrated in Figure 4.1. Only s=2 and s=4 results are shown for the s-Step method since the numerical accuracy tests with this method show loss of accuracy at s=8 (Table 3.2). For the A^TA-Orthogonal method, s values of 2, 4, and 8 are shown (k=1). The best absolute performance for this problem was achieved with the A^TA-Orthogonal 4-step Orthomin(1) method using eight processors. However, the speed-up factor of 3.76 on an eight processor system indicates that maximum parallel performance is not being achieved. This is due to remnants of serial code plus overhead. A significant source of overhead (identified with help from the Atexpert tool) is "load imbalance" in parallelizing the Modified Gram-Schmidt orthogonalization routine.

Using two processors, the load imbalance problem is largely alleviated and the parallel A^TA-Orthogonal method is efficient, as shown by the 1.85 speed-up. Nevertheless, for the two processor runs the elapsed time for the 2-step Orthomin(1) method was significantly better. Therefore, unless the user has a dedicated eight processor machine, 2-step Orthomin(1) is the most effective algorithm.

5. Summary. This project demonstrated that the value of s in the s-Step Orthomin(k) algorithm can be increased beyond s=5 by orthogonalizing the s direction vectors in each iteration. Two orthogonalization schemes were presented. The A^TA-Orthogonal s-Step Orthomin(k) algorithm provided good numerical accuracy through s=12 (k=1) for the ill-conditioned problem. For k>1, the A^TA-Orthogonal method was stable through s=8. The p-Orthogonal s-Step Orthomin(k) showed stability through s=20 (k=1) with the Householder orthogonalization showing significantly greater numerical accuracy than Modified Gram-Schmidt at s=16 and s=20. However, the p-Orthogonal method does not work well for k>1. For the PDE problem, both methods were stable through s=20 (k=1).

Performance measurements of the algorithms on a CRAY Y-MP8 computer showed the A^TA-Orthogonal 4-step Orthomin(1) algorithm to be fastest using all eight processors. However, the 2-step Orthomin(1) algorithm using two processors was only 7% slower. Improvements in the parallel performance of the Orthogonal s-Step algorithms followed by implementation on systems with more than eight processors (such as the sixteen processor CRAY Y-MP C90) may show the Orthogonal s-Step algorithms to be superior to the s-Step method.

Also note that in practice, preconditioning is used to improve the condition number of the coefficient matrix and speed up the convergence of the linear system.

6. Acknowledgements. This research was partially supported by supported by NSF grant CCR-8722260. The Minnesota Supercomputer Institute provided time on a CRAY-2. CRAY Y-MP time was provided by Cray

Research, Inc. We wish to thank the reviewers for comments that enhanced the quality of the presentation.

References

[1] A. T. Chronopoulos, "s-Step Iterative Methods for (Non)symmetric (In)definite Linear Systems," *SIAM J. of Numerical Analysis*, Vol. 28, No. 6, Dec. 1991.

[2] A. T. Chronopoulos and C. W. Gear, "s-Step Iterative Methods for Symmetric Linear Systems," *J. of Computational and Applied Math.*, Vol. 25, 1989, p. 153-168.

[3] Anthony T. Chronopoulos, "s-Step Orthomin and GMRES Implemented on Parallel Computers," *Tech. Rep. CSci No. TR 90-15, U. of Minnesota, February 1990.*

[4] H. Walker, "Implementation of the GMRES Method Using Householder Transformations," *SIAM J. Sci. Stat. Comp.*, 9 (1988), pp. 152-163.

[5] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, second edition, Johns Hopkins, 1989.

Table 2.1. A^TA-Orthogonal s-Step Orthomin(k) Compared to s-Step Ortomin(k) for PDE Problem.

		s-Step Orthomin			A^TA-Orthogonal s-Step Orthomin Modified Gram-Schmidt No Linear Systems			A^TA-Orthogonal s-Step Orthomin Modified Gram-Schmidt With Linear Systems		
s	k	iterations	max error	CPU time	iterations	max error	CPU time	iterations	max error	CPU time
1	1	471	0.0003959	0.942	-	-	-	-	-	-
2	1	173	0.000558	0.824	173	0.0005563	1.116	173	0.0005561	1.187
4	1	88	0.0003412	1.129	88	0.0003414	1.485	88	0.0003414	1.6
8	1	63	0.0004254	2.444	63	0.0003623	3.167	64	0.0004789	3.532
12	1	48	0.0191	3.738	44	0.0003969	4.426	44	0.0003975	4.884
16	1	49	0.0568	6.357	31	0.0003831	5.164	30	0.0002646	5.56
20	1	55	0.0952	10.843	28	0.005214	6.924	26	0.00552	7.219
1	2	368	0.0003892	0.874	-	-	-	-	-	-
2	2	179	0.0003959	1.104	179	0.0003957	1.411	179	0.0003959	1.476
4	2	108	0.0002909	1.976	108	0.000373	2.436	109	0.0002356	2.583
8	2	66	0.0002608	3.97	78	0.0004748	5.68	72	0.0005102	5.561
12	2	69	0.07796	8.76	51	0.0004469	7.639	49	0.0003899	7.835
16	2	***	***	***	37	0.000329	9.382	36	0.0002384	9.69
20	2	***	***	***	29	0.01133	11.146	30	0.000411	12.183
1	4	353	0.000404	1.091	-	-	-	-	-	-
2	4	213	0.0004432	1.91	213	0.0004332	2.345	213	0.0004253	2.358
4	4	125	0.000379	3.708	129	0.000281	4.419	130	0.0003655	4.467
8	4	148	0.000283	15.519	78	0.0002489	9.15	80	0.0003099	9.557
12	4	***	***	***	54	0.000275	13.131	52	0.0003671	13.227
16	4	***	***	***	60	0.0001806	25.198	60	0.0001419	26.516
20	4	***	***	***	54	0.0193	34.575	55	0.0252	37.447

CPU time = 1 processor CPU time in seconds   eps = 1.0E-6   N=130
s=1 is standard Orthomin(k)   *** = problem failed

Table 2.2. A^TA-Orthogonal s-Step Orthomin Compared to s-Step Orthomin for Ill-Conditioned Problem (eps=1.0E-10).

		s-Step Orthomin		A^TA-Orthogonal s-Step Orthomin Modified Gram-Schmidt No Linear Systems		A^TA-Orthogonal s-Step Orthomin Modified Gram-Schmidt With Linear Systems	
s	k	iterations	max error	iterations	max error	iterations	max error
1	1	701*	18811	-	-	-	-
2	1	44	0.0001447	44	0.0003767	44	0.0003768
4	1	29	0.0006337	23	0.0001111	23	0.0001111
8	1	21	0.00207	10	0.0004373	10	0.000437
12	1	22	0.0175	7	0.0003538	7	0.0003266
16	1	30	41.267	13	83.833	6	0.4465
1	4	701*	14556	-	-	-	-
2	4	38	0.0001134	38	0.0001409	38	0.0001404
4	4	37	0.0006885	18	0.0004195	18	0.0004215
8	4	94	305.87	28	0.00002342	28	0.0001475
12	4	102	5.036	41	0.001952	36	0.0006365
16	4	***	***	76	205.99	46	3.99

NOTE: alpha = 2.0E+6
s=1 is standard Orthomin(k)
* = problem reached iteration count limit
*** = problem failed

Table 2.3. A^TA-Orthogonal s-Step Orthomin Compared to s-Step Orthomin for Ill-Conditioned Problem (eps=1.0E-12).

		s-Step Orthomin		A^TA-Orthogonal s-Step Orthomin Modified Gram-Schmidt No Linear Systems	
s	k	iterations	max error	iterations	max error
1	1	701*	18811	-	-
2	1	52	0.00001061	52	8.505E-07
4	1	34	0.00001566	28	0.000008132
8	1	27	0.001034	13	0.000006738
12	1	35	0.01799	9	0.0002057
16	1	35	41.27	16	83.93
1	4	701*	14556	-	-
2	4	47	0.000009564	47	0.000008685
4	4	56	0.000006166	23	9.448E-07
8	4	121	305.87	50	0.000004859
12	4	***	***	59	0.001941
16	4	***	***	95	205.9

NOTE: alpha = 2.0E+6
s=1 is standard Orthomin(k)
* = problem reached iteration count limit
*** = problem failed

Table 3.1. p-Orthogonal s-Step Orthomin Compared to s-Step Orthomin for PDE Problem.

		s-Step Orthomin			p-Orthogonal s-Step Orthomin Modified Gram-Schmidt			p-Orthogonal s-Step Orthomin Householder		
s	k	iterations	max error	CPU time	iterations	max error	CPU time	iterations	max error	CPU time
1	1	471	0.0003959	0.942	-	-	-	-	-	-
2	1	173	0.000558	0.824	173	0.0006081	1.278	173	0.0006058	1.791
4	1	88	0.0003412	1.129	88	0.0003415	1.662	88	0.0003419	2.368
8	1	63	0.0004254	2.444	61	0.0004992	3.374	63	0.0004001	4.972
12	1	48	0.0191	3.738	44	0.0004287	4.788	43	0.0003613	6.723
16	1	49	0.0568	6.357	32	0.0004153	5.733	33	0.0003282	8.525
20	1	55	0.0952	10.843	26	0.0002288	6.88	26	0.0003472	10.003
1	2	368	0.0003892	0.874	-	-	-			
2	2	179	0.0003959	1.104	179	0.0003958	1.547	179	0.0003959	2.058
4	2	108	0.0002909	1.976	118	0.0004998	2.813	110	0.0003456	3.48
8	2	66	0.0002608	3.97	71	0.0005862	5.285	78	0.000562	7.632
12	2	69	0.07796	8.76	48	0.0003256	7.256	57	0.0003229	11.307
16	2	***	***	***	37	0.0002498	9.4	57	0.0005329	19.053
1	4	353	0.000404	1.091	-	-	-	-	-	-
2	4	213	0.0004432	1.91	213	0.0004253	2.334	213	0.0004233	2.971
4	4	125	0.000379	3.708	295	0.0002559	9.887	157	0.0004863	6.608
8	4	148	0.000283	15.519	130	0.0003292	14.527	166	0.0002661	22.532
12	4	***	***	***	92	0.0003097	21.627	125	0.0003779	35.959

s=1 is standard Orthomin(k) eps = 1.0E-6 N=130

CPU time = 1 processor CPU time in seconds *** = problem failed

Table 3.2. p-Orthogonal s-Step Orthomin Compared to s-Step Orthomin for Ill-Conditioned Matrix (eps=1.0E-10).

		s-Step Orthomin		p-Orthogonal s-Step Orthomin Modified Gram-Schmidt		p-Orthogonal s-Step Orthomin Householder	
s	k	iterations	max error	iterations	max error	iterations	max error
1	1	701*	18811	-	-	-	-
2	1	44	0.0001447	44	0.0003902	44	0.00040967
4	1	29	0.0006337	23	0.0001125	23	0.00009717
8	1	21	0.00207	11	0.00008743	10	0.001257
12	1	22	0.0175	7	0.000002226	7	0.00001121
16	1	30	41.267	6	0.0003545	6	0.00002671
20	1	45	471.2	5	0.0001649	6	0.00003692
1	4	701*	14556	-	-	-	-
2	4	38	0.0001134	128	0.0009316	138	0.001331
4	4	37	0.0006885	84	0.0002993	87	0.000103

NOTE: alpha = 2.0E+6 s=1 is standard Orthomin(k)

* = problem reached iteration count limit

Table 4.1. Vector Performance of the A^TA-Orthogonal s-Step Code Using the PERFTRACE Performance Analysis Tool

```
                    Perfview Statistics Report
                      Showing Traced Routines
                 (Sorted by MegaFlops (Descending))
                  (CPU Times are Shown in Seconds)


Name      Called   Time     Avg Tim   EX %  ACM % Mmems Mflops
-------- ------- --------- --------- ----- ----- ----- -----
SORTHOMIN      1 2.96E+00 2.96E+00  59.9  59.9 344.3 249.7   **************
MATVEC       706 4.52E-01 6.40E-04   9.1  69.0 290.6 237.7   **
MGS           87 1.50E+00 1.73E-02  30.4  99.4 313.2 195.7   *******
MESH           1 2.44E-02 2.44E-02   0.5  99.9  48.9 165.4
$MAIN          1 3.58E-03 3.58E-03   0.1 100.0  40.5  66.8
===========================================================
Totals       796 4.94E+00          100.0 100.0 328.3 231.6
```

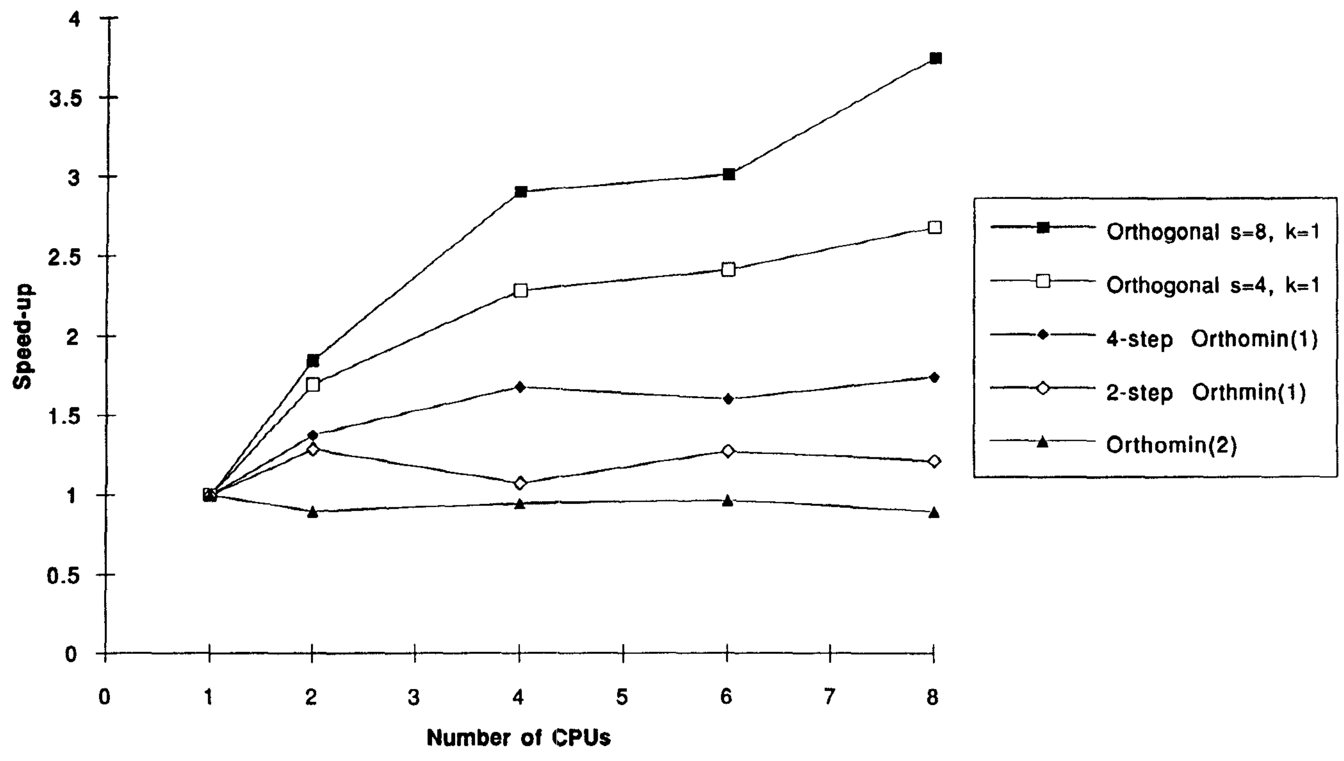
Table 4.2. Single Processor Performance.

N (size = N*N)	32		64		128		192		256	
	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter
Orthomin(2)	0.022	95	0.107	166	0.885	383	3.304	663	7.748	881
2-step Orthomin(1)	0.026	61	0.123	98	0.763	167	2.571	252	6.134	341
A^TA-Orth. 2-step Orthomin(1)	0.029	61	0.161	98	1.048	167	3.558	252	8.39	340
A^TA-Orth. 8-step Orthomin(1)	0.076	22	0.462	37	2.99	61	9.822	89	20.53	105
p-Orth. 2-step Orthomin(1)	0.037	61	0.188	98	1.205	168	3.992	251	9.634	342
p-Orth. 8-step Orthomin(1)	0.093	22	0.522	37	3.327	62	10.81	90	23.47	110
	Time = 1 processor CPU time in seconds									

Table 4.3. Parallel Computation Speed-up on a CRAY Y-MP8.

	Time 1 proc.	Time 2 proc.	Speed-up	Time 4 proc.	Speed-up	Time 6 proc.	Speed-up	Time 8 proc.	Speed-up
1-step Orthomin(2)	7.95	8.83	0.90	8.36	0.95	8.21	0.97	8.88	0.90
2-step Orthomin(1)	**6.28**	**4.87**	1.29	5.79	1.08	**4.92**	1.28	5.16	1.22
4-step Orthomin(1)	8.78	6.35	1.38	**5.19**	1.69	5.45	1.61	5.02	1.75
A^TA-Orth. 2-step Orthomin(1)	8.70	6.07	1.43	5.95	1.46	-	-	4.98	1.75
A^TA-Orth. 4-step Orthomin(1)	12.30	7.24	1.70	5.37	2.29	5.07	2.42	**4.56**	2.69
A^TA-Orth. 8-step Orthomin(1)	21.69	11.75	1.85	7.46	2.91	7.19	3.02	5.76	3.76
	Time = elapsed time in seconds on a dedicated machine, N=256 1-step Orthomin(2) is the standard Orthomin(2) algorithm								

Figure 4.1. Parallel Computation Speed-up on a CRAY Y-MP8.


Speed-up
Number of CPUs
Orthogonal s=8, k=1
Orthogonal s=4, k=1
4-step Orthomin(1)
2-step Orthmin(1)
Orthomin(2)