SIAM J. OPTIM.
Vol. 00, No. 0, pp. 000–000, May 0000

PARALLEL ADAPTIVE GMRES IMPLEMENTATIONS FOR HOMOTOPY METHODS*

MARIA SOSONKINA†, DONALD C. S. ALLISON†, AND LAYNE T. WATSON ‡

Abstract. The success of homotopy methods in solving large-scale optimization problems and nonlinear systems of equations depends heavily on the solution of large sparse nonsymmetric linear systems on parallel architectures. Iterative solution techniques, such as GMRES(k), favor parallel implementations. However, their straightforward parallelization usually leads to a poor parallel performance because of global communication incurred by processors. One variation of GMRES(k) considered here is to adapt the restart value k for any given problem and use Householder reflections in the orthogonalization phase to achieve high accuracy and reduce the communication overhead.

Key words. GMRES method, Krylov subspace methods, nonsymmetric linear systems, homotopy curve tracking, sparse matrix

AMS(MOS) subject classifications. 65F10, 65F50, 65H10

1. Introduction.

For numerous highly nonlinear realistic applications, probability-one homotopy methods are a primary solution choice, and are robust and accurate. Other nonlinear analysis methods, such as quasi-Newton methods, often fail whenever a good initial estimate of the solution to a given problem is hard to obtain. On the other hand, probability-one homotopy methods converge to a solution from an *arbitrary* starting point (outside of a set of measure zero). These methods have been successfully applied to solve Brouwer fixed point problems, zero finding problems, polynomial systems of equations, optimization problems, discretizations of nonlinear two-point boundary value problems based on shooting, finite differences, collocation, and finite elements [10], [19], [20]. However, the global convergence together with robustness and accuracy of homotopy methods results in a rather costly computational process.

Growing computer capacity is a major reason for homotopy algorithms becoming an affordable alternative to other methods. The benefits of parallel architectures for homotopy methods have been shown in [1]. Despite the serial nature of curve tracking, the burden of computation in homotopy methods is in the numerical linear algebra and user-defined function evaluations [7], which can be carried out in parallel. For large scale problems, the cost of the serial work may be negligible in comparison with linear system solving (and user-written function evaluation). As concluded in [2], a high degree of parallelism in the linear system solutions contributes much to the overall parallel performance of homotopy methods.

* Received by the editors ; accepted for publication (in revised form).

† Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106. The work of Sosonkina and Watson was supported in part by Department of Energy grant DE-FG05-88ER25068/A004, National Aeronautics and Space Administration grant NAG-1-1562, Air Force Office of Scientific Research grants F49620-96-1-0089 and F49620-96-1-0104, and National Science Foundation grant DMS-9625968.

‡ Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106.

In the context of homotopy methods, an efficient parallel implementation of a linear system solver has to satisfy the requirements established for a sequential linear solver—robustness and high accuracy in the solution. Under these requirements, a variation of the popular linear system solution tool GMRES [14] is considered in this paper. This variation [16] uses an adaptive strategy to deal with varying difficulties of linear systems which are to be solved by a homotopy method.

Section 2 gives a description of the adaptive GMRES(k) algorithm. Its parallel implementations using Householder reflections are discussed in Section 3. Section 4 contains conclusions.

2. GMRES(k) algorithm. The GMRES algorithm [14] is used to solve a linear system $Ax = b$ with an $n \times n$ nonsymmetric invertible coefficient matrix A. Similar to the classical conjugate gradient method, GMRES produces approximate solutions x_j which are characterized by a minimization property over the Krylov subspaces $K(j, A, r_0) \equiv \text{span}\{r_0,\ Ar_0,\ A^2r_0,\ \cdots,\ A^{(j-1)}r_0\}$, where $r_0 = b - Ax_0$ and j is the iteration number. However, unlike the conjugate gradient algorithm, the work and memory required by GMRES grow proportionately to the iteration number since GMRES needs all j vectors to construct an orthonormal basis for $K(j, A, r_0)$. This basis is often called an Arnoldi basis, since it is implemented by the Arnoldi procedure (see, e.g., [13]).

In practice, a restarted version GMRES(k) is used, where the algorithm is restarted every k iterations. GMRES(k) takes x_k as the initial guess for the next cycle of k iterations, and continues until the residual norm is small enough. The disadvantage of the restarted version is that it may stagnate and never reach the solution. The essence of the adaptive GMRES strategy proposed in [16] is to adapt the parameter k to the problem, in the same way a variable order ODE algorithm tunes the order k. With modern programming languages, which provide pointers and dynamic memory management, dealing with the variable storage requirements implied by varying k is not difficult.

If k in GMRES(k) is not sufficiently large, GMRES(k) can stagnate. A test of stagnation developed in [6] detects an insufficient residual norm reduction in the restart number k of steps by estimating the GMRES behavior on a particular linear system. Specifically, GMRES(k) is declared to have stagnated and the iteration is aborted if, at the rate of progress over the last restart cycle of steps, the residual norm tolerance cannot be met in some large multiple (bgv) of the remaining number of steps allowed ($itmax$ is a bound on the number of steps permitted). Slow progress of GMRES(k), which indicates that an increase in the restart value k may be beneficial [17], can be detected with a similar test. The near-stagnation test uses a different, smaller multiple (smv) of the remaining allowed number of steps. If near-stagnation occurs, the restart value k is incremented by some value m and the *same* restart cycle continues. Such incrementing is used, whenever needed, if the restart value k is less than some maximum value $kmax$. When the maximum value for k is reached, adaptive GMRES(k) proceeds as GMRES($kmax$). The values of the parameters smv, bgv, and m are established experimentally and can remain unchanged for most problems.

The convergence of GMRES may also be seriously affected by roundoff error, which is especially noticeable when a high accuracy solution is required. The orthogonalization phase of GMRES is susceptible to numerical instability. Let Q be a matrix whose columns are obtained by orthogonalizing the columns of a matrix M, and define the error matrix $E = Q^T Q - I$. The error matrices E_{MGS}, E_{HR} using the modified Gram-Schmidt and Householder reflection methods, respectively, to construct Q from M satisfy

$$\|E_{MGS}\|_2 \sim \mathbf{u}\,\mathrm{cond}(M), \quad \|E_{HR}\|_2 \sim \mathbf{u},$$

where $\mathbf{u}$ is the machine unit roundoff [4]. Clearly the orthogonalization with Householder reflections is more robust. An implementation of GMRES(k) using Householder reflections and its block version are given in [18]. It has been shown theoretically [18] that the implementation of GMRES using Householder reflections is about twice as expensive as when modified Gram-Schmidt is used. However, the Householder reflection method produces a more accurate orthogonalization of the Krylov subspace basis when the basis vectors are nearly linearly dependent and the modified Gram-Schmidt method fails to orthogonalize the basis vectors; this can result in fewer GMRES iterations compensating for the higher cost per iteration using Householder reflections. Let e_j be the jth standard basis vector and all norms the 2-norm. Pseudocode for an adaptive version of GMRES(k) with orthogonalization via Householder reflections implemented as in [6] and [18] is given in Fig. 1. Call this algorithm AGMRES(k).

choose $x, itmax, kmax, m$;
$r := b - Ax$; $\quad itno := 0$; $\quad cnmax := 1/(50\mathbf{u})$;
$xtol := \max\{100.0, 1.01 avnz\}\mathbf{u}$; $\quad tol := \max\{\|r\|, \|b\|\} xtol$;
while $\|r\| > tol$ **do**
begin
  $r^{old} := r$;
  determine $P_1 r = \pm\|r\| e_1$
    where the Householder transformation matrix P_1 is defined in [13];
  $k_1 = 1$; $\quad k_2 = k$;
L1: **for** $j := k_1$ **step** 1 **until** k_2 **do**
  begin
    $itno := itno + 1$;
    $v := P_j \cdots P_1 A P_1 \cdots P_j e_j$;
    determine P_{j+1} such that $P_{j+1} v$ has zero components
      after the $(j+1)$st;
    update $\|r\|$ as described in [14];
    estimate condition number $\mathrm{cond}(AV_j)$ of GMRES least squares
      problem via the incremental condition number ICN as in [5];
    if $ICN > cnmax$ **then** abort;
    if $\|r\| \leqq tol$ **then goto** L2
  end
  $test := k_2 \times \log\left[tol/\|r\|\right] / \log\left[\|r\|/\left((1.0 + 10\mathbf{u})\,\|r^{old}\|\right)\right]$;
  if $k_2 \leqq kmax - m$ **and** $test \geqq smv \times (itmax - itno)$ **then**

$k_1 := k_2 + 1; \qquad k_2 := k_2 + m;$
goto L1
end if
L2: $e_1 := (1, 0, \cdots, 0)^T; \qquad k := k_2;$
solve $\min_y \| \, \|r\| e_1 - \bar{H}_j y \|$ for y_j where $\bar{H}_j$ is described in [14];
$q := \begin{pmatrix} y_j \\ 0 \end{pmatrix}; \qquad x := x + P_1 \cdots P_j q; \qquad r := b - Ax;$
if $\|r\| \leqq tol$ **then** exit;
if $\|r^{old}\| < \|r\|$ **then**
if $\|r\| < tol^{2/3}$ **then**
exit
else
abort
end if
end if
$test := k \times \log\left[tol/\|r\|\right] / \log\left[\|r\| / \left((1.0 + 10\mathbf{u})\, \|r^{old}\|\right)\right];$
if $test \geqq bgv \times (itmax - itno)$ **then**
abort
end if
end

FIG. 1. *Adaptive GMRES(k).*

The rounding error of a sparse matrix-vector multiplication depends only on the nonzero entries in each row of the sparse matrix, so the error tolerance $xtol$ is proportional to the average number of nonzeros per row $avnz =$ (number of nonzeros in $A)/n$. Since GMRES convergence is normally measured by reduction in the initial residual norm, the convergence tolerance is $tol = \max\{\|r_0\|, \|b\|\} xtol$.

A possible symptom of AGMRES(k) going astray is an increase in the residual norm between restarts (the residual norm is computed by direct evaluation at each restart). If the residual norm on the previous restart is actually smaller than the current residual norm, then AGMRES(k) terminates. The solution is considered acceptable if $\|r\| < tol^{2/3}$, although this loss of accuracy may cause the client algorithm (the "outer" algorithm requiring solutions to linear systems) to work harder or fail. A robust client algorithm can deal gracefully with a loss of accuracy in the linear system solutions. If $\|r\| \geqq tol^{2/3}$, AGMRES(k) is deemed to have failed. In this latter case, the continuation of GMRES(k) would typically result in reaching a limit on the number of iterations allowed and a possible repetition of $\|r^{old}\| < \|r\|$ in later restarts. AGMRES(k) may exceed an iteration limit when it is affected by roundoff errors in the case of a (nearly) singular GMRES least-squares problem. The condition number of the GMRES least-squares problem is monitored by the incremental condition estimate [5] as in [6]. AGMRES(k) aborts when the estimated condition number is greater than $1/(50\mathbf{u})$.

3. Parallel implementations. In parallel environments, the choice of the orthogonalization process for the Krylov subspace basis vectors depends not only on the accuracy of the process but also on the amount and type of global communication it incurs. For some orthogonalization procedures, only one of the two requirements is satisfied. For example, in serial implementations of the GMRES method, the modified version of the Gram-Schmidt process is often used as being sufficiently accurate for a number of problems. However, in parallel GMRES implementations, other orthogonalization procedures are preferable since the modified Gram-Schmidt process exhibits a large communication overhead (see, e.g., [12]). Better efficiency is achieved with the classical Gram-Schmidt, but it is unstable. A compromise between the accuracy and communication overhead resulted in the development of parallel GMRES(k) variations with a non-Arnoldi basis that undergoes orthogonalization only at the end of a restart cycle (see, e.g., [3], [9]). At this point, an equivalent to the Arnoldi basis is recovered as the matrix Q in the QR factorization of the non-Arnoldi basis. The QR factorization is performed in parallel by a point or block version of Householder reflections [15] that have a high degree of parallelism and avoid all-to-all communications. Under the requirement of high accuracy, the errors associated with performing GMRES(k) iterations in a nonorthogonal basis are not acceptable. However, an efficient parallel implementation of Householder reflections can be employed successfully in a parallel version of the GMRES(k) algorithm, in which Householder reflections are used to compute the Arnoldi basis.

Here, an implementation of the Householder reflection orthogonalization in GMRES(k) proposed in [18] is adapted to work in parallel. The parallel version employs an algorithm developed in [15] for generating and applying Householder reflections. This algorithm avoids dot-products and all-to-all communications. The degree of parallelism equal to the number of processors in parallel Householder reflection generation is achieved by letting each processor create its local portion of the Householder reflection vector independently. After that, local Householder reflection vectors are assembled into a global Householder reflection vector using Givens rotations. The approach taken in [15] assumes a *fixed* ring of processors, in which the communication always starts from a fixed (first) processor.

Pseudocode for the algorithm generating Householder reflections (called HG) at the jth GMRES(k) iteration on the processor *proc* (in a ring of p processors) is given in Fig. 2. Before applying a Householder reflection, a sequence of Givens rotations has to be applied. Thus, Householder reflection application can be viewed as the Householder reflection generation process reversed with respect to the order of using Givens rotations and Householder reflections.

In Fig. 2, H and G denote the Householder transformation matrix and the Givens rotation matrix (as given in [13]), respectively; v_{loc} denotes a portion of the Krylov subspace vector $A^j r_0$ located on a processor; p, $left$, $right$ are the processors with the highest rank, with the $proc - 1$ rank, and the $proc + 1$ rank, respectively. It is also assumed that the first processor has the jth row of the input matrix.

However, the design presented in Fig. 2 admits only a special case of the matrix row distribution: assignment of a block of contiguous rows to each processor (call it block-striped partitioning), which is rarely advantageous for an arbitrary unstructured matrix. For large unstructured matrices graph-theoretical heuristics exist to produce

if ($proc = 1$) **then** $s := j$ **else** $s := 0$ **end if**
determine H_{s+1} such that $H_{s+1}v_{loc} \equiv w_{loc}$ has zeros
  after the $(s+1)$st component;
if ($proc = 1$) **then**
  send $w_{loc}(s+1)$ to $right$;
else
  receive w from $left$;
  determine G_1 such that $w_{loc}(1) = 0$; update w;
  if ($proc \neq p$) send w to $right$;
end if

FIG. 2. *Parallel Householder reflection generation.*

partitioning (call it graph-based) that minimizes the communication to computation ratio of the distributed matrix-vector multiply. In the current implementation, a graph partitioning algorithm from the MeTiS package [11] is used and the parallel version of the matrix-vector multiply is performed as in [12]. The matrix-vector multiplication requires that the components of all vectors are distributed in accordance with the corresponding matrix rows and allows overlapping of computation and communication. To use the algorithms in Fig. 2, the redistribution of a vector requires $\mathcal{O}(p^2)$ communications at each GMRES(k) iteration, which is highly impractical and reduces the efficiency gained by the distributed matrix-vector product. Thus, it is beneficial to develop an extension (call it MHG) of the algorithms in Fig. 2, which accepts an arbitrary row distribution among processors (Fig. 3). The key idea of this extension is to allow a *flexible* ring in performing Givens rotations. The flexible ring is the ring of processors in which the communication may start from an arbitrary processor that (in a given graph-based partitioning) holds the first component of the vector x. In practice, each processor in the flexible ring determines the current *ring_start* by consulting the global mapping array *mpart*, each entry of which is a processor number indexed by a matrix row number (where the row numbering refers to the original matrix before partitioning) located on that processor.

Usually, the subspace dimension is much smaller than the matrix dimension and the graph partitioning algorithm produces a balanced workload by assigning an almost equal number of rows to each processor. Thus, the case when the index s within v_{loc} becomes equal to the size of a local partition (size of v_{loc}) occurs rarely for large matrices, unless the number of processors is very large.

3.1. Experimental results. To compare the Householder reflections restricted to the block-striped partitioning with the proposed extension for the graph-based partitioning, GMRES was instrumented to collect the timing information relevant only to the parallel Householder reflection generation and application. The time spent in HG and MHG for generating and applying a global Householder reflection vector is independent of the type of partitioning and thus presents a good measure of the relative computational complexity of these two implementations. As a test problem, the matrix BCSSTK13 (2003×2003) was taken from the Harwell-Boeing collection [8]. This matrix is the stiffness matrix for a fluid flow eigenvalue problem from a dynamic

if $(j = 1)$ **then**
  $s := 1$;
else
  if ($proc$ has $(j-1)$st row) **then** $s := s + 1$;
end if
determine H_s such that $H_s v_{loc} \equiv w_{loc}$ has zeros
 after the sth component;
if ($proc$ has jth row) **then**
  $ring\_end := left$;
  send $w_{loc}(s+1)$ and $ring\_end$ to $right$;
else
  receive w and $ring\_end$ from $left$;
  determine G_s such that $w_{loc}(s) = 0$; update w;
  if ($proc \neq ring\_end$) **then** send w to $right$;
end if

FIG. 3. *Modified parallel Householder reflection generation.*

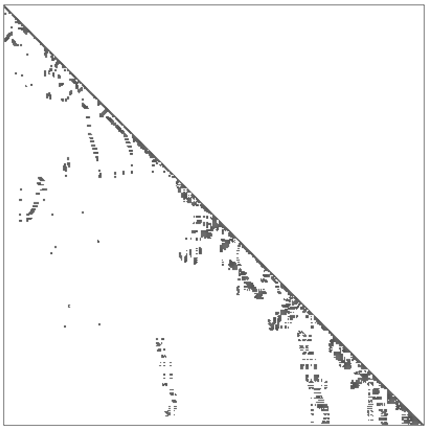

FIG. 4. *Nonzero pattern of BCSSTK13. (Due to symmetry, only the lower triangular submatrix is shown).*

analysis in structural engineering. BCSSTK13 has 83883 nonzeros and a symmetric sparsity pattern (Fig. 4).

MHG and HG generated the first Householder vector in 0.26s and 0.25s, respectively, whereas MHG was slightly faster (0.23s and 0.25s, respectively) than HG in applying a Householder reflection. (The times quoted here are the averaged parallel times over three experiments performed on 8 processors of an Intel Paragon.) Thus, the proposed MHG implementation exhibits performance comparable to HG and outperforms HG in some instances.

For the given test problem, two restart cycles of GMRES(15) were executed to determine the dependence of the overall parallel time on the type of partitioning (and thus on HG and MHG) with an increase in the number of processors. Table 1 shows the averaged results over three experiments for 4, 8, 16, 32, 64, and 100 processors, where T_p denotes the overall parallel time for GMRES(15), T^{MV} denotes the parallel time for matrix-vector multiplication, np is the maximum number of neighboring

TABLE 1
Total and matrix-vector multiplication parallel times for GMRES(15) with MHG and HG orthogonalizations.

		4	8	16	32	64	100
MHG	T_p	1.52	2.21	3.23	5.30	8.84	12.49
	T^{MV}	0.36	0.29	0.34	0.42	0.59	0.76
	np	3	7	12	17	24	25
	$Comm$	6	22	74	177	457	808
HG	T_p	1.66	1.91	3.37	5.25	8.85	12.62
	T^{MV}	0.34	0.23	0.29	0.42	0.76	1.09
	np	3	6	13	13	22	30
	$Comm$	6	17	55	135	388	699

processors, and $Comm$ is the total number of communication channels among processors. For the times observed in three experiments, the largest standard deviation was 0.02. Comparison of the results in the last two columns of Table 1 shows that with increasing numbers of processors the performance of GMRES with the MHG orthogonalization deteriorates slower than that of GMRES with the HG orthogonalization. On larger numbers of processors (cf. the last three columns), the relative difference in the number of communication channels for GMRES with MHG and HG decreases (1.31, 1.17, 1.15). On 100 processors, GMRES with MHG organizes communications among processors more uniformly by having a smaller maximum number of neighbors. Thus, with a significant increase in the number of neighbors HG performs worse than MHG for large numbers of processors.

4. Conclusions. The proposed parallel implementation of adaptive GMRES(k) takes advantage of an efficient distributed matrix-vector multiplication and the Householder reflection orthogonalization that avoids all-to-all communications and has a high degree of parallelism. The modification proposed here extends the parallel Householder reflection orthogonalization from accepting a block-striped partitioning of rows to a general graph-based partitioning. For a large number of processors, using the extended parallel implementation of Householder reflections preserves the advantage gained from applying graph-theoretical heuristics to partition a problem into subdomains.

REFERENCES

[1] D. C. S. Allison, A. Chakraborty, and L. T. Watson, *Granularity issues for solving polynomial systems via globally convergent algorithms on a hypercube*, J. Supercomputing, 3 (1989), pp. 5–20.

[2] D. C. S. Allison, K. M. Irani, C. J. Ribbens, and L. T. Watson, *High-dimensional homotopy curve tracking on a shared-memory multiprocessor*, J. Supercomput., 5 (1992), pp. 347–366.

[3] Z. Bai, D. Hu, and L. Reichel, *A Newton basis GMRES implementation*, IMA J. Numer. Anal., 4 (1994), pp. 563–581.

[4] A. Björck, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, BIT, 7 (1967), pp. 1–21.

[5] C. H. Bischof and P. T. P. Tang, *Robust incremental condition estimation*, Tech. Rep. CS-91-133, LAPACK Working Note 33, Computer Sci. Dept., Univ. of Tennessee, Knoxville, TN, May, 1991.

[6] P. N. Brown and H. F. Walker, *GMRES on (nearly) singular systems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 37–51.

[7] A. Chakraborty, D. C. S. Allison, C. J. Ribbens, and L. T. Watson, *The parallel complexity of embedding algorithms for the solution of nonlinear equations*, IEEE Trans. Parallel Distributed Systems, 4 (1993), pp. 458–465.

[8] I. Duff, R. G. Grimes, and J. G. Lewis, *Users' guide for the Harwell-Boeing sparse matrix collection (Release I)*, Tech. Report TR/PA/92/86, CERFACS, France, October, 1992.

[9] J. Erthel, *A parallel GMRES for general sparse matrices*, ETNA, 3 (1995), pp. 160–176.

[10] K. M. Irani, M. P. Kamat, C. J. Ribbens, H. F. Walker, and L. T. Watson, *Experiments with conjugate gradient algorithms for homotopy curve tracking*, SIAM J. Optim., 1 (1991), pp. 222–251.

[11] G. Karypis and V. Kumar, *MeTiS: Unstructured graph partitioning and sparse matrix ordering system*, User's Guide—Version 2.0, Dept. of Computer Sci., Univ. of Minnesota, Minneapolis, MN, 1995.

[12] G.-C. Lo and Y. Saad, *Iterative solution of general sparse linear systems on clusters of workstations*, Tech. Report, UMSI 96/117, Supercomputer Institute, Univ. of Minnesota, Minneapolis, MN, August, 1996.

[13] Y. Saad, *Iterative methods for sparse linear systems*, PWS, Boston, 1996.

[14] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual method for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[15] R. B. Sidje, *Alternatives for parallel Krylov subspace basis computation*, Numer. Linear Algebra Appl., 4 (1997), pp. 305–331.

[16] M. Sosonkina, L. T. Watson, R. K. Kapania, and H. F. Walker, *A new adaptive GMRES algorithm for achieving high accuracy*, Numer. Linear Algebra Appl., submitted.

[17] H. A. van der Vorst and C. Vuik, *The superlinear convergence behaviour of GMRES*, J. Comp. Appl. Math., 48 (1993), pp. 327–341.

[18] H. F. Walker, *Implementation of the GMRES method using Householder Transformations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 152–163.

[19] L. T. Watson, *Globally convergent homotopy algorithms for nonlinear systems of equations*, Nonlinear Dynamics, 1 (1990), pp. 143–191.

[20] ———, *Globally convergent homotopy methods: a tutorial*, Appl. Math. Comput., 31BK (1989), pp. 369–396.