

## EFFICIENT HIGH ACCURACY SOLUTIONS WITH GMRES( $m$ )\*

KATHRYN TURNER<sup>†</sup> AND HOMER F. WALKER<sup>†</sup>

**Abstract.** Consideration of an abstract improvement algorithm leads to the following principle, which is similar to that underlying iterative refinement: By making judicious use of relatively few high accuracy computations, high accuracy solutions can be obtained very efficiently by the algorithm. This principle is applied specifically to GMRES( $m$ ) here; it can be similarly applied to a number of other “restarted” iterative linear methods as well. Results are given for numerical experiments in solving a discretized linear elliptic boundary value problem and in computing a step of an inexact Newton method using finite differences for a discretized nonlinear elliptic boundary value problem.

**Key words.** GMRES( $m$ ), “restarted” iterative linear methods, high accuracy solutions, iterative refinement, large-scale linear and nonlinear systems, elliptic boundary value problems

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

**1. Introduction.** Our specific objective is to outline efficient algorithms for obtaining high accuracy with GMRES( $m$ ), the restarted version of the generalized minimal residual (GMRES) method of Saad and Schultz [10] for numerically solving general nonsingular linear systems. These algorithms are based on general observations which have broader applicability and are of interest in their own right, and we begin with them.

We consider a very abstract *improvement algorithm* for a problem in which the solution  $x^* \in \mathbb{R}^n$  is characterized as a zero of a *residual function*  $r: \mathbb{R}^n \rightarrow \mathbb{R}^n$ , i.e.,  $r(x^*) = 0$ . The algorithm centers around an unspecified *correction process* which, given a current approximate solution  $x$ , computes a correction  $z$  such that  $x + z$  is, one hopes, an improved approximation of  $x^*$ . Our presumption is that the accuracy in the computed value of  $z$  is limited by the accuracy in the computed value of  $r(x)$ . Therefore, we explicitly require in the algorithm that  $r(x)$  be computed accurately, by which we mean as accurately as is reasonably possible or affordable, or in some sense especially accurately. For similar reasons, on which we expand in the following, we require  $x + z$  to be computed accurately in the same sense.

### Algorithm GI: General Improvement Algorithm

```

LET AN INITIAL  $x$  BE GIVEN.
UNTIL TERMINATION, DO:
    COMPUTE  $r(x)$  ACCURATELY.
    DETERMINE  $z$  BY THE CORRECTION PROCESS.
    UPDATE  $x \leftarrow x + z$  ACCURATELY.
END DO.
```

The termination criterion may be based on the smallness of  $\|r(x)\|$ , or on some related but different feature, as in the iterative refinement example below.

Our interest is in the limits of accuracy which can be obtained by Algorithm GI. Since the actual error in  $x$  is unknown, we take “limits of accuracy” to mean limits of reduction of  $\|r(x)\|$ , where  $\|\cdot\|$  is an appropriate norm on  $\mathbb{R}^n$ . If we assume that

\* Received by the editors May 29, 1990; accepted for publication (in revised form) February 6, 1991.

<sup>†</sup> Department of Mathematics and Statistics, Utah State University, Logan, Utah 84322-3900. The work of the second author was supported by United States Department of Energy grant DE-FG02-86ER25018, National Science Foundation grant DMS-0088995, and United States Air Force grant AFOSR-91-0294, all with Utah State University.

the accuracy in the computed value of  $z$  is limited only by the relative error in the computed value of  $r(x)$ , and not by the smallness of  $\|r(x)\|$  per se, then it is clear that accuracy is limited only by the accuracy which can be maintained in computing  $r(x)$  and in updating  $x \leftarrow x + z$ .

A familiar example of Algorithm GI is *iterative refinement* for improving approximate solutions of a system of linear equations. (See, e.g., Golub and Van Loan [7, §3.5.3] as a general reference.) Consider a linear problem written as follows:

$$(1) \quad \text{Given } A \in \mathbb{R}^{n \times n} \text{ and } b \in \mathbb{R}^n, \quad \text{find } x^* \in \mathbb{R}^n \text{ such that } Ax^* = b.$$

Suppose that we have computed factors of  $A$  and have used them to compute an initial  $x$ . In iterative refinement, we take  $r(x) \equiv b - Ax$  and, at each step, determine a correction  $z$  by solving  $Az = r(x)$  using the previously computed factors of  $A$ . Iterative refinement is typically applied when we can afford to do most computing only in single precision, but full single precision accuracy is desired in the computed solution. Thus  $A$ ,  $b$ ,  $x$ ,  $r(x)$ , and  $z$  are stored in single precision, and the factors of  $A$  are computed and stored in single precision. The accuracy in  $z$  depends on the relative error in the computed value of  $r(x)$  and not on the smallness of  $\|r(x)\|$ , provided all computations remain within the floating point range of the machine. However, if  $r(x)$  is small, then special care must be taken to compute it accurately. The usual prescription is to compute it in double precision in a well-known way. Unless  $A$  is very ill conditioned, this results in sufficient accuracy in  $r(x)$  and  $z$  to provide improvement in  $x$  until  $z$  is so small that  $x$  and  $x + z$  have the same single precision representation, i.e., the updating  $x \leftarrow x + z$  results in no change, at which point  $x$  is the single precision representation of the exact solution and the algorithm terminates. Since only single precision accuracy is desired in  $x$ , there is no need to form  $x + z$  with more than single precision accuracy. If higher accuracy *were* desired in  $x$ , then it could be obtained by maintaining  $x$  in double precision and forming  $x + z$  with  $z$  represented in double precision; however, this is unlikely to be of practical interest.

This discussion illustrates our guiding principle: By making judicious use of relatively few high accuracy calculations within Algorithm GI, specifically in computing  $r(x)$  and in updating  $x \leftarrow x + z$ , we can obtain essentially the accuracy that would be realized by using high accuracy calculations throughout, while maintaining storage requirements and execution times only modestly above those which would be required if only low accuracy calculations were used.

We now turn to the case of particular interest, viz., that of problem (1) in which  $r(x) \equiv b - Ax$  and the correction process consists of one cycle of GMRES( $m$ ). Although we focus exclusively on GMRES( $m$ ), we emphasize that Algorithm GI and the guiding principle drawn from it are more broadly applicable. In particular, there are a number of other “restarted” iterative linear methods which could be used in place of GMRES( $m$ ) in the algorithms formulated below, e.g., a restarted version of a method of Axelsson [1], the iterative Arnoldi method of Saad [9, §3.2], and GCR( $m$ ) given by Elman [5] and Eisenstat, Elman, and Schultz [4].

For completeness, we give very brief descriptions of GMRES and GMRES( $m$ ) here; for more detailed descriptions, see Saad and Schultz [10] or Walker [12], [13]. The GMRES method begins with an initial approximate solution  $x$  and initial residual  $r(x) = b - Ax$ . At the  $k$ th iteration, a correction  $z$  is characterized as the solution of the least-squares problem

$$(2) \quad \min_{\bar{z} \in \mathcal{K}_k} \|b - A(x + \bar{z})\|_2 = \min_{\bar{z} \in \mathcal{K}_k} \|r(x) - A(\bar{z})\|_2,$$

where  $\mathcal{K}_k$  is the  $k$ th Krylov subspace generated by  $A$  and  $r(x)$ , defined by

$$\mathcal{K}_k \equiv \text{span}\{r(x), Ar(x), \dots, A^{k-1}r(x)\}.$$

The correction  $z$  is not actually computed by the method unless a termination criterion is satisfied. If it is computed, then the least-squares problem (2) is solved by using a basis of  $\mathcal{K}_k$  to reduce (2) to a  $k$ -dimensional least-squares problem for the coefficients of the basis elements. The basis used in practice is the orthonormal *Arnoldi basis*, generated inductively by

$$(3) \quad \begin{aligned} v_1 &= r(x)/\|r(x)\|_2, \\ v_{j+1} &= P_{\mathcal{K}_j}^\perp Av_j / \|P_{\mathcal{K}_j}^\perp Av_j\|_2 \quad \text{for } j = 1, \dots, k-1. \end{aligned}$$

Here,  $P_{\mathcal{K}_j}^\perp v$  denotes the orthogonal projection of a vector  $v$  onto the orthogonal complement of  $\mathcal{K}_j$ ; it can be computed using either the modified Gram–Schmidt process [10] or Householder transformations [12]. We use the modified Gram–Schmidt process below since it is used more often in practice. Note that the process (3) fails for some  $j < k$  if and only if  $P_{\mathcal{K}_j}^\perp Av_j = 0$ , i.e.,  $Av_j \in \mathcal{K}_j$ , in which case it can be shown that GMRES produces  $z \in \mathcal{K}_j$  such that  $r(x+z) = 0$ , i.e.,  $x+z = x^*$  (see [10, Prop. 2, p. 865]). The process (3) is implemented as the GMRES iterations proceed, i.e.,  $v_1$  is determined and stored initially and at the  $k$ th iteration, (only)  $v_{k+1}$  is generated and added to storage. At the  $k$ th iteration,  $k+1$   $n$ -vectors must be stored, and orthogonalizing  $Av_k$  against  $v_1, \dots, v_k$  requires  $O(kn)$  arithmetic operations.

Because storage and arithmetic costs increase as the GMRES iterations proceed, the algorithm usually implemented in practice is GMRES( $m$ ), where  $m$  is some maximum allowable number of iterations which is prescribed at the outset, typically much less than  $n$ . In one cycle of GMRES( $m$ ), beginning with a current approximate solution  $x$ , up to  $m$  GMRES iterations are performed; the iterations are terminated when either the residual norm has been reduced sufficiently or the full  $m$  iterations have been taken. The residual norm can be monitored during the iterations without actually computing the correction or the residual; see below. After the iterations have been performed, the cycle is concluded with the computation of the correction  $z$ . If  $\|r(x+z)\|_2$  is not sufficiently small, then additional cycles may be performed as needed.

We give below a detailed outline of one cycle of GMRES( $m$ ) as used here. In it, for each  $i$ ,  $J_i$  denotes a Givens rotation which “rotates” components  $i$  and  $i+1$  of vectors on which it acts; see, e.g., [7, §5.1] for definitions and properties of Givens rotations. Also, for each  $k$ ,  $(J_k \rho)_{k+1}$  denotes the  $(k+1)$ st component of the vector  $J_k \rho$  and  $w_1, \dots, w_{k+1}$  denote the first  $k+1$  components of the vector  $w$ .

**Algorithm:** One Cycle of GMRES( $m$ )

LET  $r(x) \neq 0$ ,  $m$ , AND  $\text{TOL} > 0$  BE GIVEN.

INITIALIZE:

SET  $v_1 = r(x)/\|r(x)\|_2$  AND  $w = (\|r(x)\|_2, 0, \dots, 0)^T \in \mathbb{R}^{m+1}$ .

ITERATE:

FOR  $k = 1, 2, \dots, m$ , DO:

EVALUATE  $v_{k+1} = Av_k$ .

FOR  $i = 1, \dots, k$ , DO:

SET  $\rho_i = v_i^T v_{k+1}$ .

OVERWRITE  $v_{k+1} \leftarrow v_{k+1} - \rho_i v_i$ .

END DO.

```

SET  $\rho_{k+1} = \|v_{k+1}\|_2$ .
IF  $\rho_{k+1} \neq 0$ , OVERWRITE  $v_{k+1} \leftarrow v_{k+1}/\rho_{k+1}$ .
SET  $\rho = (\rho_1, \dots, \rho_{k+1}, 0, \dots, 0)^T \in \mathbb{R}^{m+1}$ .
IF  $k > 1$ , OVERWRITE  $\rho \leftarrow J_{k-1} \cdots J_1 \rho$ .
FIND  $J_k$  SUCH THAT  $(J_k \rho)_{k+1} = 0$  AND OVERWRITE
 $\rho \leftarrow J_k \rho$  AND  $w \leftarrow J_k w$ .
SET  $R_k = [\rho]$  IF  $k = 1$  AND  $R_k = [R_{k-1}, \rho]$  IF  $k > 1$ .
IF  $|w_{k+1}| < \text{TOL}$ , THEN GO TO SOLVE.
END DO.
SOLVE:
LET  $k$  BE THE FINAL ITERATION NUMBER REACHED, LET
 $\bar{R}_k \in \mathbb{R}^{k \times k}$  BE THE UPPER TRIANGULAR MATRIX DETERMINED
BY THE FIRST  $k$  ROWS OF  $R_k$ , AND SET  $\bar{w} = (w_1, \dots, w_k)^T \in \mathbb{R}^k$ .
SOLVE  $\bar{R}_k y = \bar{w}$  FOR  $y$ .
FORM  $z = [v_1, \dots, v_k]y$ .

```

The test  $|w_{k+1}| < \text{TOL}$  at the final step of the iteration is explained by the fact that if the correction  $z$  were formed, then we would have  $\|r(x+z)\|_2 = |w_{k+1}|$ , at least in exact arithmetic; see [10, Prop. 1, p. 862]. Also, we note that  $\bar{R}_k$  is always nonsingular.

Our general efficient high accuracy algorithm centering around GMRES( $m$ ) is the following adaptation of Algorithm GI.

**Algorithm EHA:** Efficient High Accuracy Algorithm

```

LET AN INITIAL  $x$  BE GIVEN. UNTIL TERMINATION, DO:
  COMPUTE  $r(x)$  ACCURATELY.
  DETERMINE  $z$  BY ONE CYCLE OF GMRES( $m$ ).
  UPDATE  $x \leftarrow x + z$  ACCURATELY.
END DO.

```

Algorithm EHA terminates when the residual norm becomes less than a given tolerance  $\text{TOL} > 0$ . If  $\|r(x)\|_2 < \text{TOL}$  prior to beginning a GMRES( $m$ ) cycle, then the algorithm terminates immediately; if this criterion is met during a GMRES( $m$ ) cycle, then the algorithm terminates as soon as the updating  $x \leftarrow x + z$  is done.

In the following, we explore applications of Algorithm EHA. In §2, we show how the steps in Algorithm EHA can be specified so that essentially all of the accuracy obtainable from a full double precision implementation of GMRES( $m$ ) can be obtained with storage requirements and arithmetic costs which, in a typical problem, are only a little greater than those of a single precision implementation. In §3, we consider the computation of inexact Newton steps for a nonlinear problem by means of Algorithm EHA, in which products of the Jacobian (matrix) with vectors are approximated by finite differences. We show that by using high-order finite difference approximations in the computation of each  $r(x)$  and low-order finite difference approximations in each cycle of GMRES( $m$ ), we can obtain essentially the accuracy that could be obtained by using high-order finite difference approximations throughout, but at much less cost.

**2. Obtaining double precision accuracy efficiently.** Here we show how Algorithm EHA can be used to obtain essentially double precision accuracy efficiently in the solution of a linear problem (1). By double precision accuracy, we mean the accuracy which could be obtained by using double precision arithmetic throughout; we do not mean that the result will be the exact solution rounded to double precision.

In applying Algorithm EHA, we use three double precision vectors: one for the approximate solution  $x$ , one for the right-hand side  $b$ , and one for the matrix-vector product  $Ax$ . A subroutine to compute  $Ax$  in double precision is needed in addition to a subroutine that is used within GMRES( $m$ ) to compute products of  $A$  with other vectors in single precision. We apply Algorithm EHA as follows: Given a double precision approximate solution  $x$ , we first compute  $r(x)$  by computing  $Ax$  in double precision, subtracting  $Ax$  from  $b$  in double precision, and then rounding the difference to single precision. A cycle of GMRES( $m$ ) is then carried out, entirely in single precision, producing a single precision correction  $z$ . The update of the approximate solution is  $x \leftarrow x + \text{dble}(z)$ , where  $\text{dble}(z)$  indicates that  $z$  is represented in double precision in the addition. This representation is done componentwise and does not require that  $z$  be converted in toto to a double precision vector.

**2.1. Numerical experiments.** We conducted numerical experiments on non-symmetric linear systems arising from the discretization of the elliptic boundary value problem

$$(4) \quad \begin{aligned} \Delta w + cw + d \frac{\partial w}{\partial x} &= f && \text{in } D, \\ w &= 0 && \text{on } \partial D, \end{aligned}$$

where  $D = [0, 1] \times [0, 1]$  and  $c \geq 0$  and  $d$  are constants. In the experiments reported here, we took  $f \equiv 1$  and used a  $100 \times 100$  mesh of equally spaced discretization points in  $D$ , so that the resulting linear systems were of dimension 10,000. Discretization was by the usual second-order centered differences.

We compared Algorithm EHA as specified above with methods which differed only in that either exclusively single or exclusively double precision arithmetic and storage were used throughout. We refer to the latter two methods as the full single precision (FSP) and full double precision (FDP) methods, respectively. We note that in the context of these experiments, Algorithm EHA requires only a little more storage than the FSP method, which requires about half that of the FDP method. The additional storage required by Algorithm EHA over the FSP method arises mainly from the double precision storage of  $b$ ,  $x$ , and  $Ax$ . In the double precision computation of  $Ax$ , the simplicity of  $A$  in these experiments allows the nonzero entries of  $A$  to be represented by a few nonzero constants.

In GMRES( $m$ ), we used  $m = 10$  in all of the experiments reported here because that seemed to be more effective than other choices we tried. All computing was done on a Sun Microsystems SPARCstation 1 using the SunOS FORTRAN compiler.

In our first set of experiments, we took  $c = d = 10$  and used right preconditioning with a fast Poisson solver from FISHPACK [11], which is very effective for these fairly small values of  $c$  and  $d$ . We first started each method with zero as the initial approximate solution and allowed it to run for 40 GMRES( $m$ ) iterations, after which the limit of residual norm reduction had been reached. Figure 1 shows plots of the logarithm of the Euclidean norm of the residual versus the number of GMRES( $m$ ) iterations for the three methods. We note that in Fig. 1 and in all other figures below, the plotted residual norms were not the values maintained by GMRES( $m$ ), but rather were computed as accurately as possible "from scratch." That is, at each GMRES( $m$ ) iteration, the current approximate solution was formed and its product with the coefficient matrix was subtracted from the right-hand side, all in double precision. It was important to compute the residual norms in this way because the values maintained by GMRES( $m$ ) become increasingly untrustworthy as the limits

of residual norm reduction are neared; see [12]. It is seen in Fig. 1 that Algorithm EHA achieved the same ultimate level of residual norm reduction as the FDP method and required only a few more GMRES( $m$ ) iterations to do so.

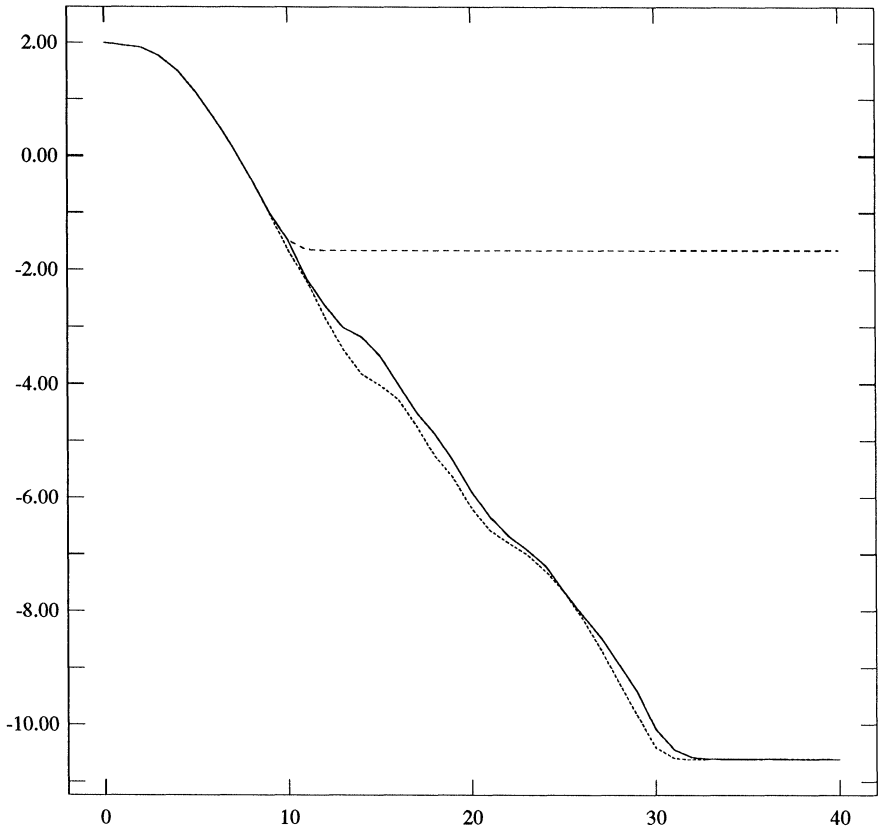


FIG. 1.  $\text{Log}_{10}$  of the residual norm versus the number of GMRES( $m$ ) iterations for  $c = d = 10$  with fast Poisson preconditioning. Solid curve: Algorithm EHA; dotted curve: FDP method; dashed curve: FSP method.

In order to assess the relative amount of computational work required by Algorithm EHA and the FDP method to reach comparable levels of residual norm reduction, we observed in each of 20 trials the GMRES( $m$ ) iteration numbers and run times required by each method to reduce the residual norm by a factor of  $10^{-12}$ . Since timing was of major interest, the residual norms were not computed “from scratch” in these trials; instead, the values maintained by GMRES( $m$ ) were used. These values were trustworthy because  $10^{-12}$  is significantly larger than the limiting residual norm reduction factor for these methods evident in Fig. 1. In each trial, the components of the initial approximate solution were obtained by generating uniformly distributed random numbers over  $[-1, 1]$ . The means and standard deviations of the run times are given in Table 1. Each method required exactly 30 GMRES( $m$ ) iterations to reach termination in every trial, which accounts for the small standard deviations.

In our second set of experiments, we took  $c = d = 100$  and carried out trials analogous to those in the first set above. No preconditioning was used in these experiments, both because we wanted to compare the methods without preconditioning and because the fast Poisson preconditioning used in the first set of experiments is not

TABLE 1

Statistics over 20 trials of run times required to reduce the residual norm by a factor of  $10^{-12}$ . Fast Poisson preconditioning;  $c = d = 10$ . Each method took 30 GMRES( $m$ ) iterations to terminate in every trial.

Method	Mean Run Time (Seconds)	Standard Deviation
EHA	28.77	.03125
FDP	45.80	.06442

cost effective for these large values of  $c$  and  $d$ . We first allowed each method to run for 600 GMRES( $m$ ) iterations, starting with zero as the initial approximate solution, after which the limit of residual norm reduction had been reached. The results are shown in Fig. 2. We note that Algorithm EHA reached the same ultimate level of residual norm reduction as the FDP method but required about ten percent more GMRES( $m$ ) iterations to reach this level.

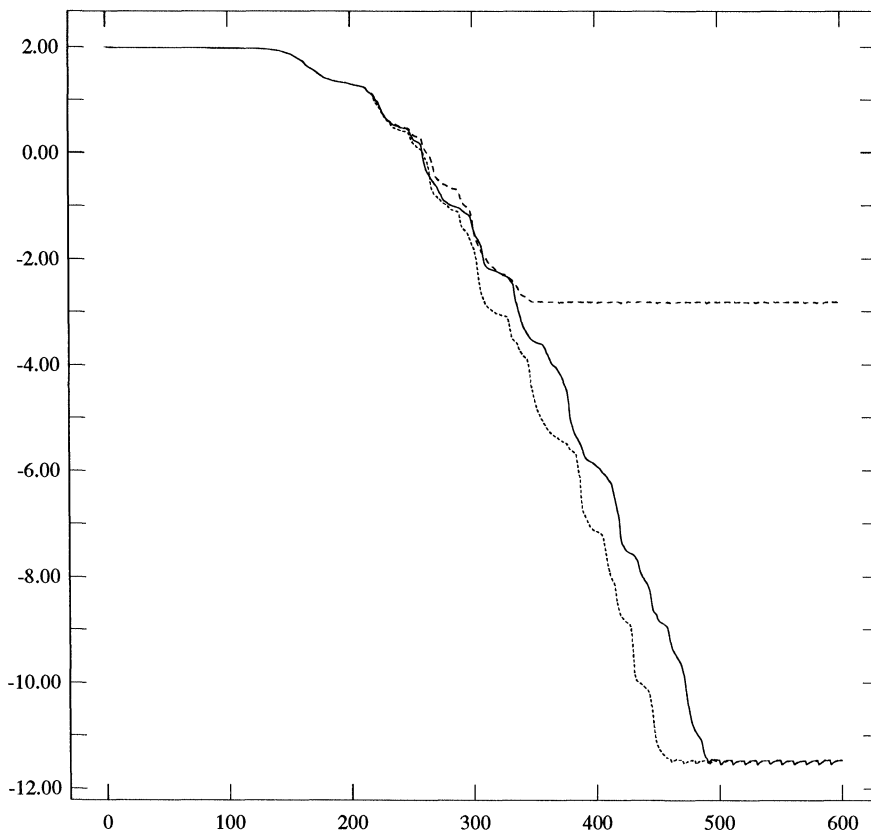


FIG. 2.  $\log_{10}$  of the residual norm versus the number of GMRES( $m$ ) iterations for  $c = d = 100$  with no preconditioning. Solid curve: Algorithm EHA; dotted curve: FDP method; dashed curve: FSP method.

We then observed in each of 20 trials the numbers of GMRES( $m$ ) iterations and run times required by Algorithm EHA and the FDP method to reduce the residual norm by a factor of  $10^{-12}$ . In these trials, the initial approximate solutions were obtained by generating random components as in the previous such trials. In contrast

TABLE 2  
*Statistics over 20 trials of GMRES( $m$ ) iteration numbers and run times required to reduce the residual norm by a factor of  $10^{-12}$ . No preconditioning;  $c = d = 100$ .*

Method	Mean Number of Iterations	Standard Deviation	Mean Run Time (Seconds)	Standard Deviation
EHA	346.2	34.18	91.57	9.068
FDP	345.9	35.56	153.0	15.72

to the previous trials, there was in these trials significant variation in both the numbers of GMRES( $m$ ) iterations and the run times for each method. Consequently, the means and standard deviations of the GMRES( $m$ ) iteration counts as well as the run times are given in Table 2.

**3. Computing inexact Newton steps using finite differences.** We now consider an inexact Newton method [3] for the solution of a nonlinear problem, written as follows:

(5)            Given  $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ ,   find  $u^* \in \mathbb{R}^n$  such that  $F(u^*) = 0$ .

Given an approximate solution  $u$  of (5), an inexact Newton method determines a step  $x$  such that  $F'(u)x \approx -F(u)$  and updates  $u \leftarrow u + x$ ; such a method is of interest when computing the exact Newton step  $-F'(u)^{-1}F(u)$  is infeasible. We shall apply Algorithm EHA to the computation of a single inexact Newton step, in which  $u$  remains fixed and  $r(x) \equiv -F(u) - F'(u)x$ .

We focus on the case in which  $F'(u)$  cannot be used analytically. In this case, we might approximate  $F'(u)$  using finite differences. However, with GMRES( $m$ ),  $F'(u)$  or an approximation of it is not explicitly needed; we require only its action on vectors  $v$ , which can be approximated, e.g., to first, second, fourth, and sixth order, respectively, by

(6)            
$$\frac{1}{\delta}[F(u + \delta v) - F(u)],$$

(7)            
$$\frac{1}{2\delta}[F(u + \delta v) - F(u - \delta v)],$$

(8)            
$$\frac{1}{6\delta}\left[8F\left(u + \frac{\delta}{2}v\right) - 8F\left(u - \frac{\delta}{2}v\right) - F(u + \delta v) + F(u - \delta v)\right],$$

(9)            
$$\begin{aligned} \frac{1}{90\delta}\left[256F\left(u + \frac{\delta}{4}v\right) - 256F\left(u - \frac{\delta}{4}v\right) - 40F\left(u + \frac{\delta}{2}v\right) \right. \\ \left. + 40F\left(u - \frac{\delta}{2}v\right) + F(u + \delta v) - F(u - \delta v)\right]. \end{aligned}$$

In an inexact Newton method,  $F(u)$  is already available; therefore, each of (6)–(9) requires a number of new  $F$ -evaluations equal to its order.

We apply Algorithm EHA in this context by using a higher-order formula, e.g., one of (7)–(9), in the residual computation preceding each cycle of GMRES( $m$ ) and using a lower-order formula, e.g., the first-order formula (6), for matrix-vector products required within GMRES( $m$ ).



**3.1. Numerical experiments.** We conducted numerical experiments in computing inexact Newton steps for discretizations of a *modified Bratu problem*, given by

$$(10) \quad \begin{aligned} \Delta w + ce^w + d \frac{\partial w}{\partial x} &= f && \text{in } D, \\ w &= 0 && \text{on } \partial D, \end{aligned}$$

where  $c$  and  $d$  are constants. The actual Bratu problem has  $d = 0$  and  $f \equiv 0$ . It provides a simplified model of nonlinear diffusion phenomena, e.g., in combustion and semiconductors, and has been considered by Glowinski, Keller, and Rheinhardt [6], as well as by a number of other investigators; see [6] and the references therein. See also problem 3 by Glowinski and Keller and problem 7 by Mittelmann in the collection of nonlinear model problems assembled by Moré [8]. The modified problem (10) has been used as a test problem for inexact Newton methods by Brown and Saad [2].

In our experiments, we took  $D = [0, 1] \times [0, 1]$ ,  $f \equiv 0$ ,  $c = d = 10$ , and discretized (10) using the usual second-order centered differences over a  $100 \times 100$  mesh of equally spaced points in  $D$ . In GMRES( $m$ ), we took  $m = 10$  and used fast Poisson right preconditioning as in the experiments in §2. The computing environment was as described in §2. All computing was done in double precision.

Letting  $F$  denote the nonlinear function obtained by discretizing (10) and letting  $u$  denote an approximate solution of the discretized problem, we used (6)–(9) in approximating Jacobian-vector products  $F'(u)v$ . The value of  $\delta$  used for each formula was chosen according to the usual heuristic when  $F$  is evaluated to full machine precision: If  $p$  is the order of the formula, then  $\delta = \mathbf{u}^{1/(p+1)}$ , where  $\mathbf{u}$  is unit roundoff (taken to be  $10^{-16}$  here).

The methods we compared in our experiments are the following:

1. Four methods, each of which used one of (6), (7), (8), or (9) exclusively for all Jacobian-vector products. We refer to these below as FD1, FD2, FD4, and FD6, respectively.
2. Three versions of Algorithm EHA, each of which used one of (7), (8), or (9) exclusively in the accurate evaluation of initial residuals and then used (6) exclusively in the GMRES( $m$ ) cycles. We refer to these below as EHA2, EHA4, and EHA6, respectively.
3. A method in which all Jacobian-vector products were evaluated analytically. We refer to this as method A.

In the first set of experiments, we allowed each method to run for 40 GMRES( $m$ ) iterations, starting with zero as the initial approximate solution, after which the limit of residual norm reduction had been reached. The results are shown in Fig. 3. In Fig. 3, the top curve was produced by method FD1. The second curve from the top is actually a superposition of the curves produced by methods EHA2 and FD2; the two curves are visually indistinguishable. Similarly, the third curve from the top is a superposition of the curves produced by methods EHA4 and FD4, and the fourth curve from the top, which lies barely above the bottom curve, is a superposition of the curves produced by methods EHA6 and FD6. The bottom curve was produced by method A.

In the second set of experiments, our purpose was to assess the relative amount of computational work required by the methods which use higher-order differencing to reach comparable levels of residual norm reduction. We compared pairs of methods EHA2 and FD2, EHA4 and FD4, and EHA6 and FD6 by observing in each of 20

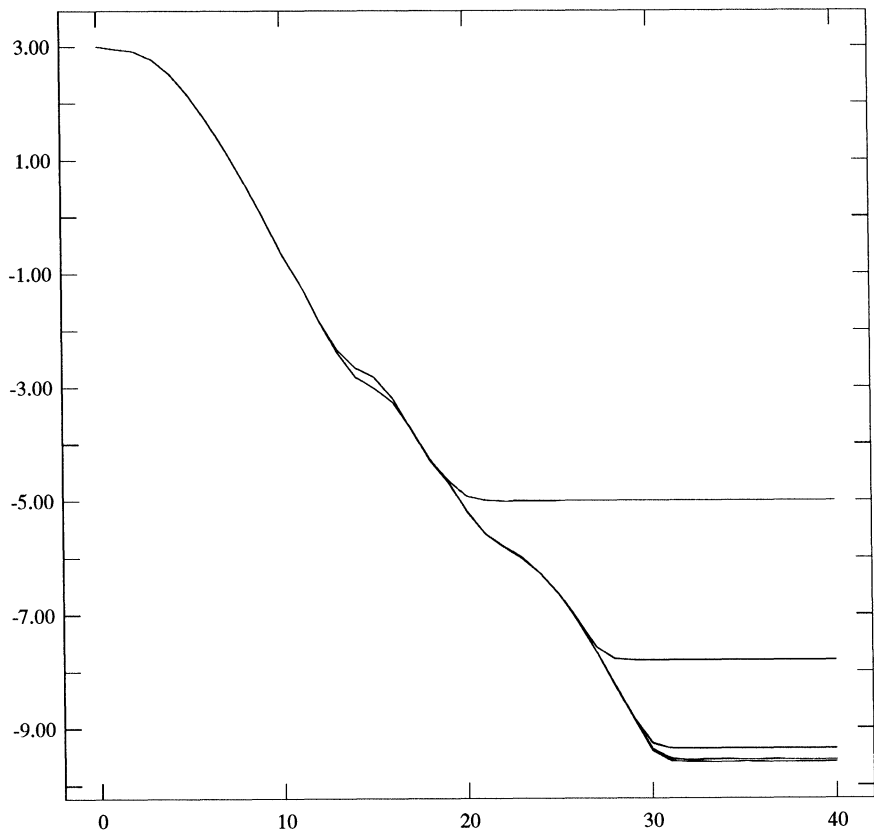


FIG. 3.  $\text{Log}_{10}$  of the residual norm versus the number of GMRES( $m$ ) iterations for the finite difference methods.

TABLE 3  
Statistics over 20 trials of GMRES( $m$ ) iteration numbers,  $F$ -evaluations, and run times required to reduce the residual norm by a factor of  $\epsilon$ . For each method, the number of GMRES( $m$ ) iterations and  $F$ -evaluations was the same in every trial.

Method	$\epsilon$	Number of Iterations	Number of $F$ -Evaluations	Mean Run Time (Seconds)	Standard Deviation
EHA2	$10^{-10}$	26	32	47.12	.1048
FD2	$10^{-10}$	26	58	53.79	.1829
EHA4	$10^{-12}$	30	42	56.76	.1855
FD4	$10^{-12}$	30	132	81.35	.3730
EHA6	$10^{-12}$	30	48	58.56	.1952
FD6	$10^{-12}$	30	198	100.6	.3278

trials the number of GMRES( $m$ ) iterations, number of  $F$ -evaluations, and run time required by each method to reduce the residual norm by a factor of  $\epsilon$ , where for each pair of methods  $\epsilon$  was chosen to be somewhat greater than the limiting ratio of final to initial residual norms obtainable by the methods. In these trials, the initial approximate solutions were obtained by generating random components as in the similar experiments in §2. We note that for every method, the numbers of GMRES( $m$ ) iterations and  $F$ -evaluations required before termination did not vary at all over the 20 trials. The GMRES( $m$ ) iteration counts, numbers of  $F$ -evaluations,

and means and standard deviations of the run times are given in Table 3.

**Acknowledgments.** We thank John Dennis and Gene Golub for stimulating conversations relating to this work.

#### REFERENCES

- [1] O. AXELSSON, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1–16.
- [2] P. N. BROWN AND Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.
- [3] R. S. DEMBO, S. C. EISENSTAT, AND T. STEihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
- [4] S. C. EISENSTAT, H. C. ELMAN, AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.
- [5] H. C. ELMAN, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT, 1982.
- [6] R. GLOWINSKI, H. B. KELLER, AND L. RHEINHART, *Continuation-conjugate gradient methods for the least-squares solution of nonlinear boundary value problems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 793–832.
- [7] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Second Edition, The Johns Hopkins University Press, Baltimore, MD, 1989.
- [8] J. J. MORÉ, *A collection of nonlinear model problems*, in Computational Solutions of Nonlinear Systems of Equations, E. L. Allgower and K. Georg, eds., Lectures in Applied Mathematics, Vol. 26, American Mathematical Society, Providence, RI, 1990, pp. 723–762.
- [9] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105–126.
- [10] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual method for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
- [11] P. N. SWARZTRAUBER AND R. A. SWEET, *Efficient FORTRAN subprograms for the solution of elliptic partial differential equations*, ACM Trans. Math. Software, 5 (1979), pp. 352–364.
- [12] H. F. WALKER, *Implementation of the GMRES method using Householder transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 152–163.
- [13] ———, *Implementations of the GMRES method*, Computer Phys. Comm., 53 (1989), pp. 311–320.