SIAM J. MATRIX ANAL. APPL.
Vol. 18, No. 3, pp. 535–551, July 1997

ESTIMATING THE ATTAINABLE ACCURACY OF RECURSIVELY COMPUTED RESIDUAL METHODS*

ANNE GREENBAUM[†]

Abstract. Many conjugate gradient-like methods for solving linear systems $Ax = b$ use recursion formulas for updating residual vectors instead of computing the residuals directly. For such methods it is shown that the difference between the actual residuals and the updated approximate residual vectors generated in finite precision arithmetic depends on the machine precision ϵ and on the maximum norm of an iterate divided by the norm of the true solution. It is often observed numerically, and can sometimes be proved, that the norms of the updated approximate residual vectors converge to zero or, at least, become orders of magnitude smaller than the machine precision. In such cases, the actual residual norm reaches the level $\epsilon\|A\|\|x\|$ times the maximum ratio of the norm of an iterate to that of the true solution. Using exact arithmetic theory to bound the size of the iterates, we give a priori estimates of the size of the final residual for a number of algorithms.

Key words. iterative methods, accuracy, finite precision arithmetic

AMS subject classifications. 65F10, 65F15

PII. S0895479895284944

1. Introduction.

Many iterative methods for solving a linear system $Ax = b$ start with an initial guess x^0 for the solution, compute the initial residual $r^0 = b - Ax^0$, and then generate updated approximations x^k and residuals r^k, $k = 1, 2, \ldots$, according to the following formulas:

$$x^k = x^{k-1} + a_{k-1}p^{k-1}, \qquad r^k = r^{k-1} - a_{k-1}Ap^{k-1}. \tag{1}$$

Here p^{k-1} is some direction vector and a_{k-1} some coefficient. The vector r^k is updated instead of computed directly as $b - Ax^k$. We are particularly interested in methods in which the coefficients and direction vectors are computed in terms of the updated vectors r^k.

Often a preconditioner M, which approximates A but is easier to invert than A, is used to accelerate convergence. In this case a vector z^k is computed by solving $Mz^k = r^k$. The solution z^k may be used in determining the new coefficient and direction vector, but it does not alter the general formulas in (1). It is these formulas that we will analyze in finite precision arithmetic.

Examples of algorithms that are usually implemented in the form (1) include the following.

- The steepest descent method for symmetric positive definite problems. Here the vector p^{k-1} is equal to z^{k-1}, and the coefficient a_{k-1} is chosen to minimize the A^{-1}-norm of r^k, $\|r^k\|_{A^{-1}} \equiv \langle r^k, A^{-1}r^k\rangle^{1/2}$, in the direction p^{k-1}.
- The conjugate gradient (CG) method for symmetric positive definite problems [16, 5]. Here the coefficient a_{k-1} is chosen to minimize the A^{-1}-norm of r^k in the direction p^{k-1}, and the direction vectors form an A-orthogonal basis for the Krylov space $[z^0, M^{-1}Az^0, \ldots, (M^{-1}A)^{k-1}z^0]$.
- BCG and CGS for general nonsymmetric problems [9, 18]. These methods use two sets of recurrences, one involving A, like formulas (1), and another involving

* Received by the editors April 21, 1995; accepted for publication (in revised form) by D. P. O'Leary July 2, 1996. This work was supported in part by DOE contract DEFG0288ER25053.
http://www.siam.org/journals/simax/18-3/28494.html

[†] Courant Institute of Mathematical Sciences, 251 Mercer St., New York, NY 10012 (greenbau@nyu.edu).

A^T. They may break down, even in exact arithmetic, due to the breakdown of the underlying two-sided Lanczos recurrence or the nonexistence of the BCG iterate at certain steps. In either case look-ahead steps may be used to avoid the steps at which the Lanczos vectors or BCG iterates are undefined [11]. We will consider only problems in which look-ahead steps are not required and x^k and r^k are updated as in (1).

• CGNR and CGNE for nonsymmetric problems [16, 6]. These methods are like the CG method for the normal equations $A^TAx = A^Tb$ or $AA^Ty = b,\ x = A^Ty$. In CGNR, the coefficient a_{k-1} is chosen to minimize the 2-norm of r^k, while in CGNE it minimizes the $(A^TA)^{-1}$-norm of r^k, $\langle A^{-1}r^k, A^{-1}r^k\rangle^{1/2}$. While the Krylov space over which the minimization is performed is the same as that used by CG applied to the normal equations, one does not actually form the normal equations, and approximate solutions and residual vectors are still computed by formulas (1).

A number of other methods are *not* usually implemented in the form (1), but, with some modifications, they could be. These include the following.

• Stationary iterative methods, such as Jacobi, Gauss–Seidel, SOR, etc. For these methods, residuals are usually computed directly. They could be updated as in (1), but there appears to be no particular advantage in doing so.

• ORTHOMIN and ORTHODIR for nonsymmetric problems [20, 23]. In standard implementations, x^k is computed as in (1), but r^k is set to $r^{k-1} - a_{k-1}q^{k-1}$, where q^{k-1} is a vector that is equal to Ap^{k-1} in exact arithmetic but might differ from this in finite precision arithmetic. The vector q^{k-1} could be explicitly set to Ap^{k-1}, but this would require an extra matrix–vector multiplication at each iteration.

• QMR for nonsymmetric problems [11]. Like BCG, this method uses two sets of recurrences, one involving A and another involving A^T, and like BCG, it can break down if the underlying two-sided Lanczos process breaks down. Such breakdowns can be avoided by using look-ahead steps. A number of different QMR implementations have been proposed [11, 12]. The one given in [2] updates x^k as $x^k = x^{k-1} + d^{k-1}$, but sets $r^k = r^{k-1} - s^{k-1}$, where s^{k-1} is a vector that would be equal to Ad^{k-1} in exact arithmetic but might differ from this in finite precision arithmetic.

• Bi-CGSTAB for nonsymmetric problems [19]. This is another modification of BCG designed to smooth the erratic convergence behavior of BCG. The implementation given in [2] updates x^k from x^{k-1} and two different vectors, while setting r^k to r^{k-1} minus A times the appropriate linear combination of these two vectors. The analysis of recurrences of this form should be very similar to that of (1).

In section 2 we consider the implementation of formulas (1) in finite precision arithmetic. A bound is given on the difference between the true residuals $b - Ax^k$ and the updated vectors r^k. Specifically, it is shown that

$$\frac{\|b - Ax^k - r^k\|}{\|A\|\ \|x\|} \le \epsilon\ O(k)\ (1 + \max_{j\le k} \|x^j\|/\|x\|), \tag{2}$$

where ϵ is the machine precision. It is argued that the last term on the right-hand side of (2)—the maximum norm of an iterate divided by the norm of the true solution—plays an important role in determining the size of the quantity on the left-hand side of (2).

It is often observed numerically, and in some cases can be proved, that the updated vectors r^k converge to zero as $k \to \infty$ or, at least, that their norms become many orders of magnitude smaller than the machine precision. For certain algorithms, such as the steepest descent method and some implementations of the CG method, this can be proved under reasonable assumptions about the condition number of the matrix just by considering the effect of individual steps [3, 21, 22]. Roughly, this is because the coefficients are chosen to minimize some norm of r^k at each step. We will not attempt to prove this here but will demonstrate numerically that the updated approximate residual vectors often become much smaller than the machine precision. In such cases, the right-hand side of (2) gives a reasonable estimate of the best attainable actual residual. Note that this analysis does *not* deal with the rate of convergence of iterative methods in finite precision arithmetic but only with the level of accuracy attainable if the iteration is carried out for sufficiently many steps and assuming that the vectors r^k become tiny.

The size of the maximum iterate divided by the norm of the true solution in (2) can be estimated for various algorithms, assuming exact arithmetic, and these estimates often hold in finite precision arithmetic as well. Using this combination of rigorous finite precision analysis of formulas (1) and exact arithmetic estimates of the size of the iterates, we consider specific algorithms and give numerical examples in section 3. For methods in which the 2-norm of the error decreases monotonically, such as the (unpreconditioned) steepest descent method and the (unpreconditioned) CG method for symmetric positive definite problems and the CGNE and (unpreconditioned) CGNR methods for nonsymmetric problems, the last term in (2) is bounded by $2 + \|x^0\|/\|x\|$. Hence, if the number of steps required to reduce the norm of r^k below $O(\epsilon)\|A\|\|x\|$ is not too large, then these algorithms will return an approximate solution for which the relative residual is of order ϵ. The relative error, $\|x - x^k\|/\|x\|$, is bounded by $\kappa(A)\epsilon$, where $\kappa(\cdot)$ is the condition number. For methods in which the 2-norm of the error may grow, but some other norm, say, the B-norm of the error, decreases monotonically, we can show only that the last factor in (2) is bounded by $1+\kappa^{1/2}(B)(1+\|x^0\|/\|x\|)$, suggesting a relative residual of order $\kappa^{1/2}(B)\epsilon$ and a relative error of order $\kappa(A)\kappa^{1/2}(B)\epsilon$. For BCG and CGS and other methods that do not necessarily reduce any standard error norm, the last factor in (2) cannot be bounded a priori. Such methods may occasionally fail to generate an accurate approximate solution for even a well-conditioned problem, although the updated vectors r^k may become tiny. An example of this is given.

Several of the previously listed algorithms have been analyzed by others. Higham and Knight [15] analyzed the effects of finite precision arithmetic on stationary iterative methods when the residuals are computed directly instead of updated. Wozniakowski [22] considered the steepest descent method and a special version of the CG algorithm, again with directly computed residuals, and gave bounds on the ultimately attainable accuracy in finite precision arithmetic. We will argue later that for the CG algorithm it is better (in terms of rate of convergence) to use the update formula in (1) than to compute residuals directly. The work most closely related to our own is that of Bollen [3], who also analyzed the effects of finite precision arithmetic on iterative methods having a form similar to (1). Bollen showed, under certain assumptions, that the vectors r^k converge at least linearly to a small value. Finally, results similar to (2) have been observed numerically by van der Vorst [19]. Van der Vorst noted that an increase in the 2-norm of the residual at intermediate steps leads to a corresponding increase in the size of the final residual. Inequality (2) shows that it is not really the size of intermediate residuals that is of importance but the size of the iterates. We give an example in which the residual remains small but intermediate iterates grow, causing a loss of accuracy in the final solution.

It should also be noted that the quantity $\|b - Ax^k\|/\|b\|$, which is often the value actually monitored, may be much larger than $\|b - Ax^k\|/(\|A\| \, \|x\|)$ if $\|b\| << \|A\| \, \|x\|$. Our results deal only with the latter expression, which we refer to as the relative residual norm. While the relative residual norm may be small even though the size of an iterate is large, the reverse cannot occur:

$$\frac{\|b - Ax^k\|}{\|A\| \, \|x\|} \leq \frac{\|b\| + \|A\| \, \|x^k\|}{\|A\| \, \|x\|} \leq 1 + \frac{\|x^k\|}{\|x\|}.$$

2. Finite precision implementation of formulas (1). We assume the following model of floating point arithmetic on a machine with unit roundoff ϵ:

$$\text{fl}(a \pm b) = a(1 + \epsilon_1) \pm b(1 + \epsilon_2), \quad |\epsilon_1|, |\epsilon_2| \leq \epsilon, \tag{3}$$

$$\text{fl}(a \text{ op } b) = (a \text{ op } b)(1 + \epsilon_3), \quad |\epsilon_3| \leq \epsilon, \quad \text{op} = *, /. \tag{4}$$

This model is valid even for machines that do not use a guard digit in addition and subtraction.

Under this model, we have the following standard results for operations involving n-vectors v and w and a number a:

$$\|av - \text{fl}(av)\| \leq \epsilon \, \|av\|, \tag{5}$$

$$\|v + w - \text{fl}(v + w)\| \leq \epsilon \, (\|v\| + \|w\|), \tag{6}$$

$$|\langle v, w \rangle - \text{fl}(\langle v, w \rangle)| \leq n \, (\epsilon + O(\epsilon^2)) \, \|v\| \, \|w\|. \tag{7}$$

With these rules, we now consider the implementation of formulas (1) in finite precision arithmetic. Throughout this section, x^k, r^k, a_{k-1}, and p^{k-1} will always denote the quantities *actually computed*. This should cause no confusion since we will never have occasion to refer to the quantities that would be generated in exact arithmetic. To keep the exposition as simple as possible, we will express terms involving ϵ^2 or higher powers of ϵ as $O(\epsilon^2)$ when there are similar terms of order ϵ present. When formulas (1) are implemented in finite precision arithmetic, the computed iterates satisfy

$$x^k = x^{k-1} + a_{k-1} p^{k-1} + \xi^k, \tag{8}$$

$$r^k = r^{k-1} - a_{k-1} A p^{k-1} + \eta^k, \tag{9}$$

where

$$\|\xi^k\| \leq \epsilon \, \|x^{k-1}\| + (2\epsilon + \epsilon^2) \, \|a_{k-1} p^{k-1}\| \tag{10}$$

and

$$\|\eta^k\| \leq \epsilon \, \|r^{k-1}\| + (2\epsilon + \epsilon^2) \, \|a_{k-1} A p^{k-1}\| + (1 + \epsilon)^2 \, \|a_{k-1} d^{k-1}\|, \tag{11}$$

where $\mathrm{fl}(Ap^{k-1}) = Ap^{k-1} + d^{k-1}$. Inequalities (10)–(11) follow from a straightforward application of rules (5)–(6). The size of the term d^{k-1} depends on the accuracy of the matrix–vector multiplication routine, and we will assume that d^{k-1} satisfies

$$\|d^{k-1}\| \leq c\ \epsilon\ \|A\|\ \|p^{k-1}\|. \tag{12}$$

It can be shown from estimate (7) that if A is an n by n matrix with at most m nonzeros in any row and if the matrix–vector product is computed in the standard way, then $c = mn^{1/2}$. We will not make this assumption here, however, because sometimes the matrix A is not stored explicitly and different procedures are used to compute the matrix–vector product.

Multiplying equation (8) by A and subtracting from b gives a recurrence for the true residual $b - Ax^k$. Subtracting from this the recurrence (9) for r^k, we find

$$\begin{aligned} b - Ax^k - r^k &= (b - Ax^{k-1} - r^{k-1}) - A\xi^k - \eta^k \\ &= (b - Ax^0 - r^0) - \sum_{j=1}^{k} (A\xi^j + \eta^j). \end{aligned}$$

Taking norms on both sides and dividing by $\|A\|\ \|x\|$ gives

$$\frac{\|b - Ax^k - r^k\|}{\|A\|\ \|x\|} \leq \frac{\|b - Ax^0 - r^0\|}{\|A\|\ \|x\|} + \sum_{j=1}^{k} \left(\frac{\|\xi^j\|}{\|x\|} + \frac{\|\eta^j\|}{\|A\|\ \|x\|} \right). \tag{13}$$

Equality can hold in (13) for certain vectors ξ^j and η^j whose norms satisfy (10)–(11).

Since the initial vector r^0 is computed directly, the first term on the right-hand side of (13) is easily bounded using rule (6) and expression (12) for the accuracy of the matrix–vector multiplication routine:

$$\|b - Ax^0 - r^0\| \leq \epsilon\ ((1 + c)\ \|A\|\ \|x^0\| + \|b\|) + c\ \epsilon^2\ \|A\|\ \|x^0\|,$$

and since $\|b\| \leq \|A\|\ \|x\|$, we can write

$$\frac{\|b - Ax^0 - r^0\|}{\|A\|\ \|x\|} \leq \epsilon(1 + c)\frac{\|x^0\|}{\|x\|} + \epsilon + c\epsilon^2 \frac{\|x^0\|}{\|x\|}. \tag{14}$$

The following lemma bounds the other terms in (13).

LEMMA 2.1. *Define*

$$\Theta_k \equiv \max_{j \leq k} \|x^j\| / \|x\|. \tag{15}$$

Assume that $1 - 2\epsilon - \epsilon^2 > 0$. *Then the terms on the right-hand side of* (13) *satisfy*

$$\sum_{j=1}^{k} \frac{\|\xi^j\|}{\|x\|} \leq (5\ \epsilon + O(\epsilon^2))\ k\ \Theta_k, \tag{16}$$

$$\sum_{j=1}^{k} \frac{\|\eta^j\|}{\|A\|\ \|x\|} \leq (\epsilon + O(\epsilon^2))\ k\ (1 + (5 + 2c)\ \Theta_k). \tag{17}$$

Proof. From (8), we can write

$$a_{j-1}p^{j-1} = x^j - x^{j-1} - \xi^j, \tag{18}$$

and substituting this expression into (10) gives

$$\|\xi^j\| \le \epsilon \|x^{j-1}\| + (2\epsilon + \epsilon^2)(\|x^j\| + \|x^{j-1}\| + \|\xi^j\|).$$

Using the assumption $1 - 2\epsilon - \epsilon^2 > 0$, this can be written in the form

$$\|\xi^j\| \le \epsilon\ (3\|x^{j-1}\| + 2\|x^j\|) + O(\epsilon^2)\ (\|x^{j-1}\| + \|x^j\|). \tag{19}$$

From this, (16) follows by bounding the sum on the left in (16) by k times the maximum term.

Using (12), the third term in expression (11) for η_j can be bounded by

$$(1+\epsilon)^2\ \|a_{j-1}d^{j-1}\| \le c\ \epsilon\ (1+\epsilon)^2\ \|A\|\ \|a_{j-1}p^{j-1}\|,$$

and expressing $a_{j-1}p^{j-1}$ as in (18) and using the bound (19) for $\|\xi^j\|$, this becomes

$$(1+\epsilon)^2\ \|a_{j-1}d^{j-1}\| \le c\ (\epsilon + O(\epsilon^2))\ \|A\|\ (\|x^j\| + \|x^{j-1}\|). \tag{20}$$

It also follows from (8) that

$$a_{j-1}Ap^{j-1} = A(x^j - x^{j-1} - \xi^j),$$

and substituting this expression into the second term in (11) and using the bound (19) for $\|\xi^j\|$, we have

$$(2\epsilon + \epsilon^2)\ \|a_{j-1}Ap^{j-1}\| \le (2\epsilon + O(\epsilon^2))\ \|A\|\ (\|x^j\| + \|x^{j-1}\|). \tag{21}$$

Finally, assume that each term $\|\eta^i\|$, $i = 1, \ldots, j-1$ is bounded by $O(\epsilon)\|A\|(\|x\| + \max_{\ell \le i} \|x^\ell\|)$. It is clear that η^1 satisfies this bound. Since r^{j-1} satisfies

$$r^{j-1} = b - Ax^{j-1} - (b - Ax^0 - r^0) + \sum_{i=1}^{j-1}(A\xi^i + \eta^i),$$

we have, using (14), (16), and the induction hypothesis,

$$\|r^{j-1}\| \le \|A\|\ \|x - x^{j-1}\| + O(\epsilon)\ \|A\|\ (\|x\| + \max_{i \le j-1} \|x^i\|). \tag{22}$$

Substituting (20)–(22) into the bound (11) for $\|\eta^j\|$, we have

$$\begin{aligned} \|\eta^j\| &\le \epsilon\ \|A\|\ \|x - x^{j-1}\| + (2+c)\ \epsilon\ \|A\|\ (\|x^j\| + \|x^{j-1}\|) \\ &\qquad + O(\epsilon^2)\ \|A\|\ (\|x\| + \max_{i \le j} \|x^i\|) \\ &\le (\epsilon + O(\epsilon^2))\ \|A\|\ (\|x\| + (5+2c) \max_{i \le j} \|x^i\|). \end{aligned} \tag{23}$$

This shows that $\|\eta^j\|$ is also bounded by $O(\epsilon)\|A\|(\|x\|+\max_{i\leq j}\|x^i\|)$, so the induction is complete and (23) is proved. Substituting the bound (23) into (17) and replacing the sum by k times the maximum term gives the desired result. $\square$

Substituting the bounds (14)–(17) into (13) gives the following theorem.

THEOREM 2.2. *The difference between the true residual $b-Ax^k$ and the computed vector r^k satisfies*

$$\frac{\|b-Ax^k-r^k\|}{\|A\|\ \|x\|} \leq (\epsilon+O(\epsilon^2))\ [k+1+(1+c+k\ (10+2c))\ \Theta_k], \tag{24}$$

where c is defined by (12) *and Θ_k is defined by* (15).

Note that Theorem 2.2 follows from a simple rounding error analysis of formulas (1). No assumptions are made about the coefficients a_{k-1} or the direction vectors p^{k-1} or about whether the algorithm even converges.

The bounds in Lemma 2.1 are not sharp. In particular, if an algorithm is near convergence, then one can expect the norm of r^{j-1} to be much smaller than $\|A\|(\|x\|+\|x^{j-1}\|)$, so the bound on $\|\eta^j\|$ in Lemma 2.1 may be a large overestimate. Still, this bound is roughly the same size as the bound on $\|\xi^j\|$, so if the bound (16) is realistic, then the bound in Theorem 2.2 will be of the right order of magnitude. Based on formula (8) and the rules for floating point arithmetic, one can expect that

$$\|\xi^j\| \sim \epsilon\ \|x^{j-1}\|,$$

so the most significant roundoff error will occur at the step where $\|x^{j-1}\|$ is largest. We can then expect

$$\frac{\|b-Ax^k-r^k\|}{\|A\|\ \|x\|} \geq \epsilon\ \Theta_k.$$

Thus, while the constant terms and the dependence on k in (24) may be overestimates, one can expect the factor Θ_k to play an important role in the size of the difference between the true and computed residuals.

We will demonstrate later that several of the algorithms listed in section 1 generate vectors r^k that approach zero (or at least something much smaller than the machine precision) as $k\to\infty$. It should be noted that once r^{k-1} has been reduced below a certain level, the approximate solution x^k remains essentially unchanged. This is because the norm of the update term $a_{k-1}p^{k-1}$ in (8) is closely related to the size of r^{k-1}, as can be seen from (9),

$$a_{k-1}p^{k-1} = A^{-1}(r^{k-1}-r^k+\eta^k).$$

It follows that if the vectors r^k, and hence $a_{k-1}p^{k-1}$, are converging to zero and if one runs well past the point where $\|A^{-1}r^{k-1}\|$ reaches $O(\epsilon)\|x^{k-1}\|$, the true residual $b-Ax^k$ will not grow like k, as suggested by the bound (24), but will remain almost constant. Denoting by S the number of steps necessary to reach this steady state, the bound (24) can be replaced by

$$\frac{\|b-Ax^k-r^k\|}{\|A\|\ \|x\|} \leq (\epsilon+O(\epsilon^2))\ [S+1+(1+c+S\ (10+2c))\ \Theta], \tag{25}$$

where

$$\Theta = \max_j \|x^j\|/\|x\|. \tag{26}$$

Note that if the iteration is started with an extremely large initial guess x^0, say, $\|x^0\| = \epsilon^{-1}$ when $\|x\| = 1$, the factor Θ_k in (24) will be large for all k and, in fact, no method of the form (1) will be likely to find a good approximate solution. This is because even the initial residual cannot be computed accurately, and the coefficients and direction vectors are defined in terms of the updated vectors r^k. We will assume from here on that the initial guess is of reasonable size compared to the true solution and that the term Θ_k becomes large only if the algorithm generates an iterate at some step that is much larger than either the true solution or the initial guess.

3. Specific algorithms.

In this section we consider the specific algorithms listed in section 1 and give numerical examples illustrating Theorem 2.2. The numerical tests were performed using MATLAB on a Sparc workstation with machine precision $\epsilon \approx 1.1e-16$. The algorithms were implemented using the formulas given in this section, and no preconditioners were used.

Solid lines in the figures represent actual relative residual norms,

$$\frac{\|b - Ax^k\|}{\|A\|\ \|x\|},$$

while dashed lines show the updated relative residual norms,

$$\frac{\|r^k\|}{\|A\|\ \|x\|}.$$

The value of Θ in (26) is indicated with an asterisk at the step at which the maximum ratio $\|x^j\|/\|x\|$ occurred. In each of the examples the vectors r^k approach zero or something orders of magnitude less than the machine precision as $k \to \infty$. It is therefore expression (25) that determines the ultimately attainable accuracy.

Using exact arithmetic theory, we also derive a priori bounds on the quantity Θ in (26) and argue or demonstrate numerically that the bounds on Θ usually hold in finite precision arithmetic as well. While the result of Theorem 2.2 is independent of any preconditioner, our bounds on Θ may be different when a preconditioner is used, and such differences are noted.

If an algorithm reduces the 2-norm of the error at each step or, more generally, if the iterates satisfy

$$\|x - x^k\| \le \|x - x^0\| \quad \forall k,$$

then we have

$$\|x^k\| \le 2\|x\| + \|x^0\|, \quad \Theta \le 2 + \Theta_0. \tag{27}$$

If the 2-norm of the error can grow, but, say, the B-norm of the error, $\|x - x^k\|_B \equiv \|B^{1/2}(x - x^k)\|$, is reduced at each step, then

$$\|x - x^k\|_B \le \|x - x^0\|_B \Longrightarrow \|x - x^k\| \le \kappa^{1/2}(B)\|x - x^0\|$$

$$\Longrightarrow \|x^k\| \le \|x\| + \kappa^{1/2}(B)(\|x\| + \|x^0\|),$$

and so we have

$$\Theta \le 1 + \kappa^{1/2}(B)(1+\Theta_0). \tag{28}$$

To obtain the best a priori bound on the actual residual, we therefore need to determine an error norm that is always reduced over its initial value in the algorithm and that is as close as possible to the 2-norm of the error.

3.1. Steepest descent and the CG method. The steepest descent and CG algorithms for symmetric positive definite problems are as follows.

```
STEEPEST DESCENT AND CG.

Given an initial guess $x^0$, compute $r^0 = b - Ax^0$, and set $p^0 = r^0$.
For $k = 1, 2, \ldots$,

Compute $x^k = x^{k-1} + a_{k-1}p^{k-1}$, where $a_{k-1} = \frac{\langle r^{k-1}, r^{k-1}\rangle}{\langle p^{k-1}, Ap^{k-1}\rangle}$.

Set $r^k = r^{k-1} - a_{k-1}Ap^{k-1}$.

Set $p^k = r^k + b_{k-1}p^{k-1}$, where
  $b_{k-1} = 0$ for steepest descent, $b_{k-1} = \frac{\langle r^k, r^k\rangle}{\langle r^{k-1}, r^{k-1}\rangle}$ for CG.
```

It is well known that, in exact arithmetic, each step of these algorithms reduces the A-norm of the error. The steepest descent method minimizes the A-norm of the error at step k in the direction of the residual r^k, while CG minimizes the A-norm of the error over all vectors of the form $P_k(A)e^0$, where e^0 is the initial error and P_k is a kth degree polynomial with value 1 at the origin. It therefore follows from (28) that

$$\Theta \le 1 + \kappa^{1/2}(A)(1+\Theta_0), \tag{29}$$

but one can say more. It was shown in [3] for the steepest descent algorithm and in [16] for the CG method that the 2-norm of the error also decreases monotonically. It therefore follows from (27) that

$$\Theta \le 2 + \Theta_0. \tag{30}$$

Unfortunately, the orthogonality properties used to establish the reduction in 2-norm of the error in CG may fail completely in finite precision arithmetic. An analogy developed in [13] and [14], however, enables one to reach a similar conclusion for finite precision computations. There it was shown that the error in the approximate solution x^k generated by a finite precision CG computation for $Ax = b$ is approximately equal to the error in an approximate solution $\bar{x}^k$ generated by the *exact* algorithm applied to a larger problem $\bar{A}\bar{x} = \bar{b}$, with initial guess $\bar{x}^0$ satisfying $\|\bar{x} - \bar{x}^0\| \approx \|x - x^0\|$. The arguments used to establish monotone convergence of the 2-norm of the error can be applied to the corresponding exact CG iterates $\bar{x}^k$ to obtain

$$\|x - x^k\| \approx \|\bar{x} - \bar{x}^k\| < \|\bar{x} - \bar{x}^0\| \approx \|x - x^0\|.$$

It follows that in finite precision arithmetic, under appropriate assumptions about $\kappa(A)$, one can also expect that (30) will hold.

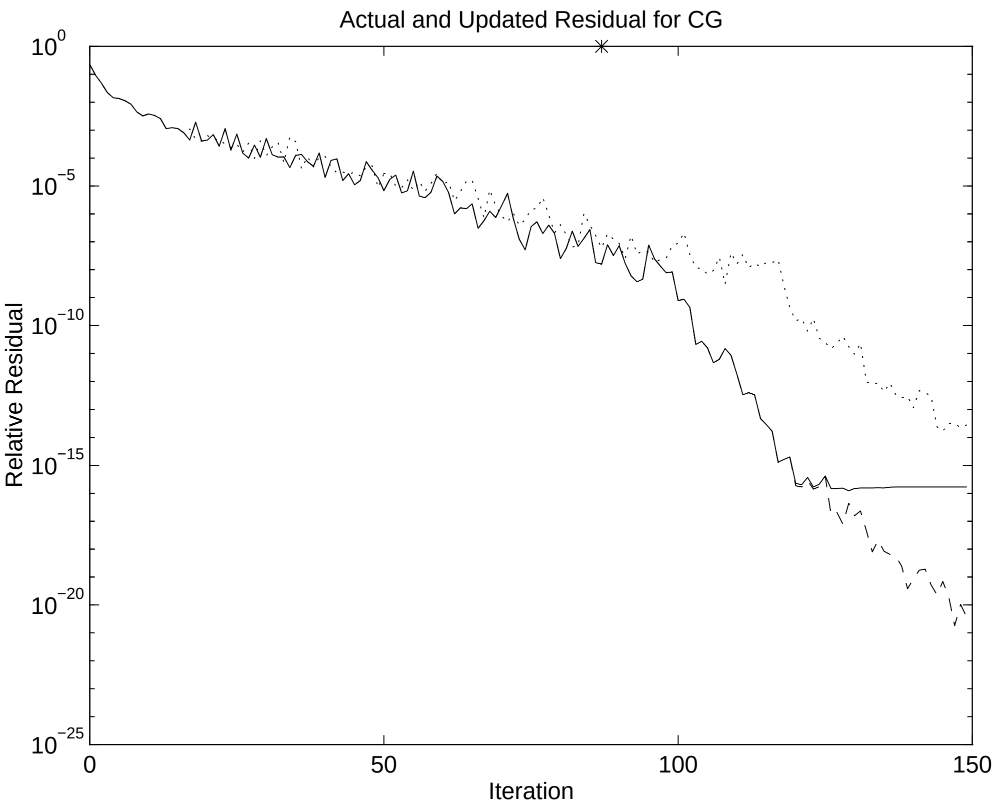

FIG. 1. *Actual residual norm (solid), updated residual norm (dashed), and actual residual norm when residuals are computed directly (dotted).*

We therefore estimate, for both algorithms, that

$$\min_k \frac{\|b - Ax^k\|}{\|A\|\ \|x\|} \le O(\epsilon)\ S \tag{31}$$

independent of κ. The number of steps S required to reach the steady state for an ill-conditioned problem may be quite large, especially for the steepest descent method, but, as noted previously, the factor S in (25) and, therefore, in (31), is usually an overestimate.

Figure 1 shows the convergence of CG for a problem of size $n = 40$ with eigenvalues geometrically distributed between 1 and 10^4:

$$\lambda_i = \kappa^{(i-1)/(n-1)}, \quad i = 1, \ldots, n, \quad \kappa = 10^4. \tag{32}$$

The matrix A was taken to be of the form $Q\Lambda Q^T$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and Q is a random orthogonal matrix. A random solution was chosen and the initial guess was set to zero.

The solid curve shows the actual relative residual norm, while the dashed curve shows the updated relative residual norm. The two curves coincide until they both reach the level of machine ϵ, at which point the dashed curve continues to decrease while the solid curve levels off. The asterisk in the figure shows the maximum ratio $\|x^k\|/\|x\|$, plotted at the step at which it occurred. In exact arithmetic, this ratio is bounded by 1 when a zero initial guess is used, and its value in finite precision arithmetic is just slightly larger than 1 for this example. It is clear that rounding errors have a great effect on the rate of convergence of CG for this problem since, in exact arithmetic, the exact solution would be obtained after $n = 40$ steps. Still, the ultimately attainable accuracy is as high as one might expect, with the relative residual reaching the level ϵ and the relative error of size approximately $\kappa(A)\epsilon$. The behavior of the steepest descent method for this problem is not shown, simply because it was so slow to converge. For other less badly conditioned problems, the steepest descent method also attained a final relative residual of size ϵ.

To demonstrate the claim made in section 1—that it is better (in terms of rate of convergence) to update the vectors r^k in the CG algorithm than to compute them directly—we also ran the above algorithm with the formula for r^k replaced by $r^k = b - Ax^k$. The dotted line in the figure shows the residual norm $\|b - Ax^k\|/(\|A\|\,\|x\|)$ for this modified algorithm. This phenomenon of slower convergence was observed before [22], and it has to do with the fact that the normalized residual vectors $(-1)^k r^k/\|r^k\|$ no longer approximately satisfy a three-term recurrence. That is, all of the known proofs of fast convergence for the CG method (convergence at least about as fast as the exact Chebyshev algorithm) [8, 13] rely on the fact that the recurrence can be written in the form

$$AQ_k = Q_k T_k + \beta_k q_{k+1} e_k^T + F_k, \tag{33}$$

where the columns of the n by k matrix Q_k are the normalized residual vectors $r^0/\|r^0\|, -r^1/\|r^1\|, \ldots, (-1)^{k-1} r^{k-1}/\|r^{k-1}\|$, T_k is a k by k tridiagonal matrix $q_{k+1} = (-1)^k r^k/\|r^k\|$, and F_k is a tiny perturbation term; e.g., $\|F_k\| \sim \epsilon\|A\|$. If residuals are computed directly, then the roundoff term F_k is roughly on the order of $\epsilon\|A\|(\|x^k\|/\|r^k\|)$, so when the residual becomes much smaller than the approximate solution, this term is no longer tiny. Of course this does not *prove* that CG will converge slowly if the residuals are computed directly, but it does explain why the known proofs of fast convergence are not applicable.

When a symmetric positive definite preconditioner M is used with either of these algorithms, it is equivalent to applying the unpreconditioned algorithm to the problem $M^{-1/2}AM^{-1/2}y = M^{-1/2}b$, $x = M^{-1/2}y$. The $M^{-1/2}AM^{-1/2}$-norm of $y - y^k$, which is the A-norm of the error, is still minimized at each step, so the bound (29) on Θ still holds. It is possible, however, that the 2-norm of the error $x - x^k$ may grow. We know only that the 2-norm of $y - y^k$, which is the M-norm of $x - x^k$, decreases monotonically. For a preconditioned problem, estimate (31) must therefore be replaced by

$$\min_k \frac{\|b - Ax^k\|}{\|A\|\|x\|} \le O(\epsilon)\ \min\{\kappa^{1/2}(A), \kappa^{1/2}(M)\}\ S. \tag{34}$$

3.2. The BCG and CGS methods. The BCG algorithm (without look-ahead) for general nonsymmetric problems is as follows.

```
BCG (without look-ahead).

Given an initial guess $x^0$, compute $r^0 = b - Ax^0$, and set $p^0 = r^0$.
Choose $\hat{r}^0$ such that $\langle r^0, \hat{r}^0\rangle \neq 0$, and set $\hat{p}^0 = \hat{r}^0$. For $k = 1, 2, \ldots$,

Compute $x^k = x^{k-1} + a_{k-1}p^{k-1}$, where $a_{k-1} = \frac{\langle r^{k-1}, \hat{r}^{k-1}\rangle}{\langle Ap^{k-1}, \hat{p}^{k-1}\rangle}$.

Set $r^k = r^{k-1} - a_{k-1}Ap^{k-1}$,  $\hat{r}^k = \hat{r}^{k-1} - a_{k-1}A^T\hat{p}^{k-1}$.

Compute $p^k = r^k + b_{k-1}p^{k-1}$,  $\hat{p}^k = \hat{r}^k + b_{k-1}\hat{p}^{k-1}$, where $b_{k-1} = \frac{\langle r^k, \hat{r}^k\rangle}{\langle r^{k-1}, \hat{r}^{k-1}\rangle}$.
```

The BCG algorithm is closely related to the two-sided Lanczos process. For details of this relationship see, for example, [10]. The BCG algorithm breaks down if either

$$\langle Ap^{k-1}, \hat{p}^{k-1}\rangle = 0 \quad \text{or} \quad \langle r^{k-1}, \hat{r}^{k-1}\rangle = 0$$

before an acceptable approximate solution has been obtained. The second condition corresponds to the breakdown of the two-sided Lanczos process, while the first indicates the nonexistence of the BCG iterate. Of course, if either of these quantities is pathologically small, then roundoff is likely to spoil the convergence of the algorithm. In such cases, look-ahead steps can be used to avoid some of the difficulties, at the price of extra work and storage. Even if we assume that these quantities do not become extremely small, however, the norm of the iterate x^k can become arbitrarily large. For this reason, we cannot give an a priori bound on Θ in (26), and, indeed, the algorithm might fail to obtain a small residual even if $\|r^k\| \to 0$.

The CGS algorithm (without look-ahead) for general nonsymmetric problems is as follows.

```
CGS (without look-ahead).

Given an initial guess $x^0$, compute $r^0 = b - Ax^0$, and set $u^0 = r^0$,
$p^0 = r^0$, $q^0 = 0$, and $v^0 = Ap^0$. Choose $\hat{r}^0$ such that $\langle r^0, \hat{r}^0\rangle \neq 0$.
For $k = 1, 2, \ldots$,

Compute $q^k = u^{k-1} - a_{k-1}v^{k-1}$, where $a_{k-1} = \frac{\langle r^{k-1}, \hat{r}^0\rangle}{\langle v^{k-1}, \hat{r}^0\rangle}$.

Compute $x^k = x^{k-1} + a_{k-1}(u^{k-1} + q^k)$.

Set $r^k = r^{k-1} - a_{k-1}A(u^{k-1} + q^k)$.

Compute $u^k = r^k + b_kq^k$, where $b_k = \frac{\langle r^k, \hat{r}^0\rangle}{\langle r^{k-1}, \hat{r}^0\rangle}$.

Set $p^k = u^k + b_k(q^k + b_kp^{k-1})$ and $v^k = Ap^k$.
```

The CGS algorithm requires two matrix–vector multiplications at each step, but no multiplications by the transpose as in BCG. In exact arithmetic, the BCG residual at step k can be written in the form $r_B^k = \phi_k(A)r^0$, where $\phi_k(A)$ is a certain kth

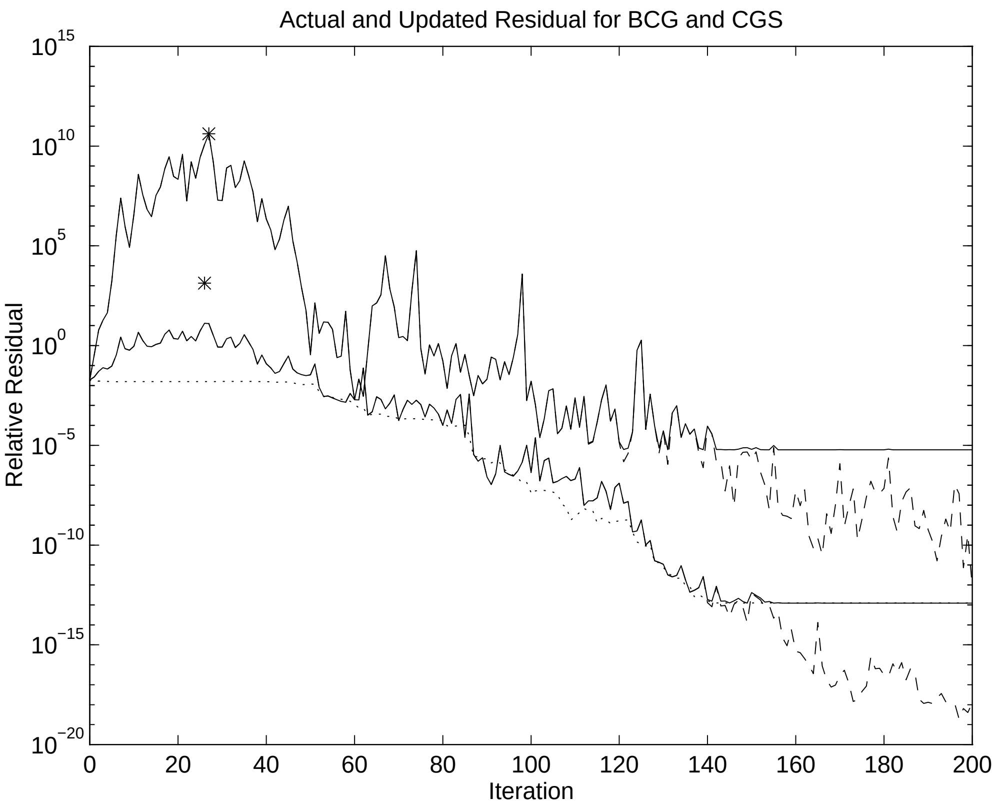

FIG. 2. *Actual residual norm (solid) and updated residual norm (dashed). Top curves are for CGS, bottom ones for BCG. The dotted line shows the actual residual norm for QMR.*

degree polynomial with $\phi_k(0) = 1$. The CGS residual at step k is $r_S^k = \phi_k^2(A)r^0$. If the BCG method is converging well, one would expect the polynomial $\phi_k(A)$ to be small and its square to be even smaller. If the norm of $\phi_k(A)$ is large, however, then that of $\phi_k^2(A)$ will be even larger. Again, the norms of the iterates x^k can become arbitrarily large and we cannot give an a priori bound on Θ in (26).

An example is shown in Figure 2. Here A was taken to be a discretization of the convection diffusion operator

$$-\triangle u + 40(xu_x + yu_y) - 100u \tag{35}$$

on the unit square with Dirichlet boundary conditions, using centered differences on a 32×32 mesh. Similar problems were considered in [11] and [4]. The solution was taken to be $u(x, y) = x(x-1)^2y^2(y-1)^2$ and the initial guess was set to zero. The initial vector $\hat{r}^0$ was set equal to r^0.

The lower solid line in the figure represents the true BCG residual norm, while the lower dashed line shows the updated residual norm. The lower asterisk indicates that the maximum ratio $\|x^k\|/\|x\|$ was approximately 10^3 for BCG. The final actual residual is of size $1.e-13 \approx 10^3\epsilon$, as predicted. Note that in this example the relative residual norm for BCG grows only to about 10^1, so the size of the final residual cannot be predicted from the size of intermediate residuals, but only from the size of intermediate iterates.

The upper solid and dashed lines in Figure 2 show the true and updated residual norms for CGS. The higher asterisk indicates that for CGS the norm of an intermediate iterate grew to approximately $4 \cdot 10^{10}$ times the norm of the true solution, and so, as predicted, the final actual residual reaches the level $6.e-6$, which is roughly $4 \cdot 10^{10}\epsilon$, instead of ϵ. Note that the example presented here was chosen specifically to illustrate the possibility of large growth in iterates and the corresponding loss of accuracy. It should not necessarily be interpreted as showing the typical behavior of the CGS algorithm.

For comparison with BCG, the dotted line in the figure shows the true QMR residual norm for the same problem when the implementation of [2] is used. The QMR convergence curve looks very much like the lower envelope of the BCG convergence curve (as shown in [7] for exact arithmetic), and QMR achieves about the same size final residual as BCG. This cannot be explained using the analysis of this paper, however, because QMR is not of the form (1) and the updated QMR residual vector (not shown) does not converge to zero but levels out at a slightly smaller value than the true QMR residual norm.

3.3. CGNE and CGNR. The CGNE and CGNR algorithms for general nonsymmetric problems are as follows.

CGNE and CGNR.

```
Given an initial guess $x^0$, compute $r^0 = b - Ax^0$, and set $p^0 = A^T r^0$.
For $k = 1, 2, \ldots$,

Compute $x^k = x^{k-1} + a_{k-1} p^{k-1}$, where
  $a_{k-1} = \frac{\langle r^{k-1}, r^{k-1} \rangle}{\langle p^{k-1}, p^{k-1} \rangle}$ for CGNE,  $a_{k-1} = \frac{\langle A^T r^{k-1}, A^T r^{k-1} \rangle}{\langle Ap^{k-1}, Ap^{k-1} \rangle}$ for CGNR.

Set $r^k = r^{k-1} - a_{k-1} A p^{k-1}$.

Compute $p^k = A^T r^k + b_{k-1} p^{k-1}$, where
  $b_{k-1} = \frac{\langle r^k, r^k \rangle}{\langle r^{k-1}, r^{k-1} \rangle}$ for CGNE,  $b_{k-1} = \frac{\langle A^T r^k, A^T r^k \rangle}{\langle A^T r^{k-1}, A^T r^{k-1} \rangle}$ for CGNR.
```

In exact arithmetic, CGNE is equivalent to the CG algorithm for the linear system $AA^T y = b$, $x = A^T y$. If y^k and $\hat{p}^k$ are the iterate and direction vector at step k of the CG algorithm for this problem, then $x^k = A^T y^k$ and $p^k = A^T \hat{p}^k$. Since CG applied to $AA^T y = b$ minimizes the AA^T-norm of $y - y^k$, CGNE minimizes the equivalent quantity, the 2-norm of the error $x - x^k$. It follows that, in exact arithmetic, $\Theta \le 2 + \Theta_0$, and since this estimate does not require the orthogonality of previous residual vectors, it can be expected to hold in finite precision arithmetic as well.

In exact arithmetic, CGNR is equivalent to the CG algorithm for the linear system $A^T A x = A^T b$. The iterates and direction vectors are the same as for the CG algorithm applied to the normal equations, but the vectors r^k are the residuals of the original linear system $b - Ax^k$. Since CG applied to $A^T A x = A^T b$ minimizes the $A^T A$-norm of $x - x^k$, CGNR minimizes the equivalent quantity, the 2-norm of the residual $b - Ax^k$. Since the 2-norm of the residual is reduced at each step, it follows from (28) that $\Theta \le 1 + \kappa(A)(1 + \Theta_0)$. Since the 2-norm of the error is also reduced at each step by the CG algorithm applied to $A^T A x = A^T b$, however, it follows that the same holds for CGNR. As argued previously, the CG method can really be expected to reduce

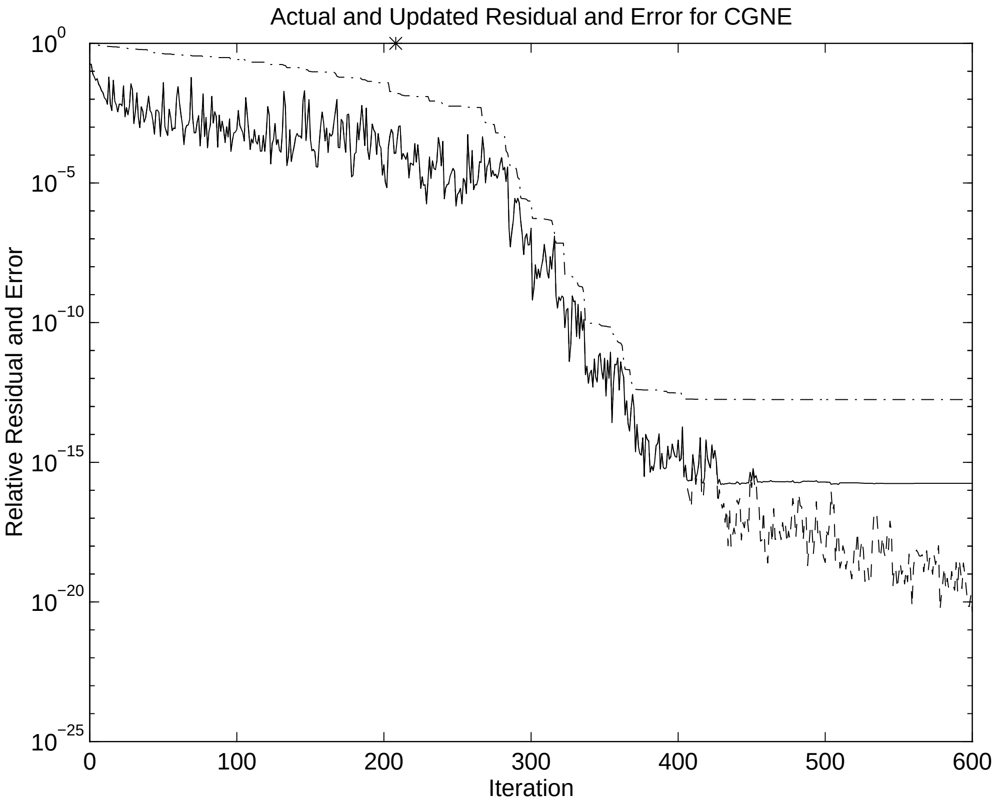

FIG. 3. *Actual residual norm (solid), updated residual norm (dashed), and actual error norm (dash–dot).*

the 2-norm of the error in finite precision arithmetic, based on the analogy developed in [13] and [14]. It therefore follows from (27) that $\Theta \leq 2 + \Theta_0$.

For both algorithms, we therefore estimate that

$$\min_k \frac{\|b - Ax^k\|}{\|A\| \ \|x\|} \leq O(\epsilon) \ S \tag{36}$$

independent of κ. Since this implies that the relative error, $\min_k \|x - x^k\|/\|x\|$, is bounded by $\kappa(A) \ O(\epsilon) \ S$, there is no loss of accuracy associated with the squared condition number of the normal equations (except for what might result from a possibly larger value of S). There may be tighter restrictions on κ, necessary to ensure the convergence of the vectors r^k to zero, but if these restrictions are met, then the accuracy attainable from CGNE and CGNR is as great as that attainable from CG.

Figure 3 shows the actual and updated residual norms as well as the actual error norm, $\|x - x^k\|/\|x\|$, for CGNE applied to a problem with singular values geometrically distributed between 1 and 10^4. The matrix A was taken to be of the form $A = U\Sigma V^T$, where $\Sigma = \text{diag}(\lambda_1, \dots, \lambda_n)$, with $\lambda_1, \dots, \lambda_n$ defined by (32), and U and V random orthogonal matrices. The initial guess was zero and the solution was random. As can be seen from the figure, the relative residual norm reaches the level ϵ, while the relative error norm reaches the level $\kappa(A)\epsilon$. The maximum ratio Θ is approximately 1, as indicated by the asterisk in the figure. Note that the jumps in the residual norm do not cause any loss of accuracy since the iterate norms remain bounded by 1. The actual and updated residual norm curves for CGNR applied to the same problem (not shown) look like smoothed-out versions of those for CGNE. The CGNR algorithm also obtained a final relative residual of size ϵ and a final relative error of size $\kappa(A)\epsilon$.

Similar tests were performed for matrices with singular values geometrically distributed between 1 and $10^1, 10^2, \ldots, 10^8$. Although some of these problems required many thousands of steps to reach the steady state, in all cases the actual relative residual norm for both algorithms leveled out at a value less than 10ϵ.

Since preconditioned CGNE is equivalent to CG applied to the normal equations $AA^Ty = b$, with a preconditioner MM^T, the algorithm still minimizes the AA^T-norm of $y - y^k$, which is the 2-norm of $x - x^k$. Estimate (36) therefore holds for the preconditioned algorithm as well.

When CGNR is preconditioned with a matrix M, it is equivalent to applying CG to the normal equations $A^TAx = A^Tb$ using the preconditioner M^TM. This is equivalent to applying unpreconditioned CG to the system $M^{-T}A^TAM^{-1}y = M^{-T}A^Tb$, $x = M^{-1}y$. The $M^{-T}A^TAM^{-1}$-norm of $y - y^k$, which is the A^TA-norm of $x - x^k$, is minimized over a Krylov space, and so it follows from (28) that $\Theta \leq 1 + \kappa(A)(1 + \Theta_0)$. Since the 2-norm of $y - y^k$, which is the M^TM-norm of $x - x^k$, also decreases monotonically, it also follows from (28) that $\Theta \leq 1 + \kappa(M)(1 + \Theta_0)$. The bound (36) must therefore be replaced by

$$\min_k \frac{\|b - Ax^k\|}{\|A\|\|x\|} \leq O(\epsilon) \ \min\{\kappa(A), \kappa(M)\} \ S \tag{37}$$

for preconditioned CGNR.

4. Alternate implementations.

From the preceding discussion it is clear that if one wishes to use an iterative method in which the norms of intermediate iterates can grow without bound, then one should avoid the use of the update formulas in (1) if the iterate norms become too large. A number of algorithms have been developed to do this. For symmetric indefinite problems, the SYMMLQ method of Paige and Saunders [17] accomplishes this goal by updating certain well-determined intermediate quantities instead of the possibly ill-determined iterates x^k. The composite step biconjugate gradient method of Bank and Chan [1] avoids large iterates by skipping steps at which they appear.

The other possibility for avoiding large iterates is to use methods of the form (1) that reduce some error norm that is not too different from the 2-norm. The ORTHOMIN and ORTHODIR methods, for example, minimize the 2-norm of the residual at each step, so that *if* they were implemented in the form (1) (which would require an extra matrix–vector multiplication at each step) then one would expect to obtain a final relative residual of size $\kappa(A)\ O(\epsilon)\ S$ in cases where the updated vectors r^k converge to zero. The QMR method minimizes the norm of a quantity that differs from that of the residual by no more than a factor $\sqrt{k+1}$ at step k, so that *if* QMR were implemented in the form (1) and assuming that the updated vectors r^k became much smaller than the machine precision, one would obtain a final residual of size less than $\kappa(A)\ O(\epsilon)\ S^{3/2}$.

Finally, while it is important to be aware of the ultimately attainable accuracy of an iterative method, this is often less important in practice than the convergence rate of the method before the ultimately attainable accuracy is achieved. Unfortunately, the analysis of convergence rates in finite precision arithmetic (and sometimes even in exact arithmetic) may be a much more difficult problem.

REFERENCES

[1] R. E. Bank and T. F. Chan, *A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems*, Numer. Algorithms, 7 (1994), pp. 1–16 .

[2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 1993.

[3] J. A. M. Bollen, *Numerical stability of descent methods for solving linear equations*, Numer. Math., 43 (1984), pp. 361–377.

[4] T. F. Chan and T. Szeto, *The composite step family of nonsymmetric conjugate gradient methods*, in PCG '94: Advances in Numerical Methods for Large Sparse Sets of Linear Equations, M. Natori and T. Nodera, eds., Keio University, Yokohama, Japan, 1994, pp. 215–228.

[5] C. Concus, G. H. Golub, and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–332.

[6] E. J. Craig, *The N-step iteration procedures*, J. Math. Phys., 34 (1955), pp. 64–73.

[7] J. Cullum and A. Greenbaum, *Relations between Galerkin and norm-minimizing iterative methods for solving linear systems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 223–247.

[8] V. Druskin and L. Knizhnerman, *Error bounds in the simple Lanczos procedure for computing functions of symmetric matrices and eigenvalues*, Comput. Math. Math. Phys., 31 (1991), pp. 20–30.

[9] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis Dundee 1975, G. A. Watson, ed., Lecture Notes in Mathematics 506, Springer-Verlag, Berlin, 1976, pp. 73–89.

[10] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica, 1 (1992), pp. 57–100.

[11] R. W. Freund and N. M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[12] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[13] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.

[14] A. Greenbaum and Z. Strakos, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.

[15] N. J. Higham and P. A. Knight, *Componentwise error analysis for stationary iterative methods*, in Linear Algebra, Markov Chains, and Queueing Models, C. D. Meyer and R. J. Plemmons, eds., IMA Volumes in Mathematics and its Applications 48, Springer-Verlag, New York, 1993, pp. 29–46.

[16] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–435.

[17] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 11 (1974), pp. 197–209.

[18] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.

[19] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comput., 13 (1992), pp. 631–644.

[20] P. K. W. Vinsome, *Orthomin, an iterative method for solving sparse sets of linear equations*, in Proc. Fourth Symposium of Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149–159.

[21] H. Wozniakowski, *Round-off error analysis of iterations for large linear systems*, Numer. Math., 30 (1978), pp. 301–314.

[22] H. Wozniakowski, *Round-off error analysis of a new class of conjugate gradient algorithms*, Linear Algebra Appl., 29 (1980), pp. 507–529.

[23] D. M. Young and K. C. Jea, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.