Iterative Refinement in Floating Point

CLEVE B. MOLER*

Swiss Federal Institute of Technology, Zurich, Switzerland

ABSTRACT. Iterative refinement reduces the roundoff errors in the computed solution to a system of linear equations. Only one step requires higher precision arithmetic. If sufficiently high precision is used, the final result is shown to be very accurate.

Introduction

Iterative refinement is a process described by Wilkinson [1] for reducing the roundoff error in the computed solution x_1 to an n-by-n system of linear equations $Ax = b$. For $m = 1, 2, \cdots$, the mth iteration involves three steps:

1. Compute the residuals, $r_m = b - Ax_m$,
2. Solve the system, $Ad_m = r_m$,
3. Add the correction, $x_{m+1} = x_m + d_m$.

If there were no roundoff errors in any of these steps, the process would converge to the correct solution in one iteration. Wilkinson has analyzed the convergence in the presence of roundoff errors from a certain type of scaled fixed point arithmetic. The use of floating point arithmetic introduces additional factors which influence the convergence. It is the purpose of this paper to extend Wilkinson's analysis to include these effects.

Any appropriate method with reasonable accuracy can be used at step 2. Gaussian elimination is the most common unless A has special properties such as positive definiteness or sparseness. However, the feature of the iterative refinement process that makes it practical is that the same matrix is involved at step 2 for all iterations. Thus, the decomposition of A computed during the calculation of x_1 can be saved and used in calculating d_m.

The only critical step is step 1, computing the residuals. It is also here that the primary difference between fixed and floating point arithmetic occurs.

Accumulated Inner Product

We assume that all the operations except those used in calculating the residuals involve normalized floating point arithmetic with t_1 digits of base β, and we assume that an "accumulated floating point inner product" with t_2 digits is used for the residuals. By such an inner product we mean the calculation of the quantity

$$b_i - \sum_{j=1}^{n} a_{ij}x_j ,$$

using multiplications which produce a t_2-digit product from t_1-digit numbers and

This work was supported in part by the U. S. Office of Naval Research.

* Now with the Department of Mathematics, University of Michigan, Ann Arbor, Michigan

additions which retain t_2 digits. The final result is truncated to t_1 digits for subsequent use.

An example of this type of inner product is provided by the IBM 7094 with $\beta = 2$, $t_1 = 27$, $t_2 = 54$. Here a single-precision floating point multiplication followed by a double-precision floating point addition produces the desired result. In fact, this can be accomplished with the FORTRAN IV statements

```
DOUBLE PRECISION SUM
SUM = SUM + A[I, J] * X[J].
```

These same operations should also be possible with the IBM System/360 where $\beta = 16$, $t_1 = 6$, $t_2 = 14$, and where a "short-form" multiply produces a "long-form" result.

However, if these machine operations are not available or if t_1 already represents high-precision arithmetic, then it is necessary to use subroutines to do the t_2-digit arithmetic. Since this may be costly it becomes important to find the smallest value of t_2 necessary to make the process effective.

Roundoff Errors

In the following analysis, we use the l_∞ vector norm

$$\| v \| = \max_i | v_i |.$$

Any other monotonic norm could be used without much change.

When the roundoff errors are included, steps 1–3 can be written

$$r_m = b - Ax_m + c_m, \tag{1}$$

$$A(I + F_m)d_m = r_m, \tag{2}$$

$$x_{m+1} = x_m + d_m + g_m, \tag{3}$$

where r_m, d_m and x_{m+1} are the vectors of floating point numbers actually computed and c_m, F_m and g_m are corrections necessary to make these equations hold exactly. We assume such an F_m exists and later, in (10), make an assumption which implies

$$\| F_m \| < 1,$$

and hence that the matrices $I+F_m$ are nonsingular. Then, if x is defined to be the true solution,

$$x = A^{-1}b, \tag{4}$$

it is possible to show that

$$x_{m+1} - x = (I + F_m)^{-1}F_m(x_m - x) + (I + F_m)^{-1}A^{-1}c_m + g_m. \tag{5}$$

Hence

$$\| x_{m+1} - x \| \leq \frac{\| F_m \|}{1 - \| F_m \|} \| x_m - x \| + \frac{\| A^{-1} \|}{1 - \| F_m \|} \| c_m \| + \| g_m \|. \tag{6}$$

Thus, in order to show that $\| x_{m+1} - x \|$ decreases as m increases, we require estimates for the quantities $\| c_m \|$, $\| F_m \|$ and $\| g_m \|$.

We assume that all the arithmetic operations are chopped, rather than rounded, since this is the more common, particularly if the t_2-digit arithmetic must be done by subroutine. We let ϵ_ν denote the "chopoff level" of t_ν-digit arithmetic, that is

$$\epsilon_\nu = \beta^{1-t_\nu}, \qquad \nu = 1, 2.$$

If rounding is used then ϵ_ν can be halved.

Extending Wilkinson's notation slightly, we denote by $fl_{\mu,\nu}(E)$ the result of evaluating the expression E, which involves t_μ-digit variables, using t_ν-digit arithmetic. For example, $fl_{1,2}(yz)$ is the product of two t_1-digit numbers truncated to t_2 significant digits. It is easy to see that

$$fl_{1,2}(yz) = yz(1 + \delta),$$

where

$$\delta = 0 \quad \text{if} \quad t_2 \geq 2t_1, \quad \text{and} \quad |\delta| \leq \epsilon_2 \quad \text{if} \quad t_2 < 2t_1.$$

With this notation, the residual calculation can be described and analyzed in detail. Temporarily suppressing the iteration index m so that subscripts can denote components, we have for the ith component

$$y_j \equiv fl_{1,2}(a_{ij}\, x_j) = a_{ij}\, x_j(1 + \delta_j), \qquad |\delta_j| \leq \epsilon_2;$$

$$s \equiv fl_{2,2}\left(\sum_{j=1}^{n} y_j\right) = \sum_{j=1}^{n} y_j(1 + \gamma_j), \qquad |\gamma_j| \leq (1 + \epsilon_2)^{n-j+1} - 1$$

$$\leq 1.06(n - j + 1)\epsilon_2,$$

provided $n\epsilon_2 \leq 0.1$, which we therefore assume;

$$p \equiv fl_{2,2}(b_i - s) = b_i(1 + \xi) - s(1 + \eta), \qquad |\xi|, |\eta| \leq \epsilon_2;$$

$$r_i \equiv fl_{2,1}(p) = p(1 + \theta), \qquad |\theta| \leq \epsilon_1.$$

(See Wilkinson [1, pp. 18–19, 24–25].)

Combining the above with (1) leads to

$$c_i = \theta(b_i - \sum a_{ij}x_j) + (1 + \theta)\{b_i\xi - \sum a_{ij}x_j[(1 + \delta_j)(1 + \gamma_j)(1 + \eta) - 1]\},$$

and thus

$$\| c_m \| \leq \epsilon_1 \| b - Ax_m \| + (1 + \epsilon_1)\epsilon_2 \| b \| + 1.06(n + 2)(1 + \epsilon_1)\epsilon_2 \| A \| \| x_m \|.$$

Finally, using $\| b \| \leq \| A \| \| x \|$ and $1.06(1 + \epsilon_1) \leq 1.2$, we obtain

$$\| c_m \| \leq [\epsilon_1 + 1.2(n + 2)\epsilon_2] \| A \| \| x_m - x \| + 1.2(n + 3)\epsilon_2 \| A \| \| x \| \quad (7)$$

as the bound for the roundoff error in step 1.

Concerning the error in step 2, $\| F_m \|$, Wilkinson proves (see [1, p. 108]) that if $|a_{ij}| \leq 1$ and if Gaussian elimination with pivoting is used, then the computed d_m satisfies

$$(A + K_m)d_m = r_m,$$

with

$$\| K_m \| \leq \rho\epsilon_1,$$

where the factor ρ depends upon n and upon the magnitude of intermediate results.

In terms of $F_m = A^{-1}K_m$ and an unscaled A this becomes

$$\| F_m \| \leq \rho k(A)\epsilon_1 , \tag{8}$$

where $k(A) = \| A \| \| A^{-1} \|$ is the condition number of A. Wilkinson points out that in practice ρ rarely exceeds n^2 and is often of order unity, especially for badly conditioned matrices. Since we are not concerned with the details of the arithmetic at the point, we simply define ρ to be the smallest value for which (8) holds for all m and state our results in terms of ρ.

For the ith component of the error in step 3 we have

$$\begin{aligned} g_i &= fl_{1,1}(x_i + d_i) - (x_i + d_i) \\ &= \theta(x_i + d_i) \end{aligned}$$

for some θ with $| \theta | \leq \epsilon_1$, and thus

$$\| g_m \| \leq \epsilon_1 \| x_m + d_m \|.$$

When the process converges $\| d_m \| \leq \| x_m \|$; hence

$$\| g_m \| \leq 2\epsilon_1(\| x \| + \| x_m - x \|). \tag{9}$$

Convergence

In order to state our conclusion clearly, we need some further notation. Let

$$\sigma = \frac{\rho + 1 + 1.2(n + 2)\epsilon_2/\epsilon_1 + 2/k(A)}{1 - \rho k(A)\epsilon_1} ,$$

$$\tau_1 = 2,$$

$$\tau_2 = \frac{1.2(1 + 3/n)}{1 - \rho k(A)\epsilon_1} ,$$

$$\mu_1 = \frac{\tau_1}{1 - \sigma k(A)\epsilon_1} ,$$

$$\mu_2 = \frac{\tau_2}{1 - \sigma k(A)\epsilon_1} .$$

As a precise form of the statement "If A is not too badly conditioned," we make the assumption that

$$0 < \sigma k(A)\epsilon_1 < 1. \tag{10}$$

If the process is to be practical, then $\sigma k(A)\epsilon_1$ must be substantially less than 1. This will ensure not only a reasonable rate of convergence, but also that μ_1 and μ_2 are both of order unity.

Now (6) through (9) can be combined to give

$$\| x_{m+1} - x \| \leq \sigma k(A)\epsilon_1 \| x_m - x \| + \tau_1\epsilon_1 \| x \| + \tau_2 nk(A)\epsilon_2 \| x \|.$$

If x_1 is computed in the same way as the subsequent corrections (in effect, using an $x_0 \equiv 0$), then $\| x_1 - x \| \leq \sigma k(A)\epsilon_1 \| x \|$. Hence, by induction,

$$\frac{\| x_m - x \|}{\| x \|} \leq [\sigma k(A)\epsilon_1]^m + \mu_1\epsilon_1 + \mu_2 nk(A)\epsilon_2 . \tag{11}$$

Because of (10) the first term on the right approaches zero as $m \to \infty$. This convergence is linear with a rate which is essentially determined by ρ and the ratio of $k(A)$ to the computer word size β^{t_1}. That is, the rate of convergence is determined by the accuracy in step 2. For example, if $\sigma k(A)\epsilon_1 = 0.1$, then each iteration gains roughly one decimal place.

However, since the elements of x are usually not t_1-digit floating point numbers, we cannot expect convergence in the sense that $\| x_m - x \|$ approaches zero. The final two terms of (11) account for this. The final accuracy attained depends primarily upon the relative size of t_1 and t_2. Errors in step 2 have a significant effect on final accuracy only when there is near equality in (10).

There are three cases to be considered:

(i) $t_2 = t_1$. That is, the residuals are computed without use of an accumulated inner product. In this case, x_m is often no more accurate than x_1. If, however, σ is greater than $\mu_2 n$ then x_2, but not x_3, shows some improvement. This usually happens only with well-conditioned matrices and a relatively inaccurate step 2.

(ii) $t_1 < t_2 < 2t_1$. Here as m increases the eventual dominant term in (11) depends upon the specific matrix. If $\mu_1\epsilon_1$ dominates then the vectors show improvement until the largest components converge to within t_1 significant digits of the corresponding components of x. For worse conditioned matrices, however, $\mu_2 nk(A)\epsilon_2$ may dominate and the limiting accuracy amounts to fewer than t_1 correct digits.

(iii) $t_2 \geq 2t_1$. That is, the residuals are accumulated to at least double precision. The limiting accuracy is then $\mu_1\epsilon_1$ for all matrices satisfying (10). This case is the most common in current practice.

Remarks

The values of μ_1 and μ_2 in practical situations can be illustrated by assuming that at least one decimal place is gained with each iteration, i.e., that $\sigma k(A)\epsilon_1 < 0.1$. Then μ_1 and μ_2 are both less than 3.

If the t_2-digit arithmetic is done by subroutine, it may be advantageous to start the iterations with t_2 equal to, say, $\frac{3}{2}t_1$ and then increase it later if the limiting accuracy is not sufficient. Nothing is lost in the first few iterations, since the value of t_2 does not materially affect the rate of convergence.

The quantity $\log_{10}(\| x_1 \|/\| d_1 \|)$ is usually a good estimate of the number of correct digits in the first computed solution. This, in turn, leads to a rough estimate of the condition of the matrix.

As with all iterative algorithms, an important practical question is when to stop. In our experience with iterative refinement we have almost always had $t_2 = 2t_1$ and have stopped when

$$\| x_m - x_{m-1} \| \leq \epsilon_1 \| x_1 \|. \tag{12}$$

Note, however, that even if (12) is satisfied, without at least a bound for $k(A)$, no precise conclusion can be made regarding $\| x_m - x \|$.

If (12) is not satisfied before m reaches some preassigned limit M, it can be said that A is too badly conditioned to be dealt with adequately using only t_1 digits in step 2. The value chosen for M is somewhat a matter of taste. If M is larger than, say, $t_1 \log_2(\beta)$, then there is danger of wasting machine time or mistakenly accept-

ing inaccurate results, whereas if M is smaller than $t_1 \log_{10}(\beta)$, then some solvable problems may be prematurely rejected. A value near $2t_1 \log_{10}(\beta)$ is probably reasonable.

Wilkinson considered a type of fixed point arithmetic ("block floating") in which the only error in the residual calculation is the final rounding. This roughly corresponds to $\epsilon_2 = 0$ for floating point and the bound (7) would be significantly simplified.

Wilkinson is able to show that "the probability is very high" that x_m actually converges to a vector $\mathbf{x}$ which is the fixed point approximation to x. The situation is quite different in floating point where each component of x_m is represented with t_1 "significant" digits. It may be impossible to compute the last few digits of small components accurately, without retaining more than t_1 digits in the large components. Thus, there is some possibility that these small components may fluctuate wildly from one iteration to the next and never converge.

The relative size of the components of x can be altered by a column scaling of A. Such a scaling can therefore be expected to have an effect on the convergence and ultimate accuracy of various components. Furthermore, if Gaussian elimination is used, a row scaling of A can affect the choice of pivots and hence the value of ρ. These scale factors may be influenced by, among other things, the choice of units in an underlying physical problem. Unfortunately, with floating point arithmetic, the consequences of scaling are not always appreciated or even understood.

Iterative refinement has been implemented on several different computers. The programs in [2] and [3] are written in FORTRAN for the 7090/94, and those in [4, 5, 6] are in ALGOL. The published ALGOL programs cannot include the code for the accumulated inner product.

ACKNOWLEDGMENT. The author wishes to thank the referee for his incisive and stimulating observations.

REFERENCES

1. WILKINSON, J. H. *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, N. J., 1963.
2. KAHAN, W. Writeups of library tape subroutines LEQU, LEQUN, FLEQU, CLEQU and DLEQU. Inst. Computer Sci., U. of Toronto, 1965 (various months).
3. MOLER, C. SOLVE, accurate simultaneous linear equation solver with iterative improvement. SHARE Distribution No. 3194, 1964.
4. MCKEEMAN, W. M. Algorithm 135. Crout with equilibration and iteration. *Comm. ACM 5* (1962), 553–555.
5. MARTIN, R. S., PETERS, G., AND WILKINSON, J. H. Iterative refinement of the solution of a positive definite system of equations. *Numer. Math. 8* (1966), 203–216.
6. BOWDLER, H. J., MARTIN, R. S., PETERS, G., AND WILKINSON, J. H. Solution of real and complex systems of equations. *Numer. Math. 8* (1966), 217–234.

RECEIVED APRIL, 1966; REVISED JULY, 1966