What Is Iterative Refinement?

Nicholas J. Higham*

March 13, 2023

Iterative refinement is a method for improving the quality of an approximate solution $\widehat{x}$ to a linear system of equations $Ax = b$, where A is an $n \times n$ nonsingular matrix. The basic iterative refinement algorithm is very simple.

1. Compute the residual $r = b - A\widehat{x}$.
2. Solve $Ad = r$.
3. Update Monday March 13, 2023$\widehat{x} \leftarrow \widehat{x} + d$.
4. Repeat from step 1 if necessary.

At first sight, this algorithm seems as expensive as solving for the original $\widehat{x}$. However, usually the solver is LU factorization with pivoting, $A = LU$ (where we include the permutations in L). Most of the work is in the LU factorization, which costs $O(n^3)$ flops, and each iteration requires a multiplication with A for the residual and two substitutions to compute d, which total only $O(n^2)$ flops. If the refinement converges quickly then it is inexpensive.

Turning to the error, with a stable LU factorization the initial $\widehat{x}$ computed in floating-point arithmetic of precision u satisfies (omitting constants)

$$\frac{\|x - \widehat{x}\|}{\|x\|} \lesssim \kappa_\infty(A)u, \qquad (1)$$

where $\kappa_\infty(A) = \|A\|_\infty \|A^{-1}\|_\infty \geq 1$ is the matrix condition number in the ∞-norm. We would like refinement to produce a solution accurate to precision u:

$$\frac{\|x - \widehat{x}\|}{\|x\|} \approx u. \qquad (2)$$

But if the solver cannot compute the initial $\widehat{x}$ accurately when A is ill-conditioned why should it be able to produce an update d that improves $\widehat{x}$?

The simplest answer is that when iterative refinement was first used on digital computers the residual r was computed at twice the working precision, which could be done at no extra cost in the hardware. If $\widehat{x}$ is a reasonable approximation then we expect cancellation in forming $r = b - A\widehat{x}$, so using extra precision in forming r ensures that r has enough correct digits to yield a correction d that improves $\widehat{x}$. This form of iterative refinement produces a solution satisfying (2) as long as $\kappa_\infty(A) < u^{-1}$.

Here is a MATLAB example, where the working precision is single and residuals are computed in double precision.

*Department of Mathematics, University of Manchester, Manchester, M13 9PL, UK (nick.higham@manchester.ac.uk).

```
n = 8; A = single(gallery('frank',n)); xact = ones(n,1);
b = A*xact; % b is formed exactly for small n.
x = A\b;
fprintf('Initial_error = %4.1e\n', norm(x - xact,inf))
r = single( double(b) - double(A)*double(x) );
d = A\r;
x = x + d;
fprintf('Second error  = %4.1e\n', norm(x - xact,inf))
```

The output is

```
Initial_error = 9.1e-04
Second error  = 6.0e-08
```

which shows that after just one step the error has been brought down from 10^{-4} to the level of $u_s \approx 5 \times 10^{-8}$, the unit roundoff for IEEE single precision arithmetic.

Fixed Precision Iterative Refinement

By the 1970s, computers had started to lose the ability to cheaply accumulate inner products in extra precision, and extra precision could not be programmed portably in software. It was discovered, though, that even if iterative refinement is run entirely in one precision it can bring benefits when $\kappa_\infty(A) < u^{-1}$. Specifically,

- if the solver is somewhat numerically unstable the instability is cured by the refinement, in that a relative residual satisfying

$$\frac{\|b - A\widehat{x}\|_\infty}{\|A\|_\infty \|\widehat{x}\|_\infty + \|b\|_\infty} \approx u \qquad (3)$$

  is produced, and
- a relative error satisfying

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim \mathrm{cond}(A, x)u \qquad (4)$$

  is produced, where

$$\mathrm{cond}(A, x) = \frac{\| \, |A^{-1}||A||x| \, \|_\infty}{\|x\|_\infty}.$$

The bound (4) is stronger than (1) because $\mathrm{cond}(A, x)$ is no larger than $\kappa_\infty(A)$ and can be much smaller, especially if A has badly scaled rows.

Low Precision Factorization

In the 2000s processors became available in which 32-bit single precision arithmetic ran at twice the speed of 64-bit double precision arithmetic. A new usage of iterative refinement was developed in which the working precision is double precision and a double precision matrix A is factorized in single precision.

1. Factorize $A = LU$ in single precision.
2. Solve $LUx = b$ by substitution in single precision, obtaining $\widehat{x}$.

3. Compute the residual $r = b - A\widehat{x}$ in double precision.
4. Solve $LUd = r$ in single precision.
5. Update $\widehat{x} \leftarrow \widehat{x} + d$ in double precision.
6. Repeat from step 3 if necessary.

Since most of the work is in the single precision factorization, this algorithm is potentially twice as fast as solving $Ax = b$ entirely in double precision arithmetic. The algorithm achieves the same limiting accuracy (4) and limiting residual (3) provided that $\kappa_\infty(A) < u_s^{-1} \approx 2 \times 10^7$.

GRMES-IR

A way to weaken this restriction on $\kappa_\infty(A)$ is to use a different solver on step 4: solve

$$\widetilde{A}d := U^{-1}L^{-1}Ad = U^{-1}L^{-1}r$$

by GMRES (a Krylov subspace iterative method) in double precision. Within GMRES, $\widetilde{A}d$ is not formed but is applied to a vector as a multiplication with A followed by substitutions with L and U. As long as GMRES converges quickly for this preconditioned system the speed again from the fast single precision factorization will not be lost. Moreover, this different step 4 results in convergence for $\kappa_\infty(A)$ a couple of orders magnitude larger than before, and if residuals are computed in quadruple precision then a limiting accuracy (2) is achieved.

This GMRES-based iterative refinement becomes particularly advantageous when the fast half precision arithmetic now available in hardware is used within the LU factorization, and one can use three or more precisions in the algorithm in order to balance speed, accuracy, and the range of problems that can be solved.

Finally, we note that iterative refinement can also be applied to least squares problems, eigenvalue problems, and the singular value decomposition. See Higham and Mary (2022) for details and references.

References

We give five references, which contain links to the earlier literature.

- Erin Carson and Nicholas J. Higham. Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions. SIAM J. Sci. Comput., 40(2):A817–A847,2018.
- Azzam Haidar, Harun Bayraktar, Stanimire Tomov, Jack Dongarra, and Nicholas J. Higham. Mixed-Precision Iterative Refinement Using Tensor Cores on GPUs to Accelerate Solution Of linear systems. Proc. Roy. Soc. London A, 476(2243):20200110, 2020.
- Nicholas J. Higham and Theo Mary. Mixed Precision Algorithms in Numerical Linear Algebra. Acta Numerica, 31:347–414, 2022.
- Nicholas J. Higham and Dennis Sherwood, How to Boost Your Creativity, SIAM News, 55(5):1, 3, 2022. (Explains how developments in iterative refinement 1948–2022 correspond to asking "how might this be different" about each aspect of the algorithm.)
- Julie Langou, Julien Langou, Piotr Luszczek, Jakub Kurzak, Alfredo Buttari, and Jack Dongarra. Exploiting the Performance of 32 Bit Floating Point Arithmetic in

Obtaining 64 Bit Accuracy (Revisiting Iterative Refinement for Linear Systems). In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, IEEE, November 2006.

Related Blog Posts

- A Multiprecision World (2017)
- Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions (2017)
- What Is an LU Factorization? (2021)
- What Is IEEE Standard Arithmetic? (2020)

This article is part of the "What Is" series, available from `https://nhigham.com/category/what-is` and in PDF form from the GitHub repository `https://github.com/higham/what-is`.