Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 20, No. 3, pp. 461–474, 2024

NEWTON'S METHOD IN THREE PRECISIONS*

C. T. Kelley

To Masao Fukushima on his 75th birthday, with thanks for his many contributions to optimization.

Abstract: We describe a three precision variant of Newton's method for nonlinear equations. We evaluate the nonlinear residual in double precision, store the Jacobian matrix in single precision, and solve the equation for the Newton step with iterative refinement with a factorization in half precision. We analyze the method as an inexact Newton method. This analysis shows that, except for very poorly conditioned Jacobians, the number of nonlinear iterations needed is the same that one would get if one stored and factored the Jacobian in double precision. In many ill-conditioned cases one can use the low precision factorization as a preconditioner for a GMRES iteration. That approach can recover fast convergence of the nonlinear iteration. We present an example to illustrate the results.

Key words: *Newton's method, iterative refinement, mixed-precision algorithms*

Mathematics Subject Classification: *65H10, 65F05, 65F10, 45G10*

1 Introduction

This paper is about using low precision arithmetic in the computation of a Newton step. In many cases the cost of the factorization of the Jacobian is $O(N^3)$, where N is the number of unknowns, and this dominates the cost of the computation. Therefore, computation of the factorization in lower precision offers significant savings in cost.

The new algorithms in this paper are based on prior results from [17, 20]. In that older work we implemented Newton's method for the nonlinear equation in two precisions. We evaluated the nonlinear residual in double precision and stored and factored the Jacobian in a lower precision (either single or half). The resulting algorithm is a nonlinear version of the classic iterative refinement method from linear algebra [10, 22].

In [17] we observed that using half precision, while offering the potential of faster computation, degraded the performance of the nonlinear iteration for ill-conditioned problems. In this paper we suggest a way to address that issue by storing copies of the Jacobian in both single and half precision, factoring the Jacobian in half precision, and solving the equation for the Newton step in single precision using the half precision factorization within iterative refinement (IR). We can make this process more robust by using the GMRES-IR method

*This work was partially supported by US Department of Energy grant DE-NA003967 and National Science Foundation Grant DMS-1906446.

 DOI: https://doi.org/10.61208/pjo-2023-030

of [3, 4], where the low precision factorization is used as a preconditioner for a GMRES iteration for the high precision problem.

We view this as an inexact Newton method [6,15] and the nonlinear convergence analysis will depend on the performance of the underlying linear iterative method, which will be IR or GMRES-IR.

1.1 Notation and basic results

We consider a nonlinear equation

$$\mathbf{F}(\mathbf{x}) = 0 \tag{1.1}$$

for $\mathbf{x} \in \Omega \subset R^N$. We will call $\mathbf{F}$ the residual in this paper. We denote the Jacobian matrix of $\mathbf{F}$ by $\mathbf{F}'$.

We will assume that the standard assumptions [8, 15, 20] hold.

Standard Assumptions

1. Equation 1.1 has a solution $\mathbf{x}^* \in \Omega$.
2. $\mathbf{F}' : \Omega \to R^{N\times N}$ is Lipschitz continuous near $\mathbf{x}^*$ with Lipschitz constant γ.
3. $\mathbf{F}'(\mathbf{x}^*)$ is nonsingular.

We will assume that we are near enough to $\mathbf{x}^*$ so that the Newton iteration

$$\mathbf{x}_+ = \mathbf{x}_c - \mathbf{F}'(\mathbf{x}_c)^{-1}\mathbf{F}(\mathbf{x}_c) \tag{1.2}$$

will converge quadratically to the solution. In (1.2) $\mathbf{F}'$ is the Jacobian matrix and, as is standard, $\mathbf{x}_c$ denotes the current point and $\mathbf{x}_+$ denotes the Newton iteration from $\mathbf{x}_c$.

One explicit way [15] to express this is to assume that

$$\mathbf{x}_c \in \mathcal{B} \equiv \{\mathbf{x} \,|\, \|\mathbf{x} - \mathbf{x}^*\| \le \rho\} \tag{1.3}$$

where

$$\rho \le \frac{1}{2\gamma\|\mathbf{F}'(\mathbf{x}^*)^{-1}\|}$$

and is small enough so that $\mathcal{B} \subset \Omega$. In that case $\mathbf{x}_+ \in \mathcal{B}$ and

$$\|\mathbf{e}_+\| \le \gamma\|\mathbf{F}'(\mathbf{x}^*)^{-1}\|\|\mathbf{e}_c\|^2 \le \|\mathbf{e}_c\|/2 \le \rho/2.$$

And so the iteration converges and remains in Ω. Here $\mathbf{e} = \mathbf{x} - \mathbf{x}^*$ denotes the error.

In practice, however, the Newton iteration is computed in floating point arithmetic and the floating point errors must be considered. To account for this (see [15] for the details) we let Δ denote the error in the Jacobian and ϵ denote the error in the residual. With this in mind the iteration is

$$\mathbf{x}_+ = \mathbf{x}_c - (\mathbf{F}'(\mathbf{x}_c) + \Delta(\mathbf{x}_c))^{-1}(\mathbf{F}(\mathbf{x}_c) + \epsilon(\mathbf{x}_c)) \tag{1.4}$$

In this paper we will assume that the errors can be bounded independently of $\mathbf{x}$, so there are ϵ_F and ϵ_J such that

$$\|\epsilon(\mathbf{x})\| \le \epsilon_F \text{ and } \|\Delta(\mathbf{x})\| \le \epsilon_J$$

for all $\mathbf{x} \in \mathcal{B}$. One can think of ϵ_F as floating point roundoff. The interesting part is ϵ_J, the error in the Jacobian.

With this in mind, the error estimate from [15] becomes

$$\|\mathbf{e}_+\| = O(\|\mathbf{e}_c\|^2 + \epsilon_J\|\mathbf{e}_c\| + \epsilon_F), \tag{1.5}$$

where $\mathbf{e} = \mathbf{x} - \mathbf{x}^*$ denotes the error, Clearly, if the errors vanish, one obtains the standard quadratic convergence theory. However, if $\epsilon_F > 0$ then one can expect the residual norms to stagnate once

$$\|\mathbf{F}(\mathbf{x})\| = O(\epsilon_F)$$

which is what one observes in practice. The estimate (1.5) is not a local convergence result. Results of this type are called *local improvement* [9,15].

We can also see that if $\epsilon_J = O(\sqrt{\epsilon_F})$, as it will be [15] if one uses a finite-difference approximation to the Jacobian with difference increment $\sqrt{\epsilon_F}$ or (assuming one computes $\mathbf{F}$ in double precision) stores and factors the Jacobian in single precision, then (1.5) becomes

$$\|\mathbf{e}_+\| = O(\|\mathbf{e}_c\|^2 + \sqrt{\epsilon_F}\|\mathbf{e}_c\| + \epsilon_F) = O(\|\mathbf{e}_c\|^2 + \epsilon_F). \tag{1.6}$$

Equation (1.6) says that the iteration with a sufficiently accurate approximate Jacobian is indistinguishable from Newton's method.

With these errors in mind, we can formulate the locally convergent (*i. e.* with no line search) form of Newton's method of interest in this paper. In Algorithm 1.1 τ_a and τ_r are relative and absolute error tolerances. $\mathbf{x}$ is the initial iterate on input and the algorithm overwrites $\mathbf{x}$ as the iteration progresses.

Algorithm 1.1. newton$(\mathbf{F}, \mathbf{x}, \tau_a, \tau_r)$
Evaluate $\tilde{\mathbf{F}} = \mathbf{F}(\mathbf{x}) + \epsilon(\mathbf{x})$;
$\tau \leftarrow \tau_r\|\tilde{\mathbf{F}}\| + \tau_a$.
while $\|\tilde{\mathbf{F}}\| > \tau$ **do**
  Solve $(\mathbf{F}'(\mathbf{x}) + \Delta(\mathbf{x}))\mathbf{s} = -\tilde{\mathbf{F}}$
  $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{s}$
  Evaluate $\tilde{\mathbf{F}} = \mathbf{F}(\mathbf{x}) + \epsilon(\mathbf{x})$;
end while

There are two sources of error in the Jacobian that contribute to Δ. One is the error in approximating the Jacobian itself and the other is the error in the solver. We will use an analytic Jacobian in this paper and store that Jacobian in single precision. Hence the relative error in the Jacobian is floating point roundoff in single precision. We used Gaussian elimination to solve for the Newton step in [17] so the solver error was the backward error in the LU factorization. We will discuss this more in § 1.3.

1.2 IEEE Arithmetic

We remind the reader of some details of IEEE floating point arithmetic [14,21]. The standard precisions in most software environments are single and double precision. Half precision was originally proposed as a storage format [13] and is not implemented in hardware on many platforms. We will describe the details of these three precisions in terms of the amount of storage a floating point number requires (the width) and the unit roundoff u. As is standard [10] we define u in terms of the floating point error in rounding the result of any binary operation $\circ = \pm, \times, \div$ applied to two floating point numbers x and y

$$fl(x \circ y) = (x \circ y)(1 + \delta),\ |\delta| \leq u.$$

Here fl is the rounding map which takes a real z in the range of the floating point number system to the nearest floating point number. If z is not in the range of the floating point number system, then attempting to compute $fl(z)$ will generate an exception. The range will be important in this paper because one must pay particular attention to that when computing a Newton step with a half-precision Jacobian. We define the range of the floating point number system as

$$\mathcal{R} = \{z \mid \sigma_L \leq |z| \leq \sigma_H\}$$

where σ_L is the smallest positive floating point number and σ_H is the largest positive floating point number.

We can now summarize the properties of the three precisions in this paper. We took the data in Table 1 from a similar table in [11].

Table 1: IEEE precisions

Precision	width (bits)	u	σ_L	σ_H
Half	16	$\approx 5 \times 10^{-4}$	10^{-5}	10^{5}
Single	32	$\approx 6 \times 10^{-8}$	10^{-38}	10^{38}
Double	64	$\approx 10^{-16}$	10^{-308}	10^{308}

When we discuss multiprecision computations we will let u_d, u_s, u_h be unit roundoff in double, single, or half precision.

Double and single precisions have been supported in hardware for decades. Recently new computer architectures such as the Apple M1 and M2 chips have been offering hardware support for half precision. However, tools such as LAPACK and the BLAS [1] do not support half precision yet and run far more slowly in half precision than they do in double precision. There is active research on extending the BLAS and LAPACK to use half precision [7]. The algorithm we propose in this paper will exploit half precision well once the tools catch up.

1.3 Newton's method in two precisions

With the background from the previous sections in hand, we can now describe the findings from [17] and then motivate the three precision algorithms.

As we said above, the equation for the Newton step

$$\mathbf{F}'(\mathbf{x}_c)\mathbf{s} = -\mathbf{F}(\mathbf{x}_c)$$

can only be approximated. The first step in such an approximation is to replace $\mathbf{F}'(\mathbf{x}_c)$ with an approximation $\mathbf{J}$. For example $\mathbf{J}$ could be a floating point evaluation of the Jacobian, perhaps in a lower precision that the one used to evaluate $\mathbf{F}$, a finite difference approximation, or a physics-based approximation that neglects part of the Jacobian. In any case, one can analyze the error in $\mathbf{J}$ directly.

In this work we solve the approximation

$$\mathbf{J}\mathbf{s} = -\mathbf{F}(\mathbf{x}_c)$$

with Gaussian elimination [10], *i. e.* an LU factorization. We compute an upper triangular matrix $\mathbf{U}$ and a lower triangular matrix $\mathbf{L}$ that, in exact arithmetic, factors $\mathbf{J} = \mathbf{L}\mathbf{U}$, so the equation for the step can be solved by two triangular solves.

However, there are errors in factorization and one really computes approximations $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$. We define $\hat{\mathbf{J}} = \hat{\mathbf{L}}\hat{\mathbf{U}}$, so the approximate factorization is the exact factorization for a different (hopefully nearby) problem. Hence, the equation we actually solve for the Newton step is

$$\hat{\mathbf{J}}\mathbf{s} = -\mathbf{F}(\mathbf{x}_c).$$

There is a subtle point in the equation for the Newton step. The matrix $\hat{\mathbf{J}}$ may be in a different precision than $\mathbf{F}$ and $\mathbf{s}$. One must take some care with this and we return to this point in § 2.2 and 2.3.

The backward error is

$$\delta\mathbf{J} = \hat{\mathbf{J}} - \mathbf{J}.$$

So the error in the Jacobian (ϵ_J in (1.5)) has has two parts, the error in $\mathbf{J}$ and the backward error in the factorization.

We will assume for this paper that $\mathbf{F}$ is computed in double precision, so ϵ_F is $O(u_d)$. The error one makes is storing the Jacobian in reduced precision is

$$\|\mathbf{J} - \mathbf{F}'(\mathbf{x}_c)\| \le u_J,$$

where $u_J = O(u_s)$ or $O(u_h)$. We will make the contribution from the backward error explicit and reformulate (1.5) as

$$\|\mathbf{e}_+\| = O(\|\mathbf{e}_c\|^2 + (u_J + \|\delta\mathbf{J}\|)\|\mathbf{e}_c\| + \epsilon_F).$$

The results in [17] show that if the Jacobian is stored and factored in single precision and the size N of the problem is not too large, then there is no difference in the iteration statistics from storing and factoring the Jacobian in double precision. So both the approximation error and the backward error in the solver are $O(u_d)$. However, if the Jacobian is stored and factored in half precision, there are differences caused by the poor accuracy of half precision, and the nonlinear iteration can converge slowly or even fail to converge.

The new algorithms in this paper use a half precision factorization as part of an iterative method to compute a Newton step with a single precision Jacobian. This approach, as we explain in § 2, requires some care.

2 Three Precision Algorithms

The three precision algorithms compute $\mathbf{F}$ in double precision and store the Jacobian $\mathbf{F}'$ in a single precision matrix $\mathbf{J}$. This means that

$$\|\mathbf{F}'(\mathbf{x}_c) - \mathbf{J}\| \le u_s\|\mathbf{F}'(\mathbf{x}_c)\|. \tag{2.1}$$

Hence, using the terminology of § 1.1

$$\epsilon_F = O(u_d) \text{ and } \epsilon_J = O(u_s). \tag{2.2}$$

Our notation for interprecision transfers is to let I_a^b be the transfer from precision u_a to u_b. If $u_a > u_b$, this promotion changes nothing

$$I_a^b(x) = x$$

if x is in precision u_a. If $u_a < u_b$, then the interprecision transfer rounds down, so

$$\|I_a^b(x) - x\| \le u_b\|x\|.$$

We will use these properties of interprecision transfer throughout the remainder of the paper. We point out that when one rounds a matrix or vector down to a lower precision, one must allocate memory for the low precision object and that there is a cost to this.

We then round $\mathbf{J}$ to half precision to obtain

$$\mathbf{J}_h = I_s^h(\mathbf{J})$$

and factor $\mathbf{J}_h$ in half precision to obtain $\hat{\mathbf{L}}\hat{\mathbf{U}}$. We use the half precision factorization as part of an iterative method to solve

$$\mathbf{Js} = -\mathbf{F}(\mathbf{x}_c).$$

We terminate that iteration when

$$\|\mathbf{Js} + \mathbf{F}(\mathbf{x}_c)\| \le \eta_J \|\mathbf{F}(\mathbf{x}_c)\|. \tag{2.3}$$

We summarize the three precision algorithm.

Algorithm 2.1. newton3p$(\mathbf{F}, \mathbf{x}, \tau_a, \tau_r, \eta_J)$
Evaluate $\tilde{\mathbf{F}} = \mathbf{F}(\mathbf{x}) + \epsilon(\mathbf{x})$;
$\tau \leftarrow \tau_r \|\tilde{\mathbf{F}}\| + \tau_a$.
while $\|\tilde{\mathbf{F}}\| > \tau$ **do**
  Compute and store $F'(\mathbf{x})$ in single precision as $\mathbf{J}$.
  Store $\mathbf{J}_h = I_s^h(\mathbf{J})$.
  Find $\mathbf{s}$ such that $\|\mathbf{J}_h\mathbf{s} + \mathbf{F}(\mathbf{x})\| \le \eta_J \|\mathbf{F}(\mathbf{x})\|$.
  $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{s}$
  Evaluate $\tilde{\mathbf{F}} = \mathbf{F}(\mathbf{x}) + \epsilon(\mathbf{x})$;
end while

In Algorithm 2.1 we want to choose $\eta_J < 1$ small enough so that the nonlinear iteration statistics are the same as those from Newton's method itself.

Algorithm 2.1 looks like an inexact Newton iteration, but differs in that the condition on the step is (2.3) rather than the classical inexact Newton condition

$$\|\mathbf{F}'(\mathbf{x}_c)\mathbf{s} + \mathbf{F}(\mathbf{x}_c)\| \le \eta \|\mathbf{F}(\mathbf{x}_c)\|. \tag{2.4}$$

If we had (2.4), then we would get a local improvement estimate [6, 15]

$$\|\mathbf{e}_+\| = O(\|\mathbf{e}_c\|^2 + \eta\|\mathbf{e}_c\| + \epsilon_F). \tag{2.5}$$

This will imply q-linear convergence of the nonlinear iteration if η is sufficiently small and the function evaluation is exact ($\epsilon_F = 0$).

If we are able to show that we can chose η_J so that (2.4) holds with $\eta = O(u_s)$, then, similar to the two precision case with $\mathbf{J}$ stored and factored in single precision, (2.5) will imply (1.6) and the nonlinear iteration statistics will be the same as Newton's method with the Jacobian stored and factored in double precision.

The use of an iterative method for the linear equation for the Newton step means that the backward error in the factorization plays no role in the analysis of the nonlinear iteration. However, that backward error does affect the convergence of the linear iteration. We will describe our two choices for the linear iteration in § 2.2 but will discuss the local improvement result for the nonlinear iteration first.

2.1 Local improvement of the nonlinear iteration

We begin by showing that $\mathbf{J}$ is nonsingular and estimating $\|\mathbf{J}^{-1}\|$. In the analysis we use the standard notation

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$$

for the condition number of a matrix $\mathbf{A}$.

Lemma 2.2. *Assume that the standard assumptions* (1.3) *hold and that*

$$4u_s\kappa(\mathbf{F}'(\mathbf{x}^*)) < 1. \tag{2.6}$$

Then $\mathbf{J}$ *is nonsingular and*

$$\|\mathbf{J}^{-1}\| \le \frac{2\|\mathbf{F}'(\mathbf{x}^*)^{-1}\|}{1 - 4u_s\kappa(\mathbf{F}'(\mathbf{x}^*))}. \tag{2.7}$$

Proof. The standard assumptions and (1.3) imply that $\mathbf{F}'(\mathbf{x}_c)$ is nonsingular and (see Lemma 4.3.1 from [15])

$$\|\mathbf{F}'(\mathbf{x}_c)\| \le 2\|\mathbf{F}'(\mathbf{x}^*)\| \text{ and } \|\mathbf{F}'(\mathbf{x}_c)^{-1}\| \le 2\|\mathbf{F}'(\mathbf{x}^*)^{-1}\|. \tag{2.8}$$

Hence, using (2.8),

$$\|I - \mathbf{F}'(\mathbf{x}_c)^{-1}\mathbf{J}\| \le \|\mathbf{F}'(\mathbf{x}_c)^{-1}\|\|\mathbf{F}'(\mathbf{x}_c) - \mathbf{J}\| \le u_s\|\mathbf{F}'(\mathbf{x}_c)^{-1}\|\|\mathbf{F}'(\mathbf{x}_c)\| \le 4u_s\kappa(\mathbf{F}'(\mathbf{x}^*)) < 1.$$

So $\mathbf{F}'(\mathbf{x}_c)^{-1}$ is an approximate inverse of $\mathbf{J}$. Therefore $\mathbf{J}$ is nonsingular and

$$\|\mathbf{J}^{-1}\| \le \frac{\|\mathbf{F}'(\mathbf{x}_c)^{-1}\|}{1 - 4u_s\kappa(\mathbf{F}'(\mathbf{x}^*))} \le \frac{2\|\mathbf{F}'(\mathbf{x}^*)^{-1}\|}{1 - 4u_s\kappa(\mathbf{F}'(\mathbf{x}^*))},$$

proving the lemma.

□

Assume the linear iterative method converges, which is not guaranteed, and that we terminate the linear iteration when (2.3) holds. To prove the local improvement estimate (2.5) we must connect (2.3) to the classic inexact Newton condition (2.4) for some $\eta < 1$. That will then imply the estimate (2.5).

We express the convergence estimates in terms of

$$P^* = \frac{4\|\kappa(\mathbf{F}'(\mathbf{x}^*))\|}{1 - 4u_s\kappa(\mathbf{F}'(\mathbf{x}^*))}. \tag{2.9}$$

Lemma 2.3. *Assume that the assumptions of Lemma 2.2 and* (2.3) *hold, that* $u_sP^* < 1/2$, *and that*

$$\eta_J < 1 - 2u_sP^*.$$

Then (2.4) *holds with*

$$\eta \le \eta_J + (1 + \eta_J)u_sP^* < 1. \tag{2.10}$$

Proof. Equation (2.3) implies that

$$\|\mathbf{J}^{-1}\|^{-1}\|\mathbf{s}\| \le \|\mathbf{J}\mathbf{s}\| \le (1 + \eta_J)\|\mathbf{F}(\mathbf{x}_c)\|$$

and hence, using Lemma 2.2

$$\|\mathbf{s}\| \leq \|\mathbf{J}^{-1}\|(1+\eta_J)\|\mathbf{F}(\mathbf{x}_c)\| \leq \frac{2\|\mathbf{F}'(\mathbf{x}^*)^{-1}\|}{1-4u_s\kappa(\mathbf{F}'(\mathbf{x}^*))}(1+\eta_J)\|\mathbf{F}(\mathbf{x}_c)\|. \tag{2.11}$$

We use (2.3) again to obtain

$$\begin{aligned}\|\mathbf{F}'(\mathbf{x}_c)\mathbf{s}+\mathbf{F}(\mathbf{x}_c)\| &\leq \|\mathbf{J}\mathbf{s}+\mathbf{F}(\mathbf{x}_c)\| + \|(\mathbf{F}'(\mathbf{x}_c)-\mathbf{J})\mathbf{s}\| \\ &\leq \eta_J\|\mathbf{F}(\mathbf{x}_c)\| + u_s\|\mathbf{F}'(\mathbf{x}_c)\|\|\mathbf{s}\| \\ &\leq \eta_J\|\mathbf{F}(\mathbf{x}_c)\| + 2u_s\|\mathbf{F}'(\mathbf{x}^*)\|\|\mathbf{s}\|.\end{aligned} \tag{2.12}$$

Combining (2.11) and (2.12) completes the proof.

□

Now suppose we can obtain $\eta_J = O(u_s) = O(\sqrt{u_d})$, then (1.6) holds and the local improvement estimate becomes

$$\|\mathbf{e}_+\| = O(\|\mathbf{e}_c\|^2 + u_d) \tag{2.13}$$

and the iteration statistics should be the same as Newton's method. We will see exactly this in the results in § 3.

The assumption that $u_s P^* < 1/2$ simply says that $\mathbf{F}'(\mathbf{x}^*)$ is not horribly ill-conditioned. Ill-conditioning of $\mathbf{F}'(\mathbf{x}^*)$ does not appear in the local improvement estimate directly, but does affect the convergence of the linear iteration, as we will see in the next section.

2.2 Iterative refinement

Our first choice for an iterative method will be classic iterative refinement [22] for solving a linear system $\mathbf{A}\mathbf{u} = \mathbf{b}$. Consistently with the application in this paper, we will assume that the linear system is in single precision and that we factor the matrix in half precision. The reader should be aware that one must store a half precision copy of $\mathbf{A}$. The basic algorithm is

Algorithm 2.1. IR$(\mathbf{A}, \mathbf{b}, \mathbf{u})$
$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{u}$
Store $\mathbf{A}_h = I_s^h(\mathbf{A})$
Factor $\mathbf{A}_h$ in half precision to obtain computed factors $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$.
while $\|\mathbf{r}\|$ too large **do**
  $\mathbf{d} = \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{r}$
  $\mathbf{u} \leftarrow \mathbf{u} + \mathbf{d}$
  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{u}$
end while

In Algorithm 2.1 $\mathbf{u}$ is the initial iterate on input and the converged solution on output. Note that we are careful to use notation to stress that we use the computed LU factors in half precision.

One can express the iteration in closed form as

$$\mathbf{u} \leftarrow (\mathbf{I} - \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{A})\mathbf{u} + \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{b}.$$

Hence, Algorithm 2.1 is a linear stationary iterative method with iteration matrix

$$\mathbf{M} = (\mathbf{I} - \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{A}).$$

So, if the half precision factorization is a sufficiently good approximation to $\mathbf{A}$, then $\|\mathbf{M}\|$ will be small and iteration will converge rapidly, at least in exact arithmetic. In the presence of rounding errors we would only expect a local improvement result.

Half precision can be very inaccurate and one must be prepared for the iteration to converge slowly or even diverge. One can show that if the low precision factorization is a reasonably good approximation to $\mathbf{A}$, then one obtains exactly the local improvement results one would like. One such estimate is from [4] using the ℓ^∞ norm on R^N. In the case here, where $u_h^2 = u_s$, one can show convergence if

$$3Nu_h\mathrm{cond}(\mathbf{A}) < 1 \tag{2.14}$$

where $\mathrm{cond}(\mathbf{A}) = \| \, |\mathbf{A}^{-1}| \, |\mathbf{A}| \, \|_\infty$ and $|\mathbf{A}|$ is the matrix with entries $|a_{ij}|$. In that case the iteration will reduce the linear residual by a factor $O(u_h)$ until

$$\|\mathbf{b} - \mathbf{A}\mathbf{u}\| = O(u_s\|\mathbf{b}\| + \|\mathbf{A}\|_\infty\|\mathbf{u}\|_\infty). \tag{2.15}$$

Now we interpret (2.15) in terms of the inexact Newton conditions (2.4) and (2.3). We have $\mathbf{A} = \mathbf{F}'(\mathbf{x}_c)$, $\mathbf{b} = \mathbf{F}(\mathbf{x}_c)$ and the solution $\mathbf{u}$ is the Newton step $\mathbf{s}$. Since $\|\mathbf{s}\| = O(\|\mathbf{F}(\mathbf{x}_c)\|)$, the estimate (2.15) implies (2.3) with $\eta_J = O(u_s)$.

If the matrix $\mathbf{A}$ is poorly conditioned, then (2.14) can fail and then iterative refinement may fail to converge or fail to satisfy (2.15). We will see this for the ill-conditioned example in § 3.

Even if $\|\mathbf{M}\| > 1$, the condition number of

$$\hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{J}$$

may be small enough to motivate using a Krylov method with the low-precision factorization as a preconditioner. We use the GMRES-IR approach from [3, 4] with left preconditioning. This means that we solve

$$\hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{A}\mathbf{d} = \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{r}$$

with GMRES to compute the defect $\mathbf{d}$ in Algorithm 2.1. The preconditioner-vector product is computed with two triangular solves. The results in § 3 show how this approach can improve simple IR in one ill-conditioned case.

2.3 Interprecision Transfers

Finally, we must discuss some important details of interprecision transfers and mixed precision operations.

For the two-precision implementation, when we solve

$$\hat{\mathbf{J}}\mathbf{s} = -\mathbf{F}(\mathbf{x}_c) \tag{2.16}$$

for the Newton step, we need to account for the interprecision transfers. If we do nothing, then the triangular factors are in precision u_J and $\mathbf{F}$ is in double precision. In that case each operation in the triangular solves will promote the low precision matrix elements to double within the CPU registers. This is called "interprecision transfer on the fly".

Interprecision transfer on the fly is $O(N^2)$ work on interprecision transfers, but can be a noticeable cost for medium to low dimensions even though the factorization cost is $O(N^3)$

work. A way to avoid this cost is to round $\mathbf{F}$ to precision u_J before the solve. One must take care if $\mathbf{x}_c$ is near the solution because rounding down, especially in half precision, could result in an underflow to zero [12]. The remedy for this is to scale $\mathbf{F}$ to a unit vector before rounding and then reverse the scaling after the linear solve. With this in mind one solves

$$\hat{\mathbf{J}}\hat{\mathbf{s}} = -I_d^J(\mathbf{F}(\mathbf{x}_c)/\|\mathbf{F}(\mathbf{x}_c)\|) \tag{2.17}$$

entirely in the lower precision. This avoids interprecision transfers during the triangular solves. The one promotes $\hat{\mathbf{s}}$ and reverses the scaling

$$\mathbf{s} = \|\mathbf{F}(\mathbf{x}_c)\| I_J^d \hat{\mathbf{s}} \tag{2.18}$$

to obtain a step $\mathbf{s}$ in precision u_J. Then one would update the solution via

$$\mathbf{x}_+ = \mathbf{x}_c + \mathbf{s}.$$

This is exactly what we do in our Julia codes [17,19]. The reader should know that the steps $\mathbf{s}$ computed with (2.16) and (2.17)-(2.18) are different, but the performance of the nonlinear iteration is unlikely to change.

For the linear iterative refinement iteration, the ideas are similar. Interprecision transfers on the fly are implicit in our discussion in § 2.2 where we view iterative refinement as a stationary iterative method. Just as in the nonlinear case, one can mitigate the interprecision transfer cost by replacing the step

$$\mathbf{d} = \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{r}$$

from Algorithm 2.1 with

$$\mathbf{d} = \|\mathbf{r}\| I_j^s(\hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1} I_s^h(\mathbf{r}/|\ \mathbf{r}\|)).$$

The iteration is no longer a stationary iterative method. Instead the iteration is

$$\mathbf{u} \leftarrow \mathbf{u} + \|\mathbf{b} - \mathbf{A}\mathbf{u}\| I_j^s \left(\hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1} I_s^h \left(\frac{\mathbf{b} - \mathbf{A}\mathbf{u}}{\|\mathbf{b} - \mathbf{A}\mathbf{u}\|} \right) \right). \tag{2.19}$$

The fixed point map is nonlinear and, because of the interprecision transfers, not even continuous. However two approaches to interprecision transfer give the same results for all but the most ill-conditioned problems.

For GMRES-IR, however, using (2.19) will not suffice. One must do the triangular solves in the higher precision, single precision in the case of this paper, and hence assume the interprecision transfer cost. One way to mitigate this cost is to map the half precision factorization of $\mathbf{J}_h$ to single precision before the solve. The cost of this is storage (one more copy of $\mathbf{J}$), but the on-the-fly interprecision cost is avoided.

3 Examples

In this section we compare some of the two precision results from [17] with the three precision method from this paper. Some of the results using half precision were poor because the half precision Jacobian was a poor approximation to the Jacobian. This problem was particularly severe for the ill-conditioned example, which we feature in this section. The example, taken from [17] is the composite mid-point rule discretization of the Chandrasekhar H-equation [5],

$$\mathcal{F}(H)(\mu) = H(\mu) - \left(1 - \frac{c}{2} \int_0^1 \frac{\mu H(\mu)}{\mu + \nu}\, d\nu \right)^{-1} = 0. \tag{3.1}$$

The nonlinear operator $\mathcal{F}$ is defined on $C[0,1]$, the space of continuous functions on $[0,1]$.

We use N quadrature points $\nu_j = (j - 1/2)/N$ for $1 \le j \le N$ and the rule is

$$\int_0^1 f(\nu)\, d\nu \approx \frac{1}{N} \sum_{j=1}^{N} f(\nu_j).$$

The discrete system is

$$\mathbf{F}(\mathbf{x})_i \equiv x_i - \left(1 - \frac{c}{2N} \sum_{j=1}^{N} \frac{x_j \mu_i}{\mu_j + \mu_i} \right)^{-1} = 0. \tag{3.2}$$

As we explained in [16, 17, 20], one can evaluate the nonlinear residual with a fast Fourier transform to in $O(N \log(N))$ work and compute an analytic Jacobian, as we did for this paper, in $O(N^2)$ work. Hence the dominant cost for large N is the factorization of the Jacobian.

For $c = .99$, the results in [17] showed a significant difference in performance for the two-precision algorithm between a low precision of single and one of half (see Figure 3.2, pg 205 and Figure 3.5 pg 208 in [17]). We will reproduce some of those data in this section to make the comparison.

3.1 Computations

The computations in this section were done in Julia [2] v 1.9 on a 2023 Apple Mac Mini with an M2 Pro processor and 32GB of memory. The M2 processor supports half precision computing in hardware and Julia 1.9 offers support for this hardware. However, as we said in the introduction, LAPACK and the BLAS do not take full advantage of the half precision hardware, so half precision computations are slow, but not as slow as the ones the author did for [17].

Our implementation of iterative refinement terminates with success when the relative residual norm is $< 10^{-6} \approx 100 u_s$ and declares that the iteration has failed if the residual norm increases. After failure the algorithm returns the solution of the linear problem with the best residual. This approach allows the nonlinear iteration to continue. We see in the results that failure of the linear iterative refinement iteration can affect the convergence of the nonlinear solver. The reason for this is that the inexact Newton condition can fail in this case and convergence can be slower that expected.

The computations used the author's SIAMFANLEquation.jl Julia package [18–20]. The files for the package are located at `https://github.com/ctkelley/SIAMFANLEquations.jl` and the associated IJulia notebook can be found at `https://github.com/ctkelley/NotebookSIAMFANL`. The GitHub repository `https://github.com/ctkelley/Newton3P` contains the codes used to produce the results in this section and instructions for reproducing those results.

Table 2 presents the residual history for ten iterations from the the ill-conditioned example from [17]. The first three histories are for Newton's method with double precision (F64), single precision (F32), and half precision (F16) Jacobians and are the same results as those from [17]. The final two columns are for IR with the Jacobian stored in single precision and the factorization done in half precision (IR 32-16) and GMRES-IR using the that factorization to precondition GMRES.

The Newton iteration with a half precision Jacobian converged very poorly for this problem. However, IR with the same half precision factorization performs as well as Newton's method using a double precision Jacobian.

Table 2: Residual Histories for Three Precisions: $N = 4096$, $c = .99$

n	F64	F32	F16	IR 32-16	IR-GM
0	1.000e+00	1.000e+00	1.000e+00	1.000e+00	1.000e+00
1	2.289e-01	2.289e-01	5.065e-01	2.289e-01	2.289e-01
2	3.934e-02	3.934e-02	2.958e-01	3.934e-02	3.934e-02
3	2.737e-03	2.737e-03	1.890e-01	2.737e-03	2.737e-03
4	1.767e-05	1.767e-05	1.255e-01	1.767e-05	1.767e-05
5	7.486e-10	7.536e-10	8.518e-02	7.538e-10	7.506e-10
6			6.068e-02		
7			4.240e-02		
8			3.195e-02		
9			2.280e-02		
10			1.713e-02		

Table 3 is a much more poorly conditioned problem. In this problem IR does very well for the first three iterations when the iteration is not close to the solution. As the iteration converges the ill-conditioning of the Jacobian at the solution begins to cause failures of the linear iteration and that affects the nonlinear iteration. GMRES-IR continues to perform well.

Table 3: Residual Histories for Three Precisions: $N = 4096$, $c = .9999$

n	F64	F32	F16	IR 32-16	IR-GM
0	1.000e+00	1.000e+00	1.000e+00	1.000e+00	1.000e+00
1	2.494e-01	2.494e-01	5.182e-01	2.494e-01	2.494e-01
2	6.093e-02	6.093e-02	3.123e-01	6.093e-02	6.093e-02
3	1.480e-02	1.480e-02	2.067e-01	1.480e-02	1.480e-02
4	3.454e-03	3.454e-03	1.421e-01	3.455e-03	3.454e-03
5	6.762e-04	6.762e-04	1.012e-01	6.766e-04	6.762e-04
6	7.049e-05	7.049e-05	7.552e-02	6.360e-04	7.049e-05
7	1.223e-06	1.223e-06	5.773e-02	5.811e-04	1.223e-06
8	3.947e-10	3.957e-10	4.543e-02	5.312e-04	3.952e-10
9			3.639e-02	4.862e-04	
10			2.949e-02	4.456e-04	

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J.D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK Users Guide*, Second Edition, SIAM, Philadelphia, 1992.

[2] J. Bezanson, A. Edelman, S. Karpinski, and V.B. Shah, Julia: A fresh approach to numerical computing, *SIAM Rev.* 59 (2017) 65–98.

[3] E. Carson and N.J. Higham, A new analysis of iterative refinement and its application of accurate solution of ill-conditioned sparse linear systems, *SIAM J. Sci. Comput.* 39 (2017) A2834–A2856,

[4] E. Carson and N.J. Higham, Accelerating the solution of linear systems by iterative refinement in three precisions, *SIAM J. Sci. Comput.* 40 (2018) A817–A847,

[5] S. Chandrasekhar, *Radiative Transfer*, Dover, New York, 1960.

[6] R. Dembo, S. Eisenstat and T. Steihaug, Inexact Newton methods, *SIAM J. Numer. Anal.* 19 (1982) 400–408.

[7] J. Demmel, M. Gates, G. Henry, X. Li, J. Riedy and P. Tang, A proposal for a next-generation BLAS, 2017. preprint.

[8] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, no. 16 in Classics in Applied Mathematics, SIAM, Philadelphia, 1996.

[9] J.E. Dennis and H.F. Walker, Inaccuracy in quasi-Newton methods: Local improvement theorems, in: *Mathematical Programming Study 22: Mathematical programming at Oberwolfach II*, North–Holland, Amsterdam, 1984, pp. 70–85.

[10] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996,

[11] N.J. Higham and T. Mary, *A new approach to probabilistic rounding error analysis*, February, 27 2019. Presentation at SIAM conference on Computational Science and Engineering.

[12] N.J. Higham, S. Pranesh and M. Zounon, Squeezing a matrix into half precision, with an application to solving linear systems, *SIAM J. Sci. Comput.* 41 (2019) A2536–A2551.

[13] *IEEE Standard for Binary Floating Point Arithmetic, Std 754-1885*, 1985.

[14] IEEE Computer Society, *IEEE standard for floating-point arithmetic, IEEE Std 754–2019*, July 2019.

[15] C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, no. 16 in Frontiers in Applied Mathematics, SIAM, Philadelphia, 1995.

[16] C.T. Kelley, Numerical methods for nonlinear equations, *Acta Numer.* 27 (2018) 207–287,

[17] C.T. Kelley, Newton's method in mixed precision, *SIAM Rev.* 64 (2022) 191–211,

[18] C.T. Kelley, *Notebook for Solving Nonlinear Equations with Iterative Methods: Solvers and Examples in Julia*, https://github.com/ctkelley/NotebookSIAMFANL, 2022,`http://dx.doi.org/10.5281/zenodo.4284687`, `https://github.com/ctkelley/NotebookSIAMFANL`. IJulia Notebook.

[19] C.T. Kelley, *SIAMFANLEquations.jl*. https://github.com/ctkelley/SIAMFANLEquations.jl, 2022, `http://dx.doi.org/10.5281/zenodo.4284807`, `https://github.com/ctkelley/SIAMFANLEquations.jl`. Julia Package.

[20] C.T. Kelley, *Solving Nonlinear Equations with Iterative Methods: Solvers and Examples in Julia*, no. 20 in Fundamentals of Algorithms, SIAM, Philadelphia, 2022.

[21] M.L. Overton, *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia, 2001.

[22] J.H. Wilkinson, *Rounding Errors in Algebraic Processes*, Dover, New York, 1994.

Manuscript received 30 July 2023
revised 10 October 2023
accepted for publication 10 October 2023

C. T. Kelley
North Carolina State University
Department of Mathematics
Box 8205, Raleigh, NC 27695-8205, USA
E-mail address: Tim_Kelley@ncsu.edu