Mixed Precision Iterative Refinement: Solutions and Variations

1 Exercise 8: Mixed Precision Iterative Refinement (2 points)

The following iterative refinement is applied to a linear system $Ax = b$ with $\kappa(A) = 10^3$.

Algorithm 1 Mixed Precision Iterative Refinement

Compute the factorization $A = LU$ in precision u_f
Compute $x = U^{-1}L^{-1}b$ in precision u_f
for $i = 1$ to n_{iter} **do**
    Compute the residual $r = b - Ax$ in precision u_r
    Compute $d = U^{-1}L^{-1}r$ in precision u_f
    Compute $x = x + d$ in precision u
end for

With $u_f = u = u_r = \text{fp32}$ and $n_{\text{iter}} = 1$, the algorithm produces a computed solution $\tilde{x}$ achieving a forward error

$$\varepsilon_{\text{fwd}} = \frac{\|\tilde{x} - x\|}{\|x\|} \approx 10^{-4},$$

as indicated in the table below.

Finish completing the table with the order of magnitude of the values that ε_{fwd} would take if we change the parameters n_{iter}, u_f, u or u_r of the algorithm as indicated. No need to justify your answers.

As a reminder, the unit roundoffs of fp64, fp32, fp16, and bfloat16 arithmetics are $2^{-53} \approx 10^{-16}$, $2^{-24} \approx 10^{-7}$, $2^{-11} \approx 10^{-4}$ and $2^{-8} \approx 10^{-3}$, respectively.

Configuration	$n_{\text{iter}} = 1$	$n_{\text{iter}} = 10$
$u_f = u = u_r = \text{fp32}$	10^{-4}	
$u_f = u = \text{fp32}, u_r = \text{fp64}$		
$u_f = u_r = \text{fp32}, u = \text{fp64}$		
$u_f = \text{fp32}, u = u_r = \text{fp64}$		
$u = u_r = \text{fp32}, u_f = \text{fp64}$		
$u = u_r = \text{fp64}, u_f = \text{fp16}$		
$u = u_r = \text{fp64}, u_f = \text{bfloat16}$		

2 Solution

Background Theory

For iterative refinement with condition number $\kappa = 10^3$, the forward error after k iterations behaves approximately as:

$$\varepsilon_{\text{fwd}}^{(k)} \approx \max\left\{\kappa u_f, (\kappa \max\{u_f, u_r\})^{k+1}\right\}$$

Key insights:

- The factorization precision u_f sets a fundamental limit: $\varepsilon \geq \kappa u_f$
- The residual precision u_r determines convergence rate
- The working precision u affects updates but is less critical
- More iterations help only if u_r is significantly better than u_f

Analysis by Row

Row 1: $u_f = u = u_r =$ **fp32**

- $n_{\text{iter}} = 1$: Given as 10^{-4}
- $n_{\text{iter}} = 10$: 10^{-4} (no improvement; limited by $\kappa u_f = 10^3 \times 10^{-7} = 10^{-4}$)

Row 2: $u_f = u =$ **fp32**, $u_r =$ **fp64**

- Better residual precision allows improvement
- $n_{\text{iter}} = 1$: 10^{-7} to 10^{-8} (one refinement step with accurate residual)
- $n_{\text{iter}} = 10$: 10^{-4} (converges to limit κu_f)

Row 3: $u_f = u_r =$ **fp32**, $u =$ **fp64**

- Better update precision, but limited by residual computation
- $n_{\text{iter}} = 1$: 10^{-4} (residual is the bottleneck)
- $n_{\text{iter}} = 10$: 10^{-4} (no improvement possible)

Row 4: $u_f =$ **fp32**, $u = u_r =$ **fp64**

- Best configuration for refinement; accurate residual and updates
- $n_{\text{iter}} = 1$: 10^{-7} to 10^{-8}
- $n_{\text{iter}} = 10$: 10^{-4} (still limited by factorization precision)

Row 5: $u = u_r =$ **fp32**, $u_f =$ **fp64**

- Excellent factorization, but refinement can't leverage it well
- $n_{\text{iter}} = 1$: 10^{-4} to 10^{-5}

- $n_{\text{iter}} = 10$: 10^{-4} to 10^{-5} (limited by u_r)

Row 6: $u = u_r = \textbf{fp64}, u_f = \textbf{fp16}$

- Poor factorization precision dominates
- $n_{\text{iter}} = 1$: 10^{-1} (limited by $\kappa u_f = 10^3 \times 10^{-4} = 10^{-1}$)
- $n_{\text{iter}} = 10$: 10^{-1} (cannot overcome poor factorization)

Row 7: $u = u_r = \textbf{fp64}, u_f = \textbf{bfloat16}$

- Very poor factorization precision
- $n_{\text{iter}} = 1$: 1 or worse ($\kappa u_f = 10^3 \times 10^{-3} = 1$)
- $n_{\text{iter}} = 10$: 1 (completely dominated by factorization error)

Completed Table

Configuration	$n_{\text{iter}} = 1$	$n_{\text{iter}} = 10$
$u_f = u = u_r = \text{fp32}$	10^{-4}	10^{-4}
$u_f = u = \text{fp32}, u_r = \text{fp64}$	10^{-7}	10^{-4}
$u_f = u_r = \text{fp32}, u = \text{fp64}$	10^{-4}	10^{-4}
$u_f = \text{fp32}, u = u_r = \text{fp64}$	10^{-7}	10^{-4}
$u = u_r = \text{fp32}, u_f = \text{fp64}$	10^{-4}	10^{-4}
$u = u_r = \text{fp64}, u_f = \text{fp16}$	10^{-1}	10^{-1}
$u = u_r = \text{fp64}, u_f = \text{bfloat16}$	1	1

Key Takeaways

1. **Factorization precision is critical:** Error cannot be less than κu_f
2. **Residual precision enables refinement:** Need $u_r \ll u_f$ for improvement
3. **Working precision u is less critical:** As long as $u \leq u_r$, it doesn't limit much
4. **More iterations help only with good u_r:** Otherwise stuck at factorization limit
5. **Don't use fp16/bfloat16 for factorization:** With $\kappa = 10^3$, errors are too large

3 Variations

Variation 1: Different Condition Numbers

Problem: How would the results change if $\kappa(A) = 10^6$ instead of 10^3?

Solution:

With $\kappa = 10^6$, the fundamental limit becomes κu_f:

Configuration	$n_{\text{iter}} = 1$	$n_{\text{iter}} = 10$
$u_f = u = u_r = \text{fp32}$	10^{-1}	10^{-1}
$u_f = u = \text{fp32}, u_r = \text{fp64}$	10^{-4}	10^{-1}
$u_f = \text{fp32}, u = u_r = \text{fp64}$	10^{-4}	10^{-1}
$u = u_r = \text{fp32}, u_f = \text{fp64}$	10^{-1}	10^{-1}
$u = u_r = \text{fp64}, u_f = \text{fp64}$	10^{-10}	10^{-10}
$u = u_r = \text{fp64}, u_f = \text{fp16}$	10^{2}	10^{2}

Key observation: With larger κ, fp32 factorization becomes problematic. Need fp64 for factorization when $\kappa > 10^7$.

Variation 2: Optimal Precision Assignment

Problem: You have a limited computational budget and can use:

- fp64 for one component (factorization, residual, or working precision)
- fp32 for the other two components

Given $\kappa = 10^3$ and $n_{\text{iter}} = 10$, which component should use fp64 to minimize error?

Solution:

Test all three options:

1. **fp64 for factorization:** $u_f = \text{fp64}, u = u_r = \text{fp32}$
   - $\varepsilon \approx \kappa u_r = 10^3 \times 10^{-7} = 10^{-4}$
2. **fp64 for residual:** $u_r = \text{fp64}, u_f = u = \text{fp32}$
   - $\varepsilon \approx \kappa u_f = 10^3 \times 10^{-7} = 10^{-4}$
   - With many iterations, converges to this limit
3. **fp64 for working precision:** $u = \text{fp64}, u_f = u_r = \text{fp32}$
   - $\varepsilon \approx \kappa u_f = 10^3 \times 10^{-7} = 10^{-4}$

Answer: All three give similar results (10^{-4}) with $n_{\text{iter}} = 10$! However:

- **For few iterations:** Use fp64 for residual (u_r)
- **For many iterations:** Use fp64 for factorization (u_f) or residual (u_r)
- **Working precision (u) is least important**

Variation 3: Mixed Precision Matrix Multiplication

Problem: Suppose the residual computation $r = b - Ax$ uses:

- fp32 to compute Ax (storage precision)
- fp64 accumulation internally
- fp64 for final subtraction $b - Ax$

How does this compare to pure fp64 residual computation?
Solution:
Pure fp64 residual: $|r_{\text{computed}} - r_{\text{exact}}| \lesssim u_{64}\|A\|\|x\|$
Mixed precision residual:

- Loading A in fp32: introduces error $\approx u_{32}\|A\|$
- Accumulation in fp64: error $\approx u_{64}\|A\|\|x\|$
- Overall: $|r_{\text{computed}} - r_{\text{exact}}| \lesssim u_{32}\|A\|\|x\|$

Conclusion: Mixed precision is almost as good as pure fp64 if:

- Matrix A is stored in fp32 anyway (saves memory)
- Accumulation is done in fp64 (via FMA instructions)
- Effective $u_r \approx u_{32}$ for practical purposes

For $\kappa = 10^3$: This gives $\varepsilon \approx 10^{-4}$ (same as pure fp32 residual).

Variation 4: Convergence Analysis

Problem: Derive a more precise formula for the error after k iterations.
Solution:
Let $e^{(k)} = x - x^{(k)}$ be the error after k iterations. The iterative refinement satisfies:

$$e^{(k+1)} = e^{(k)} - \tilde{A}^{-1}\tilde{r}^{(k)}$$

where $\tilde{r}^{(k)} = \mathrm{fl}_{u_r}(b - Ax^{(k)})$ is the computed residual.
Standard error analysis gives:

$$\|e^{(k+1)}\| \leq \left[\kappa(A)\max\{u_f, u_r\} + O(u^2)\right]\|e^{(k)}\| + \kappa(A)u_f\|x\|$$

After k iterations:

$$\frac{\|e^{(k)}\|}{\|x\|} \leq C^k\frac{\|e^{(0)}\|}{\|x\|} + \frac{\kappa u_f}{1 - C}$$

where $C = \kappa\max\{u_f, u_r\}$.
Convergence conditions:

- If $C < 1$ (i.e., $\kappa\max\{u_f, u_r\} < 1$): Converges to κu_f
- If $C \geq 1$: No convergence; diverges or stagnates

For $\kappa = 10^3$:

- fp32 residual: $C = 10^3 \times 10^{-7} = 10^{-4} < 1$ (converges)
- fp16 factorization: $C = 10^3 \times 10^{-4} = 0.1$ (converges slowly)
- bfloat16 factorization: $C = 10^3 \times 10^{-3} = 1$ (marginal)

Variation 5: Practical Recommendations

Problem: Given a linear system with $\kappa(A)$ and computational constraints, recommend a mixed precision strategy.

Solution:

Condition Number	**Recommended Strategy**
$\kappa < 10^3$	fp32 everywhere, $n_{\text{iter}} = 1$
$10^3 < \kappa < 10^7$	fp32 factorization (u_f), fp64 residual (u_r), fp32 or fp64 working (u), $n_{\text{iter}} = 3$–5
$10^7 < \kappa < 10^{13}$	fp64 factorization (u_f), fp64 residual (u_r), fp64 working (u), $n_{\text{iter}} =$ 1–2
$\kappa > 10^{13}$	fp64 everywhere; consider extended precision or regularization

Cost-benefit analysis:

- **Factorization:** $O(n^3)$ operations, done once
  - Use fp64 if $\kappa > 10^7$ and you can afford it
  - Otherwise fp32 is usually sufficient
- **Residual:** $O(n^2)$ per iteration
  - fp64 gives best refinement rate
  - But if factorization is fp32, benefit is limited
- **Triangular solves:** $O(n^2)$ per iteration
  - Match factorization precision
- **Updates:** $O(n)$ per iteration
  - Least critical; fp32 usually fine

Variation 6: Effect of Matrix Structure

Problem: How does the analysis change for:

1. Symmetric positive definite (SPD) matrices using Cholesky
2. Sparse matrices
3. Structured matrices (e.g., banded, Toeplitz)

Solution:

1. SPD matrices with Cholesky factorization $A = LL^T$:

- Condition number: $\kappa(L) = \sqrt{\kappa(A)}$
- More stable than LU factorization

- Can tolerate slightly lower precision for factorization
- For $\kappa(A) = 10^6$: $\kappa(L) = 10^3$, so fp32 Cholesky works well

2. Sparse matrices:

- Factorization is $O(n \cdot \text{nnz})$ instead of $O(n^3)$
- Residual computation much cheaper: $O(\text{nnz})$
- Can afford fp64 for everything without much cost
- Mixed precision less beneficial unless matrix is huge

3. Structured matrices:

- Specialized fast algorithms available
- May not need explicit factorization
- Residual computation remains cheap
- Mixed precision strategy depends on specific structure